SLS to DiffServ configuration mappings

Alberto Gonzalez Prieto, Marcus Brunner

Network Laboratories, NEC Europe Ltd., Adenauerplatz 6, D-69115 Heidelberg, Germany
Phone: +49 (0)6221/ 905 110 Fax: +49 (0)6221/ 905 1155
alberto@ccrle.nec.de,brunner@ccrle.nec.de

This document addresses the problem of mapping Service Level Specifications (SLS) to IP Differentiated Services (DiffServ) configuration. We introduce a two step mapping controlled via a policy-based management system. The two step mapping includes the service specification to intra-domain service mapping (Per-Domain Behavior (PDB) [PDB-DEF]) and the further mapping to the DiffServ mechanism available in the domain. The first step uses an N-dimensional room (e.g. including delay/jitter, loss and throughput) to classify the SLS into a limited set of available intra-domain services. Based on this classification, assignment of the service class, and per service class admission control is performed. The mapping system is implemented on top of a QoS Management API configuring our Linux-based DiffServ routers.

Keywords: Differentiated Services (DiffServ), Service Level Specification, QoS Management in the Internet, QoS mapping, Bandwidth Brokering

1 Introduction

The globalization and commercialization of the Internet, accelerated by IP technology's ubiquity and its flexibility for introducing and distributing new applications, has been a tremendous success story of our time. The Internet of today is a global, commercial communications infrastructure supporting a tremendous amount of information, and a diverse set of applications for and across businesses, organizations, and individual users. The emergence of distributed multimedia applications and the growing mission-critical use of the Internet highlights the need for more reliable, secure, assured, and high performance communications and delivery of application-level services.

The challenge of supporting a variety of applications with differing characteristics and requirements at adequate service levels has led to the development of Quality of Service (QoS) technologies and enabling mechanisms in the past few years. Scalable, reliable management and control of QoS will support further rapid growth of the Internet, laying the basis for effective support of even more diverse and distributed multimedia applications. The Internet Engineering Task Force (IETF) proposes the Differentiated Services (DiffServ) Architecture [DS-ARCH] as a basic mechanism for providing QoS in the Internet. DiffServ keeps its scalability by aggregating traffic flows into service classes, which are then handled differently within the network. So on the high-speed data path, only a few service levels are provided, and no per-flow state is kept. The challenge, however, lies in the control of edge-to-edge services, which may be provided to customers of an Internet Service Provider, or which are used to build true end-to-end services. Edge-to-edge services are intra-domain services, which are provided from one edge (ingress router) of a administrative DiffServ domain to the other edge (egress router). We only address the edge-to-edge services in this paper.

The basic problems in providing edge-to-edge services include the difficulty to design services, which on one hand can get sold to customers meeting their requirements, and which on the other hand can be provided by a DiffServ-enabled IP network. The issues are the finding of services specifications, finding the right DiffServ mechanisms (the IETF standard is very open in determining the mechanisms), and derive the right configuration in order to provider the service.

Various frameworks and architectures already exist in the literature [TEQ, CADENUS, AQUI, CSM]. Most of them postulate a Service Level Specification (SLS) to configuration mapping function in their architectures. The function basically takes a SLS and derives a configuration of the various entities in the DiffServ network. The function also includes an admission control part, determining whether to permit a service request. However, to the author's knowledge, non of them presented a system design for that functionality.

In this paper, we focus only on this mapping part. We propose a two step mapping from a SLS to a per-domain behavior (PDB), which is the edge-to-edge service determined by a network administrator. The second step of the mapping includes the PDB to configuration mapping, which includes determining the DiffServ Code Point (DSCP) to use, and the edge-router configuration to perform. The admission control is performed after the determination of the PDB in order to enable different admission rules for various PDBs. Some of the nice features of our approach are its high modularity, which makes it easy to extend, the high level of configurability and dynamicity, and that it is open to policy-based control.

2 Background

The environment we are focusing in this paper is an Internet Service Provider's network, where various kinds of customers are connected and different types of services are requested. The customers may be home users, business users, or other ISPs. In Figure 1, we show an ISPs network between two other administrative domains. A customer negotiates a Service Level Agreement (SLA) with that ISP. The specification of the service in a technical manner may be included into that negotiation. We refer to the technical part with the term Service Level Specification (SLS). The SLA includes legal, administrative, and economic information such as the prices of services. Furthermore, an SLA may include not only transport services, but also application-level services such as Web-hosting, E-Mail forwarding, etc. However, this paper only addresses IP-based transport services. The SLS does not need to be negotiated together with the SLA. A SLA may not include a service specification itself, but may contain a list of services the customer can request on demand [CSM]. For instance, the SLA negotiated allows a customer to get best effort service all the time and 10 hours of premium virtual wire service a month. After negotiating the agreement, the customer is added to the best effort service immediately. At a later point in time the customer may request the virtual wire service.

Furthermore, we assume that the SLS we receive from another entity is a service request for only the administrative domain under control. However, the component described in this paper is a prerequisite to provide end-to-end IP services in a multi-domain scenario as well. Providing true end-to-end services, more issues need to be addressed than we are able to solve and describe in this paper. For instance, the service subscription model may differ, the flow of money is different, requests need to be broken down to requests for each single domain. The inter-domain management of IP services is quite open and needs more research also in the area of inter-domain traffic engineering, inter-domain QoS routing etc.

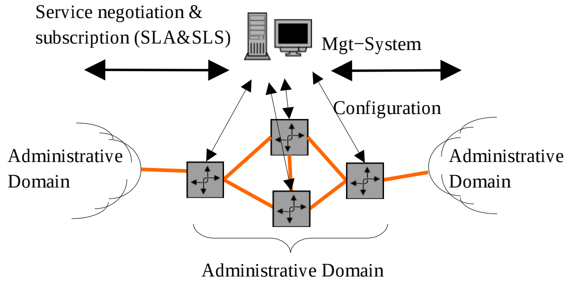

Fig. 1: Service Level Agreements and Specifications in Inter-domain Scenarios

Providing QoS in IP networks is bound to routing as well. In order to provide guarantees, the path IP packets are forwarded needs to be under control. We assume to have static routing in the network, which is a common operation method in today´s backbone networks. A different technology currently in the standardization process is Multi-Protocol Label Switching (MPLS). MPLS allows a network operator to explicitly configure the route of a set of packets. Having configurable or fixed routing, admission control, and resource reservation in place, guarantees on that path can be provided by the network. Note, that routes can be changed, but admission control and reservations on the new path needs to be performed prior to the change. So the change frequency has an impact on the dimensioning of the configuration system.

2.1 The General Architecture of the Configuration System

Figure 2 focuses on certain parts and functionalities of a management system, which is important as background information to the work presented in this paper. We only focus on the technical part of the SLA, but the SLA is still indicated. Furthermore, we exclude issues for monitoring, accounting, and other types of management tasks in IP networks.

Basically, we can divide the management system into two portions, one containing the SLS mapping, admission control, and per-SLS configuration. The other part includes the policy-based control and configuration part, where the SLS mapping and admission control is controlled and configured via policies. Additionally, the basic network configuration is given by policies. However, the relationship between the device-level policies, the configuration component, and the router is controversial, and therefore is outlined by dashed arrows in Figure 2. It heavily depends on the configuration means available for the router. A router having a COPS- or SNMPconf-based [†] interface may send policies to the router. In this case the network configuration component will basically produce policies, specific to the SLSs, and pass them to the device-level configuration entity. On the other hand, if the router interface is CLI- or SNMP-based, the configuration is mainly performed via the network configuration. Any mixture of this may be implemented as well. Actually, this is one of the reasons for introducing the QoS Management API into the system [API]. It abstracts from the underlying router access technologies as much as possible and allows to write access independent QoS control and management software.

Our paper addresses the first part, mainly the SLS mapping, admission control, and per-SLS configuration. A SLS introduced into the management system, e.g., via a Web interface, will be stored together with information about the customer requesting the service. It is then forwarded to the component labeled "QoS mapping, admission control, configuration decision". The configuration derived is then really configured in the network via the QoS Management API (see Section 4).

2.2 Service Level Specification

As a starting point we take the Service Level Specifications submitted to the IETF by the TEQUILA project [TEQ] including the parameters proposed in it. In the following we list the parameters with a very short description. **Scope** specifies on what paths the QoS policy is to be enforced. It is expressed by a couple of ingress and egress interfaces. Note that the scope parameters enables one-to-one, one-to-any, and any-to-one services. **Flow Description** indicates for which IP packet flow the QoS guarantees need to be enforced. For instance, it specifies the subnet the packets are coming from. **Traffic Envelop** describes the performance characteristics of the packets identified by the flow description. **Excess Treatment** describes how excess traffic will be processed. With excess traffic we refer to packets, which do not conform to the specified Traffic Envelop. E.g., they may be remark, dropped, shaped etc. **Performance Guarantees** describe the service guarantees offered for the packet stream described by the flow description and over the geographical/topological extent given by the scope. Performance parameters are delay, jitter, packet loss, and throughput. Note that the parameters do not need to be guaranteed deterministic, but also statistical guarantees are possible. For instance, delay smaller than 70 ms for 95% of the packets. **Service Schedule** indicates the start and end time of the service. **Reliability** indicates the maximum allowed mean downtime and the maximum allowed time to repair.

[†] With SNMPconf we refer to the work done in the IETF working group on Configuration Management with SNMP, where a way to use SNMP for policy-based management is defined (http://www.ietf.org/html.charters/snmpconf-charter.html).

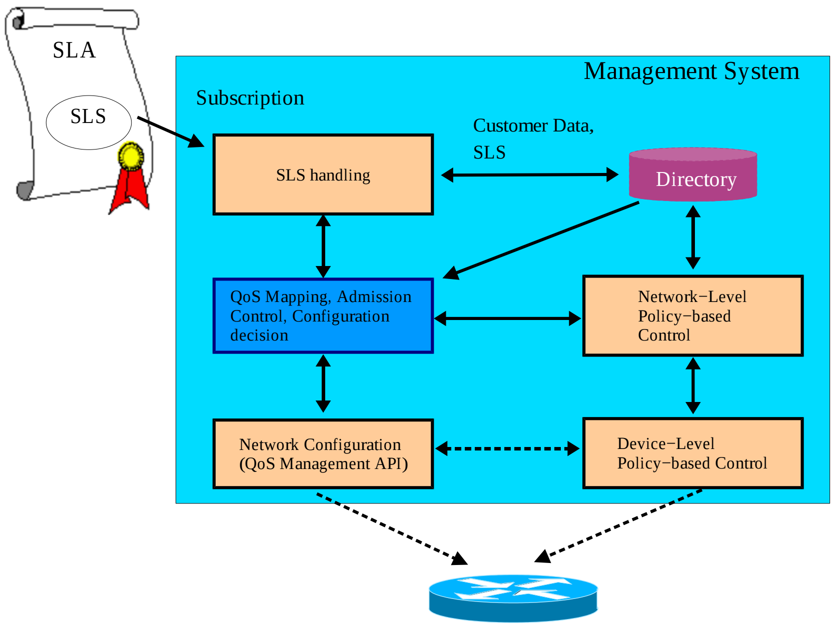
SLA
SLS
Management System
Subscription
Customer Data,
SLS
SLS handling
Directory
QoS Mapping, Admission Control, Configuration decision
Network−Level Policy−based Control
Network Configuration (QoS Management API)
Device−Level Policy−based Control

Fig. 2: Configuration Management System for DiffServ Intra-domain QoS Control

Note that we take into account quantitative as well as qualitative specifications. For instance the performance guarantee for packet losses may be "very low", "low", and "reasonable". We come back to the mapping of qualitative to quantitative numbers later.

3 Mapping Schema

In this section, we present our two step schema to map SLSs to PDBs and further to configurations. Figure 3 shows the steps as an overview of the components involved in the mapping process. The first step given a SLS is to choose the PDB which will be deployed to guarantee the customer requirements in the SLS. The second step includes finding the right configuration based on parameters of the SLS and the chosen PDB.

The SLS to PDB mapping process is broken down into pre-processing the SLS for different purposes and then based on the pre-processing choose the PDB. We pre-process the SLS since it may not be expressed as a list of parameters and values, which is what the Mapping Schema is expecting, and what the network can deal with. The SLS might be expressed in terms of "Service X", which may be a *pre-defined* service [AQUI]. These pre-defined services are mapped to the SLS parameters proposed in [TEQ]. Furthermore, the SLS could also be expressed, for simplicity, in terms of low/medium/high delay/loss/.... These values are mapped to numeric values, so that our system can deal with them.

After this pre-processing, we have the service request converted into a list of parameters containing numeric values. Based on the values, we choose the PDB which can guarantee those values. First, we evaluate those parameters which can only take a determined value and would not make sense to change them, such as the *scope*. These are the *pre-evaluated parameters*. Then, we evaluate those parameters which can take a range of values and could be changed while offering a service close to the requested one. This distinction will be useful when looking for offering a *similar service* to the customer. This is done at the PDB Selector.

After the PDB is chosen, we perform the second step in the overall mapping process, which maps the chosen PDB to the network configuration. This step includes checking whether there are enough free resources to guarantee the service requirements. If there are, we configure the routers such that the service can be provided. The configured routers are mainly the edge ones. About the core routers, we only check whether its configuration allows to offer the service and we keep track of the services that traverse them.

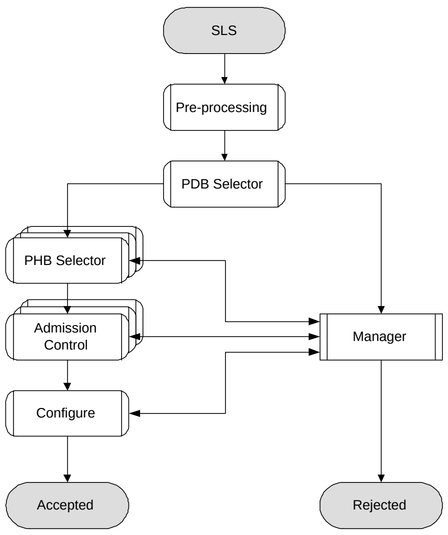

Fig. 3: Mapping Schema

A component not directly involved in the mapping process is the block named *manager*. When an exception occurs in the mapping process (such as the impossibility to find a PDB, etc ...) , the manager block is involved in the decision process. This block is regarded a interface to or implementing itself a policy decision engine. The policy engine allows an operator to specify rules supporting the decisions to be taken by the manager block. Furthermore, that block may contain an interface to an operator terminal to include human decisions into the mapping process. Although the mapping process is designed to be automatic, the schema is open to provision services manually, skipping as many automated steps as wanted. In the following, we describe every block in the mapping schema in more detail.

3.1 SLS pre-processing

SLS pre-processing deals with predefined services and the mapping of qualitative to quantitative parameters values. An SLS could be expressed as a *pre-defined* service [AQUI]. They are SLS templates with fixed values (or ranges) for some of the parameters in the SLS. So, the customer would not need to specify all the parameters but only request for a determined *pre-defined SLS* and specify only a few parameters (or none). Some customers may even prefer to ask for a "standard service" rather than to have to deal with several parameters in various offerings (may especially hold for home users). The existence of *pre-defined* SLS is useful for the customer and the network administrator, but the network deals with concrete values, so the *pre-def* block maps "service X" (which could be "medium quality videoconference") to the appropriate SLS parameters, both qualitative and quantitative. For instance, "service X" could be mapped to a one-to-one, 2 Mbps, low delay SLS. Note, that in this step still qualitative parameters are allowed.

The network deals with quantitative values not with qualitative ones, so the former ones are mapped to quantitative values or value ranges (e.g: low loss = less than 0.01%, medium loss = (0.01% , 0.1%)). This mapping gets configured by the network administrator considering the characteristics of the network. Most likely the mapping is expressed in policy rules and the configuration of this component is done via a policy service.

After the pre-processing, we have a list of parameters and their numeric values or ranges. The rest of the Mapping Schema will not need to be aware how the SLS was expressed (pre-defined Services, qualitative values, ...). It will deal only with the list of parameters and their values. Note that this mapping is sort of an adaptation of the SLSs to be understood by human beings, and may be done at the user interface level. However, since the mapping is under control of the network administrator, we combined the functionality with the other mapping and configuration processes needed.

3.2 PDB Selector

The PDB selection process is performed by first pre-evaluating parameters in order to select the appropriate selector. The selector really chooses the PDB to be used.

3.2.1 Pre-evaluation of Parameters

Pre-evaluating parameters vary from pre-processing as described before. Where pre-processing maps various representations of a SLS into real parameter values, the pre-evaluation starts taking this parameters into account for evaluating what PDB to use. Basically, we differentiate the SLS parameters according to the way they are treated in the PDB/PHB selection. We have three classes of parameters, namely non-used, pre-evaluated, and performance parameters.

Parameters *not used* in this step are those which have nothing to do with the PDB/PHB selection. They are used in later steps. This parameters include *flow description* and *excess treatment* to be used only for configuring the edge router´s classifier and traffic conformance algorithms. Additionally, *schedule* is used only in the admission control to find whether there are enough resources available at a certain time. The *reliability* parameters have no influence on the decision. It depends on the technology used in the network, and may therefore influence the configuration, e.g., whether to install a backup path or not. Furthermore, it can be checked in the admission control to find whether the requirements are met.

Pre-evaluated parameters take only a small set of diskrete values and a change of them would mean to change the service to a completely different one. This distinction will be useful when looking for offering a *similar service* as explained in 3.5. We consider the *scope* as a parameter to pre-evaluate, because various types of *scope* such as “one to one”, “one to any”, “one to few ”. . . are inherently different kind of services, and need a different handling starting at this point in the mapping schema. However, the schema is opened to include other parameters if needed. The *scope* may affect the decision of the PDB or PHB to choose as stated in [ONE2ANY]. Additionally, really changing the scope of a service makes only little sense.

Performance parameters may contain various values within a range and changing the value still keeps offering a similar service compared to the requested one. The set of parameters include at least delay, jitter, packet loss, and throughput. For instance, it may make sense to offer a downsized service, e.g., 2 Mbps instead of 3 Mbps, if not enough resources are available. The other parameters however stay the same.

Conceptually, we use the *pre-evaluated* and the *performance* parameters to select the PDB. However, we break the selection process into choosing the selector based on the pre-evaluated parameters and the PDB is chosen based on the performance parameters (see the next section).

3.2.2 Performace-PDB Selector

The *performance-PDB selector* is in its current version a 3-dimensional sub-space of the overall N-dimensional SLS parameter space. In this section, we describe how the selector works.

Every axis represents one of the *performance* parameters (delay/jitter, packet loss, throughput). Regarding delay/jitter, the closer to the origin of the axis the lower the delay. In the first version we will consider only delay. Since delay and jitter are related and controlling jitter is more complex and can be compensated by increased delay. A zero or negative value indicates unspecified. The closer the packet loss is to the origin of the axis, the higher the loss probability. A negative value indicates unspecified. In the throughput axis, a negative value indicates unspecified. Since in general, PDB attributes will be expressed as bounds or percentiles rather than as absolute values [PDB-DEF], the service will be represented as a zone rather than as a point in the N-dimensional space.

The first thing a network administrator should configure into the PDB Selector is the mapping of available PDBs in his network to the PDB Selector. Every PDB would cover a determined zone considering the performance values it can guarantee. This may depend on the technology of the network, on its topology, and on other issues. In [PDB-DEF] it is stated that any PDB specification must include (among other issues) the PDB attributes, that is: how the PDB behaves. This might include drop rate, throughput, delay bounds, etc . . . . This information is used to find the zones a PDB can cover.

When all available PDBs have been mapped, forbidden zones arise which would be the zones in the space no PDB covers, and so there will be regions of *unoffered services*. A service can be considered unoffered due to technical or business-related reasons. Technically, it might be impossible for a network to offer a particular service, because of the topology or technology used. On the other hand, there might be some services the provider does not want to offer from a business point of view, although there is no technical impediment. There are two different situations. First, there are services which may not be worth offering (for the ISP), for instance bandwidth requirements below 1 Mbps could be considered economically unworthy. So, the provider may decide not to offer those services. The second case is what we call unaffordable services, meaning extremely expensive services. An example would be a one-to-any Virtual Wire service [VW], or bandwidths over 100 Mbps . The provider might decide not to offer this services to normal customers, but some customers may get them because of their very important customer (VIC) status. The concept of VIC status of a customer arises from the interest of an ISP in treating some customers better than others. This better treatment might be offering services not offered to the rest of users, accepting a request in the admission control that otherwise would have been rejected, etc . . .

If the service request hits an *unoffered services* region, the *manager* block gets involved and it is asked whether to reject the request (default) or take any other actions, e.g., interactively ask for decision via console or by means of stored rules in the policy engine.

3.3 PHB Selector

Once the PDB is chosen, a Per-Hop Behavior (PHB) needs to be determined. Although (as it is stated in [PDB-DEF]) it is expected that a single domain will use a single PHB to implement a particular PDB, the *PHB selector* block is introduced, because the IETF standard is open to use more than one PHB. In addition, the PHB must be mapped to a DSCP [DS-FIELD] to be marked. This mapping must be configurable. Although every PHB has a recommended associated DSCP, every domain can choose the DSCP to deploy a PHB (with the exception of the Class Selector Codepoints).

3.4 Admission Control

Once the PHB is chosen, the *admission control* block decides whether there are enough resources in the network to provide the service. In case the resources are available, the *Configuration* block configures the nodes. In case there were not enough resources, the *Manager* block decides whether to reject the request or to offer a slitely different service.

The Admission Control schema is shown in Figure 4. It consists of two branches, one of them determines the route the packets will use, and so the nodes that need to be configured or at least checked for resource availability. The other branch determines the requirements the service needs, including reserved bandwidth, total delay, buffer requirements, etc . . . . When all the nodes to configure and the requirements are known, we check whether the requirements are met at all nodes in the *resource checking* block.

Note that the Admission Control Block may be empty for certain PDBs, because they rely on traffic engineering and network planing for providing guaranteed services.

The route determination branch (on the left) is responsible of determining the nodes that need to be configured, so that the traffic will meet the agreed service. The first node to determine will be the *ingress node*. It may be specified in the SLS or may not. In the latter case the ingress would be determined considering the closest node to the source in the provider domain. Where the closest node means the edge router where packets destined for the source would leave the domain. It can be easily found using the routing tables in the ISP routers. Once the *ingress node* is determined, the rest of nodes are determined using the routing tables in the nodes, until the *egress node* is reached. The egress is determined since the next hop would be to a node not belonging to the domain. At this point, the system knows the set of nodes that need to be checked to decide whether there are enough resources to offer the service. We use a simple route determination algorithm since the aim of the work are not admission control algorithms, but a mapping system. More complex algorithms could be easily plugged-in in our schema. For instance, another routing schema could be used (QoS routing, MPLS) as long as they find a path which satisfies the service requirements for the whole duration of the service, independently of the rest of services (ongoing and future ones).

The requirement determination branch (on the right) will pass the service requirements to the *resource checking* block. They will be decided depending on the PDB and PHB chosen and the values in the SLS. The requirements may include bandwidth, delay/jitter, buffer size and schedule. For most cases, this block

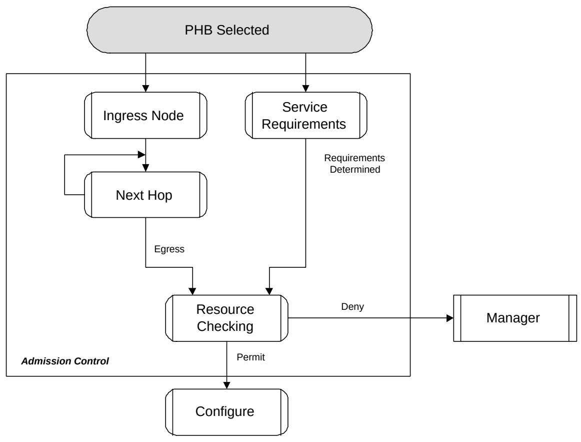

Fig. 4: Admission Control

is supposed to be very simple, it just forwards the SLS values to the *resource checking* block. However, in some cases it might be more complicated. An example service is one only assured with a certain probability such as in [ONE2ANY]. Even though requested for a determined bandwidth, not the whole bandwidth needs to be available in all the nodes. So, calculations may be done by this block to find out the free resources needed to guarantee the service. For instance, a service request may be admitted if 80% of the requested bandwidth is still free. This is very common to services statistically guaranteed. These requirements are the ones we have to check against every core router configuration. The requirements need to be met in every node.

After the two branches have determined the service requirements and the route, the two possible answers of the *admission control* block are to permit or deny the service. The decision is based on the amount of free resources to guarantee the service. The *resource checking* block decides on the aceptance considering the information provided by both branches in the *admission control* block. If the answer is "permit", the network is able to offer the requested service, so the *configuration* block will configure the nodes selected in the "route determination" branch. Note that, most likely, only the ingress node is configured whereas interior routers are configured by a separate configuration system (device-policy service in the overall picture). If the network is configured properly, then the customer is informed that the service is permitted. In case any error occurred during the configuration, the *manager* block is informed and decides about the action to be taken. E.g., try to configure the network again (may be a temporary fault), reject the service, or inform the fault management system. On the other hand, the decision of the *resource checking* might be "deny" due to different causes such as not enough bandwidth available, impossibility of achieving the required delay, etc. Actually, the decision is not just a "permit" or "deny". In the latter case it is specified why the decision is "deny". The *manager* block is informed about the reason(s) of the negative decision. The information can be a hint on what is missing, or it could be quantitatively specified what is missing in order to permit the service request. An example of a quantitative infrom is a service requests for 8 Mbps but the maximum available bandwidth on the path is 6 Mbps. Note that the available bandwidth is considered for the whole path. Although we find along the path a link where the available capacity is lower than 8 Mbps, (e.g: 7 Mbps) we check the whole path to the egress node, in case there was a link with even lower available capacity (e.g: 6 Mbps). So the found available capacity is the one that can be really offered to the customer. With this information the Manager Block is able to decide what to do. In case it decided to offer a similar service, it knows what can be offered to the customer and so the service request only needs to be evaluated by the mapping schema at most twice. It does not need to keep trying different performance specifications.

In addition, the resource checking may be configured such that it denies requests even if there are still resources available. We call the parameter to be set the utilization boundary. This is a nice feature for network operators dividing its customer base into groups of customers, some of them having a VIC status.

So the resource are only spent on premium customers. For instance, a utilization boundary at 80% of the available bandwidth is specified. So the decision of the resource checking evaluates to deny if less than 20% of the bandwidth is free. The information passed to the manager block is “deny, less than 20% of bandwidth free”, considering the 20% as a guard. The Manager block decides based on policy rules and the customer requesting whether to really reject the service or using part of the guard for that service. The decision heavily depends on the business model and whether the ISP really groups customers.

Keeping track of available resources in the network, two mechanisms may be used. In our case, we basically track the resource consumption and reservation within the management system. However, in some cases the model of the network does not correspond with the reality, in which case monitoring of the network is needed. Furthermore, the network resources to be distributed to customers by this system can be read from the network directly, or can get read from another configuration system. The configuration of the routers including classifiers, meters, action elements (markers, droppers, multiplexors, counters, . . .), and the queueing elements (schedulers, buffers, . . .) [DS-MODEL] needs to be known by the system. Another issue to be considered is how to include the real utilization of the network into the decision process.

3.5 Manager

The *manager* block gets involved in many different cases, e.g. when an exception arises. Basically, it is used only when a decision along the mapping path can not be made by a single block on its own. Although the process is automated, the idea behind introducing the *manager* block is to make it work as a interface to human beings and to consider non-technical issues as well. It can decide to skip steps in the mapping schema or changing the decisions other blocks take according to business (or any other non-technical) issues. The decisions it takes are based on policies in a form like IF situation A THEN take action “alpha”. The manager block basically has to deal with three kind of situations, *unoffered services* , rejections by the *admission control* block, and errors while configuring the network.

The *unoffered services* case takes place when the PDB Selector block finds that no available PDB in the network guarantees the requested service. There are two different situations, depending on the cause which led to the unoffered situation (Section 3.2). The technical issues, where the network is not able to offer the service because of the topology or technology implemented. In this case the Manager has to reject the request most likely. But it may try to offer a similar service (see below). The second reason are business-related issues. Although the service could be offered, the provider does not want to offer them , at least by default. The decision might be to reject the request or accepting it, according to what kind of customer is requesting. In the latter case, the service would be forwarded to the Admission Control block and decided whether there is enough resource available.

Rejection by the *admission control* block occurs when not enough free resources are available. Two cases for this situation were introduced in section 3.4 and will be analyzed separately. First, there is the case when there are really no free resources. In this situation, the Manager will be told the reason of the negative answer. Using that information it could check if a similar service can be offered (by default) or reject the request. The second case is when utilization boundaries are set and reached. The manager will have to decide whether that guard is used or it is kept free. The decision is likely to depend on business-related issues. For example, customer X would be allowed to use up till 10% of the guard bandwidth.

The manager also has to deal with configuration errors, since there might arise some problems when configuring the network. They may be due to link or node failures or errors in the mapping process. In front of this situation, the manager would just reject (by default) the request and inform of the error to another block, out of the Mapping Schema, which was in charge of maintenance.

In case the requested service could not be offered, the manager can decide to check whether a similar service can be offered. This process can only take place in the cases of *unoffered service* or admission control rejection, but not in the configuration error. A similar service denotes a service close to the original one in the *Performance-PDB Selector space*. An example would be a variation from the original in 25% of bandwidth and 15% of delay. This variation would be noticed in the PDB Selector since the service is represented by a larger zone than before. This change of the service request leads to a re-negotiation of the SLS, so that the customer could accept or reject the new offer.

3.6 Example

This example shows, how the mapping systems works with the following SLS. **Scope** ingress = A, egress = B; **Flow description**: IPsrc=C, IPdest=D, dest port=80-90; **Traffic envelop**: Token Bucket (50 Mbps, 1500 bytes); **Excess treatment**: drop; **Performance parameters**: throughput =50 Mbps, delay = low; **Schedule**: from now on, during 2 hours.

The pre-processing block needs to map the qualitative “low” delay to lets samaller than 20 ms. The rest of the SLS is already expressed as a list of parameters and their numeric values. Assume that the PDB Selector is configured such that no PDB covers that zone -¿ unoffered service. The closest zone is the one covered by the Virtual Wire PDB, but it only covers up to 20 Mbps not more, since the ISP does not want to offer high throughput services with delay guarantee. So, the PDB Selector forwards the request to the Manager. This one, considering the ”very important” (VIC) status of the customer, overrules the decision and offers that service and selects Virtual Wire as PDB to deploy. The Manager then forwards the request to the PHB Selector which selects the Expedited Forwarding PHB and 101110 DiffServ Codepoint. The Admission Control block checks whether there are enough free resources. Assume that it founds that if the service is offered an utilization threshold will be reached, then forwards the request to the manager which according to the status of the customer accepts the request and asks the configuration block to configure the appropriate nodes. The ingress node is configured with a classifier where the flow description parameters are used, a meter with the traffic envelop parameters, excess treatment set to drop, and a marker with the selected DiffServ Codepoint. We keep track of the resources assigned.

4 Implementation

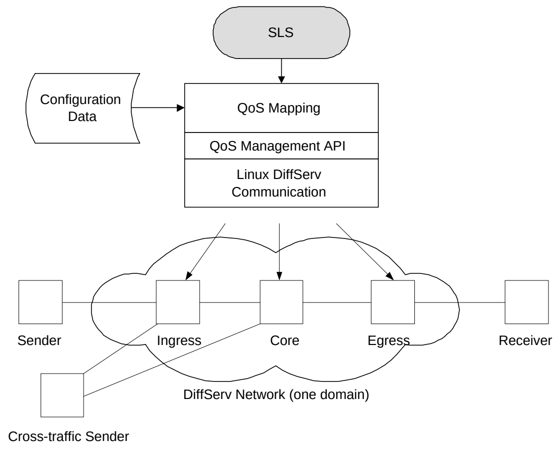

Fig. 5: Implementation

Our implementation of the mapping schema, as is shown in Figure 5, is on top of a QoS Management API [API] built in close collaboration with the University of Bern in Switzerland. The API is an object oriented Quality of Service management interface to Linux-based DiffServ routers [DS-IMPLEM], that is independent of the router hard- and software as well as from the access method to the router for management purposes. It provides classes for each type of hard- or software elements (i.e. routers, interface, classifiers, schedulers, and traffic conditioners) that is used and managed within the network. Derived classes from the base classes of the API have been developed to manage the specifics of the Linux-based routers, where as the commonalties are kept in the base class. This API is programmed in C++, so is our system.

A customer accesses the mapping system over a web server ordering a IP transport service via http. The Web interface, stores the SLS´s in an LDAP directory, where customer data and pre-defined service templates are stored as well. We have developed a text-based protocol to communicate between the web server and the mapping system. It allows the former to inform of new requests to be mapped, a negotiation between both systems in certain scenarios (where the customer takes part), and finally the mapping system to give the result of the process (accept/reject the service) to the web server, which informs the customer about the decision.

Our small testbed is based on Linux DiffServ routers. It consists of three routers: an ingress router, an interior router, and an egress router. In addition we use a sender/receiver pair and a background-traffic source. The ingress, interior and egress router form a DiffServ domain. The traffic from the sender to the receiver was routed through the chain of three routers while two background traffic flows were sent to the receiver, each entering the DiffServ domain at a different router. So small scale service setups can be performed, which however result in the expected behaviour.

5 Related work

The IETF DiffServ working group has standardized the architecture of the DiffServ framework (including a router model) and issues related to PHBs and PDBs, that is: the operational aspects. But the managing aspect has not been completely standardized. Our work focuses on this topic. Some managing approaches have been designed, we will outline some of them and compare them with our schema.

One of them is [CSM] which introduced an architecture to provide Services with QoS. In it, the central block was responsible of many functions, one of them was the mapping from SLS's values onto the equipment configuration. It was considered the most difficult part of the architecture and was still undeveloped. This is exactly the issue our work deals with. In [CSM] was outlined the difficulty that a Provider network can consist of devices from different vendors, with different configuration interfaces. We have overcome it with the design of the API, which provides an unique interface for all DiffServ routers.

Another approach is presented in [FLOWS], where the main point are end-to-end flows, which are used for network topology managing, bottleneck detection and SLA monitoring and reporting. This topic is addressed by our *admission control* block. So, our schema is compatible with those which take into account end-to-end flows in the way it is done in [FLOWS].

Another important approach about QoS management is the TEQUILA project [TEQ]. The Service Level Specifications submitted to the IETF from this project is our starting point, since we assume an SLS expressed with the parameters proposed in [TEQ]. Regarding the SLS specification, we also consider the possibility of having it expressed in terms of *pre-defined* SLS, the proposal of using *pre-defined* SLS is the main difference between the Tequila approach and the Aquila approach [AQUI]. Regarding the Tequila Functional Architecture, our Mapping schema covers partially several blocks. The first step of our mapping (from the SLS to the PDB) is be one of the necessary functions in the “Traffic Forecast” block (included in their SLS Management super-block), which generates a traffic estimation and is considered the “glue” between the SLS-Customer oriented and the rest of the architecture, like the first step of our schema. Our second mapping step (from PDB to configuration) would be used in the “Traffic Engineering” super-block, in the “Network Dimensioning” and “Dynamic Resource Management” blocks, which are responsible for mapping the traffic onto the physical network resources in long-term and short-term respectively. Since the blocks in our schema are very specific and well-defined, we can consider them as independent blocks as in the comparison with the Tequila Architecture.

6 Conclusions

We propose a two step mapping scheme for mapping Service Level Specifications (SLS) into DiffServ network configurations. The mapping process is supported and configured via a policy service. The functional block of QoS mapping is one component in a overall QoS management architecture in the Internet. However, many other parts are needed in order to get a working system. E.g., an initial configuration of the service classes is independent of the system we have so far.

Centralized management potentially has scalability problems. In our case, the parameter to observe is the number of service requests per second (processing capacity) and the total amount of currently active services (storage capacity and admissiosn control complexity). Requests per second are unknown, since no accepted traffic model exist so far. Anyway, it heavily depends on how aggregated the requests are. For instance, aggregating requests for VoIP seems to be requiered. Instead of making a request of 64 kbps per new call, make a single request of the expected traffic (which could be upgraded/downgraded later). In addition, the mapping system can be configured to reject any request with a throughput lower than X Mbps, avoiding the scalability problems and forcing users to aggregate flows.

In the future, we will work on the policy definitions and its implementation to support the mapping process. So far we have shown the mapping process only, however we have not worked on monitoring the service to find out whether we really guarantee the service in order to respond to complaining customers.

On big issues is centered around accounting, charging, and billing of services. We believe that our system can be easily used for this purposes, because of its modular nature. E.g., figuring out the price of the service may depend on the available amount of resources. We have all the information about the resources reserved etc, available. What is missing are the utilization records for each customer, which may influence the accounting. However, this is very expensive and therefore potentially an easier pricing scheme may work better.

References

[API] G. Stattenberger, T. Braun and M. Brunner: "A Platform-Independent API for Quality of Service Management", IEEE Workshop on High Performance Switching and Routing, Dallas, USA, 2001

[AQUI] S.Salsano et al.: "Definition and usage of SLSs in the AQUILA consortium", draft-salsano-aquila-sls-00, work in progress, November 2000

[AR-PDB] N. Seddigh, B. Nandy, J. Heinanen: " An Assured Rate Per-Domain Behaviour for Differenciated Services", draft-ietf-diffserv-pdb-ar-00, work in progress, February 2001

[DS-ARCH] S. Blake et al.: "An Architecture for Differenciated Services", RFC 2475, December 1998

[CADENUS] IST project "Creation and Deployment of End-User Services in Primium IP Networks (CADENUS)", http://www.cadenus.org

[CSM] R. Sprenkels, A. Pras, B. van Beijnum, L. de Goede: "A customer Service Management Architecture for the Internet", Proceedings of the 11th IFIP/IEEE DSOM 2000

[DS-FIELD] K. Nichols, S. Blake, F. Baker, D. Black: "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, December 1998

[FLOWS] J. Kim, J. Won-ki, S. Ryu, T. Choi: "Constructing End-to-End Traffic Flows for Managing Differentiated Services Networks", Proceedings of the 11th IFIP/IEEE DSOM 2000

[DS-IMPLEM] T. Braun et al.: "A Linux Implementation of a Differentiated Services Router", Networks and Services for Information Society (INTERWORKING'2000), Bergen, Norway, 2000

[ONE2ANY] M. Brunner, A.Banchs, S.Tartarelli: "An one-to-any Assured Rate Per-Domain Behaviour for Differenciated Services", draft-brunner-diffserv-pdb-one2any-ar-00, work in progress, 2001

[PDB-DEF] K. Nichols, B. Carpenter: "Definition of Differentiated Services Per Domain Behaviours and Rules for their Specification", RFC 3086, April 2001

[DS-MODEL] Y.Bernet, S.Blake, D.Grossman, A. Smith: "An Informal Management Model for DiffServ Routers", draft-ietf-diffserv-model-06, work in progress, February 2001

[TEQ] P. Trimintzios et al.: "Architectural Framework for Providing QoS in IP Differentiated Services Networks", 7th IFIP/IEEE International Symposium on Integrated Network Management

[VW] Van Jacobson, K. Nichols, K. Poduri: "The 'Virtual Wire' Per-Domain Behaviour", www.packetdesign.com/docs/vw_pdb_0.pdf