Citation for the published paper:
J. Andersson, C. Nyberg and M. Kihl, 2003,
"Performance analysis and overload control of an
open service access (OSA) architecture",
Performance and control of next-generation communication networks : [ITCom's Conference on Performance and Control of Next-Generation Communication Networks] ; 9 - 10 September 2003, Orlando, Florida, USA (SPIE proceedings series ; vol. 5244).
ISBN: 0-8194-5127-4. Publisher: The International Society for Optical Engineering (SPIE).
http://dx.doi.org/10.1117/12.509294

Performance analysis and overload control of an open service access (OSA) architecture

Jens K. Andersson*, Christian Nyberg and Maria Kihl
Department of Communication Systems, Lund Institute of Technology
Box 118, SE-221 00 Lund, Sweden

ABSTRACT

The trend of the service architectures developed in telecommunications today is that they should be open in the sense that they can communicate over the borders of different networks. Instead of each network having their own service architecture with their own applications, all networks should be able to use the same applications. 3GPP, the organization developing specifications for the 3G networks has specified the standard Open Service Access (OSA), as a part of the 3G specification. OSA offers different Application Protocol Interfaces that enable an application that resides outside a network to use the capabilities of the network. This paper analyses the performance of an OSA gateway. It is examined how the overload control can be dealt with in a way to best satisfy the operators and the 3'rd parties. There are some guiding principles in the specifications, but a lot of decisions have to be made by the implementors of application servers and OSA gateways. Proposals of different requirements for an OSA architecture exist such as, minimum amount of accepted calls per second and time constraint for the maximal total delay for an application. Maximal and fair throughput have to be prioritized from the 3'rd parties view, but profit is the main interest from the operators point of view. Therefore this paper examines a priority based proposal of an overload control mechanism taking these aspects and requirements into account.

Keywords: Open Service Access, overload control, quality of service, priorities

1. INTRODUCTION

During the last years there has been a change in the evolution of service architectures. Until recently, each network has had its own service architecture and only the operator has been able to create and introduce new services. Today a couple of consortia are developing specifications for service architectures which allow interactions between different networks. Thus, an application in one network can use capabilities from other networks. Service creation will also be much easier in the new architectures. Earlier only experts could create and deploy new services, as thorough knowledge of telecom protocols was required to be able to create an application inside the network.

Open Service Access (OSA) is the service architecture that is proposed for the 3G networks. OSA is developed by the 3GPP[7]. With OSA it becomes easier to develop and test new services outside the telecom domain. Since OSA offers an increased security and integrity, it is possible for the operators to open up their networks to independent software developers and service providers, see Rajagopulan[2].

One common problem for all service architectures is what actions to take if the control nodes become overloaded. To over provision the nodes so that they can cope with all load peaks is too expensive. If a node is overloaded, long queues of jobs will be formed which 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. But the abandoned requests also consume valuable processing time. In the worst case, an overloaded node will only work with processing abandoned requests for service.

This problem may be solved by introducing overload control mechanisms in the network. The main idea is to, during overload situations, reject some requests as early as possible, so that the accepted requests receive a good service. To be able to do overload control one needs a way of detecting when a node is overloaded and also a mechanism that rejects requests when there is overload. There must also be a way of determining how the load measure should be used to calculate the parameters of the rejection mechanism.

*jens.andersson@telecom.lth.se; phone +46462229158; fax +4646145823

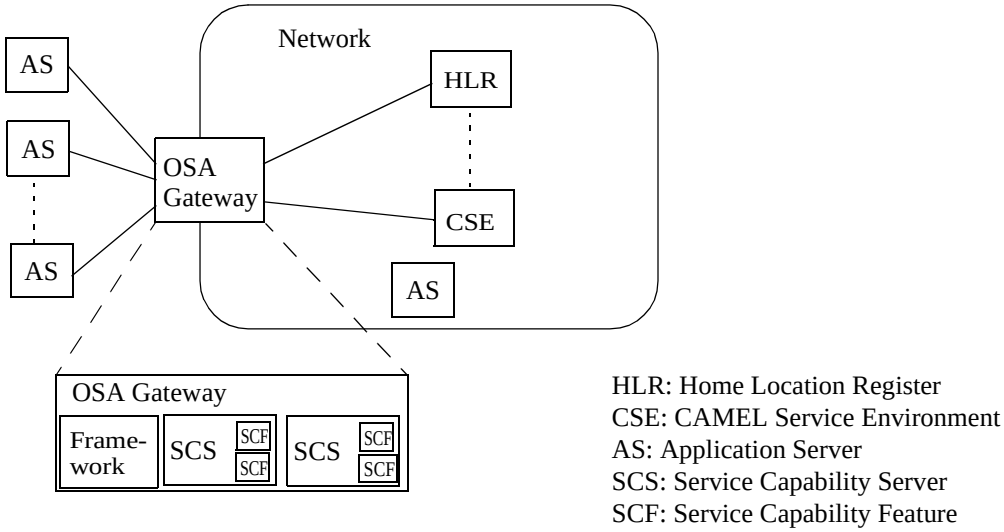

Figure 1: An example of an OSA architecture with a detailed view of the OSA Gateway

Overload control of communication systems has been a research topic for some decades in telephone networks. An early paper is Forys[8] in which the protection of control processors in telephone exchanges is discussed. Some papers on overload control in IN are Pham et al[9] and Kihl et al[10] in which overload control algorithms are suggested and investigated. The general performance of a Parlay gateway, which is almost the same as an OSA gateway was analysed in Melen et al[11]. Overload control mechanisms for an OSA architecture are proposed and investigated in Andersson et al[13].

In the context of overload control, most papers present methods on how to reject new calls in such way that the callers are treated equally. This is, of course, the fairest case from the users' point of view, but the operator's main interest is probably revenue. Therefore, we believe that an overload mechanism based on priorities should be interesting for the service providers. The priority of an application should correspond to the amount of revenue the application generates for the operator. Thus, other variables have to be included in order to maintain the user-perceived Quality of Service of the applications.

In this paper, we give a thorough description of OSA in section 2. We propose a queuing model of an OSA architecture in section 3. In section 4 priorities are discussed. An priority based overload control mechanism is proposed in section 5. In section 6 the simulation parameters are given followed by results and discussions in section 7. Finally we draw some conclusions in section 8.

2. OPEN SERVICE ACCESS (OSA)

OSA is a collection of open network Application Protocol Interfaces (APIs) that enable third party vendors to develop and deploy, with the minimum effort, applications that access the full functionality of the underlying network while still preserving its integrity. By abstracting service creation from telecommunication specific details, the development process of new applications is shortened and the creation pace of new applications can be increased.

An OSA architecture consists of three main parts, the Application Servers (AS:s), the Service Capability Servers (SCS:s), and the Framework. Fig. 1 shows one of the possible configurations of an OSA architecture. Each SCS hosts one or several Service Capability Features (SCF:s), which are abstractions of the underlying network functionality. The part referred to as the OSA gateway can be built on one or several physical entities. In Fig. 1 the Framework and both the SCS:s constitute the OSA gateway.

The Application Servers (AS:s) host the applications. Each AS can host one or several different applications and provide them with the ability to communicate with the Network via the OSA Gateway. The AS:s can be physically positioned inside or outside the network they are communicating with. An AS positioned outside the network of an operator is

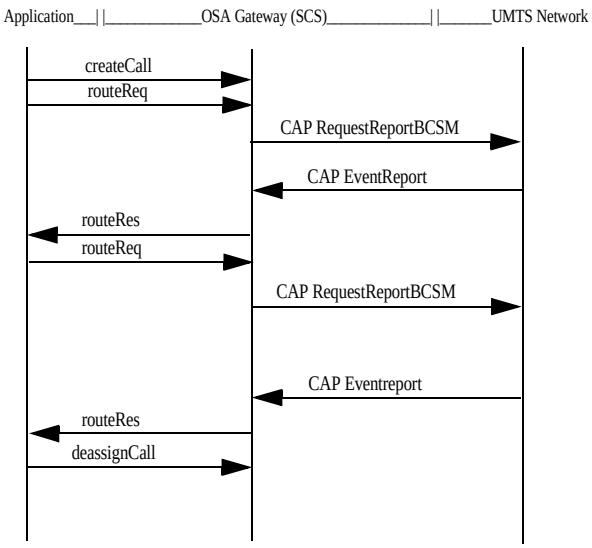

Figure 2: Message sequence diagram for an application initiated call.

typically connected to the Internet. Usually an AS is triggered by the dialling of a special number or by some kind of HTTP request. Examples of applications are conferencing, location based applications and so forth.

In an OSA architecture there can be one or several Service Capability Servers (SCS:s). More about the implementing of SCS:s can be found in Stretch[3]. The SCS provides the applications with network functionality via one or several SCF:s. An SCF consists of several narrow functions, which together makes it possible to utilize the network capability. One example is the Call Control SCF, which provides functionality to connect and establish different kind of calls to a mobile user. Another example is the Charging SCF, which provides functionality to charge the user for a service.

The Framework is the most important part in the OSA architecture. This part takes responsibility for all security aspects of OSA. For example it provides the applications with functionality like authentication before accessing the network functionality or discovery to find out which SCFs that are provided by the SCSs. All the security and integrity functionalities necessary to open up a network are provided by the Framework. It is important to notice that there is always exactly one Framework in an OSA architecture.

2.1 An example of a service in OSA

In the specification[4], a couple of OSA applications are proposed. One example is an “application initiated call”. In this application for example a customer accesses a Web page and selects a name on the page of a person or organisation to talk to. The sequence diagram of this application is shown in Fig. 2. An application setup consists of a number of OSA messages. First the application sends a createCall message to the OSA Gateway to create objects for further communication. In the Gateway the application call is translated into suitable protocols for communication with for example an UMTS network. When the A party has answered, the application is notified and then the call is routed to the B party.

2.2 Overload control in OSA

In an OSA architecture there are especially two parts sensitive to overload, the AS:s and the SCS:s. The most critical SCF seen from the aspect of overload is the Call Control SCF which connects and initiates calls. The overload related functionality is managed by the Framework. In the specification[5], there is a description of the functionality that is prepared. Information about the load condition in the SCS:s and the AS:s can be exchanged which gives the opportunity to control the load either from the application side or from the Gateway.

The load condition is described by three levels. Load level 0 corresponds to normal load, load level 1 corresponds to overload and load level 2 corresponds to severe overload. Nothing is said about how the load levels should be set or what actions they should cause, but corresponding threshold values to load level 1 and 2 can be set. Different SCS:s can have

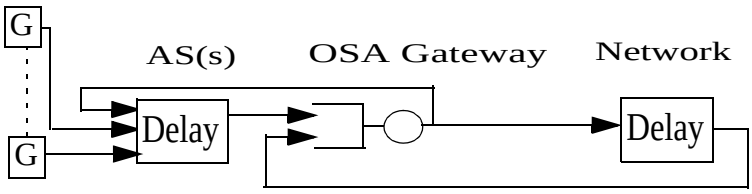

Figure 3: An OSA Model.

different threshold values for the load levels. The action an overload situation should cause on a specific application is identified in the load management policy, which is created via contract writing (see below).

It is possible for the Framework to subscribe on load information both from an AS and an SCS. The subscription can be either load information sent to the Framework at discrete times or load information sent on a load level change.

2.3 Contract writing

When a new application is introduced a contract is written with the OSA gateway through the Framework. The contract contains rules and restrictions that should be followed. Proposals of what a contract should include can be read for example in Rajagopulan[2]. A typical contract might include minimum throughput for an application and maximal delay of an application call. Another variable that might be agreed on is the charging criteria. The contracts should not only consist of constraints for the applications according to the gateway. Also the constraints that the application has to fulfil is agreed.

3. MODEL

We have modelled an OSA gateway built on a single physical entity in a multi application environment. Our model consists of *R* applications, a Gateway and a network, see Fig. 3. We do not have to specify how many SCS:s there are in the gateway as they are positioned on the same physical entity anyway. Either all or no SCS:s are overloaded.

Each G-box in Fig. 3 corresponds to a generator of new application calls to a specific application. We will assume that calls are generated according to a Poisson process or according to an MMPP, but other processes could of course be used in the simulation model. Each application is assumed to be positioned at a non-overloaded AS which is modelled as a delay with deterministic values, depending of which message that should executed.

An application, a_l, has a guaranteed rate of d_l calls per second, and a total execution time in the Gateway of $x_{tot}(l)$ seconds. Of course the system must be stable when all applications face their guaranteed rate, which implies that

$$\sum_{l=1}^{R} d_l \cdot x_{tot}(l) < 1 \qquad (1)$$

Each application belongs to a priority, p_j , $(j = 1 \ldots M)$ and has a time constraint, T_k ,$(k = 1 \ldots N)$. Each priority corresponds to a guaranteed rate of application calls per second, where p_j guarantees a higher rate than p_{j+1}. A time constraint T_k corresponds to the maximum delay a message should experience each time it passes the Gateway. In the example in Fig. 2 the application call has to pass the gateway five times and if the application call, when it is completed, has had a total residence time in the gateway larger than $5 \cdot T_k$, the call is said to be expired. The time constraints are set such that $T_1 < T_2 < \ldots < T_N$. The set of applications with time constraint *k* is denoted *A(T_k)*. The total guaranteed rate of applications with time constraint *k*, is denoted λ_k. λ_k is given by

$$\lambda_k = \sum_{l \in A(T_k)} d_l$$

The time constraints and priorities can be set independently of each other for each application.

The gateway is modelled as a single server queue, in which one message at a time is served. This means that the messages are stored in the same queue when waiting for execution independent of which SCS they belongs to. The execution times in the gateway are set to deterministic values, depending of which message that should be executed. We denote with *N(t)*

the number of messages in the gateway with a remaining time less than *t* before their deadline expires. Another variable that will be used in this article is x_{GW}, which is an estimation of the execution time of a random message in the gateway. We will use different values for this variable, see section 6.

The network is assumed to be non-overloaded and is modelled as a deterministic delay with stochastic elements from, for example, the phone pick up time in Fig. 2.

4. PRIORITIES

The priorities should be set in such way that the utility is largest for the priority 1 applications and decreasing by increasing priority levels. This is a consequence of the statement that p_j guarantees a higher rate of accepted application calls than p_{j+1}, and that we of course want to maximize the utility. In this section we propose a utility function that can be used for the settings of the priorities.

4.1 The utility function

First of all utility has to be defined. The utility should somehow correspond to the use of spending processor time on an application. We believe that revenue is the main interest for the operator or owner of the OSA gateway, which means that revenue should have influence on the utility. However, the revenue of an application cannot directly be used as a measure of utility. Assume that a very resource consuming application exists. Even if this application generates a good revenue it might happen that another application exists, which generates less revenue but has a much shorter total execution time in the Gateway. Therefore, the total revenue for the latter application during a minute may be higher than the former. So we have to weight the revenue against the total execution time in the gateway. The utility function of a_l is now described by the following expression

$$U(l) = \frac{r(l)}{x_{tot}(l)} \tag{2}$$

where *r(l)* is the revenue. *r(l)* is the income the operator receives each time an application call from a_l is served minus the cost of maintaining the network functionality that the application use. In Eurescom Technical Information[12] the players in a Parlay/OSA business environment are discussed. The operator is only one of many players, so the income to the operator per application call is not equivalent to the cost of an application call for the customer as the receipts should be distributed among several players. This means that the utility an application corresponds to is a measure of how much revenue the operator will have through executing applications of the same kind for one second.

Equation (2) gives a good estimation of the utility seen from the aspect of revenue in the short run. But in the long run it should be an advantage for the operator to have an extra goodwill parameter, G, that could be set individually and be added to (2). For example a completely new application representing a new kind of service might lead to larger revenue in the future if it is experienced well and then the development pace of similar applications might increase. So if this goodwill parameter is based on qualified market analysis this would probably increase the revenue in the long run.

The priority of an application depends on the utility function. A discrete number of priorities will be used, and thereby each priority level will correspond to a utility interval.

5. OVERLOAD CONTROL MECHANISMS

We have developed an overload control mechanism for the gateway, see Fig. 4. It consists of a controller, a gate and a selector. The *controller* makes appropriate measurements on the gateway. Also, it analyses the measurement data and determines what action that has to be taken by the *gate*, which regulates the acceptance of new application calls.

The objective of the overload control scheme is to keep all time constraints for the accepted application calls. As different applications can have different time constraints the *selector* has to decide in which order the messages in the gateway should be served. The selector uses an Earliest Deadline First (EDF) scheduling algorithm, see Lui et al[1]. The controller performs measurements of the load status in the gateway to check if it is possible for the selector to pick the messages in such order that none of the time constraints is expired. If not, the controller orders the gate to decrease the acceptance of new application calls. If the time constraints can be kept the gate is told to increase the acceptance of new application

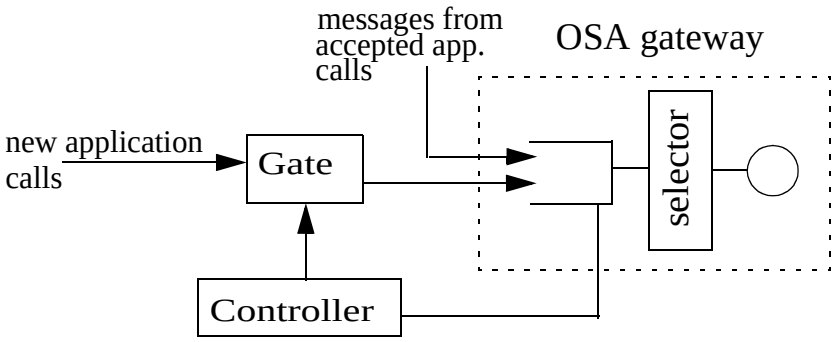

Figure 4: Model of the overload control mechanism

calls if possible.

5.1 Gate

When the gateway is overloaded, the gate starts to reject application calls. Since each application is guaranteed a minimum rate of accepted application calls per second, we have chosen to let the gate use a call gapping method, see Berger[6], to reject application calls. The time is divided into small intervals of a certain length, and then the first application call in the interval is accepted. The interval lengths depend on the guaranteed rate the application has. If, for example, an application is guaranteed at least 10 calls per second, this corresponds to a time interval of 0.1 seconds. During an overload situation (load level 1) the gate introduces call gapping on the lowest priority applications. If this action is not enough and the overload condition remains after *X* seconds, call gapping is introduced on application calls of the next priority level, and so on every *X*:th second until applications from all priority levels have their calls rejected according to the call gapping method. The parameter *X* should be set taking into account the capacity of the gateway. If a severe overload condition (load level 2) appears all the priority levels are blocked at once, only letting the guaranteed amount of application calls through.

5.2 Controller

For each arriving or departing message the controller checks if the time constraints for the messages waiting in the queue may fail if the message is admitted. The following condition should of course always be fulfilled:

$$N(T_k) \cdot x_{GW} \leq T_k, \quad (1 \leq k \leq N) \tag{3}$$

If not fulfilled, application calls with time constraint T_k will most probably fail even if the gate starts to reject arriving application calls at this stage.

Fig. 5 can be used to easier explain the calculations that is performed by the controller to decide the current load condition in the Gateway. Each time constraint can be seen as an insertion point in the waiting queue for an application of a certain time constraint. When a message is inserted into the queue, this can be seen as some kind of time axis where the messages proceeds along this time axis as the distance to their deadline. When the execution of one message is completed the next message at the front of the queue is executed.

While a message with time constraint T_3 is waiting to get first in the queue it is possible for new messages with time constraint T_1 and T_2 to arrive at the queue with a closer deadline and thereby get a closer position to service. This means that during the interval of length $T_3 - T_2$ after the arrival of a time constraint 3 message, all applications with time constraint T_1 or T_2 will have a closer deadline. Then during another interval of $T_2 - T_1$, arriving messages with time constraint T_1 will have a closer deadline. Therefore, condition (3) can be improved to also include the guaranteed rate of new application calls. This condition can be expressed as

$$\left(N(T_k) + \left(\sum_{j=1}^{k-1} \lambda_j \cdot (T_k - T_j) \right) \right) \cdot x_{GW} < T_k \qquad 1 \leq k \leq N \tag{4}$$

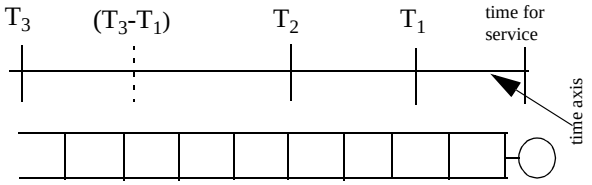

Figure 5: Abstraction of controller mechanism

If this condition is fulfilled and if the gate only let the guaranteed application calls through, the messages currently in the queue will most probably be served within their time constraints.

However, the controller should also check that the condition in (4) not is violated in the future by admitting too many calls from applications with less tough time constraints. Assume for example that an application call with time constraint T_3 arrives at time t_0. Let A be the set of all calls with a deadline in the interval $[t_0 - T_1, t_0]$ at this arrival. After $T_3 - T_1$ seconds all the calls in A will have a deadline that is less than T_1 seconds in the future. But in the time interval $[t_0, t_0 + T_3 - T_1]$ messages with time constraint T_2 and T_1 may have arrived to A and these messages will also have a deadline that is less than T_1 seconds in the future.

If a burst of application calls with time constraint T_i arrives and we start to reject calls from all priority levels, then the maximal number of messages that might be additional to an interval of length T_k can be expressed as

$$\sum_{j=k}^{i-1} \lambda_j \cdot T + \sum_{j=1}^{k-1} \lambda_j \cdot T_j \qquad \begin{matrix} 2 \le i \le N \\ i > k \end{matrix}$$
$$T = \begin{cases} T_k & T_k < (T_i - T_j) \\ (T_i - T_j) & T_k \ge (T_i - T_j) \end{cases} \tag{5}$$

and the execution time of these should be added to the execution time of all initial messages in the interval of length T_k. Just as before we also have to include that new application calls might arrive during the execution of the messages in the interval. This new condition can be described as

$$\left(N(T_i) - N(T_i - T_k) + \left(\sum_{j=1}^{k-1} \lambda_j \cdot (T_k - T_j) \right) + \sum_{j=k}^{i-1} \lambda_j \cdot T + \sum_{j=1}^{k-1} \lambda_j \cdot T_j \right) \cdot x_{GW} < T_k \qquad \begin{matrix} 2 \le i \le N \\ i > k \end{matrix} \tag{6}$$

where T is the same as in equation (5). This constraint has to be fulfilled for all possible combinations of T_k and T_i, where $i > k$.

If conditions (4) and (6) are fulfilled, the controller decides that the system has a high probability to succeed without too many expired deadlines. If any of the two conditions fail, the controller signals overload to the gate. Too further decrease the number of expired deadlines and to get a more calmly behaviour, the controller uses a marginal when signalling for overload. This marginal is created by multiplying the right hand of the conditions with a marginal factor, $f<1$. If any of the conditions are violated when the right hand side is multiplied with f, the controller signals overload (load level 1). If any of the conditions are violated without the marginal factor, the controller signals severe overload (load level 2) to the gate.

6. SIMULATION PARAMETERS

In the simulations we have used 10 applications with different behaviour. In our implementation we support two different time constraints and three priorities for the applications. The different applications also differ in execution times in the OSA Gateway, delays in the AS, delays in the network and the number of times an application needs to pass the gateway. Table 1 shows the configuration of the applications used in our simulations. The sequence diagram of application 1, 2 and

Table 1: The configuration of the different applications.

Application	Execution times in the OSA Gateway	Execution times in the AS	Execution times in the network	Priority	Time constraint
1	0.001, 0.002, 0.002, 0.002, 0.002, 0.001	0.001, 0.001, 0.001	0.008, exp(2,0)	1	0.1
2	0.002, 0.003, 0.003, 0.003, 0.003, 0.002	0.002, 0.002, 0.002	0.01, exp(0.01)	2	1.0
3	0.0001, 0.0002, 0.0002, 0.0002, 0.0002, 0.0001	0.001, 0.001, 0.001	0.01, exp(2.0)	3	0.1
4	0.0015, 0.0025, 0.0025, 0.0015	0.0015, 0.0015	0.0125	1	1.0
5	0.0001, 0.0002, 0.0002, 0.0001	0.0001, 0.0001	0.0008	2	0.1
6	0.001, 0.002, 0.002, 0.001	0.001, 0.001	0.008	3	1.0
7	0.002, 0.002	0.001, 0.001	0.018	1	0.1
8	0.003, 0.003	0.005, 0.005	0.008	2	1.0
9	0.0002, 0.0002	0.0001, 0.0001	0.0008	3	0.1
10	0.0015, 0.0015	0.0001, 0.0001	0.0035	3	1.0

3 is the same as the example shown in Fig. 2. The sequence diagrams of the other applications are not shown, but their most important properties can be seen in the table. The reason for that we have one exponentially distributed and one deterministic service time in the network for application 1, 2 and 3 is that these correspond to some kind of call establishing look alike applications. Such applications have to execute twice in the network. The first time is when the call is connected to the callers phone. In this case a deterministic delay is used, since there is probably some kind of auto phone pick up function for the caller. The second execution correspond to the B party pick up time. This pick up time is modelled to be exponentially distributed.

Priorities 1, 2, and 3 correspond to guaranteed rates, d_l, of 10, 5, and 0.5 accepted calls per second, respectively.

The marginal factor *f*, used in the controller, is set to 0.9. To prevent the system from oscillating, the load level is only changed if the controller detects overload for five consecutive arrivals or departures. The parameter *X* introduced in section 5.1 is in our simulations set to 50ms.

During the simulations we have used different values of x_{GW} depending of which features that are prioritized. We think that two different values can be motivated. Either we use the upper quartile of the execution times in the Gateway for all messages from all applications or we use the largest execution time in the Gateway. The choices are referred to as configuration 1 and configuration 2. The choice of x_{GW} to the upper quartile is motivated as it seems not realistic that only applications using messages with the largest execution times in the Gateway are used. Thereby this will give a good estimation of the maximum mean of the execution time for the messages in the queue, but with this choice we cannot guarantee the rate of expired application calls. The choice of x_{GW} to be the largest execution time among all applications is motivated as this will result in a very limited amount of expired application calls. This reasoning counts for all conditions where x_{GW} takes part in this article

7. RESULTS AND DISCUSSION

In this section the overload control mechanism will be evaluated. Simulation results are presented and the gain of the proposed overload control mechanism is discussed.

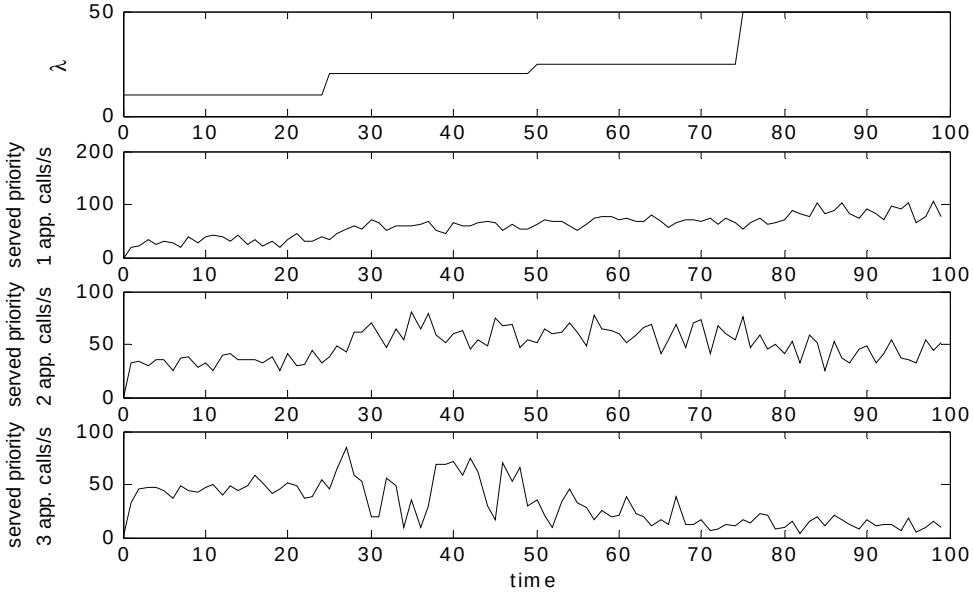

Figure 6: The outcome of a realization when configuration 1 is used. The top graph shows the mean λ of each application. There are three priority 1 applications, three priority 2 applications, and four priority 3 applications

We first consider arrivals according to a poisson process where the rate, λ, is changed every 25:th second. At first λ is 10 calls/s, and is then increased to 20, 25 and 50 calls/s. In Fig. 6 the resulting rates of completed application calls per second for each of the three priorities are shown when configuration 1 is used. During the first 25 seconds no calls are rejected, since the total number of arriving application calls is below the capacity of the Gateway. However, after 25 seconds λ is almost equal to the capacity of the Gateway and the priority 3 application calls are slightly affected. After 50 seconds, the total arrival rate exceeds the capacity of the Gateway and, therefore, application calls with the lowest priority are rejected. After 75 seconds, calls from all priority levels are rejected, but none of the priority levels have all their application calls rejected. This is because of the guaranteed amount of application calls. In the realization in Fig. 6 about 5% of the served application calls were so-called expired calls. However, most of the expired application calls break their time constraints only with a few percent of their constraints.

If configuration 2 is used the outcome of a typical realization when steady state is used is seen in Fig. 7. In this realization 0% of the served application calls are expired calls.

It is interesting to see what can be gained with a priority based rejection mechanism as proposed. An estimation of how good the outcome is can be performed by using the utility measure. Assume that the applications of priority 1, priority 2 and priority 3 corresponds to utility 3, 2 and 1 respectively. Then we can get a measure of the gain by calculating how much time the processor has spent on the different applications and take into account the utility of each application. The utility of a realization can then be defined as

$$\sum_{l=1}^{R} s(l) \cdot x_{tot}(l) \cdot U(l) \tag{7}$$

where s(l) is the number of served application calls from application l during one simulation.

In the realization shown in Fig. 6 we get a total utility of 213. The same calculation for the gain of the realization shown in Fig. 7 results in a total utility of 214. To be able to do a comparison of this result we have done the same simulation where we have used an ordinary random rejection method where 50% of the application calls are rejected during an

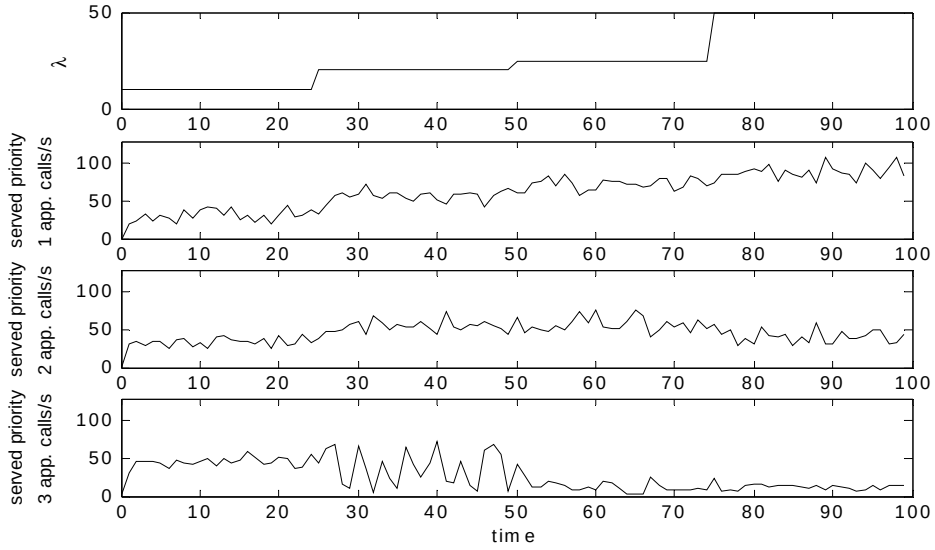

Figure 7: The outcome of a realization when configuration 2 is used. The top graph shows the mean λ of each application. There are three priority 1 applications, three priority 2 applications and four priority 3 applications

overload (load level 1) situation, and where all application calls except the guaranteed amount is rejected during an severe overload (load level 2) situation. When this overload control mechanism is used we get a total utility of 199 if the parameters are set so that 3,5% of the served application calls are expired. Observe that when this mechanism is used we cannot guarantee that the requirement of the guaranteed rates of accepted application calls per second is fulfilled during an overload (load level 1) situation.

In the plots it is seen how configuration 2 rejects more application calls than configuration 1 during an overload situation. It can also be discerned that there are larger differences between the number of accepted calls from different priorities when configuration 2 is used. When the random rejection mechanism is used all applications have about the same amount of application calls served per second. As a consequence of this we get less utility.

However, arrivals according to a Poisson Process is probably not so realistic. Therefore we have also used an MMPP with six states to generate arrivals. The states correspond to arrival rates 0, 10, 20, 30, 40 and 50 calls per second. The mean time in a state is two seconds and when we leave a state, we enter one of the others with equal probability. Under these circumstances and when configuration 1 is used the rate of expired calls is about 0.15%. And when configuration 2 is used the rate of expired calls is 0%. As we assume that the mean execution time is higher in configuration 2 than in configuration 1 we cannot allow the same queue length. Thereby the queue will become empty more often as a consequence if it is filled by messages with short execution time, and thereby the utilization is lower if configuration 2 is used.

If we do the same calculations of the gain as before, but with an process of the arrivals according to MMPP we get a total utility of about 2350 when configuration 1 is used during a simulation of 1000 seconds. The same calculation for a configuration 2 simulation during 1000 seconds results in a total utility of 2280. If we use the earlier described random rejection mechanism, with the parameters set such that the rate of expired calls is 0.1%, we get a total utility of 2180.

Clearly we get a better gain when the priority based overload control is used. The values of x_{GW} have different advantages. The choice of this value should be made depending of how the contract is written and what the arrival process look like. During an MMPP arrival process, which we think is the most realistic, the use of the upper quartile results in a better utilization and a higher gain than if we use the largest execution time, which is an argument for configuration 1. But we also get a higher fraction of broken deadlines, which is an argument for configuration 2.

8. CONCLUSIONS

We have modelled an overload control mechanism for an OSA gateway. The Overload control mechanism that is proposed is designed to support two probable requirements in an OSA architecture. It is able to guarantee a minimum rate of accepted application calls per second dependent of which priority an application correspond to. It is also designed to make sure that application calls that are accepted will meet their time constraint with a high probability. Further, discussions are held and a proposal is presented of how the priorities can be set in order to maximize the revenue for the owner of the Gateway.

The different parts in the overload control mechanism are build independent of each other such that either the gate, the selector or the controller can be exchanged without any affect on the other parts. Also the utility function can be exchanged.

By simulations the overload control is evaluated and the appearance of the wanted behaviour is verified. Also, we have shown that the total gain of the served application calls is higher when using the priority based rejection mechanism compared with a random rejection mechanism.

ACKNOWLEDGEMENT

This work has partially been financed by the Swedish Research Council, contract no 621-2001-3053.

REFERENCES

1 C. Liu, J. W. Layland, “Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment”, *Journal of the Association for Computing Machinery*, **Vol. 20 No. 1**, 46-61, 1973.
2 R. Rajagopulan, “The impact of Open Service Access on Network Services”, *http://www.wmrc.com/businessbriefing/pdf/wireless_2003/Technology/lucent.pdf*, 2002.
3 R. M. Stretch, “The OSA API and other related issues”, *B T Technol J*., **Vol. 19 No 1**, 80-87, 2001
4 ETSI standard 201 915-4 v1.3.1, “Open Service Access (OSA); Application Programming Interface (API); Part 4: Call Control SCF”, 2002
5 ETSI standard 201 915-3 v1.3.1, “Open Service Access (OSA); Application Programming Interface (API); Part 3: Framework”, 2002
6 A. Berger, “Comparison of Call gapping and Percent blocking for overload control in distributed switching systems and telecommunications networks”, *IEEE Transactions on Communications*, **vol. 39**, 407-414, 1991
7 The 3GPP home page, "www.3gpp.org"
8 L. J. Forys, “Performance Analysis of a New Overload Strategy”, *ITC 10*, 1983
9 X. H. Pham, R. Betts, “Congestion Control for Intelligent Networks”, *1992 International Zurich Seminar on Digital Communications*, 1992
10 M. Kihl, C. Nyberg, “Investigation of overload control algorithms for SCPs in the intelligent network”, *Communications IEE Proceedings*, **vol. 144**, 419-423, 1997
11 R. Melen, C. Moiso, S. Tognon, “Performance evaluation of an Parlay gateway”, *http://exp. telecomitalialab.com/pdf/06-MOISO4.pdf*, 2001
12 Eurescom Technical Information, ”Parlay/OSA Business Models: An Operator’s Perspective”, December 2002
13 J. Andersson, M. Kihl, C. Nyberg, “Performance analysis and modelling of an OSA gateway”, Accepted to PWC2003, 2003