M_CSPF: A Scalable CSPF Routing Scheme with Multiple QoS Constraints for MPLS Traffic Engineering

Daniel W. Hong, Choong Seon Hong, and Gil-Haeng Lee

In the context of multi-protocol label switching (MPLS) traffic engineering, this paper proposes a scalable constraint-based shortest path first (CSPF) routing algorithm with multiple QoS metrics. This algorithm, called the multiple constraint-based shortest path first (M_CSPF) algorithm, provides an optimal route for setting up a label switched path (LSP) that meets bandwidth and end-to-end delay constraints. In order to maximize the LSP accommodation probability, we propose a link weight computation algorithm to assign the link weight while taking into account the future traffic load and link interference and adopting the concept of a critical link from the minimum interference routing algorithm. In addition, we propose a bounded order assignment algorithm (BOAA) that assigns the appropriate order to the node and link, taking into account the delay constraint and hop count. In particular, BOAA is designed to achieve fast LSP route computation by pruning any portion of the network topology that exceeds the end-to-end delay constraint in the process of traversing the network topology. To clarify the M_CSPF and the existing CSPF routing algorithms, this paper evaluates them from the perspectives of network resource utilization efficiency, end-to-end quality, LSP rejection probability, and LSP route computation performance under various network topologies and conditions.

Keywords: Packet scheduler, flow aggregation, rate control, IntServ, DiffServ, QoS.

Manuscript received Aug. 26, 2004, revised July 11, 2005.

This work was supported by MIC and ITRC Project.

Daniel W. Hong (phone: +82 42 866 3821, email: wkhong@kt.co.kr) is with the Operation Support System Lab., KT, Daejeon, Korea. Choong Seon Hong (phone: +82 31 201 2532, email: cshong@khu.ac.kr) is with the Department of Computer Engineering, Kyung Hee University, Seoul, Korea.

Gil-Haeng Lee (email: ghlee@etri.re.kr) is with the Broadband Convergence Network Research Division, ETRI, Daejeon, Korea.

I. Introduction

Multi-protocol label switching (MPLS) is an Internet Engineering Task Force (IETF)-defined protocol that overcomes some of the shortcomings of IP-based networks. MPLS is meant for service provider core networks or large enterprise networks. There are two major benefits of MPLS traffic engineering: better total network use efficiency and better end-to-end quality for each traffic flow. To achieve these objectives, MPLS supports two traffic-engineering protocols: resource reservation protocol-traffic engineering (RSVP-TE) [1]-[4] and constraint-based label distribution protocol (CR-LDP) [1], [3], [5]-[7]. However, these protocols can manage only the bandwidth resource, the routing protocol is not defined in these protocols. In designing a network where traffic engineering is performed, the selection of routing algorithms and network topologies is believed to exert a great influence on the end-to-end quality and overall network resource utilization.

In the case of constant bit rate (CBR), traffic demands are placed dynamically, often on a first-come-first-serve basis. Routes are calculated one by one using the appropriate algorithm; for example, constraint-based shortest path first (CSPF) [8], shortest-distance path (SDP) [3], widest-shortest path (WSP) [3], or shortest-widest path (SWP) [9] algorithms as individual routes are computed meeting the quality-of-service (QoS) constraints.

On the other hand, fine-grained traffic engineering is also important. In order to achieve this, we generally consider multiple QoS metrics such as bandwidth, delay, jitter, administrative weight, and others, although this traffic engineering causes the NP-complete problem.

In addition, the minimum interference routing algorithm

(MIRA) proposed in [10] explicitly takes into account the location of the ingress and egress routers. The key idea of MIRA is to route an incoming connection over the path that least interferes with possible future requests. However, this complexity of MIRA is too great to apply in a real operational environment. Also, it does not consider the future traffic load and only meets the bandwidth constraint.

Thus, this paper proposes a scalable constraint-based shortest path first (CSPF) routing scheme, called the multiple CSPF (M_CSPF) routing algorithm, with the bandwidth and end-to-end delay QoS constraints. M_CSPF is designed to support the on-line setup of an optimal label switched path (LSP) in an MPLS network, taking into account better efficiency in terms of network resource utilization and better end-to-end quality. It consists of two steps: the first is assigning the appropriate weight to the network topology, and the second is finding the optimal shortest path.

In order to assign the appropriate weight to the network topology while taking into account the total traffic load and link interference, we propose a link weight computation algorithm (LWCA) that, for the first time, introduces the future traffic load and adopts the concept of a critical link from MIRA [10]. Future traffic load can be defined as the anticipating LSP requests between arbitrary source and destination pair. In terms of network planning, a network service provider adjusts the network capacity according to the future traffic estimation, which takes a long time. However, in terms of MPLS traffic engineering, we should consider maximization of network resource utilization under the current network capacity. Therefore, we must consider the future traffic arrival or the future LSP requests to maximize the LSP accommodation ratio. MIRA only takes into account the critical link to identify the link interference, but our approach considers both the critical link and the future traffic load. By adding the future traffic load concept, we can achieve a more enhanced LSP accommodation ratio.

In addition, we propose a bounded order assignment algorithm (BOAA) that allocates the proper orders to the node and link considering the delay based on the weighted network graph created by LWCA. The ordered network graph created by BOAA is used to find the optimal LSP route that conforms to the bandwidth and end-to-end delay constraints. BOAA is designed to enhance the LSP route computation performance by pruning some portions of the weighted network graph that exceed the requested end-to-end delay constraint in the process of the order assignment process.

Because the major objective of MPLS traffic engineering is enhancing the efficiency of network resource utilization and end-to-end quality, we measured the efficiency of network resource utilization and the end-to-end quality of the proposed algorithm and other algorithms, WSP, SWP, SDP, and MIRA, under three different network topologies: flat tree, hierarchical ring, and torus. In addition, we also measured the LSP setup rejection probability of our algorithm and MIRA under an unbalanced topology and a balanced network topology with a large number of critical links. We also compared the LSP computation performance of our algorithm and MIRA by gradually adding four nodes and by increasing the value of the end-to-end delay constraint under the torus network topology.

This paper is organized as follows: Section II describes the existing routing algorithms and QoS constraints. Section III describes the M_CSPF composed of two subsequent steps, which are weight decision and optimal route selection that takes into account the two additive metrics of bandwidth and end-to-end delay. Section IV describes the performance evaluation results of the existing algorithms and the M_CSPF routing algorithm under the target network topologies described in section II. Finally, we summarize our work and discuss some future research directions.

II. The Existing Algorithms for MPLS Traffic Engineering

There are some QoS routing algorithms [4], [11]-[16] that define the framework and techniques for QoS routing in the Internet, and focus on the selection and maintenance of packet-forwarding paths capable of meeting specific service class objectives [17]. These QoS routing algorithms compute the routing table using the parameter of unreserved bandwidth as the QoS constraint [18], and each LSP reserves the bandwidth exclusively in links through which the LSP passes. These routing algorithms accommodate successive LSPs in consideration of the unreserved bandwidth. These QoS routing algorithms can be installed with easy modification, and they are chosen by many. We consider the following CBR algorithms for verifying the efficiency and end-to-end quality of M_CSPF. CBR algorithms apply the extended interior gateway protocol (IGP) parameters to the tree to find a suitable path. Normally, the available bandwidth and hop count may be used to determine paths using the three algorithms discussed below.

Computing optimal routes subject to two or more constraints is an NP-complete problem. Mostly, the algorithms work on available bandwidth and hop count for selecting a path between a source and a destination. A constraint-based routing scheme can choose one of the following as the route for a destination with some tradeoffs between resource conservation and load balancing.

- *The Widest-shortest path* (*WSP*) [2] selects the shortest

feasible path. If there are several feasible paths, the one having the largest residual bandwidth is chosen. The WSP is an improvement of the min-hop algorithm (MHA), as it attempts to load-balance the network traffic. In fact, WSP selects a feasible path with minimum hop counts, and if there are several such paths, it selects the one with the largest residual bandwidth, thus discouraging the use of already heavily loaded links. However, WSP still has the same drawbacks as MHA since the path selection is performed among the shortest feasible paths, which are used until saturation before switching to other feasible paths. This algorithm applies Dijkstra's algorithm after line, in which the unreserved bandwidth that is less than the demand bandwidth of LSP has been trimmed.

- *The Shortest-widest path* (*SWP*) [8] selects the route in which the minimum unreserved bandwidth of the links along the route is largest among all the routes and whose unreserved bandwidth exceeds the required bandwidth of the LSP. It selects the path with the largest feasible bandwidth. If there are several feasible paths, the one with the minimum hop count is selected.
- *The Shortest-distance path* (*SDP*) [3] selects the path with the shortest distance. The distance can be defined as the sum of the inverse bandwidths of all links along the path. The SDP routes an incoming connection along the path that reaches the destination node using the minimum number of feasible links. The distance for Dijkstra's method is defined as the reciprocal of the unreserved bandwidth of the link. Then, this algorithm applies Dijkstra's method: $\min\left(\text{distance} = \sum_{j=1}^{k} 1/R_{ij}\right)$, where R_i is the bandwidth available on link i_j.

The shortest-distance approach favors the shortest paths when the network load is heavy and favors the widest paths when the network load is moderate. However, this scheme does not differentiate between the various classes of traffic, as its only measure of cost is the available bandwidth. The widest-shortest path approach can minimize bandwidth fragmentation.

On the other hand, MIRA, which is proposed in [10], explicitly takes into account the location of the ingress and egress routers. The key idea of MIRA is to route an incoming connection over a path that least interferes with possible future requests.

Specifically, an incoming connection request between (S_i, T_i) is routed with the goal of maximizing an objective function, which is either the minimum-maximum flow (maxflow) of all other ingress-egress pairs or a weighted sum of maxflows, where weights α_{ST} assigned to each *S*-*T* pair reflect the "importance" of the flow.

In order to achieve an on-line routing algorithm, MIRA keeps an updated list of the critical links, that is, the links whose use by the incoming call diminishes the maxflow between other pairs.

When a new call has to be routed between the source/destination pair S_i, T_i, MIRA determines the set L_{ST} of the critical links for all the source/destination pairs S_j, T_j other than S_i, T_i. The weight w of each link l is then set according to the equation $w(l) = \sum_{(S,T):l \in L_{ST}} \alpha_{ST}$, and the route that causes the minimum interference to other source/destination pairs is selected.

In spite of its more sophisticated functions, MIRA still has the following limitations whose effects will be shown in the discussion of numerical results:

- MIRA discourages the use of critical links based only on the number of other *S*-*T* pairs that could use them, without verifying if these *S*-*T* pairs actually use these links. Evidently, if one of these other *S*-*T* pairs introduces a low traffic in the network, the criticality of the links that diminish its maxflow is far less important than that of *S*-*T* pairs that produce a large amount of traffic. As a consequence, MIRA preserves the use of certain links that remain underutilized, thus causing a sub-optimal use of the network. To overcome this limitation, maximizing the weighted sum of the source/destination maxflows has been proposed. However, in [19], the weights are chosen offline and do not adapt to changes in the network traffic. Hence, this solution does not provide the flexibility required of an on-line routing scheme.
- In its on-line implementation, MIRA sets the link weights almost in a static way according only to their level of criticality. In fact, the only event that can cause the re-distribution of new weights is the saturation of some links, which is similar to the min-hop algorithm.
- While choosing a path for an incoming request, MIRA does not take into account how the new call will affect the future requests of the same ingress/egress pair (auto-interference).

III. A Multiple Constraint-Shortest Path First Routing Algorithm (M_CSPF)

In this section, we describe a scaleable CSPF routing algorithm that can provide an optimal route that meets the two additive metrics of bandwidth and end-to-end delay. We assume that we can identify the future traffic arrival between all source and destination pairs and that there is one LSP setup request at a time. The overall M_CSPF scheme is composed of the following three steps:

Step 1. Timing the unfavorable network topology—we prune the links when the available bandwidth (B_{ava}) is less than the

requested bandwidth (B_{req}) and when the number of fault occurrences is larger than the designated threshold assigned by the network administrator.

Step 2. Assign appropriate weight to links or nodes—we assign the appropriate weight to links or nodes. In this paper, we propose an algorithm for the computation of link weight that takes high network resource utilization and high computation performance into account under the estimated future traffic load and current link delay.

Step 3. Compute the optimal QoS route conforming to the bandwidth and end-to-end delay taking into account the current network status.

The QoS constraint parameters of an LSP can be specified in terms of minimum guaranteed bandwidth and maximum tolerable delay and/or jitter. The main goal of a QoS routing technique is to determine the path that can guarantee the constraints requested by the incoming packets and reject as few LSP requests as possible.

Let us model a network as a graph, G(N,E), where node N represents the label switch router (LSR) or label edge router (LER), and edge E represents the communication links, as shown in Fig. 1.

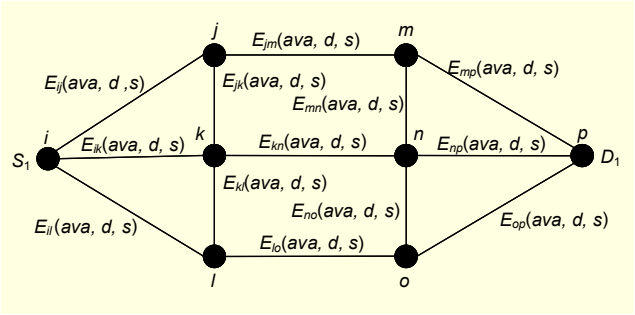

Fig. 1. An MPLS network model for supporting QoS guaranteed LSP.

The traffic enters the network at ingress node S_i and exits at egress node D_i. Each LSP requires a path from S_i to D_i. Each edge E_{ij} has some associated parameters such as available bandwidth ($E_{ij}(ava)$), delay ($E_{ij}(d)$), and link status ($E_{ij}(s)$), which indicate the fault, congestion, or performance degradation.

A new LSP can be routed over links with $E_{ij}(ava)$ greater or equal to the requested bandwidth (R_b). In addition, the link feasibility ($E_{ij}(F)$) can be defined as

$$E_{ij}(F)=\begin{cases}\text{YES}, & \text{if}(E_{ij}(ava) > R_b \text{ and} \\ & E_{ij}(s)\notin\{\text{fault}\,|\,\text{congestion} \\ & \quad |\,\text{performance degradation}\}) \\ \text{NO}, & \text{Otherwise}\end{cases}. \quad (1)$$

where a link can be feasible if the requested bandwidth (R_b) is less than or equal to the available bandwidth ($E_{ij}(ava)$), and the link status ($E_{ij}(s)$) is not in fault, has congestion, or has performance degradation. Otherwise, we trim the link ($E_{ij}(F)$==NO) from $G(N,E)$ because it is not feasible.

1. A Link Weight Computation Algorithm (LWCA)

Once the network topology has been pruned according to the rule of (1), we determine the link weight ($E_{ij}(w)$). In this section, we describe the link weight computation algorithm (LWCA) that determines each $E_{ij}(w)$, taking into account the optimal LSP provision with bandwidth and delay constraints and the future traffic arrival at every possible ingress LSR ($\forall S_i$).

Here, we borrowed the concepts of a critical link from MIRA [10] to solve the problem of future traffic arrival. However, LWCA is designed to solve the existing limitations of MIRA and to enhance the overall performance of optimal path computation, taking into account the multiple QoS metrics of end-to-end delay and bandwidth.

Let us consider a uni-directional network topology with unbalanced future traffic loads, as shown in Fig. 2. There are three ingress/egress pairs having different future traffic loads. In this example, we assume that there are heavier future traffic loads from ingress S_1 to egress D_1 than in the others (S_2-D_2 and S_3-D_3).

There is only one possible route <*a-b-c-d*> between S_1 and D_1. Also, there is only one possible route <*g-h-i-j-k*> between S_3 and D_3. However, there are two different possible routes of <*e-b-c-f*> and <*e-h-i-j-f*> between S_2 and D_2.

In order to calculate the link critical cost $E_{ij}(cc)$ at E_{ij}, we add two parameters to edge (E) of the network model described in Fig. 1. One is the total traffic load ($E_{ij}(load)$), which represents the possible traffic loads at E_{ij} to accommodate the anticipated traffic load at each ingress LSR. The other parameter is the link weight ($E_{ij}(w)$) parameter, which maintains the link weight that is computed with our LWCA, taking various aspects into account.

In addition, we define two additional parameters for computing $E_{ij}(w)$. We define the anticipated traffic load at ingress LSR S_i as $T_{load}(S_i)$. Referring to Fig. 2, the $T_{load}(S_1)$ is

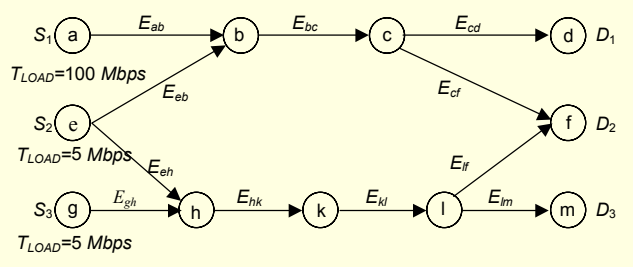

Fig. 2. A uni-directional network topology with unbalanced future traffic loads.

Table 1. Link weight table taking into account the critical link cost, traffic load, and delay.

Attributes \ Links	E_{ab}	E_{bc}	E_{cd}	E_{eb}	E_{cf}	E_{eh}	E_{jf}	E_{gh}	E_{hk}	E_{kl}	E_{lm}
E_{ij} (d) (ms)	2	1	1	1	2	2	1	1	1	2	1
E_{ij} (ava) ($Mbps$)	500	500	500	500	500	500	500	500	500	500	500
$P_{candidate}$ (S_1, D_1) T_{load} (S_1)=100 $Mbps$	◎	◎	◎								
$P_{candidate}$ (S_2, D_2)		◎		◎	◎						
T_{load} (S_2)=5 $Mbps$						◎	◎		◎	◎	
$P_{candidate}$ (S_3, D_3) T_{load} (S_3)=5 $Mbps$								◎	◎	◎	◎
E_{ij} (cc)	1	2	1	1	1	1	1	1	2	2	1
E_{ij} ($load$)	100	105	100	5	5	5	5	5	10	10	5
E_{ij} (w)	2000.098	1000.205	1000.098	1000.005	2000.005	2000.005	1000.005	1000.005	1000.020	2000.020	1000.005

100 Mbps. We define all the possible candidate routes between S_i and D_i as $P_{candidate}(S_i,D_i)$. For example, the $P_{candidate}(S_2,D_2)$ under the network topology shown in Fig. 2 can be <*e-b-c-f*> and <*e-h-i-j-f*>. The pseudo-code for LWCA is described in Fig. 3, and the example of link weight computation under the network topology in Fig. 2 is shown in Table 1. At first, we initialize the link critical cost ($E_{ij}(cc)$) for $\forall E_{ij}$ as zero. Next, we determine all possible routes between all possible ingress and egress pairs using the Dijkstra algorithm. In the case of Fig. 2, there are three possible ingress/egress pairs (<*S1,D1*>, <*S2,D2*>, and <*S3,D3*>). There are four possible routes as shown in Table 1.

On finding all possible routes between every ingress and egress pairs, we determine the link interference by counting the number of appearances of each E_{ij} along all the possible routes and set the number of appearances of each E_{ij} to $E_{ij}(cc)$. For example, in Table 1, $E_{bc}(cc)$ can be 2 because E_{bc} appeared twice, once for $P_{candidate}(a,d)$ and once for $P_{candidate}(e,f)$. If $E_{ij}(cc)$ is one, it means that there is no interference. The effect of interference is proportional to the number of $E_{ij}(cc)$.

Next, we compute the $E_{ij}(load)$ for each E_{ij} using the following equation:

$$E_{ij}(load) = \sum (T_{load}(S_i),\ if\ E_{ij} \in P_{candidate}(S_i, D_i)) . \quad (2)$$

As seen in Table 1, $E_{bc}(load)$ is 105 Mbps, where 100 Mbps is for $P_{candidate}(S_1,D_1)$ and 5 Mbps is for $P_{candidate}(S_2,D_2)$.

With such information as $P_{candidate}(S_i,D_i)$, $E_{ij}(cc)$, $E_{ij}(load\ d)$, $E_{ij}(load)$, and $E_{ij}(s)$, we compute link weight ($E_{ij}(w)$) according to (2). In computing $E_{ij}(w)$, we combine two aspects. The first is the future traffic arrival and link interference, which is computed by $(\frac{E_{ij}(load)}{1024} \times E_{ij}(cc))$, where we divide the total load expressed in Mbps at E_{ij} by 1024 strands for kbps and multiply the link critical cost. The other is the delay, which is computed as ($E_{ij}(d)$×1,000), where 1,000 is a random large number that distinguishes the delay from the traffic arrival and link interference.

Algorithm for LWCA:

1. initialize all *as* 0;
2. for ($\forall S_i$-D_i pairs)
3. compute $P_{candidate}$ (S_i, D_i) using *Dijkstra* algorithm;
4. end for
5. count the appearance of E_{ij} along all $P_{candidate}$ (S_i, D_i) and
6. set the number of appearance of E_{ij} to E_{ij} (cc);
7. compute E_{ij} (load) for $\forall E$;
8. for ($\forall E_{ij}$)
9. compute E_{ij} (w) taking into account E_{ij} ($load$), E_{ij} (s),
10. E_{ij} (d) with following the rule of (3)
11. end for

Fig. 3. Pseudo-code for link weight computation algorithm (*LWCA*).

For example, the link weight in terms of the traffic load and link interference at E_{bc} in Table 1 can be 0.205 ($E_{bc}(load)$=105/1024×($E_{bc}(cc)$=2)), and the link weight in terms of the delay at E_{bc} can be 1000 (=($E_{bc}(d)$=*1*)×1,000) because the delay at $E_{bc}(d)$ is one. Therefore, the $E_{bc}(w)$ can be 1000.205 (=0.205+1,000).

We maintain the computed link weight information together with the network topology in the traffic-engineered database described in Fig. 1. We assume that $T_{load}(i,j)$ is changed once every two or three months, that is to say, it shows a nearly static nature in terms of an on-line routing algorithm. In addition, the value of $E_{ij}(cc)$ is not changed if there is no change in the

network topology that subsequently affects the change in $P_{candidate}(S_i,D_i)$. Therefore, we need to compute the link weight ($E_{ij}(w)$) only in the case of a network topology change, which is one of the major differences from the others [3], [10]-[13], [20], and [21].

In the case of MIRA [10], it needs to identify the critical link to minimize the interference during every LSP setup request, whereas our scheme does not. Therefore, our scheme can greatly enhance the route computation time, which will be proven with the performance evaluation in section V.

2. An Algorithm for Optimal Route Selection

After computing the link weight for all links ($\forall E_{ij}(w) \in G(N,E)$), we need to find an optimal route to setup the MPLS LSP that conforms to multiple QoS metrics such as bandwidth, end-to-end delay, and hop count. In this section, we propose an algorithm to provide the optimal route with multiple metrics based on the predetermined link weight. The optimal route selection procedure is composed of two steps:

Step 1. Trim the unfavorable links or nodes and assign appropriate orders to each node and link that will be used to find the optimal route in the next step.

Step 2. Select the optimal route that meets the multiple QoS parameters such as bandwidth and end-to-end delay.

We define the LSP setup request as $P(R_b,R_d,S,D)$, where R_b is the requested bandwidth, R_d is the requested end-to-end delay, S is the ingress node, and D is the egress node. For the first step of our route provision procedure, we would like to describe the method for assigning orders to each node and link along the weighted network graph that was composed by the LWCA. The order assignment procedure is composed of two steps: the first is trimming the unfavorable links ($E_{ij}(w)==\infty$ or $E_{ij}(ava)< R_b$) from the weighted network graph $G(N,E_w)$, and the second is assigning appropriate orders to each node ($N(o)$) and edge ($E(o)$) using the following algorithm, which is called the bounded order assignment algorithm (BOAA).

BOAA is designed to minimize the complexity of order assignment by limiting the traversing of the network topology when the accumulated delay is exceeded by the requested delay bound (R_d). Intrinsically, the end-to-end delay metric is a sort of path metric, which means that we can determine the end-to-end delay after creating the route between an ingress node and an egress node. However, our approach can determine the violation possibility of the end-to-end delay at the stage of traversing the network topology instead of the completion of route computation.

Figure 4 shows the pseudo-code for BOAA. In the order

Algorithm for BOAA

```
1.  for all N(v, o) ← (no, ∞);
2.  for all E(o) ← ∞;
3.  N_active ← P(D) and N_dest ← P(S);
4.  N_active (v, o) ← (yes, 0);
5.  H_factor ← 1,000,000 and D_factor ←1,000;

6.  Procedure BOAA (N_active, N_dest)
7.      if(Quotiant(N_active)(o)/D_factor)≥R_d && N_active≠N_dest)
8.          N_active(v) ← yes and return;
9.      if(N_active == N_dest) return;
10.     N_active (v) ← yes;
11.     for each E connected to N_active do
12.         N_passive ← E_adj;
13.         if(E(w) == ∞)
14.             E(v,o) ← (yes, ∞);
15.         else if (E(o)> (N_active(o) + H_factor))
16.             E(v,o) ← (yes, N_active(o)+H_factor);
17.         if(N_passive (o)> E(o)) {
18.             N_passive(o) ← E(o);
19.             BOAA(N_passive, N_dest);
20.         }
21.     end of for
22. end of Procedure
```

Fig. 4. Pseudo-code for BOAA.

assignment to the node and link, the node has two attributes of order and visiting flag ($N(o,v)$), where $N(o)$ is the order and $N(v)$ indicates whether N was visited or not. On the other hand, each link maintains three attributes $E(w,d,o)$, where $E(w)$ is the link weight assigned by LWCA, $E(d)$ is the delay at the link, and $E(o)$ is the order. In order to assign the order while taking into account the shortest path constraint, we also define one hop as a random large number of 1,000,000 (H_{factor}=1,000,000), which is multiplied by a random number taking into account the delay (D_{factor}=1,000) as described in LWCA by 1,000 (= 1,000 (for delay) × 1,000 (for hop count)).

BOAA determines the orders of each node and link, taking into account the combination of the link weight assigned by LWCA and hop count. However, we do not care about the available bandwidth in the order assignment process.

There are two kinds of nodes: one is the active node (N_{active}), and the other is the passive node ($N_{passive}$). Node N_{active} represents the active node that allocates the proper weights to all of the neighboring nodes and links connected to it. Node $N_{passive}$ represents the passive node order that is assigned by an active node, that is to say, $N_{passive}$ is a neighboring node of N_{active}.

Figure 5 shows the weighted network graph that was created by LWCA, as described in the previous section. In order to assign the appropriate order to the node and link, we need

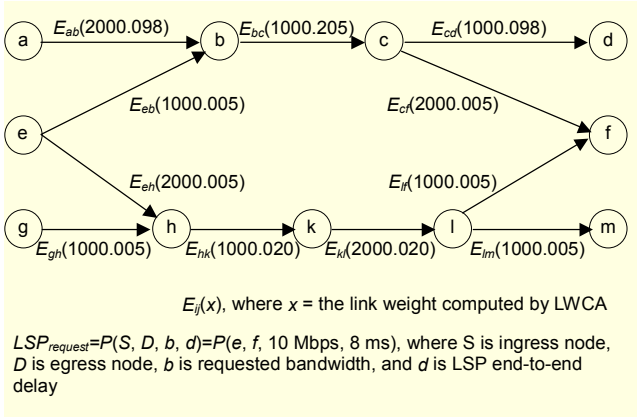

Fig. 5. An example of the weighted network graph created by LWCA.

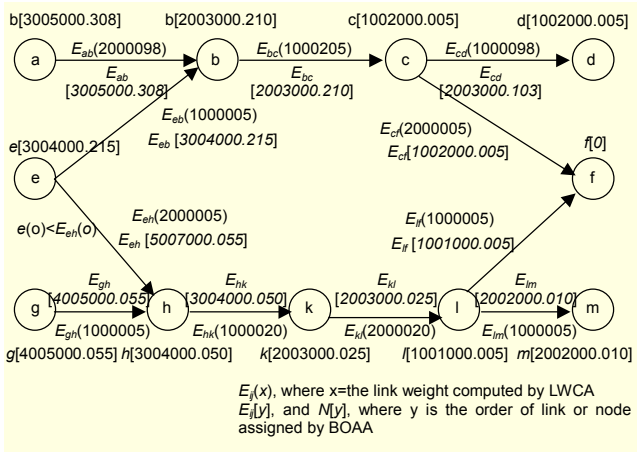

Fig. 6. The order assigned network graph by BOAA with the weighted network graph shown in Fig. 5.

the LSP setup information and weighted network graph as shown in Fig. 5. Also, Fig. 6 shows the result of order assignment using BOAA under the weighted network graph of Fig. 5. In addition, a path setup request is composed of the ingress node, egress node, requested bandwidth, and end-to-end delay constraint. In this example, we assume that the ingress node is e, the egress node is f, the requested bandwidth is 10 Mbps, and the end-to-end delay constraint is 10 ms, as shown in Fig. 5.

We traverse the weighted network graph from the egress node ($P(D)$) to ingress node ($P(S)$). Therefore, the first N_{active} can be ($P(D)$) as shown in line 3 in Fig. 4. Initially, we assign ($yes,0$) to $N_{active}(v,o)$, and we set H_{factor} to 1,000,000 and D_{factor} to 1,000. Henceforth, we traverse the weighted network graph until all nodes are traversed ($\forall N(v)==yes$). When a node is N_{active}, the adjacent node connected to the N_{active} with an edge (E) can be $N_{passive}$.

We determine the order of link ($E(o)$) connected to the N_{active} using the following rule:

$$E(o)=\begin{cases} E(o), & \text{if}(E(w)\neq\infty \text{ and } (E(o)\leq N_{active}(o)+H_{factor})); \\ N_{active}(o)+H_{factor}, & \text{if}(E(w)\neq\infty \text{ and } (E(o)>N_{active}(o)+H_{factor})); \\ \infty, & \text{if}(E(w)==\infty); \end{cases} \quad (3)$$

The order of the link to which ($E(o)$) is connected is determined by the order of $N_{active}(N_{active}(o))$ and the weight of the link ($E(w)$). If the weight of the link ($E(w)$) connected to N_{active} is infinite (∞), $E(o)$ can be ∞ regardless of the order of $N_{active}(o)$. If $E(w)$ is not infinite and $E(o)$ is less than or equal to $N_{active}(o)$ plus one, $E(o)$ will not be changed. However, if $E(w)$ is not infinite and $E(o)$ is greater than $N_{active}(o)$ plus one, $E(o)$ will be replaced with the value of $N_{active}(o)$ plus one.

On the other hand, we need to determine the order of each node ($N(o)$). If the visiting flag of $N_{passive}(v)$ is yes, then we traverse another link that is not yet traversed. If $N_{passive}(v)$ is no, we assign the order of $N_{passive}$ using the following rule:

$$N_{passive}(o)=\begin{cases} E(o), & \text{if}(N_{passive}(o)>E(o)); \\ N_{passive}(o), & \text{otherwise}. \end{cases} \quad (4)$$

The rule for assigning orders to the passive node is simpler than the order assignment for each link. If the order of passive node ($N_{passive}(o)$) connected with a link is greater than that of the link ($E(o)$), $N_{passive}(o)$ is replaced with $E(o)$. If $N_{passive}(o)$ is less than or equal to $E(o)$, we do not change $N_{passive}(o)$. For example, if node h is an active node, the order of passive node $e(o)$ is determined by the value of $E_{eh}(o)$ in Fig. 6. Because $e(o)$ is less than $E_{eh}(o)$, $e(o)$ is not changed.

When we traverse the network graph to assign orders to both the node and link, we evaluate the feasibility of the active node ($N_{active}(f)$ and whether it violates the constraint of the end-to-end delay at each N_{active} using the following rule:

$$N_{active}(f)=\begin{cases} unfeasible, & \text{if}\left(Quotiant(\frac{N_{active}(o)}{D_{factor}})\geq R_d\right); \\ feasible, & \text{otherwise}. \end{cases} \quad (5)$$

If $N_{active}(f)$ is not feasible, then we do not traverse its branch any more, which results in a better performance in optimal route selection conforming to the end-to-end delay requirement (R_d).

On finishing the order assignment for all the nodes and links, we select the most optimal route that conforms to the multiple constraints of bandwidth and delay with the order assigned network graph. In the process of route selection, we adjust the available bandwidth of each link along the selected path.

We traverse the order assigned network graph from the ingress node until we reach the egress node by reducing the available bandwidth of the selected link ($E_{ij}(ava)$-R_b).

Starting from the ingress node, we select a link that meets the

following rule until the egress node is reached:

$$\min \sum_{i=1}^{n} \left(\mathrm{mod} \left(\frac{\mathrm{mod}\left(\frac{E_i(o)}{H_{factor}} \right)}{D_{factor}} \right) \right), \quad (6)$$

where n is the number of links connected to a node.

For example, in the case of ingress node e, there are two candidate links of E_{eb}[3004000.215] and E_{eh}[5007000.055], as shown in Fig. 6. If we apply (5) to these two links, E_{eh} has less residual value than E_{eb}. Therefore, we select E_{eh} as an optimal link in terms of node e. Upon selecting an optimal link, we change the available bandwidth of the selected link using the equation $E_{ij}(ava)=E_{ij}(ava)-R_b$, where R_b is the requested bandwidth.

If there are two or more links that have the same value after applying the equation in terms of any node, we select the minimum hop route by applying the following rule:

$$\min \sum_{i=1}^{n} \left(Quotiant\left(\frac{E_i(o)}{H_{factor}} \right) \right), \quad (7)$$

where n is the number of links having the same value that is computed by (6).

If there are two or more links that meet the requested end-to-end delay bound, we select the route with the minimum hop count because the longer hop route consumes more bandwidth than the shorter hop route.

However, if there are two or more links that have the same hop count and the same values derived by (6) and (7), then we select the link that meets the following rule:

$$\min \sum_{i=1}^{n} \left(Quotiant \left(\frac{\mathrm{mod}\left(\frac{E_i(o)}{H_{factor}} \right)}{D_{factor}} \right) \right), \quad (8)$$

where n is the number of links having the same values that were computed by (6) and (7).

Therefore, if there are two or more links that have the same hop count and the same values derived by (6) and (7), then we select the link that has the least end-to-end delay.

As a result of applying the rules provided above, the selected route between ingress node e and egress node f that meets the requested bandwidth of 10 Mbps and the end-to-end delay of 10 ms can be <*e-h-k-l-f*> under the sample network topology of Fig. 6.

The main disadvantages of MIRA are its computation complexity and lack of consideration for the total traffic load offered to the network. The route selected by MIRA follows the minimum number of critical links (<*e-b-c-f*> in the example in Fig. 2 because the links E_{bc}, E_{hk}, and E_{kl} are critical links for the (S_2,D_2) node pair. On the other hand, MIRA traverses the network topology several times based on every ingress/egress pair to determine the critical links, which causes computation complexity.

However, our algorithm is somewhat different from MIRA in terms of two aspects. Our algorithm is a scalable critical link identification scheme that is a table-driven approach to reduce the computation complexity, which we need to compute all possible paths ($P_{candidate}(S_i,D_i)$) among the candidate ingress and egress pairs ($\sum_{i=1}^{n}(S_i, D_i)$) only when the network topology is changed. However, MIRA's approach computes and identifies the critical link every time a new LSP setup request arrives.

Another difference is that MIRA does not take into account the unbalanced future traffic loads between every ingress and egress pairs, but our algorithm does. For example, let us assume that the critical links under the network topology in Fig. 2 are E_{bc}, E_{hk}, and E_{kl} and that the ingress node is e and the egress node is f. In MIRA's case, it selects the route that has the least number of critical links. Therefore, MIRA selects route <*e-b-c-f*> as the optimal one. On the other hand, our algorithm selects the route that has the least weight, reflecting four additional aspects such as unbalanced future traffic load, critical link, and end-to-end delay. Of course, our algorithm and MIRA both take into account the requested bandwidth. Therefore, under the same network topology, our algorithm will select route <*e-h-k-i-f*> as the optimal one.

IV. Performance Discussion

In this section, we discuss the various performance issues of the proposed M_CSPF algorithm in comparison with the other existing algorithms described in section III.

1. Network Topologies

In terms of network topology, as the number of links and nodes increases, more alternative routes become available for selection. Therefore, the accommodation capacity of LSP becomes higher in general, but the complexity of finding the feasible route will be increased in proportion to the number of increased links or nodes. In addition, the equipment and operating costs of the network grow as the number of links and nodes increase, so a topology with a small number of links and nodes is desirable for network service providers [22], [23].

In this paper, we consider three different network topologies: tree network, ring network, and star network topologies, which

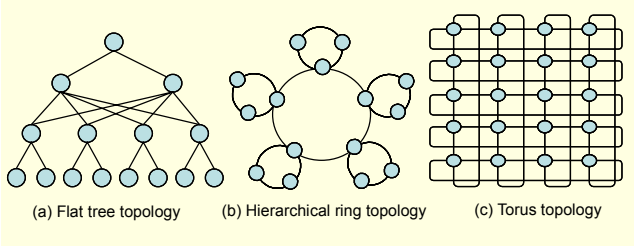

Fig. 7. The network topologies for verifying the existing CBR algorithms and our M_CSPF algorithm.

are shown in Fig. 7. In order to strictly verify the CBR algorithms, including our M_CSPF, we selected some complex network topologies.

The characteristics of the network topology should depend on the number of available routes for each pair of ingress/egress nodes. As the number of links increases, more alternative routes become available for selection. So, the accommodation capacity of LSP becomes higher in general. However, the equipment needs and operating cost of the network grow as the number of links increases, so a network topology with a small number of links is desirable for network service providers [23].

In this paper, we evaluate the proposed algorithm under three generic network topologies: flat tree, hierarchical ring, and torus topologies. The flat tree topology is a network with additional links from lower layers to higher layers. The hierarchical ring topology is a network that connects rings hierarchically. On the other hand, the torus topology is a grid network whose edge nodes are linked.

2. Performance Discussion

The main purposes of MPLS traffic engineering are to improve and control the end-to-end quality and enhance the efficiency of network resource utilization. Therefore, the evaluation of MPLS traffic engineering schemes should consider these two viewpoints. To evaluate the proposed algorithm and compare it with the others, we evaluated the random LSP accommodation of symmetric topologies using practical algorithms for MPLS traffic engineering, either actually installed by some vendors or ones that can be installed easily.

In order to evaluate the performance in terms of efficiency of network resource utilization and end-to-end quality, we defined some simulation parameters, which are shown in Table 2.

A. Efficiency of Network Resource Utilization

In order to evaluate the efficiency of network resource utilization of various algorithms, including our algorithm, we use the following criterion, which gives the capacity of the network:

Table 2. Simulation parameters.

Parameter	Value
Bandwidth of each link	500 Mbps
Propagation delay of each link	Discrete value, 2, 3, and 6 (ms)
Bandwidth required by LSPs	Discrete value, 1, 10, and 50 (Mbps)
End-to-end delay required by LSPs	Discrete value, 8 and 10 (ms)
Max queue length of each LSR	16,348 (packets)

$$\text{Network capacity} = \left(\frac{\text{the number of accomodated LSPs}}{\text{total number of LSPs}}\right). \quad (9)$$

We tried to create 500 LSPs between the random ingress and egress pairs with random discrete bandwidth for all algorithms and random discrete end-to-end delay constraint for our algorithm only. In this experiment, we did not consider the total traffic load, which will be discussed in a later section. We only measured the number of accommodated LSPs of each algorithm in this section.

Table 3 shows the comparison of the characteristics and average accommodation ratio of each algorithm under the three different network topologies. Our algorithm showed the best accommodation ratio over WSP, SWP, SDP, and MIRA.

In the case of well-known CSPF algorithms such as WSP, SWP and SDP, they showed a lower accommodation ratio than MIRA and our algorithm because they did not support the concept of a critical link. However, MIRA and our algorithm, which both support the concept of a critical link for future interference, showed a more enhanced accommodation ratio compared to WSP, SWP and SDP because they can minimize future interference, supporting the critical link concept that was originally proposed by MIRA and modified by our algorithm.

Each algorithm showed different accommodation ratios according to the network topologies, namely, flat tree, hierarchical ring, and torus topologies. All algorithms showed the best accommodation ratio under the torus topology and showed the worst accommodation ratio under the hierarchical ring topology, as shown in Fig. 8 and Table 2.

MIRA and M_CSPF showed nearly the same accommodation ratio, but MIRA did not support the end-to-end delay constraint and total offered traffic load. Because our algorithm, M_CSPF, supports the concept of total offered traffic load, it showed a slightly enhanced performance (1 to 1.5%) compared to MIRA under the three different network

Table 3. Comparison of the characteristics and average accommodation ratio of each algorithm under the three different network topologies.

Algorithms	Characteristics				Average accommodation ratio		
	Bandwidth guarantee	Delay guarantee	Critical interference	Total offered traffic load	Flat tree	Hierarchical ring	Torus
WSP	yes	no	no	no	62%	43%	68%
SWP	yes	no	no	no	60%	39%	68%
SDP	yes	no	no	no	66%	41%	79%
MIRA	yes	no	yes	no	86%	63%	87%
M_CSPF	yes	yes	yes	yes	87%	63%	91%

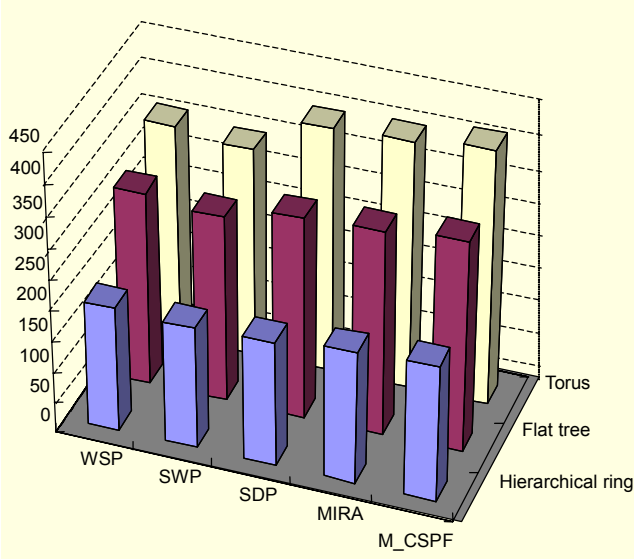

Fig. 8. Comparison of the number of accommodated LSPs under the three different network topologies.

topologies. In addition, our algorithm can provide the LSP that meets the end-to-end delay constraints, which are not supported by MIRA.

B. End-to-End Quality

In order to evaluate the performance of routing algorithms, E. Crawley [24] used the throughput value by statistical multiplexing. However, in this model, each LSP reserves links using the maximum transfer rate, so hardly any throughput deterioration appears. Instead of E. Crawley's [24] approach, we used the one-way delay and packet loss rate, which are affected by quality deterioration earlier than the throughput, as the evaluation criterion of the end-to-end delay [21]-[26]. The delay can be defined as the sum of the delays at each link along the LSP's route, and the packet loss arises in each link independently.

Figure 9 shows the one-way delays and path computation complexity for all algorithms under the hierarchical ring

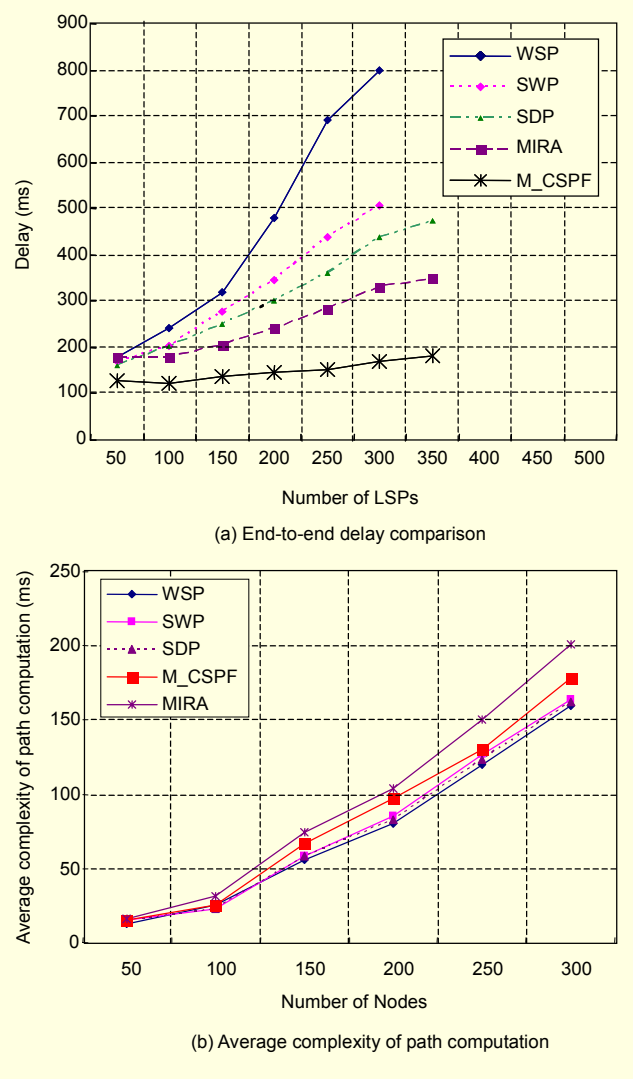

Fig. 9. Comparison of end-to-end delay and path computation complexity under the hierarchical ring topology.

topology. In terms of end-to-end delay, the hierarchical ring topology showed the worst LSP accommodation ratio compared to the two other topologies, which are flat ring and

torus. However, the same tendencies were found for the other topologies.

Because all the algorithms, with the exception of the algorithm proposed in this paper, did not support the end-to-end delay constraint, we measured the delay under the created LSPs without end-to-end delay constraint. However, in the case of our algorithm, we measured the delay with an end-to-end delay constraint of 6 ms.

As shown in Fig. 9, our algorithm showed the best performance compared to the others in terms of end-to-end delay and path computation complexity under the hierarchical ring topology. Because our algorithm can create an LSP with the end-to-end constraint under the weighted network graph while taking into account the total offered traffic load, which is impossible in the others, our algorithm showed the best performance in terms of end-to-end delay, including a performance enhancement of approximately 7% compared to MIRA. However, MIRA and our algorithm showed some greater performance degradation than WSP, SWP, and SDP. Because WSP, SWP, and SDP consider only the bandwidth constraint, they showed good performances compared to MIRA and our algorithm.

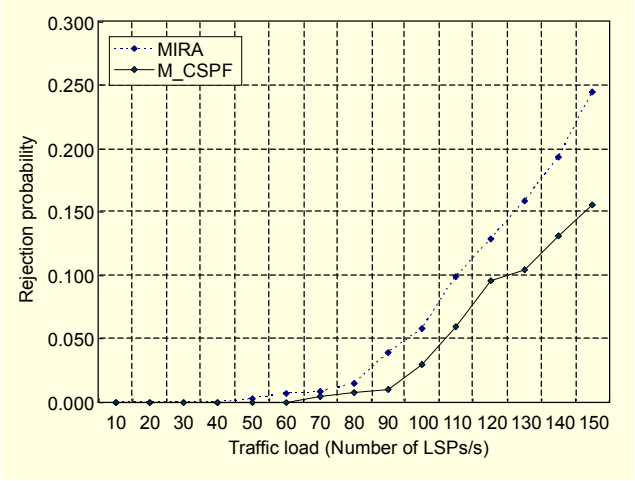

Fig. 10. LSP rejection probability versus the average offered load of the network in Fig. 3.

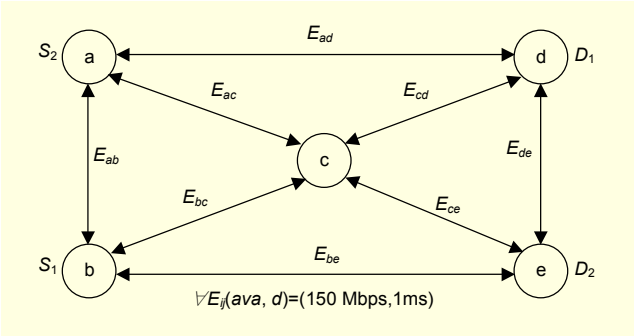

Fig. 11. Network topology with a large number of critical links.

Next, we describe the performance comparison between MIRA and our algorithm, M_CSPF, under a network topology with unbalanced offered load and a network topology with a large number of critical links.

At first, we measured their performances under a network topology with unbalanced offered load, as shown in Fig. 2. There are three ingress and egress pairs, which are (S_1,D_1), (S_2,D_2), and (S_3,D_3). Each pair of ingress and egress has different offered traffic loads. We assumed that the traffic load (S_1,D_1) is five times higher than (S_2,D_2) and (S_3,D_3). In addition, the offered traffic loads at (S_2,D_2) and (S_3,D_3) are the same.

Our algorithm showed a performance enhancement of approximately 22% compared to MIRA. The difference was caused by the method of controlling the traffic load. In the case of MIRA, it considers the critical link and not the future traffic load, which limits the possible candidate routes between any ingress and egress pair. However, our algorithm simultaneously deals with the future traffic load and the critical link, which can maximize the accommodation probability.

In addition, we measured the performance of rejection probability by gradually increasing the offered traffic load under the bi-directional network topology with a large number of critical links, as shown in Fig. 11. We assumed that the available capacity and delay of all links are 150 Mbps and 1 ms, respectively. We also assumed that balanced traffic was offered at each ingress nodes of S_1 and S_2.

In the case of MIRA and M_CSPF, there are six critical links (E_{ab}, E_{bc}, E_{be}, E_{ad}, E_{cd}, and E_{de}) for an LSP setup between the ingress/egress pair (S_1,D_1), and there are six critical links (E_{ab}, E_{bc}, E_{ad}, E_{ad}, E_{be}, and E_{de}) for an LSP setup between the ingress/egress pair (S_2,D_2). This means that, in the case of MIRA, the possible route between the (S_1,D_1) pair can be <*a-c-e*> and the possible route between the (S_2,D_2) pair can be <*b-c-d*>. Thus, the critical disadvantage of MIRA is that the rejection probability is much higher than in our algorithm (M_CSPF) because MIRA does not take the traffic load into account, only the number of critical links. However, our algorithm showed a more balanced rejection probability compared to MIRA because it simultaneously takes into account the offered traffic load and the number of critical links.

In addition, we should evaluate the performance of our algorithm in the case of network topology change. As described in the previous section, our algorithm computes the link weight using the LWCA only when there are changes in the network topology and total traffic load. We also described how our algorithm can enhance the path computation performance because it can limit the graph search when the accumulated link delay exceeds the requested end-to-end delay constraint.

In order to evaluate the performance of our algorithm under the situation of network topology change and variant end-to-end delay constraint with the torus network topology, as shown in Fig. 7 (c), we measured the path computation performances of our algorithm and MIRA by gradually expanding the network topology and simultaneously increasing the end-to-end delay constraint. At each step, we added four nodes to the existing torus network topology, as shown in Fig. 7 (c), and increased the end-to-end delay by 0.1 ms starting from 0.5 to 1.5 ms.

As shown in Fig. 13, our algorithm showed the best LSP computation performance in cases where the network topology is not changed. However, both MIRA and our algorithm showed nearly the same LSP computation performance in cases where the network topology is changed. However, in a real network environment, the network topology is not as frequently changed as in the simulated condition. Therefore, our algorithm showed the same LSP computation performance as MIRA under the worst case of frequent network topology change. However, our algorithm showed a greatly enhanced LSP computation performance when the network topology is not changed.

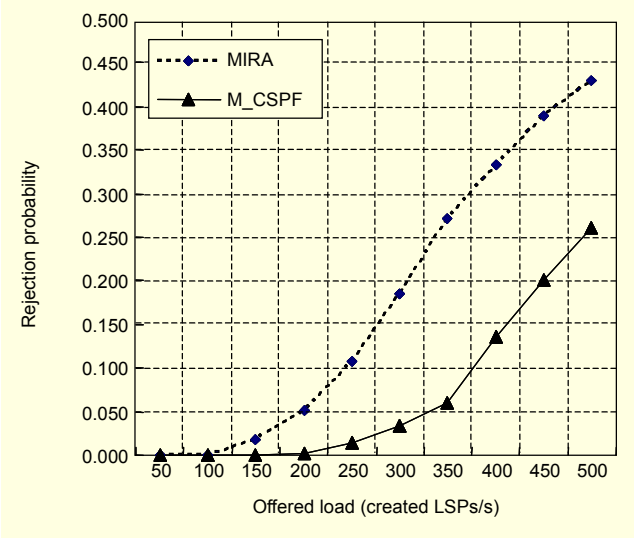

Fig. 12. Comparison of rejection probability under the large number of critical links.

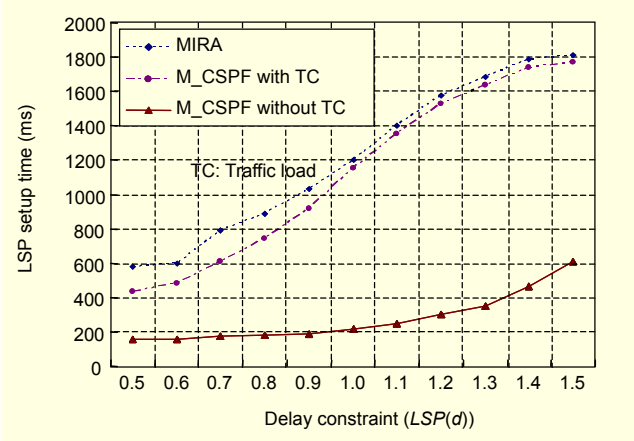

Fig. 13. LSP computation performance according to network topology change and gradual increment of end-to-end delay constraint.

On the other hand, because MIRA does not support the end-to-end delay constraint, it showed a nearly linear increase in LSP computation time regardless of the end-to-end delay bound. However, our algorithm showed a more enhanced performance than MIRA even in the case of network topology change because it does not traverse the network topology any more when the accumulated delay at any link exceeds the end-to-end delay constraint. However, MIRA traverses all the network topology to compute the network topology because it basically uses the Dijkstra algorithm to compute the LSP route.

On the other hand, bandwidth utilization can be defined as the aggregate of all traffic currently being consumed on a hop or path. We measured the bandwidth utilization under the three different network topologies of flat tree, hierarchical ring, and torus. As a result of our evaluation, M_CSPF showed high bandwidth utilization of an average of 98% because it takes into account bandwidth, future traffic arrival, and link interference. However, the bandwidth utilizations of WSP, SWP, SDP, and MIRA were less than that of our algorithm.

Therefore, our algorithm is suitable for the complex network topology with bounded end-to-end delay constraint. Since the value of the bounded delay constraint is small, the LSP path computation performance will be more enhanced compared to MIRA because the bounded delay constraint takes the effect of network topology pruning.

As a result of the performance evaluation, our algorithm showed the best performance, with the exception of the path computation complexity, among all the others while conforming to the optimal path provision meeting multiple constraints such as bandwidth and end-to-end delay. Our algorithm showed a lower LSP setup rejection probability than WSP, SWP, DSP, and MIRA because our algorithm simultaneously takes into account the total offered traffic load and the link interference. In addition, by introducing the bounded order assignment concept, we can also enhance the LSP route computation performance when the network topology is changed, showing a slight enhancement in performance compared to MIRA in the case of a network topology change. Also, our algorithm showed a more enhanced LSP computation performance by reducing the value of the end-to-end delay constraint.

V. Conclusion

Because the major objective of MPLS traffic engineering is

the enhancement of network resource utilization efficiency and end-to-end quality, we measured the efficiency of network resource utilization and end-to-end quality of the proposed algorithm and other algorithms such as WSP, SWP, SDP, and MIRA under three different network topologies: flat tree, hierarchical ring, and torus. In addition, we also measured the LSP setup rejection probability of our algorithm and MIRA under an unbalanced topology and a balanced network topology with a large number of critical links. We also compared the LSP computation performance of our algorithm and MIRA by gradually adding four nodes and by increasing the value of the end-to-end delay constraint under the torus network topology.

By separating the link weight computation step from the LSP route provision, we can achieve the enhanced performance LSP route provision for MPLS traffic engineering. Our algorithm showed the best performance compared to WSP, SDP, SWP, and MIRA under the flat tree, hierarchical ring, and torus network topologies. In terms of efficiency of network resource utilization, we found that our algorithm showed a 2% enhancement in LSP accommodation ratio compared to MIRA. In terms of the end-to-end quality, our algorithm showed a performance enhancement of approximately 7% compared to MIRA in the case of end-to-end delay.

In addition, the performance comparison of LSP rejection probabilities under the network topology, where the unbalanced total traffic load arrives at every ingress nodes, showed that the rejection probability of our algorithm was less than MIRA's by approximately 20%.

In terms of LSP computation performance, our algorithm also showed a better performance than MIRA in cases where the network topology was not changed, while our algorithm revealed that its LSP computation performance in the case of network topology change is slightly more enhanced than MIRA's because our algorithm has the bounded order assignment. By pruning some parts of the network topology in the process of traversing the network topology, our algorithm enhanced its LSP route computation performance better than MIRA. We also found that our algorithm showed a good LSP route computation performance by minimizing the value of the end-to-end constraint because the portion of the pruned network topology will be widened by minimizing the value of the end-to-end constraints.

Thus, our algorithm is suitable for providing the optimal LSP route computation with the bandwidth and delay constraints under a complex network topology. While our algorithm showed a good performance under every condition and topology, our algorithm is particularly good for application in LSP computation where the end-to-end delay constraint is tight.

Summary of Notation

$G(N,E)$	Network graph, where N is node and E is link
B_{ava}	Available bandwidth
B_{req}	Requested bandwidth
E_{ij}	Communication link (edge) connecting LSR_i and LSR_j
$E_{ij}(ava)$	Available bandwidth on E_{ij}
$E_{ij}(F)$	Feasibility on E_{ij}, which can be YES or NO
$E_{ij}(S)$	Status on E_{ij}, which can be normal, fault, congestion or performance degradation
$E_{ij}(w)$	Weight assigned to E_{ij}, which is determined by LWCA
$E_{ij}(load)$	Future traffic load on E_{ij}
$E_{ij}(cc)$	The number appearance of E_{ij} as the critical link
$E_{ij}(d)$	Delay on E_{ij}
S_i	A source LSR_i to create LSP
D_i	An destination LSR_i to create LSP
$T_{load}(S_i)$	Anticipating traffic load that will be injected at the ingress LSR, S_i
$P_{candidate}(S_i,D_i)$	All the possible routes between source LSR, S_i and destination LSR, D_i
$N(o)$	Order assigned to node
$E(o)$	Order assigned to edge
N_{active}	Active node
$N_{passive}$	Neighbor node that is connected to active node via edge
$G(N,E_w)$	Weighted network graph that is created by LWCA
N_{dest}	Destination node
$P(R_b,R_d,S,D)$	Path creation requirement, where R_b is requested bandwidth, R_d is requested end-to-end delay, S is source node and D is destination node
H_{factor}	A random large number, which is multiplied by a random number taking into account the delay
$N_{active}(o)$	Order of active node, N_{active}
$N_{passive}(o)$	Order of passive node, $N_{passive}$
$N_{active}(f)$	Feasibility of active node, N_{active}, which can be YES or NO
$N_{passive}(f)$	Feasibility of passive node, $N_{passive}$, which can be YES or NO

References

[1] D. Awduche, J. Malcolm, J. Agogbua, and M. O'Dell, Requirements for Traffic Engineering Over MPLS, *RFC 2702*, Sept. 1999.

[2] R. Guerin, Ariel Orda, and D. Williams, "QoS Routing Mechanisms and OSPF Extension," *Proc. of 2nd Global Internet Miniconference (Joint with Globecom'97)*, Nov. 1997.

[3] B. Davie and Y. Rekhter, "MPLS Technology and Applications," Morgan Kaufmann Publishers, 2000.

[4] R. Guerin, A. Orda, and D. Williams, "QoS Routing Mechanism and OSPF Extensions," *Proc. of IEEE GLOBECOM'97*, Nov. 1997.

[5] P. Aukia, M. Kodianlam, P.V. Koppol, T.V. Lakeshman, H. Sarin, and B. Suter "RATES: A Server for MPLS Traffic Engineering," *IEEE Network*, vol. 14. no. 2, Mar.-Apr. 2000.

[6] Int'l Eng. Consortium, A Comparison of Multiprotocol Label Switching (MPLS) Traffic-Engineering Initiatives, http://www.iec.org/online/tutorials/, Web ProForum Tutorials, 2003.

[7] Nortel Network, "Using Constraint-Based Routing to Deliver New Services," http://www.nortelnetworks.com/products/library/collateral/55046.25-10-99.pdf.

[8] Q. Ma and P. Steenkiste, "On Path Selection for Traffic with

Bandwidth Guarantees," *Proc. of IEEE Int'l Conf. on Network Protocol*, Oct. 1997.

[9] Q. Ma and P. Steenkiste, "On Path Selection for tra.c with Bandwidth Guarantees," *Proc. of IEEE ICNP*, 1997, pp. 191-202.

[10] M. Kodialam and T.V. Lakshman, "Minimum Interference Routing with Applications to MPLS Traffic Engineering," *Proc. of INFOCOM,* Mar. 2000.

[11] Z. Wang and J. Crowcroft, "QoS Routing for Supporting Multimedia Applications," *IEEE JSAC*, vol. 14, no. 7, 1996, pp. 1228-1234.

[12] P.S. Qingming Ma, "On Path Selection for Traffic with Bandwidth Gurantees," *Proc. of IEEE Int'l Conf. on Network Protocols*, Oct. 1997, pp. 191-202.

[13] Yufei Wang and Zheng Wang, "Explicit Routing Algorithms for Internet Traffic Engineering," *Computer Commun. and Networks*, 1999, pp. 582-588.

[14] E. Crawley, R. Nair, and H. Sandick, "A Framework for QoS-based Routing in the Internet," *IETF RFC2386*, Aug. 1998.

[15] Z. Wang and J. Crowcroft, "Quality of Service Routing for Supporting Multimedia Applications," *IEEE J. on Selected Areas in Comm.*, vol. 14, no. 7, Sept. 1996, pp. 1228-1234.

[16] R. Boutaba, W. Szeto, and Y. Iraqi, "DORA: Efficient Routing Algorithm for MPLS Traffic Engineering," *Int'l J. of Network and Systems Management, Special Issue on Traffic Eng. and Management*, vol. 10, no. 3, 2002, pp. 311-327.

[17] Woo-Seop Rhee, Jun-Hwa Lee, Jea-Hoon Yu, and Sang-Ha Kim, "Scalable Quasi-Dynamic-Provisioning-Based Admission Control Mechanism in Differentiated Service Networks," *ETRI J.*, vol. 26, no. 1, Feb. 2004, pp. 27-37.

[18] Hun-Jeong Kang, Myung-Sup Kim, and James W. Hong, "Streaming Media and Multimedia Conferencing Traffic Analysis Using Payload Examination," *ETRI J.*, vol. 26, no. 3, June 2004, pp. 203-217.

[19] X. Xiao and L.Ni, "Internet QoS: A Big Picture," *IEEE Network*, vol. 13, no. 2 , Mar.-Apr. 1999, pp. 8-18.

[20] Y. Wang and Z. Wang, "Explicit Routing Algorithms for Internet Traffic Engineering," *Proc. of IEEE ICCCN99*, Oct. 11-13, 1999.

[21] K. Kompella and D.O. Awduche, "Notes on Path Computation in Constraint-Based Routing," *Internet-Draft*, July 1999.

[22] L.L.H. Andrew and A.A.N.A. Kusuma, "Generalised Analysis of a QoS-Aware Routing Algorithm," *Proc. of IEEE Globecom'98*, 1998, pp. 118-123.

[23] Satoshi Kamei and Takumi Kimura, "Evaluation of Routing Algorithms and Network Topologies for MPLS Traffic Engineering," *Proc. of IEEE GLOBECOM 2001*, no. 1, Nov 2001, pp. 25-29.

[24] E. Crawley, R. Nair, B. Rajagopalan, and H. Sandick, A Framework for QoS-Based Routing in the Internet, *RFC 2386*, Aug. 1998.

[25] G.R. Ash, Traffic Engineering & QoS Methods for IP-, ATM-, & TDM-Based Multiservice Networks, *Internet-Draft*, July 2000.

[26] X. Xiao, A. Hanan, and B. Bailey, "Traffic Engineering with MPLS in the Internet," *IEEE Network*, vol. 14, no. 2, Mar.-Apr. 2000.

Daniel W. Hong received the BS, MS degrees in computer science from Hannam University, Konkuk University, and PhD in computer engineering from Kyung Hee University, Korea, in 1987, 1989, and 2005 respectively. In 1993 he joined Korea Telecom (KT), where he worked on a TINA-C related Service and Network Management Project as a Member of Technical Staff. For the last few years, he has been working on various research and system design and development projects in KT. Beginning in 1996, he helped develop an ATM network management system for 5 years. Since 2002, he has been involved in the project of the design and implementation of KT New Operations Support System (NeOSS) as a Director. His research interests include MPLS/GMPLS traffic engineering, flow-through service provisioning architecture in telecommunications environment, MPLS VPN, active network management, and policy-based network management. He is a Member of IEEE, IEICE, KNOM, and Korean Institute of Communication Sciences (KICS).

Choong Seon Hong received the BS and MS degrees in electronics engineering from Kyung Hee University, Seoul, Korea, in 1983 and 1985. In 1988 he joined KT, where he worked on N-ISDN and Broadband Networks as a Member of Technical Staff. In September 1993, he joined Keio University, Japan. He received the PhD degree at Keio University in March 1997. He worked for the Telecommunications Network Lab, KT as a Senior Member of Technical Staff and as the Director of the Networking Research Team until August 1999. Since September 1999, he has worked as a Professor of the School of Electronics and Information, Kyung Hee University. His research interests include network management, network security, sensor networks, and mobile networking. He is a Member of IEEE, IEICE, IPSJ, KISS, KIPS, and KICS.

Gil-Haeng Lee is a Principal Member of Engineering Staff at ETRI, Korea. He received the BS degree in computer science from Chonnam National University, Korea, and the MS and PhD degrees in computer science from KAIST in 1986 and 1996. His research interests are in SLA, CNM, NMS, load balancing and distributed processing, network management, speech recognition, and real time DBMS.