SIMULATION OF RESOURCE ALLOCATION POLICIES IN A MULTI-DOMAIN IP ENVIRONMENT WITH QoS CONSTRAINTS

MIHAI STANCIU , EUGEN BORCOCI, RADU LUPU

Key words: IP, Quality of service (QoS), Multidomain, Aggregation, Resource allocation.

Next generation Internet should offer support for end to end (E2E) services with quality of services (QoS) guarantees. One method to solve the E2E QoS problem in scalable manner is to establish multi-domain E2E aggregated paths having controlled QoS characteristics. The resources for these pipes should be allocated and installed in each domain by the domain managers. This paper studies several allocation methods, from the point of view of resource utilization degree versus the amount of required signalling when considering a whole chain of domains. The proposed method is currently implemented in a research project.

1. **INTRODUCTION**

At the date of this writing, in the Internet there is no agreed and standardised way to ensure that a End-to-End QoS-aware service can be offered between any 2 endpoints. Several extensive studies have been performed; among these, the European IST projects AQUILA [1], TEQUILA [2], MESCAL [3] and CADENUS [4] have produced theoretical models, followed by the implementation of demonstrators, which greatly contributed at the specifications of concepts and architectural solutions which will eventually lead to a QoS-enabled Internet. Some of these results have been incorporated in the IST project ENTHRONE [5].

A key concept in the QoS terminology is the SLA (*Service Level Agreement*) which is a contract between two parties which specifies the service guarantees; it is translated into its technical counterpart, called the SLS (*Service Level Specification*) containing the actual QoS parameters.

In a multi-domain environment, SLS signalling at the per-flow level is not scalable. One solution is to establish multi-domain E2E QoS enabled aggregated paths. DiffServ technology *bandwidth brokers* (*BB*) have been proposed to control

"Politehnica" University of Bucharest, Electronics, Telecommunications and Information Technology Faculty, 1–3 Bd. Iuliu Maniu, Bucharest, E-mail: ms@elcom.pub.ro

Rev. Roum. Sci. Techn. – Électrotechn. et Énerg., **52**, *1,* p. 81–88, Bucarest, 2007

the domain resources [6]. Generalization of the BB concept leads to *resource domain managers.*

From a topological standpoint, multiple models can be defined [7]: *centralized, hub, cascade*, and *mixed*. It is shown that the most scalable is the cascade model which is adopted in this study.

2. **RESOURCE ALLOCATION POLICIES**

We consider a multi-domain environment structured as the *cascade* model mentioned before (Fig. 1).

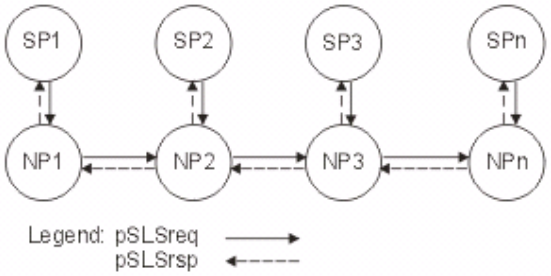

Fig. 1 – Domains and signalling messages.

In each domain there is a *Service Provider* SP and a *Network Provider* NP. The role of the SP in this scenario is to offer services to customers by reserving pipes with QoS guarantees between the customer and a Content Provider located somewhere in the network; the SP itself does not store the content, it only makes it accessible to the customer according to a SLA/SLS. The role of the NP is to provide network services (*e.g.* IP routing) to all the other actors in this scenario. Because of the QoS constraints and the limited bandwidth available in a network, the NP will reserve upon request of a SP a certain amount of bandwidth for a service in a certain QoS class. This bandwidth will be subtracted from the total amount available for that QoS class. Thus, at a certain NP and a given time, it is possible to perform only a limited number of successful reservations, until the bandwidth is exhausted.

Because of the cascade model, a path between a SP (and its customer) and a Content Provider will most likely traverse multiple NPs. A successful end-to-end reservation will be the result of multiple successful reservations at each NP. Any failure at a certain NP in the chain will mean the whole reservation will be unsuccessful.

The reservation process works as follows: a SP issues the *pSLSreq* reservation message (*provider SLS request*) which reaches the associated NP, triggering a certain action which will be detailed below. If this NP is not the intended destination, it will forward the message to its peer NP and so on, all the way to the destination NP. Answer messages are called *pSLSrsp* (*provider SLS response*) and will contain either a positive acknowledgement (ACK) or negative acknowledgement (NAK) indicating the success/failure.

The *pSLSreq* and *pSLSrsp* are called *horizontal* signalling since they travel along the multi-domain chain. At each domain, the reservation consists of a series of *vertical* signalling messages between the local entities of the domain.

We identify 3 *resource allocation/reservation policies* for this multiple-reservations model; in this paper, the words *allocation* and *reservation* can be used interchangeably, since an amount of resources which is reserved or allocated for a certain service is no longer available to other services.

We will detail the 3 policies using the following simplified scenario: in a multi-domain cascade chain, we assume each SP (SP1-SP*n*) wants to establish a pipe to the last NP (NP*n*), where the content provider is located. No other NP than NP*n* will be the target. This is a worst-case scenario.

Policy 1: early allocation. In this policy, the *pSLSreq* determines at each NP a verification (check) of the available resources and, if successful, the allocation of the resources.

If at a certain NP the check is successful, the required amount is reserved, and the total amount is decremented. Following that, the *pSLSreq* message is forwarded to the downstream NP where the process is repeated, until the *pSLSreq* reaches NP*n*. Here a *pSLSrsp*(ACK) is generated and sent on its way upstream. Each NP forwards this message upstream until it reaches the source NP, which in turn delivers the message to the SP. No additional processing is done at the NPs.

If at any NP the check is unsuccessful, a *pSLSrsp*(NAK) message is created and sent upstream. At each NP, this message triggers the *de-allocation* of the bandwidth allocated by the request message, followed by the forwarding of the message to the next upstream NP, until the source is reached.

This policy allows a fast overall response because it allows parallelism between the vertical signalling to install resources in the AS and horizontal signalling along the chain. The main disadvantage is related to the possible unsuccessful reservation in a downstream NP, while in the upstream NPs, allocation has already been done:

– some bandwidth allocated by a NP, may be later de-allocated, without being used, if a *pSLSrsp*(NAK) is received from a downstream NP;

– in such a case, all vertical messages consumed for allocation/release are useless;

– while waiting a downstream response, the NP might refuse a new request for the same resources, from other SPs, because it already allocated them (later it may discover that allocation was a waste).

Policy 2: early check, late allocation. This policy separates the *checking* on the downstream way and the *allocation* on the upstream way:

At each NP, the *pSLSreq* determines a check of the resource availability. If this is successful, *no* allocation is made; the *pSLSreq* message is forwarded to the downstream NP where the process is repeated, until the *pSLSreq* reaches NP*n*. Here the inverse journey begins: the required resources are allocated, a *pSLSrsp*(ACK) is generated and sent on its way upstream. Each NP allocates the requested bandwidth, and forwards this message upstream until it reaches the source NP, which in turn delivers the message to the SP.

If, on the downstream way of the *pSLSreq*, at any NP the check is unsuccessful, a *pSLSrsp*(NAK1) message is created and sent upstream, similar to policy 1. We call this message NAK1 to indicate it is generated on the downstream journey. At each NP, this message is simply forwarded to the next upstream NP, until the source is reached.

If, on the upstream way of the *pSLSrsp*(ACK), the allocation fails, the message is changed to a *pSLSrsp*(NAK2) which is then forwarded upstream until it reaches the source, and a special *pSLSreq* indicating de-allocation is sent downstream, to the NPs which already have allocated the resources, as a result of the *pSLSrsp*(ACK). This *pSLSreq* triggers de-allocation but no other response message.

This policy eliminates the main drawback of policy 1: no allocation is done until all checks are successfully performed on the downstream way. On the upstream way however, there is a risk of allocation failure, because the resources which were available when the check was performed may no longer be available now, since a certain time has elapsed.

The advantage is fewer resources being held unused and de-allocated subsequently, with the consequence of fewer useless vertical signalling. The disadvantage is a longer response time because vertical-to-horizontal signalling parallelism is no longer possible.

Policy 3: no check, late allocation. This policy is a simplified version of policy 2.

On the downstream journey, the *pSLSreq* is simply forwarded to the next NP, without any check or allocation. When it reaches the end NP, the check is performed, the required resources are allocated, a *pSLSrsp*(ACK) is generated and sent on its way upstream. Each NP checks and allocates the requested bandwidth, and forwards this message upstream until it reaches the source NP, which in turn delivers the message to the SP.

If at either NP on the upstream way, the resources cannot be allocated, the behaviour is the same as in policy 2: a *pSLSrsp*(NAK) replaces the ACK and is forwarded upstream, and de-allocation is performed downstream. Here we don't need more than an ACK type.

The advantage of this policy is twofold: the response time is the fastest of all policies, and it is the simplest to implement since the downstream path requires no vertical actions. However, we expect a heavier rate of failures on the upstream way, most probably at the NPs closer to the destination, which receive the most of the requests.

A mathematical modelling of these policies is awkward at best, because of the many variables involved. Only a model based on heavily simplified assumptions can yield analytical solutions, and because of these oversimplifications it will probably be next to useless in real-world scenarios. So the authors' approach has been to develop a simulator in order to comparatively evaluate these policies.

3. **THE SIMULATION MODEL**

A simulator was developed (following the system structure in Fig. 1) using the SDL formal description language [8]. We used the Telelogic Tau 4.4 SDL suite [9] which comprises a SDL editor, a simulator and a validator among other modules.

The system comprises NP and SP blocks instantiated from *block types*; hence we can create a chain of arbitrary length with minimal effort. We chose a length of 4 in order to allow for a reasonable simulation time.

Besides the NP and SP blocks, we needed some additional blocks for a workable system. There is a *STAT* block collecting the numerical values by sending *get_stat* messages; these values are returned as *stat* messages. The data is processed and the results are written to disk files. Also, the initial parameters are read at the beginning by the *STAT* block from a disk file and are sent to the other blocks via *params* messages. The input parameters are:

- the total simulation time,
- the initial bandwidth available at each NP,
- the $t1$ interval between *pSLSreq* messages,
- the $t2$ forwarding time between NPs,
- the bandwidth requested in the *pSLSreq* messages.

The actual values for the last 3 parameters are generated randomly using an uniform distribution between 2 limits which are specified instead of a fixed value. Other distributions can be used. For the bandwidth, we generally use a $[-n \ldots +n]$ interval; negative values mean de-allocation, hence we do not saturate the NPs

after a certain run time; each SP knos not to try to de-allocate more then it has allocated.

The output parameters are: send/receive times, reservation messages count, useful allocated band, wasted band, time for these allocations, numbers of ACK/NAK/NAK1/NAK2.

4. **SIMULATION RESULTS**

We performed several simulations in order to comparatively evaluate policies 1, 2, 3 by means of the number of vertical messages they generate; a smaller number is better because it provides for scalability in a very large multi-domain environment, in which a large number of *pSLSreq* messages are generated. We ran separate simulations for the case $t_1 > t_2$ and $t_1 < t_2$, and we varied the initial band from 2 to 35 (conventional) units, while the requested band was between –5 and 5 units.

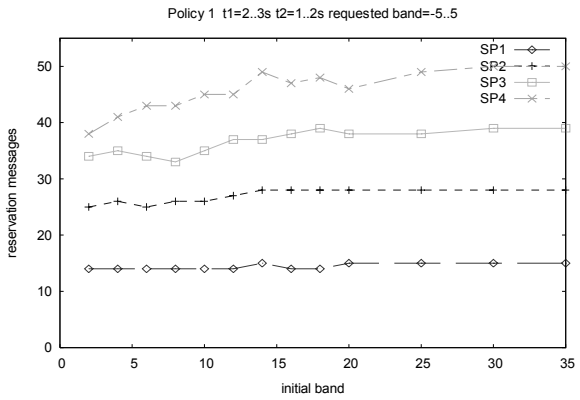

Fig. 2 – Number of reservation messages, policy 1, $t_1 > t_2$.

The number of resulting messages can be plotted for each policy and each SP; an example is shown in Fig. 2. However, a more relevant comparative study can be made not by using the number of messages, but their *frequency* defined as:

$$f = \frac{N_{msg}}{T_{sim}},$$

where N_{msg} is the average number of vertical messages for all the initial bandwidth values and both situations ($t_1 > t_2$ and $t_1 < t_2$), and T_{sim} is the simulation time; here we chose a time of 35 seconds.

The combined experimental results are numerically shown in Table 1 and plotted in Fig. 3.

Table 1

Frequency of vertical signalling messages

	Policy 1	Policy 2	Policy 3
SP1	0.41	0.1	0.13
SP2	0.77	0.34	0.35
SP3	1.03	0.57	0.58
SP4	1.28	1.04	1.35

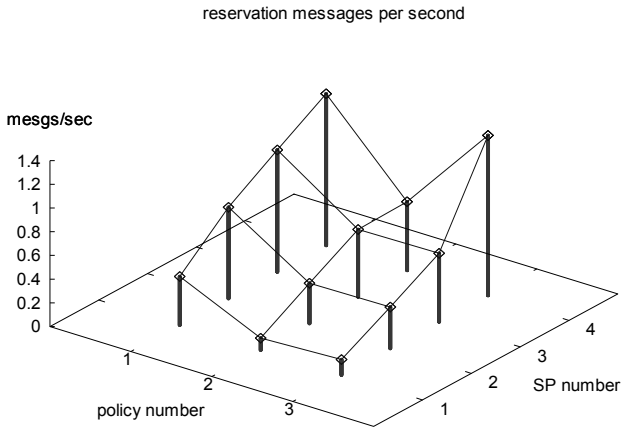

Fig. 3 – Frequency of vertical signalling messages.

5. **CONCLUSIONS AND FUTURE WORK**

By examining the results in Fig. 3, we can conclude that policy 1 generates the most vertical messages, 4 times more than policies 2/3 at NP/SP1 and twice more at NP/SP2,3. At the fourth (and final) domain, the number of messages is approximately the same using policy 1 and 3, and is half this value for policy 2. Hence, if we aim at the lowest number of vertical messages, we should choose policy 2, followed by policy 3 and 1.

The greatest number of vertical messages obtained at NP4 with policy 3 can be explained since NP4 is the one which begins the allocation/check process in the chain, and if a failure happens here, no more allocations/checks will be needed. So, NP4 “screens” the rest of the NPs from the great number of unsuccessful requests.

The first results obtained using this simulator are encouraging, since they allowed us to evaluate the behaviour of the system prior to implementation. An implementation is under way, using policy 3.

Since this simulator represents work in progress, no comparison has been performed yet between policies 1, 2, 3 with regard to other parameters, such as response time, amount of wasted bandwidth, etc. Also, the average ratio of ACK/NAK messages should be determined, as it effectively represents a *service factor* and should be as high as possible for a policy to be efficient. Additional results will be reported when they become available.

ACKNOWLEDGMENT

This work has been supported partially by the projects FP5 IST 507613 - EuroNGI and FP5 IST 507637 - ENTHRONE.

Received on 16 July, 2006

REFERENCES

1. T. Engel, H. Granzer, B. Koch, et al, *AQUILA: Adaptive Resource Control for QoS Using an IP-Based Layered Architecture*, IEEE Communications Magazine, January 2003.
2. A. Asgari, P. Trimintzios, M. Irons, et al, A Scalable real-time monitoring system for supporting traffic engineering, www.ist-tequila.org/publications/ipom02-monitoring.pdf .
3. M. Howarth, P. Flegkas, G. Pavlou *et al*., *Provisioning for Interdomain Quality of Service: the MESCAL approach,* IEEE Communications magazine, June 2005.
4. S.P.Romano, ed., *Resource Management in SLA Networks*, CADENUS Deliverable D2.3, May 2003, http:// wwwcadenus.fokus.fraunhofer.de.
5. A.Asgari, ed., et.al., *Specification of protocols, algorithm, and components, the architecture, and design of SLS Management*, ENTHRONE IST Project Public Deliverable D24F, July 2005, http://www.enthrone.org.
6. R.Yavatkar, D.Pendarakis, and R.Guerin, *A Framework for Policy-Based Admission Control*, RFC 2753, Jan. 2000.
7. A. Asgari (Ed.), P. Morand, M. Boucadair *et al.*, *D1.4:Issues in MESCAL Inter-Domain QoS Delivery: Technologies, bidirectionality, Inter-operability and financial settlements*, www.mescal.org, January 2004.
8. * * *, Telelogic Tau suite, http://www.telelogic.com.
9. * * *, The SDL Forum, http://www.sdl-forum.org.