SERVICE MANAGEMENT FOR DIFFERENTIATED SERVICES NETWORKS

Elionildo Menezes[1] *Djamel Sadok*[2] *Judith Kelner*[3] *Paulo Pereira*[4] *Paulo Pinto*[5]
[1,2,3] Universidade Federal de Pernambuco. Centro de Informática. Caixa Postal 7851.
CEP 50732–970. Recife – PE – Brazil, e-mail: {esm, jamel, jk}@di.ufpe.br
[4]Instituto Superior Técnico, Universidade Técnica de Lisboa.
INESC, Rua Alves Redol, 9. 1000-029 Lisboa, Portugal, e-mail: prbp@inesc.pt
[5]Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa.
UNINOVA, Quinta da Torre. 2825 Monte de Caparica, Portugal, e-mail: pfp@uninova.pt

ABSTRACT

This work presents and evaluates a functional architecture and a framework for mapping service management policies and constraints into differentiated services (DiffServ) mechanisms. Several simulation scenarios described through the use of service policies are introduced and analyzed. It is shown that the use of service level management allows for efficient dynamic traffic engineering of DiffServ backbones. It is also shown that the use of active policies can assure better service to users by better enforcing service level agreements.

1 INTRODUCTION

The increasing growth of the Internet and corporate networks raises new concerns related to mechanisms such as routing, resource reservation, traffic engineering and management [7]. Furthermore, with the introduction of new Internet and corporate services such as e-business, VoIP (Voice over IP) and multimedia applications, the best-effort packet forwarding paradigm is not capable of attending the QoS (Quality of Service) requirements of such a wide range of services. This paradigm deprives the network core of any form of intelligent traffic forwarding [22]. Its design simplicity was surely one of the contributions for the success of the TCP/IP backbone technology.

This work presents both a model and an a functional architecture for the implementation and management of Internet services according to policies defined in the form of service contracts. Section two presents recent work on management frameworks for QoS based management. Section three discusses both the model and a new architecture for contract based service management. Finally, section four presents several scenarios for contract based policy management of DiffServ domains whose simulation results are discussed in the fifth section of this paper.

2 STATE OF THE ART

High-level service management may be achieved through the definition and mapping of corporate policies onto network resources. Similarly, high-level user contracts may be supported at the network level using mechanisms such as admission control, packet prioritization, traffic engineering, QoS routing and resource reservation. The IETF DiffServ working group has identified the need for service policies in order to allow border domain routers to correctly classify packets by consulting such policies. In this paper, a complete architecture shows how the integration between policy contracts and the DiffServ architecture is achieved.

2.1 Service Level Management

SLM (Service Level Management) is a new paradigm for the integrated management of network resources, systems, applications and services according to policies defined with service contracts [19]. Such policies define rules and restrictions that control resource allocation to the supported services as shown

in figure 1 [23] and are expressed in the form of contracts or SLAs (Service Level Agreements). SLAs may determine the quality of the offered services in terms of availability, security information, response time, delay, throughput, etc. SLAs may be static or dynamic [29]. Static SLAs suffer little change (limited to periodic review of the contracts between users and service providers). Dynamic SLAs are designed to continuously and automatically adapt to network changes in order to maintain the service according to the contract.

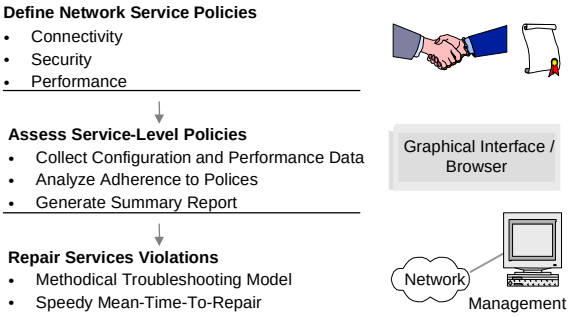

Figure 1. Processes Involved in Service Management.

SLM capable products include InfoVista™ [13], Netsys [23], HP IT Service Management [25] and Spectrum [6], although many of these implement limited SLM functionality.

2.2 Differentiated Services

Introducing QoS into what traditionally has been a best-effort packet network is not straightforward. Many IETF working groups have been setup in the last decade to address this issue. Among the adopted solutions, are [27]: the Integrated Services Model (IntServ) [3] which uses RSVP (Resource Reservation Protocol) [4] to make individual per flow reservations; the Differentiated Services Model (DiffServ) [2], a prioritization scheme for aggregations of flows; the MPLS (Multiprotocol Label Switching) [21] which uses tag based forwarding paths through a network; the Traffic Engineering techniques [1] is also used to plan and configure network resources; and finally QoS based routing is used to compute routes that attend given QoS restrictions [9]. The QBone project [14] is currently pioneering the use of the DiffServ packet prioritization behaviour at core routers (known as Per Hop Behavior – PHB). A clear advantage of DiffServ over the IntServ approach is scalability, specifically when considering large backbones.

In the DiffServ architecture, border routers are responsible for aggregate classification of packets and their policing according to static or dynamic contracts or SLAs, whereas core routers merely forward these marked packets according the DSCP (DiffServ Code Point) information within a packet, see figure 2 [17].

The combination of PHB based forwarding at the core and border packet classification and conditioning, allows a DiffServ domain to support various services. Note that the actual definition of services is outside the scope of the IETF and that so far two PHBs have been defined, namely, EF (Expedited Forwarding) [16] and AF (Assured Forwarding) [11]. The EF PHB offers rigid QoS guaranties and may be used to implement services that require bounded delay and guaranteed bandwidth. Examples of these include circuit emulation, voice and video services. The AF PHB on the other hand, offers limited QoS guarantees and may be used for applications such as Web access. The best-effort packet forwarding (BE) is maintained for low priority and background traffic.

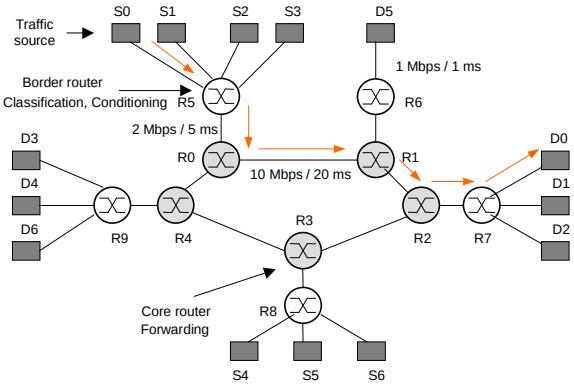

Figure 2. Example of a DiffServ Network.

2.3 Network Management Policies

Service policies are an important instrument in the correct implementation of QoS support, since SLAs are described in high-level abstract notations that are human readable; and are continuously consulted by network elements in order to ensure that decisions at this level comply with corporate policies or user contracts.

Service policies are used in order to optimize, monitor and control the use of network resources and services. Policies may be based on parameters describing traffic origin and destination such as IP addresses, port numbers, network and subnet information.

The IETF has been working on a common SLA specification syntax and semantics. The adopted notation defines a policy as consisting of a set of rules describing actions which should be undertaken under given situations [15]. Furthermore, more complex policies may be built combining simpler ones in order to ease their specification [24]. Many studies are underway to define policy repositories and how these may be accessed [18]. Solutions vary from the

adoption of existing systems and protocols including SNMP (Simple Network Management Protocol) [8], LDAP (Lightweight Directory Access Protocol) [12] and HTTP (HyperText Transfer Protocol) [10] to the creation of new ones such as in the case of the COPS (Common Open Police Service) [5]. It is likely that a combination of access techniques may be used.

3 A SERVICE MANAGEMENT MODEL

3.1 Motivation

The DiffServ IETF group has, according to its agenda, defined QoS mechanisms that heavily rely on the presence of policies to apply QoS related packet classification. On the other hand, work on SLM merely defines how services may be managed, but lacks definition of the actual underlying mechanisms. The mapping of DiffServ policies into its actual mechanisms remains a challenge that is addressed in this section.

3.2 Proposed Model – Management Planes

The proposed model, shown in figure 3, structures service management into four distinct planes with increasing service abstraction levels. The fourth plane, the management plane, is orthogonal to the other three planes. These are described next:

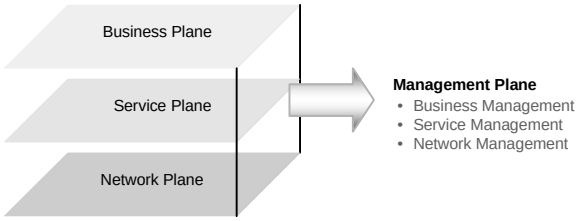

Figure 3. A 4 Plane Model for Service Management.

• *Business Plane*: defines and evaluates services using human readable and few technical information in an effort to approximate the contract specification and management to the user. Users at the business level would be able to monitor their business services and act upon them without the need to deal with their implementation details. Business contracts may be drawn with items related to the time of offering the service, operation and maintenance costs, how fast services may recover, structure of support teams, relative service priority, penalties for contract violation and termination. Business contracts have judicial value and establish an agreement between clients and their service provider. The latter is represented by the business manager. In this work, such contracts are referred to as CLAs (Customer Level Agreements).

• *Service Plane*: defines services using a more technical profile specifying QoS parameters such as delay, jitter, bandwidth, forwarding priority, traffic conditioning policy, redundancy schemes used to guaranty service availability, etc. Although service plane contracts are defined on an individual basis, they nevertheless involve the specification of requirements to be met by each of the DiffServ traffic aggregation classes. Here, a service manager is the entity responsible for the definition, monitoring and possible service policy modification in order to attend service contracts also known as SLAs (Service Level Agreements). A SLA may be seen as a set of CLAs of various clients as shown in figure 4.

• *Network Plane*: the contracts that determine the technology used by the infrastructure belong to this plane. These contracts support the offered CLAs specified in the SLAs. In this plane, a contract is referred to as a TCA (Traffic Conditioning Agreement) in conformity with DiffServ terminology. TCAs, for example, can be used to establish parameters for router queues and routing algorithms on the basis of services. The network manager is the entity responsible for management duties at this plane.

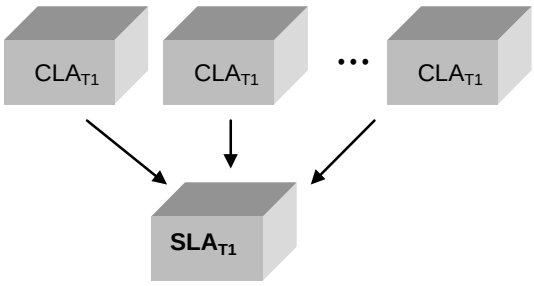

Figure 4. SLA aggregating CLAs of the same type.

• *Management Plane*: defines management activities to co-ordinate all three previous planes. It is important to emphasize the importance of the interaction between these planes in order to assure that contract changes are reflected at all levels and that business, service and network planes cooperate through the common management plane.

Although all planes include support for the fault, configuration, performance, accounting and security OSI management functional areas, the structure of management data may differ between the planes.

3.3 Functional Architecture

Figure 5 shows the architecture associated to the model described in the previous section. The depicted example is that of an Internet Service Provider (ISP) offering three DiffServ based commercial services, namely, Enterprise, Standard and Light. These are described next:

• *ENTERPRISE*: this service has the best performance when compared to the other two. It offers rigid bounded delay guaranties. Hence, it is ideal for delay sensitive applications such as videoconferencing. The network offers higher priority to ENTERPRISE traffic. It is implemented using the DiffServ EF PHB and traffic conditioning is achieved by discarding packets that are out of the negotiated profile. It is expected that EF based services, such as the ENTERPRISE service take up a small percentage of the network capacity, but be priced much higher than current best effort service.

• *STANDARD*: ideal for clients looking for a service that performs better than LIGHT, but cannot, or would not, pay for the limited and more expensive ENTERPRISE service. This service offers minimum QoS guaranties, whereby the network seems lightly loaded. It is suitable for applications that are less sensitive to delay and bandwidth requirements such as Web navigation, and e-mail. This service is implemented using the DiffServ AF PHB, where traffic conditioning is achieved by marking out of profile packets according to the negotiated service contract.

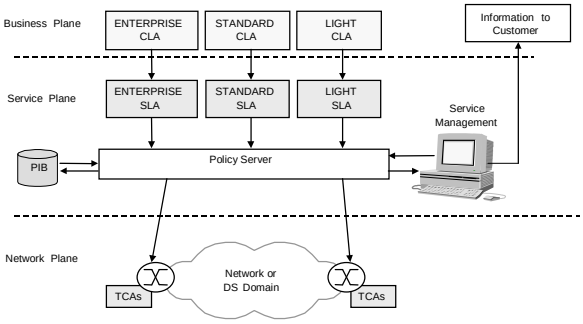

Figure 5. A Functional Architecture for Service Management

• *LIGHT*: characterized by its occupation of whatever network resources are left. This is, in other words, a best effort service with no real guaranties. It is expected that this type of service is mostly used for background traffic applications such as backups and file transfers.

In the proposed architecture, CLAs associated to each of the three services (ENTERPRISE, STANDARD or LIGHT) are aggregated to form a corresponding SLA. These are then mapped into policies, which are stored into policy repositories, also known as Policy Information Bases (PIBs). PIBs are accessed by policy servers responsible for propagating the stored policies over network elements such as border routers, and configuring their TCAs in order to enable the border routers to perform traffic management within its DiffServ Domain. Periodically, and in the case of important events, the policy server sends management reports to the service management application. This application builds this information in the form of user reports that it sends to them with information about whether their service contracts are being honoured.

4 SIMULATIONS

In this section, a number of scenarios showing the use and mapping of service management concepts over DiffServ domains were simulated. The objectives of such simulations include: validating the proposed architecture, showing its benefits, proving its viability, showing the important role of policy servers in controlling network performance and QoS. The simulations were written using version two of the Berkeley network simulator [26] in both Tcl and C++ code. The network was configured to support Differentiated Services through the inclusion of DiffServ components for traffic conditioning, metering, shaping and packet dropping. Other functionality added to the simulator include support for EF and AF PHBs besides BE.

4.1 The Simulated Network

The network topology used in the first simulations was presented in figure 2 of section 2.2.

4.2 Simulation Parameters

In order to approximate the simulated scenarios to the real ones, a number of traffic sources were defined to generate different traffic patterns. Table 1 shows a list of the sources used in the simulated network including CBR (Constant Bit Rate), alternate traffic fonts or On/Off, remote access using Telnet and file transfer using FTP.

Source	Traffic	PHB	Rate (Mbps)	Destina -tion
S0	CBR	EF	0.1	D0
S1	Telnet	AF11	0.8	D1
S2	FTP	BE	-	D2
S3	FTP	BE	-	D3
S4	On/Off	AF11	1.0	D4
S5	CBR	EF	1.0	D5
S6	On/Off	AF11	1.0	D6

Table 1. Parameters describing Traffic Sources used in the simulation.

Packet sizes were configured to 576 bytes and 1 KB for CBR and other traffic sources respectively. Furthermore, all simulations are allowed to run for a minimum of 60 seconds. A single domain was simulated in this work in order to avoid dealing with

inter-domain contracts, which was considered outside the scope of this work. The following network resource allocations were made for the three DiffServ traffic classes: 5% for the EF PHB, 40% for the AF PHB and the remaining 55% were allocated to the BE (Best-Effort) PHB. The EF queues are droptail, the AF queues are RIO, and the BE queues are RED.
In the first two case studies, each traffic shaper was configured with the following parameters: peak rate of 500 Kbps, burst size of 16 KB and a queue length of 3. Whereas in the case of the third case study, the peak rate was altered to 1 Mbps in order to use all the capacity of the links between sources S4 and S6 and the border router R8, which generated 1 Mbps each as shown in figure 6. The schedulers used in all simulations were configured with the parameters defined in [20] that are presented in table 2.

	Scheduler Parameters
ef-queue-length	40
af-queue-length	62
be-queue-length	150
af-queue-rio-params	0.002 30 60 50 15 30 10
be-queue-red-params	0.002 50 145 20
ef_queue_weight	1
af_queue_weight	8
be_queue_weight	11
aggregate-bytes-thresh	4000

Table 2. Scheduler Configuration Parameters.

5 CASE STUDIES AND RESULTS

The simulated scenarios show the use of policy servers for service management and validate the architecture proposed in section 3.3. The terminology used to build the policy rules is based on definitions from [18].

5.1 First Case Study

A CLA representing a client contracting the ENTERPRISE service during periods in the morning and in the afternoon. It is assumed that the EF traffic from this service is associated to a CBR source with a 100 Kbps transmission rate, according to its SLA. In this case, it is desirable to configure the policy server to make ENTERPRISE bandwidth available to other services and traffic classes at night, for example. To achieve this, the policy used contains the following rules:

Rule 1: Offer High priority for traffic from source S0, between 7h and 19h

```
If  (  (source == S0) && (timeOfDay == 0007-0019) )
then
     priority = High
endif
```

Rule 2: Lower priority for traffic from source S0 during morning hours between 19h and 7h

```
If  (  (source = = S0) && (timeOfDay = = 0019-0007) )
then
     priority = 0
endif
```

In order to simulate this scenario, the following parameters have been considered: source S0 is initially inactive during the first 30 seconds and stops at 60 seconds of simulation time. The graphs from figure 6, clearly show that when using the established policies, source S3 takes advantage of the available idle bandwidth during this period also showing how FTP traffic adjusts to the available bandwidth. Since this is a best effort traffic source, only throughput has been measured. Other performance data such as delay and jitter are also considered in the next two scenarios. Similar considerations are made about source S2. Furthermore, there was no change in S1's throughput, delay and jitter since its traffic follows the imposed SLA rules.
In other words, this scenario depicts the importance of actions from the policy server in the engineering and control of backbone traffic to ensure better use of network bandwidth and other resources.

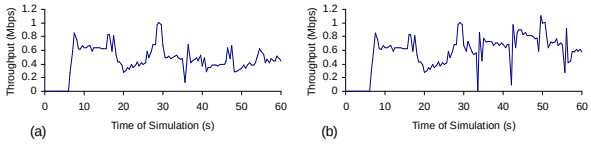

Figure 6. S3 traffic throughput. (a) Without the use of available bandwidth between 30s and 60s. (b) Using available bandwidth between 30s and 60s.

5.2 Second Case Study

Next, a scenario where a user contracts the ENTERPRISE service for an application sensitive to both delay and jitter is presented. The simulation assumes that at a given time, the default route used for forwarding packets from this application suffers a problem and becomes unavailable. A new route is then established in spite of presenting higher delay and jitter values than those required by this application. The service policy for this CLA is described by the next rule.

Rule 1: Monitor and notify service management about contract violations.

```
If  (  (source = = S0) && (LinkDown(R0,R1) )
then
   if ( ( delay_now > delay_S0 ) || ( jitter_now > jitter_S0 ) )
   then
      LevelService = LOW
   endif
endif
```

In the simulation, source S0 generates traffic at a rate of 100 Kbps using the ENTERPRISE service. Figure 7, shows the actual throughput, delay and jitter associated to traffic flow. Although, both throughput and jitter have been maintained within the bounds of the service contract, the maximum delay has suffered a great deal. In the case of the application requiring delays inferior to 0.1 second, the policy server must alert both the policy manager to take action and the user to be aware of this contract violation.

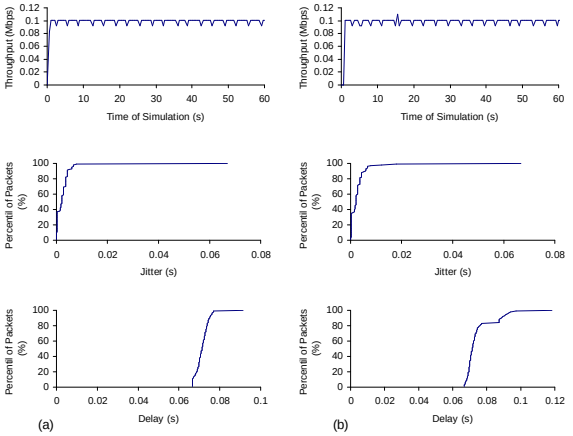

Figure 7. Throughput, delay and jitter for S0. (a) Without link failure between R0 and R1. (b) With link failure between R0 and R1.

5.3 Third Case Study

This case study considers the scenario of a personnel department making salary payments for a 10 days duration starting in the last five days of each month. The payroll processing requires access to corporate databases distributed among two sites. Since the payroll activity for this company is considered as a very important service, a special service contract has been established between this department and its network provider which may be another company sector or a foreign entity. Note that this does not in anyway affect the architecture or the performance studies in this scenario. The contract stipulates that access during this period must be exclusive in terms of access to network resources. The STANDARD service has been selected for use in this scenario since the payroll application presents relatively flexible QoS parameters when consulting the databases. Such operations are generally sporadic and generate high bandwidth variable rate transmissions interleaved with periods of little activity.
The policies associated with this scenario are shown next:

Rule 1: Disable access to source S5 during this period.

```
If  (  (source = = S5) &&
       (  ( dayOfMonth in last5days ) || ( dayOfMonth in [1–5] ) ) )
    then
        priority = 0
endif
```

Rule 2: Offer exclusive access with maximum priority to the department of personnel during the period of payroll.

```
If  (  (  (source = = S4) || (source == S6) ) &&
       (  ( dayOfMonth in last5days ) || ( dayOfMonth in [1–5] ) ) )
    then
        priority = High
endif
```

Rule 3: Disable traffic generated by source 6 after payroll has completed.

```
If  (  (source = = S6) &&
       ! ( ( dayOfMonth in last5days ) || ( dayOfMonth in [1–5] ) ) )
    then
        priority = 0
endif
```

Rule 4: Reset traffic priorities for sources 4 and 5 to normal once payroll processing has completed.

```
If  (  (  (source = = S4) || (source = = S5)  ) &&
       ! ( ( dayOfMonth in last5days ) || ( dayOfMonth in [1–5] ) ) )
    then
        priority = Normal
endif
```

In the simulation of this scenario, two On/Off traffic sources have been considered in order to attend the high level requirements specified above. The time frame for elaborating the payroll is simulated as the time interval between 20 and 35 seconds. During this period, only sources S4 and S6 generate traffic at a 1 Mbps rate each. Source S5 is disabled and is only reactivated (allowed to transmit) after this period. Figure 8 shows the results of this simulation.

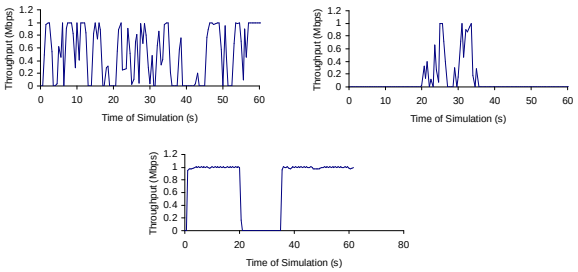

Figure 8. Throughput for sources S4, S5 and S6. Above, throughput graphs for S4 and S6, respectively. Below, the throughput for source S5.

It is shown in this scenario that the policy server gives priority to both traffic flowing from both sources S4 and S6 while removing traffic from S5 during the payroll processing period. This procedure does not require the need for manual intervention. Outside the payroll time frame, default priority values are restored to all traffic sources.

5.4 Fourth Case Study

This case study uses the network topology illustrated in figure 9 to evaluate different user admission policies and different traffic classification policies. The network has a core of 15 nodes divided amongst 3 regions. Each node has one point of presence (PoP) attached, as shown in figure 10, where clients arrive to have network access, producing traffic from different applications. In this case study the bandwidth allocations were changed so that EF has 50% of the bandwidth, AF has 40%, and BE has only 10% plus whatever remains unused by the other traffic classes.

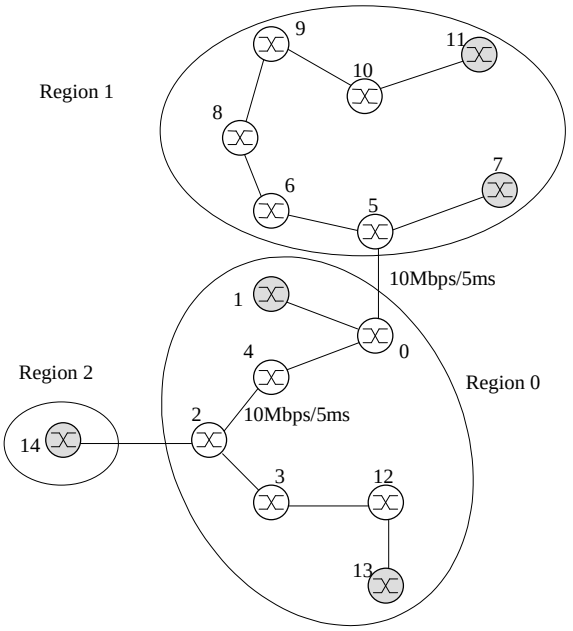

Figure 9. Network topology with 15 core nodes divided amongst 3 regions.

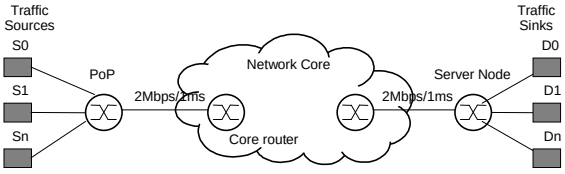

Figure 10. Model of a PoP connected to a core node.

At the Business Plane level, 4 different user admission policies where used:

a. Users are accepted until a fixed maximum number of users (30) is reached for each PoP.
b. Users are accepted until a fixed maximum number of users is reached for each class, (Enterprise ($1/3^{rd}$), Standard ($1/3^{rd}$)), after which the users get the lower available service class, possibly Light.
c. Similar to **a**, but users are accepted until a maximum number of users depending on the current PoP load is reached.
d. Combination of **b** and **c**, all maximums depend on current PoP load.

At the Service Plane level, the traffic is mapped into one of the available DiffServ PHBs. Four different policies may be used to restrict this mapping according to the user application involved, as shown in table 3:

Policy	telnet/ TCP	CBR/ UDP	OnOff/ UDP	HTTP/ TCP	ftp/ TCP
1	any	any	any	any	any
2	any	any	any	AF BE	BE
3	EF AF	EF AF	EF AF	AF BE	BE
4	EF AF	EF AF	EF AF	BE	BE

Table 3. Policies to map application traffic to PHBs.

Table 4 allows us to analyse the effect of using policies 1, 2, 3 and 4. If too much traffic is allowed to use the EF PHB, as is the case of policy 1, the router schedulers will be overloaded, causing an unacceptable drop ratio for the EF PHB. The other PHBs are even more affected since they are also overloaded and have lower scheduling priorities. Policy 2 maps the traffic from bandwidth intensive applications (http and ftp) to the lower priority PHBs. This reduces greatly the traffic load offered to the EF and AF PHBs, resulting in much lower packet drop ratios, delays and jitter for all the traffic classes. Policy 3 is similar to policy 2, but maps the delay sensitive application traffic onto the EF or AF PHBs to offer them lower delays and jitter. This causes the network load on the EF and AF PHBs to increase slightly, resulting in a slight increase in the AF drop ratio, delay and jitter, that is not noted on the EF PHB. As a result, the network load on the BE traffic slightly decreases, causing less packet drops, but the delays increase slightly since there is more higher priority traffic. Finally, policy 4 also forces http traffic to the BE PHB, allowing only bandwidth limited protocols into the EF and AF PHBs, reducing the load on all classes.

Policy	1-a	2-a	3-a	4-a
Drop R. EF	0.00318	0.0	0.0	0.0
Drop R. AF	0.00369	0.00031	0.00046	0.0
Drop R. BE	0.00644	0.00570	0.00462	0.00325
Drop Ratio	0.00381	0.00250	0.00199	0.00257
Delay EF	0.15220	0.04304	0.04233	0.04117
Delay AF	0.30501	0.18095	0.19571	0.04300
Delay BE	1.12240	0.49381	0.51509	0.30225
Jitter EF	0.02325	0.00385	0.00418	0.00327
Jitter AF	0.04334	0.02891	0.03300	0.00369
Jitter BE	0.13060	0.08388	0.07323	0.03041

Table 4. Effect of Policies 1, 2, 3 and 4.

As was noted before, excessive traffic in the EF and AF classes causes service degradation. Policy **b** is the

simplest policy to overcome this problem, by counting users in each class, and placing users that exceed their class quota in the next lower available class. These users get worse service than what they have desired, but globally, all users get better service. A limit of 1/3rd of the total (10) is allowed in class Enterprise and 1/3rd (10) in class Standard. In addition to this problem, several users in different nodes can be sharing common links or accessing the same destination application, causing local peaks in the network utilisation that can also cause problems.

Policy **c** was based in the work of [28] and monitors link load to dynamically adjust the maximum number of allowed clients in each PoP. This is an example of an active policy, as it continuously adapts to the network state, allowing to implement a dynamic SLA. A TCA at the Network plane specifies the maximum packet drop ratio for each PHB. The disadvantage of this policy is denial of service to new users during peak traffic periods. This policy works by reducing the maximum number of clients by 10 if there are more than 50 packet drops in a 2 seconds period. Current users remain active, but new users are refused access.

Finally, policy **d** is also an active policy that additionally adjusts the limits of policy **b** according to local load in each traffic class: in each 2 seconds period, the EF and AF packet drops are analysed. If there are any EF, or 5 AF packet drops, the maximum number of clients is set to the current number of clients minus 3, keeping a minimum of 5 clients. The maximum number of clients is increased back to allow 3 new clients, if there are no packet drops, growing until its normal value.

Policy	1-b	1-c	1-d	3-d
Drop R. EF	0.00274	0.00301	0.00189	0.0
Drop R. AF	0.00330	0.00349	0.00113	0.00019
Drop R. BE	0.01387	0.00689	0.02193	0.00629
Drop Ratio	0.00430	0.00366	0.00546	0.00276
Delay EF	0.14750	0.15412	0.13770	0.04173
Delay AF	0.30466	0.30815	0.21972	0.15310
Delay BE	1.46016	1.15868	1.09771	0.47777
Jitter EF	0.02211	0.02319	0.01673	0.00403
Jitter AF	0.04248	0.04400	0.02882	0.02627
Jitter BE	0.17078	0.13534	0.16900	0.08260

Table 5. Effect of policies b, c and d.

Table 6 shows the throughputs the different applications are getting. It can be noted that the telnet, CBR, and OnOff applications have approximately the same throughput for the different policies with a slight decrease for CBR with policy 1 because of the higher packet drop ratio. The Http and ftp applications get a highly variable throughput as it depends on the network load.

Tables 7 and 8 show the average delay and jitter the different applications are getting. Here the conclusions are similar, as telnet, CBR and OnOff applications get better service with policy 2 and 3, and best service with policy 4. The use of policy **d** produces worse results, as some users get a lower service class.

Throughput	1-a	2-a	3-a	4-a
telnet	7144	7101	7181	7090
CBR	54841	56543	57192	57329
OnOff	24962	25121	24822	24894
HTTP	117458	194637	180552	92066
ftp	186543	72040	78129	135425

Table 6. Throughput in bps for different policies.

Delay	1-a	2-a	3-a	3-d	4-a
telnet	0.4601	0.2418	0.0902	0.1444	0.0434
CBR	0.4866	0.1954	0.1065	0.1250	0.0430
OnOff	0.4784	0.2084	0.0903	0.1322	0.0399
HTTP	0.3105	0.2528	0.2799	0.2665	0.3188
ftp	0.2957	0.4696	0.4968	0.4261	0.2934

Table 7. Average delay in seconds for different policies.

Jitter	1-a	2-a	3-a	3-d	4-a
telnet	0.1895	0.1167	0.0394	0.0710	0.0046
CBR	0.0591	0.0425	0.0261	0.0283	0.0034
OnOff	0.0799	0.0542	0.0256	0.0392	0.0035
HTTP	0.0458	0.0340	0.0376	0.0395	0.0388
ftp	0.0289	0.0635	0.0655	0.0497	0.0259

Table 8. Average Jitter in seconds for different policies.

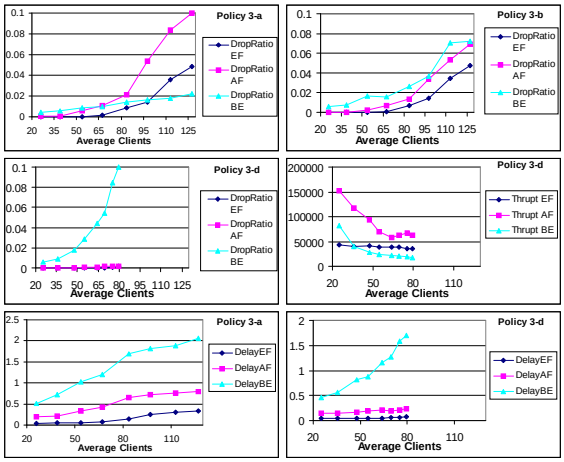

Figure 11. Effect of changing the number of users.

Figure 11 shows the effect of changing the number of users with different policies. The use of policy **a** results in an excessive packet drop ratio for the EF and AF PHBs, that even exceeds the BE drop ratio for large numbers of clients. The use of policy **b** has the main effect of decreasing the AF packet drop ratio,

keeping it below the BE packet drop ratio that is increased as some clients are downgraded to this PHB. The use of policy **d** keeps the AF and EF packet drop ratios at very low values, increasing significantly the BE drop ratio as more clients are allowed into the system.
Policy **d** also significantly reduces the EF and AF packet delays as can also be seen from the last two graphics of figure 11. In addition, this policy has the effect of preventing system degradation for large number of clients, as they are downgraded to lower traffic classes. Figure 11 also presents the throughput graphic for different number of clients, showing that the EF traffic gets almost the same throughput for the allowed range of clients, while the BE, and in less extent, the AF traffic decrease significantly as the number of clients increase.
This case study shows that active policies (policy **d**) adapt to network state, outperforming non-active policies, and offering users the best possible service.

5.5 Fifth Case Study

This case study uses the network previously illustrated in figure 9 to evaluate link failure recovery solutions to maximise service availability to users. The grey routers have the capacity to establish backup links to another grey router to provide alternate routes in case of main link failure. The topology was designed to minimise the total number of links and grey routers. To accomplish this, the network has a path within each region connecting all the routers. In addition, two grey routers are placed in each region in nodes with only one link, the nodes most likely to become isolated by link failure. Finally, each region is connected to another region by a link between nodes with two links, the nodes less likely to become isolated by link failure.
The failure recovery algorithm is a distributed hierarchical algorithm as suggested in the work of [28]. The recovery algorithm is composed of an algorithm running within each region and a top-level algorithm managing backup links between regions. These algorithms work at the network plane deciding automatically when and which backup links should be established or terminated. Both algorithms start in the nodes or regions with a failure, and search for the best backup link to establish. As the backup links have an inferior transmission rate, the policy used to map application traffic to PHBs is changed during the failure to assure the best service to the most important service classes. The best one is policy 4, that can be further modified to discard ftp traffic, if necessary.
Table 9 shows the recovery algorithm success rate. The algorithms local to each region solve about half the failures. The global algorithm solves an additional 19% of the failures. 15% of the failures are multiple link failures that cannot be solved with the chosen topology, causing at least one node to remain disconnected from the rest of the network during the failure. Finally, 17% of the failures correspond to cases where there was already a backup link established and no action was necessary.

	Total	Percentage
Total Failures	506	100%
Solved by Intra-Region Algor.	248	49%
Solved by Inter-Region Algor.	96	19%
No Problem	86	17%
Not Solved	76	15%

Table 9. Failure recovery algorithm success rate.

From this case study, it is shown the advantage of using a distributed and hierarchical algorithm, since the local algorithms operating internally in each region solve about half of the problems, but the top-level algorithm is still needed to solve problems that involve different regions.

6 CONCLUSIONS

This work presented a four layered model for service management, based on DiffServ related policies. The mapping of policies between business, service and network layers was also discussed. A functional architecture was described next through an example with various services with different QoS requirements. The concepts of both the model and the associated architecture were validated through the simulation of some scenarios. This work has shown the importance of integrating service level management through the use of policy servers in order to support different levels of QoS requirements. Using a DiffServ domain, it was shown that traffic engineering may be made through the implementation and management of service policies that prioritize flows, optimize traffic and monitor user contracts. It was also shown that the use of active policies further helps maintaining the SLA, improving the QoS offered to users for wider network parameter variations.
The proposal presented in this work represents an innovation, as it integrates different views, still isolated, like SLM, DiffServ and network policies.
Other proposals, discussed next, have some additional limitations, because either there is no structuring of the management activity in different planes reflecting the network services abstraction levels, or because they do not include the elements discussed in section 2. In the first case, we refer the work of [30], where contracts are created with a low-level notation, based on DiffServ parameters (queue sizes, PHBs, etc.) not associated with commercial services, or the user perspective. In the second case, [31] proposes a

model with several layers, but does not integrate with DiffServ, nor shows how to implement management policies in a network. The main benefit of our proposal is integrating a service management hierarchy, and implementing it through simulation into a network scenario with differentiated services provided by DiffServ.

ACKNOWLEDGMENTS

The authors acknowledge the support of ICCTI/ CAPES - 99 - Procº 423/CAPES.

REFERENCES

[1] D. Awduche et al., "Requirements for Traffic Engineering over MPLS", June 1999, <draft-ietf-mpls-traffic-eng-01.txt>

[2] S. Blake et al., "An Architecture for Differentiated Services", RFC 2475, December 1998.

[3] R. Braden, D. Clark, & S. Shenker, "Integrated Services in the Internet Architecture: An Overview", RFC 1633, June 1994.

[4] R. Braden, "Resource ReSerVation Protocol (RSVP) - Version 1 Functional Specification", RFC 2005, September 1997.

[5] J. Boyle, "The COPS (Common Open Policy Service) Protocol", February 1999, <draft-ietf-rap-cops-06.txt>

[6] Lundy Lewis, "Spectrum Service Level Management - Definition, Offerings, and Strategy", March 1998, http://www.cabletron.com/white-papers/spectrum/slm.pdf

[7] Kenneth L. Calvert, M. B. Doar, & E. W. Zegura, "Modeling Internet Topology", IEEE Communications Magazine, June 1997.

[8] J. D. Case et al., "Simple Network Management Protocol (SNMP)", RFC 1157, May 1990.

[9] E. Crawley et al., "A Framework for QoS-Based Routing in the Internet", RFC 2386, August 1998.

[10] E. Fielding, "Hypertext Transfer Protocol - HTTP/1.1", RFC 2616, June 1999.

[11] J. Heinamen et al., "Assured Forwarding PHB Group", RFC 2597, February 1999.

[12] L. Howard "An Approach for Using LDAP as a Network Information Service", RFC 2307, March 1998.

[13] InfoVista(TM) - Service level Management Solution: Building a Service Management Environment, March 1998. http://www.3com.com/products/dsheets/400365a.html

[14] Internet2, QBone Inititative. http://www.internet2.edu/qos/qbone

[15] "Introduction to QoS Policies" - White Paper, July 1999. http://www.qosforum.com

[16] V. Jacobson & K. Nichols "An Expedited Forwarding PHB", RFC 2598, February 1999

[17] James F. Kurose, Keith W. Ross, "Computer Networking - A Top-Down Approach Featuring the Internet". http://www.seas.upenn.edu/~ross/book/Contents.htm

[18] H. Mahon, Y. Bernet, S. Herzog, "Requirements for a Policy Management System", October 1999 <draft-ietf-policy-req-01.txt>

[19] John McConnell, "Service Level Management - Leveraging Your Network Investments", July 1998. http://www.3com.com/technology/tech_net/white_papers/500654.html

[20] Sean Murphy, "Some notes on the use of my ns diffserv software", September 1999. http://www.teltec.dcu.ie/~murphys/ns-work/diffserv/index.html

[21] E. Rosen et al., "Multiprotocol Label Switching Architecture", April 1999, <draft-ietf-mpls-arch-05.txt>

[22] Saltzer et al., "End to End Arguments in System Design", ACM Transactions in Computer Systems, November 1984. http://www.reed.com/Papers/EndtoEnd.html

[23] Service Level Management with Netsys: A Model-Based Approach, 1998. http://www.cisco.com/warp/public/cc/cisco/mkt/enm/netsys/man4/tech/slnet_wp.htm

[24] J. Strassner et al., "Terminology for describing network policy and services", February 1999, <draft-ietf-policy-terms-01.txt>

[25] The HP IT Service management Reference Model - White Paper; March 1998. http://www.hp.com/pso/frames/services/whitepapers/wp-itsm-overview.html

[26] VINT Network Simulator (version 2). http://www-mash-cs.berkeley.edu/ns

[27] X. Xiao & L. M. Ni, "Internet QoS: A Big Picture", IEEE Network, March/April 1999.

[28] P. Pereira, P. Pinto, "Algorithms and Contracts for Network and Systems Management". IEEE Latin American Network Operations and Management Symposium, Rio de Janeiro, Brazil, 3-5 December 1999, pp.385-396. ISBN: 85-900382-3-8.

[29] Yoram Bernet et al., "A Framework for Differentiated Services", <draft-ietf-diffserv-framework-02.txt>, February 1999.

[30] R. J. Gibbens et al., "An Approach to Service Level Agreements for IP Networks with Differentiated Services", Article submitted to Royal Society, 2000. http://www.statslab.cam.ac.uk/~richard/research/papers/sla/

[31] René Wies, "Policies in Network and Systems Management – Formal Definition and Architecture"; Journal of Network and Systems Management, Plenum Publishing Corp., pp. 63-83, March 1994.