ADAPTATION OF TRAFFIC AGGREGATES FOR DEFECTIVE QOS ARCHITECTURES

HERMANN DE MEER*† AND SPYRIDON RETZEKAS*

† Faculty of Mathematics and Computer Science, University of Passau, Germany

** Department of Electrical and Electronic Engineering, University College London, UK*

demeer@fmi.uni-passau.de

Abstract: It is anticipated that congestion may still be an occasional matter of fact under the Differentiated Services Framework. Based upon this observation, we propose a Quality of Service architecture that uses segmented adaptation mechanisms on whole traffic aggregates, in order to allow for the co-operation between network operators in case of congestion. *Service curves* are introduced as a tool for defining Quality of Service levels internally in each Service Level Agreement. Edge routers are responsible for the smooth movement between the predefined service curves in case of congestion. Moreover, a domain administrative entity (Service Level Agreement Broker) is used in order to monitor the adaptation/recovery procedures and the processes associated with resource trading and billing. We examine the structure and operation of the Service Level Agreement Broker and some signalling issues related to the proposed architecture. Some simulations implemented on the Network Simulator 2 software show the existence of congestion effects under certain conditions in DiffServ domains, as well as the impact of aggregated congestion control architectures to congested domains.

1. INTRODUCTION – MOTIVATION

QoS architectures within the confines of the DiffServ model are currently being introduced into the Internet to provide preferred handling of privileged traffic-load classes [Nichols and Carpenter, 2001]. The ultimate goal of the DiffServ framework is to ensure that the targeted QoS levels can be always guaranteed through a series of proper resource provisioning procedures and strict traffic conditioning rules at the edges of the DiffServ domains.

Although this approach seems promising in real world things are not ideal. Conditioning and provisioning are rather static, slow procedures and the aggregation/deaggregation procedures taking place in complex DiffServ domains could cause packets being dropped from the queues of the edge and core routers even if all provisioning and conditioning rules are met. Moreover, there are also stochastic events that network operators may be unable to control (e.g. link failures) that may have a negative impact on the delivered QoS. In addition to these, we should keep in mind that network operators can not always guarantee the proper resource provisioning assumed by DiffServ as this approach would probably lead to a non-viable economic model for their business. Resource overbooking is very likely to remain a common practice in the near future as stated in sources both from academia [Le Faucheur et al, 2001] and industry [Class of Service, 2001].

In any case, DiffServ architectures are striving to provide edge-to-edge guarantees within one administrative domain (Per Domain Behaviours – PDB) by combining proper traffic conditioning at the edges and proper configuration of the core nodes according to some well-defined Per Hop Behaviours (PHB) [Nichols and Carpenter, 2001]. In general, any of the disturbances mentioned in the previous paragraph could lead to an error such as congestion. Congestion may lead to a violation of the targeted QoS, referred to here as a **QoS fault on a segment**.

The desired goal of providing end-to-end QoS is sought to be achieved by combining many Per Domain Behaviours, while provisioning procedures and conditioning rules are part of an SLA [Goderis, 2001] between the two operators. This means that the probability of having a violation of the agreed end-to-end QoS, referred to here as **end-to-end QoS fault,** is likely to be even much higher than the probability of a QoS fault on a single network segment.

Since congestion appears likely to be an occasional but unavoidable matter-of-fact characteristic of a DiffServ domain, the implications of such events, that may trigger end-to-end QoS faults, should be taken systematically into account as part of the **QoS semantics** of any given architecture. QoS

architectures should try not only to reduce the probability of congestion occurrence but also to control and minimise the impact of congestion effects i.e. QoS faults, on privileged traffic. Based on this last observation ASA (Active Segmented Adaptation) is suggested as an essential constituent to any QoS architecture. ASA as introduced in [DeMeer and O'Halon, 2001], intends to take care of any congestion effects by adaptation of traffic aggregates between peering domains.

The first aim of this paper is to present stronger evidence about the need for congestion control architectures, under the DiffServ framework. The next step is to extend the former work and address various challenges related to the implementation of this architecture. This includes the introduction of the SLA Broker as a domain management entity and the signalling issues related to the operation of the SLA Broker, in particular, and the ASA architecture in general. Finally, we show how congested DiffServ domains can benefit from the existence of the ASA architecture through some scenarios simulated with the NS2 [NS2] software. The rest of the paper is organised as follow:

In section 2 a series of simulations show that congestion effects are still possible in a DiffServ enabled network, suggesting implied QoS faults should be taken systematically into account as part of the overall QoS model of any given architecture. Section 3 presents the basic outlines of ASA. Section 4 presents the operational requirements and some implementation issues related to our proposal. This includes the introduction of a domain management entity responsible for monitoring the adaptation procedures in case of congestion and the resource trading between peer domains. We call this entity SLA Broker. Section 5 describes in detail the structure of the SLA Broker, the various operational parts and the algorithms used. Section 6 refers to some signalling issues associated with the proposed architecture. Section 7 presents some simulations showing the impact of the ASA architecture to the congestion scenarios described in section 2. Conclusions follow in section 8.

2. CONGESTION EFFECTS UNDER THE DIFFSERV FRAMEWORK

The aim of this section is to present a series of simulations showing that under certain circumstances congestion is still possible in a DiffServ enabled network. The first simulation scenario shows that subject to aggregation/deaggregation of traffic sources even under correct provisioning and conditioning, congestion is still possible. The second scenario shows that in the case of a congested link over-provisioning of resources, even if it is economically possible, does not help a lot since it just shifts the bottleneck to another area. The third scenario presents the detrimental impact that stochastic events, like a link failure, may have on the delivered QoS. Finally, the fourth example shows the negative impact that slightly overbooked resources may have on the overall healthiness of a system when special source synchronisation criteria are met.

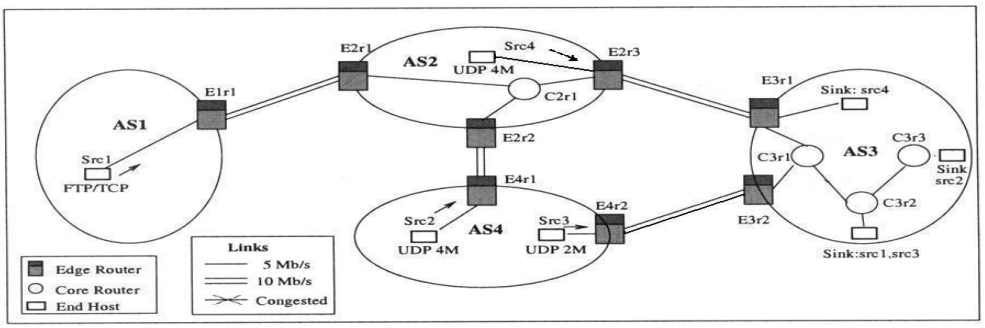

Figure 1. Test Scenario for examples 1 & 2

The simulation topology for the first two scenarios is presented in Figure 1. Four autonomous systems (AS) are interconnected by properly defined DiffServ links with capacity as indicated in the legend of Figure 1, unless otherwise stated in our examples. All intra-domain links are also DiffServ capable. Edge routers are labelled by names with initial letter R, followed by a number indicating the particular AS, the router belongs to, and a symbol indicating the particular router in that domain. Core routers are indicated by an initial letter C. Hosts are simply designated by the source running on it.

Assured Forwarding (AF) links [Heinanen et al, 1999] have been set up between the nodes with the edge routers configured for time sliding window with a three-colour marker policy (TSW3CM)[Frang et al, 2000]. One FTP source is running over a TCP connection from end host src1 in AS1 to its sink in AS3. One 4Mbps UDP CBR source is running from src2 in AS4 to its sink in AS3. Similarly src3 in AS4 is transmitting a 2Mbps CBR stream and src4 a 4Mbps CBR stream to their sinks in AS3. CBR streams have randomised departure times in order to avoid phase effects. Properly configured RED queues are used in each edge and core router. Dynamic routing is used so that alternative routes can be used in case of link failure. In normal operation the OSPF routing protocol is used.

2.1 Congestion Effects Under Proper Resource Provisioning

It seems appropriate before proceeding to the simulation scenario to explain what we mean by the term "proper resource provisioning". Provisioning rules and algorithms for optimal provisioning [Fulp and Reeves, 2001] depend on issues like the routing protocol used (static or dynamic) and if the resource provisioning is done on a single domain (per hop provisioning) or across multiple peer domains (per domain provisioning).

The rules that follow are based on two arguments. The first one is that the capacity of a single inter- or intra-domain link should be greater or equal to the bandwidth of the traffic streams carried by this link. The second argument is that the maximum delay of a link should be less or equal than the minimum accepted delay of the carried flows. The formalisation of these rules is as follows:

A. Static Routing Rules

Per Domain Provisioning

If traffic from n traffic sources s_i (i = 1 to n) each of which have a Quality Vector $Q_i = (B_i,d_i)$, (where B_i = the desired bandwidth and d_i = the maximum delay), is to be carried from a Domain D1 to a Domain D2 through an inter-domain link D1D2 then the link must have the following properties:

- $B_{D1D2} \geq \sum_i Bi$
- $d_{D1D2} \leq Min(di)$, i= 1 to n

Per Hop Provisioning

If an internal link l_i carries traffic from n traffic sources s_i (i = 1 to n) each of which have a Quality Vector $Q_i = (B_i,d_i)$, (where B_i = the desired bandwidth and d_i = the maximum delay), then the link's quality vector must have the following properties:

- $B_{li} \geq \sum_i Bi$
- $d_{li} \leq Min(di)$, i= 1 to n

If the link or the domain carries a portion of the traffic coming from source s_i then B_i represents the portion of this traffic while d_i is still the maximum accepted delay.

B. Dynamic Environment Rules

Per Domain Provisioning

If traffic from n traffic sources s_i (i = 1 to n) each of which have a Quality Vector $Q_i = (B_i,d_i)$, (where B_i = the desired bandwidth and d_i = the maximum delay), is to be carried from a Domain D1 to a Domain D2 through m possible different inter-domain links l_j, j=1 to m, with quality vectors $Ql_j = (Bl_j, dl_j)$ then the following properties must be held:

- $\sum_{j=1}^{m} Blj \geq \sum_{i=1}^{n} Bi$
- $\forall s_i$, i=1to n, with $Q(s_i) = \{B,d\}$ $\exists$ $l_k \subseteq l_j$: $dl_k \leq d$ and $\sum_{j=1}^{k} Blj \geq \sum_{i=1}^{r} Bi$, where r is the potential number of sources served by l_k group of links

Per Hop Provisioning

Let p_{ij} be a path from node i to node j of a domain consisting of a series of intra-domain links $\{l_i, l_k, \ldots, l_k,\ldots,l_j\}$. Then for every source s_i with quality

vector Q = {B_i, d_i}, the following rules must be valid:

- $\forall s_i$, i=1to n, with Q(s_i) = {B,d} that has to be routed from node i to node j $\exists$ p_{ij} : $\forall l_i \subseteq p_{ij}$, $dl_i \leq d$ and $Bl_i \geq B$.
- $\forall l_i$ potentially servicing r sources, $Bl_i \geq \sum_{i=1}^{r} Bi$ and $dl_i \leq Min(di)$, i= 1 to r

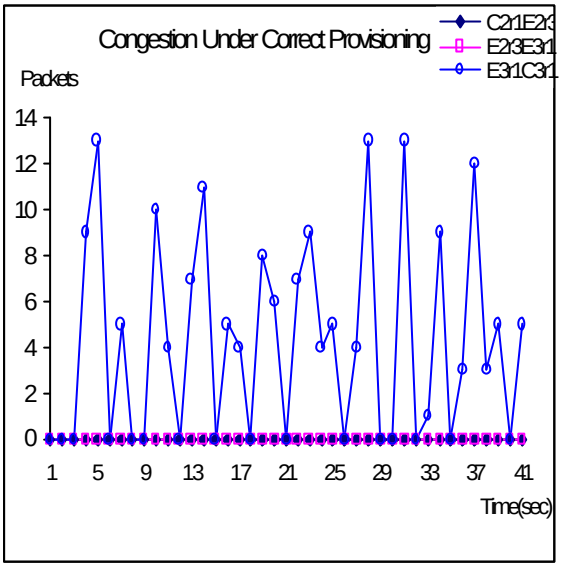

Figure 2. Packet drops in a properly provisioned DiffServ Domain.

According to these rules in order to have a properly provisioned network for this example the capacity of the E2r3-E3r1 link is set to 9Mbps and the capacity of the C3r1-C3r2 link to 7Mbps. All other links have the capacity stated on the legend of Figure1.

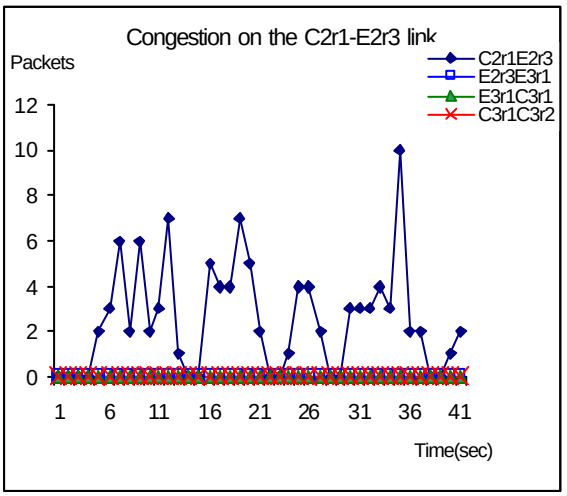

Figure 3. Congestion on the C2r1-E2r3 link

Figure 2 represents the packet loss per second for a period of 40secs at certain links (C2r1-E2r3, E2r3-E3r1 and E3r1-C3r1). According to the results some packets are dropped from the queue of C3r1 router. The maximum number of packets dropped in a single second is 13. Having set the packet size at 2000bytes this means that the maximum loss is up to 13*2000*8=0.208Mbps. On a 5Mbps link this is about 4.2% of the total traffic. Also, notice that there are no packet drops on the 9Mbps E2r3-E3r1 link and since 4Mbps are deaggregated at the E3r1, normally the 5Mbps of the E3r1-C3r1 link should be enough for the rest of the traffic. What happens here is that the aggregation of the 2Mbps stream from traffic src3 with the rest of the traffic causes some packet drops from the queue of the C3r1 router. It is obvious that there is no provisioning error since the C3r1-C3r2 link that accepts the whole of the traffic is set to 7Mbps, which is enough for the 2Mbps coming from src3 and the 5Mbps offered by the E3r1-C3r1 link. It is the aggregation of these two different traffic streams that causes some packets being dropped from the RED queue of the C3r1 router. Therefore, it can be argued that in more complex domains with more aggregated/deaggregated streams, packet drops will be possible even under correct resource provisioning.

2.2 Single Hop Over-Provisioning In Slightly Congested Domains

In this scenario the network topology is presented in Figure 1 and the links' capacities are the ones described in section 2.1. We have created a small bottleneck on the C2r1-E2r3 link by allowing FTP traffic coming from source1 to enter AS2 at rates higher than 1Mbps. The goal is to examine whether hop-by-hop over-provisioning can help congested DiffServ domains in an effective way.

Four different consecutive links are monitored. C2r1-E2r3, E2r3-E3r1, E3r1-C3r1 and C3r1-C3r2.

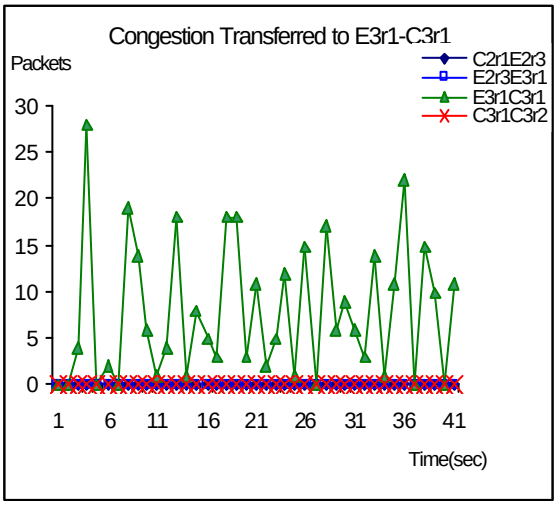

Figure 4. Congestion mainly to E2r3-E3r1

Under the initial system configuration, congestion effects appear on the C2r1-E2r3 link as presented in Figure 3. The administrator of the AS2 decides to add more capacity and this link is upgraded from 5Mbps to 5.2Mbps. Figure 4 shows the impact of this upgrade. The initially congested C2r1-E2r3 is relieved but the real problem is not solved as now congestion is transferred mainly to the E2r3-E3r1 link and partially to the E3r1-C3r1.

The next step is to add some more capacity to the E2r3-E3r1 link and upgrade it to 9.2Mbps. The impact of this change is presented in Figure 5. According to the simulation results the congestion is now transferred fully to the E3r1-C3r1 link.

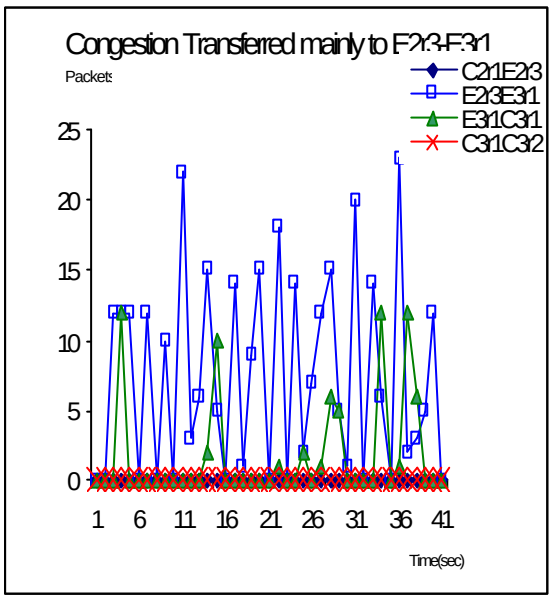

Figure 5. Congestion on the E3r1-C3r1 link

Moving a step further and upgrading this link form 5Mbps to 5.2Mbps just moves congestion to the C3r1-C3r2 region as shown in Figure 6. Since C3r2 router is connected to traffic sinks, adding some more capacity to the C3r1-C3r2 link would solve the congestion problem.

What is illustrated by this example is that in case of network congestion hop-by-hop over-provisioning of resources does not help much as it just shifts bottlenecks from one domain to another or to different hops within the same domain. End-to-end over-provisioning on the other hand would solve the problem but such solutions may not always be economically viable and may also require co-operation between operators of different domains in order to be successful. Moreover, over-provisioning across multiple domains is a slow procedure (may take from hours to weeks) and obviously is not suitable for an environment where actions should be taken at definitely shorter timescales. In addition to this, just adding more capacity to the system, only solves the problem temporarily. Users will be happy to use all the bandwidth offered to them (greedy behaviour) and after a short period of time the extra resources will be exhausted and congestion effects will be back.

What is really needed here is some sort of dynamic co-operation of the domains at the traffic aggregates level. As stated in [DeMeer and O'Halon, 2001] "QoS error recovery actions would be performed on whole traffic aggregates in accordance with the DiffServ model and are suggested to be specified as part of (bilateral) SLAs among providers." This is the main concept of the

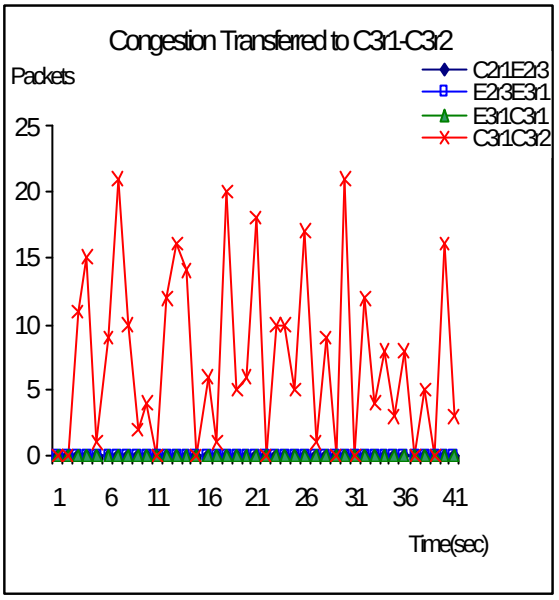

Figure 6. Congestion on the C3r1-C3r2 link

ASA architecture, which is discussed in more detail in this paper.

2.3 Impact Of Stochastic Events (Link Failure) On The Network Performance

Even if provisioning could be done in such a way that there is no congestion in the network, there are always stochastic events that unfortunately can not be controlled a priori but may have a negative impact on the QoS of the delivered traffic. Link failure is such an event and it is shown here that an incident like that does not only cause bottlenecks inside a network but also heavily penalises TCP traffic coming from various domains. In order to simulate this scenario some changes have been made to the topology presented in Figure 1. An AF 5Mbps link has been added between the E4r1 and E4r2 routers. This means that since OSPF is used, the whole traffic from AS4 is transferred to AS3 through the established 10 Mbps E4r2-E3r2

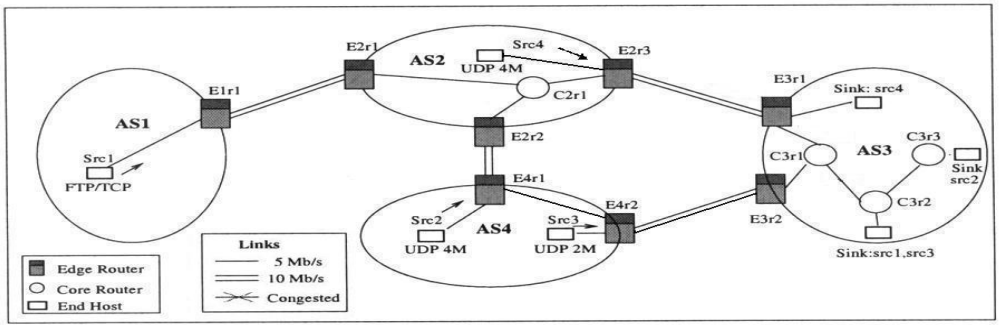

Figure 7. Test Scenario for examples 3&4

intradomain link. Also, the capacity of the E2r2-C2r1, E3r2-C3r1and C3r1-C3r2 links is set to 7Mbps. This scenario is presented in Figure 7. All the other topology links follow the legend presented in Figure 7. Dynamic Routing is used in order to detect and react to any changes in the topology. The traffic sources are the same as these described earlier in the first simulation scenario.

The simulation scenario assumes that between 10sec and 20sec the link E4r2-E3r2 fails, so the whole traffic from AS4 is transferred to AS3 through AS2 (Dynamic Routing is used). This has a great impact on the C2r1-E2r3 link as it causes heavy congestion, which also results in heavy penalisation of the FTP traffic coming from AS1. Figure 8 shows the simulation results.

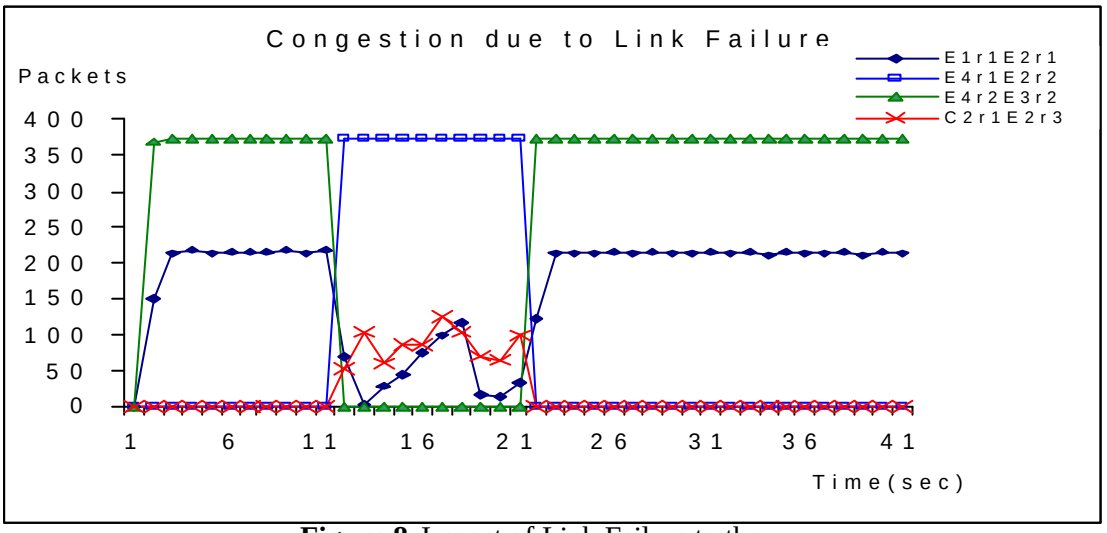

Figure 8. Impact of Link Failure to the delivered QoS

The E4r2E3r2 line represents the number of packets transferred from AS4 to AS3 through the specific link. From 10sec to 20sec this line drops to 0 packets (link failure). The E4r1E2r2 link represents the amount of packets transferred from AS4 to AS3 through AS2. This happens only when the E4r2-E3r2 link fails (10sec to 20sec). In fact the E4r1E2r2 and E4r2E3r2 lines are complimentary. The C2r1E2r3 represents the amount of packets **dropped** at the specific link. According to the above results, the E4r2-E3r2 link failure causes heavy congestion on the AS2. Finally the FTP traffic offered to AS2 from AS1 is represented from the E1r1E2r1 line. It is obvious that FTP sources are heavily penalised because of the specific link failure.

It is assumed in this scenario that proper, well-defined re-routing mechanisms exist in order to take action in case of any, hopefully rare, stochastic events. This is not always a trivial task especially in DiffServ networks [Raymont and Srihari]. This fact stresses even more the negative impact that any possible stochastic errors may have on the performance of the DiffServ architecture.

2.4 Congestion Under Resource Overbooking

Correct Provisioning across multiple domains is not always an economically viable solution for the network operators. Overbooking of resources is a common practice in today's networks and this is

not expected to change dramatically over the next few years [Le Faucheur et al, 2001], [Class of Service, 2001]. What operators hope when following this approach, is that traffic sources will not be synchronised (offer their maximum traffic burst at the same time). In this scenario the impact of a single overbooked link to the delivered QoS under the DiffServ Framework is presented.

The topology is the same as presented in Figure 7. To make the scenario a bit more complicated each host offers the following types of traffic:

- **src1:** 4Mbps CBR UDP traffic and FTP/TCP traffic (up to 1Mbps)
- **src2:** 4Mbps CBR UDP traffic and FTP/TCP traffic (up to 1Mbps)
- **src3:** 2x2Mbps CBR UDP traffic
- **src4:** 4Mbps CBR UDP traffic

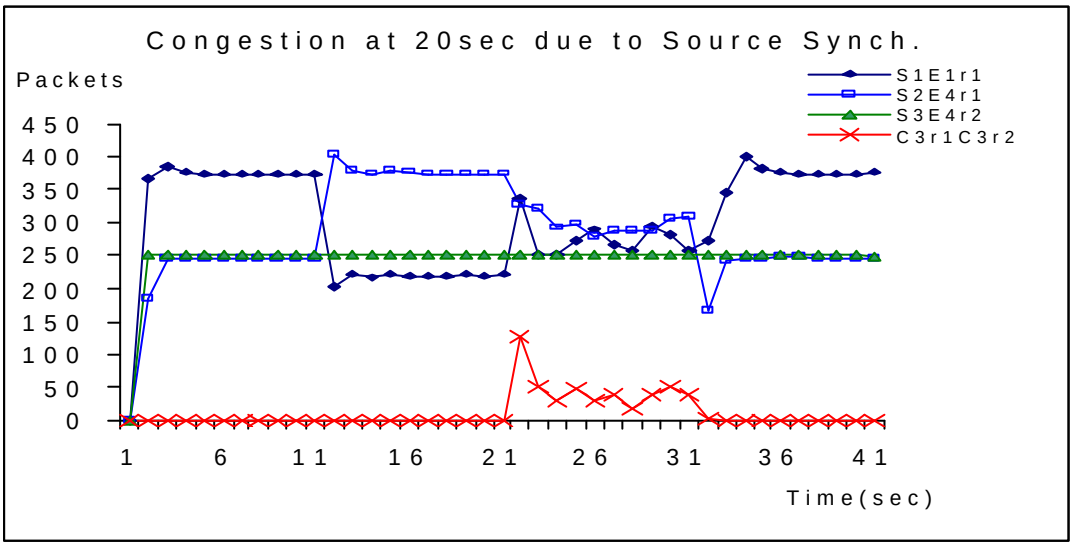

Figure 9. Congestion due to Overbooked Link and Source Synchronisation

The capacity of the links follows the legend of Figure 7 with 2 exceptions. E3r2-C3r1=9Mbps and C3r1-C3r2=12Mbps (overbooked link). No link failure is assumed. The simulation results are presented in Figure 9.

The S1E1r1 line describes the traffic offered by Src1 to the network. From 1sec to 10sec both FTP and UDP sources offer traffic while from 10sec to 20sec only the UDP source is on. The inverse procedure happens with the traffic offered by src2, which is represented by the S2E4r1 line. Src3 offers at all times the same amount of UDP traffic. The C3r1C3r2 line shows the dropped packets at the overbooked link. The critical period starts at 20sec where both src1 and src2 offer their maximum traffic at the same time. Congestion at the overbooked link is severe and this has a negative impact on the FTP traffic offered by src1 and src2. Later in this paper we will show results of how such a congested network would have benefited by the incorporation of the ASA architecture as part of the overall DiffServ framework.

3. BASIC OUTLINES OF ASA ARCHITECTURE

All the simulations presented in Section 2 show that congestion is still possible under the DiffServ Framework. Thus, DiffServ could be characterised as a **defective** QoS architecture. The term "defective" is not in contradiction with the value and the usefulness of the DiffServ proposal. It only suggests a more realistic view on the way that DiffServ domains are likely to operate. This means that any, hopefully rare, QoS faults should be considered well in advance when examining the semantics of a QoS architecture and precaution should be taken against them. Only if possible congestion effects are completely ignored then a defective QoS architecture may prove inefficient and incapable of offering predefined QoS guarantees.

As presented in Section 2.2, providing extra resources in order to mask QoS faults does not always solve the problem if not done on an end-to-end basis. Moreover, It is generally questionable whether end-to-end provisioning is always possible in a reliable way and at reasonable cost. Thus, some other form of co-operation is needed in order to experience end-to-end QoS and reduce the impact

of congestion effects to a minimum. IP networks tend to be self-organised; this means that central approaches create new bottlenecks and do not provide viable solutions. Instead, more distributed and hop-by-hop (or peer-by-peer) solutions stand much more possibilities to be successful in the modern IP environment.

The desired co-operation could partly be realised on the application level, as it happens today with many TCP-friendly applications like e-mail, ftp, web browsing. Such behaviour is always welcome, but the problem is that, subject to various reasons, such an application specific co-operation cannot always be guaranteed. For example, many applications are not willing to adapt in a dynamic QoS environment because they are time critical and use native UDP as transport protocol. This means that although congestion avoidance and recovery across multiple IP networks can benefit from the co-operation at the application level, unfortunately this sort of co-operation cannot be taken for granted and operators should not rely only on this for providing end-to-end QoS solutions.

On the other hand, in DiffServ-enabled networks the idea of traffic aggregates is introduced. This means that different traffic flows with the same QoS requirements are aggregated in a single traffic class, which receives the same quality behaviour across multiple autonomous systems. The received quality is stated in the SLAs established between the network operators. What is suggested by the ASA architecture in [DeMeer and O'Halon, 2001] is that QoS fault recovery actions would be performed on whole traffic aggregates in accordance to the DiffServ model for scalability and flexibility reasons. Moreover, the ASA architecture suggests that the recovery actions should be part of the bilateral SLAs established among providers. The introduction of **service curves** [Jain and Ramakrishnan, 1988; Hashem, 1989] can help the definition of different QoS levels within one SLA. Adaptation and recovery are now realised as smooth movement between the pre-established service curves.

The service curve/arrival curve concept has been introduced and used in different places but especially by R.L Cruz and J-Y Le Boudec. Service curves are used within the ASA architecture in order to simplify the "rigid definitions in traditional SLA"[DeMeer and O'Halon, 2001] of QoS guarantees as the basis of SLA. It is still work in progress how these service curves will be defined and what the formalisation will be. The goal, however, is that they are to be used as a tool in order to express and maybe quantify different, possibly related, QoS levels within the same SLA. Service curves represent an abstract linear filter for the domain and are closely related to the policies within a domain. In that sense the internal details do not need to be fully visible to the adjacent domain operators/providers. What peer operators definitely need to know is the service curve according to which they will adapt the aggregate in case of congestion.

A service curve example is presented in Figure 10. It describes the relationship between the bandwidth guarantees offered to an AF-based SLA and the percentage of packets marked at the RED gateway (sign of congestion). The service curve represented by the thick black line acts obviously as a filter. When the percentage of marked packets is less than P1, then the bandwidth guarantee is set to BW1. If the marking percentage is between P1 and P2, then bandwidth guarantees fall to BW2. In the same vein

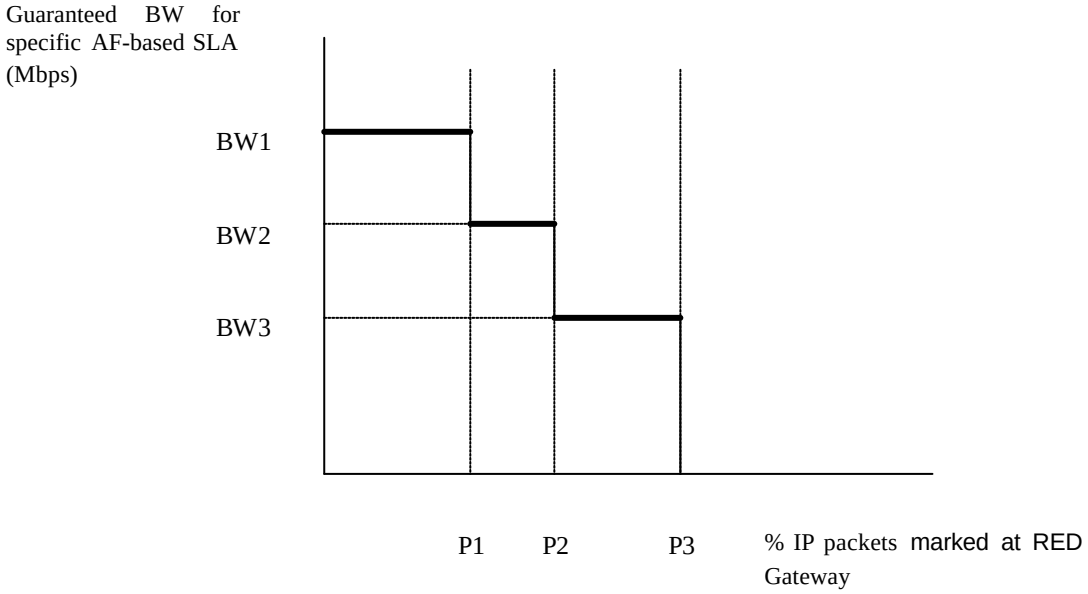

Figure 10. Service Curve Example

if the marking percentage is between P2 and P3 the guaranteed bandwidth falls to BW3. For marking percentages greater than P3, then traffic is degraded to Best Effort (zero bandwidth guarantees). As soon as both operators are aware of the specific curve and the values of P1, P2, P3, BW1, BW2, BW3 they can co-operate in case of congestion automatically without any human intervention.

The idea behind the proposed architecture is inspired by the way TCP works and the belief that elasticity between service providers is crucial for the success of the DiffServ model. It is obvious that the viability of the best-effort model is primarily based on the elasticity and TCP-friendliness of the popular applications. In the same vein, providers should be elastic and allow in-profile traffic to be adapted in the hopefully rare case of a QoS fault. This sort of behaviour establishes a backward compatibility with the TCP-like end-to-end congestion control but now this method is extended to whole domains. The analogy between TCP and ASA architecture is as follows:

In the Best Effort TCP world some active queuing mechanism like RED [Floyd et al, 1993] is used inside the IP routers. When congestion occurs, IP packets are marked or dropped. This is a signal for the TCP congestion avoidance algorithm to reduce its window size and make a slow start. Thus, the applications back off and the network recovers from congestion. In the same analogy in the DiffServ world, some sort of congestion notification signalling should trigger the dynamic adaptation of traffic aggregates between neighbour Autonomous Systems.

This means that upon the occurrence of congestion the network operators or service providers should co-operate by allowing the smooth movement between different service curves, as these are already defined inside their bilateral SLAs. For example providers may agree to reduce the exchanged data rate to a certain percentage of the initial value upon the reception of a congestion signal. The adaptation of traffic aggregates is performed at the edges and a special network entity called SLA Brokers monitors and supports the ASA procedures when needed.

4. OPERATIONAL REQUIREMENTS OF THE ASA ARCHITECTURE

The philosophy and the basic outlines of the ASA architecture are described in Section 3. In this section we give a more engineering approach to our proposal and try to explain how this architecture can be implemented within the confines of the DiffServ Framework. The goal is to identify the operational requirements of the ASA architecture and propose appropriate solutions.

4.1 Congestion Detection And Notification

The ASA architecture tries to reduce the probability on any congestion effects, on the one hand, and impact of any existing congestion effects to the rest of the network, on the other. Therefore, it is crucial to have a well-established mechanism for detecting congestion effects and notifying the edge routers or any other possible domain management entities about these faults.

The congestion detection algorithm used should have some specific characteristics so that it can be easily integrated with the DiffServ environment and the proposed adaptation techniques. The operator's key resources are bandwidth and buffer space. Usually the operator is willing to trade off buffer resources to ensure high utilisation of bandwidth [Floyd et al, 1993]. This means that an operator may decide to give buffer space in order to store packets in the router if the capacity of the link is less than the traffic load of the serviced traffic flows. In this way, packets are not dropped but the system suffers from high queuing delays. High delays have a negative impact on real time traffic. Thus, both high queuing delays and high drop rates are not desired. Two metrics, which could be used to indicate congestion, are the Average Queue Length and the Packet Drop Rate. This means that any suggested congestion detection algorithm should be based on **at least** one of this metrics.

Another desired characteristic for the candidate algorithm is that it should be suitable for a wide variety of environments and transport protocols. Because it will be used in some sort of rate adaptive environment (smooth movement between the predefined service curves) the method should fit better to environments with rate-based flow control rather than window based.

The algorithm should be able not only to drop the congested packets but also to mark them by setting special flags in the packet header. This is very important, as the congested packets in our architecture should not only be removed from the network but they should be used in order to notify the edge routers about the congestion effects.

In addition to the above, the selected algorithm should be fair in the sense that it does not discriminate against any particular connection or traffic class. This means that the fraction of marked packets for each connection should be roughly proportional to the connection's share of the bandwidth. Last desired feature for the congestion detection algorithm would be to provide

mechanisms in order to be able to identify the misbehaving aggregated flows in the domain. This sort of information would be very useful in our approach since it could help us choose in a fair way the SLAs that should adapt.

Several methods have been proposed, with some of the most popular being the DECbit congestion avoidance scheme [Jain and Ramakrishnan, 1988] and the Random Drop and Drop Tail Gateways [Hashem, 1989]. But the most widely used method today, especially under the DiffServ framework, which has all the desired properties mentioned above, is the RED (Random Early Drop) Gateway [Floyd et al, 1993].

Coming to the congestion notification procedure, the desired solution should work with any Transport Protocol, which may or may not provide ACK signals. Moreover, since in case of wireless links a missed ACK is not always a congestion signal it can be argued that implicit Congestion Notification is not a suitable solution. The Internet Community is considering using Explicit Congestion Notification (ECN) for IP by just using bit flags in IP packet header.

This approach is well suited for the DiffServ framework, where the IPv4 TOS octet is split into 2 parts. The first 6bit part is used as a DS-field (DSCP values) and the last two bits are used for Explicit Congestion Notification. The full in-detail description, of how the 2 ECN bits are used in IP networks, is presented in [Ramakrishnan et al, 2001].

4.2 Active Network Devices For The ASA Architecture

Since ASA architecture is applicable on DiffServ enabled domains, we assume that there is a number of edge and core routers in each administrative domain. This means that in case of congestion the marked packets must be forwarded to the edge routers. In fact, the packets should be forwarded to the ingress router of the specific aggregate who is responsible for the adaptation of the specific SLA. An obvious solution in order to ensure that congestion notification signals arrive to the correct edge router is to broadcast them. Since this approach may cause traffic overhead in the domain, it is also possible to unicast such notifications by using an active intermediary that would be responsible for specifically routing the congestion notification signals.

When the percentage of the marked packets reaches a certain threshold, the edge routers may decide to start an adaptation procedure. Setting the appropriate threshold to the edge routers depends on how much the administrator wants to protect its network by trading off performance with some potential profit loss (coming from the adaptation of traffic aggregates). In the same way if no marked packets arrive for a certain period of time the edge routers may try a recovery procedure for the adapted SLAs.

The adaptation procedure has definitely an impact on the QoS guarantees offered to the peer domains. How each peer operator will adapt its own domain(s) to the new situation is an internal problem. It is possible either to use internal techniques to mask the fault or to carry on with the adaptation of traffic aggregates with its neighbour IP networks in order to reach a stable point of operation. In case adaptation moves all the way back to the access domains, then each local policy router (or bandwidth broker) should have its own rules and policies for the non-elastic applications. It is obvious that local policy routers should provide incentives for the applications to back off or they should by default penalise the non-elastic ones. In any case this is a local policy and has no impact on the operation and design of the proposed architecture.

It is important to notice here is that the main complexity of the ASA architecture is transferred to the edge routers. The core routers are only responsible for treating the DSCP-marked traffic flows according to some well-defined PHBs as well as issuing the appropriate signals in case of congestion. This establishes a compatibility with the overall DiffServ model and we believe that restricting the majority of the ASA overhead at the edges is crucial for the success of the proposed architecture.

On the other hand, our approach introduces a dynamic factor on the execution of the SLAs and also embraces a certain administration and housekeeping overhead on the overall system. This means that issues like the smooth movement between the various predefined service curves and the introduction of different fault classification and adaptation mechanisms have to be faced successfully. A clear view is needed at all times about the status of each SLA. Moreover, issues like profitability, fairness, complexity and communication overhead must be considered before starting an adaptation or recovery procedure.

In order to cater for the administration and housekeeping procedures and serve the need for resource trading between peer domains, a domain management entity, called SLA Broker, is introduced. The SLA Broker is considered to be the Policy Decision Point of the ASA architecture while edge routers are the Policy Enforcement

Points. SLA Brokers must communicate with the edge routers and, moreover, peer SLA Brokers may also need to communicate mutually in order to exchange information about pricing, statistics, policies and the status of the bilateral SLAs. The concept of the SLA Broker, introduced here, does not assume a single centralised network device. SLA-Broker can also be realised as a distributed algorithm of the domain running in several points.

Different communication interfaces are defined between the various entities of the ASA architecture. As mentioned in the previous paragraph, SLA Brokers have their own communication requirements with the domain edge routers (intra-domain level) and peer SLA Brokers (inter-domain level). Moreover, a communication path is definitely needed between peer edge routers as well as between the edge and the core routers of

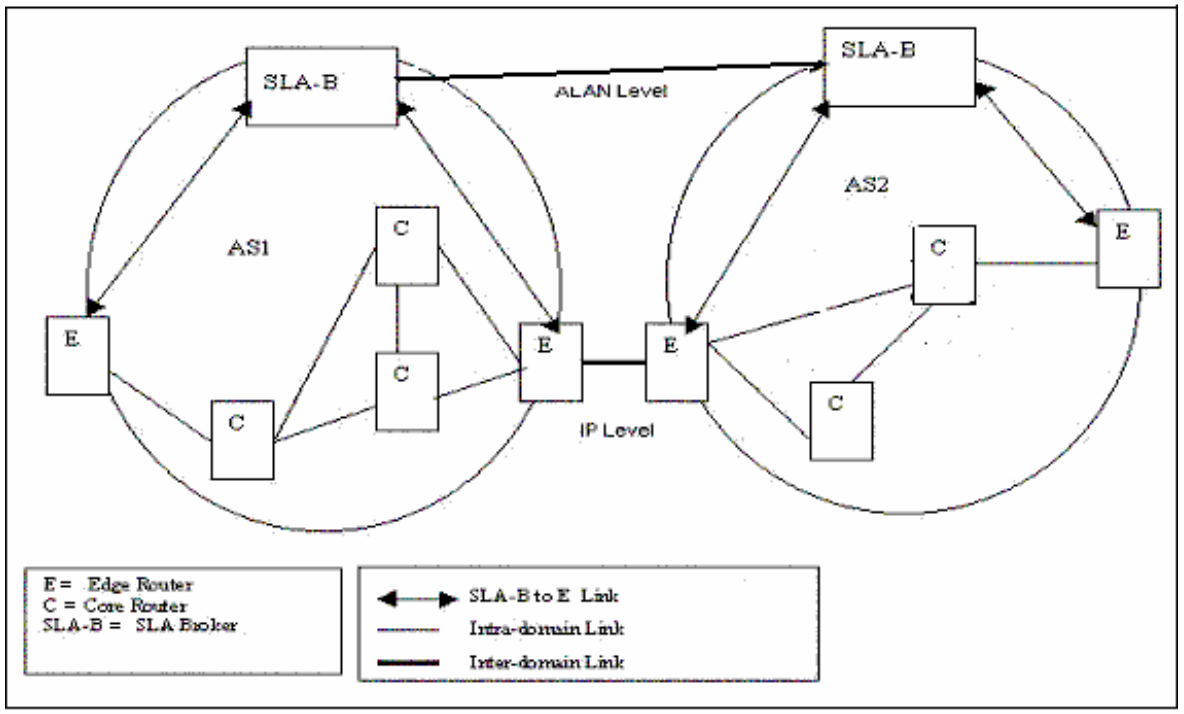

Figure 11. Block Diagram of ASA Architecure

every administrative domain. Figure 11 describes the various communication interfaces that apply in a minimum ASA configuration. Please note that no specific signalling solutions are proposed here for each of the defined interfaces. This is done in detail in section 6 of this paper.

Peer edge routers communicate at the IP or "router" level while the communication between SLA Brokers or SLA Brokers and edge routers is happening at the application level. Since edge routers may be re-configured by the SLA Brokers some sort of Active Networking mechanism is required. The concept of Application Level Active Networking (ALAN) as defined in [Ghosh et al, 2000] or the bi-level Active Networking approach [Crowcroft et al, 2001] are used in order to provide an easy mechanism for the programming of the edge routers during the SLA set-up time.

The ALAN approach leaves the basic network infrastructure unchanged and provides a framework for deployment of application level active services. General-purpose computational nodes are placed inside the network at strategic locations to provide execution environments that can be dynamically installed to perform application level functions. These nodes interfere in a limited way with the network transport mechanisms. An ALAN-based ASA implementation is described in [DeMeer and O'Halon, 2001].

On the other hand, the bi-level Active Networking as described in [Crowcroft et al, 2001] is based on the existence of Active Routers (AR) that may be reprogrammed by special data packets, and Active Servers (AS). AS act as proxies for any additional services offered by the network such as Transcoding or Dynamic VPNs. AR and AS communicate/compliment each other in order to serve any possible "active " flows.

5. STRUCTURE, OPERATION AND ALGORITHMS OF THE SLA BROKER

5.1 Operational Requirements And Structure Of The SLA Broker

The concept of the SLA Broker as a management entity in each domain has been introduced in section 4. The SLA Broker is responsible for setting up DiffServ connectivity and supporting the adaptation of traffic aggregates mechanisms mainly residing in the edge routers. In fact, SLA Brokers support a whole range of operations, through a set of internal engines, databases and I/O interfaces to other network devices like edge routers. SLA Brokers are the Policy Decision Points (**PDP**) of the proposed architecture. They communicate with the edge routers (Policy Enforcement Point – **PEP**), the peer SLA Brokers and the domain administrator, who is responsible for loading the current policies, through a management console. Before defining the architecture of an SLA-B, we should examine the range of operations that it should support. The SLA Broker should:

- Advertise possible bids
- Service SLA requests from neighbour domains and make SLA requests to neighbour domains
- Monitor the Segmented Adaptation of Aggregates under congestion as well, for the necessary recovery procedures
- Keep track of all sold and bought SLAs, the default resources of the residing domain and the internal status of each SLA, for administration and pricing reasons
- Configure dynamically the edge routers of the domain during the SLA set-up time and in the rare case that the pre-loaded policy has to be changed during the system operation for performance reasons.

All these requirements mean that the SLA-B should have a Database for keeping all needed information and some operational engines. These engines according to some algorithms and policies loaded by the administrator and by taking into consideration the databases information and external inputs (e.g. congestion signals or signals from edge routers) will take actions on a whole range of different events. These events will normally have to do with procedures of advertising bids (Bid Advertiser), accepting or rejecting requests (SLA Trader), performing SLA requests to peer domains (SLA Trader) or supporting the adaptation of traffic aggregates in case of congestion (Active Segmented Adaptation – ASA Engine).

During the system operation the SLA Broker may need to exchange data and update the peer SLA Brokers or reconfigure the domain edge routers. It is NOT suggested here that each time congestion arises new code has to be downloaded on the edge routers. It is anticipated that the dynamic execution of the SLAs through the smooth movement between the predefined service curves is happening at the edge routers without the SLA Broker intervention. On the other hand, during the SLA set-up time the SLA Broker should download the desired policy to the edge router and moreover, there may be certain cases where a new policy has to be downloaded to the edge routers because of a severe change or fault in the domain. We believe that the concept of Policy-based extensible hierarchical Network management as proposed by the TEQUILA workgroup in [Flegkas et al, 2001], is important in our approach. Figure 12 presents the block structure of an SLA Broker.

5.2 The SLA Broker Database

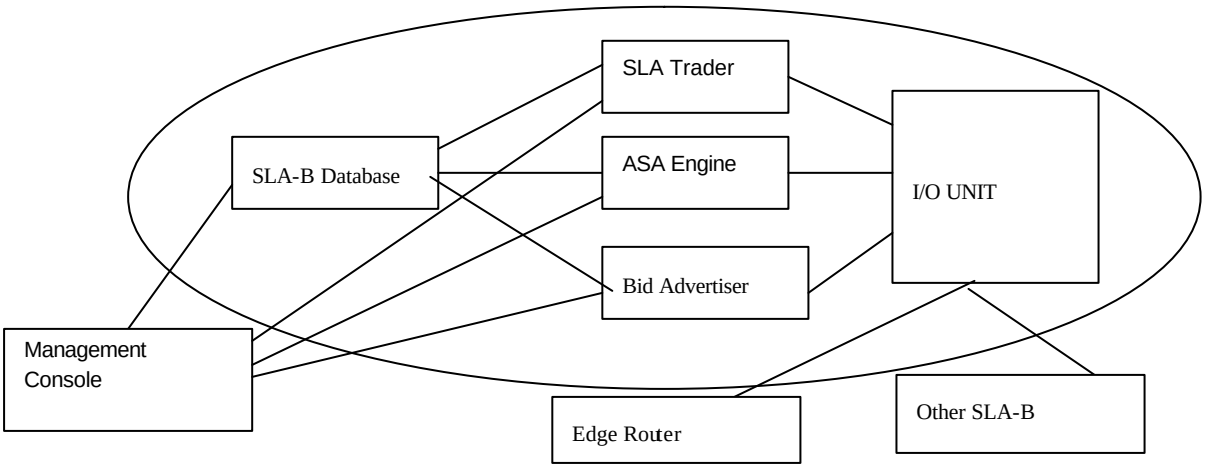

Figure 12. Structure of the SLA Broker

The SLA-B Database during the system bootstrapping should hold information regarding the established links between the residing domain and the neighbouring ones. The knowledge of the a priori established connections is very helpful for the bid advertiser in order to advertise a number of different possible bids to the peer domains as well as for the SLA trader that will decide whether to accept SLA requests from peer domains.

During the system operation, SLAs are exchanged between peer domains and the database must keep track of all these SLAs and the current internal status of each SLA (service curve) at any period of time. The SLA Broker should always have a clear view of the type of traffic that is serviced through its domain and the way the resources are allocated in the edge and core routers for each one of supported traffic class. The algorithms running on the different engines of the SLA Broker need this sort of information in order to be more effective.

In addition to the above, it might be helpful for the SLA-B to hold information about the past service usage and the bids made between the neighbouring domains. By using this information and the current usage patterns, the SLA Trader engine may be able to decide if bandwidth should be bought in advance and then higher capacities can be advertised by the bid advertiser at lower prices.

5.3 The Bid Advertiser And SLA Trader Engines

The SLA Trader takes decision on the acceptability of an SLA request from the peer domains. It examines the available resources, the requested QoS and reports to the edge router about the decision. A record of accepted and rejected SLA requests must be kept in the database.

The SLA Broker is also responsible for accepting or rejecting bid offers from other domains. The decision depends highly on a profitability analysis algorithm that is loaded by the system administrator and runs in the SLA trader engine. Usually this algorithm sets a window of past trades and based on this window calculates the average past price per Mbps and it also uses prediction techniques like time series or regression analysis in order to determine the future demand. If the product of the predicted demand by the average price is greater than the price of the offered bid then the bid is accepted [Frankhauser et al, 1999].

On the other hand, there are several different trading engines which try to sell the available resources of the domain to peer domains by combining information from the past and the present usage patterns (information kept in the SLA-B Database). Some well-known traders like the *greedy trader*, the *trendy* trader and the *profitable* trader are presented in detail in [Frankhauser et al, 1999]. The analysis of the various trading engines and profitability analysis algorithms based on classical and modern economical theory is far beyond the scope of this paper.

The bid advertiser tries to combine information from the SLA Database (domain resources and current utilisation) in order to advertise a series of possible bids to the peer domains. In fact, the bid advertiser is a mechanism used in order to make known the results of the trading algorithm, running in the SLA Trader engine, to the peer domains in the form of bid offers.

A big variety of different SLA bids is always desired because if the peer SLA Broker has many choices, it is more likely to accept a bid, thus, buy our "product". On the other hand, very frequent bid advertisements have a large communication overhead. Since the resources are owned by the source SLA Broker, the plethora of different bids to be advertised may end up in eating a considerably high percentage of the resources, leaving less space for selling services. This means that very frequent bid advertisements may lead to congestion effects, therefore causing QoS "faults" in a properly provisioned DiffServ domain.

The above states clearly that an efficient lightweight advertising protocol is needed between the peer systems. We examine a solution based on the use of BGP-4 as a way to distribute flexible QoS information in section 6.

5.4 The ASA Engine

Finally, the ASA Engine is responsible for supporting the adaptation procedure in case of congestion and for the recovery operation when congestion effects are no longer present. This means that the ASA engine, during the SLA set-up time, should download to the edge routers a certain policy associated with the adaptation and recovery procedures of the specific SLA. After that the edge router is responsible for the reacting to any congestion signals. Upon receiving a congestion signal the edge router stops "hearing" for additional congestion signals and chooses an SLA to adapt. The service curve of the SLA is changed and the local SLA Broker Database is updated as well as the peer SLA Broker for billing and consistency reasons. Then, the edge router starts "hearing" again for congestion signals. If congestion persists, the whole procedure has to be repeated. If no congestion occurs for a reasonable period of time (defined by the administrator) then the recovery

procedure starts in order to bring the downgraded SLA(s) back to their original status. This recovery procedure may fail and in that case it may be repeated after a random period of time and preferably with another downgraded SLA.

The most crucial part of the algorithm, which in fact determines its effectiveness, is the policy downloaded by the ASA engine, based on which the edge routers select the SLAs that will adapt upon the reception of a congestion notification signal. We argue that this policy is a trade-off between simplicity, low protocol traffic overhead, profitability and effectiveness of the segmented adaptation procedure. A variety of different policies should exist in order that the best one can be applied for each particular case. There may be algorithms that only care about low signalling overhead and thus upon congestion notification move all SLAs to another service curve, while there may be other approaches that care only for the profitability or the fairness. Usually the most effective methods are these who combine two or more of the above policies. These methods, proposed here, have their motivation in some methods for bandwidth brokers and resource traders as proposed in [Frankhauser et al, 1999; Makris, 1999; Frankhauser and Plattner, 1999]. The most important out of them, which best suit our architecture, are:

- The **lazy policy.** In this case, the algorithm chooses all SLAs willing to adapt and moves them to another Service Curve. The obvious advantage is that no extra protocol information is needed about the source of congestion. The disadvantages are that this policy is not fair, it has a great negative impact on the profitability (as compensation for adaptation must be given to all SLAs) and it creates lots of traffic overhead in order to reconfigure the edge routers and inform peer SLA-Brokers about the adaptation. Moreover, fragmentation of resources may be caused.

- The **priorities policy.** The different SLAs are sorted to groups of different priorities. During the adaptation procedure, the method selects first SLAs of the lowest priority and if the congestion is not avoided, it moves to higher priority SLAs. This method is better than the lazy policy but it has again problems of fairness, as it does not connect the real source of congestion with the adaptation procedure.

- The **profitable policy.** The goal is the maximisation of profit. The algorithm examines which SLAs have the lowest demands for compensation and selects them for adaptation. It may be a popular method among administrators but since it is not sure that the real SLA congestion source is penalised the method is not really fair. The computational cost may also prove bigger than in any other method as it is more probable that low compensation SLAs do not take much system resources so more than one will need to be adapted in order to relieve congestion. Moreover, since more than one SLAs may have to be degraded this can cause fragmentation of resources in our system.

- The **fair policy**. This method requires that together with the congestion notification signals information about the source of congestion be also received. That is, some sort of routing information about the misbehaving packets or the packets that take a really large portion of the available bandwidth is needed. With this information, the algorithm can easily find the SLAs that are responsible for congestion and adapt them. The problem with this approach is that it requires some extra communication overhead and that it does not care for the profitability part of the adaptation procedure. In any case it almost assures that congestion is really avoided at the first place.

- Finally, the **trendy policy.** The proposed algorithm tries to combine the **profitable** and **fair** policies together. According to this method, the edge router receives information about the source of congestion and it tries to make the fairest choice at the lowest compensation cost. We are talking about an optimisation problem so this method is computationally expensive and it also has a high communication overhead. On the other hand, it provides the best solution to the problem of choosing a suitable SLA to adapt in order to move congestion away from a domain. It is obvious that complexity and efficiency trade-off to give the best result.

It is important, here, to notice that no matter which policy is chosen, the algorithm does terminate. This happens because the adaptation may happen smoothly (movement between neighbour Service Curves), but the last Service Curve of each SLA would normally correspond to Best Effort Traffic. This means that if congestion keeps going on, then all traffic will be eventually degraded to Best Effort. In this case, there is a big chance that there is some severe fault in the core network that requires operator intervention.

After the suitable SLA for adaptation is chosen, then some sort of housekeeping must be done in the SLA Brokers. First, the local database of the SLA-B must be updated with the new status of the

adapted SLA. Moreover, accounting and billing information are held for the compensation of the adapted SLA. These are internal procedures. After that, in order for the new policy to be enforced, the edge routers of the domain must be reconfigured. Finally, the peer SLA Brokers must be informed about the adaptation. Since peer SLA Brokers have kept their databases updated and they both have knowledge of the status of their bilateral SLAs, then the only thing that should be communicated between them is the SLA ID and the new service curve of this SLA. This approach reduces highly the communication overhead between neighbouring domains, because of the adaptation and recovery procedures of the ASA architecture.

If adaptation is performed and no congestion signals are received for a defined period of time, then the edge routers start bringing the adapted SLAs back to their original status gradually and maybe one by one. The selection of the adapted SLA that should be recovered depends directly on the policy used in order to decide which SLA will adapt. For example, if the network operators have used the **priority policy,** now they must choose the SLA with the highest priority to recover. If they used the **profitable policy,** then they must now choose the SLA that requires the highest compensation. The **lazy policy** selects SLAs in a FIFO way, while the **fair policy** and the **trendy policy** should work on a LIFO basis. The recovery should happen with smooth movements between the predefined service curves and not in an abrupt manner. Some simulation results about recovery procedures presented in section 7 highly justify this recommendation.

If an adapted SLA tries to recover and congestion effects appear again because the recovery attempt made was unsuccessful, the procedure is abandoned and the edge routers of the domain are coming back to their previous state. Moreover, the specific SLA is marked so that it is not the first one to be chosen when the recovery procedure starts again. Care should be taken to avoid SLA deadlocks due to this measure. An obvious solution is to associate a timer with these SLAs and when this timer expires then the SLAs can be chosen for recovery again.

As long as the recovery is confirmed, the SLA Broker should update its billing system and confirm the change to the peer SLA Brokers. The peer SLA brokers may then also choose to trigger their own recovery procedures since they now have some higher quality assurances from their peer domains. This recovery may well propagate all the way back to a single traffic source or application.

6. SIGNALLING ISSUES

Different signalling requirements apply to various parts of the ASA architecture. In order to make the analysis of the candidate solutions clearer three main signalling interfaces are defined. The first one is between the SLA Broker and the edge routers, while the second one is between peer entities of the ASA architecture. That captures the cases of SLA Broker to SLA Broker and edge router to edge router signalling, respectively. The third interface is between the core and the edge routers. Figure 13 describes in short these three interfaces and the arrows between the different entities of the ASA domains represent the most appropriate signalling

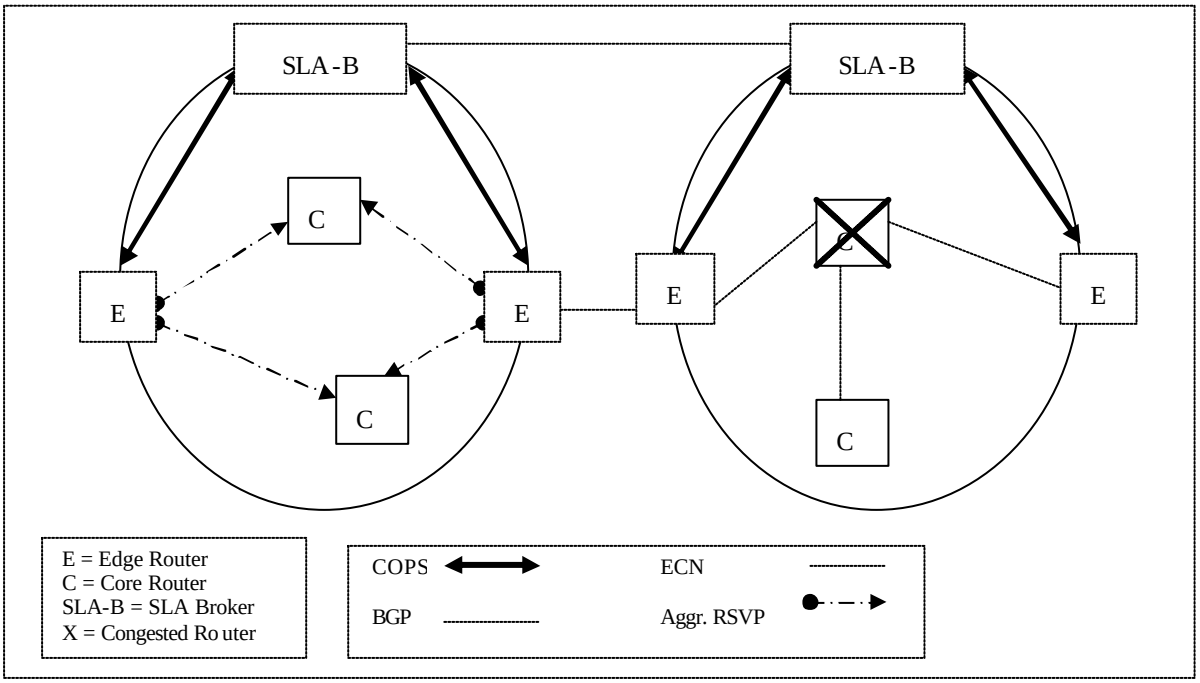

Figure 13. ASA Signaling

solutions as described in the rest of this section.

The SLA Broker is the PDP (Policy Decision Point), while the edge routers are the PEPs (Policy Enforcement Points) of the ASA architecture. Based on this observation, the candidate signalling solution should be appropriate for a Client-Server communication model between the different entities. Moreover, it should be well structured and easily extensible in order to match any particularities of the ASA architecture. Finally, it should impose low communication overhead within a domain so as not to eat up many of the system resources. The COPS (Common Object Policy Service) protocol as described in [Durham et al, 2000] has all the desired properties so it is a strong candidate solution.

The edge router, upon receiving a request for establishing DiffServ connectivity from the peer domains, issues a **COPS REQ** message to the SLA Broker, with the specific request parameters and any other information needed. The SLA Broker examines the available resources, may run some sort of profitability algorithm and, based on the policy loaded by the administrator and the information held in the local database, issues a **COPS DEC (Decision)** message to the edge router. The router receives the COPS DEC message (it maybe be accompanied with some sort of configuration file) and then issues a **COPS RPT (Report)** message. It must be noted here that the PDP (SLA Broker) is the only and ultimate authoritative entity of the domain. This means that the edge routers report only on the reception of the configuration file. The acceptability of the new configuration is taken for granted. The same sort of message sequence is used in any other case the edge router needs to communicate with the SLA Broker.

Within the ASA framework, signalling between peer entities is mainly used for exchanging routing and QoS information, advertising possible bids and keeping the databases of peer SLA Brokers in a consistent state. Instead of having different signalling protocols for exchanging these different types of information, it would be better to combine all these different requirements under a single signalling solution in order to keep the implementation and administration overhead to a minimum. Since BGP-4 is already used for exchanging routing information between peer domains, it only needs to be slightly extended so as to be able to accommodate ASA related information. A way to do this is proposed by O. Bonaventure in [Bonaventure, 2001]. According to this approach a flexible QoS attribute can be used in order to associate QoS information with a **BGP UPDATE** message. Eight bits are used for the identification of the type of QoS information that can be exchanged through the QoS attribute. This leads to 256 different QoS types, which is enough for all the ASA related information, like bid prices, the current state of an SLA or the new value of a database entry.

Intra-domain signalling is used in order to support two basic operations of the ASA architecture. The first one is the notification of the edge routers for any congestion effects in the core domain. The second one is associated with the reservation of resources in the core nodes of the DiffServ domain in order to support the different SLAs negotiated at the edges. An explicit congestion notification mechanism is described in detail in section 4.1. For resource reservation at the core routers, any of the current existing solutions may be used. In particular, Aggregated RSVP [Baker et al, 2001] may be a good signalling solution at the intra-domain level as it takes into consideration the aggregated nature of the DiffServ approach and reduces the amount of soft states that have to be held in the domain routers when compared to the original RSVP solution.

7. SOME SIMULATION RESULTS

Referring to the congestion scenarios presented in sections 2.3 and 2.4 we have performed some early simulations using the NS2 simulator in order to study the effectiveness of adaptation of traffic aggregates. The results are really promising and show that if congestion is to be considered as part of the overall semantics of any QoS architecture then the method of segmented adaptation of traffic aggregates could become a reliable solution for congestion control.

7.1 Adaptation For Congestion Scenario 2.3

In this case congestion effects occur in the link C2r1-E2r3 of Figure 7 for a time period from 10sec to 20sec (see Figure 14). Assuming that upon the occurrence of congestion the ASA Engine of the edge router of AS2 is triggered and the traffic offered to AS2 from AS4 is adapted from about 360 packets/sec to 240 packets/sec. The results of this adaptation are almost immediate and are presented in Figure 15. The C2r1-E2r3 link recovers from congestion and a small number of packet drops that occur after the adaptation can be explained from the way the RED queue of the C2r1 router works. In fact, what happens here is that some packets already queued before the adaptation procedure are dropped some time later. The number of packets dropped after adaptation is very small and not capable of triggering another adaptation procedure.

Another interesting result coming from this simulation is that while during congestion the FTP source of AS1 is almost completely starved after adaptation it gains a generous portion of its original bandwidth. This shows that ASA architecture also reduces the impact of congestion to the other traffic sources of the system. Figures 14 and 15 present the results of congestion and recovery respectively.

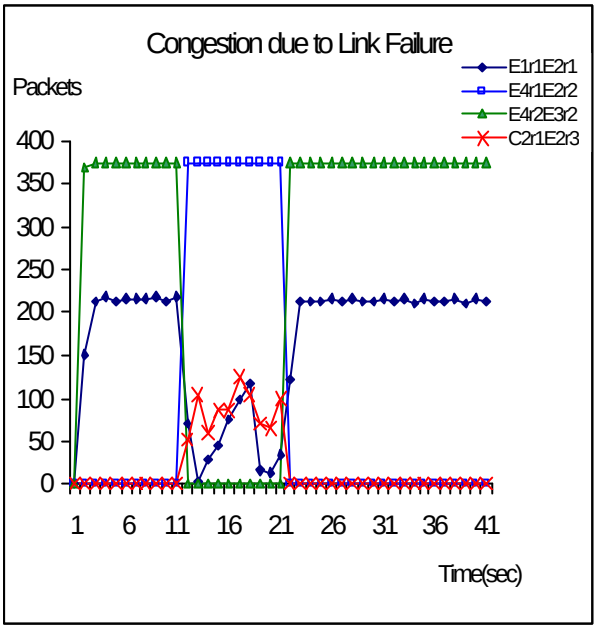

Figure 14. Congestion Graph for Scenario 2.3

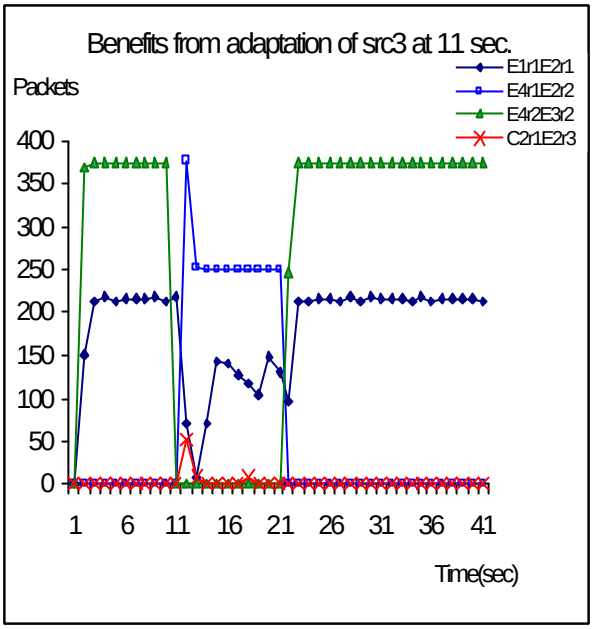

Figure 15. Adaptation Graph for Scenario 2.3

7.2 Adaptation For Congestion Scenario 2.4 And Some Recovery Considerations

In this scenario the C3r1-C3r2 link of AS3 is slightly overbooked so when src1, src2 and src3 try to give their maximum traffic burst simultaneously congestion occurs. Refer to section 2.4 for more information about this congestion scenario.

The adaptation is realised by cutting down the traffic offered by S2 to the network from 240 packets/sec to 120 packets/sec. This sort of adaptation has an immediate impact on the system, as the link C3r1-C3r2 does not suffer from congestion any more, and moreover, the traffic offered by src1 and src2 has gained back a generous portion of the initial value. Figures 16 and 17 show the exact results produced by the NS2 simulator.

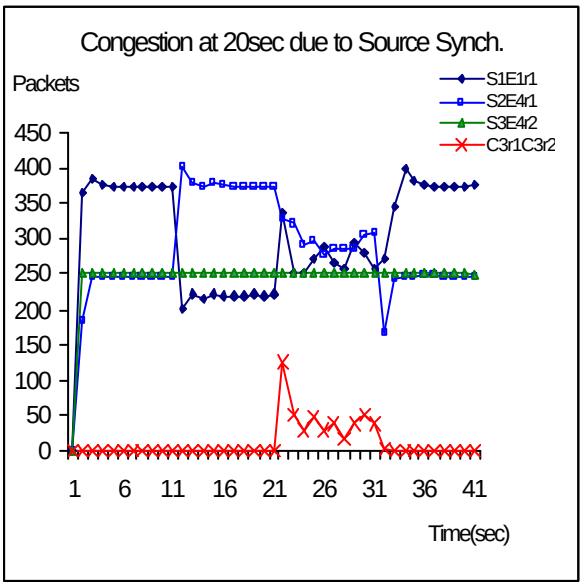

Figure 16. Congestion Graph for Scenario 2.4

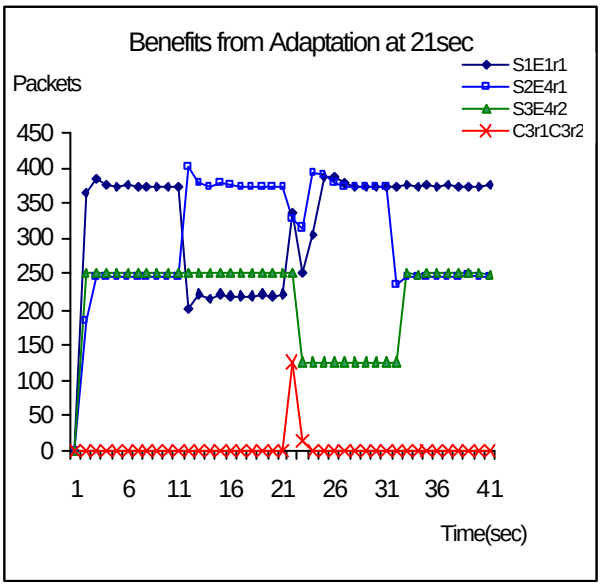

Figure 17. Adaptation Graph for Scenario 2.4

In this scenario we move a step further and show the impact of an unsuccessful, abrupt recovery attempt of the adapted traffic aggregate coming from src2 at time 26sec. The results presented in Figure 18 are really interesting and show that special care is needed when deciding to start a recovery procedure because a recovery failure penalises heavily the rest of the traffic sources. In this example, while the edge Router of AS3 is supposed to detect the congestion almost immediately and forces the recovered SLA back to its adapted status, the impact on the traffic sources 1 and 2 is really heavy and of broader timescales. For example, while the unsuccessful recovery procedure lasts about 0.3secs the affected traffic sources need more than 3secs to come back to their original status.

This result suggests that a TCP-friendly way of recovery with a slow start technique may well be suited for our proposal. In this way backward compatibility with the TCP-based congestion avoidance algorithm could be established not only in the way domains back-off, through the smooth movement between the predefined service curves, but also in the way recovery procedures take place after adaptation.

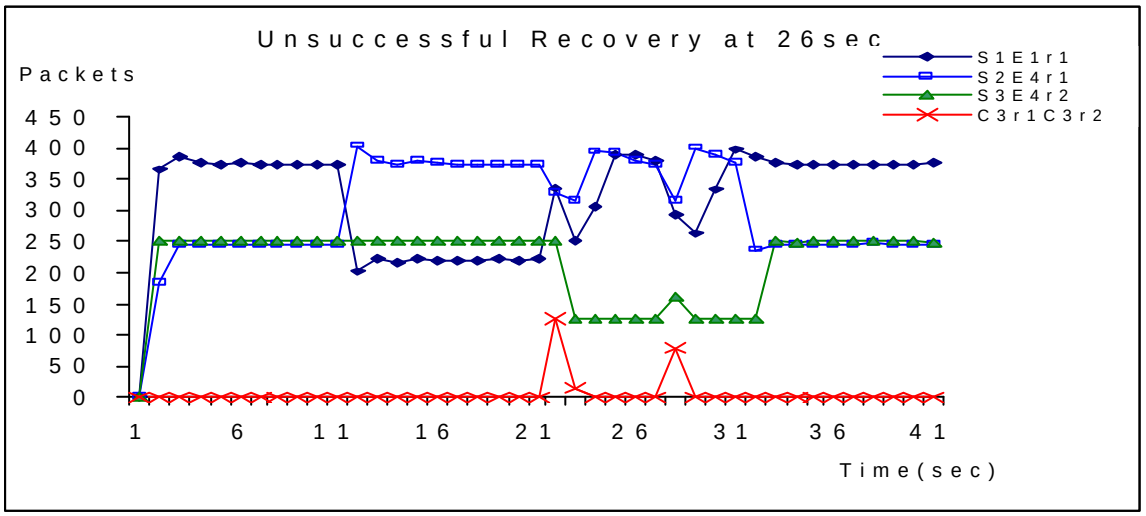

Figure 18. Unsuccessful Recovery Operation

8. CONCLUSIONS AND FUTURE WORK

In this paper we have presented a series of simulations showing that under the DiffServ framework congestion is still possible under certain conditions. This means that DiffServ could be characterised as a defective QoS Architecture. This is not a real drawback for The DiffServ approach, as long as congestion effects are not ignored but are taken systematically into account when designing QoS solutions. Thus, we believe that even if congestion effects are rare and not very intense, they are still present; so, they must be faced as a possible error to the delivered QoS. This means that congestion should be considered as part of the overall semantics of any QoS architecture.

Elasticity and co-operation among service providers may prove vital for the success of the DiffServ model, in analogy to the viability of the best-effort model that is based on the elasticity and TCP-friendliness of co-operating applications. According to this approach an Active Segmented Adaptation (ASA) [DeMeer and O'Halon, 2001] mechanism has been introduced as part of any QoS architecture. The basic operational requirements and the implementation issues of the ASA architecture have been explained in some detail. The service curve concept has been introduced for the definition of different QoS levels inside each SLA. This means that adaptation and recovery procedures as proposed by ASA can now be realised as smooth movement between these predefined service curves.

Moreover, in each domain there are tasks of longer timescales than the ones executed at the router level. These tasks are normally related to issues like domain housekeeping, trading of resources, pricing of clients, monitoring of the QoS mechanisms used and performance optimisation. For this reason a domain management entity, called SLA Broker, has been introduced as part of the overall ASA architecture. The structure, operation and algorithms of the SLA Broker as well as the signalling between the SLA Broker, the residing and peer domains have been defined in detail. Domain management entities like Bandwidth or SLA Brokers are expected to play an important role in the next generation data networks. In particular,

they will serve the self-organisation of IP domains, the monitoring and the adaptation of the QoS guarantees offered to peering domains and the assurance of an economically viable model based on the trading of domain resources.

Some early simulation results presented in this paper are really promising and show the effectiveness of ASA architecture in congested DiffServ domains, as well as the possible problems associated with unsuccessful recovery procedures. We believe that adaptation and recovery should be done in such a way that would establish the TCP-friendliness of our proposal in accordance to the TCP congestion avoidance scheme.

While a solid framework for the ASA architecture has been defined in this paper, there is still big area for research especially in the algorithms associated with selecting the appropriate SLA for adaptation, as well as in the recovery algorithms. These two areas and the signalling issues of the ASA architecture are the main points on which we are going to focus our research efforts and try to come up with some well justified answers in the near future.

ACKNOWLEDGEMENTS

Mr. S. Retzekas would like to thank the "Bodossaki Foundation" for funding his postgraduate studies, part of which is this research. This research has been funded in part by the Alpine grant of BT Exact.

Both authors would like to also thank Mr M. Amine Houyou for essential discussions, prove reading and editing this paper.

REFERENCES

Baker et al, 2001, "Aggregation of RSVP for IPv4 and IPv6 Reservations", RFC 3168, F. Baker et al., (September), www.ietf.org/rfc/rfc3168.txt

Bonaventure O. FUNDP, 2001, Internet Engineering Task Force. Internet Draft, "Using BGP to distribute flexible QoS information" (February). http://www.infonet.fundp.ac.be/doc/reports/draft-bonaventure-bgp-qos-00.txt

Class of Service, 2001, "Mechanisms to Communicate and Deliver Class of Service in a Packet Network" Tenor Networks WhitePaper, February 2001, http://www.tenornetworks.com/whitepapers/classofservice.html

Crowcroft J., de Meer H., Karlberg K., 2001 "An Architecture for Bi-Level Active Networking", , Opensig, London, http://www.cs.ucl.ac.uk/staff/J.Crowcroft/talks/Bi-Level_AN_files/

Cruz R. L., 1991, "A calculus for network delay, part I: Network elements in isolation", (January), *IEEE Trans. Inform. Theory*, vol. 37, pp. 114–131

De Meer H. and O'Halon P., 2001, "Segmented Adaptation of Traffic Aggregates". IWQOS 2001, June 6-8, in Karlsruhe, Germany.

De Meer H. et al, 2002, "Analysis of Existing QoS Solutions", Internet Draft, (May), http://search.ietf.org/internet-drafts/draft-demeer-nsis-analysis-01.txt

Durham D. et al, 2000, "The COPS Protocol", [RFC 2748], (January), http://www.ietf.org/rfc/rfc2748.txt

Flegkas P., Trimintzios P., Pavlou G., Andrikopoulos I. and Cavalcanti C.F., 2001, "Policy-based Extensible Hierarchical Network Management", , Proceedings of Workshop on Policies for Distributed Systems and Networks (Policy January 2001), Spinger-Verlang LNCS-1995, Bristol, UK.

Floyd S., Jacobson V., and Berkeley L., 1993, "Random Early Detection Gateways for Congestion Avoidance" Laboratory. University of California. (August), IEEE/ACM Transactions on Networking. http://www.icir.org/floyd/papers/red/red.html

Frang W. et al, 2000, "A Time Sliding Window Three Colour Marker (TSW3CM)", RFC 2859, (June), www.ietf.org/rfc/rfc2859.txt

Frankhauser G. and Plattner B., 1999, "Bandwidth Brokers as Mini-Markets", (December), MIT Workshop on Internet Service Quality Economics

Frankhauser G., Schweikert D and Plattner B.., 1999, Service Level Agreement Trading for the Differentiated Services Architecture. Computer Engineering and Networks Lab. Swiss Federal Institute of Technology, Zurich, (May). http://www.tik.ee.ethz.ch/~gfa/paper/TR59.pdf

Fulp E. W. and Reeves D. S., 2001, "Optimal Provisioning and Pricing of Differentiated Services Using QoS Class Promotion", (September), Proceedings of the INFORMATIK 2001, Workshop on Advanced Internet Charging and QoS Technology (ICQT'01), http://arqos.csc.ncsu.edu/papers/2001-09-icqt01-diffserv.pdf

Ghosh A., Fry M. and Crowcroft J., 2000 "An Architecture for Application Layer routing in Active Networks", LNCS 1942, H.Yasuda, Ed. 2000, pp.71-86, Springer

Goderis D., 2001, "Service Level Specification Semantics, Parameters and negotiation requirements", IETF Internet Draft TEQUILA Consortium. (June), http:// www.ietf.org/internet-drafts/draft-tequila-sls-01.txt

Hashem. E, 1989, "Analysis of Random Drop for gateway congestion control", Report LCS TR-465, Laboratory of Computer Science, MIT Cambridge MA, p.103.

Heinanen et al, 1999, "Assured Forwarding PHB Group", RFC 2597, (June), http://www.ietf.org/rfc/rfc2597.txt

Jain R. and Ramakrishnan K.K, 1988, "Congestion Avoidance in Computer Networks with a Connectionless Network Layer: Concepts, Goal & Methodology", (April), Proc. IEEE Comp. Networking Symposium, pp.134-143

Le Boudec J. Y., 1998, "Application of Network Calculus to Guaranteed Service Networks", IEEE transactions on information theory, (May), vol. 44, no. 3,pp. 1087-1096

Le Faucheur F. et al, 2001, "Requirements for support of DiffServ aware MPLS Traffic Engineering", (November), Internet Draft Work in Progress, http://www.ietf.org/proceedings/01dec/I-D/draft-ietf-tewg-diff-te-reqts-02.txt

Makris J., 1999, "Bandwidth Brokers, Not exactly Nasdaq", Data Communications Magazine, (May), http://www.data.com/issue/990507/brokers.html

NS2, The Network Simulator NS2, http://www.isi.edu/nsnam/ns

Nichols K. and Carpenter B., 2001, "Definition of Differentiated Services, Per Domain Behaviours and Rules for their specification", Nichols K. and B. Carpenter, Internet Draft, (January), http://www.ietf.org/internet-drafts/draft-ietf-diffserv-pdb-def-03.txt.

Ramakrishnan K. et al, 2001, "The Addition of Explicit Congestion Notification (ECN) to IP", [RFC 3168], (September), www.ietf.org/rfc/rfc3168.txt

Raymont L and Srihari R, "DiffServ and MPLS. Concepts and Simulation", Department of Electrical and Computer Engineering, Virginia Polytechnic Institute and State University. http://csgrad.cs.vt.edu/~sraghava/qos/dsmpls.pdf

BIOGRAPHY

Hermann de Meer has led several national and international projects in modelling and performance evaluation, data communications, and quality of service. He has been an Assistant Professor at Hamburg University, Germany, a Visiting Professor at Columbia University in New York, USA, a Research Fellow of the Deutsche Forschungsgemeinschaft (DFG) and a Reader at University College London. He is currently appointed as a Professor at the University of Passau, Germany. Prof. Hermann de Meer is co-authoring a book on Queuing Networks and Markov Chains - Modelling and Performance Evaluation with Computer Science Applications published by John Wiley in 1998. His research interests include Performance Modelling and Simulation of Computer and Communication Systems, Quality of Service, Internet Protocols, Peer-to-Peer Networks and Home Networking.

Spyridon Retzekas: has been a graduate student at UCL (University College London). He has currently successfully completed his masters program.