Quality of Service APIs for a 3GPP Mobile and Pervasive Large Scale Augmented Reality Gaming Middleware

Pedro Ferreira, João Orvalho and Fernando Boavida
Centro de Informática e Sistemas, Department of Informatics Engineering,
University of Coimbra, Pólo II, Pinhal de Marrocos, 3030-290, Coimbra, Portugal
{pmferr,orvalho,boavida}@dei.uc.pt

Abstract

Ubiquitous or pervasive computing is a new kind of computing, where specialized elements of hardware and software will have such a high level of deployment that their use will be fully integrated with the environment. Augmented reality extends reality with virtual elements but tries to place the computer in a relatively unobtrusive, assistive role. Specialized network middleware solutions for large scale mobile and pervasive augmented reality games are, to our knowledge, inexistent. The work presented in this paper focuses on the creation of such type of network middleware for mobile and pervasive entertainment, applied to the area of large scale augmented reality games.

In this paper we propose and describe APIs and architecture for Quality of Service specification, negotiation and provision on the client side (using J2ME) and on the server side (using J2SE). The paper discusses architectural and implementation aspects.

Keywords. Quality of Service, Pervasive Computing, 3GPP, Augmented Reality, Mobile Gaming

1. Introduction

Mark Weiser theorized about a new kind of computing, called ubiquitous or pervasive computing, where specialized elements of hardware and software would be so ubiquitous no one would notice their presence [1]. According to Mark Weiser the technology required for ubiquitous computing would come in three parts: inexpensive, low power computers including equally convenient displays, software for ubiquitous applications, and networks that tie them all together.

In the current decade we are witnessing the merging of telecommunications and IT worlds [2]. The Internet Protocol (IP) is the network layer protocol in the 3GPP specifications, and the current trend in developing new telecommunications networks is to utilize Internet protocols. So, the network that ties all things together is now possible. But there are many issues under study in the Internet community. These are mobility, quality of service, security, management of networks and services, discovery, ad - hoc networking and dynamic configuration, and geospatial location.

Low cost, low power computers including equally convenient displays are also coming closer to reality. In fact, we can consider the latest PDA's and mobile phones an early version of Weiser's ubiquitous computers.

A significant requirement of pervasive applications is fast service development and deployment [2], which implies the introduction of various service and application frameworks and platforms. For this, middleware is a common solution. The benefits of middleware utilization are the improved programming model, and the hiding of many implementation details, which make middleware based application development much faster.

It is now becoming quite clear that entertainment, and more specifically mobile gaming, will be one of the killer applications of future wireless networks [3]. However, mobile gaming applications face issues that are different from fixed network applications. These issues include fluctuating connectivity, quality of service and host mobility. Another issue is how to manage game state consistency with a dynamic mobile networked environment in which devices may be physically close but topologically distant. Further yet, there is the issue of how to manage multiple wireless network connections such as, for example, GPRS and IEEE 802.11 at the same time.

Augmented reality extends reality with virtual elements while keeping the computer in an assistive, unobtrusive role [4]. It is possible to create games that place the user in the physical world through geographically aware applications. The latest mobile phones are being equipped with GPS receivers and there are software and hardware tendencies from the largest manufacturers to equip mobile phones with more advanced context-aware technology. Current mobile phones are equipped with cameras and some of the latest ones are coming with some form of 3D rendering technology [5][6]. Bluetooth technology and increasing miniaturization will allow, in the near future, specialized pervasive equipment for augmented reality. The opportunity for some inexpensive augmented reality is here.

To the best of our knowledge, there is no specialized network middleware solution for large-scale mobile and pervasive augmented reality games. The main objective of this work is the creation of such network middleware for mobile communications that will enable integrated large-scale augmented reality applications to be built around it.

The middleware that is being created evolved from previous work in the area of interactive distributed multimedia, more specifically in state transmission for a collaborative virtual environment middleware platform, the Status Transmission Framework (STF) [7][8]. This platform extended ARMS – Augmented Reliable CORBA Multicast System [9][10] – with capabilities for the handling of state transmission in distributed collaborative virtual environments.

In this context mechanisms are being studied, proposed and evaluated to deal with issues such as Mobility (fluctuating connectivity, host mobility and handling of multiple simultaneous network connections), quality of Service – QoS (minimizing delay and jitter ,and reliability), security (authentication and prevention of cheating), management of Networks and Services, discovery, ad-hoc networking and dynamic configuration, geospatial location and orientation, scalability, consistency, multimedia data heterogeneity, data distribution and replication.

The architecture we are talking here has been partly published (without the QoS handling capabilities) in previous work. In [11] we talked about the general architecture we were thinking of building, in seminal terms, in [12] we specified a little more about the architecture that was going to be built. In [13] we introduced a sensor-actuator personal area network controller API for sensors and actuators based on Java CLDC and Java Bluetooth, and the corresponding API on the central coordinating device of that personal area network.

In [14] we described a new reliable multicast protocol capable of working in the many to many scenario, nak based, that avoided duplicated and was source based, and that worked on ipv4 or ipv6 – sixrm.

This protocol is the base for communication between distributed servers, as part of ARMSV6, a corba event system extended to use multicast.

In [15] we completely describe the architecture of the system already built and working, and tested, still without QoS.

This paper concentrates on the QoS handling issues of the middleware, that are a new contribution to Java 2 Micro Edition and to our knowledge, also to J2SE, and to our system.

The main contribution of this paper is the definition of an QoS handling architecture both on the client (UE) and on the distributed gaming server.

To our knowledge, this is the first java API for Quality of Service handling in 3GPP networks ever proposed.

In the rest of the paper, first we address quality of service in 3GPP, after which we describe the general architecture of the STF QOS API on J2ME (client UE). We then proceed to discuss the general architecture of the STF QOS API on the distributed servers, and immediately after that the functional tests to which the implementation was subject and the carried out simulations. Subsequently, we discuss the integration of the API with the rest of the STF architecture, following which we present some conclusions and guidelines for further work.

2. Quality of Service in 3GPP

In this section we briefly introduce the 3GPP End-to-End QoS architecture.

2.1 General Architecture

The 3GPP general architecture for End-to-End QoS is relatively complex. Basically, QoS is present in all levels of UMTS, from the radio protocols to the higher level voice and packet data protocols. And in this way 3GPP guarantees end-to-end QoS in UMTS. For more details, please see [16]. Here, we just describe the more important elements for our API.

2.2 The role of the PDP context

The PDP (Packet Data Protocol) Context is a virtual link between the UE and the GGSN, passing by the SGSN, that transports packets of some protocol, usually IPv4 or IPv6. The PDP Context is defined by some properties like the QoS profile it uses, the APN

(Access Point Name) it connects to, and the type of packets it transports.

By changing the QoS profile, we change the QoS properties of the connection.

2.3 The role of RSVP

Resource Reservation Protocol (RSVP) [21] and related standards [17][18][19][20] may be used on the UE client side for QoS resource reservation after initial secondary PDP Context activation and influence the way the network will handle packets for our flows of media. They may also be used on the server side to handle reservations.

3. General architecture of the STF QOS API for J2ME

We now discuss the general architecture of the Status Transmission Framework version 2 QoS API on J2ME (that is, on the UE, the central coordinating device of the personal area network).

3.1 General Architecture

The general architecture is based on two APIs for QOS: The PDP Context Handler API and the RSVP API. Both are used at the same time to guarantee quality of service on a 3GPP network with RSVP support (witch is optional) and that may or may not use uses Service Based Local Policy between the PDF (Policy Decision Function) and the GGSN as specified on [16].

3.2 The PDP Context handler API

The PDP Context handler API is an Application Programming Interface that allows us to activate and deactivate PDP contexts with all its characteristics including QoS characteristics.

Table 1 shows the classes of package pt.uc.dei.lcst.stf.pan.qos, the package of the PDP Context handling API and corresponding functions.

3.3 The RSVP API

The RSVP API on the UE (J2ME) is based on the RSVP specifications [17][18][19][20][21] and is used to alter the way the GGSN in particular (if it supports RSVP) and other routers on the way to the distributed servers (which are located on the IP Multimedia Subsystem) allocate resources to the connection in question.

The GGSN affects (see [16]), if supporting RSVP, the PDP context traffic handling.

Table 2 shows the classes of package pt.uc.dei.lcst.stf.pan.qos.rsvp, the package of the RSVP API and corresponding functions. This package belongs to the bigger STFPAN API.

Class	Function
DestinationPortRangeType	Defines a port range
FlowLabelType	Defines a Ipv6 flow label
IPV4SourceAddress	An Ipv4 Source Address
IPV6SourceAddress	An Ipv6 Source Address
PDPContext	A PDP Context per se
PDPContextListener	Listener for event related to PDP Contexts
PDPContextManager	The manager of PDPContexts
PDPPacketFilter	Base class of all Packet filters
PDPTrafficFlowTemplate	A PDPContext traffic flow template
QoSException	An exception occurred
QoSProfileIE	The QoS Specification
ProtocolIDNextHeaderType	A Protocol ID NextHeader
SecurityParameterIndexType	A Security Parameter Index
SingleDestinationPortType	A single destination port.
SingleSourcePortType	A single source port.
SourcePortRangeType	A source port range.
TypeOfServiceTrafficClassType	A type of service traffic class.

Table 1 - Classes of pt.uc.dei.lcst.stf.pan.qos

4. General architecture of the STF QOS API on the distributed servers

On the distributed servers, we use RSVP (Resource Reservation Protocol) and related standards [21][17][18][19][20].

4.1 The RSVP Architecture

The RSVP architecture on the distributed servers is very similar to the architecture on the UE clients, with differences in implementation and in configuration of course. But the list of classes is the same as in Table 2, except that the package is now pt.uc.dei.lcst.stf.qos.rsvp that is part of the bigger STFServer API.

Class	**Function**
RSVPMessage	Base message
RsvpException	A RsvpException
RsvpExceptionNoService	RsvpException: No service available
RsvpListener	Listener of messages
RsvpManager	Rsvp manager
RsvpObject	Base object
RsvpObjectForwarded	Forwarded object
RsvpObjectIgnored	Ignored object
RsvpObjectIntServAdSpec	AdSpec object
RsvpObjectIntServFlowSpecObject	FlowSpec object
RsvpObjectIntServerSenderTSpec	TSpec object
RsvpObjectIntegrity	Integrity object
RsvpObjectIpv4ErrorSpec	Ipv4 Error Spec object
RsvpObjectIpv4FilterSpec	Ipv4 Filter Spec object
RsvpObjectIpv4ResvConfirm	Ipv4 ResvConfirm object
RsvpObjectIpv4RsvpHop	Ipv4 RsvpHop object
RsvpObjectIpv4ScopeList	Ipv4 ScopeList object
RsvpObjectIpv4SenderTemplate	Ipv4 SenderTemplate object
RsvpObjectIpv4UDP	Ipv4 session object
RsvpObjectIpv6ErrorSpec	Ipv6 ErrorSpec object
RsvpObjectIpv6FilterSpec	Ipv6 FilterSpec object
RsvpObjectIpv6ResvConfirm	Ipv6 ResvConfirm object
RsvpObjectIpv6RsvpHop	Ipv6 RsvpHop object
RsvpObjectIpv6ScopeList	Ipv6 ScopeList object
RsvpObjectIpv6SenderTemplate	Ipv6 SenderTemplate object
RsvpObjectIpv6UDP	Ipv6 Session object
RsvpObjectList	List of objects
RsvpObjectNull	Null object
RsvpObjectPolicyData	PolicyData object
RsvpObjectStyle	Style object
RsvpObjectTimeValues	TimeValues object
RsvpPATH	PATH message
RsvpPathErr	PATHERR message
RsvpPathTear	PATHTEAR message
RsvpRESV	RESV message
RsvpResvErr	RESVERR message
RsvpResvTear	RESVTEAR message
RsvpResvConf	RESVCONF message

Table 2 - classes of package pt.uc.dei.lcst.stf.pan.qos.rsvp

5. Functional tests

To test our architecture we have made some functional tests, on a simulated implementation of the PDP context Handler architecture and on a protocol implementation of RSVP.

5.1 Emulation of the PDP Context Handler architecture

We did not have access to a platform where we could implement real PDP context activation and deactivation, in a way that we could test it, in Java. So, we emulated the API, implementing all its functionality internally in such a way that applications can be made in the emulator using this API and in the future, a real implementation (not emulated), probably using a native interface to the native features of real UEs, can really allocate and deallocate PDP contexts.

Using this emulation environment, functional tests were made to the proposed API, enabling us to confirm its operational capabilities.

5.2 RSVP implemented by UDP encapsulation with a simulated router in between

As for RSVP, we implemented RSVP, both on the emulated UE and on the distributed server using UDP encapsulation, which is a feature of the protocol [21].

Our implementation does not support the features of integrity checking (optional in all messages), and policy data (stated in the RFC as further study item).

A future alternative implementation could implement these items.

For testing, we built a program that simulated a router with UDP encapsulation support and tested in an isolated fashion the communications of the RSVP protocol between a program that sent and received data (so it needed reservations), an RSVP simulated router, and a similar program (actually the same program) running on another machine.

The functional tests consisted of, considering both computers as senders and receivers at the same time, setting up reservations in RSVP having a router in between the two (actually, the simulated router).

The tests only targeted the protocol, no real reservations at the network layer were made.

The tests were successful and we proceeded with the integration of the API with the rest of the Status Transmission Framework Middleware.

6. Integration of the API with the STF middleware

We now describe the process of integrating the QoS APIs on the rest of the Status Transmission Framework version 2.0 Middleware.

6.1 On the PAN

On, the PAN, that is, on the STFPAN API, we added the package pt.uc.dei.lcst.stf.pan.qos and the package pt.uc.uc.dei.lcst.stf.pan.qos.rsvp to the already existent other packages of the STFPAN API.
Then we altered the session initiation sequence to include QoS negotiation with PDP activation and modification and RSVP negotiation.
We also altered the sequence of session termination to include RSVP resource freeing.

6.2 On the Distributed Servers and Central Server

On the distributed servers and central server we added to the STFSERVER API the pt.uc.dei.lcst.stf.qos.rsvp package.
Then we altered the sequence of session initiation and termination.

6.3 Session Management with our APIs for QoS

Session initiation and termination with our APIs for QoS is done in the way specified in [16] when using RSVP. The only change is that we have a central application server that coordinates the distributed server, that does all the RSVP handling.

7. Conclusions

In this paper we have proposed a set of APIs for QoS handling in both the 3GPP UE (in J2ME) and the distributed server of a middleware for mobile and pervasive large scale augmented reality games previously proposed.
We provided a simulated implementation of PDP contexts and a real implementation of RSVP through UDP encapsulation and functionally tested the RSVP protocol.
The tests that were carried out enabled us to conclude that the API is adequate for our platform over 3GPP networks according to the latest specifications of 3GPP End-To-End Quality of Service.
Future work on this API will consist of implementing, testing, and evaluating it on real devices.

Aknowledgments. This work was partially financed by the Portuguese Foundation for Science and Technology (FCT) and the FP6 CONTENT Network of Excellence (NoE)

8. References

[1] M. Weiser, "The Computer for the Twenty - First Century", Scientific American, pages 94–104, Sept. 1991.

[2] Kimmo Raatikainen, Henrik Bærbak Christensen, Tatsuo Nakajima, "Application Requirements for Middleware for Mobile and Pervasive Systems", Mobile Computing and Communications Review, Volume 6, Number 4, October 2002, pp. 16 – 24 , ACM Press

[3] Keith Mitchell, Duncan McCaffery, George Metaxas, Joe Finney, Stefan Schmid and Andrew Scott, "Six in the City: Introducing Real Tournament – A Mobile IPv6 Based Context-Aware Multiplayer Game", Proceedings of NetGames'03, May 22-23, 2003, Redwood City, California, USA, pp. 91-100, ACM Press

[4] Hideyuki Tamura, Hiroyuki Yamamoto, and Akihiro Katayama, “Mixed Reality:Future Dreams Seen at the Border between Real and Virtual Worlds”, Virtual Reality, November/December 2001, pp. 64 –70, IEEE

[5] Nokia – Developer resources (Forum Nokia), http://www.forum.nokia.com/, Accessed April 2004

[6] Sony Ericsson Developer World, http://developer.sonyericsson.com/, Accessed April 2004

[7] João Orvalho, Pedro Ferreira and Fernando Boavida, “State Transmission Mechanisms for a Collaborative Virtual Environment Middleware Platform”, Springer-Verlag, Berlin Heidelberg New York, 2001, pp. 138-153, ISBN 3-540-42530-6 (Proceedings of the 8th International Workshop on Interactive Distributed Multimedia Systems – IDMS 2001, Lancaster, UK, September 2001)

[8] Pedro Ferreira, “State transmission in distributed, collaborative, virtual reality environments”, M.Sc. thesis, Universidade de Coimbra - FCTUC – Department of Informatics Engineering, October-2002

[9] João Orvalho, Fernando Boavida, “Augmented Reliable Multicast CORBA Event Service (ARMS): a QoS-Adaptive Middleware”, in Lecture Notes in Computer Science, Vol. 1905: Hans Scholten, Marten J. van Sinderen (editors), Interactive Distributed Multimedia Systems and Telecommunication Services, Springer-Verlag, Berlin Heidelberg, 2000, pp. 144-157. (Proceedings of IDMS 2000 – 7th International Workshop on Interactive Distributed Multimedia Systems and Telecommunication Services, CTIT / University of Twente, Enschede, The Netherlands, October 17-20, 2000).

[10] João Gilberto de Matos Orvalho, “ARMS – Uma plataforma para aplicações multimédia distribuídas, com qualidade de serviço”, Phd Thesis, December 2000, DEI-FCTUC

[11] Pedro Ferreira , “Network Middlware for Large Scale Mobile and Pervasive Augmented Reality Games”, in Proc. of the CoNext 2005 - ACM Conference on Emerging Network Experiment and Technology, pp. 242-243, CoNext 2005 - ACM Conference on Emerging Network Experiment and Technology, Toulouse, France, October 2005

[12] Pedro Ferreira and João Orvalho and Fernando Boavida, Large Scale Mobile and Pervasive Augmented Reality Games, in Proc. of the EUROCON 2005 - The International Conference on "Computer as a Tool", pp. 1775-1778, Vol. 1, # 1, EUROCON 2005 - The International Conference on "Computer as a Tool", Belgrade, Serbia and Montenegro, November 2005

[13] Pedro Ferreira, João Orvalho and Fernando Boavida, “Middleware for embedded sensors and actuators in mobile pervasive augmented reality”, in Proc. of the INFOCOM 2006 (IEEE XPLORE), INFOCOM 2006 Student Workshop, Barcelona, April 2006

[14] Pedro Ferreira and João Orvalho and Fernando Boavida, “Sixrm: Full Mesh Reliable Source Ordered Multicast”, in Proc. of the SoftCom2006 - 14th International Conference on Software, Telecommunications & Computer Networks, SoftCom2006 - 14th International Conference on Software, Telecommunications & Computer Networks, Split, Croatia, September 2006

[15] Pedro Ferreira, João Orvalho and Fernando Boavida, “A middleware architecture for Mobile and Pervasive Large Scale Augmented Reality Games”, to be published in Proceedings of the Conference on Networks and Services Research (CSNR) 2007, Fredericton, New Brunswick, Canada, May 2007

[16] 3GPP TS23.207 V6.0.0, “3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; End to End Quality of Service (QoS) concept and architecture (Release 6)”, September 2005

[17] J. Wroclawski, “The Use of RSVP with IETF Integrated Services”, RFC2210, IETF Network Working Group, September 1997

[18] S.Shenker, J. Wroclawski, “General Characterization Parameters for Integrated Service Network Elelements”, RFC 2215, IETF Network Working Group, September 1997

[19] J.Wroclaswki, “Specification of the Controlled-Load Network Element Service”, RFC 2211, IETF Network Working Group, September 1997

[20] S.Shenker, C.Partridge, R. Guerin, “Specification of Guaranteed Quality of Service”, RFC2212, IETF Network Working Group, September 1997

[21] R. Braden (Ed.), L.Zhang, S. Berson, S. Herzog, S. Jamin, “Resource Reservation Protocol – version 1 Fuctional Specification”, RFC 2205, IETF Network Working Group, September 1997