Quality of Service Provisioning for Supporting Premium Services in IP Networks

P. Trimintzios† , T. Baugé‡ , G. Pavlou† , L. Georgiadis§ , R. Egan‡ and P. Flegkas†

† Centre for Communication Systems Research, University of Surrey, Guildford, Surrey, GU2 7XH, United Kingdom

‡ Thales Research & Technology (UK) Ltd, Worton Drive Reading, RG2 0SB United Kingdom

§ School of Electrical and Computer Engineering, University of Thessaloniki, P.O. Box 435, 54006, Thessaloniki Greece

***Abstract*--Given the emergence of IP networks and the Internet as the multi-service network of choice, it is plausible consider its use for transporting demanding multimedia traffic with high bandwidth and low delay and packet loss requirements. Emerging technologies for quality of service such as Differentiated Services and MPLS can be used for premium quality traffic. In this paper we present a Traffic Engineering and Control System that starts from agreed services with customers and provisions the network according to the expected traffic demand so as to meet the requirements of contracted services while optimizing the use of network resources. We devise a non-linear programming formulation of the problem and show through simulations that we can achieve the objectives and meet the requirements of demanding customer traffic.**

I. Introduction

DIFFERENTIATED Services (DiffServ) [1] is seen as the emerging technology to support Quality of Service (QoS) in IP backbone networks in a scalable fashion. Multi-Protocol Label Switching (MPLS) [2] can be used as the underlying technology to support traffic engineering. It is possible to use these technologies to support multimedia traffic with stringent real-time requirements. This can be done through careful traffic forecast based on contracted premium services with customers and subsequent network provisioning in terms of routing and resource management strategies. In this paper we show that this is a feasible solution for guaranteeing QoS for demanding multimedia traffic. In order to provide adequate quality guarantees over an IP Autonomous System (AS) that serves demanding traffic, we propose to use the DiffServ framework together with MPLS for Traffic Engineering (TE).

Customers have contractual Service Level Agreements (SLAs). Routing protocols determine paths across the network based on a shortest path algorithm. Conventional protocols do not take into account the load on different parts of the network, which may lead to certain areas of the network being congested while others underutilized. DiffServ selectively distributes resources among the various classes of traffic, which helps maintain quality for selected traffic aggregates but does not tackle prolonged congestion. In order to provide quality guarantees while optimizing the use of network resources, it may be best to take a longer but less congested path through the network [3]. This requires alternative routing methods to complement DiffServ operation.

II. A Provisioning System for QoS Delivery and Traffic Engineering

We present the functional model that enables an Internet Service Provider (ISP) to automatically provision its ASs in order to satisfy the customers' requirements. We are focusing on defining a time-dependent IP Traffic Engineering and Control System that operates at medium (hours to days) and long (weeks) timescales.

We assume that customers have bilateral SLAs that contain both an administrative aspects (terms, conditions, pricing information, etc.) and technical parts, known as a Service Level Specification (SLS) [4]. Our system focuses on how to satisfy the SLSs of quality-critical customers. Fig. 1 shows the model our proposed system, which has evolved from our initial quality of service framework described in [5].

The system in Fig. 1 is designed to provide an automated mechanism to determine, implement and monitor network configurations, which satisfy the following criteria:

1. Deliver the services as defined in SLSs (customer perspective),
2. Optimize the use of network resources (ISP perspective)

The system is decomposed in three main sub-systems: SLS management, Traffic Engineering (TE) and Monitoring.

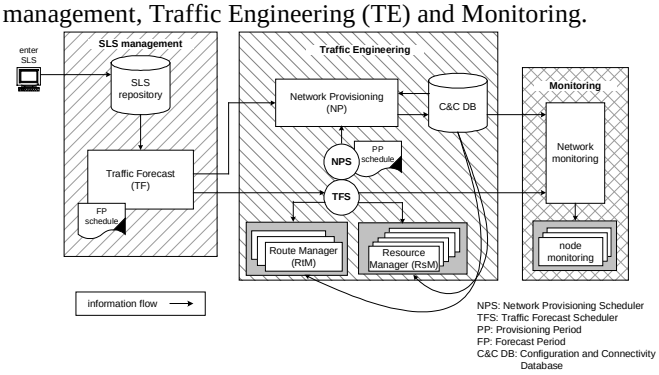

Fig. 1. Detailed decomposition of IP Traffic Engineering and Control System.

A. From SLSs to Classes of Service

Network provisioning requires forecasts, which are based on the following information: subscribed SLSs , administrators' policies (e.g. the forecasting period), measurements and historical data. We are considering the following SLS types: video, audio, which both have stringent delay and loss requirements, and network control and background traffic which has less strict delay and certainly smaller throughput requirements.

TABLE I
Definition of Class Of Service (CoS) and Traffic Trunk

Class of Service (CoS)	PHB Scheduling Class	Delay	Loss probability
Traffic Trunk	Ingress IP address	Egress IP address	CoS	Rate

III. NETWORK PROVISIONING AND TRAFFIC ENGINEERING

Network provisioning is the main module of our traffic engineering system (Fig. 1). It provides guidelines for *provisioning* the network according to traffic estimates and resource utilization policies. Its objectives are both traffic and resource-oriented. The former relate to the obligation towards customers, through the SLSs. These obligations induce a number of restrictions about the treatment of traffic. The resource-oriented objectives are related to the network operation, and they are results of the high-level business policy that dictates the network should be used in an optimal fashion.

A. Network and traffic model

The network is modeled as a directed graph $G = (V, E)$, where V is a set of nodes and E a set of links. With each link $l \in E$ we associate the following parameters: the link physical capacity C_l, the link propagation delay d_l^{prop}, the set of the physical queues K, i.e. PHB Scheduling Classes (PSCs), supported by the link. For each PSC, $k \in K$ we associate a bound d_l^k (deterministic or probabilistic depending on the PSC) on the maximum delay incurred by traffic entering link l and belonging to the $k \in K$, and a loss probability p_l^k of the same traffic.

The basic traffic model of network provisioning is the traffic trunk. A traffic trunk is an aggregation of a set of traffic flows characterized by similar end-to-end performance requirements [3]. Also, each traffic trunk is associated with one ingress node and one egress node, and is unidirectional. Each trunk corresponds to a CoS. The set of all traffic trunks is denoted by T. Each trunk $t \in T$, from node v_i to egress node v_e, is associated with a capacity requirement B_t.

The following information for each traffic trunk is directly available because of the CoS definition (see Table I):

- The PSC of the traffic carried on the trunk, thus inferring the PHB $k \in K$ of the trunk.
- The bound D_t maximum end-to-end delay.
- The maximum end-to-end loss probability, P_t.
- The capacity B_t requirement.

B. Cost definition and optimization objectives

We want to provide a set of routes for each traffic trunk. For each ingress-egress pair these routes are implemented as Label Switched Paths (LSPs) at the routers. We also need to compute the amount of bandwidth that is to be allocated to each PSC at all the interfaces of the network.

The *primary objective* of such an allocation is to ensure that the requirements of each traffic trunk are met as long as the traffic carried by each trunk is at its specified minimum bandwidth. This objective ensures that the SLS requirements are met. The design objectives are further refined to incorporate other requirements such as:

(a) Avoid overloading parts of the network while other parts are under loaded.
(b) Provide overall low network load (cost).

The last two requirements do not lead to the same optimization objective. In any case, in order to make the last two requirements more concrete, the notion of "load" has to be quantified. In general, the load (or cost) on a given link is an increasing function of the amount of traffic the link carries. This function may refer to link utilization or may express an average delay, or loss probability on the link. QoS seen by the traffic using the different PHBs varies, so the link load induced by the traffic of each PHB may vary.

Let x_l^k denote the capacity demand for $k \in K$ satisfied by link l. Then the link cost induced by the load is a convex function, $f_l^k(x_l^k)$, increasing in x_l^k. The total link cost per link is defined as

$$F_l(\overline{x}_l) = \sum_{k \in K} f_l^k(x_l^k) \tag{1}$$

where $\overline{x}_l = \{x_l^k\}_{k \in K}$ is the vector of demands for all PHBs of link l. In order to take into account that link capacities may be different, the cost may be modified to reflect equivalent utilization by normalizing with the link capacities C_l.

If the appropriate buffers have been provided at each router and the scheduling policy has been defined, then $f_l^k(x_l^k)$ may specify the *equivalent capacity* needed by PHB $k \in K$ on link l in order to satisfy the loss probability associated with that PHB. Hence, the total cost per link is the total equivalent capacity allocated on link l. This definition has the following drawbacks: 1) the cost definition depends on the PHB implementation, 2) the cost functions may not be known, or may be too complex. Hence approximate cost functions must be used.

In this work we defined the following cost function:

$$f_l^k(x_l^k) = \begin{cases} \dfrac{x_l^k}{C_l - x_l^k} & , \; x_l^k \le h_l^k C_l \\ \dfrac{h_l^k}{\left(1 - h_l^k\right) e^{1/1 - h_l^k}} e^{\frac{x_l^k}{(1-h_l^k) h_l^k C_l}} & , \; x_l^k > h_l^k C_l \end{cases} \tag{2}$$

Function (2) combined with (1), when load x_l^k is less than a percentage h_l^k of the link capacity, gives as a total cost function the average number of packets in the system based on the hypothesis that each queue behaves as a M/M/1 queue [6]. We can now formally define the objectives (a) and (b):

$$\text{minimize} \quad \max_{l \in E} F_l(\overline{x}_l) = \max_{l \in E} \left\{ \sum_{k \in K} f_l^k(x_l^k) \right\} \tag{3}$$

$$\text{minimize} \quad \sum_{l \in E} F_l(\overline{x}_l) = \sum_{l \in E} \left(\sum_{k \in K} f_l^k(x_l^k) \right) \tag{4}$$

The following compromises between the previous two:

$$\text{minimize} \quad \sum_{l \in E} \left(F_l(\overline{x}_l) \right)^n = \sum_{l \in E} \left(\sum_{k \in K} f_l^k(x_l^k) \right)^n, \quad n \ge 1 \tag{5}$$

when $n = 1$, this objective, (5), reduces to (4), while when $n \to \infty$ it reduces to (3). The exponent n is important for the results presented in section IV. Because of (5), even if the cost function $f_l^k(x_l^k)$ is linear instead of (2), the problem becomes a non-linear optimization problem.

C. Handling the delay and loss constraints

Each traffic trunk is associated with an end-to-end delay and loss probability constraint of the traffic (CoS) belonging to the trunk. Hence, the trunk routes must be designed so that these two QoS constraints are satisfied. Given the traffic trunks' route $r = \{l_1, l_2, ..., l_n\}$, the end-to-end delay is bounded by the sum of the propagation delays and the per node delay over the route. The same holds for the end-to-end loss probability. Let d_l^k be the bound on the maximum delay on the link l associated with PHB k, d_l^{prop} is the propagation delay, and p_l^k is the maximum loss probability of that link.

Both the constraints above are constraints on additive path costs under specific link costs (d_l^k and p_l^k respectively). However the problem of finding routes satisfying these constraints is in general NP-complete [7]. Usually, loss probabilities and delay for the same PHB on different nodes are of similar order. Therefore we use the maximum delay and loss probability of a particular PHB over the whole network in order to transform the delay and loss constraints to an upper bound on the number of hops along a route. For a traffic trunk $t \in T$ let H_{D_t}, H_{P_t} denote the upper bound caused by the delay and loss probability constraints respectively.

D. Optimization problem formulation

For each traffic trunk $t \in T$ we denote as R_t the set of (explicit) routes defined to serve this trunk. For each $r_t \in R_t$ we denote as b_{r_t} the capacity we have assigned to this explicit route. We seek to optimize (5), subject to:

$$\sum_{l \in r_t} 1 \leq \min\{H_{P_t}, H_{D_t}\}, \quad \forall r_t \in R_t \tag{6}$$

$$\sum_{r_t \in R_t} b_{r_t} = B_t \tag{7}$$

$$x_l^k = \sum_{t \in T: k_t = k} \sum_{r_t: l \in r_t, r_t \in R_t} b_{r_t} \tag{8}$$

$$b_{r_t} \geq 0 \qquad \forall r_t \in R_t, \forall t \in T \tag{9}$$

Equation (6) is the hop-count constraint, which is the result of the delay and loss QoS requirements of the SLSs. Equation (7) implies that the explicit routes per traffic trunk should be equal to the trunks' capacity requirements. Finally, (8) ties the optimization objective with the optimization parameters. In the formulation above, we have not included any constraint for the physical link capacity. We handle this by defining the cost function (2) to increase very fast when the total link flow is greater than the physical link capacity.

This is a network flow problem and considering the non-linear formulation described above, for the solution we use the general gradient projection method [8]. This is an iterative method, where we start from an initial feasible solution, and at each step we find the minimum first derivative of the cost function path and we shift part of the flow from the other paths to the new path, so that we improve our objective function (5). If the path flow becomes negative, the path flow simply becomes zero in order to handle constraint (9).

The optimization variables are the capacity variables b_{r_t} assigned to each route of each trunk, i.e. $\mathbf{b} = \{b_{r_t} : r_t \in R_t, t \in T\}$. In order to apply the gradient projection method we need to handle all the constraints. Equation (9) is the non-negativity constraint ($\mathbf{b} \geq \mathbf{0}$), while (8) ties the cost function variable x_l^k with the optimization variables b_{r_t}. Thus (5) becomes:

$$\bar{F}(\mathbf{b}) = \sum_{l \in E} \left(\sum_{k \in K} f_l^k \Big(\sum_{t \in T: k_t = k} \sum_{r_t: l \in r_t, r_t \in R_t} b_{r_t} \Big) \right)^n \tag{10}$$

The constraint (7) states that the capacity assigned to each variable b_{r_t} should be equal to the trunks' capacity requirements. Therefore at each iteration i and for each of the trunks $t \in T$, one of the variables b_{r_t} $r_t \in R_t$, say $b_{\bar{r}_t}$, is substituted by $b_{\bar{r}_t} = B_t - \sum_{r_t \in R_t - \{\bar{r}_t\}} b_{r_t}$.

The hop-count constraint (6) is handled as follows. At each step of the algorithm we are required to find a minimum weight path for each trunk $t \in T$ with the weights being the first derivative of the cost function (5). This is the minimum first derivative length path, which should be included in the optimal solution. In order to consider the hop-count constraint, the minimum weight path computation algorithm has to check whether the path found satisfies the hop-count constraint as well. If not, then we need to find another path (not necessarily with the minimum weight but with a total weight less than at least one of the paths in R_t) which has hop-count at least the hop-count constraint. This procedure is done by either using a *k-shortest path* algorithm [9], or by re-trying a shortest path on a reduced graph.

The algorithm above is the main provisioning algorithm we use in this work. This is the first non-linear formulation of the traffic engineering optimization problem, and the only one that combines the objectives in a non-linear manner.

IV. Performance Evaluation

A. Experimental setting

The main topology used is shown Fig. 2. More complex and larger topologies (up to 300 nodes) where also used, according to the models for random topology generation presented in [10]. For the final results we opted for 90% confidence level with a confidence interval 8-10% of the corresponding values.

The initial feasible solution (step 0) of the iterative procedure described in section III.D was set to be the same as if the traffic trunks were to be routed with a shortest path first (SPF) algorithm. That corresponds to the case that all the traffic of a CoS from ingress to an egress is routed through the shortest path. The routing metric used for the SPF algorithm was set to be inversely proportional to the physical link capacity.

We selected the edge nodes to be in 30-60% of the total network nodes. The SLS types offered were multimedia-oriented SLSs (see Table II).

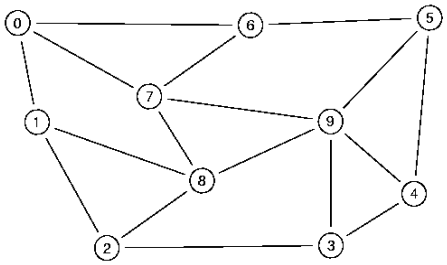

Fig. 2. Fixed topology used for experimentation, 10 nodes –34 links.

We define as the *total throughput* of a network the sum of the capacities of the first-hop links emanating from all edge nodes. This is the upper bound of the throughput, and in reality it is a much greater than the real total throughput a network can handle. In our experiments we used 70% load of the *total throughput*, as the highly loaded condition, and a 40% for medium load. The set of SLSs that adhere to a load profile are assigned randomly at the edge node pairs. SLSs are aggregated per ingress egress pair into a set of traffic trunks.

TABLE II
LOADING PROFILES FOR EACH SLS TYPE

	SLS type		
	Video	Audio	Control Traffic
Medium load 40%	30%	9%	1%
High load 70%	50%	18%	2%

B. Simulation results

Fig. 3 shows the link loads for the 10-node topology shown in Fig. 2 of the first step and after the algorithm has run. It is clear that at step 0 solution, which corresponds to the SPF, parts of the network are over-utilized while others have no traffic at all. After the final step, which corresponds to the final output of our provisioning algorithm, balances the traffic over the whole network. Fig. 4 shows the mean and standard deviation of the link loads for the 10-node network after each iteration of the algorithm. We can observe that the traffic becomes more balanced over the network after each iteration (standard deviation reduces). We run those experiments with the exponent $n = 2$ of (5). The same behavior also observed with larger topologies.

Now we are going to look at the effect of exponent n of the cost function as we defined it in (5). This parameter compromises between the two objectives, minimizing the maximum link load and minimizing the overall cost. Fig. 5 shows the results of the experiment with varying the exponent of the cost function for the 10-node network. We can see that the maximum link load reduces as n increases, while the mean link load slightly increases since minimizing the maximum link utilization results in solutions with paths having more links. The solution for $n = 1$ means that we only optimize (4), and we do not take into account (3). We can observe that the reduction of the maximum link load, and the corresponding increase of the average link load, is important at the first increase of n from 1 to 2, but further increments result only a small change. This is a consequence of the fact that we have the hop-count constraint that limits the number of links per path. The same behavior persists for larger networks.

We calculated the average delay as a function of the link utilization according to the following formula:

$$\frac{s_{avg}}{F_{all}} \sum_{l \in E} \frac{x_l^k}{C_l - x_l^k} \tag{11}$$

where F_{all} is the total in expected rate and s_{avg} is the mean packet size. The above formula is based on the assumption that each queue (PSC) at each link can be modeled as M/M/1 queue [6]. Although, this is not true, we can only use the qualitative nature of the result. Fig. 6 shows the average total delay in seconds of the solutions provided for the various network sizes according to (11). The algorithm manages to keep the average total delay for all network sizes at about the low levels. This is particularly important since at the first step we have many links with utilization more than 100%, which yields very high (infinite) total average delay. Another important observation is that the algorithm results in average delay for medium and high loaded networks being very close, almost proportionally to their relative load difference. This is an indication that even for high loads the algorithm manages to restrain the level of increase of average delay.

V. CONCLUSIONS

Supporting demanding services requires dedicated networks with high switching capacity. In this paper we investigate the possibility of using common IP packet networks, with DiffServ and MPLS, as the key QoS technologies, in order to support such traffic. We proposed a provisioning system for such networks, targeting to support demanding SLSs while at the same time optimizing the use of network resources. We seek to place the CoS demands to the network to avoid overloading parts of the network and minimize the overall network cost. We devised a non-linear programming formulation and we proved though simulation that we achieve our objectives. We believe it is possible to support demanding traffic through IP networks, as long as the appropriate SLSs are defined and agreed. They can then be used to calculate anticipated traffic while the provisioning traffic engineering takes into account the anticipated demand and QoS constraints.

ACKNOWLEDGMENTS

This work was partially supported by the EPSRC/LINK Project GR/M84169 “Production of Broadcast Content in and object-oriented IP-based Network” (PRO-NET), and the EC IST Project IST-1999-11253 “Traffic Engineering for Quality of Service in the Internet at Large” (TEQUILA). The authors would like to thank their project partners for the constructive discussions while working on this paper.

REFERENCES

[1] S. Blake, et al., “An Architecture for Differentiated Services”, Informational RFC-2475, Dec. 1998
[2] E. Rosen, A. Viswanathan, R. Callon, "Multiprotocol Label Switching Architecture", IETF Standards Track RFC-3031, January 2001
[3] D. Awduche, et al., “Requirements for Traffic Engineering Over MPLS”, Informational RFC-2702, Sept. 1999
[4] D. Grossman, “New Terminology for DiffServ”, IETF Internet Draft, draft-ietf-diffserv-new-terms-06.txt, November 2001

[5] P. Trimintzios, et al., "Providing Traffic Engineering Capabilities in IP Networks using Logical Paths", In 8th IFIP Workshop on Performance Modeling and Evaluation of ATM & IP Networks, UK, July 2000
[6] D. Bertsekas, and R. Gallager, *Data Networks*, Prentice Hall, 1992
[7] Z. Wang, and J. Crowcroft, "Quality of Service Routing for Supporting Multimedia Applications", *IEEE JSAC*, vol. 14, no. 7, Sept. 1996
[8] D. Bertsekas, *Nonlinear Programming*, (2nd ed.) Athena Scientific, 1999
[9] D. Eppstein, "Finding k-shortest paths", *SIAM Journal on Computing*, vol.28, no 2, pp.652-673, 1998
[10] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee. "How to model an internetwork", In Proc. INFOCOM 96, vol.2, pp. 594-602, March 1996

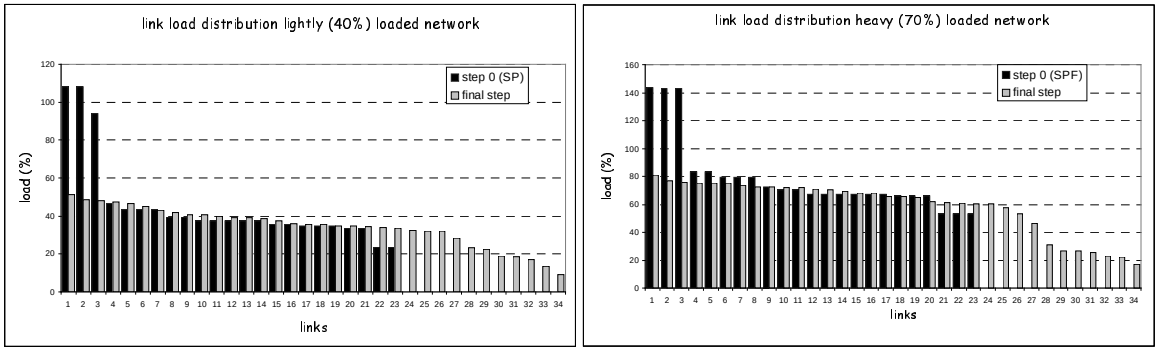

Fig. 3. Link load distribution of after the initial and the last step of the provisioning algorithm for medium and high load with n=2 .

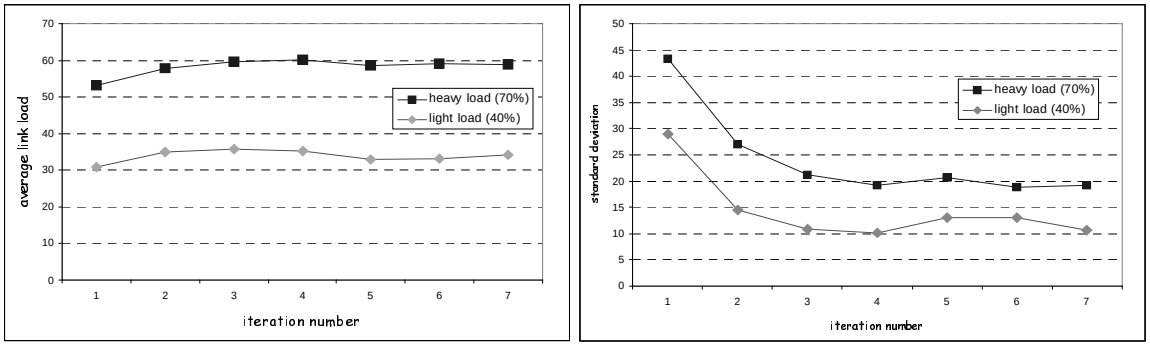

Fig. 4. Average and standard deviation of link load after each of the 7 iterations of the algorithm for medium and high load with n=2 .

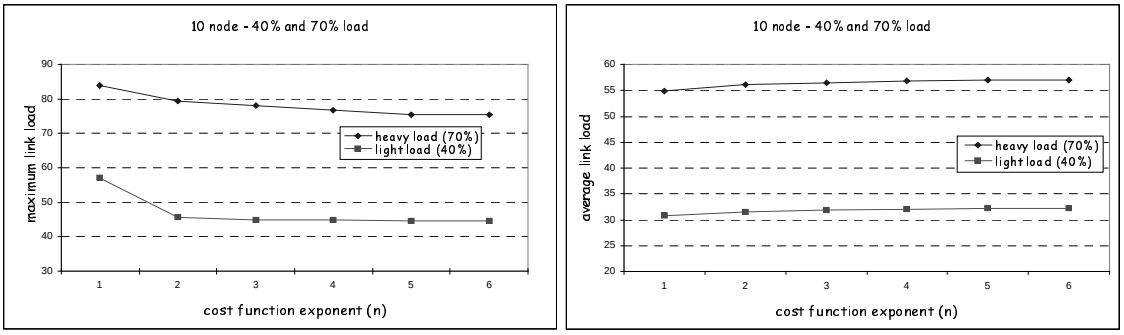

Fig. 5. The effect of changing the cost function exponent n, i.e. compromising between the objectives defined in (3) and (4).

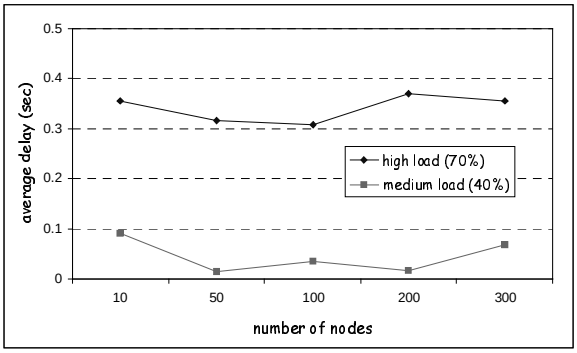

Fig. 6. Average overall network delay for various network sizes with $n = 2$.