Policy-based Network Dimensioning for IP Differentiated Services Networks

Panos Trimintzios[†], Paris Flegkas[†], George Pavlou[†], Leonidas Georgiadis[‡], and David Griffin[§]

[†]Centre for Communication Systems Research, University of Surrey, Guildford, Surrey, GU2 7XH, United Kingdom
Correspondence to: p.trimintzios@ieee.org

[‡]School of Electrical and Computer Engineering, Aristotle University of Thessaloniki, P.O. Box 435, Thessaloniki, 54 124, Greece

[§]Department of Electrical and Electronic Engineering, University College London, Torrington Place, London, WC1E 7JE, United Kingdom

***Abstract*—Given the emergence of IP networks and the Internet as the multi-service network of the future, it is plausible to consider its use for transporting demanding traffic with high bandwidth and low delay and packet loss requirements. Emerging technologies for scalable quality of service such as Differentiated Services and MPLS can be used for premium quality traffic. We are looking at the problem of intra-domain provisioning in an automated manner from an Internet Service Provider's (ISPs) point of view, i.e. we want to satisfy the contracts with our customers while optimizing the use of the network resources. We also need to be able to dynamically guide the behavior of such an automated provisioning system in order to be able to meet the high-level business objectives. The emerging policy-based management paradigm is the means to achieve this requirement. In this paper we devise first a non-linear programming formulation of the traffic engineering problem and show that we can achieve the objectives and meet the requirements of demanding customer traffic through the means of an automated provisioning system. We extend the functionality of the automated system through policies. Finally, we present example scenarios of the enforcement of network dimensioning policies.**

Keywords— **IP, Differentiated Services, Network Dimensioning, Policy-based Networking.**

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. We use these technologies to support premium traffic with stringent QoS requirements, through careful traffic forecasting 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 there is a feasible solution for guaranteeing QoS for demanding premium traffic. In order to provide adequate quality guarantees for demanding traffic over an IP Autonomous System (AS), customers should have contractual Service Level Agreements (SLAs). ISPs on the other hand want to meet the customers' demands as these are described in the Service Level Specification (SLS) [3], which is technical part of an SLA, while at the same time optimizing the use of network resources.

Policy-based Management has been the subject of extensive research over the last decade [4]. Policies are seen as a way to guide the behavior of a distributed system through high-level, declarative directives. We view policy-based management as a means of extending the functionality of management systems dynamically, in conjunction with pre-existing "hard-wired" logic [5]. Policies are defined in a high-level declarative manner and are mapped to low-level system parameters and functions, while the system intelligence can be dynamically modified added and removed by manipulating policies.

The rest of the paper is organized as follows. Section II presents the resource provisioning system architecture together with the policy-based extensions. In section III we present the proposed network dimensioning algorithm and simulation results. In section IV we present policy enforcement examples for network dimensioning. Finally section V, concludes this work and provides suggestions for extending this work.

II. Resource Management Architecture

In [6] we have presented a system for supporting QoS in IP DiffServ Networks. This architecture can be seen as a decomposition of a Bandwidth Broker (BB) realized as a hierarchical, logically and physically distributed system.

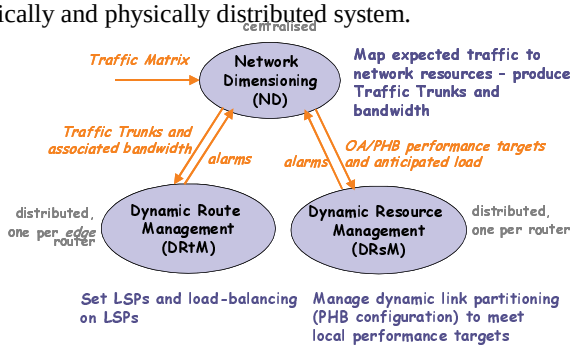

Figure 1 Resource management architecture components

The TE aspects of this architecture are shown in Figure 1. The Network Dimensioning (ND) is responsible for mapping traffic requirements to the physical network resources and for providing provisioning directives in order to accommodate the

predicted traffic demands.

The lower levels intend to manage the resources allocated by ND during the system operation in real-time, in order to react to statistical traffic fluctuations and special arising conditions. They are realized as the Dynamic Route (DRtM) and Dynamic Resource Management (DRsM), which both monitor the network resources and act to medium to short term fluctuations. DRtM operates at the edge nodes and is responsible for managing the routing processes in the network. It influences the parameters based on which the selection of one of the established MPLS Labeled Switched Paths (LSPs) is effected at an edge node with the purpose of load balancing. An instance of DRsM operates at each router and aims to ensure that link capacity is appropriately distributed among the PHBs. It does so by managing the buffer and scheduling parameters. We forecast the anticipated traffic based on the currently subscribed SLSs and on data from measurements. Thus, the provisioning of the network is effectively achieved by both taking into account the long-term service level subscriptions in a time dependent manner (ND) and the dynamic network state (DRtM, DRsM).

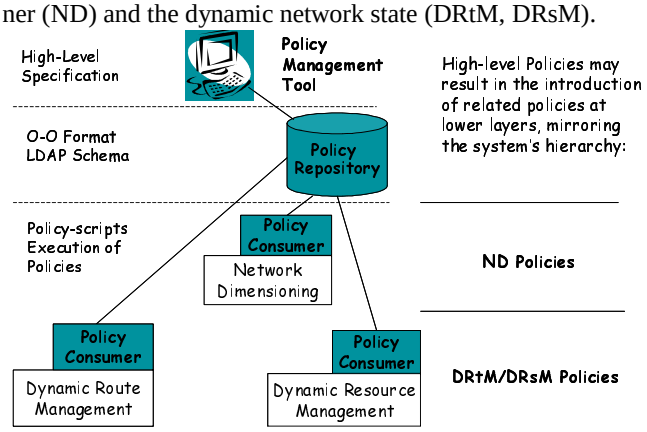

Figure 2 Extensions for enabling policy-based resource management

We extended the traffic engineering system architecture to be able to drive its behavior through policies. The resulting extended system architecture is depicted in Figure 2. The Policy Management extensions include components such as the Policy Management Tool, Policy Repository, and the Policy Consumers. A single Policy Management Tool provides a policy creation environment to the administrator where policies are defined in a high-level declarative language and after validation and static conflict detection tests, they are translated into information objects and stored in a repository. The Policy Repository is a logically centralized component but physically distributed since the technology for implementing this component is the LDAP (Lightweight Directory Access Protocol) Directory. Activation information is passed to the responsible Policy Consumer in order to retrieve and enforce them.

III. NETWORK DIMENSIONING

ND is responsible for the long to medium term configuration of the network resources. By configuration we mean the definition of LSPs as well as the anticipated loading for each PHB on all interfaces, which are subsequently being translated by DRsM into the appropriate scheduling parameters (e.g. priority, weight, rate limits) of the underlying PHB implementation. ND does not provide absolute values but they are in the form of ranges, constituting directives for the function of the PHBs, while for LSPs they are in the form of multiple paths to enable multi-path load balancing. The exact PHB configuration values and the load distribution on the multiple paths are determined by DRsM and DRtM respectively, based on the state of the network, but should always adhere to ND directives.

ND runs periodically, first requesting the predictions for the expected traffic per Ordered Aggregate [7] (OA) in order to be able to compute the provisioning directives. The dimensioning period is in the time scale of a week while the forecasting period is in the time scale of hours. The latter is a period in which we have considerably different predictions as a result of the time schedule of the subscribed SLSs.

The 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 as results of the high-level business policy that dictates the network to be used optimally. The dimensioning functionality is summarized in Figure 3.

Input:
- Topology and link properties (capacity, propagation delay, PHBs)

Pre-processing:
- Request traffic forecast, i.e. the potential traffic trunks (TT)
- Obtain statistics for the performance of each PHB at each link
- Determine the maximum allowable hop count K per TT according to the PHB statistics

Optimisation phase:
- Start with an initial allocation (e.g. using shortest path for each TT)
- Iteratively improve the solution: for each TT find a set of paths:
  - The minimum bandwidth requirements of the TT are met
  - The hop-count constraint K is met
  - The overall cost function is minimized

Post-processing:
- Allocate any extra capacity to the resulted paths of each OA according to resource allocation policies
- Sum the path requirements per link per OA, give minimum (optimisation phase) and maximum (post-processing phase) allocation directives to DRsM
- Give the paths calculated in the optimisation phase to DRtM
- Store the configuration into the Network Repository

Figure 3 Network Dimensioning algorithm overview

A. *Network Dimensioning Algorithm*

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. Ordered Aggregates (OAs), supported by the link. For each OA, $k \in K$ we associate a bound d_l^k (deterministic or probabilistic depending on the OA)

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 ND is the traffic trunk (TT), which is an aggregation of a set of traffic flows characterized by similar edge-to-edge performance requirements [8]. Each traffic trunk is associated with an ingress and one egress node, and is unidirectional. The set of all traffic trunks is denoted by T.

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. However, with the possible exception of heavily loaded conditions, there will generally be multiple feasible solutions. 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. Let x_l^k denote the capacity demand for OA $k \in K$ satisfied by link l and $u_l^k = x_l^k / C_l$ the link utilisation. Then the link cost induced by the load on OA $k \in K$ is a convex function, $f_l^k(u_l^k)$, increasing in u_l^k. The total cost per link is defined as $F_l(\overline{u}_l) = \sum_{k \in K} f_l^k(u_l^k)$, where $\overline{u}_l = \{u_l^k\}_{k \in K}$ is the vector of demands for all OAs of link l. The total cost per link is an approximate function, e.g. $f_l^k(u_l^k) = a_l^k u_l^k$.

We provide an objective that compromises between the two a) and b), that is avoid overloading parts of the network and minimize overall network cost:

$$\text{minimize} \quad \sum_{l \in E} (F_l(\overline{u}_l))^n = \sum_{l \in E} \left(\sum_{k \in K} f_l^k(u_l^k) \right)^n, \quad n \geq 1 \qquad (1)$$

When $n = 1$, the objective (1) reduces to objective a), while when $n \to \infty$ it reduces to b).

Each traffic trunk is associated with an end-to-end delay and loss probability constraint of the traffic belonging to the trunk. Hence, the trunk routes must be designed so that these two constraints are satisfied. Both the constraints above are constraints on additive path costs under specific link costs. However the problem of finding routes satisfying these constraints is, in general, NP-complete [9]. Given that this is only part of the problem we need to address, the problem in its generality is rather complex.

Usually, loss probabilities and delay for the same PHB on different nodes are of similar order. We simplify the optimization problem by transforming the delay and loss requirements into constraints for the maximum hop count for each traffic trunk (TT). This transformation is possible by keeping statistics for the delay and loss rate of the PHBs per link, and by using the maximum, average or n-th quantile in order to derive the maximum hop count constraint.

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 maximize (1), such that the hop-count constraints are met, the explicit routes per traffic trunk should be equal to the trunks' capacity requirements.

This is a network flow problem and considering the non-linear formulation, for the solution we use the general gradient projection method [10]. 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, improving our objective function (1). If the path flow becomes negative, the path flow simply becomes zero. This method is based on the classic unconstraint non-linear optimization theory, and the general point is that we try to decrease the cost function through incremental changes in the path flows.

B. Simulation results

The topologies used for experimentation were random, according to the models for random topology generation presented in [11]. For the final results presented bellow we opted for 90% confidence level, whereas the confidence interval was 8-10% of the corresponding values. The initial solution (step 0) of the iterative procedure is 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 particular class from ingress to an egress is routed through the same shortest path. The routing metric used for the SPF was set to be inversely proportional to the physical link capacity.

The edge nodes were 40-60% of the total network nodes. We defined as the *total throughput* of a network the sum of the capacities of the first-hop links emanating from all edge nodes. We used 70% load of the *total throughput*, as the highly loaded condition, and a 40% for medium load.

Figure 4 shows the maximum of the link load distribution for the different topology and traffic loading profiles. We show the results after the first step and the final step algorithm. 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 dimensioning algorithm, the traffic is balanced over the network.

We can see that the algorithm manages to reduce the maximum link load below 100% for all the cases, while the SPF algorithm gives solutions with more than 300% maximum link load utilization. In these experiments, the standard deviation of

the link load utilization from the average reduces to more than half of that in the case of SPF.

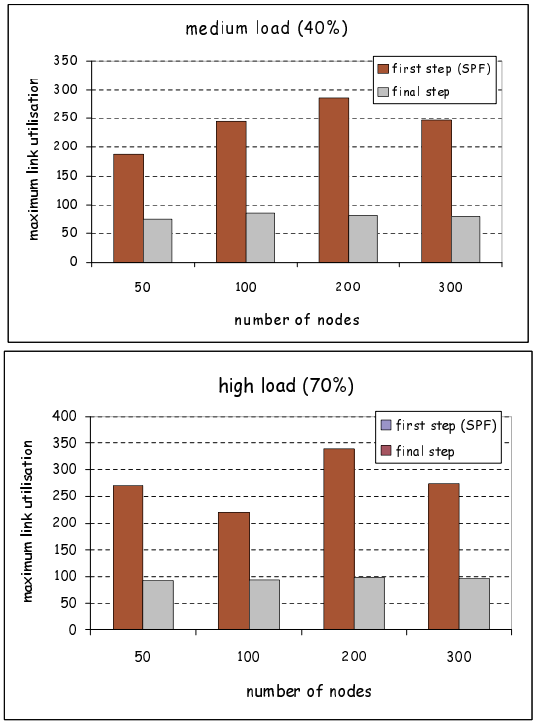

Figure 4. Maximum link load utilisation for medium and highly loaded traffic profile conditions

We run those experiments with the exponent in (1) being $n = 2$. This value compromises between minimizing the total (sum) of link costs and minimizing the maximum link load. In section IV we are going to look at the effect of exponent n of the cost function.

Finally, in Table I we show the average running time of the various experiments conducted. We can see that even for quite large networks the running times are relatively low. For example for 300 node networks, for medium load the running time is about 17 minutes, and for high load about 25 minutes. These times are perfectly acceptable taking into account the time-scales of the ND system operation.

TABLE I: AVERAGE RUNNING TIME IN SECONDS FOR THE VARIOUS NETWORK SIZES

Network Size	Medium load	High load
10	0.055	0.061
50	9.761	10.164
100	123.989	302.079
200	529.532	1002.245
300	981.175	1541.937

IV. POLICY-DRIVEN NETWORK DIMENSIONING

In the architecture shown in Figure 1, ND besides providing long-term guidelines for sharing the network resources, it can also be policy influenced so that its behavior can be modified dynamically at run-time reflecting high-level, business objectives. The critical issue for designing a policy capable resource management component is to specify the parameters influenced by the enforcement of a policy that will result in different allocation of resources in terms of business decisions. These policies that are in fact management logic, are not hard-wired in the component but are downloaded on the fly while the system is operating. The full range of network dimensioning policies can be found in [5]. In the next subsection, we present two examples of the policies demonstrating the way they are being realized by the policy consumer, attached to ND as shown in Figure 2.

In order to demonstrate the results of the enforcement of policies we used a 10-node 36-link random topology and a traffic load of 70 % of the total throughput of the network.

Our first example (P1) concerns a policy rule that wants to create an explicit LSP following the nodes 4, 9, 7, 6 with the bandwidth of the TT being 2 Mbps that is associated with this LSP. The administrator enters the policy rule in the Policy Management Tool using our proprietary policy language, which is then translated in LDAP objects according to an LDAP schema based on the Policy Core LDAP Schema [12] and stored in the Policy Repository. The syntax of our language as well as the extension to the Policy Core Information Model [13] with specific classes that reflect the policies described in the previous section are presented in [5]. The policy rule is entered with the following syntax:

```
If OA==EF and Ingress==4 and Egress==6 then
        Setup_LSP 4-9-7-6 2Mbps                (P1)
```

After this rule is correctly translated and stored in the repository, the Policy Management Tool notifies the Policy Consumer associated with ND that a new policy rule is added in the repository, which then goes and retrieves all the associated objects with this policy rule. From the policy objects the consumer generates code that is interpreted and executed on the fly representing the logic added in our system by the new policy rule. In our implementation, we have chosen TCL as the scripting language due to the ease with which it interfaces with C, since the ND component is implemented in C. Details on the implementation of the Policy Consumer can be found in [5]. The pseudo code of how the above policy is realized by the Policy Consumer is shown in Figure 5.

```
TT_OA: the set of TTs belonging to OA
For each tt_i ϵ TT_OA we get the following:
 v_ingress, v_egress : ingress, egress nodes
 b(tt_i): bandwidth requirement of tt_i
for each tt_i ϵ TT_EF do
   If ((v_ingress == 4) and (v_egress == 6))
     add_LSP ("4-9-7-6", 2000)
     b(tt_i) = b(tt_i) – 2000
   Else
     Policy not executed – TT not found
```

Figure 5. Pseudo-code produced for enforcing (P1)

As it can be seen from the above pseudo-code, it first searches for a TT in the traffic matrix that matches the criteria specified in the conditions of the policy rule regarding the OA, the ingress and egress node. If a TT is found then it executes

the action that creates an LSP with the parameters specified and subtracts the bandwidth requirement of the new LSP from the TT in the traffic matrix file so that the ND algorithm will run for the remaining resources. Note that if the administrator had in mind a particular customer for this LSP then this policy should be refined into a lower level policy enforced on the DRtM component, mapping the address of this customer onto the LSP.

The second example (P2) of a policy rule concerns the effect of the cost function exponent in the capacity allocation of the network. When increasing the cost function exponent the optimization objective that avoids overloading parts of the network is favored. If the administrator would like to keep the load of every link below a certain point then he/she should enter the following policy rule using again our policy notation:

```
If maxLinkLoad>80% then IncreaseExponent 1   (P2)
```

The same procedure explained in the previous example is followed again and the policy consumer enforces this policy by generating a script, which is shown in Figure 6.

```
maxLinkLoad: maximum link load utilization
after the end of the optimization algorithm
n: cost function exponent (initially = 1)
optimization_algorithm n
while (maxLinkLoad > 80 )
    n = n+1
    optimization_algorithm n
```

Figure 6 Pseudo-code produced for enforcing (P2)

As it can be observed from Figure 7 the enforcement of the policy rule caused the optimization algorithm to run for 4 times until the maximum link load utilisation at the final step drops below 80%. The exponent value that achieved the policy objective was $n = 4$.

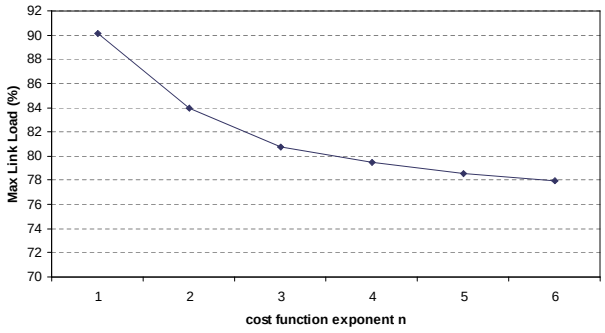

Figure 7 Effect of the cost function exponent on the maximum link load utilisation

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 provision the network for such traffic. We proposed an automated provisioning system, targeting to support demanding SLSs while at the same time optimizing the use of network resources. We seek to place the traffic demands to the network in such a way as to avoid overloading parts of the networks and minimize the overall network cost. We devised a non-linear programming formulation and we proved though simulation that we achieve our objectives. Moreover, we presented how this system can be policy-driven and described the components of necessary policy-based system extensions that need to be deployed in order to enhance or modify the functionality of policy influenced components reflecting high-level business decisions.

As a continuation of this work, we are focusing on defining policies for the rest of the components of the TE system and explore the refinement of policies entered at the ND to lower level policies that apply to Dynamic Resource and Route Management components forming a policy hierarchy.

ACKNOWLEDGMENT

This work was partially supported by the EC IST Project IST-1999-11253 “Traffic Engineering for Quality of Service in the Internet at Large” (TEQUILA) and the EPSRC/LINK Project GR/M84169 “Production of Broadcast Content in and object-oriented IP-based Network” (PRO-NET). 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”, IETF Informational RFC-2475, December 1998

[2] E. Rosen, A. Viswanathan, R. Callon, "Multiprotocol Label Switching Architecture", IETF Standards Track RFC-3031, January 2001

[3] D. Goderis, *et al.* “Service Level Specification Semantics and Parameters”, IETF draft-tequila-sls-01.txt, December 2001 (available at: www.ist-tequia.org/sls)

[4] M. Sloman, “Policy Driven Management For Distributed Systems”, *Journal of Network and Systems Management*, vol. 2, no. 4, pp. 333-360, Plenum Publishing, December 1994.

[5] P. Flegkas, P. Trimintzios, G. Pavlou, “A Policy-based Quality of Service Management Architecture for IP DiffServ Networks”, *IEEE Network Magazine*, vol. 16, no. 2, pp. 50-56, March/April 2002.

[6] P. Trimintzios, *et al.* “A Management and Control Architecture for Providing IP Differentiated Services in MPLS-based Networks”, *IEEE Communications Magazine*, vol. 39, no. 5, May 2001

[7] D. Grossman “New Terminology and Clarifications for DiffServ”, IETF Informational RFC 3260, April, 2002

[8] T. Li, and Y. Rekhter, “Provider Architecture for Differentiated Services and Traffic Engineering (PASTE)” IETF RFC-2430, October 1998

[9] Z. Wang, and J. Crowcroft, “Quality of Service Routing for Supporting Multimedia Applications”, *IEEE Journal of Selected Areas in Communications*, vol. 14, no. 7, pp. 1228-1234, September 1996

[10] D. Bertsekas, *Nonlinear Programming*, Athena Scientific, 1999

[11] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee, “How to model an internetwork”, In Proceedings of IEEE INFOCOM 96, vol.2, pp. 594-602, San Francisco, March 1996

[12] J. Strassner, *et al.*, “Policy Core LDAP Schema”, IETF draft-ietf-policy-core-schema-14.txt, January 2002

[13] B. Moore, *et al.*, “Policy Core Information Model – Version 1 Specification”, IETF Standards Track RFC-3060, February 2001.