A Study of Bandwidth Guaranteed Routing Algorithms for Traffic Engineering

Phuong-Thanh Cao Thai[1] and Hung Tran Cong[2]

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

Abstract. Due to the fast growths of computer networks, traffic engineering (TE) which tries to satisfy both quality of services and resource utilization requirements is an important research area. Among TE mechanisms, routing algorithm – a strategy to select paths for traffic – plays a crucial role and there have been many TE routing proposals. This paper gives a thorough study of these algorithms by presenting their key ideas and mathematical descriptions, and then by various experiments so to analyse their performances with different metrics. The paper also discusses the trend of routing algorithms and future research directions.

Keywords: bandwidth guaranteed routing, traffic engineering.

1 Introduction

Besides traditional network services, next generation network applications such as voice over IP, video on demand and web games require certain quality of services (e.g. minimum available bandwidth guaranteed) described as customer service-level agreements (SLAs). Network providers try not only to satisfy those SLAs but also to optimize the network resources for profits. As a result, traffic engineering which is defined as techniques to manage traffic flows through networks with the joint goals of service performance and resource optimization has attracted much attention. One of the most important TE implementation is reactive routing that dynamically selects routes for data flows based on states of the network in order to balance traffic loads as well as conform to quality of services (QoS) requirements. Although there are several types of QoS criteria such as bandwidth, delay, and loss ratio, most academic works consider bandwidth as a primary constraint because others can be efficiently converted into bandwidth demand [1]. As a consequence, this paper focuses on reactive bandwidth guaranteed TE routing algorithms.

Specifically, key ideas of those algorithms are reviewed and, for the first time, their mathematical formulations are presented in the unified manner. Moreover, the performances are compared and analysed by various experiments with two metrics: accepted ratio and calculation time. The rest of the paper is organized as

N.T. Nguyen et al. (Eds.): *Adv. Methods for Comput. Collective Intelligence*, SCI 457, pp. 313–322.
DOI: 10.1007/978-3-642-34300-1_30 © Springer-Verlag Berlin Heidelberg 2013

follow. In section 2, after definitions and notations, bandwidth guaranteed routing algorithms are classified into three categories: single pair aware, minimum interference and machine learning. Section 3 presents simulated experiments and analysis. Finally, section 4 presents conclusions and the future research directions.

2 Routing Algorithms

2.1 Problem Definition and Notations

A network topology with n node and m links is considered. Each link has its own capacity and residual bandwidth at a given time. Traffic demands, which require certain bandwidths from ingress to egress nodes, are handled by the routing algorithm so as to maximize the number of accepted demands. Table 1 lists the mathematical notations.

Table 1. Notations used in formulations

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
$c(l)$	Capacity bandwidth of link l
$r(l)$	Residual bandwidth of link l
$d(s, d, b)$	A traffic demand from ingress node s to egress node d with required bandwidth b
D	A set of all ingress-egress pairs
p_{sd}	A routing path from s to d
P_{sd}	A set of all paths from s to d

The goal of routing algorithms is:

$$\begin{cases} \text{Maximize number of satisfied demands} \\ \text{subject to} \\ \text{Find } p_{sd} \text{ for } d(s,d,b)/\forall l \in p_{sd} : r(l) \geq b \end{cases} \quad (1)$$

Since ingress-egress pairs have commodity integral flows (i.e. common links as well as sequences of links), the routing problem summarized in (1) is NP-hard [2]. Moreover, the algorithms can be generalized as table 2.

2.2 Single Pair Aware Algorithms

The simplest solution is Minimum Hop Algorithm (MHA) where all weights are statically equal to 1. Dijkstra or Bellman-Ford algorithm is applied to find least hop counts paths. It means shortest paths are always selected and their links are quickly congested whereas others underutilize. To improve this property,

Table 2. General TE routing algorithm

Input	A network graph $G(N, L)$ with sets of links and residual bandwidths A traffic demand $d(s, d, b)$
Output	A satisfied bandwidth path from s to d, p_{sd}, toward the optimal goal in (1) Or no route satisfying the request
General algorithm	1. Calculate all link weights $w(l)$ 2. Remove links that have residual bandwidth less than b 3. Find the least cost path p_{sd} based on weights of remaining links

Widest Shortest Path algorithm (WSP) [3] selects paths having maximum bottleneck link bandwidth between the shortest equal-length ones. Additionally, Bandwidth Constrained Routing Algorithm (BCRA) [4] combines three parameters (link capacity, residual bandwidth, and path length) to calculate link weights.

$$w(l) = cost(l) * load(l) + 1$$

$$cost(l) = \frac{10^8}{C(l)}$$

$$load(l) = \frac{c(l) - r(l)}{c(l)}$$

Heavy weight values mean low capacities and/or heavy loads so those links are likely avoided, whereas hop count is reflected by the addition of 1. Experiments confirm that such combination of properties improves the routing performance.

Despite of using different network parameters, the above algorithms are classified as single-pair-aware because they greedily find good routes for the being demanded ingress-egress pair but not consider other ones. It might noticeably affect future requests. As a result, minimum interference solutions are proposed.

2.3 Minimum Interference Algorithms

The first Minimum Interference Routing Algorithm (MIRA) uses the maxflow-mincut characteristic [1]. When a routing request arrives, a maxflow residual graph is computed to determine a mincut set for each other ingress-egress pair. According to the maxflow-mincut theory [5], a bandwidth decrease of a link belonging to the mincut set will lead to the same amount reduction of the corresponding maxflow value. It means if such links are selected to route the current request, they will interfere with future demands of other pairs (i.e. narrow maxflow of them). MIRA defines those links as critical and assign more weights to them.

$$w_{sd}(l) = \sum_{(s',d') \in D \setminus (s,d)} \alpha_{s'd'} \quad \text{if } l \text{ is critical of } (s', d')$$

$\alpha_{s'd'}$ reflects the importance of the pair (s', d')

MIRA computes link weights from all ingress-egress pairs except the current demanded pair. Such current pair is excluded because the algorithm aims to prevent interference with the other ones. When all pairs are treated equally ($\alpha = 1$), a weight is the frequency of link's criticality, and the more critical a link is, the less it is chosen. Evaluations prove that MIRA outperform MHA in term of accepted ratio. Inspired by MIRA, different minimum interference algorithms are proposed.

Authors of NewMIRA [6] comment that MIRA only takes links of mincut sets into account, whereas all links that put up maxflows might affect future demands. Therefore, NewMIRA calculates a link's criticality by its load contribution to maxflow and the residual bandwidth.

$$w_{sd}(l) = \sum_{(s',d') \in D \setminus (s,d)} \frac{f_l^{s'd'}}{\theta^{s'd'}.r(l)}$$

$\theta^{s'd'}$ is the maxflow of the pair (s', d')

$f_l^{s'd'}$ is the subflow of $\theta^{s'd'}$ through link l

High interfering links are ones that largely contribute to maxflows and/or have small remaining bandwidths. Similar to MIRA, the NewMIRA also excludes the current pair (s,d) from weight calculation.

Dynamic Online Routing Algorithm (DORA) [7] does not use maxflows for interference but use the numbers of time links appearing in disjointed routing paths. Specifically, a link criticality of one (s,d) is decreased if that link is a part of any path from s to d, and increased if it belongs to paths of the other pairs.

$$criticality_sd(l) = \sum_{(i,j) \in D} \sum_{p_{ij} \in dP_{ij}} v_l$$

$$v_l = \begin{cases} 0 & \text{if } l \notin p_{ij} \\ -1 & \text{if } l \in p_{ij} \text{ and } ij \equiv sd \\ 1 & \text{if } l \in p_{ij} \text{ and } ij \not\equiv sd \end{cases}$$

dP_{ij} is the disjointed path set of the pair (i, j)

Realizing that the criticalities are computed solely by the network topology (i.e. by identification of disjointed paths); those values are prior-determined and only recalculated once the topology changes. This calculation is called the offline phase to differ from the online reactive routing phase. When a routing request arrives (i.e. the online phase), weights are formed from the corresponding criticalities and the current bandwidths. Because the interference is pre-determined, DORA selects routes more quickly than the above algorithms, especially when there are many ingress-egress pairs.

$w_{sd}(l) = (1 - \alpha).N_{C_{sd}(l)} + (\alpha).N_{RRB}$

$N_{C_{sd}(l)} \in [0, 100]$ is the normalization of the $criticality_{sd}(l)$

$N_{RRB} \in [0, 100]$ is the normalization of the reciprocal of $r(l)$

$\alpha \in [0, 1]$ is the proportion parameter

Similar to DORA, Bandwidth Guarantee with Low Complexity algorithm (BGLC) [8] has two phases. The critical values are directly proportional to the frequencies of links in all possible paths rather than the disjointed paths of just other pairs. In addition, the online phase also involves the residual bandwidths.

$$w(l) = criticality(l).\frac{1}{r(l)}$$

$$criticality(l) = \sum_{(i,j)\in D}\sum_{p_{ij}\in P_{ij}} \frac{v_l}{|P_{ij}|}$$

$$v_(l) = \begin{cases} 0 & \text{if } l \notin p_{ij} \\ 1 & \text{if } l \in p_{ij} \end{cases}$$

$|P_{ij}|$ is the number of all paths from i to j

Besides the minimum interference ideas, additional routing algorithms are recently proposed in the extent of machine learning applications.

2.4 Machine Learning Algorithms

Random race – a machine learning technique – is applied in Random Race based Algorithm for TE (RRATE) [9] to improve route computation time. The routing race approach is summarized as follow:

- The offline phase selects k shortest paths for each ingress-egress pair (s, d) as racing candidates and initialize a race reward value x_{i_sd} for the path i of (s, d).
- The online phase includes two stages: learning and post-learning. These stages are conducted separately for each ingress-egress pair.
- In the learning stage, when a demand $d(s, d, b)$ arrives, costs of the k selected paths are computed based on the number of critical links and the maximum residual bandwidths. Specifically, critical links are determined by the MIRA's maxflow-mincut definitions (i.e. critical links belong to mincut sets).

$$cost(p_{i\_sd}) = k_1.C_{i\_sd} + k_2/R_{i\_sd}$$

C_{i_sd} is the number of critical links in the path p_{i_sd}

$$C_{i\_sd} = \sum_{(s,d)\in D}\sum_{l\in p_{i\_sd}} v_l$$

$$v_(l) = \begin{cases} 1 & \text{if } l \text{ is critical of } (s,d) \\ 0 & \text{if } l \text{ is not critical of } (s,d) \end{cases}$$

R_{i_sd} is the maximum remaining bandwidth in p_{i_sd} if $d(s, d, b)$ is routed through p_{i_sd}

$$R_{i\_sd} = max_{l\in p_{i\_sd}}(r(l) - b)$$

k_1, k_2 are the moderation parameters

- High cost values mean there are more critical links and/or small remaining bandwidths. Therefore, routes are chosen in the increasing order of costs. For example, the smallest cost path is first checked for bandwidth requirement. If all links satisfy the demand then traffic is routed through that path; otherwise, the second smallest cost path is considered and so on. Additionally, whenever a path is selected, its corresponding $x_{i,sd}$ is accumulated by 1. The racing of those reward values continues until one of the paths reaches a pre-defined threshold N. Then, k paths of the (s, d) pair are sorted by the decreasing order of their respective rewards. Further requests of (s, d) are handled by the post-learning stage.
- In the post-learning stage, there is no computation but a route is selected within the sorted paths. Particularly, paths are verified against bandwidth requirement in the order of racing positions (i.e. the reward values). If the first route does not satisfy the demand, next ones are inspected. The process repeats until a satisfied route is found or all k paths are checked.
- Both two phases are reset if the network topology changes. However, normal networks do not change frequently so the post-learning stage of RRATE reduces the routing decision time.

Using the same random race technique, Paths Optimal Ordering Algorithm (POOA) [10] modifies RRATE in several aspects.

- The offline phase not only selects k shortest paths for each ingress-egress pair but also computes critical values. A link criticality relates to its subflows constituting the maxflows. Given that a link l belongs to one or more paths from s to d, the criticality is determined as:

$$criticality(l) = \frac{\sum_{(s,d)\in D} f_l^{sd}}{\sum_{(s,d)\in D} \theta^{sd}}$$

  - f_l^{sd} is the subflow of link l throw the maxflow θ^{sd}

- The learning state (online phase) calculates path costs by the criticalities and the residual bandwidths

$$cost(p_{i\_sd}) = \sum_{l\in p_{i\_sd}} \frac{criticality(l)}{r(l)}$$

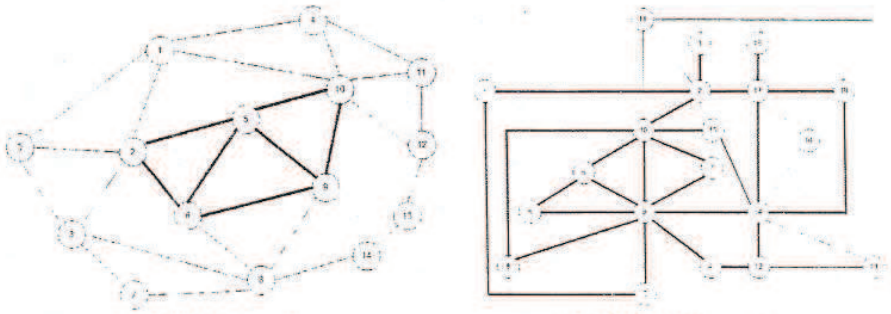
  Similar to RRATE, paths with low costs are considered first and the winning route is rewarded. However, the POOA racing threshold is not those reward values but is the whole position orders of k paths. In particular, after each demand is routed, the learning stage sorts k paths by their accumulated rewards. If that order is changed comparing to the order of the last demand, then the race is reset. In other words, the race ends on the condition that the path order continuously remains k times. (The POOA threshold is fixed to k instead of another N value of RRATE.)

– The post-learning state is the same as RRATE. It is noticed that the POOA cost computation is faster than RRATE's because the criticalities are pre-computed. Nevertheless, the later race process may be longer than the former due to the racing mechanism. Experiments will further compare and analyse them.

3 Experimets and Analysis

3.1 Simulation Environment

All the above algorithms are implemented in the network simulator NS2 [11] for experiments. Two different network topologies are simulated: one (figure 1(a)) is adapted from many previous TE routing works such as [1], [7], [10], and the other (figure 1(b)) inherits the real CESNET MPLS topology [12]. Both networks' links are bidirectional and have two types of capacity. The higher (the thicker links in the figure) is 4800, 10000 and the lower (the thinner ones) is 1200, 1000 bandwidth units respectively.



(a) MIRA topology (b) CESNET topology

Fig. 1. Network topologies

For each topology, three routing scenarios are evaluated. The first scenarios constantly demands 2000 static paths that stay in the network forever after being setup. The second one sets 2000 requests dynamically. Those requests arrive randomly according to the Poisson distribution of mean $\lambda = 80$ demands per time unit; whereas their holding (routing) time are distributed by the Exponential mean $\mu = 30$ time units. The third scenario is mixed between 200 static and 1800 dynamic requests. In this case, the distribution values are $\lambda = 40$ and $\mu = 10$. Furthermore, four ingress-egress pairs of (0, 12), (4, 8), (3, 1), and (4, 14); and eight pairs of (0, 18), (1, 11), (3, 16), (4, 7), (5, 13), (6, 19) , (15, 0) and (19, 8) are set for MIRA and CESNET topologies respectively. The former network has random bandwidth demands between 10, 20, 30, and 40 units. Meanwhile, the later arbitrarily needs 40, 80, 120, or 160 units for a request.

Two metrics are used to evaluate the algorithms in the aspect of the optimal goal described in (1). Firstly, percentage of accepted requests is compared. Obviously, the higher the accepted percent is, the better an algorithm performs.

The second metric is the average of computing time which is counted when a request arrives until it is accepted or rejected. This metric indicates the complexity of the reactive online routing phase, and should be minimized.

3.2 Evaluation Results

To obtain confident results, experiments are repeated several times with either different requests or algorithms' parameters. Comments in this paper describe the overall observation although there are few exceptions. Table 3 shows evaluating values with following parameters.

- For DORA, the bandwidth proportion $\alpha = 0.5$
- For RRATE, the moderation parameters $k_1 = k_2 = 0.5$, the number of preselected path $k = 25$, and the racing threshold $N = 10$
- For POOA, $k = N = 20$

Among single pair aware algorithms, the traditional ones (MHA and WSP) accept least number of demands in most experiments, whereas BCRA which considers more network's properties achieves better performances. There is an exception where WSP gains second highest percentage in the example results of mixed request experiment (table 3(c)). It might cause by equal-length paths in

Table 3. Comparison of accepted percent (in %) - computing time (in miliseconds) subject to number of requests (NoR)

(a) Results of static requests on MIRA network

NoR	MHA	WSP	BCRA	MIRA	NewMIRA	DORA	BGLC	RRATE	POOA
100	100-0.13	100-0.13	100-0.27	100-2.04	100-2.07	100-0.29	100-0.36	100-2.06	100-0.46
500	64.00-0.13	64.00-0.16	66.50-0.23	67.00-1.59	67.00-1.53	93.40-0.27	67.00-0.31	67.00-1.75	62.60-0.45
1000	36.20-0.12	37.20-0.14	41.40-0.22	41.50-1.24	41.50-1.14	39.70-0.27	43.50-0.31	38.20-0.44	39.20-0.42
1500	24.13-0.12	24.80-0.14	29.00-0.20	31.53-1.10	28.60-0.85	31.07-0.26	27.87-0.30	25.47-0.34	26.13-0.41
2000	18.10-0.12	18.60-0.13	21.75-0.19	23.65-1.02	21.45-0.86	25.35-0.25	20.90-0.28	19.10-0.29	19.60-0.40

(b) Results of dynamic requests on CESNET network

NoR	MHA	WSP	BCRA	MIRA	NewMIRA	DORA	BGLC	RRATE	POOA
100	100-0.16	100-0.21	100-0.35	100-4.74	100-2.52	100-0.37	100-0.31	100-5.25	100-0.59
500	86.60-0.17	83.80-0.21	84.80-0.33	84.60-4.55	89.60-2.61	85.80-0.37	86.20-0.30	94.00-3.06	88.20-0.45
1000	79.50-0.17	77.20-0.21	81.10-0.33	80.20-4.47	85.90-2.62	80.10-0.37	78.60-0.29	90.80-1.73	85.10-0.42
1500	79.27-0.17	76.73-0.21	82.53-0.33	81.07-4.67	87.13-2.85	80.53-0.36	79.93-0.30	90.40-1.21	86.20-0.40
2000	80.40-0.17	78.35-0.21	82.70-0.33	83.15-4.80	87.05-2.88	81.15-0.36	81.30-0.29	88.10-0.94	86.30-0.38

(c) Results of mixed requests on CESNET network

NoR	MHA	WSP	BCRA	MIRA	NewMIRA	DORA	BGLC	RRATE	POOA
100	100-0.59	100-0.68	100-0.35	100-5.30	100-2.49	100-0.50	100-0.30	100-5.37	100-0.59
500	77.00-0.25	75.00-0.30	76.80-0.33	76.80-4.67	76.40-2.39	78.40-0.39	76.40-0.29	78.00-3.21	76.00-0.47
1000	70.10-0.21	71.30-0.26	71.70-0.32	70.50-4.45	69.90-2.27	71.60-0.37	70.50-0.28	72.40-2.13	69.80-0.42
1500	66.50-0.20	69.87-0.24	68.20-0.32	67.13-4.39	65.93-2.22	68.20-0.37	68.00-0.28	70.47-1.73	66.13-0.42
2000	65.95-0.19	70.20-0.23	66.65-0.33	66.50-4.37	64.15-2.20	68.05-0.37	69.60-0.28	71.20-1.46	64.90-0.40

the network topology. Nevertheless, such result is not the overall trend. On the other hand, this category has the fastest computing time due to no additional calculation except the shortest path algorithm.

In addition, interference minimum algorithms attain considerable improvements of accepted percent over shortest path algorithms. Specifically, DORA accepts 7% higher number of requests than MHA in the static test (table 3(a)). Table 3(b) also depicts a 10% improvement of NewMIRA over WSP. However, there is no leading algorithm within the interference minimum category. For example, DORA obtains the highest value (25.35%) in the static experiment but the lowest one (81.15%) in the dynamic test. Meanwhile, in this test, NewMIRA outperforms the others with 87.05% comparing to the second highest of MIRA only at 83.15%. In the term of route selecting time, there is a significant difference between one-phase and two-phase algorithms. Particularly, the computing time of MIRA is even 10 ten times greater than DORA and BGLC's because MIRA recalculates maxflows for each routing demand whereas the others predetermine link's criticalities and need much less time to form link weights.

Finally, the machine learning algorithms (RRATE and POOA) also have good performances. Especially, RRATE achieve the best accepted percent in many test sets. Values of the dynamic and mixed requests experiments (table 3(b) and 3(c)) are the examples. Moreover, the average computing times of RRATE and POOA are compatible to the others. After the learning stage (e.g. after the first 100 requests in table 3(b)), the time significantly reduces. On the other hand, performances of these algorithms are greatly affected by the heuristic racing parameters (k and N values of RRATE; k value of POOA). For example, in the experiment of mixed requests on CESNET topology, when k is changed to 20 and N to 5, the average computing time of RRATE at 2000 requests decreases three times from 1.46 to 0.50 ms, whereas the corresponding percentage even increases from 71.2 to 71.5 %. However, it is almost impossible to choose the best values for all cases because the algorithms' performances depend on not only the network topology but also the routing requests.

4 Conclusion

This paper presents a study of traffic engineering routing algorithms. First, the routing problem is defined in the aspect of quality of services and traffic engineering objectives. Then, well known algorithms are described within three different categories: single pair aware, interference minimum, and machine learning. Finally, after thorough experiments, following conclusions are obtained:

- Traditional shortest path algorithms are not good enough for traffic engineering requirements. The application of network properties such as links' capacities, residual bandwidths can improve the routing effectiveness.
- The interference idea which computes links' criticalities based on the effects of current selections on future routing demands clearly enhances the number of accepted requests. Nevertheless, network topology and routing requests vary algorithms' performances so there is no best solution.

- Recently, machine learning techniques have been introduced for further improvements of TE routing algorithms. Because the requests themselves greatly impact the routing decisions, using history demands and routing data is a promising direction for traffic engineering problem.

References

1. 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)
2. Even, S., Itai, A., Shamir, A.: On the complexity of time table and multicommodity flow problems. In: Proceeding SFCS 1975 Proceedings of the 16th Annual Symposium on Foundations of Computer Science, pp. 184–193. IEEE (1975)
3. Guerin, R.A., Orda, A., Williams, D.: QoS routing mechanisms and OSPF extensions. In: Global Telecommunications Conference, GLOBECOM 1997, vol. 3, pp. 1903–1908. IEEE (1997)
4. Kotti, A., Hamza, R., Bouleimen, K.: Bandwidth constrained routing algorithm for MPLS traffic engineering. In: International Conference on Networking and Services (ICNS 2007), Athens, Greece, p. 20 (2007)
5. Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network flows: Theory, algorithms, and applications. Prentice-Hall, Englewood Cliffs (1993)
6. Wang, B., Su, X., Chen, C.: A new bandwidth guaranteed routing algorithm for MPLS traffic engineering. In: Conference Proceedings of 2002 IEEE International Conference on Communications, ICC 2002 (Cat. No.02CH37333), pp. 1001–1005 (2002)
7. Boutaba, R., Szeto, W., Iraqi, Y.: DORA: efficient routing for MPLS traffic engineering. Journal of Network and Systems Management 10, 309–325 (2002), doi:10.1023/A:1019810526535
8. 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), Kuala Lumpur, Malaysia, pp. 205–210 (2009)
9. 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)
10. Lin, N., Lv, W.-F., Yang, T.: Paths optimal ordering algorithm based on MPLS traffic engineering. In: 2009 Ninth International Conference on Hybrid Intelligent Systems, Shenyang, China, pp. 492–495 (2009)
11. Issariyakul, T., Hossain, E.: Introduction to network simulator NS2. Springer, New York (2011)
12. Novák, V., Adamec, P., Šmrha, P., Verich, J.: CESNET2 IP/MPLS backbone network design and deployment in 2008 (2008)