Policy-driven Traffic Engineering for Intra-domain Quality of Service Provisioning

Panos Trimintzios, Paris Flegkas, and George Pavlou

Centre for Communication Systems Research
School of Electronics and Physical Sciences
University of Surrey, Guildford, Surrey, GU2 7XH, U.K.
{P.Trimintzios, P.Flegkas, G.Pavlou}@eim.surrey.ac.uk

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 optimising the use of the network resources. We need to be able to dynamically guide the behaviour 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. We define resource provisioning policies, and we present example scenarios of their enforcement.

1 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 premium traffic with stringent QoS requirements. This can be done 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 this 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), we propose to use the DiffServ framework together with MPLS for Traffic Engineering (TE). Customers 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 optimising 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 behaviour of a network or 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 organised as follows. Section 2 presents the Traffic Engineering and resource provisioning system architecture together with the policy-based extensions. In section 3 we present our network dimensioning algorithm and the corresponding simulation results. In section 4 we enlist potential policies related to network dimensioning and we present policy enforcement examples. Section 5 presents the related work and finally section 6, concludes and suggests extensions of this work.

2 Architecture

In [6] we have designed a system for supporting QoS in IP DiffServ Networks. This architecture can be seen as a detailed decomposition of the concept of an extended Bandwidth Broker (BB) realized as a hierarchical, logically and physically distributed system. A detailed description can be found in [6].

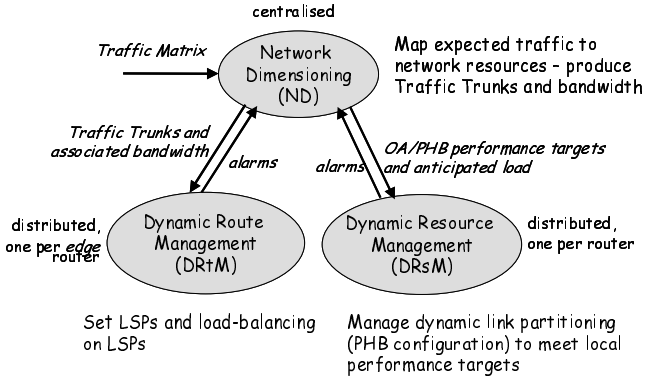

Fig. 1 The components of the Traffic Engineering system

The Traffic Engineering (TE) aspects of this architecture are shown in Fig. 1 The Network Dimensioning (ND) component is responsible for mapping traffic requirements to the physical network resources and for providing Network Dimensioning directives in order to accommodate the predicted traffic demands. The lower level of the traffic engineering part intends to dynamically manage the resources allocated by Network Dimensioning during the system operation in real-time, in order to react to

statistical traffic fluctuations and special arising conditions. This part is realized by 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 mainly influences the parameters based on which the selection of one of the established MPLS Labelled 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 in that link. It does so by managing the buffer and scheduling parameters according to the guidelines provided by ND. 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.

We extended the traffic engineering system to be able to drive its behaviour through policies. The resulting extended system architecture is depicted in Fig. 2. The Policy extensions include components such as the Policy Management Tool, Policy Repository, and the Policy Consumers. A single Policy Management Tool exists for providing 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 object-oriented representation (information objects) and stored in a repository. The Policy Repository may be physically distributed since the technology for implementing this component is the LDAP (Lightweight Directory Access Protocol) Directory. After policies are stored, activation information is passed to the responsible Policy Consumer in order to retrieve and enforce them.

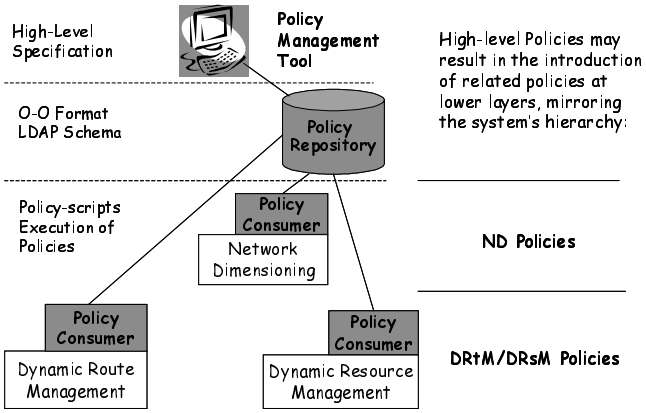

Fig. 2 Policy-driven Traffic Engineering Architecture

The methodology for applying policies to a hierarchically distributed system, like our architecture, is described in detail in [5]. Our model assumes many instances of Policy Consumers, every instance attached to the component that is influenced by the policies this Policy Consumer is responsible to enforce, complementing the static management intelligence of the above layer of hierarchy. Policies may be introduced at every layer of our system but higher-level policies may possibly result in the introduction of related policies at lower levels, mirroring the system hierarchy.

Input:

Network topology, link properties (capacity, propagation delay, supported 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 maximum allowable hop count K per TT according to PHB statistics

Optimisation phase:

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

Post-processing:

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

Fig. 3 The basic Network Dimensioning functionality

3 Network Dimensioning

Network Dimensioning (ND) performs the provisioning and 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. The values provided by ND are not absolute but are in the form of a range; 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 the ND directives.

ND runs periodically, getting 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. For example, ND might run every Sunday evening and provide multiple configurations i.e. one for each period of each day of the week (morning, evening, night). So, effectively the resource provisioning cycle is at the same time scale of the forecasting period.

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, more specifically they are results of the high-level business policy that dictates the network should be used in an optimally. The basic Network Dimensioning functionality is summarised in Fig. 3.

3.1 Network Dimensioning Algorithm

The network is modelled 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). A traffic trunk is an aggregation of a set of traffic flows characterized by similar edge-to-edge performance requirements [8]. Also, each traffic trunk is associated with one ingress node and one egress node, and is unidirectional. The set of all traffic trunks is denoted by T. Each trunk $t \in T$, from ingress node v_i to egress node v_e ($v_i, v_e \in V$), is associated with bandwidth requirements in the form of a minimum $B_t^{\min}$ and a maximum $B_t^{\max}$, where the minimum represents the requirement of the currently subscribed SLSs aggregation, and the maximum reflects the over-provisioning policies. We view TTs as the abstract representation of traffic with specific characteristics.

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, and (b) provide overall low network load (cost).

The last two requirements do not lead to the same optimisation 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 must be 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. Then the link cost induced by the load on OA $k \in K$ is a convex function, $f_l^k(x_l^k)$, increasing in x_l^k. The total link cost per link is defined as: $F_l(\bar{x}_l) = \sum_{k \in K} f_l^k(x_l^k)$, where $\bar{x}_l = \{x_l^k\}_{k \in K}$ is the vector of demands for all OAs 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.

Provided that 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 has the following drawbacks: the cost definition depends on the PHB implementation at the routers, and the cost functions may not be known, or may be too complex. Hence we are using approximate functions, e.g. $f_l^k(x_l^k) = a_l^k x_l^k$.

We can now formally define the objectives (a) and (b) described above as follows:
Avoid overloading parts of the network:

$$\text{minimize} \quad \max_{l \in E} F_l(\bar{x}_l) = \max_{l \in E} \left\{ \sum_{k \in K} f_l^k(x_l^k) \right\} \tag{1}$$

Minimize overall network cost:

$$\text{minimize} \quad \sum_{l \in E} F_l(\bar{x}_l) = \sum_{l \in E} \left(\sum_{k \in K} f_l^k(x_l^k) \right) \tag{2}$$

We provide an objective that compromises between the previous two:

$$\text{minimize} \quad \sum_{l \in E} \left(F_l(\bar{x}_l)\right)^n = \sum_{l \in E} \left(\sum_{k \in K} f_l^k(x_l^k) \right)^n, \quad n \geq 1 \tag{3}$$

When $n = 1$, the objective (3) reduces to (2), while when $n \to \infty$ it reduces to (1). Because of (3), even if the cost function $f_l^k(x_l^k)$ is linear function of load, the problem remains still a non-linear optimisation problem.

Handling the delay and loss constraints. 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 (d_l^k and p_l^k respectively). 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 optimisation 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. By using the maximum we are too conservative (appropriate for EF traffic), while by using an average we possibly underestimate the QoS requirements e.g. for AF traffic we may use the 80-th percentile. The accuracy of the statistics is determined by the period used to obtain them, methods like exponential weighted moving average over long period must be used.

Optimisation problem. 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 optimise (3), 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 described above, for the solution we make use of 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, so that we improve our objective function (3). If the path flow becomes negative, the path flow simply becomes zero. This method is based on the classic unconstraint non-linear optimisation theory, and the general point is that we try to decrease the cost function through incremental changes in the path flows.

The optimisation variables are the capacity variables b_{r_t} assigned to each route of each trunk, i.e. $\mathbf{b} = \{b_{r_t} : r_t \hat{I} R_t, t \hat{I} T\}$. In order to apply the gradient projection method we need to handle all the constraints. The non-negativity constraint, is handled by defining a cost function which increases very fast after $x_l \ ^3 \ C_l$. At each iteration i and for each of the trunks $t \hat{I} T$, one of the variables b_{r_t} $r_t \hat{I} R_t$, say $b_{\bar{r}_t}$, is substituted by $b_{\bar{r}_t} = B_t - \sum_{r_t \hat{I} R_t - \{\bar{r}_t\}} b_{r_t}$, in order to equal the .capacity assigned to each variable b_{r_t} to the trunks' capacity requirements.

The hop-count constraint is handled as follows. At each step of the algorithm we are required to find a minimum weight path for each trunk $t \hat{I} T$ with the weights being the first derivative of the cost function (3). The minimum weight path computation algorithm has to check whether the path found satisfies the hop-count constraint. 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) that meets the hop-count constraint. This can be achieved by using a *k-shortest path* algorithm [11].

We control the iterative procedure described above by ending the iterations when the relative improvement in a step $i + 1$ from step i, is less than a parameter ε. More specifically the iterative procedure terminates when

$$\left| \frac{F(\mathbf{b}^{i+1}) - F(\mathbf{b}^{i})}{F(\mathbf{b}^{i+1})} \right| < \varepsilon . \qquad (4)$$

3.2 Simulation results

The topologies used for experimentation were random, according to the models for random topology generation presented in [12]. We opted for a 90% confidence level and achieved confidence interval of about 8-10% of the corresponding values. The initial solution (step 0) of the iterative procedure of the ND algorithm is as if the traffic trunks were to be routed with a shortest path first (SPF) algorithm. This corresponds to the case where all traffic of a particular class 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. The parameter ε was set to 0.001.

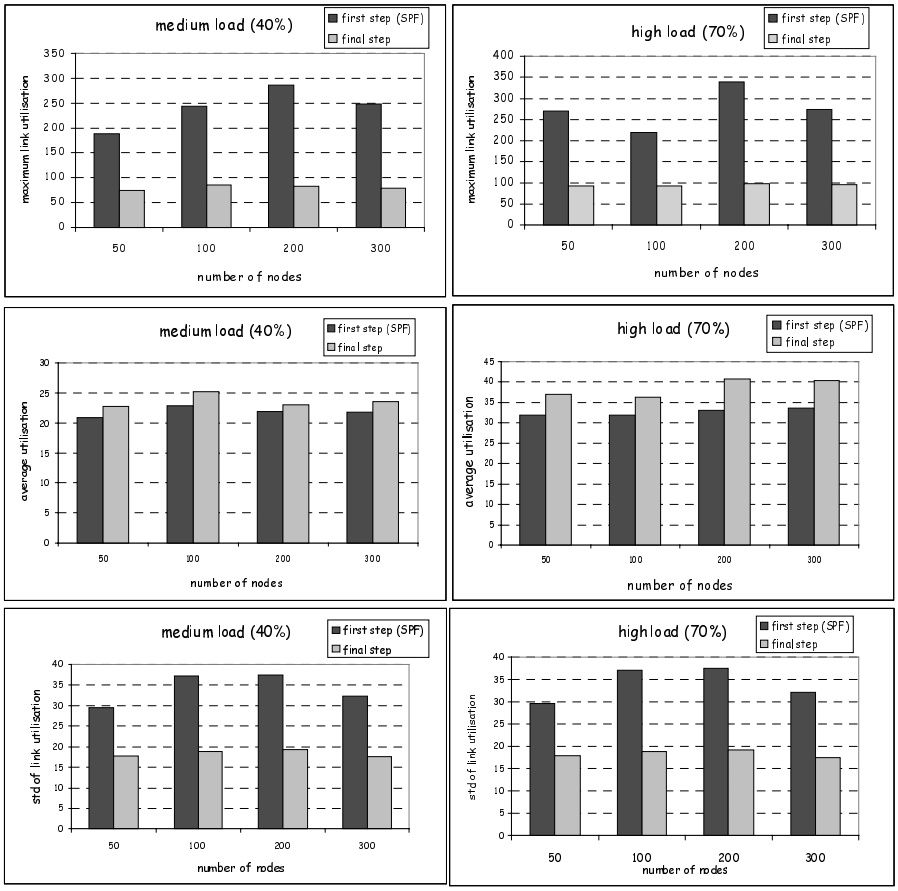

Fig. 4 Maximum, average and standard deviation of link load utilisation for different network sizes and network loading profiles

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. 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, since the sum of the first-hop link capacity imposes the limit, but the rest of the backbone might not be able to handle that traffic. In our experiments we used 70% load of the *total throughput*, as the highly loaded condition, and a 40% for medium load.

Fig. 4 shows the maximum, average and standard deviation 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 utilisation.

The average link utilisation increases slightly, since the algorithm uses paths with more number of links than the shortest path, and therefore the same load is accounted in more links than in the SPF case. We can also see that the traffic load is balanced over the network since the standard deviation of the link load utilisation from the average reduces to more than half of that in the case of SPF.

We run those experiments with the exponent in equation (3) being $n = 2$. This value compromises between minimizing the total (sum) of link costs and minimizing the maximum link load. In section 4.2 we are going to look at the effect of exponent n of the cost function.

Finally, in Table 1 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 timescales of the ND system operation.

Table 1 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

4 Policy-driven Network Dimensioning

In the architecture shown in Fig. 1, ND besides providing long-term guidelines for sharing the network resources, it can also be policy influenced so that its behaviour 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.

4.1 Network Dimensioning Policies

We identify two categories of policies, *initialisation policies*, which concern policies that result in providing initial values to variables, which are essential for the functionality of ND and do not depend on any state but just reflect decisions of the policy administrator. The second category, *resource provisioning policies*, concerns those that influence the way it calculates the capacity allocation and the path creation configuration of the network. Such policies are those that their execution is based on the input from the traffic forecast module and on the resulting configuration of the network.

Since dimensioning is triggered mainly periodically, the policy administrator should specify this period. The priority of this policy should be specified in order not to cause any inconsistencies when re-dimensioning is triggered by notifications sent from the dynamic control parts of the system, that is when DRtM and DRsM are unable to perform an adaptation of the network with the current configuration. Another parameter that should be defined by policies is the cost-function used by ND. The administrator should be able either to choose between a number of pre-specified cost functions and/or setting values to parameters in the desired function. For example, if the approximate cost function used by ND is linear to the bandwidth allocated to a PHB, that is $f_l^k(x_l^k) = a_l^k x_l^k$ where x_l^k is the bandwidth allocated to PHB k on link l and a_l^k is a constant, the value of this constant could be specified by the policy administrator depending on the cost, i.e. importance, of a particular PHB. Another constraint imposed by policies is the maximum number of alternative paths that ND defines for every traffic trunk for the purpose of load balancing. Finally, the exponent n , as defined in equation (3), is another parameter that is specified by policies allowing the administrator to choose the relative merit of low overall cost and avoiding overload parts of the network.

The policy administrator should be able to specify the amount of network resources (giving a minimum, maximum or a range) that should be allocated to each OA. This will cause ND to take into account this policy when calculating the new configuration for this OA. More specifically, ND should allocate resources in a way that does not violate the policy and then calculate the configuration taking into account the remaining resources. A more flexible option should be for the policy administrator to indicate how the resources should be shared in specific (critical) links. After the optimisation phase ends, ND enters a post-processing stage where it will try to assign the residual physical capacity to the various OAs. This distribution of spare capacity is left to be defined by policies that indicate whether it should be done proportionally to the way resources are already allocated or it can be explicitly defined for every traffic class. A related policy is to specify the way the capacity allocated to each OA should be reduced because the link capacity is not enough to satisfy the predicted traffic requirements. ND actually translates the delay and loss requirements on an upper bound on the number of hops per route, the way this translation is done can also be influenced by policy rules. For example, the safest approach to satisfy the TT requirements would be to assume that every link and node belonging to the route induces a delay equal to the maximum delay caused by a link and node along the route. So, this policy rule will allow the administrator to decide if the maximum, average or minimum delay or loss induced by a node or link along the route should be used to derive the hop count constraint. Policies that allow the administrator for a particular reason to explicitly specify an LSP that a TT should follow can also be defined. Of course, this should override the algorithm's decision about the creation of the LSP for this TT.

4.2 Policy Enforcement Examples

In order to demonstrate the results of the enforcement of policies we used the topology depicted in the Fig. 5 as input to the ND component and a traffic load of 70 % of the total throughput of the network as defined in section 3.2.

Fig. 5 Topology used for the policy examples

Our first example (P1) is a policy 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 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 [13] and stored in the Policy Repository. The syntax of our language as well as the extension to the Policy Core Information Model [14] with specific classes that reflect the policies described in the previous section are presented in [5]. The policy is entered with the following syntax:

If OA=EF and Ingress=4 and Egress=6 **then** SetupLSP 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 retrieves all the associated objects with this policy rule. From these objects the consumer generates code that is interpreted and executed representing the logic added in our system by the new policy rule. The pseudo-code produced by the Policy Consumer for (P1) is shown in Fig. 6. It 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. Note that if the administrator had in mind a particular customer then this policy should be refined into a lower level policy enforced on DRtM, mapping the address of this customer onto the LSP.

```
TT_OA: the set of TTs belonging an OA
v_i, v_e: ingress, egress nodes
b(tt): bandwidth requirement of tt

for each tt ∈ TT_EF do
  if ((v_i == 4) and (v_e == 6))
    add_lsp ('4-9-7-6', 2000)
    b(tt) = b(tt) - 2000
  else
    policy failed - TT not found
```

(P1)

```
maxLinkLoad: maximum link load
utilisation after optimisation
n=1: cost function exponent


optimisation_algorithm n
while (maxLinkLoad > 80 )
  n = n+1
  optimisation_algorithm n
```

(P2)

Fig. 6 Pseudo-code produced for enforcing policy (P1) and policy (P2)

The second example (P2) of a policy rule concerns the effect of the cost function exponent in the capacity allocation of the network. As we mentioned earlier by increasing the cost function exponent the optimisation objective that avoids overloading parts of the network is favoured. 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 in our system by using our policy notation as:

```
If maxLinkLoad > 80% then Increase Exponent by 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 Fig. 6.

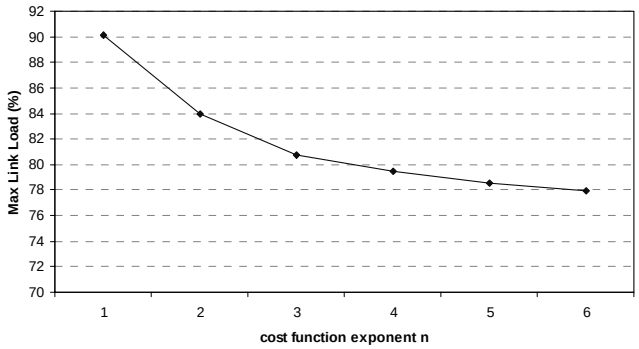

Fig. 7 Effect of the cost function exponent on the maximum link load utilisation

As it can be observed from Fig. 7 the enforcement of the policy rule caused the optimisation algorithm to run for 4 times until the maximum link load utilisation at the final step drops below 80%. The policy objective is achieved when $n = 4$.

5 Related Work

Next we describe some of the related works. This should not be considered an exhaustive survey of all related works. We have to mention that none of the previous works used a non-linear formulation and solution, and none of them considered and used the policy framework implications, as we do in this work.

The Traffic Engineering working group of the Internet Engineering Task Force (IETF) has chartered to define, develop, and specify principles, techniques, and mechanisms for traffic engineering in the Internet. Their main output up to now, is the definition of the basic principles for traffic engineering [15], and the requirements to support the interoperation of MPLS and Diffserv for traffic engineering [16].

Two similar works with the work presented in this paper are Netscope [17] and RATES [18]. Both of them try to automate the configuration of the network in order to maximise the network utilisation. The first one uses measurements to derive the traffic demands and then by employing the offline algorithm described in [19] tries to offload overloaded links. The latter uses a semi-online algorithm described in [20] to find the critical links which if they are chosen for routing would cause the greatest interference (i.e. reduce the maximum flow) of the other egress-ingress pairs of the net-

network. Both of these works do not take into account any QoS requirements of the traffic and try only to minimize the maximum load of certain links.

The work described in this paper can be categorised as a time-dependent offline traffic engineering [15] Such problems can modelled as multi-commodity network flow optimisation problems [21]. The related works use such optimisation formulations, focusing on the use of linear cost functions, usually the sum of bandwidth requirements, and in most of the cases try to optimise a single criterion, minimize total network cost, or combine multiple criteria in a linear formula.

In Mitra et al [22] the traffic-engineering problem is seen as a multi-priority problem, which is formulated as a multi criterion optimisation problem on a predefined traffic matrix. This approach uses the notion of predefined admissible routes, and the objective is to maximise the carried bandwidth. The main objective of [23], is to design Virtual Private networks (VPNs), which will have allocated bandwidth on the links such that, when the traffic of a customer is optimally routed, a weighted aggregate measure over the service provider's infrastructure is maximized, subject to the constraint that each VPN carries a specified minimum. The weighted measure is the network revenue is the sum of the linear capacity costs for all links.

In [24] a model is proposed for off-line centralized traffic engineering over MPLS. The traffic engineering uses the following objectives: resource-oriented or traffic-oriented traffic engineering. The resource-oriented problem targets to load balance and minimise the resource usage. The objective function is a linear combination of capacity usage and load balancing, subject to link capacity constraints. The traffic-oriented model suggests an objective function that is a linear combination of fairness and throughput, where throughput is the total bandwidth guaranteed by the network and fairness as the minimum weighted capacity allocated to a traffic trunk. In [25] the authors propose an algorithm, which has a pre-processing phase and an on-line phase. In the pre-processing the goal is to find paths in the network to accommodate as much traffic of the traffic classes as possible from the source to the destination node. The algorithm minimizes a link cost function of the bandwidth assigned to each link for a traffic class. The second phase performs the on-line path selection for LSPs requests by using the pre-computed output of the multi-commodity pre-processing phase.

Works like [19], [26], and [27] try to achieve the optimal routing behaviour by appropriately configuring the shortest path routing metric. Wang et al. in [26] proved theoretically that any routing configuration, including the optimal one, could be achieved by the appropriate setting of the shortest path routing metric.

Finally, as far as online traffic engineering algorithms is concerned, they are mainly extensions of the QoS-routing paradigm [28]. These approaches are heuristics, known as Constraint-Shortest Path (CSPF). They utilise information kept in traffic engineering databases, which are populated from the routing flooding mechanisms, about link capacities, unreserved capacity, etc. Other online traffic engineering approaches, e.g. [29], [30], focus on load balancing on multiple equal or non-equal cost paths. These works are complementary to this work, since they can be used in conjunction (e.g. as parts of DRtM or DRsM functionality).

6 Conclusions and Further Work

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 optimising 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.

We showed 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. We enlisted the policy parameters that influence the behaviour of dimensioning and presented enforcement of two example policies. As a continuation of this work, we will be focusing on defining policies for the rest of the components of the TE system and explore the issue of the refinement of policies to lower level policies forming a policy hierarchy. Also we intend to look at conflict detection and resolution mechanisms specific to our problem domain.

Acknowledgements

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, 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 Informational RFC-2430, October 1998
[9] Z. Wang, and J. Crowcroft, "Quality of Service Routing for Supporting Multimedia Applications", *IEEE JSAC*, vol. 14, no. 7, pp. 1228-1234, September 1996
[10] D. Bertsekas, *Nonlinear Programming*, (2nd ed.) Athena Scientific, 1999
[11] D. Eppstein, "Finding k-shortest paths", *SIAM J. on .Computing*, vol.28, no 2, pp.652-673, 1998
[12] 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, USA, March 1996
[13] J. Strassner, et al., "Policy Core LDAP Schema", IETF draft-ietf-policy-core-schema-14.txt, January 2002
[14] B. Moore et al., "Policy Core Information Model – Version 1 Specification", IETF Standards Track RFC-3060, February 2001
[15] D. Awduche, et al. "Overview and Principles of Internet Traffic Engineering", IETF Informational RFC-3272, May 2002
[16] F. Le Faucheur, et al. "Requirements for support of Diff-Serv-aware MPLS Traffic Engineering", IETF Internet draft, <draft-ietf-tewg-diff-te-reqts-05.txt>, work in progress, June 2002
[17] A. Feldmann and J. Rexford, "IP Network Configuration for Intradomain Traffic Engineering", *IEEE Network Magazine*, vol. 15, no. 5, pp. 46-57, September 2001
[18] P. Aukia, et al. "RATES: A Server for MPLS Traffic Engineering", *IEEE Network Magazine*, vol. 14, no. 2, pp. 34-41, March 2000
[19] B. Fortz, and M. Thorup, "Internet Traffic Engineering by Optimizing {OSPF} Weights", In Proc. of IEEE INFOCOM 2000, pp. 519-528, Israel, March 2000
[20] M. Kodialam, and T.V. Lakshman, "Minimum Interference Routing with Applications to Traffic Engineering", in Proc. IEEE INFOCOM00, pp. 884-893 March 2000
[21] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin, *Network Flows: Theory, Algorithms and Applications*, Prentice Hall, 1993
[22] D. Mitra, and K. G. Ramakrishnan, "A Case Study of Multiservice, Multipriority Traffic Engineering Design for Data Networks", In Proc. IEEE GLOBECOM 99, pp. 1087-1093, Brazil, December 1999
[23] D. Mitra, J.A. Morrison and K.G. Ramakrishnan, "Virtual Private Networks: Joint Resource Allocation and Routing Design", In Proc. IEEE INFOCOM 99, USA, March 1999
[24] F. Poppe, et al. "Choosing the Objectives for Traffic Engineering in IP Backbone Networks Based on Quality-of-Service Requirements", In Proc. Workshop on Quality of future Internet Services (QofIS'00), pp. 129-140, Germany, September 2000
[25] S. Suri, et al. "Profile-based Routing: A New Framework for MPLS Traffic Engineering", In Proc. of the 2nd International Workshop on Quality of future Internet Services (QofIS'01), pp. 138-157, Portugal, September 2001
[26] Z. Wang, Y. Wang, and L. Zhang, "Internet Traffic Engineering without Full Mesh Overlaying", In Proc. of IEEE INFOCOM 2001, Alaska, April 2001
[27] Y. Breitbart, M. Garofalakis, A. Kumar and R. Rastogi, "Optimal Configuration of OSPF Aggregates", In Proc. of IEEE INFOCOM02, New York, USA, June 2002
[28] S. Chen, K. Nahrstedt, "An Overview of Quality-of-Service Routing for the Next Generation High-Speed Networks: Problems and Solutions", *IEEE Network Magazine*, vol. 12, no. 6, pp. 64-79, November 1998
[29] A. Elwalid, C. Jin, S H. Low, and I. Widjaja, "MATE: MPLS Adaptive Traffic Engineering", In Proc. of IEEE INFOCOM2001, pp. 1300-1309, Alaska, USA, April 2001
[30] Z. Cao, Z. Wang, and E. Zegura, "Performance of Hashing-based Schemes for Inter-