Efficient agent-based selection of DiffServ SLAs over MPLS networks within the ASP service model[*]

Thanasis G. Papaioannou[a], Stelios Sartzetakis[a,c] and George D. Stamoulis[a,b]

[a] Institute of Computer Science (ICS), Foundation for Research & Technology – Hellas (FORTH)
P.O. Box 1385, GR 711 10, Heraklion, Crete, Greece

[b] Department of Informatics, Athens University of Economics and Business (AUEB), Greece

[c] Correspondence: S.Sartzetakis (e-mail: stelios@ics.forth.gr, telephone: +30-81-391729, telefax: +30-81-391601) – Other author e-mail: {pathan, gstamoul}@ics.forth.gr

ABSTRACT

The demand for QoS provisioning support over Internet grows continuously. One of the factors contributing to this demand is the increasing penetration of Application Service Providers (ASPs) to the market. This necessitates the development of mechanisms for the efficient realization of Service Level Agreements (SLA). In this paper, we develop and evaluate an approach for efficient SLA selection and implementation (support, policing/shaping, and charging) in a DiffServ-over-MPLS network domain. We describe how this approach is applied in a realistic service provision scenario based on the ASP service model. A negotiation process between a user and a network provider is introduced; thus the user can choose from the alternative options for allocation of resources the one that better matches his needs. For the purposes of negotiation, we develop an appropriate utility model that expresses user preferences in a simple yet informative way. Furthermore, we discuss the implementation of our approach in a small-scale experimental DiffServ-over-MPLS network, for the case of a simple scenario of ASP services provision. We also assess the economic efficiency of our approach by means of simulation experiments, the results of which advocate that our approach is incentive compatible, in the sense that individual optimization by each user (in SLA selection) also leads to improved social welfare. Our approach is quite general and can be combined with several policies for network management, or as a complement to the traffic engineering procedures.

[*] A preliminary version of this work was presented at *SPIE International Symposium on Information Technologies 2000 (Program on Internet Performance and Control)*, Boston, USA, November 2000. This work was supported in part by the European Commission under IST project M3I (IST-1999-11429).

Keywords: SLA negotiation, utility, ASP, DiffServ

Running Header: Efficient Selection of DiffServ SLAs

1. INTRODUCTION

The continuously growing demand for Quality of Service (QoS) provisioning support over Internet has led researchers to define various possible solutions to this problem. The most scalable and less demanding solution, in terms of modifications to the existing Internet infrastructure, is the Differentiated Services (DiffServ) architecture [1]. It is summarized in applying a forwarding Per-Hop Behavior (PHB) to traffic aggregates at each hop, through a code-point assignment at every IP flow[1] according to a Service Level Agreement (SLA). While this approach is attractive for its scalability properties it has certain drawbacks, since little care has been taken for QoS provisioning on a per-application or a per-user basis. For this reason, Integrated Services (IntServ) [2] operation over DiffServ has been proposed [3]. Under this approach, DiffServ domains are viewed as single hops in the Resource Reservation Setup Protocol (RSVP) [4] process. Within the DiffServ domains, bandwidth for the traffic aggregates is reserved either statically or using RSVP (per traffic flow or per traffic aggregate [5]) or/and Bandwidth Brokers, as in [6], where the resource allocation process for end-to-end (across domain boundaries) SLA provision is addressed.

Multi-Protocol Label Switching (MPLS) [7] supports the notion of traffic aggregates and furthermore it can offer different QoS level for each traffic aggregate. At the same time, MPLS provides some very important features that facilitate traffic engineering. Therefore, DiffServ over MPLS seems to be a very promising approach for efficient QoS provisioning in terms of both scalability and flexibility.

An SLA contains among others the Service Level Specification (SLS)[8], which can be viewed as the traffic contract. The SLS contains guarantees regarding the throughput, the packet loss probability, or the delay experienced by the traffic; such guarantees can be qualitative (e.g. low, high) and/or quantitative (e.g. a certain peak rate). While the specification of the SLS has to be as technical and detailed as possible, in order for the network provider to accomplish resource allocation as needed, it should also be made easy for the user to specify or select a predefined SLS in a simple way. Furthermore, the SLSs have to be

[1]An IP flow denotes the packets flowing through an IP connection, which is uniquely identified by the tupple of parameters (source IP address, source port, destination IP address, destination port, transport protocol). For convenience, IP flow will be referred as "flow" in the rest of the paper.

negotiated between different administrative domains as well as between administrative domains and users. Only thus users can select the SLSs they really need and the network can allocate its resources in an efficient way, according to its particular policy. In fact, this should be done in such a way that both goals are met at the same time, which would amount to incentive compatibility.

The role of Application Service Providers (ASP) in the service provision industry is emergent. ASPs are third-party entities that manage and distribute software-based services and solutions to customers across a wide area network from a central data center. ASPs enable companies to realize significant cost savings, by out-tasking to ASPs complex enterprise applications. This out-tasking solution enables a company's technical staff to focus less on maintaining a complex enterprise software application and more on utilizing the application efficiently to improve its business processes and decision-making capabilities. By leveraging its data center and in-house support staff over multiple customers, an ASP provides companies with high quality cost-effective IT solutions. Thus, it is considered of particular importance to provide to this industry a particular mechanism of negotiating, supporting, policing/shaping, and charging SLSs for its customers.

In this paper, we develop and evaluate an approach for efficient SLS selection and implementation in a DiffServ domain within a MPLS network. A negotiation process between a user and a domain is introduced; thus the user can select from alternative options for resource allocation the one that better matches his needs. We assume that services are charged according to a usage-based scheme that provides the user with the right incentives. For the purposes of selection, we develop an appropriate utility model that expresses user preferences in a simple yet informative way. The communication of the user QoS preferences to the network as well as that of the results of the resource allocation process back to the user is done simply and effectively. The remainder of this paper is organized as follows. In section 2, we describe the overall architecture proposed for negotiation and enforcement of SLSs over a DiffServ domain in the context of the ASP service model. In section 3, we apply our method in a realistic ASP service provision paradigm, providing certain examples of services. In section 4, we describe the adopted charging scheme and justify our choice. In section 5, we present a distribution of the information that is essential for the rationality and the scalability of our approach. In section 6, we propose a model for user utility that expresses user preferences for SLS selection. In section 7, we describe in detail the communication among components of the architecture during the negotiation process. In section 8, we present some simulation experiments that indicate that using our negotiation process for SLS selection, the right incentives for the user, which are provided by the adopted charging scheme, are maintained. In section

9, we present specific choices for the implementation of our architecture in an experimental network and discuss certain performance issues. Finally, in section 10, we present some concluding remarks and possible extensions of our work.

2. THE ARCHITECTURE

We consider a multi-hop DiffServ network domain in which telecommunication services are provided in various QoS levels. This DiffServ domain is built on top of MPLS. Multiple Connectivity Providers share the control and ownership over the network resources. The overall responsibility about the end-to-end network service provision belongs to one or many Network Service Provider(s) (NSP(s)) who has contracts with certain Connectivity Provider(s). Also, various Application Service Providers (ASPs) register to NSPs and offer to their customers access to QoS-differentiated application services. Each customer subscribes himself to an ASP in order to have access to its application services through a service portal. Upon subscription, a customer (e.g. a company) specifies user profiles for the users of the service (e.g. the employees of the company). Each user profile contains information about user preferences regarding services, including those on application and network QoS. Also, the ASP associates a User Agent (UA) to each user. This agent negotiates the user's SLSs with the NSP on behalf of the user and according to his profile. The user profiles and the UAs are stored in a directory service in the data center of the ASP and they are downloaded to users' hosts upon logging into the services of the ASP. In this negotiation process, the representative of the NSP is a software entity to be referred to as Policy Server (PS). The PS is responsible for making the proper decisions for resource allocation that guarantee efficiency with respect to network utilization, while enforcing incentive compatibility by means of charging. The policy directions followed by the PS are stored in a well-known Policy Directory (PD), which is unique for the network domain (e.g. restrictions on bandwidth usage or a certain charge discount for specific users or applications). A software entity, referred to as the Information Directory (ID), holds information about the current network domain state (e.g. the available bandwidth or the packet loss probability per QoS class). This information is updated over time. This way the network provider is able to apply his policy rules for specific flows or users. For performance reasons, it is preferable to have multiple instances of the same PS as we explain in section 9.

The User Agent has to select the proper SLS x in order to maximize the *net benefit* of the user, i.e. to solve the maximization problem:

$$\max_x \{u(x) - c(x)\} \tag{1.1}$$

where *u*(*x*) is the utility function [9] of the user (expressing his preferences and his willingness to pay per SLS) and *c*(*x*) is a function that gives the expected charge for a service provision that is compliant to an SLS. If the maximum value is negative, then no SLS is selected and the user is served *best effort class*, which is offered for free. The PS can compute in advance the expected (or, in certain cases the actual) charge for a service provision. Thus, the User Agent negotiates with the nearest PS for the proper SLS, in order to solve the above optimization problem. The complexity of this maximization problem will be discussed in section 7. An important feature of our architecture is that it is necessary neither for the UA to know the location of the PS prior to the beginning of negotiation, nor for the PS to know the location of the UA. Indeed, we employ the RSVP RESV message to communicate the QoS requirements (in terms of the lowest acceptable QoS class) of the user's receiving application to the NSP, if the receiver of the traffic initiates QoS provision. Generally, we assume that a certain QoS class corresponds to a certain scheduling priority and/or a certain dropping probability, resulting to a minimum throughput and/or a maximum delay for a certain transmission rate. On the other hand, if the sender of the traffic has QoS requirements for the user's sending application, then these are communicated to the NSP, using the RSVP PATH message. It should be noted that in both of the above cases, RSVP PATH and RESV messages are exchanged between the source and the destination of the traffic, as in the standard RSVP protocol; however, we exploit only one such message in order to communicate the user's lowest acceptable QoS class to the NSP. Using RSVP, an application requests QoS in a clear and controllable way that is appropriate for the resource allocation process. In fact, we assume that the user only specifies the lowest acceptable QoS class in an abstract way e.g. adjusting a slide bar in the application. After SLS selection the PS associates the traffic flow of this SLS with a certain QoS class that is allocated specific MPLS resources, and finally notifies the sender of the service traffic, asking it to begin service provision according to the specified SLS.

Certain QoS classes are provisioned along each path (Label Switched Path (LSP)). A NSP has to ensure uniform properties for a certain QoS class over all the LSPs in which the QoS class is provided. That is, a QoS class provides the same QoS level over all LSPs. In order to achieve this, a NSP must implement certain mechanisms for balancing the traffic load among the QoS classes on the various LSPs. However, a NSP could allow different properties for the same QoS class over different LSPs and let the users (with their SLS selection) to achieve the load balancing. In this case, a QoS class in different paths has been allocated different resources in terms of capacity and buffer, and thus it provides a different QoS level. For simplicity, in the rest of the paper, we assume that the NSP employs appropriate mechanisms for balancing the traffic load among the QoS classes on the various LSPs, thus rendering the selection of path irrelevant to the net benefit of the user. Henceforth, the term QoS class essentially refers to a QoS level.

The operating point parameters[2] (i.e., *space* parameter s and *time* parameter t, see [11]) and other statistical characteristics are different per QoS level. We introduce here the notion of the *noncompliance risk*, which is defined as an a priori upper bound on the percentage of traffic that will not be treated in accordance to the SLS. The noncompliance risk specifies the upper tails of the distributions of the QoS parameters of a SLS; e.g., an upper bound on the end-to-end delay per packet, that can only be violated for a percentage of packets less than or equal to r. Alternatively, the non-compliance risk could be considered as the percentage of time in which the QoS parameters of a SLS are not met. The purpose of the noncompliance risk is twofold:

- Assure customers of a basic set of expectations that the provider promises to meet.
- Protect the provider by limiting liability, in cases of QoS violations.

Note that, although the noncompliance risk is a factor computable by the network, it is understandable by the users too. In order to give users the right incentives, different QoS levels are associated with different charges and possibly different noncompliance risks for the same traffic flow. The PS has to find the eligible QoS levels for the traffic of each new SLS and to get the corresponding charges by the Connectivity Provider(s). The UA has to make the final selection of the SLS that maximizes the user's net benefit.

3. THE SERVICE PROVISION SCENARIO

In this section, we describe a realistic service provision paradigm within our architecture. For simplicity, we assume that in the network considered one NSP has the responsibility for the provision of network services. Also, we assume that in this network, one ASP offers application services such as video-on-demand, ftp, broadcast TV, or video-conference. In such a case, the ASP and NSP have to coordinate to deliver the services, which have to be provided consistently by both the network and the ASP's

[2] The space parameter s expresses the statistical multiplexing capability of a link. If the capacity of the link is very large in relation with the peak rate of the various flows, then s=0. However, s increases if the peak rate of the various flows is large relatively to the capacity of the link. The time parameter t expresses the time prior for overload to occur. The parameters (s, t) together represent a network operating point, which depends on factors such as bandwidth, buffer sizes, traffic mix, and can be estimated from traffic measurements.

data center, together with their support, shaping/policing and charging. This is possible via eXtensible Markup Language (XML), which provides a flexible way of communication, through a Document Type Definition (DTD) common for the ASP and NSP.. In general, the service scenario is as follows (see Figure 1):

1. The ASP publishes the services offered using XML. That is, a service portal receives notification of the new service offerings from the ASP via XML. Then, the service portal adds them to the retail portfolio.
2. The administrator of customer-company (or a certain user who is actually the customer) accesses the portal and subscribes to an application, and subsequently the ASP and the NSP are notified, via XML, about the new customer.
3. Then, according to the ASP model of service provision, the ASP offers the customer the necessary software in order for the user(s) of the customer to instantiate the subscribed services. That is, the ASP offers the application components that are necessary for the user(s) in order i) to request from the NSP the necessary network resources according to their QoS requirements, and ii) to be delivered the service. This software, depending on the specific service provided, may include various voice and video encoders and decoders, and/or featured software for sending and receiving voice, video and data (server and client functionality).
4. A user (which is associated with a subscribed customer) logs into an ASP service having certain QoS requirements.
5. The PS (being the representative of the NSP) negotiates the selection of SLS with the user's UA.
6. The PS provides the necessary network resources to satisfy the QoS requirements of the selected SLS by the user.
7. Then, the PS reserves the required service content(s) at the ASP's data center for the user.
8. The user is provided the service with the requested QoS.

Figure 1. The service provision scenario.

The communication of the QoS requirements and the SLS selection process depicted in steps 4 and 5 of Figure 1, are different in the cases that the service is provided to a single or multiple users in a *service session*. For convenience, we consider first the case of service provision to a single user, and subsequently the case of service provision to multiple (i.e. two or more) users.

3.1 The Case of a Single User

We consider the case where the service is provided to a single user in a service session. Examples of such services are video-on-demand (with pre-stored or live content) and ftp (for downloading data). We assume for simplicity that the user requires QoS in his receiving traffic; then, the communication of the QoS requirements and the SLS selection process are as follows (see Figure 2):

1. The user specifies his lowest acceptable QoS class through his service interface in order to instantiate a service session, and initiates a connection with the ASP's data center.
2. The application component of the user sends along the route to the ASP's data center a RSVP RESV message (step 2b in Figure 2) with the required QoS class for the receiving flow of the service session.
3. The RSVP RESV message is intercepted by the ingress router of the network (through an appropriate protocol, such as the Common Open Policy Service (COPS), the Simple Network Management Protocol (SNMP) or the Telnet Client (TelnetCLI)) and sent to the Policy Server (PS).
4. The negotiation process that we describe in section 7 takes place between the PS and the UA of the user, in order the SLS that maximizes the net benefit of the user to be selected.
5. The parameters of each RSVP RESV message are modified according to the results of the negotiation process, and then the message travels along the route to the ASP's data center. The modified RSVP RESV message contains information for the categorization of the traffic of the receiving flow of the user to a QoS class.
6. When the RSVP RESV message reaches the ASP's data center, a new service session is instantiated for the user with the required QoS.

Figure 2. The communication of QoS requirements and the SLS selection steps in the case of service provision to a single user in service session.

The RSVP PATH messages (2a) that are shown in Figure 2 and sent between the user premises and the ASP's data center are only used for the operation of the RSVP protocol. If, however, the user required QoS in his sending traffic (as opposed to the receiving traffic, which is the case depicted in Figure 2), then RSVP PATH messages would be used for the communication of the QoS requirements to the PS. In such a case, the roles of RSVP PATH and RESV messages would be interchanged.

3.2 The Case of Multiple Users

In case that the service is provided to two or more users concurrently, two sub-cases may arise: 1) the QoS provided to each user is independent from the QoS provided to the other users, and 2) the QoS provided to a user is inter-dependent with the QoS provided to the other users. We discuss these sub-cases separately below.

3.2.1 Independent QoS for Different Users

In this sub-case, the QoS negotiation and provision can be considered separately for each user. Thus, the communication of the QoS requirements and the SLS selection process for each user are accomplished in the same way as in the case of the service provision to a single user in a service session. An example of such a service is broadcast television, where multiple users may receive the same content, each with different QoS.

3.2.2 Inter-dependent QoS for Different Users

In this sub-case, the QoS negotiation and provision cannot be considered separately for each user. Instead, QoS has to be considered uniformly, i.e. a single SLS has to be selected for all the users of the service for all sending and receiving flows. For example, for two users talking over a telephony application, it is not meaningful for them to be served at different QoS classes, because the QoS class of the traffic when being sent would not coincide with that of the same traffic when being received. For convenience, we consider in the discussion below a specific example of such a service, namely, the multi-party video-conference. Also, for simplicity, we assume that the communication is done through the ASP's data center, that the service involves only one flow of information per direction between the ASP's data center and each user, and that all users are associated to the same customer. We consider a "join period" during which each participating user communicates his QoS requirements (which are the same both for his sending and receiving flows), sending a RSVP RESV message along the route to the ASP's data center. The lowest acceptable QoS class for the service provision is the maximum required QoS class of all users. Also, the maximum acceptable noncompliance risk for the service provision is the minimum of the maximum acceptable noncompliance risks specified by the users. After the end of the join period, the SLS negotiation process takes place between the PS and a Customer Agent (CA) that represents the customer in this process. The CA selects the SLS that maximizes the net benefit of the customer, i.e.

$$\max_x \left\{ \sum_i \left[u_i(x) - c_i(x) \right] \right\} \tag{3.1}$$

where x is a SLS, $u_i(x)$ is the utility and $c_i(x)$ is the charge of a participating user i. The CA communicates with the UA of the participating user i in order to find $u_i(x)$ and other information such as the maximum acceptable noncompliance risk by the user i. After SLS selection, the PS sets the rules to the ingress points of the network for the traffic categorization of the service according to the selected SLS. Another alternative would be for each user to perform negotiation separately, and then for the CA to select the highest of the QoS classes individually selected as the uniform QoS class for all users.

4. THE CHARGING SCHEME

The Policy Server (PS) computes the charge for each offered SLS on the basis of effective bandwidth, using the formula below (see [12]):

$$\begin{aligned} &c(x) = \bar{a}_i(x;m)p_iT \\ &\bar{a}_i(x;m) = \frac{1}{s_it_i}\log\left[1+\frac{m}{H(t_i)}(e^{s_it_iH(t_i)}-1)\right] \\ &H(t_i) = \min\{h, \rho+\beta/t_i\} \end{aligned} \tag{4.1}$$

x in the above formula is the SLS in terms of leaky bucket parameters $[h, (\rho, \beta)]$ and a QoS class i, $H(t)$ is the *effective peak rate*, which is sufficient for the calculation of $\bar{a}_i(x;m)$, $m \in [0, \rho]$ is the mean rate of the traffic to be induced by the source, and T_i is the duration of the flow for that QoS class. For simplicity, the parameter T_i will be denoted as T in the remainder of the paper. The parameters m and T are either known to the application server in advance (e.g., for a video movie), or can be estimated by the User Agent on the basis of past information. p_i is price per unit of effective bandwidth for $\bar{a}_i(x;m)$, which in theory should coincide with the *shadow price* of the constraint that limits the resources for a QoS class i. s_i, t_i parameters are the operating point of a QoS class, i.e. the operating point of the most congested link[3] of the path(s) over which QoS class is implemented. In fact, $\bar{a}_i(x;m)$ is an upper bound on the actual effective bandwidth and corresponds to that of an on/off source. The actual effective bandwidth of a source is computed according to the so-called inf-sup formula, derived by means of Large Deviations techniques [11].

[3] The charging scheme of equation (4.1) is based on the assumption that all flows traverse a certain link that it is the most congested one (bottleneck link), and they are charged according to resource usage on this link. If this assumption is not applicable, then the charge can be computed by adding the outcome of (4.1) over all links of the path traversed by each flow. Our negotiation approach can be modified to cover this case as well, which however is more complicated.

This charging scheme is applicable to the calculation both of the expected charge of a new flow, and of the actual charge to be paid by the user after service provision. These two values are not expected to vary too much. This difference is due to the discrepancy between the a priori mean rate calculated by an application server for a piece of content in a QoS class and the actual mean rate with which the service is provided. This is not important, because the guarantees provided by a DiffServ network, as currently defined by the IETF, can be loose and users are actually charged according to the resources they consume, i.e. their actual mean rate measured by the network during service provision. The scheme provides users with the incentive to choose the tightest possible traffic profile (h, ρ, β), for a resource usage minimizing the actual effective bandwidth and thus the actual charge to be paid by users. Also, through negotiation, users select the QoS class that maximizes their net benefit and thus it is closer to their actual needs. As explained in [12], this charging scheme is fair (i.e. reflects actual usage), if the variance of the ratio of $\bar{a}(x;m)$ to the actual effective bandwidth $\alpha(x;m)$ is small, for the set of those x and m that characterize the services of interest. Also, it gives the incentives to users to select SLSs that better match their actual needs.

Consider that we have a set I of QoS classes in the DiffServ network, where the variable $i \in I$ has the value 0 for the lowest and the value 1 for the highest QoS class. Thus, the value i of an intermediate QoS class can be viewed as the QoS it provides relatively to the highest QoS class. In this case the price p_i per unit of effective usage for each QoS class i can be given by an explicit function $P(i)$ of i. In general, $c(x)$ has to be increasing when QoS improves, in order for users to have the right incentives to select the QoS class that optimizes their net benefit (i.e. attain the best trade-off between QoS and charge).

5. DISTRIBUTION OF INFORMATION

In order for the described architecture to be feasible and scaleable, and for the traffic of a new flow to be assigned a SLS using our negotiation process, information has to be available and exchanged among the components in a proper way. In this section we describe and justify a certain distribution of information that serves these goals.

A User Agent (UA), being a representative of the user, stores the utility model and has access to the corresponding user profile. Furthermore, the user's application is able to communicate to the corresponding UA any necessary information, such as the upper bound on the noncompliance risk acceptable by the user. The Policy Server (PS) has full control of the network resources, by being aware of the available bandwidth, the operating point (s, t), the price per unit of bandwidth and the noncompliance risk of each QoS class. The PS offers the users some information on the service classes provided, by posting the values of parameters t for each QoS class; the reason why this is useful will be explained below. In order to proceed with our negotiation approach, we have to specify how the PS finds the traffic parameters for the SLSs in each QoS class (which are subsequently conveyed to the resource reservation request). This issue is discussed below.

The ASP's data center posts information regarding the performance of the traffic transmitted either by itself or by an application server component downloaded by the ASP to the user premises. For all the services it provides, the ASP computes this information and stores it to its data center. Furthermore, each piece of content is transmitted under different encoding in each service class (i.e. QoS class). For example, a particular video may be MPEG-encoded, with various quality factors. Thus, the ASP's data center posts the following information: the traffic parameters in terms of token bucket parameters (ρ, β), peak rate h, and the mean rate m with which the traffic of each content is sent over each class as well as the duration T of the service if both are known, which is the case for pre-stored content. If the content is live as for example in the case of the broadcast television service, then the duration T of the service as well as the mean rate m have to be estimated by the UA of the user (based on past information) and to be passed to the PS during the negotiation process that we describe in section 9. Furthermore, the ASP has the incentive, due to competition, to compute and employ the traffic parameters that *minimize the charge* for each piece of content in each class. The ASP should pass this information to the relevant server components or to store it to the ASP's data center. An ASP is able to accomplish this if it is aware of the way the NSP charges. As already explained in section 3, we assume that the NSP charges proportionally to the effective bandwidth of the user's SLS. The ASP has to derive the traffic parameters $\{h, (\rho, \beta)\}$ that minimize this charge. (Note that in equation (4.1), we used the index of the QoS class i for certain quantities. For simplicity, we drop this index in the discussion to follow, since the procedure described below should be applied for each QoS class separately.) The peak rate h for each piece of content in each QoS class is determined by the amount of shaping performed by the server component (or by the ASP's data center) for this specific content: A smaller peak rate results from a larger amount of shaping, which however influences the playout quality due to the shaping delay introduced. As explained in [12], given the value of the peak rate h, there are various pairs of the token bucket parameters (ρ, β) for which all the traffic sent would be conformant. These pairs belong to an indifference curve such as the one shown in

Figure 3. The effective bandwidth corresponding to a SLS is increasing in the effective peak $H(t)=min\{h, \rho + \beta/t\}$. An ASP has simply to choose the pair (ρ, β) that minimizes H. If the minimizer of H is h, then the selection of (ρ, β) does not affect the charge. Otherwise, if the minimizer of H is $\rho+\beta/t$, then the pair (ρ^*, β^*) that minimizes H is that point at which the tangent to the

indifference curve has slope $-t$; see Figure 3. We assume that this point is always chosen, even if it turns out that $h<\rho^*+\beta^*/t$. Thus, an ASP derives the parameters $\{h, (\rho^*, \beta^*)\}$ that minimize the estimate of the charge per time unit for a certain content in a QoS class that is characterized by t and posts this information to its data center. It is worth noting that it is not necessary for the PS to post the values of the space parameter s. Moreover, the above procedure for selecting (ρ, β) is valid regardless of whether the duration T of the flow is known or has to be estimated.

Figure 3. The indifference curve of parameters (ρ, β) for a given peak rate (shaping delay) and for a specific percentage of traffic to be conforming.

A brief description of the distribution of information is as follows: The user specifies the lowest acceptable QoS class for the traffic to be received and his maximum acceptable noncompliance risk, through his interface downloaded by the ASP to the user premises. The ASP calculates and posts (in the ASP's data center) the traffic parameters in each QoS class that minimize the charge for the user. Thus, the receiver component has only to transmit inside the RSVP RESV message the QoS class that is required by the user. Using this information, the PS is able to negotiate with the UA of the user the selection of SLS, and then to fill the RSVP RESV messages with the corresponding parameters to the selected QoS class.

So far, we have presented a distribution of information, where each piece of information is stored by a component that has the incentive to do so. In the next sections, we explain how the components act and interact in order for the UA to make an efficient selection of the SLS.

6. MODELLING USER UTILITY

In this section we discuss how the UA selects the best SLS on behalf of the user. First, we have to model user preferences in a *user utility* depending on the various parameters involved in the SLS. The traffic information in this SLS includes some of the RSVP parameters, namely the peak rate h and the token bucket parameters (ρ, β). Note that a flow with the same traffic parameters $\{h, (\rho, \beta)\}$ is differently served by different QoS classes, in terms of throughput, delay, jitter and packet loss. The user requires the lowest acceptable QoS class by adjusting a slide bar determining the requested QoS in his user interface. Also, he specifies an upper bound for the acceptable noncompliance risk r by the network. In particular, the SLS should comprise the tuple of parameters (h, ρ, β, QoSclass, r) that maximizes the net benefit of the user. Trying to compute five parameters simultaneously is a very complicated task. However, this can be considerably simplified by assuming that only the charge-minimizing triplet {*h,* (*ρ, β*)} of each QoS class is an eligible choice. Thus, it only remains to select the QoS class and the noncompliance risk of a SLS.

The ASP's data center posts the traffic parameters per service content in each QoS class that minimizes the expected charge. In order for a user to be served by these optimized parameters per QoS class, he has to pay an amount of money per QoS class. It is assumed that the maximum amount the user is willing to pay (denoted as W_{max}) is also contained in the user profile. This amount corresponds to the best possible contract for the service and for a specific content. Under the above assumptions, the UA has to select only the QoS class and the noncompliance risk values so as to maximize the user's net benefit. In order to express the corresponding user-preferences we have developed a simple utility model that serves our purposes. In particular, the utility function for an SLS x is given by the formula below:

$$\begin{aligned} & u(x) = W(x) f(r_{user}, r_{netw}), \quad r_{netw} \le r_{user} \\ & \text{where} \quad W(x) = U(QoS_{DF}) W_{max} \end{aligned} \tag{6.1}$$

In the above formula, x is the SLS in terms of peak rate h and token bucket parameters (ρ, β) *and* QoS class. (However, recall that given the QoS class, all h, ρ and β values can be taken as given.) r_{user} is the maximum percentage of noncompliance with the SLS accepted by the user and r_{netw} is the noncompliance risk expected to be offered by the network for an SLS. $f(r_{user}, r_{netw})$ is a normalized function that expresses the user's increased satisfaction for low values of $r_{netw.}$ $W(x)$ is a monetary expression of the user's utility for this SLS when there is no risk of QoS violation (i.e. $r_{netw} = 0$); that is, $W(x)$ is the user's willingness to pay for this traffic contract in the certain QoS class. Note that the user satisfaction should be maximized (with $f = 1$) when $r_{netw} = 0$, and decrease as r_{netw} approaches r_{user}. This is due to the assumption that a user is satisfied with any r_{netw} that is lower than or equal to r_{user}. Therefore, when r_{netw} increases and approaches r_{user}, user satisfaction should decrease. In fact, we expect f to have the shape depicted in Figure 4. For each particular user the "decreasing" segment of the curve, the coordinates of its saddle point, and its minimum value will depend on the sensitivity of the user to the noncompliance risk offered by the network. Finally, $U(QoS_{DF})$ is the normalized utility factor that corresponds to this QoS class of DiffServ, and expresses the percentage of W_{max} that the user is willing to pay for this QoS class; this factor equals 1 for the highest such class.

Figure 4. A possible $f(r_{user}, r_{netw})$ function.

The network services that the user asks for are either elastic or guaranteed. Note that we assume that in general a higher QoS class results in a higher scheduling priority and/or a lower dropping probability: both result in higher bandwidth consumption by the higher QoS class under the requirement that the same traffic is transported. Thus, the shape of the function $U(x)$ (with respect to the DiffServ QoS classes) can be reasonably taken to be the same as that of the user utility expressed as a function of bandwidth, which is the case thoroughly studied in [10]. Therefore, for elastic services the utility function is concave, as shown in Figure 5a, due to the diminishing marginal utility induced by an additional unit of bandwidth when the amount of bandwidth already allocated increases. For guaranteed services, the utility function is initially convex and then concave with respect to bandwidth; see Figure 5b. However, for guaranteed services it does not make much sense to consider the convex segment of the utility curve as acceptable, because the marginal increase in utility due to a small increase in the bandwidth to be allocated is increasing; this implies that the user has the tendency to require more and more bandwidth thus "abandoning" this segment. Therefore, the guaranteed utility function, as an approximation, can be taken as identical to elastic except for the fact that it starts at an index of QoS class larger than 0. Recall that the UA of a guaranteed user knows this starting point, which is specified by the user through a sliding bar!

Figure 5. A normalized utility function U with respect to QoS class for elastic (a) and guaranteed (b) services.

Finally, note that for both functions $U(QoS_{DF})$ and $f(r_{user}, r_{netw})$ included in the expression of $u(x)$, we have only discussed on their shape. For each particular user, the exact such functions depend on the details of his preferences. A practical approach to cope with this is for the UA to employ a typical function $U(QoS_{DF})$ and a typical function $f(r_{user}, r_{netw})$, and then customize these by means of a supporting *learning* procedure. This matter is left for future research.

7. THE SLS SELECTION PROCESS

In this section, we describe the SLS selection process that takes place for the categorization of the traffic of a new flow to a service class. Depending on whether the mean rate m and the duration T of the new flow are known a priori by the ASP or they are estimated by the User Agent of the user, the SLS selection process is different. Let us first consider the case that the mean rate m and the duration T of a new flow are known a priori to the ASP, i.e. the service content is stored to the ASP's data center. In this case, we have the following steps in the SLS selection process:

1. The SLS request of a user application reaches the closest Policy Server (PS). The RSVP RESV message contains the lowest QoS class that will be acceptable by the user.
2. The PS identifies the UA that is associated with the user of the application through the location information enclosed in the RSVP RESV message and starts negotiating on the SLS that will be finally offered.
3. In the beginning of the negotiation process, the UA informs the PS of the parameter r_{user} that expresses the maximum acceptable value of noncompliance of the requested SLS with the SLS offered. Using this parameter, the PS is able to communicate with the Information Directory (ID) and to find the eligible offers that satisfy the user requirements. Each offer is composed of QoS class, noncompliance risk r_{netw}, the parameters (s, t) of the QoS class and an expected charge $c(x)$.
4. In order to compute $c(x)$, the PS employs the available information about the traffic parameters for each offer in terms of peak rate h, token bucket pair (ρ, β), the mean rate m and the duration T of the service flow per QoS class for the specific piece of content that is ordered from the ASP's data center, and the information contained in the ID. The ID contains network state information, as the operating point (s, t), the available bandwidth and the noncompliance risk of a certain QoS.
5. The PS communicates the offers [$c(x)$, r_{netw}, QoS class] to the UA, which solving the net benefit maximization problem selects the proper SLS for the user. For simplicity, we assume that this optimization is carried out exhaustively for the discrete set of QoS classes offered. (It is plausible that the complexity of this process can be improved if the concavity and monotonicity properties of the functions involved are exploited.) If the maximum net benefit in the SLS selection process for a new flow is negative, i.e. the negotiation process does not find a SLS that serves the QoS requirements of a user for the flow, due to a relatively low willingness to pay or a highly loaded network domain for the required QoS classes, the traffic of the new flow is served best effort. The sequence of messages exchanged between the components during the SLS selection process is depicted in Figure 6.

On the other hand, if the mean rate m and the duration T of the new service flow are not known a priori to the ASP's data center (for example in the case of the broadcast television service), they have to be estimated by the User Agent based on previous executions of the service with similar content. In this case, steps 3 and 4 differ from those depicted in Figure 6 in the following:

- In step 3, the UA informs PS for the maximum acceptable value of r_{user}, for the requested content, for the estimated mean rate m of the new service flow, and for its estimated duration T.
- In step 4, the PS calculates the expected charge for each offered SLS using the parameters (h, ρ, β) that are announced by the ASP's data center for each service content per QoS class, the mean rate m and the duration T that are estimated by the UA, and the information received from the ID.

Figure 6. The SLS selection process, when the content of the service flow is stored to the ASP's data center.

It should be noted that Figure 6 applies directly to the case of a single-user application (see subsection 3.1), or a multi-user application with independent QoS for the various users (see subsection 3.2.1). In the case of multiple users with inter-dependent QoS (see subsection 3.2.2), the following modifications apply:

- If customer net benefit is maximized, according to formula 3.1, then all steps of Figure 6 are the same except for the fact that Customer Agent replaces the User Agent.
- If each user performs net benefit maximization individually (as discussed in the end of subsection 3.2.2), then there should be added to Figure 6 a penultimate step, in which the Customer Agent combines all outcomes taken from the UAs of the users, in order to decide on the uniform QoS class. In fact, this uniform QoS class will be set (by the Customer Agent) tto the maximum of the individually selected QoS classes.

In the single-usercase, the SLS selection process involves the exchange of a few messages only. In fact, this number of messages equals the number of SLSs (i.e., QoS classes) eligible for this user, which in general is small., For this reason, the number of iterations needed for solving the net benefit maximization problem in (1.1) is small. Additionally, each iteration only requires the computation of $u(x)$ and $c(x)$, which is straightforward. Therefore, the overhead for solving the net benefit maximization problem is small. The above also apply for the communication and computation overhead per user in the case of multiple users, with the addition of one more message sent by each User Agent to the Customer Agent. Therefore, the communication and computation overhead per user of the SLS selection process is minimal in both the single- and the multi-user cases.

8. SIMULATION EXPERIMENTS AND RESULTS ON ECONOMIC EFFICIENCY

Recall that as described in section 4, the charging scheme employed provides the right incentives to users for SLS selection that reflects their actual requirements. However, is incentive compatibility maintained for certain users that apply this negotiation process to select their SLSs? That is, is social welfare (i.e. the sum of user utilities) also promoted when each user performs individual optimization?

To investigate this we performed repeatedly the following simulation experiment: We compute the social welfare resulting by employing the negotiation process for sharing an amount of resources K to a number of users N (whose QoS is independent of each other's), and compare it with the social welfare resulting when sharing the same amount of resources equally to the N users. In this experiment, all users request the same service and content (e.g. the same video). Half of the users are taken as elastic and the rest of them as guaranteed. Furthermore, the utility functions are parameterized and users have different willingness to pay. All parameters and willingness to pay follow uniform distributions of various means and variances (the exact parameterization and the distributions of the various parameters as well as of the willingness to pay are discussed below). The net benefit maximization process selects a certain QoS class (and, thus, resource consumption) for each user to be served. These selections also determine the total amount of resources K that should be allocated and the total charge C_{tot} for the users. The charging function $c(k)$ of the network for a QoS class k, using the charging scheme we mentioned, can have any increasing shape; i.e. it can be either linear (e.g. if the parameter of QoS class differentiation is bandwidth), or convex (e.g. if the parameter of QoS class differentiation is cell loss probability), or concave (according to a specific policy of the network provider) or a mix of them. We compute the social welfare U_{nb} emerging under the negotiation process.

Then, we share the resources K equally to each user, by offering to all of them the QoS class charged for C_{tot}/N. This corresponds to resource usage K/N for each user. Adding the new utility values for the new QoS class of the same users (i.e. maintaining the same utility functions as previously), we obtain the new total utility U_{fair}. We noticed that $U_{nb} > U_{fair}$ for all the shapes of the charging function and for various uniform distributions of willingness to pay. This advocates that incentive compatibility is maintained with the proposed negotiation process.

Figure 7. The range of the utility functions of guaranteed and elastic users (a), and (b) the various shapes of the charging function (concave, linear, convex).

In particular, regarding user utilities and charging for different QoS classes, we used the following formulae in our simulation experiments (see also Figure 7):

$$\begin{aligned}
&U_{elastic}(x) = \left(1 - e^{-\lambda x}\right)W_{max} \\
&U_{guaranteed}(x) = \left\{\left[\tan^{-1}(30x - \lambda) + 1.3\right]/3\right\}W_{max} \\
&C_{linear}(x) = x\,C_{max} \\
&C_{concave}(x) = \left[\log(1 + 35x)/3.6\right]C_{max} \\
&C_{convex}(x) = x^2 C_{max} \\
&W_{max} = W_{var} + W_c
\end{aligned} \tag{8.1}$$

where x is the QoS class of an SLS, and λ is a random variable that is uniformly distributed in the interval [2.25, 10]. W_{max} is the willingness to pay expressed as the sum of a fixed term $W_c = 2000$ and a term W_{var}, which is a random variable that is uniformly distributed in the interval [0, 2E[W_{var}]]. E[W_{var}] is fixed (for all simulated users) in each experiment, and in successive experiments takes the values 30000, 35000, …, 100000. Also, $C_{max} = 60000$ is the charge for the service in the highest QoS class. Figure 8 depicts the percentage of difference in the social welfare attained using the negotiation process for net benefit maximization (U_{nb}) and that using the equal split (fair process) of network resources (U_{fair}), for the three shapes of charging function and for E[W_{var}] ranging between 30000 and 100000. (A continuous set of QoS classes was assumed.)

Figure 8. The percentage of difference between U_{nb} and U_{fair}, for the various shapes of the charging function, when E[W_{var}] ranges from 30000 to 100000, and W_c= 20000.

In Figure 8, we observe that the percentage differences between U_{nb} and U_{fair} are always positive, and that their shapes are convex in E[W_{var}]. Therefore, our approach is always more efficient than the fair process. The improvement achieved using our SLS selection process in social welfare is substantial in the cases of concave and linear charge. On the other hand, under convex charging, the mean QoS class costs less than the mean charge, implying that the fair process results in a better QoS class than the mean. Therefore, the loss in social welfare due to the fair process is limited. It should also be noted that when E[W_{var}] is large, then almost all of the users select a high QoS class. Thus, the fair process does not alter significantly the SLS of most of the users, and the associated loss in social welfare is very small. We have performed the above simulation experiments for various values of the parameters and all results are in favor of incentive compatibility when using the net benefit maximization process to select the optimal SLSs for the users.

9. IMPLEMENTATION ISSUES

So far, we have described the architecture, and the SLS selection approach. In this section, we present specific choices adopted in the implementation of our approach in a real network domain providing QoS-differentiated video-on-demand services to a single user in a service session. The network domain consisted of three routers and three end-hosts. The routers in the network ran Cisco MPLS implementation, while the end-hosts ran Windows 2000. RSVP was employed for communicating the resource allocation requests, because it provides a clear and controllable way for applications to express their QoS requirements. Windows 2000 platform was used for convenience as it supports RSVP and performs the shaping, policing and marking of the flows according to RSVP signaling. Windows 2000 platform is not mandatory for the implementation of our architecture, as any RSVP implementation downloaded to user premises as a plug-in would satisfy our purposes. In that case, the ingress routers of our testbed would perform the tasks of shaping, policing and marking. Furthermore, according to the process described in section 5, the computation of the RSVP traffic parameters for the resource reservation request can be transparent for the end-user, who may ask for a required QoS class in an abstract way (i.e. adjusting a slide bar, thus asking for a more expensive and better service or for a less expensive and inferior one). In the ingress routers, we used the protocol Common Open Policy Service (COPS) {[13], [14]} in order to intercept RESV messages and forward them to the Policy Server, which is partly a COPS server (see Figure 9). Recall that the PS location is not known to the UA. The choice of COPS was preferred from other eligible protocols (SNMP, TelnetCLI), because it facilitates modifications of reservation requests according to the results of the negotiation process as well as discovery of PS location. The Policy Directory (PD) and the Information Database (ID) are implemented using Microsoft Active Directory. The PS uses LDAP for the storage and retrieval of information to PD and ID.

A user's receiving application component sends a RESV message to the ASP's data center with the QoS request for a new flow reception. Upon reception of the RESV message from an ingress router of our network, the following steps take place:

1. The RESV message is redirected to the PS.
2. The PS using the destination address contained in the SESSION object of the RESV message identifies the user's UA and negotiates with it.
3. A specific SLS is selected by the negotiation and the PS uses COPS in order to change the RESV message properly. The new RESV message contains the traffic parameters with which the flow is served in the selected QoS class and a DCLASS object [15] with the DSCP of the QoS class.

4. The RESV message passes transparently through the other routers of the network, i.e. the RSVP protocol is used only for signaling purposes and not for resource allocation.

The ASP's data center, receiving the RESV message, sets the shaping, policing and marking rules for the new flow transmission. The specific DSCP contained in the DCLASS object of the received RESV message is used by the server component for marking the packets of the new flow.

Figure 9. The use of COPS protocol to intercept QoS request and employ QoS provision according to the SLS selection process.

Note that the use of RSVP along with the COPS protocol provides a means of controlling resource usage according to the results of the negotiation process. All resource requests are forwarded to PS, which modifying them according to the results of the negotiation, controls the resource allocation procedure in a network domain. The PHBs are applied to the traffic entering the MPLS network according to the DS values of the packets using the E-LSP method described in[16]. According to the E-LSP method, the information about the PHB to be applied by the network is conveyed to the EXP field of the Shim Header of the MPLS packets, for up to eight traffic aggregates along a LSP. Finally, the state information for the SLSs and the traffic aggregates in the various PHBs are stored in a Microsoft Active Directory using LDAP.

The proposed architecture can be used as a framework through which a network provider can enforce the results of a SLS selection process, according to certain policies for resource allocation. Indeed, the implementation demonstrated *visually* the fact that, for the same video-movie, users willing to pay more are served by a higher QoS class and receive better quality. This is accomplished with minimal processing overhead and modification requirements for the network core, since traffic classification, policing, shaping and marking operations are done by the edge hosts, or by the ingress routers in case that Windows 2000 platform is not used. Furthermore, the signaling overhead is also minimal as every reservation request is processed only once and its state is stored in the Policy Server only. Thus, in terms of scalability, our architecture shares the same properties with the DiffServ architecture, which scales well for large network domains. It is preferable in terms of performance to have multiple instances of PS spread inside the network domain. This way, the processing requirements for a particular PS and the communication overhead of the SLS selection are reduced. Indeed, in this case, the requests are shared among the PSs and the links traversed by the messages exchanged during the negotiation are less. Using our approach for SLS selection, the network can offer the right incentives to the individual users for resource consumption.

It should be noted that there are also certain other issues that are important for the implementation of our approach in a commercial network domain, such as security (e.g., in the transport and execution of mobile agents, in the transport of user traffic etc.), accounting and monitoring. We have refrained from dealing with such issues, since we believe that they can be taken care of by complementing our architecture with existing relevant prototypes and/or commercial products, which should be integrated with the implementation described in this section.

10. CONCLUSION AND FUTURE WORK

In this paper we developed and evaluated an efficient SLA selection mechanism for a DiffServ-over-MPLS network domain within the Application Service Provide (ASP) service model. We also presented a framework for SLA support, negotiation, policing/shaping and charging. Our approach was implemented in a real network domain providing a video-on-demand service, demonstrating its features.

An important feature of our approach is the simplicity of user's procedure for selecting optimal SLS parameters, which has been presented in full detail. This procedure involves a model of user preferences, expressed by means of a simple yet informative utility function. This function can be refined and customized by means of learning. This is an interesting direction for future research, together with the application of our negotiation process for proper SLA selection between network domains. We assessed our approach by means of simulation experiments. The results indicate that it is incentive compatible, in the sense that individual optimization by each user (in SLA selection) also leads to improved social welfare. Another important feature of our approach is the distribution of information enabling the SLA selection; under this distribution, each of the components involved only possesses those pieces of information for which it has an incentive to store. Finally, although not presented in the paper, our architecture can be extended so as to cover end-to-end dynamic SLA provision across DiffServ network domains.

Our work can also be employed as a complement to the traffic engineering procedures, by modifying the price per unit of effective usage for each QoS class, either dynamically (thus balancing the load among the QoS classes), or statically (thus offering users the incentives to select certain paths instead others). Therefore, our architecture offers to the network provider the flexibility to apply pricing and/or resource allocation policies for improving efficiency in network operation.

ACKNOWLEDGEMENTS

The authors are grateful to Magda Chatzaki, Costas Courcoubetis, and Vassilios Siris for useful discussions on the subject of the paper.

REFERENCES

[1] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang and W. Weiss, "An Architecture for Differentiated Services", IETF RFC: 2475, December 1998.

[2] J. Wroclawski, "The Use of RSVP with IETF Integrated Services", IETF RFC: 2210, September 1997.

[3] Y. Bernet, R. Yavatkar, P. Ford, F. Baker, L. Zhang, M. Speer, R. Braden, B. Davie, J. Wroclawski and E. Felstaine, "A Framework For Integrated Services Operation Over DiffServ Networks", IETF Internet Draft: <draft-ietf-issll-diffserv-rsvp-05.txt>, May 2000.

[4] R. Braden, L. Zhang, S. Berson, S. Herzog and S.Jamin, "RSVP Functional Specification", IETF RFC: 2205, September 1997.

[5] R.Guerin, S. Blake and S. Herzog, "Aggregating RSVP-based QoS Requests", IETF Internet Draft: <draft-guerin-aggreg-RSVP-00.txt>, November 1997.

[6] K. Nichols, V. Jacobson, L. Zhang, "A Two-bit Differentiated Services Architecture for the Internet", IETF RFC: 2638, July 1999.

[7] E. C. Rosen, A. Viswanathan, R. Callon, "Multiprotocol Label Switching Architecture", IETF Internet Draft: <draft-ietf-mpls-arch-07.txt>, July 2000.

[8] D. Goderis, Yves T'joens, Christian Jacquenet, George Memenios, George Pavlou, Richard Egan, D. Griffin, P. Georgatsos, L. Georgiadis and P. Van Heuven. Service Level Specification Semantics, Parameters and negotiation requirements. IETF Internet Draft: <draft-tequila-sls-01.txt>, June 2001.

[9] H. R. Varian, *Microeconomic Analysis - Third Edition*, W. W. Norton & Company, New York, 1978 (reprinted 1984, 1992).

[10] S. Shenker. Fundamental Design Issues for the Future Internet. *IEEE Journal on Selected Areas in Communications,* vol. 13, no. 7, September 1995.

[11] C. Courcoubetis, F. Kelly and R. Weber, "Measurement-based usage charges in communication networks", Statistical Laboratory Research Report 1997-19, University of Cambridge, 1999.

[12] C. Courcoubetis and V. Siris, "Managing and Pricing Service Level Agreements for Differentiated Services", In *Proceedings of IEEE/IFIP IWQoS'99,* London, May 31 – June 4, 1999.

[13] D. Durham, J. Boyle, R. Cohen, S. Herzog, R. Rajan and A. Sastry, "The COPS Protocol", IETF RFC: 2748, January 2000.

[14] S. Herzog, J. Boyle, R. Cohen, D. Durham, R. Rajan and A. Sastry, "COPS Usage for RSVP", IETF RFC: 2749, January 2000.

[15] Y. Bernet, "Format of the RSVP DCLASS Object", IETF Internet Draft: <draft-ietf-issll-dclass-01.txt>, October 1999.

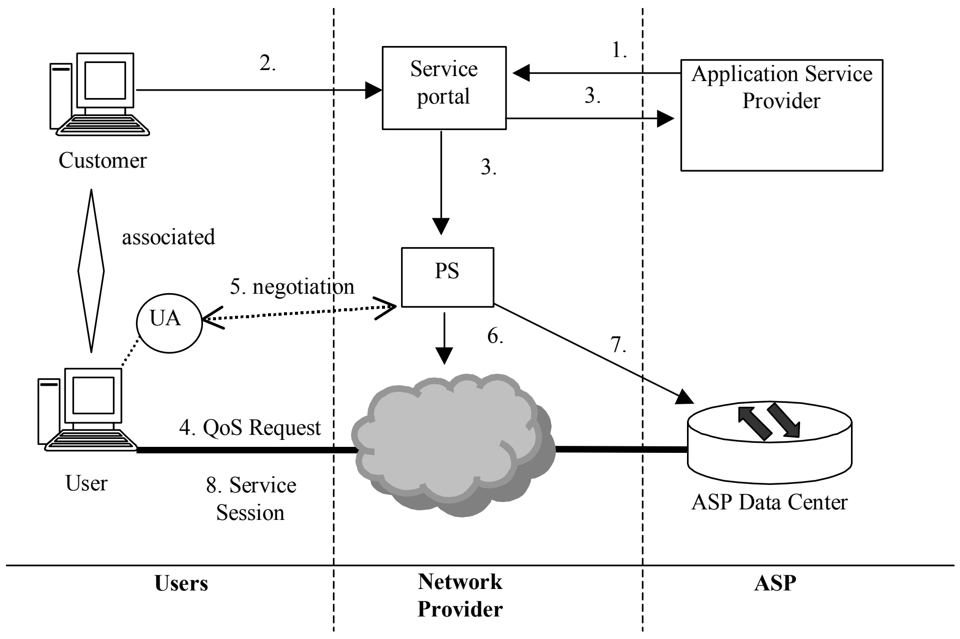
[16] F. Le Faucheur, S. Davari, P. Vaananen, R. Krishnan, P. Cheval and J. Heinanen, "MPLS Support of Differentiated Services", IETF Internet Draft: <draft-ietf-mpls-diff-ext-07.txt>, March, 2000.

Athanasios G. Papaioannou received his Diploma in Computer Science (1998) from the University of Crete, Heraklio, Greece, and the M.S. degree (2000) in Networks and Telecommunications from the Computer Science Department of the University of Crete, Heraklio, Greece. From 1996 to 2000 he was a member of the Telecommunications and Networks Division of the Institute of Computer Science, Foundation for Research and Technology Hellas (ICS-FORTH), as a research trainee (1996-1998) and a research assistant (1998-2000). Since May of 2001 he is a PhD student in the Department of Informatics of Athens University of Economics and Business. His interests are in QoS protocols for telecommunication services, QoS negotiation and provision in Internet, and accounting and charging for telecommunication services.

Stelios S. Sartzetakis received his Ph.D. degree in Electrical and Computer Engineering from the National Technical University of Athens, his M.Eng. in Systems and Computer Engineering from Carleton University of Ottawa, Canada, and his B.Sc. degree in Mathematics from Aristotelian University of Thessaloniki. He heads the telecommunications and networks laboratory of ICS-FORTH. He participated and coordinated more than ten international research projects. He contributed to the research in the area of broadband network management, network performance evaluation, QoS and traffic management, charging and accounting of ATM and Internet traffic and services, and the design of large optical networks. He was principal in the creation of FORTHnet, the first Internet Service Provider in Greece. Dr. Sartzetakis is also visiting professor at the Dept. of Computer Science of University of Crete. He has a research record of over twenty-five publications in refereed conferences, journals and book chapters, and serves as a reviewer and member of several conference committees. He worked for years as an independent consultant for private, public companies and governments, in Greece and abroad.

George D. Stamoulis received the Diploma in Electrical Engineering (1987, with highest honors) from the National Technical University of Athens, Greece, and the M.S. (1988) and Ph.D. (1991) degrees in Electrical Engineering from the Massachusetts Institute of Technology, Cambridge, Massachusetts, USA. From 1991 to 1993 he was a Research Associate with the Communication Networks group at the NTUA, participating in RACE projects, while serving in the Hellenic Navy, as a lecturer in the Hellenic Naval Academy. From 1993 to 1995 he was with the Development Programmes Department of INTRACOM, as a coordinator of RACE projects. From 1995 to 2000 he was an Assistant Professor at the Computer Science Department of the University of Crete, Heraklion, Greece, and a member of the Telecommunications and Networks Division of the Institute of Computer Science, Foundation for Research and Technology Hellas (ICS-FORTH). Since the end of 2000 he is an Associate Professor in the Department of Informatics of Athens University of Economics and Business (AUEB). His research interests are in network economics, agents for optimal selection of service contracts on behalf of the user, auction mechanisms for bandwidth and user strategies, and charging for telecommunications services.

LIST OF FIGURES



2.
Service portal
1.
3.
Application Service Provider
Customer
3.
associated
PS
5. negotiation
UA
6.
7.
4. QoS Request
User
8. Service Session
ASP Data Center
Users
Network Provider
ASP

Figure 1.

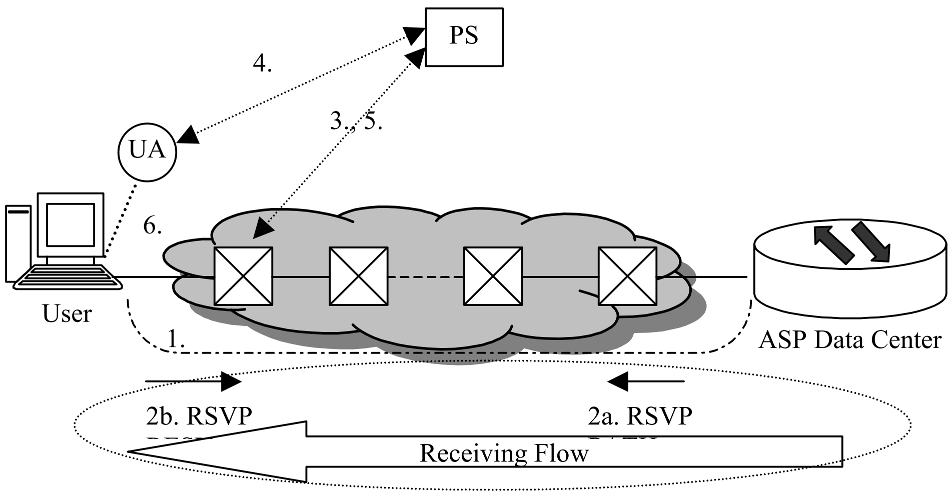
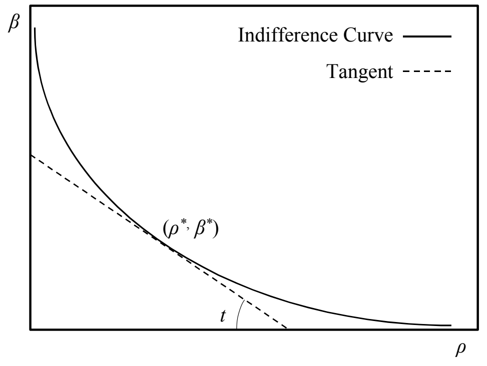
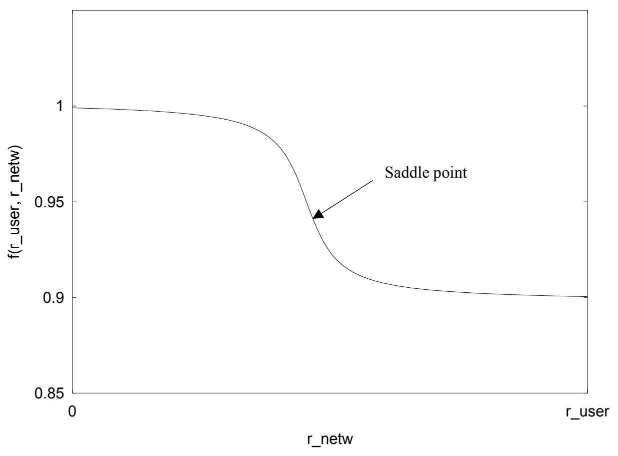

PS
4.
3., 5.
UA
6.
User
1.
ASP Data Center
2b. RSVP
2a. RSVP
Receiving Flow

Figure 2.



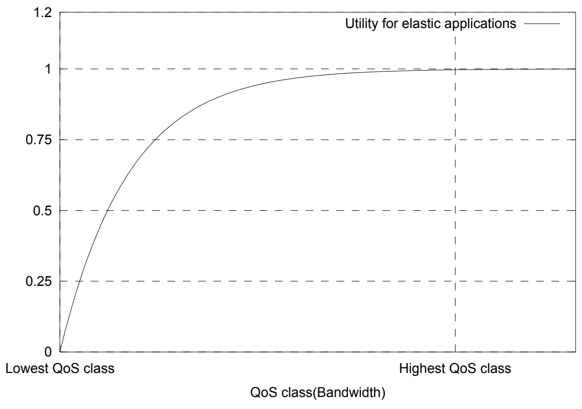
β
Indifference Curve
Tangent
(ρ^*, β^*)
t
ρ

Figure 3.

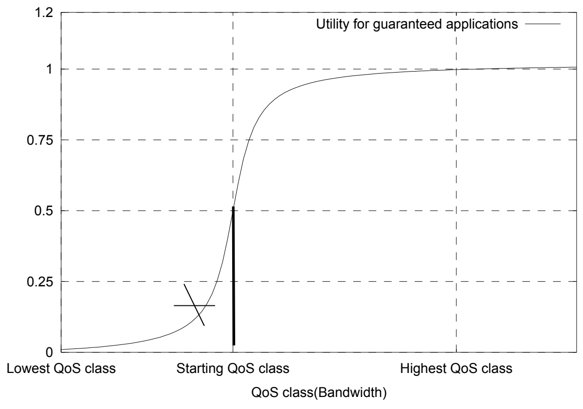


f(r_user, r_netw)
1
0.95
0.9
0.85
0
r_user
r_netw
Saddle point

Figure 4.


Utility for elastic applications
1.2
1
0.75
0.5
0.25
0
Lowest QoS class
Highest QoS class
QoS class(Bandwidth)

(a)


Utility for guaranteed applications
1.2
1
0.75
0.5
0.25
0
Lowest QoS class
Starting QoS class
Highest QoS class
QoS class(Bandwidth)

(b)

Figure 5.

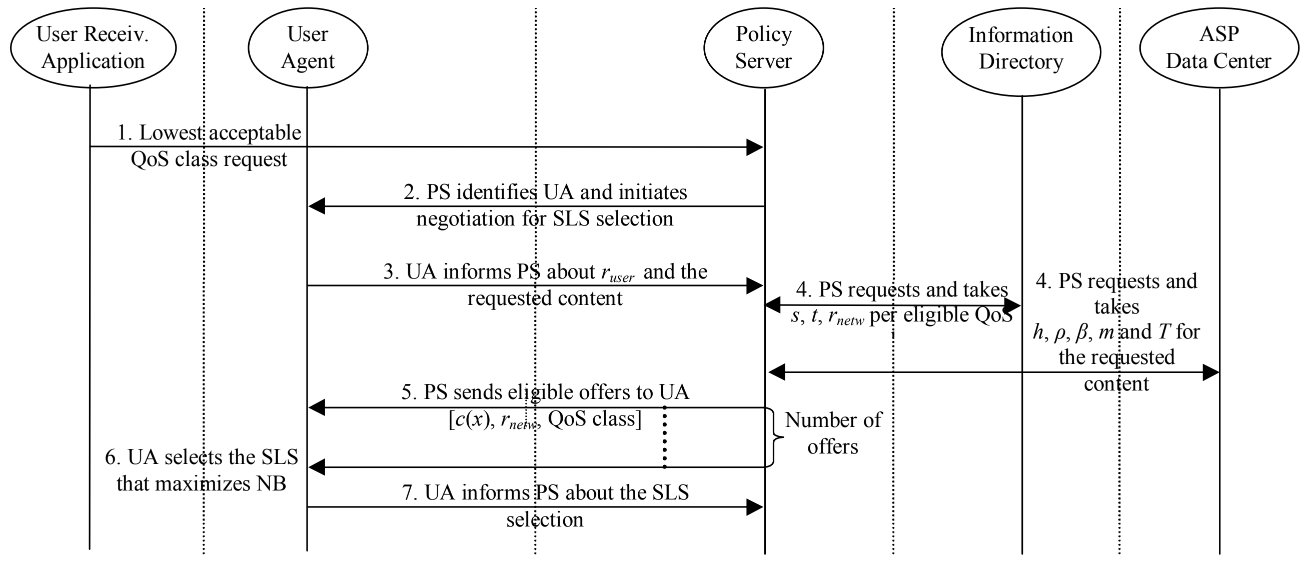
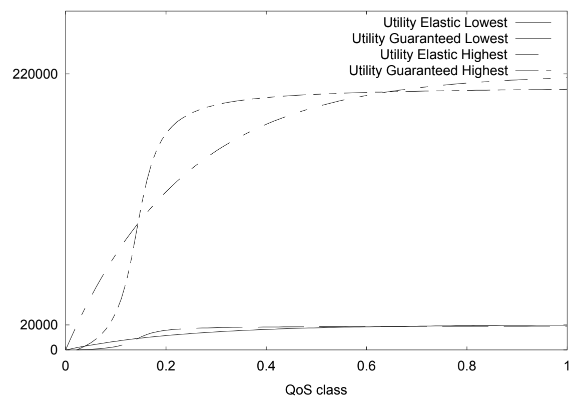
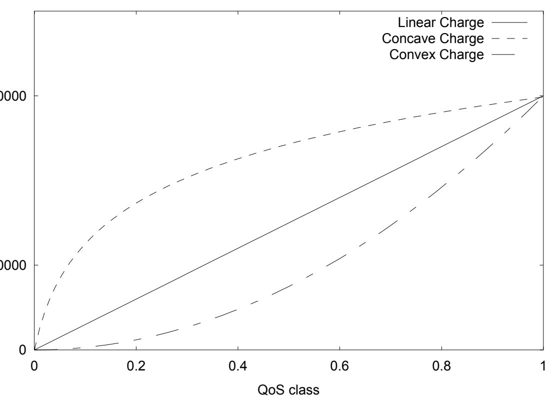

User Receiv. Application
User Agent
Policy Server
Information Directory
ASP Data Center
1. Lowest acceptable QoS class request
2. PS identifies UA and initiates negotiation for SLS selection
3. UA informs PS about r_{user} and the requested content
4. PS requests and takes s, t, r_{netw} per eligible QoS
4. PS requests and takes h, ρ, β, m and T for the requested content
5. PS sends eligible offers to UA [$c(x)$, r_{netw}, QoS class]
Number of offers
6. UA selects the SLS that maximizes NB
7. UA informs PS about the SLS selection

Figure 6.



Utility Elastic Lowest
Utility Guaranteed Lowest
Utility Elastic Highest
Utility Guaranteed Highest
220000
20000
0
0
0.2
0.4
0.6
0.8
1
QoS class

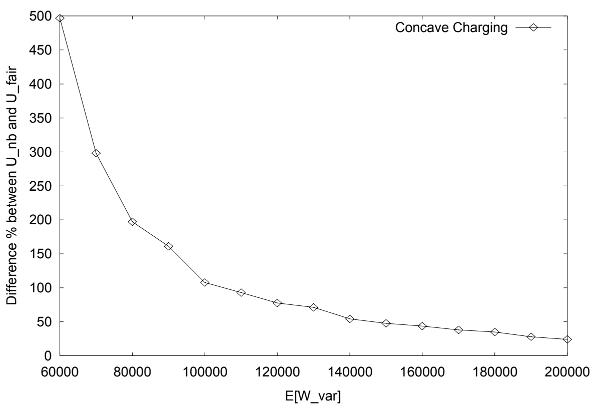
(a)

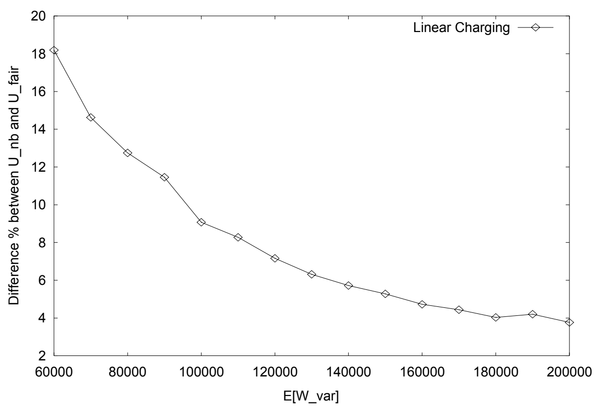


Linear Charge
Concave Charge
Convex Charge
60000
20000
0
0
0.2
0.4
0.6
0.8
1
QoS class

(b)

Figure 7.


Concave Charging
Difference % between U_nb and U_fair
E[W_var]
0
50
100
150
200
250
300
350
400
450
500
60000
80000
100000
120000
140000
160000
180000
200000


Linear Charging
Difference % between U_nb and U_fair
E[W_var]
2
4
6
8
10
12
14
16
18
20
60000
80000
100000
120000
140000
160000
180000
200000

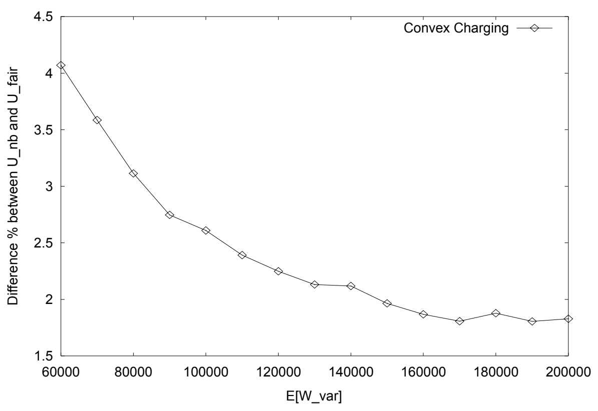

Convex Charging
Difference % between U_nb and U_fair
E[W_var]
1.5
2
2.5
3
3.5
4
4.5
60000
80000
100000
120000
140000
160000
180000
200000

Figure 8.

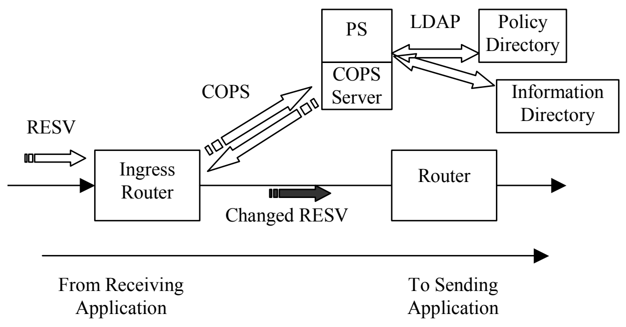

PS
LDAP
Policy Directory
COPS Server
COPS
Information Directory
RESV
Ingress Router
Router
Changed RESV
From Receiving Application
To Sending Application

Figure 9.