Delay-based Quality of Service through Intra-Domain Differentiated Routing with Optimised Link Weight Setting

Ioannis Papanagiotou
Centre for Communication Systems Research (CCSR)
University of Surrey
Guildford, England
i.papanagiotou@surrey.ac.uk

Michael Howarth
Centre for Communication Systems Research (CCSR)
University of Surrey
Guildford, England
m.howarth@surrey.ac.uk

Abstract— **The wide range of applications that are supported on the Internet requires it to deliver a diverse set of classes of service. For example, network providers need to support both delay-sensitive multimedia services and less time-sensitive applications such as web browsing and peer-to-peer transfers. Differentiated routing is one approach to delivering these different classes of service. In this paper we use an evolutionary algorithm to set link weights for our previously described Intra-Domain Differentiated Routing algorithm (IDDR) so as to optimise the delay differentiation between two classes of flows. In order to provide good robustness, the link weight setting is based on multiple traffic matrices. Results demonstrate that the delay for high QoS traffic is typically less than 60% of the delay for best effort traffic, and that this can be achieved across a wide range of traffic matrices. We also show that the delay differentiation is achieved by sending high QoS flows along paths that have fewer links and that those links have in general lower delay.**

Keywords-component: QoS; Intra-domain; Delay-based; Service Differentiation; Evolutionary Algorithm;

I. INTRODUCTION

As the Internet has evolved, the variety of content carried has grown, for example covering security sensitive bank transaction and delay sensitive multimedia content. In the past five years the latter has grown to be a significant portion of current internet traffic. Therefore, it has become important to take into consideration not only the volume of data to be routed but also the data diversity. Modern demands on the Internet call for improved routing approaches that incorporate the ability to support Quality of Service (QoS).

Although end to end QoS in the entire Internet is a desirable goal, the first step to achieving this goal is to deploy architectures that support QoS in each domain separately. We have presented in a previous paper the Intra-Domain Differentiated Routing Algorithm (IDDR) [1], which demonstrated substantial service differentiation through differentiated routing in a pure IP network. The advantages presented by this algorithm are important. Firstly it is an extension of the widely used Open Shortest Path First (OSPF) algorithm. Therefore it is easily incorporated to most networks as there is no need for any network architecture change. Being a hop-by-hop algorithm makes IDDR very dynamic as decisions are made in each node separately and there is no need for any centralized control entity

Traditionally network operators in plain IP networks can modify link weights to control, to a certain extent, the traffic distribution in the network. Since a change in only one link weight may trigger more than one shortest path to change, such a practice can often trigger shifts in traffic, some of which are undesirable, and lead to congestion elsewhere in the network or otherwise affect the delivered quality of service. The improvement of network performance via changing link weights is a complex task. To optimize IDDR's performance, in this paper we describe how we use an evolutionary algorithm (Genetic Algorithm) to modify the link weights so as to optimise our chosen QoS metric, delay differentiation. Genetic Algorithms are heuristic mechanisms for finding near-optimal solutions to complex optimization problems [2]. We show that IDDR's performance is substantially improved by optimising the link weights.

A. *Related Work*

There has been a substantial amount of research on intra-domain QoS in the past. Most approaches are based on

differentiated forwarding. Lately though, differentiated routing has attracted attention. Most texts on QoS through differentiated routing refer to algorithms based on Multiprotocol Label Switching (MPLS) framework [3, 4, 5]. Although MPLS provides increased control over the network resources, its main drawback is the need for a centralized control entity. In case of failure or lack of connectivity with the control entity, the network is vulnerable and unable to react to any node failures or any other unforeseen circumstances.

Multi-topology Routing is another method for supporting differentiated routing. Most of the multi-topology routing texts address more network resiliency and load balancing rather than service differentiation [6, 7, 8].

For plain IP networks a large part of the research work on differentiated routing has focused on load balancing and network resilience [9, 10], while other texts have concentrated on algorithms that work as extensions of OSPF [11] and on problems that may arise when accounting for QoS based on more than one metric [12].

II. IDDR

IDDR [1] is based on Shortest Path First with Emergency Exits (SPF-EE) [13]. The latter was developed as a fault-tolerant, congestion avoidance algorithm, whose main goal is to increase network resilience. IDDR uses a modified version of SPF-EE to provide service differentiation. Its service differentiation capabilities have been demonstrated for two QoS classes.

IDDR works as follows. Given link state information about the network, nodes run Dijkstra's algorithm [14] in order to calculate the shortest paths (SPs) to all destinations. Each node builds its own routing tree and those of its neighbours. Each leaf of the tree consists of the node's ID and the cost of the path from the root of the tree to the node. Each node then computes every available alternative path (AP). Computing APs is done as follows. The current node looks at its neighbour routing trees for the destination. The the neighbour must not the SP's next hop of the current node and the current node must not be included in the AP, i.e. the destination, on the neighbour's routing tree, is not on a sub-tree rooted under the current node itself. If both the above criteria are met then this neighbour is considered an exit, i.e. the next hop to an AP. The node retrieves the cost to the exit and then it adds it to the cost between the exit and the destination; therefore each node can deduce the cost of each AP, if more than one. The APs are then classified and inserted in the routing table in order of ascending cost. In case a node fails to route a flow both by SP or AP, it will attempt to route through a Reverse Alternative Path (RAP): when SP and APs are all congested, the node will send the packet to any neighbour and command the neighbour to route it through his APs. If the neighbour cannot route the flow as well it will send it to an arbitrary neighbour other than the sender and so on until it reaches the RAP limit, which is there to prevent loops. To avoid loops we have also set another parameter, namely the reroute limit. This dictates how many times will a flow be rerouted to an AP. Reroute limit is more dominant than the RAP limit in the sense that the RAP limit will be considered only if reroute limit is not exceeded. In this paper we set the reroute limit to 2 (in our previous paper [1] we set the reroute limit to 1).

We aim to provide two qualitatively differentiated classes of QoS flows: high QoS, i.e. low delay; and best effort, i.e. higher delay. IDDR allows best effort (BE) traffic to be routed only through alternative paths (APs), while high QoS flows are routed through the shortest paths (SPs). If the SPs are over-utilised the additional high QoS flows are also allowed to use APs. The higher cost (usually longer) paths taken by the BE traffic increase the total delay of the best effort flows, increasing service differentiation. To avoid problems such as out-of-order delivery of TCP packets or UDP jitter, packets of the same flow need to follow an identical path.

To limit the traffic volume on links used by high QoS flows we use a parameter, the IDDR threshold n, which bounds percentage link utilisation by the value of n. The introduction of this link utilisation gives the network operator some control over the delay of the high QoS flows. The reduced shortest path traffic volume reduces the delay suffered by the high QoS flows. This method using the IDDR threshold, by its nature, also reduces the throughput of the network since it effectively reduces the capacity of its affected links.

III. GENETIC ALGORITHM

Genetic Algorithms (GAs) are heuristic mechanisms for solving complex optimization problems. They are based on "natural selection", based on Darwin's evolution theory. "Natural selection" acts to safeguard and collect any advantageous genetic mutation. In this paper the functional advantage we wish to preserve is a set of link weights that improves our fitness function, i.e. a hybrid metric of both throughput and service differentiation.

We now briefly review terminology associated with the genetic algorithm and which are used in this paper as reviewed from [15]. A gene represents a link weight. Each set of genes forms a chromosome which is a complete set of link weights for the topology. A set of a number of chromosomes is called a genome. A genome is populated, i.e. created, within one generation.

There are four basic methods, which drive a selection algorithm. In order of execution these are: Elitism, Selection, Crossover and Mutation. During elitism the GA will select a small percentage of the total genome population, with the best chromosomes (elite of chromosomes) and store them on the new generation's genome exactly as they are. After elitism, in order to further populate the next generation's genome two chromosomes (parents) are selected using Roulette Wheel Selection (RWS). RWS is a selection method that assigns each chromosome with a probability of selection which is

directly and linearly proportional to its fitness. The fitness is provided by the cost function. The GA then carries on with the crossover method and creates a new chromosome (offspring) by interchanging the parent chromosomes at a user specified crossover point. The crossover method used in this GA is the single crossover method. After this stage the offspring is mutated by the mutation method so that it retains most of its parents' characteristics but at the same time it is not identical to its parents.

We now define the principal parameters associated with GA. *Population size* is the size of each genome, i.e. the amount of chromosomes making up the genome. *Elitism size* refers to the amount of chromosomes the elitism method stores in the next generation's genome as explained in Section 3. *Crossover rate* is the percentage of chromosomes that will undergo the crossover method. *Crossover point* is the place at which the interchange of parent chromosomes will occur to create the offspring chromosome. It is a randomly generated number in the space between zero and the number of genes in each chromosome. *Mutation rate* refers to the percentage of chromosomes that will undergo the mutation method. Finally *mutation level* and *mutation degree* are the maximum value that will be added to the current value of the selected gene and the number of genes that will be selected for mutation respectively. The latter selection process is random. Table I gives the value used in this paper of each parameter.

TABLE I. GA PARAMETER VALUES

Population Size	200	**Mutation Rate**	40
Elitism Size	15	**Mutation Level**	10
Crossover Rate	100	**Mutation Degree**	50

Finally, we introduce the fitness function which returns a value that states the degree of optimization achieved by each chromosome. The fitness function is of the following form:

$$f = \alpha \cdot (T) + \beta \cdot (U_d) + \gamma \cdot (D_d) \qquad (1)$$

where f is fitness, T is throughput fitness, U_d is utilization fitness and D_d is delay fitness, each of which is explained below:

• Throughput fitness: the ratio of the throughput (defined in section C) to the total traffic injected into the network.

• Utilisation fitness: we sort the network's N_L links in descending order of utilization, and define utilization difference by the following formula:

$$U_d = \frac{\left[\sum_{i=0}^{(N_L/2)-1} \eta_i - \sum_{i=N_L/2}^{N_L} \eta_i\right]^{OSPF}}{\left[\sum_{i=0}^{(N_L/2)-1} \eta_i - \sum_{i=N_L/2}^{N_L} \eta_i\right]^{IDDR-GA}} \qquad (2)$$

where U_d is the utilisation difference and η_i is the utilization of link *i*. The numerator of Eq.(2) is calculated using the network when it runs OSPF with link weights set inversely proportional to link capacity, while the denominator is calculated when the network runs IDDR and its link weights are optimized by GA. A network with a more even spread of traffic across the links has a higher value of U_d. The numerator of the equation gives a sensible scaling to the value of U_d.

• Delay fitness: a measure of how much the delay of the best effort flows is higher than the delay of the high QoS ones, defined as follows:

$$D_d = \frac{\delta_{BE} + \delta_{OSPF}}{\delta_{QoS}} \qquad (3)$$

where δ_{BE} is the mean delay of all BE flows and δ_{QoS} is the mean delay of all QoS flows with IDDR as the routing protocol. δ_{OSPF} is the mean delay of all flows when we are using OSPF, and is introduced to give a higher weighting for networks that have a low value of δ_{QoS}.

The factors α, β and γ sum to 1, and they control the extent to which throughput, utilization fitness and delay fitness influence the fitness function and therefore the performance of IDDR.

IV. SIMULATION DESIGN

The software used to simulate both the operation of IDDR in a QoS-enabled network and the GA algorithm has been implemented in C++.

A. Delay Modelling

The delays imposed on the flows along their path to destination use a simple model [19] from queuing theory. In this, we model the link delay as a function of link utilisation and link capacity:

$$t_q = \frac{\rho}{1-\rho} \qquad (4)$$

where ρ is the link utilisation and t_q is the link delay.

Even though this is a simple model and does not consider the heavy tail distribution illustrate typically demonstrated by the Internet traffic flow, it provides useful insights to the performance of the algorithm.

B. Test Topologies

We investigate the extent to which IDDR can be improved by means of GA, using three topologies: (a) random topologies created by the BRITE topology generator [16], hereafter called Random Brite (RB) topologies; (b) the test topology introduced by Calle [17], depicted in Fig. 1; and (c) the real topology of the Géant research network [18].

The random BRITE topologies each consist of 15 nodes with link connectivity assigned using the Waxman model. In our simulations we used four different RB topologies, and the results presented below are an average from these different topologies. In our RB topologies, each node has at least 3 links attached and the link capacities are uniformly distributed in the range from 10 to 100 units. Calle's topology consists of 15 nodes, 5 of which are interconnected with high capacity links and form the core of the topology. In particular, we have assumed the core links (shown in Fig. 2

in bold) have a capacity of 100 units while the remaining links each have a capacity of 50 units. Finally the Géant topology uses its link weights and capacities of the real network.

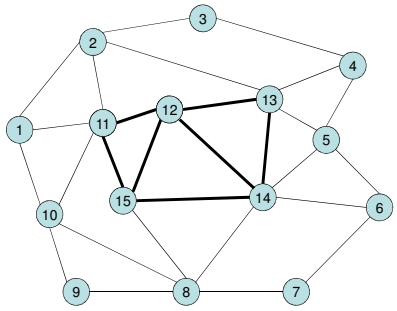

Figure 1. Calle Topology [17]

C. IDDR Setup

Before we present simulation results, we introduce the following terms:

• Total traffic is the sum of all the data rates (bandwidths) of flows injected into the network.

• QoS ratio is the percentage of total traffic that requires preferential treatment, i.e. is "high QoS" or low delay traffic.

• Network utilisation: defined by the following formula:

$$\eta = \frac{\sum b_i}{\sum c_i} \qquad (5)$$

where η is the network utilisation, b_i is the traffic volume on link i, and c_i is the capacity of link i. Network utilisation is therefore the aggregate fraction of total network resources that are used. It illustrates the demand of each algorithm on the network resources.

The total traffic injected to the RB topologies is 900 flows while in the Calle topology we set the total traffic to 1950 flows. All flows have assigned a bandwidth of one unit. The traffic quantities were chosen to give a network utilization of around 50 %. The only delay we take into consideration is queuing delay; according to our simple model (Eq. (4)) this varies only slightly for link utilisations up to around 45%. Therefore, if we use a lower network utilisation, we find that the delay differentiation depends primarily on what we call the Difference in Link Number Per Path (DLNPP), which is defined by the following formula:

$$DLNPP = \left[\left(\frac{\overline{HC_{BE}}}{\overline{HC_{QoS}}} \right) - 1 \right] \times 100\% \qquad (6)$$

where $\overline{HC_{BE}}$ is the average hop count per path of the best effort flows and $\overline{HC_{QoS}}$ is the average hop count of the high QoS flows.

By using a higher network utilisation we shall show that delay differentiation is not only a function of DLNPP but also, most importantly, depends on the observation that high QoS flows take "faster" links (i.e. less utilized) and BE flows follow "slower" links (i.e. those with higher utilization). Flows injected to the RB topology are randomly generated and originate from any node to any other node. For the Calle topology we have introduced the notion of edge nodes and core nodes. The former are the nodes that form the perimeter of the topology (i.e. nodes 1-10 in Fig. 1) while core nodes are all other nodes, i.e. those that are not on the perimeter (nodes 11-15 in Fig. 1). In our work, traffic in the Calle topology is randomly generated from any edge node to any other edge node. This models networks that mainly handle transit traffic. For the Géant topology real traffic matrices (TMs) from TOTEM have been used [18].

Our analysis and evaluation of the network is based on the following metrics:

• Path Delay: the sum over all the links along the path used by any individual flow of the queuing delays experienced on each link, as given by equation (4) above. This enables us to calculate the service differentiation IDDR offers.

• Throughput: the actual volume of traffic that is successfully passed by the network. In some tests the utilisation of some links reaches the limiting capacity, and not all the traffic injected into the network (the "total traffic") can be accommodated on the network.

• Delay Differentiation: the percentage by which the delay of the best effort flows is higher than the delay of the high QoS ones, defined as follows:

$$D_d = \frac{\delta_{BE}}{\delta_{QoS}} \qquad (7)$$

where δ_{BE} is the mean delay of all BE flows and δ_{QoS} is the mean delay of all QoS flows with IDDR as the routing protocol, while δ_{OSPF} is the mean delay of all flows when we are using OSPF.

V. Results And Analysis

To maximize the robustness of the solution obtained, for the Calle and each RB topology we used 5 different random TMs during the Genetic Algorithm. In the case of the Geant topology the random TMs were replaced with real TMs from Totem [18]. The network configuration represented by each chromosome is tested against each topology to obtain the fitness function (Eq. (1)). We calculate throughput and delay differentiation for each TM, then average them, and calculate the fitness function based on the average throughput and average delay differentiation over all 5 TMs. Once the algorithm has produced a set of link weights, we then test this set of link weights against 100 other random traffic matrices in order to generate the results presented in this Section.

For our initial results we ran the GA on Calle's topology with the fitness set to optimize only delay differentiation ($\alpha = 0$, $\beta = 0$, $\gamma = 1$ in Eq. (1)). This resulted in a substantial improvement in delay differentiation but at the cost of a lower throughput. To counteract the decrease in throughput, we then introduced the throughput factor T in the fitness function (2).

For all three topologies Figure 4(a) illustrates the variation in delay differentiation of the topologies with respect to the delay differentiation factor γ, while figure 4(b) depicts the throughput as a function of γ. A delay differentiation of 100% means that the mean BE delay is twice that of the QoS delay.

Table II shows the throughput obtained for each topology when running pure OSPF, and the throughput and delay differentiation achieved with pure IDDR (no GA); here, the link weights are inversely proportional to the link capacities in case of CL and RB topologies, and for the Géant topology we use the original link weights. Figure 4(a) shows that the Genetic Algorithm has substantially improved the delay differentiation compared to plain IDDR (given in Table II).

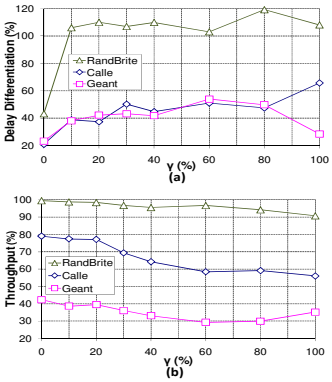

Figure 4. Delay Differentiation (a) and throughput (b) against γ, no utilisation difference (β=0)

Figure 4(b) shows that generally the throughput of all topologies drops as γ increases. We seek a value of γ for which we have a substantial delay differentiation while at the same time giving only a small drop in throughput. The results suggest that optimal γ values for RB, Calle and Géant topologies would be 20%, 30% and 20% respectively. For all topologies and especially RandomBrite the delay differentiation with the GA (Fig. 4(a)) is much improved compared to pure IDDR (Table II), while at worst case in the Calle topology the throughput loss is only 8% (at γ=30%).

TABLE II. OSPF AND PURE IDDR RESULTS

	OSPF	IDDR	
	Throughput	**Throughput**	**Delay Diff.**
Calle	72%	78.33%	28.58%
RandBrite	97.13%	98.36%	31.55%
Géant	37.71%	39.13%	27.79%

However, during our experiments we noticed that using the link weights given by GA resulted in some links not being used at all, and this may suggest why throughput drops. In an attempt to improve the throughput we therefore introduce the utilization difference factor in our fitness function to spread traffic across as many links as possible and therefore minimize congestion problems. We set β = 0.5 and then we tested for γ in the range 0.2 to 0.5, varying α so that the sum of the three factors is always 1.0. Figure 5(a) and 5(b) illustrate the variation in delay differentiation of the topologies and throughput respectively, with respect to the delay differentiation factor γ.

We can see in Figure 5(a) that for the Calle topology the delay differentiation has dropped considerably to levels close to plain IDDR. That means that for Calle topology plain IDDR performs close to optimally, as the GA cannot significantly improve the service differentiation. For the RB topology the delay differentiation is reduced at low γ. In the Géant topology the service differentiation using β = 0.5 is similar to Figure 4(a).

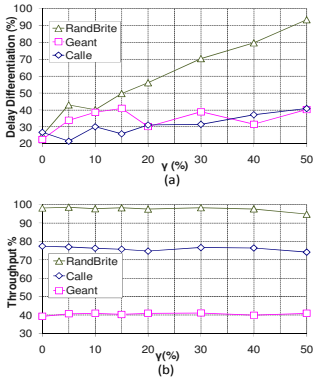

Figure 5. Delay differentiation (a) and throughput (b) against γ, with utilisation difference β=0.5

However, the throughput is more constant, even at high values of γ. It is important to note that for all topologies the throughput for any value of γ is up to 6% percent higher than that we have achieved using simple OSPF, and in any case as good as that achieved when using pure IDDR.

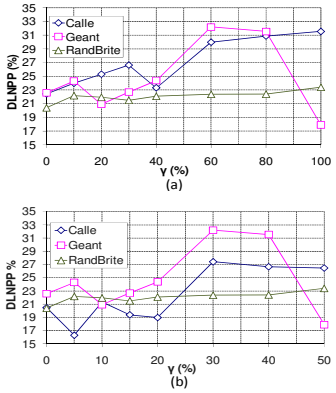

Figure 6. DLNPP vs. γ. (a) utilisation difference β = 0, and (b) utilisation difference β = 0.5 (b)

The delay differentiation could be attributed to two factors: the number of links per path, i.e. hop count, and the delay imposed by each link. We now demonstrate that path length alone can not justify the delay differentiation achieved by IDDR. Figure 6 illustrates the DLNPP, i.e. the percentage by which the average length of the paths followed by the BE flows is larger than the average length of the paths followed by the High QoS flows, as a function of γ.

Figures 6(a) and 6(b) show that the large delay differentiation exhibited by the RB topologies is only partly related to DLNPP. If DLNPP were the only factor affecting the delay differentiation, the achieved delay differentiation would be much lower, for example approximately 22% for the RB topologies. Yet our simulations show delay differentiation in excess of 90% for RB. For the Géant and CL topologies DLNPP is responsible for a larger fraction of the delay differentiation, since in an evenly loaded network it is responsible for up to half of the achieved delay differentiation. However it is evident that in the enhanced IDDR with GA we achieve delay differentiation, which comes at least partly from selecting paths for the high QoS flows that include less utilized links. Indeed, when we look in detail at the utilization of individual links, our results show that the highly utilized links carry on average more BE traffic (approximately 45% high QoS, 55% BE) while less utilized links carry on average more high QoS traffic (approximately 55% high QoS, 45% BE). The algorithm is not able to provide further separation of high QoS and BE flows because the flows that the topologies support are very dense: it is not feasible to separate links so that some have low utilization and carry high QoS flows while others carrying BE flows have high utilization. This supports our conclusion that link delay is a significant contribution to the delay differentiation, as explained earlier.

VI. CONCLUSION

Quality of service is an important metric for enabling the modern Internet to provide a balanced service to the wide range of applications that use the modern Internet. We have previously introduced IDDR as a mechanism for giving a simple two delay class service based on differentiated routing. In this paper we have improved IDDR by using an evolutionary algorithm to optimise the link weights used by IDDR. Results show that across a range of topologies we can improve the delay differentiation from an average of approximately 29% across all three topologies with pure IDDR to a maximum of 70% using the genetic algorithm when $\gamma \approx 60\%$-80% and $\beta = 0\%$ and a maximum of 57% when $\gamma \approx 50\%$ and $\beta = 50\%$. There is a tradeoff between the amount of delay differentiation and throughput, and a reasonable balance can be obtained in which the throughput provided by the network only decreases by around 2% in average, when opting for a delay differentiation of 57%. The solutions exhibit excellent robustness, by being optimized across a number of test traffic matrices. Finally, we have demonstrated that the delay differentiation is achieved by sending high QoS flows along paths that have fewer links and that those links have in general lower delay.

ACKNOWLEDGMENT

Ioannis Papanagiotou is supported by the UK Engineering and Physical Sciences Research Council (EPSRC).

REFERENCES

[1] Papanagiotou, I., Howarth, M.: Intra-domain delay-based quality of service using differentiated routing, In: IFIP/IEEE Management of Multimedia and Mobile Networks and Services (MMNS 08), pp 127-138, (2008), doi: 10.1007/978-3-540-87359-4_12

[2] Ericssonm M., Resende, M.G.C., Pardalos, P.M.: A genetic algorithm for the weight setting problem in OSPF routing. In: J. Combinatorial Optimization vol. 6, pp.299-333, (2002).

[3] Li, Z., Zhang, Z., Wang, L.: A novel QoS routing scheme for MPLS traffic engineering. In: International Conference on Communication Technology Proceedings (ICCT), vol. 1, pp. 474-477. IEEE (2003) doi: 10.1109/ICCT.2003.1209122

[4] Wang, B., Su, X., Chen, C.L.P.: A new bandwidth guaranteed routing algorithm for MPLS traffic engineering, In: International Conference on Communications (ICC), vol. 2, pp. 1001-1005. IEEE (2002) doi: 10.1109/ICC.2002.997005

[5] Calle, E., Marzo, J.L., Urra, A., Vila, P.: Enhancing MPLS QoS routing algorithms by using the network protection degree paradigm. In: Global Telecommunications Conference (GLOBECOM), vol. 6, pp 3053-3057, IEEE (2003) doi: 10.1109/GLOCOM.2003.1258796

[6] Gjessing, S.: Implementation of two Resilience Mechanisms using Multi Topology Routing and Stub Routers. In: International Conference on Internet and Web Applications (ICIW), pp. 29-29. IEEE (2006) doi: 10.1109/AICT-ICIW.2006.110

[7] Hansen, A.F., Kvalbein, A., Cicic, T., Gjessing, S., Lysne, O.: Resilient routing layers for recovery in packet networks. In: International Conference on Dependable Systems and Networks Proceedings (DSN), pp. 238-247. IEEE (2005) doi: 10.1109/DSN.2005.81

[8] Kvalbein, A., Hansen, A. F., Cicic, T., Gjessing, S., Lysne, O.: Fast IP Network Recovery Using Multiple Routing Configurations. In: Proceedings in 25th IEEE International Conference on Computer Communications (INFOCOM), pp. 1-11, IEEE (2006) doi: 10.1109/INFOCOM.2006.227

[9] Sahoo, A.: An OSPF based load sensitive QoS routing algorithm using alternate paths. In: Computer Communications and Networks Proceedings, pp. 236-241. IEEE (2002) doi: 10.1109/ICCCN.2002.1043072

[10] Sahoo, A.: A load-sensitive QoS routing algorithm in best-effort environment. In: MILCOM Proceedings, vol. 2, pp. 1206-1210. IEEE (2002)

[11] Guerin, R.A., Orda, A., Williams, D.: QoS routing mechanisms and OSPF extensions. In: Global Telecommunications Conference (GLOBECOM), vol. 3, pp.1903-1908. IEEE (1997) doi: 10.1109/GLOCOM.1997.644603

[12] Wang, Z., Crowcroft, J.: Quality-of-service routing for supporting multimedia applications. In: IEEE Journal on Selected Areas in Communication, vol. 14, issue 7, pp. 1228-1234. IEEE (1996) Moy, J., Cascade Communication, Corp.: OSPF Version 2. In: RFC 2178. IETF (1997) doi: 10.1109/49.536364

[13] Wang, Z., Crowcroft, J.: Shortest path first with emergency exits. In: ACM SIGCOMM Computer Communication Review , vol. 20, issue 4. ACM (1990)

[14] Tanenbaum, A.: Computer Networks(4th Edition).Prentice Hall, 2003

[15] "Introduction to genetic algorithms – Tutorial with interactive java applets", http://www.obitko.com/tutorials/genetic-algorithms/

[16] "Boston university Representative Internet Topology gEnerator", http://www.cs.bu.edu/brite

[17] Calle, E., Marzo, J., Urra, A.: Protection performance components in MPLS networks. In: Computer Communications, vol. 27, issue 12, pp.1220-1228, Science Direct (2004) doi: 10.1016/j.comcom.2004.02.025

[18] TOTEM (TOolbox for Traffic Engineering Methods) Project, http://totem.run.montefiore.ulg.ac.be/datatools.html.

[19] Adan, I., Resing, J.: Queueing theory. In: http://www.win.tue.nl/~iadan/queueing.pdf (2002)