# QoS based Service Provisioning in NGN/NGS Context

Soumia Kessal
Department INFRES
TELECOM ParisTech - LTCI - UMR 5141 CNRS
*kessal@telecom-paristech.fr*

Noëmie Simoni
Department INFRES
TELECOM ParisTech - LTCI - UMR 5141 CNRS
*simoni@telecom-paristech.fr*

*Abstract*— **Today, service provisioning suggests reserving network and equipment resources to satisfy a user request, but in the Next Generation Network and Next Generation Service (NGN / NGS) context the user's needs are becoming increasingly complex and need to take account the user mobility and preferences changes in the behavior of the service. Thus, it is necessary to reserve the "Service Resource" as well as the network and equipment resources. In this paper we propose a mechanism for service resource provisioning similar to that of routing networks. This service resource is shared between several users and is allocated dynamically according to its current QoS. The dynamic management of this service resource allows us to anticipate degradations and thus to always satisfy the user Service Level Agreement.**

***Keyword; Service Provisioning; QoS; Service Resource; Queue Management; NGN/NGS Context***

## I. INTRODUCTION

In the upcoming NGN/NGS context, the only way for the service providers to gain a long-term market position is by participating ubiquitous service environment to federate a wide variety of services and heterogeneous network supports. On the other side, the user desires to benefit from a continuous service delivery from this environment during his mobility while keeping his customisation. Once we put the user as the centre of the consideration, an end-to-end service session should integrate user's preferences with the dynamic ambient context, which includes the resources of end terminals, access networks, core networks and services. The provisioning of all these resources becomes as a challenge.

However, in the NGN dynamic context, where the degradation of the QoS is more probably caused by the mobility of a user, the existing solutions can partially resolve the problem but with an interruption of the service delivery. In fact, conceptually, the service should be an important type of resource besides network and equipment resources. Thus, the ubiquitous service environment can serve or even anticipate the service discontinuity. On the other hand, with the increasingly complex new services like IPTV, VoIP, etc, and with increasingly dynamic contexts like the mobile or ubiquitous world, the management of these services becomes a requirement. Furthermore, this management is also expected to be optimized the usage of the service when it is shared with several users requests. As a result, we should not only provision and monitor the service as well as the network resources, but also dynamically verify the conformation to the QoS contract established between service providers and users.

In this paper we present a service provisioning based on the QoS of the service in an NGN/NGS context. In this context end-users claim the access to their services anywhere, anytime and anyhow, without having to comply with system, application and network constraints. Thus, we must assure a continuous session to the user that best answers his ambient environment, his preferences changes and his QoS requirements. For this purpose, we propose to manage the behavior of the service by QoS during usage, taking into account the mobility or change in the ambient environment of users. The services that we provision are ubiquitous and mutual between different users.

The paper is organized as follows:
In Section 2, the existing work in the service provisioning in SON and SOA are presented in order to show the advantage of our proposal. In Section 3, our proposition for a QoS based service provisioning is detailed. In order to show the management of the service mutualisation by a QoS Agent, we present in Section 4 our implementation by Java Message Service (JMS) and Enterprise Java Bean (EJB). We finish by a conclusion in Section 5.

## II. RELATED WORK

In the NGN/NGS context, the user requirements in the application level are more and more complex. So the management of resources needs to be more and more flexible, with the result that equipment and network provisioning are not enough and service provisioning is necessary to face today's user requirements. In this section we analyse solutions proposed by the Service Oriented Architecture (SOA) [1] and by the Service Overlay Network (SON).

### *A. Service provisioning in SOA*

We find interesting solutions that take into account the QoS for the media delivery (at network level for any type of service) but without incorporating the service elements (service delivery).

M. Boniface et al. propose in [2] a dynamic service provisioning using SLAs to maximise resource utilisation. This solution is based on the predicted evolution of the QoS characteristics of the equipment and the network to anticipate SLA violation but not at the service level

X. Li et al. propose in [3] a scalable SLA-driven QoS management platform which manage various distributed resources and adapt to the dynamic changes of the resource availabilities to meet the client QoS requirements.

S. Chen et al. propose in [4] a context-aware dynamic resource management middleware for service oriented applications, which aims to handle the inherent dynamics of the local devices and the network.

### B. *Service provisioning in SON*

Solutions on SON [5] have gradually evolved through:

SATO (Service-aware Adaptive Transport Overlay) [6], which reorganize the ambient resources to adapt to the provision of the service.

QSON (QoS-aware SON) [7], which divides SATO into several subnets, each of them corresponds to a level of QoS.

DSON (Dynamic SON) [8], which aims at composing service specific delivery platform dynamically with distributed service components for reliable service provisioning allowing mobility regardless of underlying transport technologies.

But, the service provisioning can assure certain QoS when mobility occurs. However, the granularity of the equivalent QoS subsets is not ideal enough to have agile and dynamic adaptation when QoS degradation occurs. One of the challenges of our solution is that it anticipates SLA violation by dynamic Current QoS monitoring of the service component.

## III. Proposition

To take into account the different types of resources (Service, Network and Equipment) we propose abstraction of each type of resource as a network of resources, which self-manages in autonomous manner. These resource networks are called Virtual Private x Networks (VPxN, x: Service, Connectivity and Equipment) (Figure 1). The advantage to have these VPxN is that during mobility or preferences changes each resource in the VPxN can be replaced by another ubiquitous resource to always maintain the user SLA. When we select resources in the VPxN, we have their Offered QoS and their logical address. Then, once the user sends his request to use some services in the VPSN, we provision the resources on all the levels (VPxN) according to their Current QoS. We select dynamically these resources since between the session initiation and usage the current QoS of each resource can change. This VPxN concept enables us to have service continuity during mobility by dynamically selecting the resources according to their current QoS in the user's ambient area.

Mobility makes us partition the global network in successive ambient networks. For that, we must have a dynamic reselection to maintain E2E QoS during mobility and to optimally use the local resources (nearest ubiquitous resources). Then, according to the user location and thanks to the resource ubiquity, we select the resource, which meet his preferences and which satisfy his requested QoS.

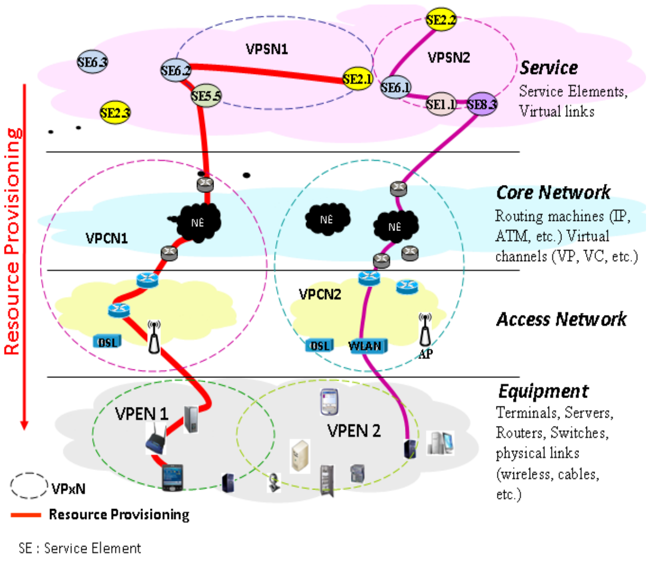


Figure 1. Resource provisioning at all visibility levels

The service must be provided regardless of the level of mobility and dynamicity that we face and according to the QoS. Thus, in addition to the media delivery which ensures the delivery of the flow from a source to a destination, we have a service delivery which manages mobility with service elements which are ubiquitous, mutual, self managed and autonomous. This service delivery ensures service continuity regardless of the source and the destination

Our proposal consists in managing the provisioning in the service layer in a similar way to that of the network. In fact, the network resources, typically routers and virtual circuits, offer their capabilities in a distributed manner facing whatever the user. To ensure the delivery of the flow from a source to a destination (media delivery), the network resources are managed dynamically, so for example when a router is congested, the flow is re-routed to another router by maintaining the requested QoS of the user. Similarly, we propose to manage the service itself as a resource by provisioning its QoS dynamically. In order to maximize the usage of the service when it is shared among several requests, we propose to integrate a queue in this service. The service accepts the requests in the queue according to its current QoS. Thus, in addition to the management of the node (the service), we manage the link between the services by this queue. In case of service congestion, we can send the requests to another ubiquitous queue to maintain the SLA of the requests.

In this section we propose in (§A) to provision the service during usage as a resource: 'Service Resource' and according to its current QoS. In order to share this resource in an optimal way we propose (§B) to integrate a queue in the SE. The queue is controlled by a Queue QoS Agent. The resource is shared according to its Current QoS among several user requests. In case the QoS of a request is not guaranteed, it is sent to another ubiquitous service resource. Finally we show the queue management mechanism (§C).

## A. *Provisioning of the "Service Resource" based on QoS*

At usage, we reserve dynamically the resources selected in the VPSN (Pre Provisioning phase) according to their current QoS. Our service resource being shared by several users, we should allocate it dynamically according to its current QoS. The current QoS aggregates the four criteria of our QoS model (Availability, Reliability, Capacity and Delay).

To manage this service mutualisation we propose a Queue for each service that contains the user requests accepted by this SE. The requests are accepted according to the same QoS model [9] in terms of:

- Availability/Accessibility: the SE is available by the number of ubiquitous service elements and queues deployed to cover the demand (dimensioning).
- Capacity: for the SE, it represents the average processing capacity for a unit of time. For a queue, it represents the size of the queue.
- Delay: for the SE, it represents the average processing time. For the queue, it represents the average waiting time in the queue for each request. It is managed by a Queue_QoS Agent.
- Reliability: for the SE it represents the treatment rate unexecuted. For the queue it represents the requests rate not served in the queue.

The SE_QoS Agent, Queue_QoS Agent and Ubiquitous SE assure the respect of the SLA for each request.

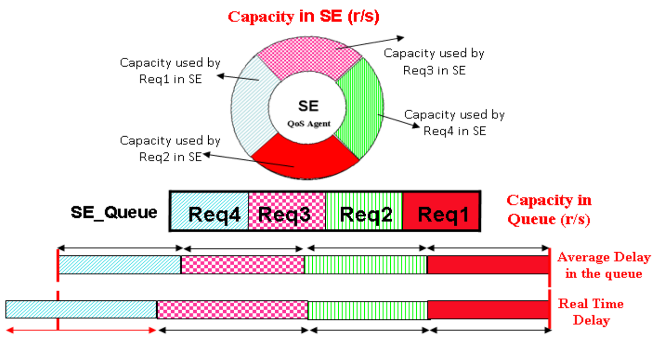

Capacity in SE (r/s)
Capacity used by Req1 in SE
Capacity used by Req3 in SE
SE
QoS Agent
Capacity used by Req4 in SE
Capacity used by Req2 in SE
SE_Queue
Req4
Req3
Req2
Req1
Capacity in Queue (r/s)
Average Delay in the queue
Real Time Delay

Figure 2. "Service Resource" and Mutualisation Example

During the provisioning process, the requests to use a service are put in the SE_Queue that meets their QoS. The advantage of our proposal is that we take into account all the QoS criteria during the provisioning (Figure 2).

## B. *Self-Management of the "Service Resource"*

All requests in the queue have an E2E QoS to respect and a time out warning not to be exceeded. If the waiting time in the queue exceeds a threshold warning, the request is sent towards another equivalent SE which can satisfy its request (Ubiquitous SE). To manage this ubiquity we introduce the concept of community of interest. In our proposition, each ubiquitous SE is part of a Virtual Service Community (VSC) containing multi service provider service elements functionally and QoS equivalent. The advantage of these communities is to anticipate the SLA violation by sending requests that are not served by a SE to another ubiquitous SE in the same platform or in another platform.

In addition, a QoS agent is integrated into each SE in order to ensure an autonomous management of these communities. The QoS Agent in the SE controls the Current QoS of the SE. in case where this Current QoS exceeds the threshold QoS, it sends an "Out Contract" notification to the others SEs in the same VSC to replace it in the VPSN.

Since, we have proposed for each SE a queue which contains the accepted requests by this SE, we have also the Virtual Queue Communities (VQC) which contain the queues of the ubiquitous SEs in the VSC. These VQCs allows us to maintain the SLA of each request in the queue. The advantage to have theses VQC is that we can send the requests not served immediately to the available queue in the community. Then, we can respect the delay of requests.

## C. *Queue management Mechanism*

A service element and its queue admit requests according to the four QoS criteria. The management mechanism of the service element and its queue is represented in Figure 3:

1. The provisioning service of the SE_QoS Agent looks at the Demanded QoS of each received request and compares it with the Current QoS of the SE and will thus decide admission of the request in the queue of the SE. The unaccepted requests will be sent towards another ubiquitous SE.
2. The accepted requests put in the queue for Activation on the SE. Each request in the queue has a "Time Out warning" which represents the deadline of the request not to be exceeded.
3. The Queue_QoS Agent controls dynamically the QoS of the queue. When it detects that a request exceeds its average waiting time in the queue, it sends a notification to the VQC Maintenance.
4. The VQC execute its algorithm to select which requests may expire and then interacts with the data base (INFOWARE) to send the request with expired time out

warning to another ubiquitous queue which has enough QoS.

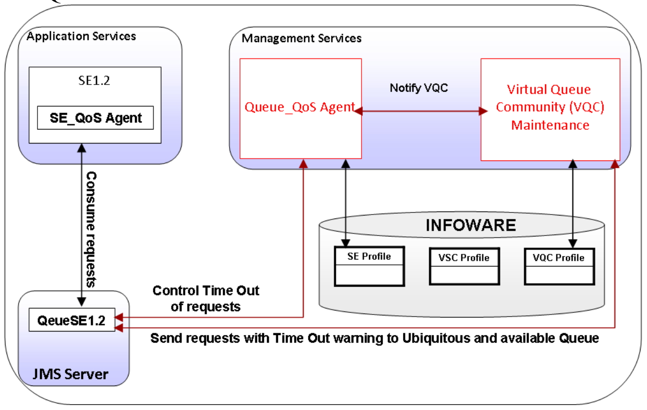


Figure 3. Management Mechanism

## IV. Feasibility

In order to validate our proposition, we have developed Management Services (Queue_QoSAgent and VQC Maintenance) and Application services as independent Enterprise Java Beans (EJBs) [10]. The latter permit the creation of different autonomous and loosely coupled components. EJBs are deployed on Glassfish v3 application server [11]. Glassfish v3 is JEE6 certified and consequently supports different APIs such as JMS, JNDI and JDBC. Our queues are in a JMS Server [12]. We use Open Message Queue (OpenMQ) as our messaging server, which provides a complete JMS implementation. An Open MQ is launched inside each SE resource at the same time when the SE is launched. Each EJB Application Service (our Service Element) has an attached queue and a Message Driven Bean (MDB) which listens to the coming requests in the queue. It consumes requests from his Queue.

We have a Queue_QoS Agent which controls the QoS of the Queue.

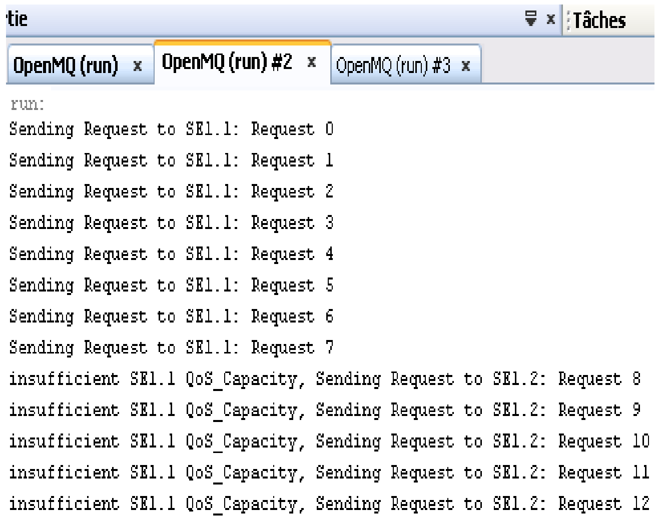
Each SE is part of Virtual Service Community (VSC) grouped by same functionality and equivalent QoS.

Each queue is part of Virtual Queue Community (VQC) which contains the queues of the SEs in the VSC.

The relevant information of these VSCs, VQC and QoS parameters is stored in our Knowledge base: INFOWARE [13] [14].

### A. *Scenario 1: Service Provisioning by current QoS*

When a request arrives, the Provisioning Service will see if the Current QoS of the SE can accept the request in the queue. For example, if SE1.1 can treat 8 requests as maximum according to its Current QoS Capacity, and there are already 8 messages waiting in the queue, the coming requests will be transferred to other ubiquitous SE (in our example, to SE1.2). If the request can be accepted, a JMS Message is created according to the content of the request. Figure 5 gives the result of an example of accepting requests by SE1.1 with respect to QoS_Capacity criteria. In the case of insufficient QoS_Capacity, the requests are sent to SE1.2.



```
run:
Sending Request to SE1.1: Request 0
Sending Request to SE1.1: Request 1
Sending Request to SE1.1: Request 2
Sending Request to SE1.1: Request 3
Sending Request to SE1.1: Request 4
Sending Request to SE1.1: Request 5
Sending Request to SE1.1: Request 6
Sending Request to SE1.1: Request 7
insufficient SE1.1 QoS_Capacity, Sending Request to SE1.2: Request 8
insufficient SE1.1 QoS_Capacity, Sending Request to SE1.2: Request 9
insufficient SE1.1 QoS_Capacity, Sending Request to SE1.2: Request 10
insufficient SE1.1 QoS_Capacity, Sending Request to SE1.2: Request 11
insufficient SE1.1 QoS_Capacity, Sending Request to SE1.2: Request 12
```

Figure 4. Service Provisioning according to Current QoS Capacity

### B. *Scenario 2: Queue Management*

In Figure 5, we have an example where the request “Req3” is in the queue of the service element SE5.5 since its QoS accept this request. The queue already contains three requests (Req0, Req1 and Req2). During usage, the request “Req0” will take more processing time of SE5.5 than what was expected on average. This is due for example to a larger number of data to be processed. The Queue_QoS Agent Detect this delay exceeded and then notifies the VQC Maintenance. In this scenario, the waiting Time of the request “Req3” will exceed its average waiting time. So, “Req3” will be sent to another ubiquitous SE which satisfies the requested QoS of “Req3”. The VQC Maintenance chooses a ubiquitous available queue to send Req3. This allows us to guarantee the SLA of all the accepted requests in the queue. The chosen queue by the VQC Maintenance is Queue_SE5.4.

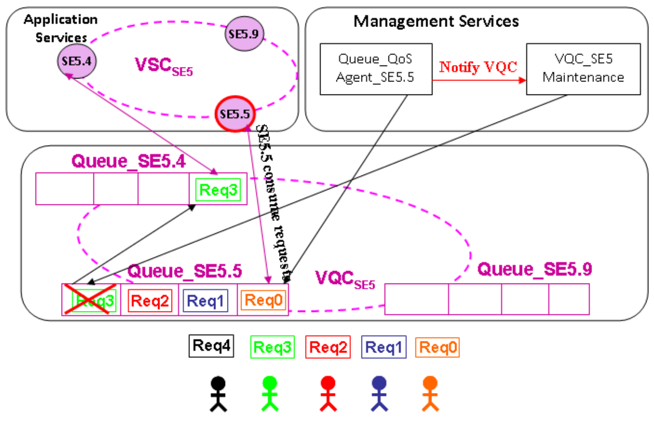


Figure 5. Queue Management Example

## V. Conclusion

In this paper we have described mechanisms that allow service provider to provision the “Service Resource” in the service layer, in similar way as the routing in the network layer. This resource is shared dynamically among several users. To manage the service mutualisation we propose to add a queue in the Service resource. The number of accepted requests in the queue depends on the current QoS of the service resource. Our solution optimizes the usage of service resource by taking into account the four QoS parameters of the SE and its link (queue). We have implemented queue management mechanism by using EJB and JMS. Our future work consists on developing enough service elements on multiple platforms to evaluate and discuss the performance of our solution. For the self-management of our SE, we will perform the QoS Agent automata by taking into account all the events that can receive the SE and how we can treat them.

## ACKNOWLEDGEMENT

Special thanks to Doctor Chunyang YIN and to Master Student Xiaofei XIONG. And thanks to all the participants in the UBIS project, financed by the French ANR VERSO 2008 in which is situated our work.

## References

[1] T. Pollet, G. Maas, J. Marien and A. Wambecq, ‘‘Telecom Services Delivery in a SOA’’, 20th IEEE/ International Conference on Advanced Information Networking and Applications 2006, doi: 10.1109/AINA.2006.322.

[2] M. Boniface, S. Phillips, A. Sanchez-Macian and M. Surridge, “Dynamic Service Provisioning Using GRIA SLAs”, Service-Oriented Computing, NFPSLA-SOC'07, Sept 2007, Vienna

[3] X. Li, H. M. Chan, T. Hung, K. H. Tong and S. J. Turner, “Design of an SLA-driven QoS Management Platform for Provisioning Multimedia Personalized Services”, AINAW '08 Proceedings of the 22nd International Conference on Advanced Information Networking and Applications, doi: 10.1109/WAINA.2008.256

[4] S. Chen, J. Lukkien, R. Verhoeven, P. Vullers and G. Petrovic, “Context-Aware Resource Management for End-to-End QoS Provision in Service Oriented Applications”, IEEE GLOBECOM Workshop 2008, pp. 1-6, doi: 10.1109/GLOCOMW.2008.ECP.50

[5] J.W. Kim, S.W. Han, D.H. Yi, N. Kim, and C.C.J. Kuo, “Media-Oriented Service Composition with Servic Overlay Networks: Challenges, Approaches and Future Trends”, Journal of Communications, Vol. 5, No .5. (2010), pp. 374-389, May 2010. doi:10.4304/jcm.5.5.374-389.

[6] M. Stiermerling et al., "System Design of SATO & ASI", IST Project Ambient Networks, D12, http://www.ambientnetworks.org/Files/deliverables/D12-F.1_PU.pdf

[7] M. Song, and B.Mathieu, “QSON: QoS-aware Service Overlay Network”, CHINACOM 2007, pp. 739 – 746, doi: 10.1109/CHINACOM.2007.4469494

[8] E. Kyu Kim, Y. Kim and I. Chong, “Architectural model and service scenario of dynamic service overlay network (DSON)”, First International Conference on Ubiquitous and Future Networks, ICUFN 2009, pp. x – xiii, doi: 10.1109/ICUFN.2009.5174273

[9] N. Simoni, B. Mathieu, C. Yin and M. Song, “ Autogestion de service par la QoS dans un Réseau Overlay ”, GRES 2007, Tunisia

[10] R.P. Sringanesh, G. Brose and M. Silverman: Mastering Entreprise JavaBeans 3.0. Wiley Publishing Inc. , 2006

[11] SUN Microsystems, https://glassfish.dev.java.net

[12] Open MQ, https://mq.dev.java.net/

[13] N. Ornelas, N. Simoni, K. Chen, and A.Boutignon, “VPIN: User-session knowledge base for self-management of ambient networks”, UBICOMM’08, pp. 122-127, Oct. 2008, Spain, doi: 10.1109/UBICOMM.2008.71.

[14] N.Ornelas, N.Simoni, C.Yin, and A.Boutignon, “VPIN: An event based knowledge inference for a user centric information system”, Journal On Advances in Internet Technology, Vol. 2, N° 1, pp. 29-44, Sept. 2009.