Performance Analysis and Modelling of an OSA Gateway

Jens Andersson, Christian Nyberg and Maria Kihl

Department of Communication Systems
Lund Institute Of Technology, Sweden
Box 118 221 00 Lund
Telephone +46 46 222 91 58
Telefax +46 46 14 58 23
{jens.andersson,christian.nyberg,maria.kihl}@telecom.lth.se

Abstract. It is foreseen that you in the future should be able to use the same services independent of where you are positioned or which terminal that is used. The open service architectures provide these opportunities. Open Service Access (OSA) is an example of such an architecture and it is part of the specification delivered by 3GPP. This paper explains the OSA architecture and presents a model of an OSA gateway. Further, it discusses and proposes some feasible overload control mechanisms for the gateway. The behaviour of the mechanisms is investigated through simulation.

1 Introduction

During the last years there has been a change in service architectures towards so called open service architectures. One of the first open service architectures that was successfully developed is PARLAY specified by the Parlay group. In Parlay a set of standard application interfaces (APIs) is defined. These will enable applications residing outside the network to access and control network resources.

Open Service Access (OSA) is the service architecture that is proposed for the 3G networks. OSA is based on the concept of PARLAY, and is developed by the 3GPP [4]. It is foreseen that there will be a great demand of services and in order to respond to this demand the pace of the development has to speed up.

One common problem for all service architectures is what actions to take if the control nodes become overloaded. Overloaded nodes leads to long waiting times for service. If the waiting times get too long, customers will abandon the request for service and perhaps make a retry. These abandoned requests consume valuable processing time. In the worst case, an overloaded node will only be processing abandoned requests for service. Thereby the need of an overload control mechanism is obvious.

Overload control has been around for some decades. In Wildling [7] the protection of telephone exchanges is discussed. One paper on overload control in IN is Kihl [8].

Very few papers have been published on load issues for open service architectures. However, the performance of a Parlay gateway is analysed in Melen [9].

In this paper, we investigate overload control mechanisms for the OSA service architecture. We propose a queuing model for the most critical nodes in the architecture and investigate different ways of measuring load and rejecting customers.

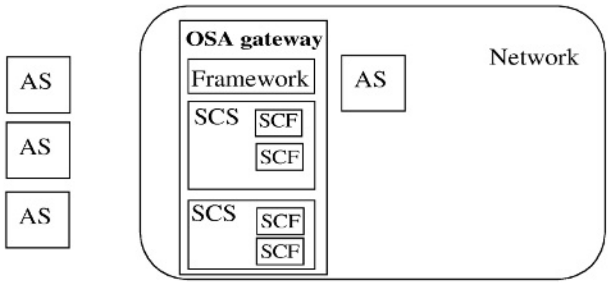

Fig. 1. The architectural picture of an OSA architecture

The paper is organized as follows: in section 2 a description of OSA is given. In 3 the simulation model is presented. The proposals for overload control mechanisms can be found i section 4 followed by the results and discussion in section 5. Finally we draw some conclusions in section 6.

2 Open Service Access (OSA)

From the beginning OSA was an acronym for Open Service Architecture, but it has been re-termed to Open Service Access. OSA offers an increased security and integrity enabling the operators to open up their networks to independent software developers and service providers. Thereby the number of feasible service providers has increased.

2.1 Architecture

OSA consists of three parts, the Application Servers (AS:s), the Service Capability Servers (SCS:s), and the Framework. Fig. 1. shows one possible configuration of an OSA architecture. The part referred to as the OSA gateway can be built on several physical entities. In Fig. 1. the Framework and both the SCS:s constitute the OSA gateway.

The AS:s host the applications. An application is usually triggered by the dialling of a special number or by some kind of HTTP request. The AS:s can be physically positioned inside or outside the network they are communicating with. An example of a typical OSA application in a 3G network is an “application initiated call” proposed in [3]. The sequence diagram of this service is shown in Fig. 2.

In an OSA architecture there can be one or several SCS:s, see Stretch [5]. The SCS provides network functionality to the applications via one or several SCF:s. An SCF consists of several narrow functions, which together makes it possible to utilize the network capability. Examples of SCF:s are Call Control, Mobility and Charging SCF. For example the Call Control SCF provides functionality to establish different kinds of calls to a mobile user. The Framework can be seen as a separate SCS providing the applications with basic mechanisms, like authentication before accessing the network functionalities or the possibility to find out which SCF:s that are provided by the SCS:s. It is important to notice that there is always exactly one framework in an OSA gateway.

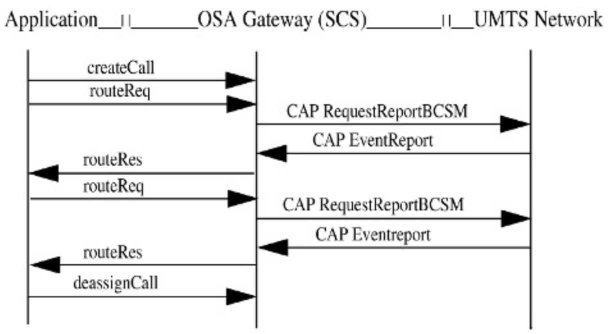

Fig. 2. Message sequence diagram for an application initiated call

2.2 Overload Control in OSA

In an OSA architecture the AS:s and the SCS:s are especially sensitive to overload. It is possible for both the AS:s and for the SCS:s to have overload control.

The overload related functionality is managed by the Framework as described in the specifications [2]. Information about the load condition in the SCS:s and the AS:s can be exchanged between the AS:s and the Framework. This gives the opportunity to control the load either from the AS or from the Framework.

There are three load levels, 0, 1 and 2 corresponding to normal load, overload and severe overload respectively. Nothing is said about how the load levels should be set or what actions they should cause. The actions should be defined in the load management policy, which is created via contract writing.

3 Simulation Model

We have developed a model consisting of one AS and a gateway containing one SCS and a Framework, Fig. 3. Each new application call is authenticated by the Framework. The execution time for this is assumed to be negligible, thus the gateway can be modelled as only the SCS. In the AS the application described in Fig. 2. is implemented. Of course in a real system there will be many applications with different behaviour. However, one is enough to create an overload situation and to evaluate the behaviour of an OSA gateway. The arrivals of the application calls are modelled as a Poisson process with the rate calls each second. The SCS is modelled as a one server queue with capacity of serving 100 application calls per second. The capacity of the AS is dimensioned so the overload will appear in the SCS and thereby the AS can be seen as a delay.

In Fig. 2. it is shown that each service has to execute in the network twice. The first time is modelled as a delay of 10 ms and the second is modelled as an exponentially distributed delay with mean 2 s. The other service times are set as follows: If a message in the SCS results in a new message the execution time is 2 ms, else 1 ms. Each delay in the AS in Fig. 3. is 1 ms.

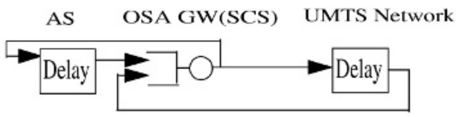

Fig. 3. The simulation model

4 Overload Control Mechanisms

An overload control mechanism should measure the current load and reject new calls if necessary. In our model the Framework rejects new application calls.

When the gateway is overloaded the waiting times get too long. In [6] a maximal delay of 100 ms is proposed and that value will be used here. If a completed application call has had a mean delay in the OSA gateway longer than 100 ms, it is said to be an expired call.

The main objectives for the overload control mechanism in this paper are to maximize the throughput and minimize the number of expired calls. To do this the number of calls in the gateway should fluctuate as little as possible so that the server is kept busy as much of the time as possible at the same time as the queue length should be kept short.

4.1 Measurement Methods

Two ways of measuring the load, A and B, are proposed below. In both cases, the measured load level at an arrival is compared to a threshold. If the measured load level is above the threshold at five consecutive arrivals or departures the load level is increased (if smaller than 2). If it is below the threshold at five consecutive arrivals or departures it is decreased (if it is larger than 0).

Method A measures the total number of application calls in the SCS, network and AS. Method B measures the number of calls in just the SCS. Calls in the network and AS will sooner or later come back to the SCS and demand processing. Method A takes this into account, B does not.

To estimate the threshold values when A is used we set T_{tot}=E(total time in system for an application call), T_{scs}=E(total time in the SCS) and x_{scs}=E(service time in the SCS for each application call). If is the threshold it must satisfy

$$\left(\hat{A} \cdot \frac{T_{scs}}{T_{tot}}\right) = \frac{100}{x_{scs}} \Rightarrow \hat{A} = \frac{100 \cdot T_{tot}}{x_{scs} \cdot T_{scs}} \tag{1}$$

to satisfy the requirement so that calls not are expired.

When method B is used the threshold can be calculated as

$$\left(\hat{B} = \frac{100}{x_{scs}}\right) \tag{2}$$

Table 1. Threshold parameters used in the simulations

Measurement method	Static method load level 1	Static method load level 2	Dynamic method load level 1	Dynamic method load level 2
Method A	190	210	190	200
Method B	40	45	30	35

4.2 Rejecting Methods

This paper proposes two methods for rejecting calls, the static method and the dynamic method. Both methods use Percent blocking [1] where R_f=P(a call is rejected).

The static method works like this: when load level is 0, R_f is 0, when load level is 1, R_f is set to 0.5 and when load level is 2, R_f is set to 1.

The dynamic method tries to stabilize the measured load just below the threshold. When load level 1 is reached R_f is increased by 0,1. If load level 1 remains after X seconds, R_f is increased one more time etc. If load level 2 is reached, R_f is increased by 0.4 in the same way. If the load level is 0, R_f is decreased by 0,1 every X:th second. Of course R_f must always be in the interval [0, 1].

In our simulations X is 25*E(total execution time in the SCS for one application call).

5 Results and Discussion

The rejecting and measurement methods will be compared in this section. The comparisons will be done with both constant and varying average arrival rates, λ. The threshold values are chosen such that the fraction of expired services never exceeds 0.5%. The used threshold values are shown in Table 1.

5.1 Comparisons of Measurement Methods

In the steady state case when we let λ keep the same value during a long interval method B gives a better throughput. We also conclude that method A is more sensitive to changes of the threshold values. If the thresholds are lowered to decrease the rate of expired calls there is a sharp decrease of the throughput when method A is used.

However, steady state arrival rates are not very realistic. It seems more probable that the value of λ is rather bursty and shifts at random times. Fig. 4. shows the simulation results when λ is randomly varying between the discrete values 0, 50, 100 and 150 calls per second and the times between changes of λ are exponentially distributed with mean 2.0 seconds. The upper plot shows how λ is varying over 200 seconds with the mean 81.6 calls per second. In the plots it can be discerned that method B is better than method A, because of the smaller variations in the number of application calls in the SCS. The mean throughput values corresponding to each of the graphs starting from the upper are 60.3, 57.6, 63.5 and 60.4 respectively.

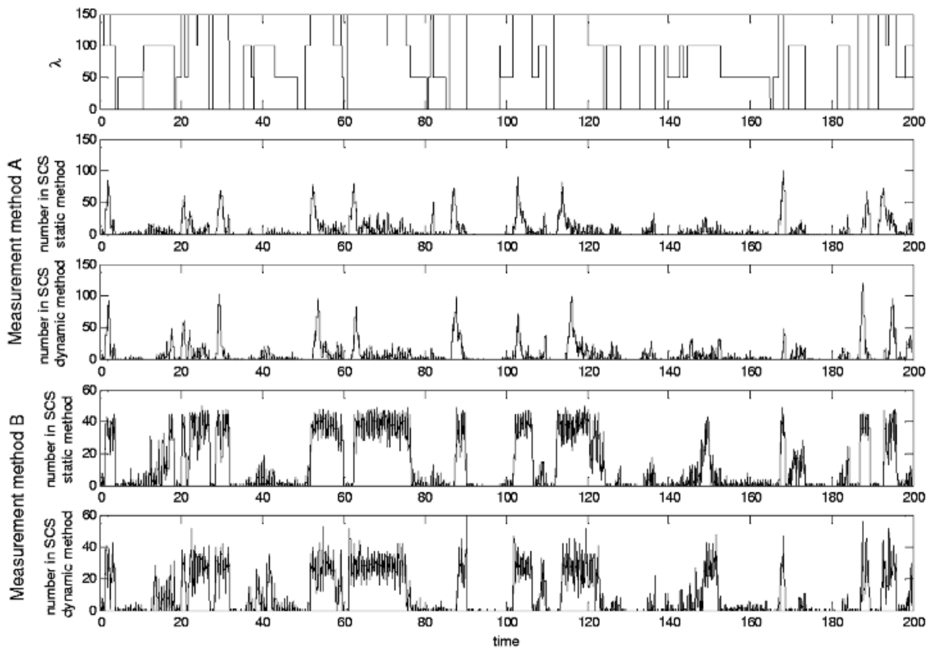

Fig. 4. The number of services in the SCS is plotted as a function of time when is varying as the top plot. In each plot different measurement method or rejecting method is used

5.2 Comparisons of Rejecting Methods

In the steady state case the throughputs are about the same irrespective of which rejecting method that is used. However, when λ shifts it can be discerned that the static method has a better behaviour from transients point of view in Fig. 4. When there is a change from $\lambda = 0$ to $\lambda = 150$ it can be discerned how the dynamic method has a slow reaction. This means that the static method fulfils the requirements just as well as the dynamic method and it seems to have a better behaviour concerning transients.

6 Conclusions

In the OSA architecture it has not been defined how to measure the load or how to react on an overload situation. In this paper it is shown that the throughput is larger when the number of calls in the SCS is used as a measure of the load than when the total number of active calls are used as a measure. We have also proposed two rejection methods of which the static method seems to have the best behaviour.

References

1. Berger, A.: "Comparsion of Call gapping and Percent blocking for overload control in distributed switching systems and telecommunications networks", IEEE Transactions on Communications, vol. 39, 1991, pp 407–414
2. ETSI standard 201 915-3 v1.3.1 “OSA API; Part 3: Framework”, july 2002
3. ETSI standard 201 915-4 v1.3.1 “OSA API; Part 4: Call Control SCF”, july 2002
4. The 3GPP home page, "www.3gpp.org"
5. Stretch, R.M.: B T Technol J. The OSA API and other related issues, Vol. 19 No 1, Jan 2001, pp 80–87
6. Eurescom Technical Information, “Non-functional aspects and requirements related to Parlay/OSA products”, june 2002
7. Wildling, K., T. Karlstedt: “Call Handling and Control of Processor Load in SPC-systems”, ITC 9, Torremolinos 1979
8. Kihl, M, Nyberg, C: "Investigation of overload control algorithms for SCPs in the intelligent network", Communications IEE Proceedings, vol. 144, pp 419–423, 1997
9. Melen, R., Moiso, C., Tognon, S.: Performance evaluation of an Parlay gateway, "http:// exp. telecomitalialab.com/pdf/06-MOISO4.pdf", 2001