A Novel Traffic Engineering Routing Algorithm

Cao Thai Phuong Thanh[1], Ha Hai Nam[2], and Tran Cong Hung2

[1] Saigon University, Vietnam
ctpthanh@sgu.edu.vn
[2] Post & Telecommunications Institute of Technology, Vietnam
namhh@ptit.edu.vn, conghung@ptithcm.edu.vn

Abstract. This paper introduces a traffic engineering routing algorithm that aims to accept as many routing demands as possible on the condition that a certain amount of bandwidth resource is reserved for each accepted demand. The novel idea is to select routes based on not only network states but also information derived from routing data such as probabilities of the ingress egress pairs and usage frequencies of the links. Experiments with respect to acceptance ratio and computation time have been conducted against various test sets. Results indicate that the proposed algorithm has better performance than the existing popular algorithms including Minimum Interference Routing Algorithm (MIRA) and Random Race based Algorithm for Traffic Engineering (RRATE).

Keywords: routing algorithm, traffic engineering, bandwidth guaranteed, history routing data

1 Introduction

There are increasingly more network applications e.g. live stream and online games which require certain quality of service (QoS) guarantees. Beside satisfying the QoS requirements, network providers need optimizing their network performance so that they can effectively fulfil as much customer demand as possible. Therefore, traffic engineering (TE), which manages network activities with dual objectives of QoS satisfaction and network optimization, becomes important. Routing is a powerful technique of TE as it allows for controlling network data flows.

Generally, routing algorithms are categorized into proactive and reactive ones. The former pre-selects paths for routing demands based on fixed network information, while the later uses dynamic network states to establish routes upon receiving demands. As a result, reactive algorithms adapt to traffic demands and have good routing performance [1]. This paper focuses on reactive TE routing with minimum bandwidth criteria - one of the most popular QoS condition. Besides network states (e.g. network topology, link residual bandwidth) which are used in existing solutions [8], the proposed algorithm introduces a novel idea to exploit historical routing data such as request probability and link usage frequency for route selection.

The rest of the paper is organized as follows. Section 2 presents the definition of the TE routing problem and a review of related solutions. The proposed algorithm named Traffic Engineering routing Algorithm with Routing Data (TEARD) is discussed in section 3. Section 4 firstly describes the experimental setup; secondly compares the proposed algorithm with the other popular ones with respect to acceptance ratio of demands and average computation time. Finally, section 5 discusses conclusions and future work.

2 Problem definition & related works

2.1 Problem Definition

A network topology with n nodes and m links is considered. Each link has its own capacity and residual bandwidth at a given time. A routing demand, which requires a path with certain bandwidth from an ingress node to an egress node, is handled by the routing algorithm. The algorithm sequentially processes demands with an assumption that network states such as topology, link bandwidths and ingress-egress (ie) pairs are available. Table 1 lists the mathematical notations used to describe the problem.

Table 1: Problem notations

Symbol	Description
$G(N, L)$	A direct graph presents the network topology $N(\|N\| = n)$ is a set of nodes $L(\|L\| = m)$ is a set of links
IE	A set of all ingress egress pairs (ie)
$c(l)$	Capacity bandwidth of link l
$r(l)$	Residual bandwidth of link l
$d(i, e, b)$	A traffic demand requesting b bandwidth units from an ingress node i to an egress node e.
p_{ie}	A routing path from node i to node e
P_{ie}	A set of all paths from node i to node e

The objective of TE routing algorithm is to maximize the number of accepted demands. Since ingress-egress pairs have commodity integral flows (i.e. links of paths are common), the TE routing problem is NP-hard [5]. Most of dynamic routing algorithms first calculate link weights based on network states then use shortest path algorithms (e.g. Dijkstra or Bellman-Ford) to select the least weighted route. Table 2 generalizes the steps of dynamic routing algorithms.

2.2 Related Works

The simplest and most widely used routing solution is Minimum Hop Algorithm (MHA) [10]. As its name implies, MHA selects paths that have minimum hop

Table 2: General steps of TE routing algorithms

Input	A network graph $G(N, L)$ with necessary information e.g. link bandwidths A traffic demand $d(i, e, b)$
Output	A satisfied bandwidth path from s to d, p_{ie}, toward the optimal goal of maximizing the number of accepted demands Or no route satisfying the demand
General algorithm	1. Calculate link weights $w(l)$ 2. Temporarily remove links that have residual bandwidth less than b 3. Find the least weight path p_{ie}. If found then return p_{ie}, otherwise rejects the demand.

counts for routing. It means the same shortest path is chosen for each ingress egress pair until at least one of its link cannot satisfy bandwidth demands. This static selection scheme results in network bottleneck and under-utilization.

Minimum Interference Routing Algorithm (MIRA) [6] exploited knowledge of ingress egress pairs so that routing paths of one pair interfere as little as possible with paths of the others. The interference is measured by link criticality based on the maxflow-mincut theory [2]. Particularly, when a routing demand arrives, MIRA identifies mincut sets for all of the ingress egress pairs except the one being requested. Links that belong to a mincut are considered critical because if they are used to route data then maxflow of the corresponding ingress egress pair is decreased. The critical links are considered interference with other ie pairs. Therefore, MIRA sets link weights as the number of times the links are critical. This idea allows MIRA to accept noticeably more requests than MHA. However, computation time of MIRA is also significantly longer than MHA's due to the maxflow-mincut calculation.

Random-based Routing Algorithm for Traffic Engineering (RRATE) [7] employed a machine learning technique - random race - to improve the MIRA computation time. Each ie pair of RRATE has two stages: learning and post-learning. In the learning stage, upon receiving a routing demand, costs of k pre-selected paths are calculated based on maxflow-mincut criticality and link residual bandwidths. The least cost path is then selected and its racing reward is accumulated. The race between those paths in term of reward values will finish if there is a path whose reward reaches a pre-defined value N. After that, the corresponding ie pair moves into the post-learning stage and costs are not calculated any more. Alternatively, demands of that pair will be routed by the path having maximum racing value and its links satisfy bandwidth constraint. The post-learning stage reduces the computing time of RRATE compared to MIRA.

Additionally, [3] proposed another two-phase routing algorithm called Bandwidth Guarantee with Low Complexity (BGLC). In BGLC, link criticality is considered as occurrences of links in all of possible paths. It means the more paths a link belongs to, the more critical it is (i.e. the more interference it causes). Because BGLC uses only network topology to calculate criticality, this

process is done before routing demands arrive in an offline phase. On the other hand, the online phase is the process to identify route for arriving demands. The online phase of BGLC simply divides criticalities by residual bandwidths to have link weights before applying Dijkstra. BGLC has low computation time (low complexity as its name) because link criticality is calculated in the offline phase and recalculation is needed only when the network topology changes.

3 Proposed Algorithm

This section describes the proposed algorithm named Traffic Engineering routing Algorithm with Routing Data (TEARD). In order to make the algorithm adaptive to routing demands, link weights are calculated from not only network topology and residual bandwidths but also from routing data.

Firstly, ingress egress pairs are considered. For each ie pair, link criticality is determined as the occurrence rate of the link in all paths of the pair. For instance, if a pair ie has 5 paths and a link l appears in 3 of them, then the criticality of link l for the pair ie is $crit_{ie}(l) = 3/5$. This criticality is calculated in the offline phase because it is determined by the network topology only. Furthermore in the online phase, the link criticality for each pair is multiplied by probability of the pair being requested. This modification makes the critical values dynamically adapt to actual routing requests. For example, when link l appears in many paths of pair ie, the critical value $crit_{ie}(l)$ is high and the algorithm will try to avoid this link. However, if ie is infrequently requested, such avoidance may have negative effect on the overall routing performance. In this case, the multiplication by low probability of ie helps reducing $crit_{ie}(l)$. The link criticality for ie pairs is calculated as follows:

$$crit_1(l) = \sum_{ie \in IE} (crit_{ie}(l) * prob(ie) * 100) \tag{1}$$

Where $crit_{ie}(l)$ is the criticality of link l for the ingress egress pair ie.

$$crit_{ie}(l) = \frac{\sum_{p_{ie} \in P_{ie}} v_l}{|P_{ie}|} \tag{2}$$

$$v_l = \begin{cases} 0 & \text{if } l \notin p_{ie} \\ 1 & \text{if } l \in p_{ie} \end{cases}$$

$|P_{ie}|$ is the number of all paths from i to e

$prob(ie)$ is the probability that the pair ie is requested. In this paper, $prob(ie)$ is statistically calculated from ie pairs of historical routing requests.

Secondly, bandwidth information, especially residual bandwidth, plays an important role in route selection. In order to prevent bottleneck, the less residual bandwidth a link has, the less likely it should be chosen for new paths i.e. its critical value should be high. Similarly, a link that has used large of its bandwidth

also should not been chosen. As a result, dynamic bandwidth states are taken into link criticality as follow:

$$crit_2(l) = \frac{used(l)}{r(l)} = \frac{c(l) - r(l)}{r(l)} * 100 \quad (3)$$

Thirdly, usage frequencies of links within the routing process also contribute to critical value as shown in equation (4). The idea is to balance link selection by setting higher value to a link that has been selected more times.

$$crit_3(l) = \frac{|P_l|}{|P|} * 100 \quad (4)$$

Where $|P|$ is the total number of established paths and $|P_l|$ is the number of established paths containing link l.

Finally, equation (5) combines three above critical parts into the TE weight of link l; and table 3 describes steps of the proposed algorithm TEARD.

$$w(l) = k_1.crit_1(l) + k_2.crit_2(l) + k_3.crit_3(l) \quad (5)$$
$$k_1, k_2, k_3 \text{ are moderation parameters}$$
$$0 < k_1, k_2, k_3 < 1 \text{ and } k_1 + k_2 + k_3 = 1$$

Table 3: The Traffic Engineering Routing Algorithm using Routing Data (TEARD)

Input	A network $G(N, L)$ Traffic demands $d(s, d, b)$
Output	A selected path p_{sd} Or no path
Offline phase	1. For each pair ie, calculate $crit_{ie}(l)$ of every link using equation (2)
Online phase	For each $d(s, d, b)$: 1. For all links l in L, calculate $w(l)$ using equation (5) 2. Temporarily remove links whose $r(l) < b$ 3. Apply Dijkstra algorithm to find the least weighted path from s to d If found: return the path p_{sd} and update necessary information. Else: reject the demand.

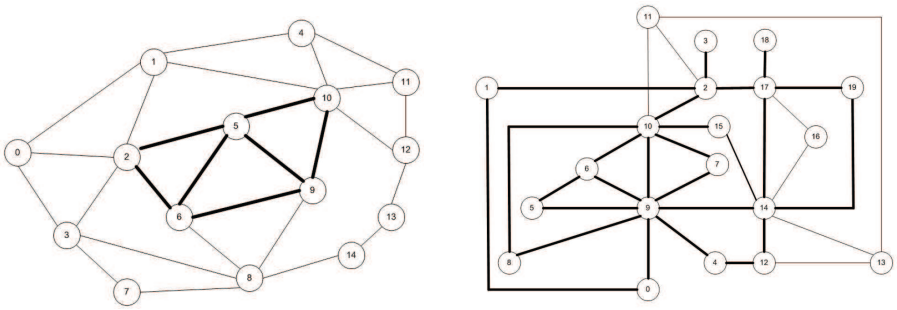
4 Performance Evaluation

The proposed algorithm TEARD is evaluated against three popular TE routing algorithms: MHA, MIRA, and RRATE. Two performance metrics are considered: acceptance ratio and average computation time. The acceptance ratio reflects the percentage of demands for which algorithms are able to establish bandwidth-reserved path. According to the problem definition, routing algorithms aim to maximize this metric. Meanwhile, the computation time is the average time needed for algorithms to handle routing demands. The average time is computed in the online phase and should be minimized.

4.1 Experimental Setup

Algorithms are evaluated in two routing scenarios: static and dynamic ones. In static scenario, routing demands arrive at the same rate. If the arrived demands are accepted they will statically hold bandwidth resource along their paths. On the other hand, paths in dynamic scenario will release bandwidths after a certain holding time. Arrival time of dynamic demands is randomly distributed by the Poisson distribution of mean λ requests per time unit, and their holding time is distributed by the Exponential with mean μ time units.

Additionally, figure 1 shows two experimental network topologies: MIRA which is adapted from previous TE routing work e.g. [6], [3], [4]; and CESNET, which is inherited from a real MPLS topology [9]. Both topologies contain bidirectional links with two types of capacity. MIRA has 4800 and 1200 bandwidth units respectively for thicker and thinner links in figure 1(a). Similarly, capacities of CESNET topology (figure 1(b)) are 10000 and 1000 bandwidth units. Moreover, for MIRA topology, source and destination nodes are selected between four ingress-egress pairs (0, 12), (4, 8), (3, 1), and (4, 14), and bandwidths are randomly requested between 10, 20, 30, and 40 units. Meanwhile, ingress egress pairs and demanded bandwidths of CESNET topology are (0, 18), (1, 11), (3, 16), (4, 7), (5, 13), (6, 19), (15, 0), (19, 8) and 40, 80, 120, 160 units, respectively.



(a) MIRA topology (b) CESNET topology

Fig. 1: Simulation network topologies

4.2 Performance Results

Table 4 presents performance results with following parameters:

- The static scenario has total of 1000 demands while the dynamic one has 2000.
- MIRA topology: four ie pairs are demanded by the approximate rates of 10%, 20%, 30%, and 40%. Dynamic demands have the Poisson arrival time $\lambda = 40$ and the Exponential holding time $\mu = 20$.

– CESNET topology: the probabilities of eight ie pairs are 5%, 10%, 15%, and 20% (duplicated). Dynamic demands have $\lambda = 80$ and $\mu = 30$. Those dynamic time parameters are selected based on topology sizes where the bigger network has more demands and longer holding time.
– RRATE takes moderation parameters $k_1 = k_2 = 0.5$, number of pre-selected path $k = 30$, and racing threshold $N = 15$. Those parameter values are chosen for they give the best results after trying different values.
– TEARD takes moderation parameters $k_1 = 0.3$, $k_2 = 0.4$, $k_3 = 0.3$. Values of those parameters are discussed in the next subsection.

Table 4: Results of acceptance ratio (in percent) - computation time (in miliseconds) subject to number of requests (NoR)

(a) Static scenario on MIRA

NoD	MHA	MIRA	RRATE	TEARD
100	100- 0.19	100- 2.23	100- 3.05	100- 0.29
500	94.40- 0.10	94.40- 2.16	94.40- 1.91	94.60- 0.24
1000	56.90- 0.04	58.70- 1.02	59.30-0.31	61.20- 0.11

(b) Static scenario on CESNET

NoD	MHA	MIRA	RRATE	TEARD
100	100- 0.20	100- 8.36	100- 10.42	100- 0.54
500	79.60- 0.12	79.60- 8.23	79.00- 8.78	80.00- 0.41
1000	48.10- 0.05	48.10- 3.79	46.70-3.55	49.70- 0.19

(c) Dynamic scenario on MIRA

NoD	MHA	MIRA	RRATE	TEARD
100	100- 0.22	100- 2.36	100- 3.10	100- 0.30
1000	71.30- 0.09	72.80- 2.22	74.60- 0.62	74.30- 0.23
2000	63.60- 0.08	64.70- 2.40	64.85- 0.34	67.10 - 0.23

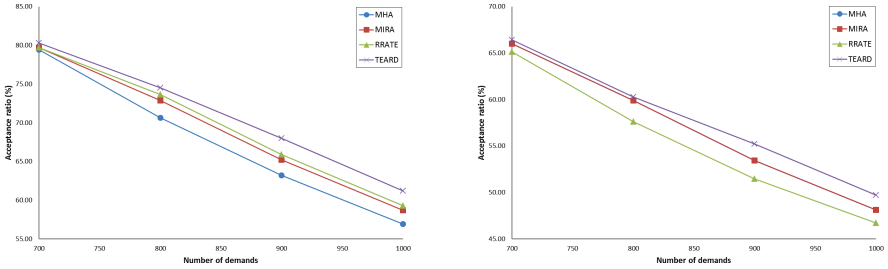
(d) Dynamic scenario on CESNET

NoD	MHA	MIRA	RRATE	TEARD
100	100- 0.21	100- 8.38	100- 10.21	100- 0.44
1000	62.70- 0.11	62.60- 7.22	62.90- 5.34	64.10- 0.38
2000	42.70- 0.10	42.30- 6.74	42.95- 3.84	43.75 - 0.37

Figure 2 plots acceptance ratios versus last 300 demands in static scenario. It is clearly observable that TEARD accepts more demands than the other algorithms. For instance, with MIRA network after 1000 demands, acceptance ratio of TEARD (61.2%) is 4.3% higher than MHA's (56.9%). Similarly, on CESNET network, results of MHA and MIRA (which are identical) are lower than those of TEARD (48.1% compare to 49.7%). Additionally, the advantage of TEARD is also observed from figure 3 that presents accepted results of the 1000^{th} to the 2000^{th} dynamic demands. Figure 3(b), for example, shows that TEARD accepts most demands, followed by RRATE, MHA; and MIRA accepts the least.

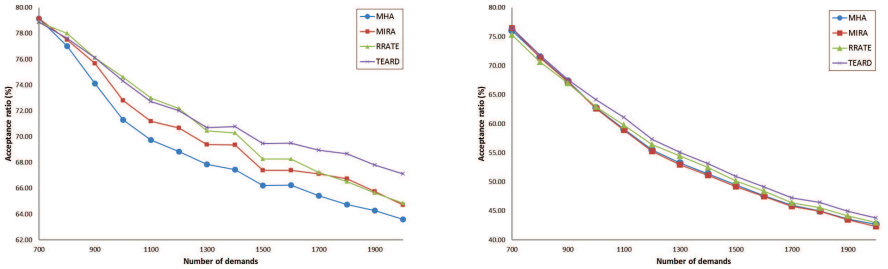
According to literature, interference idea helps dynamic routing algorithms such as MIRA and RRATE outperform MHA in term of acceptance ratio. It is demonstrated by experiments with MIRA topology where MHA always has the lowest results. However, in many experiments with CESNET topology, MHA accepts more demands than MIRA and RRATE. It is because CESNET topology is inherited from a real network [9] and is optimized for the shortest path algorithm. Consequently, the improvements of TEARD over MHA on CESNET are meaningful despite small increases in the metric values (figure 2(b), 3(b)).

On the other hand, average computation time is investigated. Obviously, MHA is the fastest algorithm. Whereas, MIRA is the slowest due to the maxflow-mincut calculation. It is clearly shown in figure 4 where computation time of MIRA is higher than MHA about 25 times (figure 4(a)) and even 67 times



(a) Results on MIRA topology (b) Results on CESNET topology

Fig. 2: Compare of acceptance ratios in static scenario



(a) Results on MIRA topology (b) Results on CESNET topology

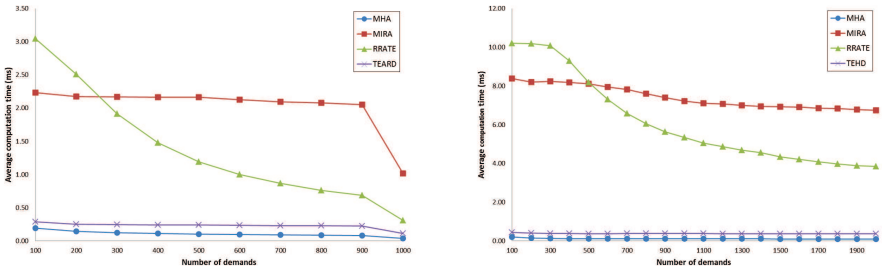
Fig. 3: Compare of acceptance ratios in dynamic scenario

(figure 4(b)). Figure 4 also shows that RRATE requires long time to handle first several hundreds of demands, but after this learning stage the time of RRATE considerably decreases. Nevertheless, TEARD achieves much lower computation time than both MIRA and RRATE. Particularly, figure 4(a) shows that TEARD is 9 times faster than MIRA, and 3 times faster than RRATE. In figure 4(b), the average computation time of TEARD is only 0.37 ms compared with 6.74 ms of MIRA (18 times lower) and 3.84 ms of RRATE (10 times lower).

In order to obtain confident evaluation, experiments are repeated with different sets of random demands as well as different parameter values. The above results are examples of various conducted tests. Experimental results indicate that TEARD achieves not only higher acceptance ratio but also lower average computation time than the other routing algorithms

4.3 Effect of the Parameters $k1$, $k2$, and $k3$

This subsection investigates how the moderation parameters affect performance of TEARD. When changing k_s, link weights are changed that leads to different acceptance ratios, but the computation time is not affected. 36 sets of $k1$, $k2$, and $k3$ values, for each trial, are generated on the condition $0 < k_1, k_2, k_3 < 1$ and $k_1 + k_2 + k_3 = 1$. Table 5 shows the 3 highest (top 3 rows) and the 3 lowest



(a) Results of static scenario on MIRA topology

(b) Results of dynamic scenario on CESNET topology

Fig. 4: Compare of average computation times

acceptance ratios (bottom 3 rows) of the experiments presented in subsection 4.2.

Table 5: Variation of the acceptance ratio with variation of $k1$, $k2$, $k3$

(a) Results of dynamic scenario on MIRA network

k_1	k_2	k_3	%
0.1	0.3	0.6	67.20
0.3	0.4	0.3	67.10
0.5	0.2	0.3	67.10
...			
0.1	0.8	0.1	65.55
0.1	0.7	0.2	65.45
0.7	0.1	0.2	65.45

(b) Results of static scenario on CESNET network

k_1	k_2	k_3	%
0.2	0.2	0.6	50.10
0.3	0.2	0.5	50.00
0.4	0.3	0.3	50.00
...			
0.2	0.1	0.7	49.60
0.4	0.1	0.5	49.50
0.6	0.2	0.2	49.40

As observed from table 5, variation of the moderation parameters does not significantly increase or decrease the acceptance ratio. Moreover, in many experiments, even the worst result of TEARD is better than that of other algorithms. The bottom row of table 5(b) is an example in which acceptance rate of TEARD (49.4%) is higher than MHA, MIRA (48.1%) and RRATE (46.7%). As a consequence, moderation values should be approximately equally (e.g. $k_1 = 0.3, k_2 = 0.4, k_3 = 0.3$ in subsection 4.2) so that all three critical parts have impacts to link weights.

5 Conclusion

This paper introduces TEARD, a dynamic traffic engineering routing algorithm with minimum bandwidth constraint. The objective of dynamic routing is to maximize acceptance ratio so algorithms should be adaptive to routing demands. Therefore, TEARD considers not only network information but also data from historical routes. Evaluation has been conducted with different routing demands

as well as different network topologies and routing scenarios. Experimental results illustrate that TEARD accepts more demands in less computation time than other popular TE routing algorithms.

One of directions for future work is the influence of link failure on routing performance. Extension of QoS parameters in the proposed routing algorithm is also possibly potential for future exploitation.

References

1. Agrawal, H., Jennings, A.: Evaluation of routing with robustness to the variation in traffic demand. Journal of Network and Systems Management 19(4), 513–528 (2011)
2. Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network flows : Theory, algorithms, and applications. Prentice-Hall, Englewood Cliffs, N.J. (1993)
3. Alidadi, A., Mahdavi, M., Hashmi, M.R.: A new low-complexity QoS routing algorithm for MPLS traffic engineering. In: 2009 IEEE 9th Malaysia International Conference on Communications (MICC). pp. 205–210. Kuala Lumpur, Malaysia (2009)
4. Boutaba, R., Szeto, W., Iraqi, Y.: DORA: efficient routing for MPLS traffic engineering. Journal of Network and Systems Management 10, 309–325 (2002), 10.1023/A:1019810526535
5. Even, S., Itai, A., Shamir, A.: On the complexity of time table and multi-commodity flow problems. In: Proceeding SFCS '75 Proceedings of the 16th Annual Symposium on Foundations of Computer Science. pp. 184–193. IEEE (1975)
6. Kar, K., Kodialam, M., Lakshman, T.: Minimum interference routing of bandwidth guaranteed tunnels with MPLS traffic engineering applications. IEEE Journal on Selected Areas in Communications 18(12), 2566–2579 (2000)
7. Oommen, B.J., Misra, S., Granmo, O.: Routing Bandwidth-Guaranteed paths in MPLS traffic engineering: A multiple race track learning approach. IEEE Transactions on Computers 56(7), 959–976 (2007)
8. Thanh, C.T.P., Hung, T.C.: A study of bandwidth guaranteed routing algorithms for traffic engineering. In: Nguyen, N.T., Trawiński, B., Katarzyniak, R., Jo, G.S. (eds.) Advanced Methods for Computational Collective Intelligence, Studies in Computational Intelligence, vol. 457, pp. 313–322. Springer Berlin Heidelberg (2013)
9. Václav Novák, Petr Adamec, Pavel Šmrha, Josef Verich: CESNET2 IP/MPLS backbone network design and deployment in 2008 (2008), http://www.cesnet.cz/doc/techzpravy/2008/CESNET2-ip-mpls-backbone-in-2008/
10. Wang, Z., Crowcroft, J.: Quality-of-service routing for supporting multimedia applications. IEEE Journal on Selected Areas in Communications 14(7), 1228–1234 (1996)