Defective QoS Architectures and Inter-Domain Co-Operation

Spyridon Retzekas and Hermann de Meer

Department of Electrical and Electronic Engineering, University College London

Abstract: 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 needed in order to monitor the adaptation/recovery procedures and the processes associated with resource trading and billing. The structure and operation of the Service Level Agreement Broker and the signalling issues related to the ASA architecture are examined in this paper.

1. Introduction

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 [1]. 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 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 other 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, network operators may not always be able to guarantee the proper resource provisioning assumed by DiffServ, as this approach would probably lead to a nonviable economic model for their business. Resource overbooking is likely to remain a common practise in the near future as stated in sources both from academia [2] and industry [3].

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 rules at the edges and proper configuration of the core nodes according to some well-defined Per Hop Behaviours (PHB) [4]. In general, any of the disturbances mentioned in the previous paragraph could lead to congestion. Congestion can be seen as the possible cause to **"fault" in delivering a networking service with a required QoS**. Such faults within a single administrative domain are referred to here as **QoS faults in a network segment**. The desired goal of providing end-to-end QoS is sought to be achieved by combining many Per Domain Behaviours, while provisioning and conditioning rules are part of a SLA [5] between the two operators. This means that the probability of having a violation in the agreed end-to-end QoS, referred to here as **end-to-end QoS fault,** may even be much higher than the probability of a single segment QoS fault.

Since congestion appears likely to be an occasional but unavoidable matter-of-fact characteristic under the DiffServ framework, 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 how 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 QoS faults are completely ignored then a defective QoS architecture may prove inefficient and incapable of offering predefined QoS guarantees to the network operators.

The rest of the paper is organised as follows. Section 2 presents the basic outlines of the ASA Architecture as introduced in [1]. Section 3 gives a brief overview of the basic operations and the structure of a management entity used within the ASA architecture called SLA Broker. Some signalling issues are discussed in Section 4 and, finally, conclusions and future work are described in section 5.

2. The ASA Architecture

The arguments presented in Section 1 show that some sort of co-operation is needed in order to experience end-to-end QoS and to reduce the impact of congestion effects to a minimum. 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 can not 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 can not 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 [1] 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** [6],[7] can help the definition of different QoS level 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"[1] 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 level 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.

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 described in more detail in [1].

The edge routers are responsible for the adaptation procedure in case of congestion and for the recovery of the adapted SLAs when congestion effects disappear. Some active queuing mechanism, like RED [8], is assumed to be used in the core routers. Upon exceeding some utilisation thresholds the RED queue marks some packets. Setting the ECN bits [9] at the IP header can do the marking. The marked packets are forwarded to the edge routers. This can be done, for example, through broadcasting in order to ensure that the congestion notification reaches the suitable edge router. The edge routers are responsible for notifying the SLA Broker about the congestion.

When the percentage of marked packets reaches a threshold, edge routers may start an adaptation procedure, which is realised as smooth movement between the predefined service curves inside each SLA. How peer operators will react to the adaptation depends on the local policy. It is possible either to use local techniques to mask the fault or to propagate congestion signals across peering domains, possibly all the way back to the access domains. In this case, each local policy router should have its own rules and policies for the non-elastic applications. If no congestion occurs for a reasonable period of time then the recovery procedure starts in order to bring the adapted 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 adapted SLA.

The current implementation of the ASA architecture is based on the ALAN [10] and the bi-level Active Networking [11] approach to allow for flexibility and re-programming of policies.

3. Domain Management and SLA Brokers.

Edge routers are responsible for the adaptation and recovery procedures as described in Section 2. But in a dynamic DiffServ environment there are many operations associated with the domain management and the dynamic trading of resources between the peer domains. Moreover, some sort of monitoring mechanism is needed for the adaptation and recovery procedures. The term of the SLA Broker is introduced under the ASA architecture, as a domain management entity, responsible for 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 Points – **PEP**), the peer SLA Brokers and with the domain administrator, who is responsible for

loading the current policies, through a management console. The block structure of the SLA Broker and its internal engines are presented in Figure 1.

The SLA-B database keeps information about the properties of the established links between peer domains. Since SLAs are exchanged between peer domains the database must be kept updated. In this way the SLA Broker has always a clear view of the type of traffic that is serviced through its domain and the way the resources are allocated at the edge and core routers with respect to each supported traffic class.

The bid advertiser combines information from the SLA database (domain resources and current utilisation) in order to advertise a series of possible bids to the peer domains. The SLA Trader takes decision on the acceptability of a 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 may be kept in the database. The SLA Broker is also responsible for accepting or rejecting bid offers from other domains. The decision depends directly on the outcome of the profitability analysis algorithm used [12]. The analysis of the various possible trading engines and profitability algorithms is far beyond the scope of this paper.

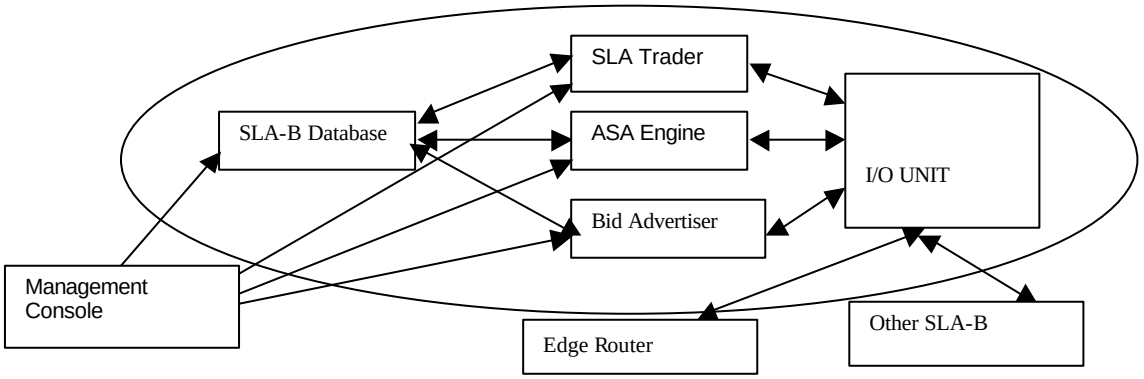

Figure 1. Structure of the SLA Broker

The ASA engine is responsible for monitoring the adaptation/recovery procedures and downloading adaptation policies to the edge router during the SLA set-up phase. There is a variety of policies that can be downloaded to the edge routers to help them decide which SLAs will be chosen for adaptation upon the reception of a congestion signal. The chosen policy is a trade-off between simplicity, low protocol traffic overhead, profitability and effectiveness of the segmented adaptation procedure. For example, algorithms may only care about low signalling overhead and thus upon congestion notification move all SLAs to another service curve (**lazy policy**), while other approaches care only for the profitability (**profitable policy**) or the fairness (**fair policy**) [12]. Usually the most effective methods are these who combine two or more of the above policies. For example the **trendy policy** receives information about the source of congestion (high signalling overhead) and tries to make the fairest choice at the lowest compensation cost thus it combines the fair and profitable policy. Whatever policy is chosen the same should be used for selecting the SLAs that will be recovered first when congestion effects are cured. The ASA engine is also responsible for pricing the SLAs according to the adaptation occurring from time to time, holding information about past adaptation patterns and inform the peer SLA Brokers about the status of the bilateral SLAs mainly for consistency reasons.

4. Signalling Issues

Different signalling requirements apply to various parts of the ASA architecture. In order to make the analysis of the candidate solutions clearer the three main signalling interfaces are explained. 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. Due to space limitations we restrict our discussion to the first two interfaces.

As mentioned in Section 3, the SLA Broker is the PDP (Policy Decision Point) while the edge routers are the PEP (Policy Enforcement Points) of the architecture. The **COPS** (Common Open Policy Service) protocol [13] can be used for signalling between these two entities as it is simple, well structured and imposes low communication overhead within a domain. 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 stored in the local database, issues a COPS DEC (Decision) message to the edge router. The router receives the COPS

DEC message (accompanied with some configuration file) and then issues a COPS RPT (Report) message as a confirmation.

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. The goal is to combine all these different requirements under a single signalling solution in order to keep the implementation and administration overhead to a minimum. A way to do this is proposed by O. Bonaventure in [14]. According to this approach a flexible QoS attribute can be used in order to associate QoS information with a **BGP UPDATE** message. The extension of BGP-4 as proposed in [14] caters only for the exchange of QoS information. In order for Bonaventure's proposal to match the requirements of the ASA architecture, it will be extended so that it will be able to incorporate ASA-specific information, like bid prices, the current state of a SLA or the new value of a database entry.

5. Conclusions and Future Work

This paper presents the basic philosophy behind the ASA architecture, which is primarily based on the co-operation between peer domains and the introduction of service curves inside each bilateral SLA in order to cater for congestion effects under the DiffServ framework. 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 [1].

A new network entity called SLA Broker is introduced in this paper. SLA Brokers act as domain management modules and are responsible for a series of operations, mainly associated with providing DiffServ connectivity between peer domains, pricing of services, domain housekeeping and monitoring of the adaptation and recovery procedures happening in the edge routers. Moreover, they are responsible for re-configuring the edge routers with new adaptation policies when the existing ones are no longer effective. Finally, we presented some remarks on signalling issues associated with the ASA Architecture.

In the near future we plan to run some extensive simulations in order to validate our architecture with respect to the effectiveness of the elasticity between network operators.

References

[1] "Segmented Adaptation of Traffic Aggregates". Hermann de Meer, Piers O'Halon. IWQOS 2001, June 6-8, 2001 in Karlsruhe, Germany.

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

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

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

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

[6] "A calculus for network delay, part I: Network elements in isolation", R. L. Cruz, IEEE Trans. Inform. Theory, vol. 37, pp. 114–131, Jan. 1991

[7] "Application of Network Calculus to Guaranteed Service Networks", Jean-Yves Le Boudec, IEEE transactions on information theory, vol. 44, no. 3,pp. 1087-1096, May 1998

[8] "Random Early Detection Gateways for Congestion Avoidance" Sally Floyd and Van Jacobson. Lawrence Berkeley Laboratory. University of California. August 1993 IEEE/ACM Transactions on Networking. http://www.icir.org/floyd/papers/red/red.html

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

[10] " An architecture for application layer routing in Active Networks", A.Ghosh, M.Fry and J.Crowcroft, LNCS 1942, H.Yasuda, Ed. 2000, pp.71-86, Springer

[11] "An Architecture for Bi-Level Active Networking", J. Crowcroft, H. de Meer, Ken Karlberg, Opensig 2001, London http:// http://www.cs.ucl.ac.uk/staff/J.Crowcroft/talks/Bi-Level_AN_files/v3_document.htm

[12] "Service Level Agreement Trading for the Differentiated Services Architecture". G. Fankhauser, David Schweikert, Bernhard Plattner. Computer Engineering and Networks Lab. Swiss Federal Institute of Technology, Zurich, May 1999. http://www.tik.ee.ethz.ch/~gfa/paper/TR59.pdf

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

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