QoS ad hoc Internetworking: Dynamic Adaptation of Differentiated Services Boundaries[1]

Michael Smirnov

GMD FOKUS, Kaiserin-Augusta-Allee, 31
tel. +49 30 34637113
fax.: +49 30 34638000
E-mail: smirnow@fokus.gmd.de

Abstract. The Internet needs QoS internetworking to become QoS aware in the end-to-end sense. We address only one issue of this - adaptation of the differentiated services architecture, in particular, dynamic creation of DiffServ virtual boundaries. Our solution is fully distributed and data driven, therefore it could be considered as QoS ad hoc internetworking on contrary to statically configured DiffServ. The proposal is two fold. First, to support invariance under aggregation we suggest to maintain Per Domain Behaviours (PDB) based on Per Path Behaviours, and, second, to use group communication based on native IP multicast for needed QoS signalling and resource control. The paper shows that due to flexible grouping policies the approach has high scalability and good deployment potential.

1. Introduction

The motivation to add dynamic adaptation to the DiffServ is many fold. First, DiffServ as a QoS enforcing technology is only needed in the Internet when and where there is a congestion. Then, statically configured DiffServ routers will loose buffering capacity in the absence of priority traffic. This buffering capacity could be used to accommodate more packets when unavoidable bursts occur. Last but not least: in the absence of any congestion, packet classification, metering, marking, shaping and policing performed by DiffServ edge router will waste router resources. The latter was shown by multiple measurements by TF-TANT: *"... the minimum nodal delay added by the DiffServ router ..., is equal to the transmission time of 2 BE packets*" [1]. We interpret this result in favour of best effort service for transmission of flows even with various priorities being set if there is no congestion. Following these motivations the ideal case would be to have DiffServ in a router up but not running until a congestion occur.

The paper is structured as follows. First, we shortly summarise known DiffServ issues, then present our requirements and framework. One of the main requirements is

[1] Research outlined in this paper is partly funded by the IST project CADENUS "Creation and Deployment of End-User Services in Premium IP Networks" http:// www.cadenus.org/

S. Rao and K.I. Sletta (Eds.): INTERWORKING 2000, LNCS 1938, pp. 186-195, 2000.

to support dynamic service creation over DiffServ networks, i.e. to be able to maintain for particular microflows of IP datagrams the so called invariance under aggregation. Practically that means, that terms and conditions defined in a service level agreement (SLA) for this microflow are to be met Internet wide while the micro flow itself becomes anonymous within a DiffServ bahaviour aggregate. This major requirement is addressed then in the main part of the paper presenting a new paradigm for resource control in a distributed environment allowing dynamic creation of services. We argue that only when based completely on a group communication paradigm a resource control framework will meet this requirement. A short introduction of a particular mediated group communication service applicable in this case is followed by a number of brief examples. The paper is concluded by a summary of QoS ad hoc internetworking.

2. DiffServ Issues

An early explanation of the DiffServ stated: "*Code-points should be looked at as an index into a table of packet forwarding treatments at each router*" [2]. In the naive understanding this might mean that differently marked packets can actually take different routes within the network. Not excluding this possibility, the paper focuses mainly on a traditional differentiation of packet forwarding paths inside a router.

It is well understood currently that DiffServ as such is not a service but a building block. Services will be built by adding rules to behaviours, such as:

- Rules for initial packet marking,
- Rules for how particular aggregates are treated at boundaries,
- Rules for temporal behaviour of aggregates at boundaries.

The latter aspect can be addressed e.g. with regard to treatment of out-of-profile portion of the traffic from a particular class. Below we concentrate on the common denominator for all the behaviour rules – resource control, and show how to achieve efficient and scalable resource control in a DiffServ friendly manner, as well as being also conformant to SLAs.

3. Requirements and Framework

Services built on top of DiffServ should have desired end to end QoS properties. There is a need to have standard common understanding of how these services should be constructed, e.g. which rules should be applied at which points. While this work probably will be started soon by the IETF, this paper proposes another framework, outlining how a set of rules defining services should be created, configured and controlled in a DiffServ domain, and between domains, while meeting three main requirements oultined below.

The Internet mainly needs QoS enforcement when and where a congestion occurs. While congestion is always dynamic the first requirement is dynamic reaction to

congestion situations. This reaction should take part of dynamic QoS enforcement protecting high priority flows in the congested area.

The enforcement needs to be scalable, or/and *evolveable* [3], which is our second requirement. The latter assumes systemic nature of the framework facilitating and supporting many services. We argue that operating with groups rather than with individual flows or flow aggregates provides both scalability and evolveability.

Lastly, QoS Enforcement should conform to end-to-end requirements, i.e. preserve invariance under aggregation [4].

We distinguish a general framework which is actually a service creation framework – a structured set of user and network service components and associated operations at the boundary of the DiffServ domain - and QoS specific framework outlining the QoS inter-working between network entities participating in the service creation.

These entities, defined through the business model, are: user (customer), wholesale (reflected e.g. by TCS at DiffServ meter) and retail (reflected e.g. by SLA mapped to TCS and SLS) network providers and subsequent interfaces at the service creation level[2].

4. The New Paradigm for Network Resource Control

Resource control problems in IP networks are traditionally solved either in a Network provider domain by means of network management and/or over-provisioning, or by means of dedicated protocols using the same IP infrastructure as data flows carrying network payload. This paradigm does not scale when, e.g. dynamic service creation is required. Flow differentiation or simply network QoS is regarded here as an example of network services which is to be created dynamically, triggered either by a flow itself or by a combination of a flow and network conditions, e.g. congestion.

In a very generic form network resource control takes form of decison making supported by two operations: *GET(information on resource usage)* and *SET(parameters of resource usage)*. We may think of this two generic operations as of distinct interfaces from a control entity to a network.

Further, the decision making can be represented as a set of rules which usually take form of pairs (conditions, actions). If all of the required conditions are true, then the pre-defined set of actions is enforced. After any action is taken, however, a new condition, or a set of conditions are taking place. If resource control rules are to be combined in a sequence, then some care should be taken to avoid unwanted dependencies between a given set of conditions after an action (post-conditions) and pre-conditions of a next action in a Resource Control Sequence (RCS).

Fig. 1 represents a case when a resource control decision procedure is given as an RCS of three rules. Each rule is defined as a set of pre-conditions and an action to be taken. Some of the pre-conditions are those obtained from the *GET* interface and reflect particular aspect of the network state, other conditions can be internal to the

[2] The general architecture of the Cadenus framework is to appear as “Cadenus Framework” white paper at URL http://www.cadenus.org

decision making itself. These internal conditions may be timers' state and other conditions configuring the complete resource control sequence in such a way that there are no unwanted dependencies; such an RCS is referred to as *safe*. Similarly, actions defined in an RCS can be those invoked at the *SET* interface, or internal actions setting internal pre-conditions to guarantee safeness of the resource control.

Resource control sequence:
{(IF c1, c2, c3 THEN a1), (IF c4, c5 THEN a2), (IF c6 THEN a3)}

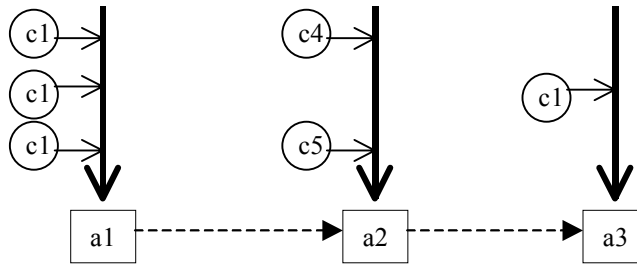

Fig. 1. Example of a simple resource Control Sequence

Safe configuration of conditions in a sequence of rules can be done for a *limited* set of rules and, more important, for a *static* set of rules. Usually safeness is guaranteed by some additional modelling of rules interaction, which is done off-line when an RCS is designed. In case of IP networks deploying dedicated resource control protocols it leads to an isolation of these protocols. When isolated, each protocol is safe by itself, however it is to be configured separately. Hence, the rule interaction problem is pushed towards the off-line configuration. Clearly, the complexity of multiple configurations combined with the complexity of safeness makes it close to impossible to achieve any level of dynamicity in service creation.

On the other hand, there are multiple attempts to centralise the resource control problem. These are motivated by an obvious benefit of having all resources state information at the disposal of a decision making. Theoretically, it facilitates dynamic creation of services, provided that the resource control latency of a centralised system is bounded acceptably. Centralisation, after all, imposes even more complexity induced by the need to maintain itself, usually by a hierarchy. The hierarchy of *GET* and *SET* interfaces tends to waste network resources on self-maintenance, to introduce additional heterogeneity of network resource control protocols at different levels of the hierarchy. In particular, additional heterogeneity may be wished due to different aggregation levels of data passing through *GET* and *SET* interfaces.

The new paradigm, discussed below, aims at dynamic service creation based on a uniform mechanism, thus providing better scaling properties. Instead of distributing

network state information (pre-conditions) via dedicated distributed but isolated protocols, or via centralised hierarchcical architectures, the new paradigm distributes event notification information via a single mechanism of group communication.

The sequence of actions in Fig. 1 is safe only because it was designed to be safe. Any attempt to change the sequence

$$RCS = \{R1, R2, R3\} \quad (1)$$

to a sequence, say, RCS1 = {R1, R2, R4, R3} should be checked with regard to all possible interactions of rules. Thus, assuming that

R1 = (IF c1, c2, c3 THEN a1);

R2 = (IF c4, c5 THEN a2);

R3 = (IF c6 THEN a3), and

R4 = (IF c7, c8 THEN a4),

we will need to check any potential unsafeness caused by interaction of at least post-conditions of R1, R2 and pre-conditions of R4, that is between a1, a2 and c7, c8, and to check any potential unsafeness caused by interaction of post-conditions of R4 and pre-conditions of R3, that is between a4 and c1. Given that pre-conditions and post-conditions may be complex objects, and safety checks are to be performed on whole ranges of values of state data items, this task is far from being trivial.

What does it mean actually to check safeness between an action and a set of conditions, like between a1 and c7 above? An action ai is unsafe with regard to a condition cj if post-conditions of ai are conflicting with cj. While it is not simple to define the nature of a conflict in a generic way, one distinction - between *global* and *action-local* safeness - helps to imagine possible taxonomy of safenesses. Let us define that ai is *action* ak *locally unsafe* to condition cj if cj is a pre-condition of ak. Then, action ai is globally safe to condition cj is there is no such ak that

1. cj is a pre-condition of ak, and
2. ai is action ak locally unsafe to cj.

Let us further define that *event* is the combination of an action and of all its post-conditions. Each action's post-conditions are at the same time pre-conditions of other actions, possibly subject to relevant aggregation. Defining the notion of event as above, we "freeze" a temporal continuity of actions. Fig. 2 illustrates this concept. Events e1, e2, e3 are defined as

$$\{ai: bi1, bi2, \ldots, bin\}, \quad (2)$$

that is any of the post conditions bi is associated with the parent action, i.e. it is not *anonymous*. An event, as the association of an action and its post-conditions makes sense only

i. until all possible conflicts are checked, and
ii. in the environment which allows dynamic creation of RCSs.

Imagine, that RCS in Fig.2 was generated dynamically, for example by a hot plug-in of R2 between R1 and R3. Then, event e1 is to be checked for safeness to c4 and

c5. While, c4 and c5 are conditions which just appeared in the RCS it may happen e.g., that c4 is just a negation of b1, etc. If e1 is safe to R2, then R2 can be fired[3]. Real RCS is more properly modelled by a branching tree, rather than by a linear sequence.

In this case e1 is safe to all children in RCS tree (e.g. {R.2.1, R2.2, R2.3}) if there are no conflicts between post-conditions of e1 and any of pre-conditions of {R.2.1, R2.2, R2.3}. If a conflict occurs with, say, pre-conditios of R2.2, then the truncated RCS tree, i.e. {R.2.1, R2.3} still can be fired.

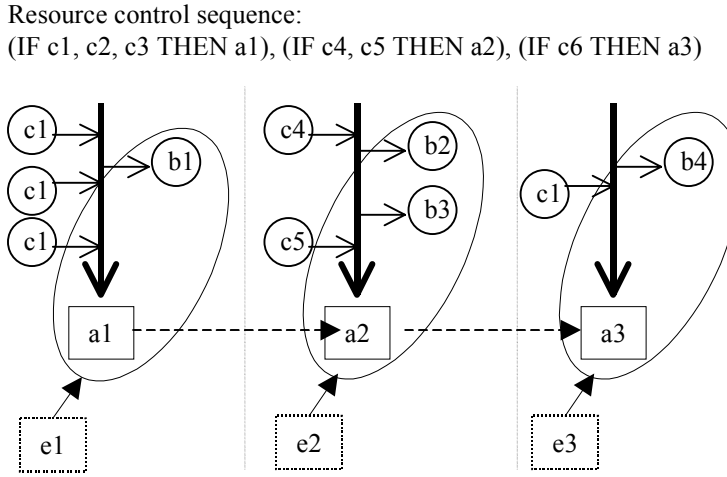

Fig. 2. To the definition of an event

Why not dynamically store all pre- and post-conditions in a single database and run a global safety check over this database before firing any rule? Such an approach is explicitely a non-target for this work. While there are some obvious advantages of a conditions data base approach, it happens to be practically feasible (scalable and efficient) for centralised service creation environments only. If rules are distributed across multiple network nodes, and the latency of the safety check is to be minimal, not to introduce additional delays for time-critical flows in the network, then there is the need for a "hop-by-hop[4]" safety check mechanism.

After all safety checks (against a single set of pre-codnitions in the linear case, and against multiple sets in the case of RCS tree) are performed, the particular event is obsolete. That means, we no longer need to keep an association (**2**); and post-conditions bi1, bi2, ... , bin may become *anonymous*.

[3] The notion of firing is borrowed from Petri Nets.

[4] Here a hop is a single rule in an RCS which, of course can coincide with a physical hop of an IP network.

By so introduced event based safety check, we achieve desired automicity of resource control in a distributed environment allowing dynamic service creation. There is still another problem in such environments. Namely, in order to perform a safety check as described above, each rule needs to know the set of child rules.These child rules are to be notified on the event by a mechanism which will be equally good for a one-to-one or for one-to-many parent-child relations.

5. Group Communication

What is the best event notification mechanism for such an environment? Bearing in mind, that RCS can be a branching tree, we inevitably should conclude that a multicast technology should be very advantageous for event notification. Unfortunately, it is not possible to use native IP multicast as the basis for event notification because it largely lacks controls, mainly for group membership. Service model for IP multicast allows any host to become a group member

There have been many discussions in recent years about practical applicability of IP multicast service model [5]. The lack of providers' control over multicast membership and content has been identified as the main drawback of IP multicast [6]. In order to make multicast more deployable many argued that the service model itself should be changed. An excellent background material– motivation and requirements - for this view could be found in [7].

Recently we have presented another approach [8]. We argue that there is no need to modify IP multicast service model while it should be considered as a building block for higher level services (we call these *Group Communication Services*) which should feature needed controls. Further, to facilitate controls over membership and content we propose to make service components programmable to an extent which is currently feasible. Rather than following active networks paradigm in tailoring network elements to meet user and provider requirements we follow slightly modified client server approach to achieve the same goal.

We introduce a new service model dubbed Mediated Group Communication. In this model we have a new logical entity above IP multicast – Group Communication Mediator (GCM). Main semantics of IP multicast are retained (it's our building block), i.e. we retain group membership as being under the supervision of receivers, we do not consider a sender to a group to be necessarily a member of a group. We build a GCM to be back compatible with native IP multicast [5].

Consider a set of hosts H and a set of groups G in which these hosts participate. We define then a meta group G_H as a union of all the groups $\{g_1, g_2, \ldots\}$ which belong to G and one special group g_0 which consists of one or more GCMs. We show in [8] that all other members of G are sending their control messages to g_0, and g_0 in turn controls group R – e.g. a group of all multicast border routers in a GCM domain.

The criteria for group membership can be derived from the conditions of safety checks, thus each network node, based on its state information and services requested will know which pre-conditions for its required actions it has to check continuously. Based on this, the node joins dynamically a number of *safety groups* which advertise event notifications with relevant conditions. It is interesting to investigate what is the

relation of physical network topology and safety adjacency between safety group members, especially for such services as QoS on demand, e.g. triggered by network congestion.

6. Groups: Engineering View

DiffServ architecture scales because it groups microflows into flow aggregates. Similarly, we propose PDBs to be constructed out of per path behaviours (PPB), which in turn, are path dependant groups of DiffServ per hop behaviours. Basically, the whole framework for SLA based internetworking can be built upon principles of multicast.

There are two obvious motivations for considering multicast as a primary technology to build a framework. The first motivation comes from, so to say, topological prospective – we see that QoS enforcement in the network should include a number of network nodes to collaborate in order to provide a preferential treatment for a flow traversing these nodes. The situation is more complicated when resource reservations are used. In this case, to be reserved resources are to be kept in mind foreseeing future reservations with stringent time and probability guarantees – basically it is an over-provisioning.

The second motivation is the intention to design a QoS provisioning framework with the requirement of flexibility and support of heterogeneous QoS provisioning paradigms and IP network technologies. Such a framework being designed over a group communication paradigm will have an essential feature of hot plug-in. That is a new framework component can be added by just subscribing to a particular multicast group. If the native IP multicast service model is taken as the basis for group communication of the framework components then it is not possible to control the process of joins and leaves. Therefore we plan to investigate into the area similar to mediated group communication applied for a QoS provisioning framework. To support required signalling we propose group communication over native IP multicast using private addressing space. We plan to link this to the DiffRes [9] architecture being currently under simulation.

7. Examples

As one example let us consider a new authentication module plug-in into the framework. In the native IP multicast it will be quite simple for an intruder to inject a misbehaving component by simply joining even a secret IP multicast group address. We need membership controls. This seems to complicate the framework, however in reality it brings an advantage of flexibility at the level of services supported by dynamic creation and configuration.

Another example can be per path behaviours introduced earlier. PPB - a concept allowing inter-domain QoS transaction set-up and end-to-end tariffed transactions. In statically configured DiffServ network groups of micro-flows and flow aggregates

(PPBs and PDBs groups) automatically define groups of edge routers, which control membership for these groups and recover their end-to-end properties. When statically configured edge QoS enforcement (at the boundary of DiffServ domain) can't meet end-to-end requirements for a PPB or PDB group, then we propose dynamic creation of virtual DS boundary (by, e.g. activating DS mechanisms in interior routers following thresholds derived from committed SLAs).

As an example of economic perspective of this approach let us consider the use of airline industry analogy.

An airline company presents a wholesale entity with quasi-static allocation of quality classes, travel agents present a set of retailers with ability to create service dynamically based on service components provided by the wholesale.

When congestion happens at wholesale, then two things are possible : resource reallocation (if Business class is congested the boundary between Business and Economy classes is dynamically adjusted), or/and selective upgrade (if Economy is congested, and there are still available seats in the Business class) based on a relative "value" of an economy passenger(e.g. frequent traveller account state). Both cases are examples of dynamic creation, or re-configuration of a service conformant to an SLA but also utilising resources of an aircraft in an autonomous manner.

8. Conclusions and Future Work

The summary of QoS ad hoc interworking:

- Resources are partitioned dynamically between quality classes in a fully distributed manner,
- Dynamic QoS and charging differentiation are enforced only when a network is experiencing congestion,
- QoS/service creation is facilitated by [re-]enforcing of [virtual] boundaries and distributed resource control for PPB and PDB groups,
- Resource control is facilitated by the event notification mechanism which allows dynamic creation of a service in a fully distributed manner,
- Event notification is based on the notion of safety groups where group membership is dynamic, however controlled.

We plan to evaluate our architecture for a number of end-user services (especially for VPN and IP Telephony based on SIP), to prototype it and to measure actual benefits.

References

1. *Testing of EF in the wide in presence of stream aggregation and congestion*, TF TANT, Quantum project, Feb, 2000, URL: http://www.cnaf.infn.it/~ferrari/tfng/qosmon/ef/wan/wan.html
2. K. Nichols, *Update on the IETF Diffserv Working Group* , NANOG 13 , Detroit, MI, 1998, URL http://www.nanog.org/mtg-9806/ppt/nichols/ tsld006.htm

3. D. Hutchison, *Systemic Quality of Service*, Dagstuhl seminar Quality of Service in Networks and Distributed Systems, May, 2000, URL http:// www.dagstuhl.de/DATA/Seminars/00/
4. K. Nichols, B. Carpenter *Definition of Differentiated Services Behaviour Aggregates and Rules for their Specification*, IETF work in progress, Feb., 2000 URL http://www.ietf.org/internet-drafts/draft-ietf-diffserv-ba-def-01.txt
5. Deering, S., *Host Extensions for IP Multicasting*, IETF, RFC 1112, August, 1989
6. IP Multicast Backgrounder. How IP Multicast alleviates network congestion and paves the way for next-generation network applications, An IP Multicast Initiative White Paper, URL: http://www.ipmulticast.com/community/ whitepapers/backgrounder.html
7. Diot., C., Levine, B.N., Lyles, B., Kassem, H., Balensiefen, D., *Deployment Issues for the IP Multicast Service and Architecture*, - IEEE Network Magazine, Special Issue on Multicasting, Jan./Feb. 2000
8. Smirnov, M., Sanneck, H., Witaszek, D. "*Programmable Group Communication Services over IP Multicast*", Proceedings of the IEEE Conference on High Performance Switching and Routing, June 2000, Heidelberg, IEEE, pp. 281-290.
9. D. Sisalem, S. Krishnamurthy, and S. Dao, "*DiffRes*: *A light weight reservation protocol for the differentiated services environment,"* tech. rep., HRL Laboratories, Malibu, USA, Dec. 1999. Under submission, URL: ftp://ftp.fokus.gmd.de/pub/step/papers/Sisa9912:DiffRes.ps.gz