Computer Communications 33 (2010) 1557–1565

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Multiple path selection algorithm for DiffServ-aware MPLS traffic engineering

Wu-Hsiao Hsu [a], Yuh-Pyng Shieh [a,*], Jenhui Chen [b]

[a] Department of Computer Science and Information Engineering, Ming Chuan University, Taoyuan, Taiwan
[b] Department of Computer Science and Information Engineering, Chang Gung University, Taoyuan, Taiwan

ARTICLE INFO

Article history:
Received 5 September 2008
Received in revised form 7 April 2010
Accepted 9 April 2010
Available online 21 April 2010

Keywords:
DiffServ-TE
MSA
CT
LSP

ABSTRACT

This paper proposes a new per-class bandwidth constraint algorithm, called the multipath selection algorithm (MSA), for a DiffServ-aware traffic engineering (DiffServ-TE). The MSA comprises three steps. First, a given source uses the MSA to find multiple label switch paths (LSPs) from the source to a destination for a specific class type (CT). Second, the source uses the available bandwidth of the CT on all the links along these LSPs to allocate the initial traffic to the selected LSPs. Third, the source dynamically adjusts traffic to these LSPs based on individual round trip time. Simulation results indicate that the proposed algorithm offers better performance than existing approaches in average transmission time, average packet loss rate, average throughput, and available bandwidth variance for each link.

1. Introduction

The current approach in providing quality of service (QoS) on the Internet is based on the DiffServ protocol [1]. DiffServ divides traffic into a small number of classes and allocates network resources on a per-class basis. In this architecture, packets are marked with different DiffServ code points (DSCP) at edge routers. Each DSCP is associated with a particular QoS class characterized by per-hop-behavior (PHB), and the core routers treat each packet with a specific PHB based on the DSCP carried by the packet. The PHB is achieved through a combination of scheduling and queue management schemes. The DiffServ architecture includes several standardized PHBs. The expedited forwarding (EF) PHB provides a low-loss, low-latency, low-jitter, and assured bandwidth service. The assured forwarding (AF) PHB supports services in which the customers are likely to get the negotiated service level agreement (SLA) without any guarantees.

Multiprotocol label switching (MPLS) traffic engineering (MPLS-TE) [2] makes it possible to establish bandwidth guaranteed label switched paths (LSPs) using constraint-based routing algorithm. However, since MPLS-TE operates without referring to different classes, it may not be optimal in a DiffServ network. Several analyses of integrating DiffServ and MPLS-TE can be found in [3–8]. The studies [3,4] introduce the concept of DiffServ-aware traffic engineering (DiffServ-TE). DiffServ-TE makes separate bandwidth reservations for different classes of traffic. Hence, traffic flows toward a given destination can be forwarded on separate LSPs based on class. For this purpose, the studies define the concept of a class type (CT) as the set of traffic trunks crossing a link, which is governed by a specific set of bandwidth constraints. A given traffic trunk belongs to the same CT at all links. The TE class is introduced as a pair of a CT and a preemption priority allowed for that CT. The IETF requires the support of up to eight CTs, referred to as CT0 through CT7. By definition, each CT is assigned to either a bandwidth constraint (BC), or a set of BCs. A CT represents a class in the DiffServ-TE architecture much like PHB represents a class for DiffServ. Note that flexible mappings between CTs and PHBs are possible.

The study [5] analyzes the QoS performance for different types of services in a DiffServ-TE network, including VoIP, real time video, and best effort data traffic. The study [6] proposes an architecture for the MPLS restoration routing of DiffServ traffic. This architecture, called per class aggregate information with preemption (CAIP), provisions two key QoS features for multimedia traffic: prioritized guaranteed bandwidth and fast restoration in the event of an element failure.

The study [7] proposes a new preemption policy that includes an adaptive scheme aimed at minimize rerouting. This policy combines the three main preemption optimization criteria: number of LSPs to be preempted, priority of the LSPs, and preempted bandwidth. All the studies above focus on preemption policy and restoration routing.

The study [8] proposes a Max–Min bandwidth constraint model that guarantees each CT without causing resource fragmentation. This paper also develops three new bandwidth preemption

* Corresponding author. Address: Department of Computer Science and Engineering, Ming-Chuan University, 5 Teh-Ming Rd., Gwie Shan District, Taoyuan, Taiwan. Tel.: +886 3 3507001/3429; fax: +886 3 3593874.
E-mail address: arping@gmail.com (Y.-P. Shieh).

0140-3664/$ - see front matter Crown Copyright © 2010 Published by Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2010.04.015

algorithms for three bandwidth constraint models, respectively, and focuses on how to design a bandwidth manager to support DiffServ-TE.

This paper proposes a new per-class bandwidth constraint algorithm, called the multipath selection algorithm (MSA), for a DiffServ-TE network. Unlike previous studies, which focus on preemption policy, restoration routing, or bandwidth manager, the proposed MSA finds multiple LSPs per-class and allows flexible division of traffic over these LSPs. The MSA comprises three steps. First, a given source uses the MSA to find multiple LSPs from the source to a given destination for a CT. Second, the source uses the available bandwidth of the CT on all the links along these LSPs to allocate initial traffic. Third, the source dynamically adjusts traffic for these LSPs based on individual round trip time.

1.1. Bandwidth constraint models

The maximum allocation model (MAM) [9], the Russian doll model (RDM) [10], and the maximum allocation with reservation (MAR) [11] are three IETF-proposed bandwidth constraint models for supporting DiffServ-TE. The author of [12] compared these three models and concluded that the RDM best matches DiffServ-TE. Hence, the MSA proposed in this study uses the RDM as a bandwidth constraint model. The RDM provides each class with a minimum amount of bandwidth, but lower priority classes can use the bandwidth of higher priority classes when that bandwidth is available.

1.2. The link state interior gateway protocol (IGP)

In the proposed MSA, each node in a DiffServ-TE network works in conjunction with the extensions of the open shortest path first (OSPF) protocol [13]. In the extended OSPF protocol, each node running a link state QoS routing protocol uses reliable flooding to exchange link state advertisements (LSAs) with its neighboring routers. Each LSA must advertise the available bandwidth per-CT on every link. Based on the reliable flooding of LSA, all nodes build identical link state databases that depict the entire OSPF network topology interconnected by a group of nodes. When the available bandwidth per-CT of one or more links changes, the link state database in each node must be updated immediately.

The remainder of this paper is organized as follows. Section 2 introduces the notation and problem in this study. Section 3 presents the proposed algorithm. Section 4 describes the simulation model and results, and Section 5 provides some conclusions.

2. Notation and problem description

2.1. Notation

Before formally introducing the proposed algorithm, the notation used throughout this paper will first be described. Let $G = (V, L)$ denote a DiffServ-TE network, where V is the set of nodes and L is the set of links. Suppose that a node represents a router. A path p from a node x to a node y is a sequence of nodes and links $x = v_0, l(v_0, v_1), l(v_1, v_2), \ldots, l(v_k, v_{k+1}), y = v_{k+1}$ and is denoted by $p(v_0, v_1, v_2, \ldots, v_k, v_{k+1})$. $AB(p)$ represents the maximum available bandwidth of path p, and is defined as $AB(p) = \min\{l_a^{(v_i, v_{i+1})}(CT_j) | 0 \leqslant i \leqslant k\}$. A link capacity between nodes x and y is denoted by $l_c(x,y)$. Let n denote the number of CTs, and BC_i denote the bandwidth reserved by CT_i, $0 \leqslant i \leqslant n$. The available bandwidth of a link between nodes x and y for the CT_j can be denoted by $l_a^{(x,y)}(CT_j)$, $0 \leqslant j \leqslant n$. Since the proposed MSA uses the RDM as the bandwidth constraint model, the link available bandwidth $l_a^{(x,y)}(CT_j)$ can be computed as follows,

$$l_a^{(x,y)}(CT_j) = \sum_{i=j}^{n} l_a^{(x,y)}(CT_i) + \left(l_c(x,y) - \sum_{i=0}^{n} l_a^{(x,y)}(BC_i) \right). \quad (1)$$

2.2. Problem description

In Fig. 1, the number indicated at each link represents the current link available bandwidth for a CT_j (for example, $l_a^{(A,B)}(CT_j) = 8$), and the bandwidth requested for a particular LSP from the source A to the destination H is 10 Mbps. Clearly, no single path has enough bandwidth to meet the 10 Mbps requirement. If the round trip time from source A to the destination H is also considered, finding a path which meets the request and has the minimum round trip time is a NP problem [14]. In fact, the network architecture in Fig. 1 shows that there is more than one path from the source A to the destination H. Thus, concurrent multi-path transmission can meet the request when a single path transmission cannot.

3. The multipath selection algorithm (MSA)

The main purpose of the MSA is to meet the requested bandwidth by finding multi-LSPs for a CT in a DiffServ-TE network. The MSA procedures can be divided into two parts:

(1) Finding multi-LSPs for the request.
(2) Allocating network traffic to the selected LSPs.

3.1. Finding multi-LSPs

The MSA uses two metrics to find multi-LSPs: the path round trip time and the available bandwidth of each link comprising the path. Since router queuing delay completely dominates the path transmission delay from a source to a destination, the source selects the path with the fewest hops as a LSP. The reason for selecting a path with as few hops as possible is that more links on a path not only consume more network resources, but also increase propagation delay [15]. In other words, a LSP with fewer hop counts (HCs) will have a shorter path round trip time.

The three principles of finding the multi-LSPs are as follows:

(I) All the found LSPs have no loop.
(II) The source selects the path with sufficient available bandwidth and the fewest number of nodes as a LSP. Any link in the LSP can be selected repeatedly by other LSPs as long as the link has enough available bandwidth. When a link no longer has enough bandwidth to carry more LSP traffic, these links are not selected.
(III) If nodes want to keep the most current view of the available bandwidth on all links in the network, they must update the link state database frequently. However, frequent link state database updates are neither scalable nor practical every

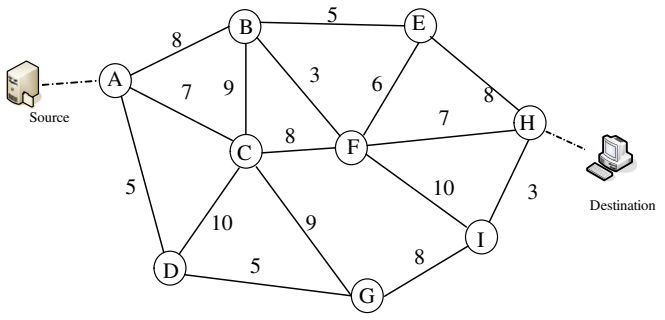

Fig. 1. A simple network architecture.

time the available bandwidth of a link changes. Consequently, the proposed algorithm does not distribute the link state database whenever the available bandwidth of a link changes. Instead, the source node records the changed values once the available bandwidth of a link changes.

Fig. 2 shows the execution steps of finding LSPs.

Taking the network topology shown in Fig. 1 as an example, suppose that Node A represents the source and Node H represents the destination. The steps for finding LSPs are as follows:

Step 1: Since the boundary node is responsible for the complex computation in the DiffServ-TE network, Node A (which is a boundary node) in Fig. 1 determines how many LSPs should be used to transmit traffic. Fig. 1 shows that Nodes B, C, and D are connected to Node A after Node A checks the link state database. Thus, with Nodes B, C, and D as starting points, Node A can determine that the number of LSPs is three.

Step 2: Let LSP_B represent the LSP starting with Node B. Node A uses the link state database to find the LSP_B that has sufficient available bandwidth and the fewest number of nodes passing through it to reach Node H. First, Node A sets the HC value of Node B at 0. Then, Node A uses the link state database to find all the nodes adjacent to Node B. Finally, the HC values of all nodes adjacent to Node B are set as 1. For example, Fig. 3 shows that Node A sets the HC values of Nodes C, E, and F at 1.

Step 3: Repeat Step 2 for other nodes. Node A set all the nodes adjacent to the nodes whose HC value is 1 at 2 instead. Fig. 4 shows that the HC values of all the nodes adjacent to Nodes C, E, and F are set at 2. Note that if these adjacent nodes already have a HC value, their HC values do not need to be changed because the updated HC value may be larger than or equal to the original HC value. Again, Fig. 4 clearly shows that Nodes B, C, E, H, and I are adjacent to node F. The HC values of nodes H and I are set at 2, but Node A does not change the HC values of nodes B, C, and E. Likewise, Node A sets the HC values of Nodes D and G at 2.

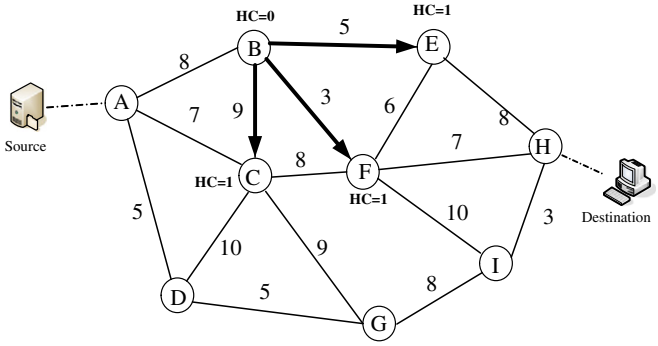

Fig. 3. Node A sets the HC values of Nodes C, E and F at 1.

Step 4: Repeat this process until the destination Node H is found. Fig. 4 shows that Node A sets the HC value of Node H at 2. Node A records the individual HC values from Node B to each of the nodes in the network, as Table 1 indicates. The shortest path from Node B to Node H passes through two hops. Hence, Node A may find a path which starts from Node B and reaches the destination Node H through the nodes with HC values of 0, 1 and 2, according to the link state database and Table 1.

Step 5: There may be more than one shortest path from Node B to Node H (passing through two hops). Fig. 4 shows that there are two shortest paths from Node A to Node H: path $p_1(A,B,E,H)$ and path $p_2(A,B,F,H)$. In this case, Node A selects the path with

```
Let SRC and DEST be two nodes in G to denote the source and the
destination of LSPs.

LSPS findLSP(V,L,SRC,DEST,AB){
    Let Q={} be a first-in-first-out queue of nodes.
    Let HC be an integer field associated with nodes to denote the hop count of
        shortest paths from SRC.
    Let HCC=-1 be a temporary hop count variable.
    Set HC(SRC)=HCC and push into Q.
    Do{
        Pop the front element of Q as node x.
        Let R={y|(x,y)∈L and HC(y)=null} be the neighbors of the node x.
        For each y∈R, let HC(y)=HCC and push y into Q.
    }While(DEST∉Q).
    Let LSPS={(DEST)} be a set of paths on G(V,L) and initially there is only one
        path (DEST) with zero length in LSPS.
    Let SUCC=true be a boolean variable to denote whether the construction
        of LSPS is completed.
    Do{
       Set SUCC=true.
       For each path p in LSPS{
            Let p=(x,v1,v2,...) be a path starting from node x.
            If x=SRC, continue next path p.
            Let y be a neighbor of x and HC(y)=HC(x)-1.
            Set p=(y,x,v1,v2,...).
           Set SUCC=false;
       }
    }While(SUCC is false).
    Finally return LSPS.
}
```

Fig. 2. The steps to find LSPs.

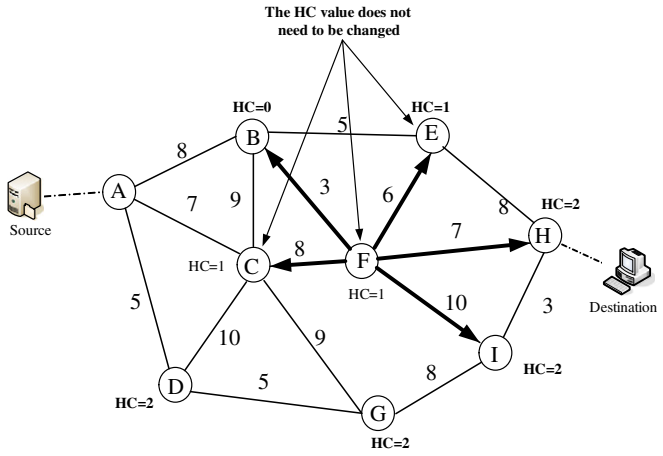

Fig. 4. Node A sets the HC values of the nodes adjacent to Nodes C, E, and F at 2.

Table 1
All the HC values recorded by Node A.

Destination node	Hop count
A	0
B	0
C	1
D	2
E	1
F	1
G	2
H	2
I	2

the largest available bandwidth. That is, $AB(p_1)$ and $AB(p_2)$ can be computed as follows:

$$AB(p_1) = min\{l_a^{(A,B)}(CT_j), l_a^{(B,E)}(CT_j), l_a^{(E,H)}(CT_j)\} = min\{8,5,8\} = 5\ \text{Mb}.$$

$$AB(p_2) = min\{l_a^{(A,B)}(CT_j), l_a^{(B,F)}(CT_j), l_a^{(F,H)}(CT_j)\} = min\{8,3,7\} = 3\ \text{Mb}.$$

Thus, Node A selects path p_1(A,B,E,H). At this time, the LSP_B is found and denoted by LSP_B(A, B, E, H).

Step 6: The available bandwidth of each link the LSP_B passes through must be updated. In this update procedure, Node A subtracts the maximum available bandwidth of the LSP_B from the available bandwidth of each link passing through the LSP_B. Thus, Node A subtracts 5 Mb from the available bandwidth of links l(A,B), l(B,E) and l(E,H). Fig. 5 shows the updated result. Note that only Node A records the update for the available bandwidth of each link. Therefore, the available bandwidth of each link in the link state database remains unchanged.

Step 7: Node A takes Nodes C and D as starting points, and repeats Steps 2 through 6 to find the LSP_C and LSP_D, separately. According to Principle II described in Section 3.1, l(B,E) will not be selected again because Node A has updated the available bandwidth of this link to 0. Fig. 6 shows that the LSP_C and LSP_D are found and denoted by LSP_C(A,C,F,H) and LSP_D(A,D,G,I,H), respectively.

3.2. Minimizing the maximum round trip time of selected LSPs

Assume that the delay from the source to the destination is at its minimum when the round trip time of each selected LSP is the same or similar. If this assumption is true, allocating and adjusting traffic to these selected LSPs involves minimizing the round trip time between the source and the destination. Therefore, before describing how to allocate and adjust traffic, this assumption must be validated first.

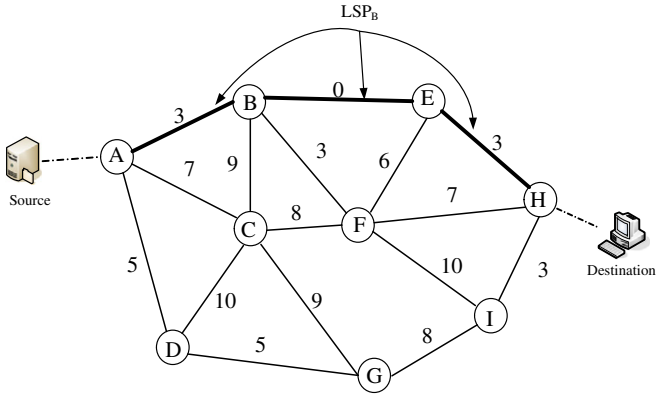

Fig. 5. Node A updates the available bandwidth of the links that the LSP_B passes through.

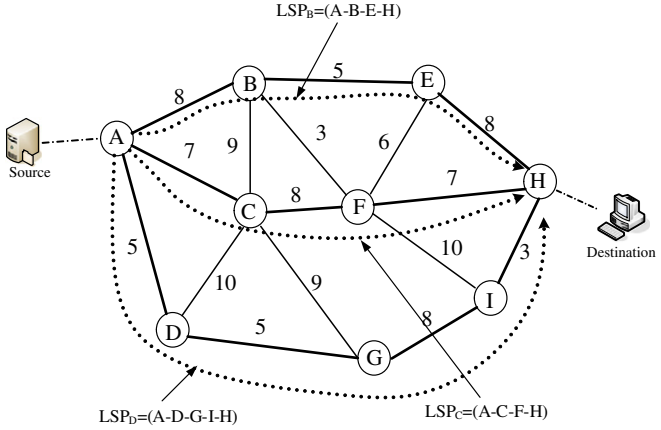

Fig. 6. Three LSPs found by node A.

Suppose that N LSPs from the source to the destination have been found, and the variables required are defined as follows:

T: the maximum round trip time among N LSPs.
t_i: the round trip time from the source to the destination via the LSP_i when it carries no traffic.
c_i: the round trip time from the source to the destination via the LSP_i when it carries some traffic.
m_i: the allocated traffic for the LSP_i.

Validation:

To validate that the delay from source to destination is at its minimum when the round trip time of each selected LSP is equal or similar, set $T = (t_1 + d_1) = (t_2 + d_2) = \cdots = (t_N + d_N)$, where $d_i = c_i - t_i$, $1 \leqslant i \leqslant N$.

Assuming that there exists a T', $T' < T$, both T' and T have the same network environment, and the amount of traffic load is also the same. T' can be represented as follows:

$$T' = max\{(t_1 + d_1'), (t_2 + d_2'), \ldots, (t_N + d_N')\} \quad (2)$$

Similarly,

$$T = max\{(t_1 + d_1), (t_2 + d_2), \ldots, (t_n + d_n)\} = (t_1 + d_1) = (t_2 + d_2) = \cdots = (t_N + d_N) \quad (3)$$

Through (2) and (3), see that

$$(t_1 + d'_1) < (t_1 + d_1), (t_2 + d'_2) < (t_2 + d_2), \ldots, (t_N + d'_N) < (t_N + d_N) \quad (4)$$

In (4), t_i can be eliminated

$$d'_1 < d_1, d'_2 < d_2, \cdots, d'_N < d_N \quad (5)$$

Since a longer round trip time leads to more transmission traffic, $d'_1 < d_1, d'_2 < d_2, \cdots, d'_N < d_N$ represent

$$m'_1 < m_1, m'_2 < m_2, \cdots, m'_N < m_N \quad (6)$$

If the right and the left of (6) are added, then

$$m'_1 + m'_2 + \cdots + m'_N < m_1 + m_2 + \ldots + m_N \quad (7)$$

If $T' < T$, then (7) must hold, which conflicts the premise. Thus, T is the minimum value.

3.3. Allocating network traffic to the selected LSPs

The proposed approach uses two steps to allocate network traffic to the selected LSPs:

Step 1: The source allocates the initial traffic to each selected LSP according to its individual maximum available bandwidth.
Step 2: The source adjusts traffic dynamically based on the round trip time of the LSPs.

The following section describes these allocation steps in detail.

Step 1. Initial traffic is allocated according to the maximum available bandwidth of each LSP.

Suppose N LSPs are found, the round trip time of LSP_i is $(t_i + d_i)$, and the bandwidth requested for a particular LSP is b_r Mb. First, the source obtains the maximum available bandwidth of each LSP from the link state database. Then, initial traffic is allocated to each LSP according to its individual maximum available bandwidth. The traffic allocated to each LSP is proportional to its individual maximum available bandwidth. Thus, the allocated initial traffic for any LSP_j is

$$\left[\frac{AB(LSP_j)}{\sum_{i=1}^{N} AB(LSP_i)}\right] \times b_r \text{Mb}. \quad (8)$$

Finally, the source measures the round trip time of each LSP. Let M_t be the average round trip time, which can be denoted as follows:

$$M_t = \frac{\sum_{i=1}^{N}(t_i + d_i)}{N}. \quad (9)$$

Step 2: Dynamic adjustment of each LSP's traffic.

After the source measures the round trip time of each LSP and calculates the average round trip time, the following condition is used to determine whether or not the traffic carried on each LSP must be adjusted:

Condition: Determine whether or not the range of the round trip time is too large. This range is defined as the difference between the LSP with the longest round trip time and the LSP with the shortest round trip time. Take Figs. 6 and 7 as examples, and suppose that the round trip times of LSP_B, LSP_C, and LSP_D are 28 ms, 37 ms, and 25 ms, respectively. Thus, the range of round trip time is 12 ms (=37 ms − 25 ms).

When the range of round trip time is greater than a pre-defined threshold, say τ_t, the range is too large. As a result, not all the round trip times are similar. In this case, the procedure to dynamically adjust the traffic of each LSP is started.

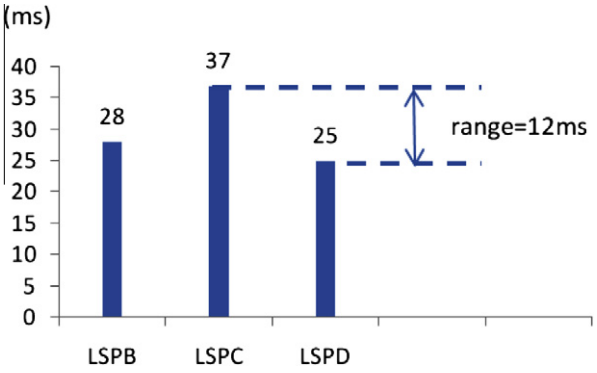

Fig. 7. The range of round trip time.

To adjust the traffic of each LSP, any LSP with heavy traffic must release traffic to LSPs with light traffic. In this study, the traffic carried by the $LSP_j$is heavy when the round trip time $(t_j + d_j)$ is greater than M_t. Therefore, the LSP_jmust release traffic to shorten its round trip time. On the other hand, the LSP_j traffic is small when the round trip time $(t_j + d_j)$ is less than M_t. Therefore, the LSP_jmust receive traffic to increase its round trip time. Note that the released or received amount of traffic should be close to zero if the round trip time $(t_j + d_j)$ is close to M_t. LSPs with heavy traffic and light traffic are called heavy LSPs and light LSPs, respectively.

For any heavy LSP_j, the source calculates the amount of traffic the LSP_jmust release using the formula (10). This formula is based on the round trip time $(t_j + d_j)$, M_t and a specific parameter α, called the tuning factor, which is used to control the amount of released traffic. The amount of released traffic is proportional to the difference between the round trip time $(t_j + d_j)$ and M_t.

$$\left\{\alpha \times \frac{(t_j + d_j) - M_t}{M_t} \times \left[\frac{AB(LSP_j)}{\sum_{i=1}^{N} AB(LSP_i)}\right] \times b_r\right\} \text{Mb}. \quad (10)$$

Suppose that the amount of traffic released by all heavy LSP_js is R_t Mb, and there are x light LSPs. In this case, the source allocates R_t to all the x light LSPs. The first step in this allocation process is to calculate the amount of traffic allocated to the light LSP with the longest round trip time. The next step is to take this allocated traffic as the basis of calculating the traffic allocated to other light LSPs. The traffic allocated to other light LSPs is based on the proportion of each individual round trip time to the longest round trip time. For example, suppose that LSP_e, $1 \leqslant e \leqslant x$, is the longest round trip time $(t_e + d_e)$ among x light LSPs. The source then allocates the following traffic to the LSP_e

$$\frac{R_t}{\sum_{i=1}^{N}\left(\frac{t_e + d_e}{t_i + d_i}\right)} \text{Mb}. \quad (11)$$

For any LSP_y, $1 \leqslant y \leqslant x$, $y \neq e$, the traffic allocated to the LSP_y is

$$\left(\frac{t_e + d_e}{t_y + d_y}\right) \frac{R_t}{\sum_{i=1}^{N}\left(\frac{t_e + d_e}{t_i + d_i}\right)} \text{Mb}. \quad (12)$$

Since a LSP has a smaller maximum available bandwidth when it has a longer round trip time, the traffic allocated to any LSP_y is based on the inverse of its round trip time $(=1/(t_y + d_y))$. As mentioned above, the source takes the LSP_e with longest round trip time $(t_e + d_e)$ as a base to allocate traffic; therefore, the allocated amount of traffic to any LSP_y is $[1/(t_y + d_y)]/[1/(t_e + d_e)]$ times the allocated amount of traffic to the LSP_e. That is, the allocated amount of traffic to any LSP_y is $[(t_e + d_e)/(t_y + d_y)]$ times the allocated amount of traffic

to the LSP_e (as shown in formula (12)). Note that the sum of individually allocated traffic of x light LSPs must equal the traffic released from the heavy LSP_j. That is,

$$\left(\frac{t_e+d_e}{t_1+d_1}\right)\frac{R_t}{\sum_{i=1}^{N}\left(\frac{t_e+d_e}{t_i+d_i}\right)}+\left(\frac{t_e+d_e}{t_2+d_2}\right)\frac{R_t}{\sum_{i=1}^{N}\left(\frac{t_e+d_e}{t_i+d_i}\right)}+\cdots+\left(\frac{t_e+d_e}{t_x+d_x}\right)\frac{R_t}{\sum_{i=1}^{N}\left(\frac{t_e+d_e}{t_i+d_i}\right)}=R_t. \quad (13)$$

3.4. The discussion of dynamic adjustment

Normally, a LSP that releases traffic will have a shorter round trip time, and a LSP that receives the released traffic will have a longer round trip time. As a result, all the round trip times will be similar. However, one main issue with this design is whether or not the dynamic adjustment will cause the network iteratively adjust and become unstable. For example, suppose the range of round trip time is D and D is greater than the τ_t. *Step 2* changes the value of D to D'. When D' is larger than D, the dynamic adjustment is invalid since the range is too large. Therefore, the dynamic adjustment must be repeated. If D' is always larger than D, the network will be iteratively adjusted and become unstable.

To overcome the possible problem, the tuning factor, say α, is used to control the amount of released traffic. As mentioned above, if D' is larger than D after *Step 2* is complete, this adjustment is invalid. In this case, the traffic carried on each LSP is not updated and remains unchanged. α times its original value by a constant value r ($\alpha = \alpha \times r$) and *Step 2* is repeated until D' is less than or equal to D ($D' \leqslant D$). If D' is less than D and still larger than τ_t, the traffic carried on each LSP is updated, α is reset to its initial value, and *Step 2* is repeated. Otherwise, the network is completely adjusted and all the round trip times will be similar. Fig. 8 shows the dynamic adjustment flow chart (assuming that the initial value of α is 1).

Fig. 9 shows the dynamic adjustment procedures. Let LSPS be the set of selected LSPs, tfc be a function from the LSPS to the real numbers to denote the traffic assigned to each LSP, and rtt be a function from the LSPS to the real numbers to denote the round trip time of each LSP. Also, let TFC be the type of functions from the LSPS to the real numbers. The main purpose of the dynamic adjustment procedures is to find a TFC function that performs the dynamic adjustment on each selected LSP.

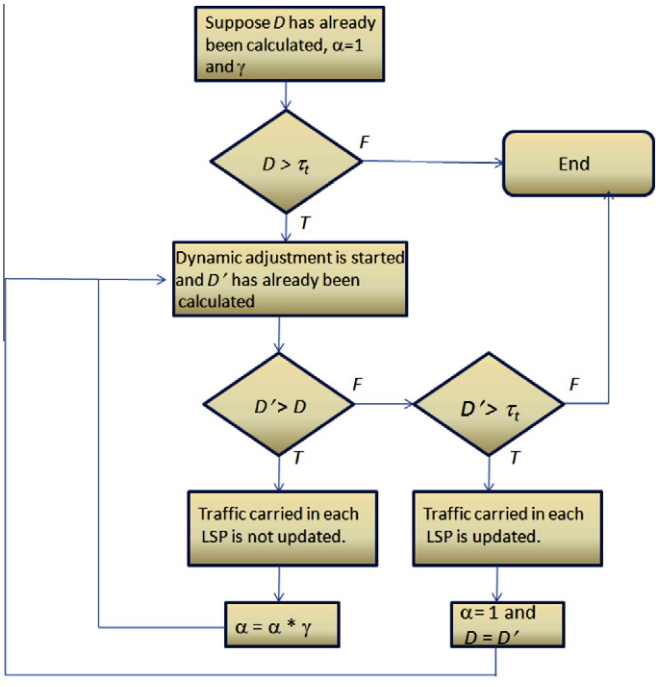

Fig. 8. The dynamic adjustment flow chart.

3.5. Description of the example of allocating traffic to a LSP

Taking Fig. 6 as an example, source A first calculates that the maximum available bandwidth of LSP_B is $AB(LSP_B) = min\{8,5,8\} = 5$ Mbps; the maximum available bandwidth of LSP_C is $AB(LSP_C) = min\{7,8,7\} = 7$ Mbps; the maximum available bandwidth of LSP_D is $AB(LSP_D) = min\{5,5,8,3\} = 3$ Mbps. Source A then uses the maximum available bandwidth of LSP_B, LSP_C, and LSP_D to calculate their individual initial traffic. The initial traffic of LSP_B is $\left[\frac{5}{5+7+3}\right] \times b_r$ Mb; the initial traffic of LSP_C is $\left[\frac{7}{5+7+3}\right] \times b_r$ Mb; the initial traffic of LSP_D is $\left[\frac{3}{5+7+3}\right] \times b_r$ Mb.

Suppose that τ_t equals 1 ms, α equals 1, and the round trip times of LSP_B, LSP_C, and LSP_D are 28 ms, 37 ms, and 25 ms, respectively. M_t is 30 ms after calculation. The range of the round trip time is 12 ms (=37 ms − 25 ms), which is greater than τ_t, and only the round trip time of LSP_C is longer than M_t. Thus, only LSP_C needs to release its traffic. Source A releases part of the LSP_C traffic and allocates it to LSP_B and LSP_D. Since the initial traffic of LSP_C is $\left[\frac{7}{15}\right] \times b_r$ Mb, the LSP_C releases $1 \times \left(\frac{37-30}{30}\right) \times \left(\frac{7}{15} \times b_r\right)$ Mb of traffic and allocates it to the LSP_B and LSP_D. Since the LSP_B has a longer round trip time than the LSP_D, the following traffic may be increased for LSP_B

$$\left(\frac{\frac{7}{30}\times\frac{7}{15}\times b_r}{\frac{28}{28}+\frac{28}{25}}\right)=\left(\frac{25}{53}\right)\times\left(\frac{7}{30}\right)\times\left(\frac{7}{15}\right)\times b_r \text{ Mb}.$$

The increased traffic of the LSP_B can serve as the basis of calculating the increased traffic of the LSP_D

$$\left(\frac{28}{25}\right)\left(\frac{\frac{7}{30}\times\frac{7}{15}\times b_r}{\frac{28}{28}+\frac{28}{25}}\right)=\left(\frac{28}{53}\right)\times\left(\frac{7}{30}\right)\times\left(\frac{7}{15}\right)\times b_r \text{ Mb}.$$

After the source A calculates and adjusts the traffic of each LSP, the round trip time of the LSP_B and LSP_D may increase slightly, but the round trip time of LSP_C decreases slightly. This process is repeated until D' is less than τ_t, which achieves the goal of decreasing the delay from the source to the destination.

4. Simulation results and analysis

4.1. Simulation environment

A DiffServ-TE network with 20 nodes is randomly distributed in a rectangular coordinate grid. Each node is located at integer coordinates. The number of links between any two nodes is decided randomly, and can range from one to four. The available link bandwidth between nodes is randomly set from 1 Mb to 100 Mb. Two nodes are selected as the source and the destination. A LSP requests 10 Mb of bandwidth, and the total data quantity transmitted by the source is 100 Mb. There are two CTs, CT_0 and CT_1, and they are mapped to AF PHB and EF PHB, respectively. Each CT is assigned to a BC. BC_0 and BC_1 are set at 90% and 70% of the link capacity, respectively. The rest of the link capacity is reserved for best-effort traffic. Assume that CT_0 has a lower priority than CT_1. Thus, CT_0 can use the bandwidth of CT_1 when that bandwidth is available. The initial values of α and r are set at 1 and 0.5, respectively. τ_t is fixed at M_t*5%. Note that the M_t is the original value before adjustment. For example as described in Section 3.5, $\tau_t = 30*5\% = 1.5$ ms.

```
TFC traffic-adjust(LSPS, tfc, rtt, τ_t, α, r){
    Calculate D = max{rtt(p)|p∈LSPS} − min{rtt(p)|p∈LSPS} .
    Calculate M_t=average{rtt(p)|p∈LSPS}.
    Calculate HLSPS={p|p∈LSPS and rtt(p)> M_t}. // HLSPS is the set of
                                                   heavy LSPs.
    Calculate LLSPS={p|p∈LSPS and rtt(p)< M_t}. // LLSPS is the set of light
                                                   LSPs.
    Release traffic carried on each heavy LSP_j , LSP_j ∈ HLSPS according to the
        formula in step 2.
    Reallocate the released traffic to each light LSP_y in LLSPS according to the
        formula in step 2.
    The new amount of traffic and round trip time of each LSP are recorded in tfc′ and
        rtt′, respectively.
    Calculate D′ = max{ rtt′ (p)|p∈LSPS} − min{ rtt′ (p)|p∈LSPS} .
    If D′>D, then we recursively call traffic-adjust(LSPS, tfc, rtt, τ_t, α× r, r) and
        return its result.
    If D′<τ_t, we return tfc′.
    else we recursively call traffic-adjust(LSPS, tfc′, rtt′, τ_t, 1, r) and return its result.
}
```

Fig. 9. The dynamic adjustment procedures.

A TCP-based ping is used to measure the round trip time of an established LSP. The TCP-based ping is achieved in three steps. First, the source sends a SYN packet to the destination. Second, the destination acknowledges the SYN packet with a SYN/ACK packet. Finally, when the source receives the SYN/ACK packet, it finishes the process by sending an ACK to the destination. The round trip time is measured using SYN and SYN/ACK between source and destination. The TCP-based ping is a useful way to record the round trip time of an established LSP without sending any ICMP packets.

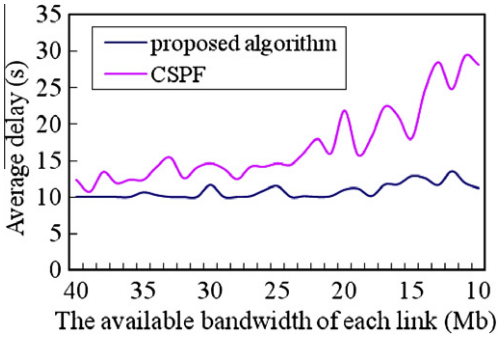

Fig. 10. Comparison of average delay.

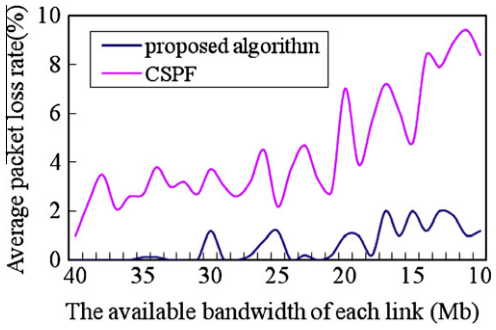

Fig. 11. Comparison of average packet loss rate.

The simulation platform is based on FreeBSD, and the C++ language is used to code the simulation environment.

4.2. Simulation results and discussion

Figs. 10–15 depict the simulation results. Figs. 10–15 compare and evaluate two different approaches, namely the constrained shortest-path-first (CSPF) algorithm and the proposed algorithm. CSPF is enhanced to take into account CT-specific bandwidth as a constraint when computing a path. This simulation considers four performances metrics, namely average delay, average packet loss rate, average throughput, and variance of available bandwidth in all the links. For the simulation results in Figs. 10–14, each data point on the *x*-axis was determined through 20 randomly-produced network topologies, and each data point on the *y*-axis was determined by averaging the results of 20 computations for each algorithm. Fig. 15 shows the number of adjustment for each simulated network.

4.2.1. Comparison of average delay

Fig. 10 shows the result with the *x*-axis displaying the available bandwidth of each link and the *y*-axis displaying the average delay

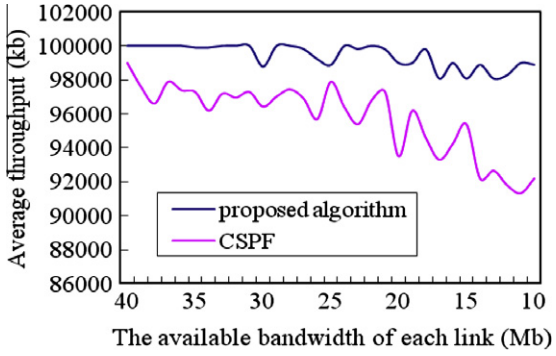

Fig. 12. Comparison of average throughput.

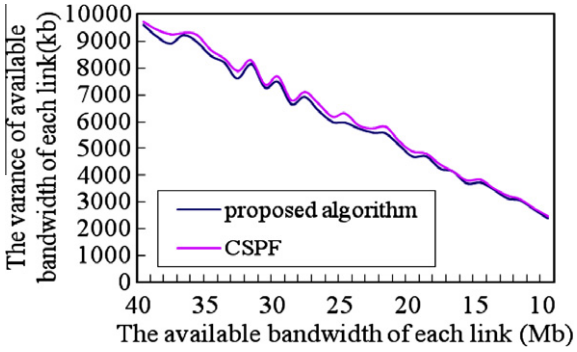

Fig. 13. Comparison of variance of available bandwidth for each link.

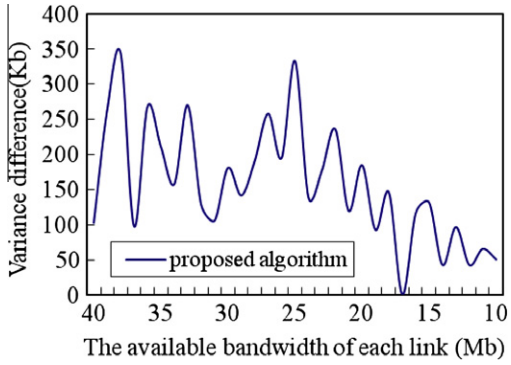

Fig. 14. Comparison of difference in variance of available bandwidth.

from the source to the destination. The number 40 on the *x*-axis means that the available bandwidth of each link can be selected from [1 Mb to 40 Mb] at random, while 10 means that the available bandwidth of each link can be randomly selected from [1 Mb to 10 Mb]. Thus, the available bandwidth of each link decreases along with the value of the *x*-axis. Experimental results show that the average delay difference between the proposed algorithm and CSPF is within 5 s when the source transmits 100 Mb of data and the available bandwidth of network is sufficient. On the contrary, the proposed algorithm produces a much smaller average delay than CSPF when the available bandwidth of network decreases.

Fig. 10 shows that it is not easy for the CSPF to find a path which meets the requested bandwidth when the link's available bandwidth is less than 21 Mb. The average delay determined by the proposed algorithm increases gradually when the available bandwidth of the link is less than 15 Mb. This is primarily because the available bandwidth of each link is small. In addition, the average delay produced by the CSPF is abnormally high when the available bandwidth of the link is 13 Mb and 11 Mb. This is because the maximum available bandwidth from the source to the destination is very small in some paths.

4.2.2. Average packet loss rate comparison

Fig. 11 shows the result with the *y*-axis displaying the average packet loss rate for all selected LSPs from the source to the destination. The packet loss rate is defined as follows:

$$\text{Average Packet loss rate } (\%) = \frac{\text{Number of lost packet}}{\text{Number of transmitted packet}}. \tag{14}$$

Experimental results show that the packet loss rate based on the proposed algorithm is below 2%. The main reason for this small loss rate is that most of the found LSPs meet the requested bandwidth. On the contrary, it is not easy for CSPF to find a LSP when the available network bandwidth becomes smaller. This is why CSPF exhibits a greater increase in average packet loss rate than the proposed algorithm.

4.2.3. Average throughput comparison

Fig. 12 shows the result with the *y*-axis displaying the throughput. Normally, throughput is inversely proportional to the packet loss rate. Fig. 12 shows that both the proposed algorithm and CSPF produce optimal average throughput when the available bandwidth of network is sufficient. However, CSPF exhibits a severe decrease in average throughput when available bandwidth becomes smaller. On the contrary, the average throughput based on the proposed algorithm decrease only slightly when the available bandwidth is smaller. In other words, the proposed algorithm can effectively increase the average throughput of the network.

4.2.4. Comparison of variance of available bandwidth of all links

Figs. 13 and 14 show the results with the *y*-axis displaying the variance of available bandwidth of all the links for a CT *j* on the network. The variance is defined as follows:

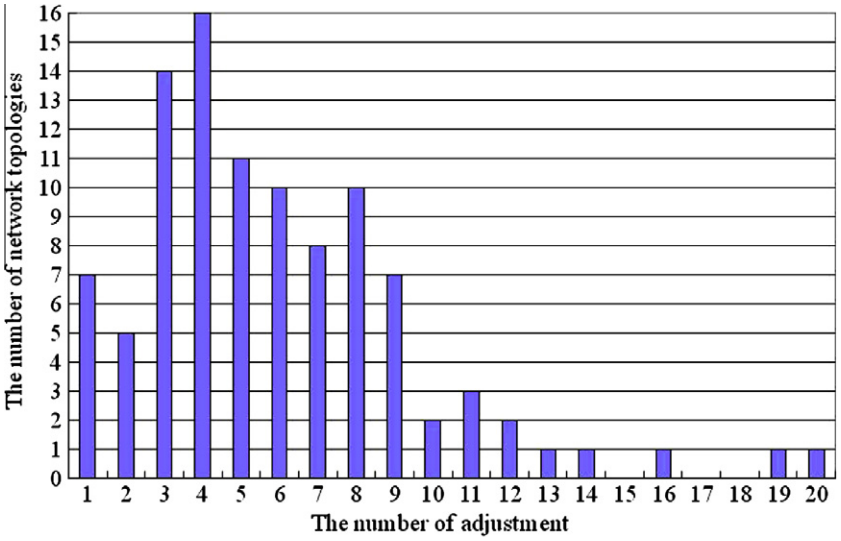

Fig. 15. The number of adjustments.

$$\text{Variance} = \frac{\sqrt{\sum_{i=0}^{|L|}(l_a^i(CTj) - \frac{\sum_{i=0}^{|L|} l_a^i(CTj)}{|L|})^2}}{|L|}. \quad (15)$$

The available bandwidth of each link and the average available bandwidth of all the links are closer when the variance is smaller. In other words, the difference in the available bandwidth of each link in the simulated network is insignificant.

Fig. 13 shows that the proposed algorithm produces a lower variance of available bandwidth than that produced by CSPF. That is, the proposed algorithm can minimize the difference of available bandwidth in all network links. Fig. 14 shows that the greatest difference in variance between the proposed algorithm and the CSPF is approximately 100–350 Kb, which indicates that the proposed algorithm can reduce a great difference in available bandwidth for all network links.

4.2.5. The number of dynamic adjustment

As mentioned in Section 3.4, the tuning factor α controls the amount of released traffic to avoid the problem of a network being iteratively adjusted and becoming unstable. To validate this situation, this study randomly produces 100 DiffServ-TE network topologies, each of which includes 20 nodes. All DiffServ-TE networks are dynamically adjusted, and the number of adjustments in each network is recorded. Fig. 15 shows the results with the x-axis displaying the number of adjustments and the y-axis displaying the number of network topologies. These experimental results show that the average number of adjustment is 5.96, the standard deviation of adjustment is 3.30, the maximum number of adjustments is 20, and the minimum number of adjustments is 0. There are 3 and 1 network topologies whose number of adjustments is 0 and 20, respectively. These results show that 90% of network topologies can be completely adjusted within 10 attempts, 80% of network topologies can be completely adjusted within 8 attempts, and 53% of network topologies can be completely adjusted within 5 attempts.

It is obvious that a network is never iteratively adjusted. Thus, all the produced networks can be completely adjusted within a reasonable number of iterations.

5. Conclusions

The algorithm proposed in this paper uses the available bandwidth of each link as the basis for finding multiple LSPs. After finding feasible LSPs, it was proven that the delay from the source to the destination is at its minimum when the round trip time of each LSP is the same or similar. Moreover, the source adjusts the traffic of each LSP based on the round trip time of each LSP and the average round trip time of all the selected LSPs. This decreases the delay from the source to the destination.

Simulation results clearly demonstrate that the proposed algorithm yields a better average delay, average packet loss rate, average throughput, and available bandwidth variance in each link than those based on the CSPF. In addition, all the simulated networks can be completely adjusted. Therefore, the proposed algorithm is feasible with a higher efficiency.

References

[1] S. Blake et al., An Architecture for Differentiated Services, RFC 2475, December 1998.
[2] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, J. McManus, Requirements for Traffic Engineering Over MPLS, RFC 2702 (Informational), September 1999.
[3] F.L. Faucheur, L. Wu, B. Davie, S. Davari, P. Vaananen, R. Krishnan, P. Cheval, J. Heinanen, RFC 3270: Multi-protocol Label Switching (MPLS) Support for Differentiated Services, May 2002.
[4] F.L. Faucheur, W. Lai, RFC 3564: Requirements for Support of Differentiated Services-Aware MPLS Traffic Engineering, July 2003.
[5] Dongli Zhang, Dan Ionescu, QoS Performance Analysis in Deployment of DiffServ-aware MPLS Traffic Engineering, SNPD 2007, vol. 3, pp. 963–967, July 30 2007–August 1.
[6] Fahad Rafique Dogar, Zartash Afzal Uzmi, Shahab Munir Baqai, CAIP: a restoration routing architecture for DiffServ aware MPLS traffic engineering, DRCN 2005, October 16–19, 2005.
[7] Jaudelice C. de Oliveira, Caterina Scoglio, Ian F. Akyildiz, George Uhl, New preemption policies for DiffServ-aware traffic engineering to minimize rerouting in MPLS networks, IEEE/ACM Transactions on Networking 12 (4) (2004) 733–745.
[8] Tong Shan, Oliver W.W. Yang, Bandwidth management for supporting differentiated-service-aware traffic engineering, IEEE Transactions on Parallel and Distributed System 18 (9) (2007) 1320–1331.
[9] F. Le Faucheur et al., Maximum allocation bandwidth constraints model for diff-Serv-aware MPLS traffic engineering, IETF Internet draft (2004).
[10] F. Le Faucheur et al., Russian Dolls Bandwidth Constraints Model for Diff-Serv-Aware MPLS Traffic Engineering, RFC 4127, June 2005.
[11] J. Ash, Max allocation with reservation bandwidth constraints model for diffserv-aware MPLS traffic engineering and performance comparison, IETF Internet draft (2004).
[12] F. Le Faucheur, Considerations on bandwidth constraint models for DS-TE, IETF, Internet Draft, work in progress, June 2002.
[13] G. Apostolopoulos et al., QoS Routing Mechanisms and OSPF Extensions, RFC 2676, August 1999.
[14] A.S. Fraenkel, D. Lichtenstein, Computing a perfect strategy for n^*n chess requires time exponential in n, Automata, Languages, and Programming, Springer LNCS 115 (1981) 278–293.
[15] Roch Guerin, Ariel Orda, Computing shortest paths for any number of hops, IEEE/ACM Transactions on Networking 10 (5) (2002) 613–620.