A Middleware Architecture for Mobile and Pervasive Large-Scale Augmented Reality Games[1]

Pedro Ferreira, João Orvalho and Fernando Boavida
LCT – Laboratory of Communications and Telematics
Centre of Informatics and Systems of Coimbra University
Polo 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 describe the architecture of the system, which works over a 3GPP network. The paper presents architectural and implementation aspects, along with evaluation results. The results show that the proposed solution is able to cope with the stringiest requirements of augmented reality systems.

1. Introduction

Mark Weiser [1] 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. According to Mark Weiser [1] 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.

[1] This work is being partly financed by the Portuguese Foundation for Science and Technology (FCT) and CONTENT NoE.

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.

This paper concentrates on the architectural issues of the proposed middleware, with emphasis on the validation of the main architectural choices, having in mind the requirements identified in the next section.

The main contribution of this paper is the definition of an architecture of a middleware for large scale mobile and pervasive augmented reality games and the steps taken to evaluate this architecture.

2. Requirements

Pervasive Large Scale Augmented Reality Game Applications have many requirements.

Reliable multicast of packets is a requirement of large scale distributed entertainment applications. However, it has been shown that current solutions for that effect have significant problems [11], examples of those being the nak or ack explosion problems. The reliable multicast transport working group of IETF [12] has been addressing the problem of reliable multicast transport, but only for one-to-many scenarios, leaving out many-to-many configurations. The experimental protocol PGM [13], the only many-to-many protocol that has reached RFC status in IETF, requires support by network elements. This is hard to get implemented in reality.

Delay, jitter and bandwidth are also crucial for augmented reality applications. These parameters are at the base of the experienced audio-visual quality as well as other sensitive information.

Requirements that are common to all system levels are security, trust, privacy and accounting (Authentication, Authorization and Accounting). In addition, manageability is another important requirement, as large scale systems must remain under effective control, in order to avoid the risk of global breakdown.

These requirements are taken into account in the architecture proposed in this paper and presented in the following section.

3. Architecture

The system targeted by the proposed middleware is composed of 3 levels: the back-office central level, the large scale network level, and the personal area network level.

The back-office central level consists of one or more of a series of parallel servers and serves as the main controlling station of the game administrator, the person responsible for starting, stopping and managing game performance and general maintenance tasks.

The large-scale network is the standard 3GPP network, where servers are distributed according to some logic of spatial distribution, typically corresponding to aggregations of cells of the mobile communications network.

The personal area network level consists of the network of pervasive devices dedicated to personal communications and to augmenting reality, which the person carries. These may be sensors, actuators, and other devices that can communicate using Bluetooth or other means of communication. All these communicate with the mobile host, probably just a cell phone or specialized device connected to the large-scale 3GPP network. In this way, the player is so enabled to play games of augmented reality irrespective of his/her location.

Targeting this architecture allows the study, evaluation and proposal of mechanisms to deal with issues of scalability, multimedia data heterogeneity, data distribution and replication, consistency, security, geospatial location and orientation, mobility, quality of service, management of networks and services, discovery, ad-hoc networking and dynamic configuration.

We consider that building augmented reality

applications using a network middleware is better that building them standalone. This is because then many game applications may then use the same application programming interface (API) to leverage network resources, giving it much faster service development and deployment.

The middleware presented in this paper is being built according to the characteristics of agile pervasive middleware [14], such as application-awareness, mobility, integration, interoperability, scalability, portability, adaptability, robustness and simplicity of evolution.

4.1 Central level

At the central level, there is one server, which may be constituted by more than one parallel server, running Java Standard Edition 1.5.0. There will also be database servers, which may or may not be integrated with the same server.

This server or collection of servers will be connected to the HSS (Home Subscriber Server) of the 3GPP Network by the DIAMETER protocol SH application and are, together, an IMS (IP Multimedia Subsystem) application server.

All authentication, accounting, and authorization will happen through this interface. All management of the game servers will happen through this server.

Status Transmission Framework version 2.0 APIs for the server side include a DIAMETER [15] API which includes the base protocol, the CX and DX [16] applications and the SH applications [17] of 3GPP. This would communicate preferably through SCTP [18][19] (we also developed a java SCTP API that presently only works under Linux, but can be easily extended to other platforms, as soon as those platforms support SCTP natively) if available. If not, TCP will be chosen. The DIAMETER API implementation supports TLS [20] and works over IPSec.

The terminal (UE) from the personal area network will communicate with the central server through SIP [21] to initiate the session, authenticate itself and get the details for the session through SDP [22] negotiation (that's another API we have developed, the J2ME SDP API - in the server side we use JAIN SDP API based on JSR 141[23]).

The API we use on the server side for SIP communication is JAIN SIP [24].

The SIP and SDP exchanges include enough information to choose a distributed server to communicate with, according to the terminal's geographical location. The terminal geographical location is acquired through the use of the J2ME Location API (JSR 179) [25] on the mobile terminal.

The schematics of the middleware architecture the central level are represented in Figure 1.

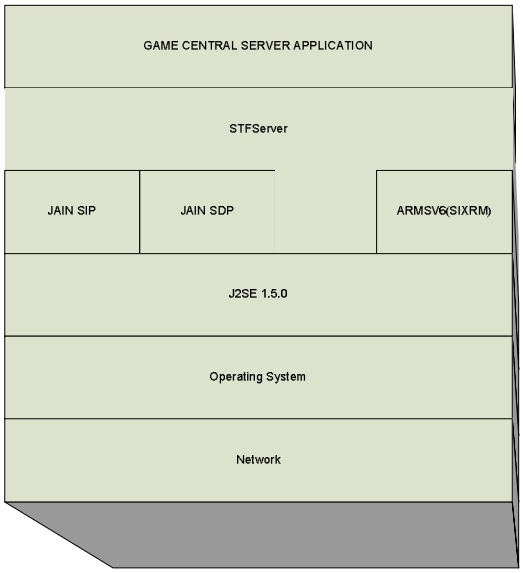

Figure 1 - Central level architecture

4.2 Large Scale Distributed Level

At the distributed server level, there are multiple distributed servers, linked to geographical coverage areas which in the extreme may even be linked to the cells of the mobile network, which will distribute the load off the main server.

These servers run Java Standard Edition 1.5.0, also. They will have integrated database servers running on the same or different computers.

These servers will be interconnected by a reliable multicast protocol capable of working in an IPv6 network, without the support of network elements, capable of working in the many-to-many scenario, without the nak implosion problem but nak based, source ordered and avoiding duplicates: The Sixrm Protocol [26]. This protocol will also connect these servers with the central server.

The schematic of each of the distributed servers on the large scale distributed level is presented in Figure 2.

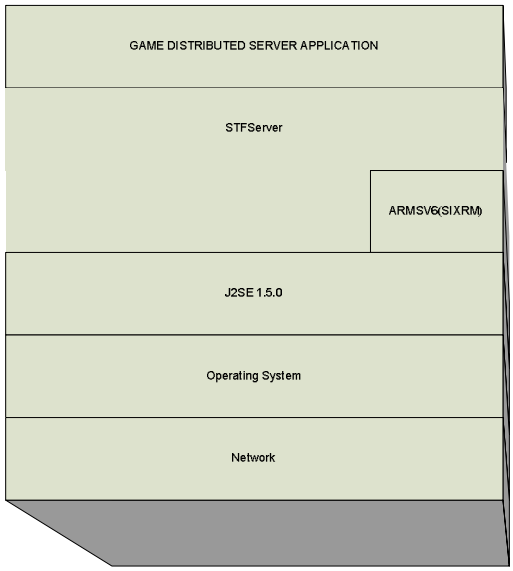

Figure 2 - Large Scale Distributed Level Architecture

4.3 Personal Area Network Level

At the personal area network level we will find the most diversified types of devices. The main device will probably be a cell phone or a specialized device for game playing.

The required characteristics for this device is that it must support the Java language, more specifically, Java Micro Edition, in its Connected Limited Device Configuration (CLDC) version 1.1 [27], and the MIDP – Mobile Information Device Profile - version 2.0 [28].

This central device must support also the Java Bluetooth API (JSR-82) [29], the Java SIP (Session Initiation Protocol) API for J2ME (JSR-180) [30] and the location API for J2ME (JSR-179) [25].

Other devices that are needed on the personal area network level are input and output devices. These devices must also support at least Java (same version and configuration) and the Bluetooth API [29].

Output devices are essentially video and audio output devices. Video and audio output devices should also support, besides Java (CLDC 1.1) and Bluetooth for Java Micro edition (JSR-82), the Mobile 3D graphics API (JSR-184)[31], and the Mobile Media API for J2ME (JSR-135) [32].

As for input devices, in the real world environment, the user is often used to using one or both hands to perform a task. Therefore, the input devices used with wearable computers need to be designed with this requirement in mind. Appropriate input devices need to be utilized to allow the user to efficiently manipulate and interact with objects. For data entry or text input, body mounted keyboards, speech recognition software, or hand held keyboards are often used. Devices such as IBM's Intellipoint, trackballs, datagloves, etc., are used to take the place of a mouse to move a cursor to select options or to manipulate data. One of the main advantages of using a wearable computer is that it allows the option of hands free use.

Common factors in the design of input devices are that they all must be unobtrusive, accurate, and easy to use on the job.

In order for any digital system to have an awareness of and be able to react to events in its environment, it must be able to sense the environment.

This can be accomplished by incorporating sensors, or arrays of various sensors (sensor fusion) into the system. Sensors are devices that are able to take an analogue stimulus from the environment and convert it into electrical signals that can be interpreted by a digital device with a microprocessor.

For a sensor or array of sensors to be supported by the Status Transmission Framework version 2.0, it must be accompanied by hardware that translates its electrical impulses to digital signals transmitted over Bluetooth communications over the personal area network to the central device.

The central device will coordinate all the augmented reality experience for the user, using all the multimedia capacities of the other devices and eventually, even own multimedia capacities of the central personal area network device.

Here, we have developed the Status Transmission Framework version 2.0 PAN API and the SENSACT API [33].

Fig.3 shows the minimum required architecture for a sensor , fig 4 shows the minimum required architecture for a actuator, and fig 5 shows the minimum required architecture for the central game playing device.

5. Sixrm reliable multicast

Reliable multicast of packets is a requirement of large scale distributed entertainment applications. However, it has been shown that current solutions for that effect have significant problems [10], examples of those being the nak or ack explosion problems. The reliable multicast transport working group of IETF [11] has been addressing the problem of reliable multicast transport, but only in the one- to-many approach, not in the many-to-many area. The experimental protocol PGM [12], the only many-to-many protocol that has reached RFC status in IETF,

requires

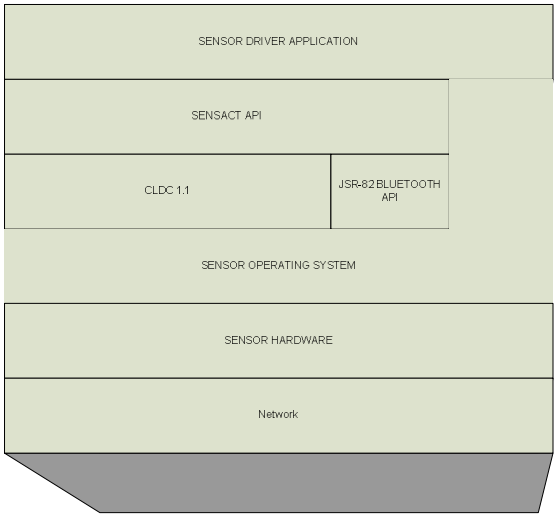

Figure 3 - Sensor architecture (Personal Area Network level)

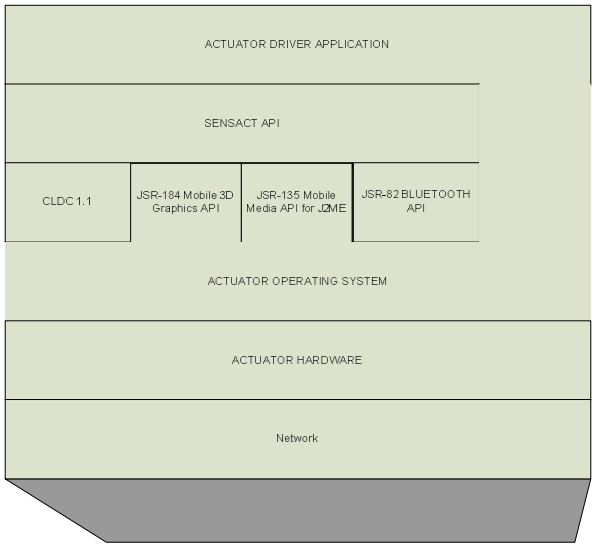

Figure 4 - Actuator architecture (Personal Area Network level)

support by network elements. This is hard to get implemented in reality. We felt it was necessary to create a protocol capable of working in the many-to-many scenario, without the nak implosion problem but nak based, source ordered and that avoided duplicates. We also needed a protocol that could work in ipv6, and that does not require the support of network elements. So, we created Sixrm. We use it for communications between the large scale distributed servers and between those servers and the central server.

Sixrm is described extensively in [26].

6. The SENSACT API

The SENSACT API, which was introduced in [33], is the part of the system which enables the sensors and actuators to be deployed with the help of java CLDC 1.1 and communicate by Bluetooth with the help of JSR-82. This API and correspondent protocols make part of the personal area network level of the system.

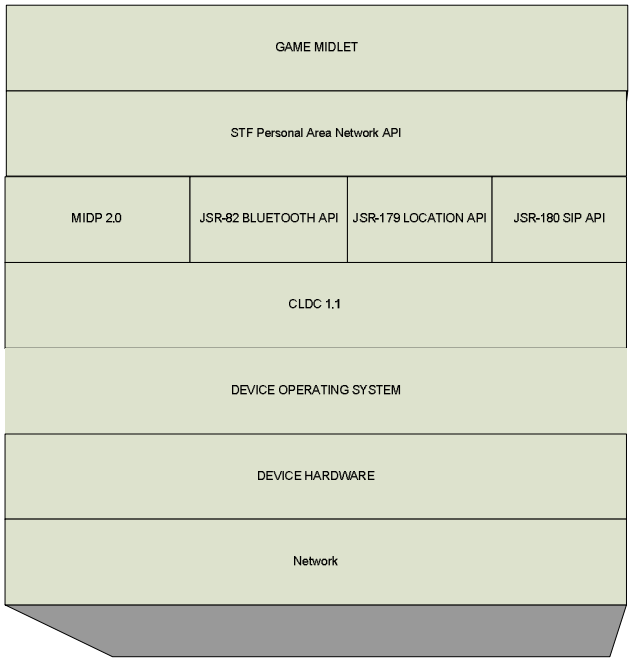

Figure 5 - Central Game Device Architecture (Personal Area Network level)

The SENSACT API is a small footprint set of classes occupying less than 60KBytes, constituted by 5 java packages.

In order to deploy one sensor/actuator, one has to have the respective hardware, a CLDC 1.1 implementation that runs on the hardware, a Bluetooth API (JSR-82) implementation for that hardware and platform, and the SENSACT API.

7. The STF Personal Area Network API

The STF – Status Transmission Framework – version 2.0 PAN – Personal Area Network level API is the correspondent API on the central game device of the user that communicates with the SENSACT API on the sensors and actuators.

This is the API on the central personal area network device that communicates with the distributed servers on the large scale network level of the system. We describe here the packages responsible for the personal area

network communications, which are responsible for communicating with the SENSACT API on the sensors and activators, and which functions are listed in Table 1.

Session Description Protocol [22] is used in SIP[21] messaging to the central server and to the IMS – IP Multimedia Subsystem [35] of 3GPP to negotiate session parameters and QoS. For this, we make use of Java SIP-API (JSR-180) [30] and the help of our developed SDP helper classes in pt.uc.dei.lcst.stf.pan.sdp .

8. ARMSV6

ARMS – The Augmented Reliable corba Multicast System [9][10] – was extended to work in IPV6 in large scale networks by substituting, in its new version ARMSV6, the reliable multicast protocol it used by the Sixrm protocol.

Now, the distributed servers communicate using ARMSV6, and so, the Sixrm reliable multicast protocol.

ARMS [9][10] is a improvement for the corba event service that, maintaining compatibility with the standard corba event service, adds reliable multicast communication to it. ARMSV6 does the same thing but now supporting ipv6 through Sixrm reliable multicast protocol.

9. STFServer

The middleware that runs on the central level and on the large scale distributed server level is the STFServer API. This API makes use of ARMSV6 (and so, Sixrm).

It is an API that runs on top of J2SE 1.5.0 and its objective is to run on the IMS – IP Multimedia Subsystem core [34][35][36]– as an Application Server.

The central server is an application server, the distributed game servers are also application servers.

Table 2 shows the STFServer API packages and corresponding functions.

10. Tests

In this section we present experimental results concerning several key system components, namely results for the personal area networks APIs, for the large scale network APIs, and for common system APIs.

10.1 Personal Area Network APIs

The personal area network part of the system, both the STF PAN API and the STF SENSACT API, was subject to extensive functional and performance tests, with various kinds of simulated sensors and actuators and a simulated reading and actuating application using Java Wireless Toolkit 2.3 Beta from Sun Microsystems running in a series of emulators in a Pentium 4 3.6 GHz System with 1 Gb Memory. We present here results for delay and jitter for two of those tests, one for a sensor, a simple simulated position sensor, and one for an actuator, a simple force actuator.

The tests ran for about ten minutes. The sensor sent values every 100 milliseconds and the actuator received values every 100 milliseconds, from the compatible status transmission framework personal area network API on the main game controller device.

Figure 5 shows the delay for the SensorValue messages received on the controller game device from the position sensor, while Figure 6 shows the jitter, measured as the variation of the delay between the current message delay and the last message delay for the same scenario.

Figure 7 shows the delay for the ActuatorValue messages arriving at the actuator from the game device, while Figure 8 shows the jitter, measured as the variation of the delay between the current message delay and the last message delay for the same scenario.

Table 1 - Packages of the STF PAN API

pt.uc.dei.lcst.stf.pan
Main package of the STF PAN API. Main services like session control, late join, state receiving and transmitting, state representation, timewarp support, checkpoint and distributed checkpoint support. Location and orientation support.
pt.uc.dei.lcst.stf.pan.comms.area
Contains a pack of discovery messages, sensor and actuator classes already implemented that greatly simplify building the most common types of sensors and actuators, and classes to create more sensors and actuators, and managers for the Personal Area Network. Is effectively, the Personal Area Network API for small devices.
pt.uc.dei.lcst.stf.pan.comms.link
Implemented channels of communication and helper classes (including Bluetooth, but also ways of communicating with the servers)
pt.uc.dei.lcst.stf.pan.comms.sdp
Contains an API for SDP (Session Description Protocol [24]) representation in Java.
pt.uc.dei.lcst.stf.pan.net
Creates a general channel architecture that can be used to abstract any communication mean, be it Bluetooth or other (in the future, for example, zigbee could be supported by simply extending classes). It also contains implementations of communication means to reach the distributed servers (TCP).
pt.uc.dei.lcst.stf.pan.persist
Extends persistence to the world of J2ME CLDC with a low bandwidth alternative to the standard Java 2 SE serialization and externalization mechanisms. Package exactly equal to the SENSACT of persistence.

Table 2 - STFServer API packages

pt.uc.dei.lcst.stf
Main package of the Status Transmission Framework version

2.0 Server Side. It contains the logic that complements the functions that the Status Transmission Framework version 2.0 provides to the mobile part of applications and also the logic that STF v2.0 provides to the server side of the application. That includes state representing, latejoining, state receiving and transmitting, timewarp support, checkpoint and distributed checkpoint support. Location and orientation support.
Pt.uc.dei.lcst.stf.applications
Package that implements some applications. Namely, the central game server, the distributed game server, and some partially simulated components of 3GPP IMS for testing.
pt.uc.dei.lcst.stf.applications.messages
Messages specifically used by the applications of pt.uc.dei.lcst.stf.applications.
pt.uc.dei.lcst.stf.arms
Interface package for working with ARMSV6.
pt.uc.dei.lcst.stf.diameter
Package that implements the base diameter protocol [15] in java.
pt.uc.dei.lcst.stf.diameter.cxdx
Package that implements the 3GPP CXDX Diameter Application [16] protocol in java.
Pt.uc.dei.lcst.stf.diameter.sh
Package that implements the 3GPP SH Diameter Application [17] protocol in java.
pt.uc.dei.lcst.stf.net
Creates a general channel architecture that can be used to abstract any communication mean.
Pt.uc.dei.lcst.stf.persist
A low bandwidth alternative to the standard Java 2 SE serialization and externalization mechanisms.
pt.uc.dei.lcst.stf.sctp
Implements support for the SCTP [18][19] protocol in java in platforms that support it (java.net style). SCTP is the default protocol for communications in the DIAMETER[15] AAA protocol. Currently, Linux is supported.
Pt.uc.dei.lcst.stf.uelink
Implements specific channels of communication with the UE (mobile terminal – the personal area network central game machine).

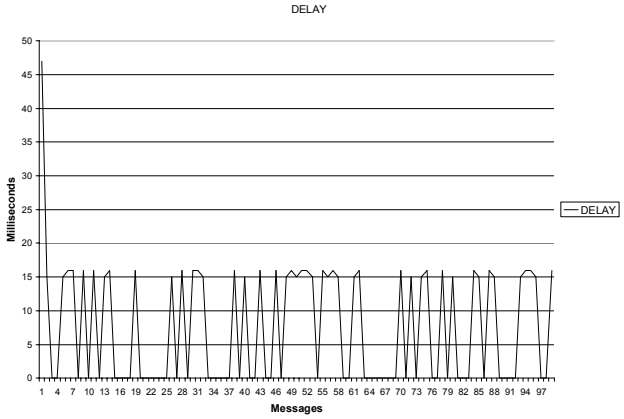

Figure 5 - Delay in milliseconds from messages received at the game controller device from a sensor

Based on the results shown in these figures we can conclude that our set of APIs, both the SENSACT API and the STF PAN API, have a consistent delay that varies from 0 to 15 milliseconds, and consequently a jitter that varies also in the order from 0 to 15 milliseconds, in the case of receiving data from a sensor in the main game controller device. In the case of receiving data in the actuator, the delay is in the area of 15 or 16 milliseconds, and the jitter varies from 0 to 1 millisecond.

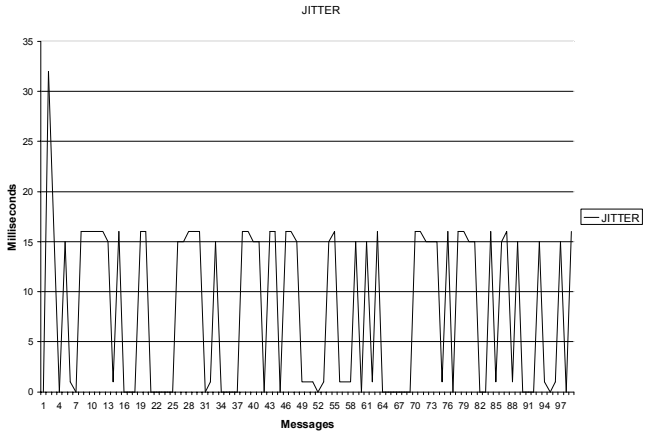

Figure 6 - Jitter in milliseconds from messages received at the game controller device from a sensor

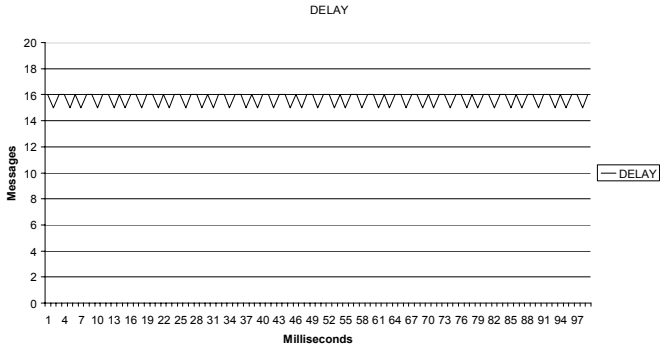

Figure 7 - Delay in milliseconds from messages received at the actuator from the main game controller device

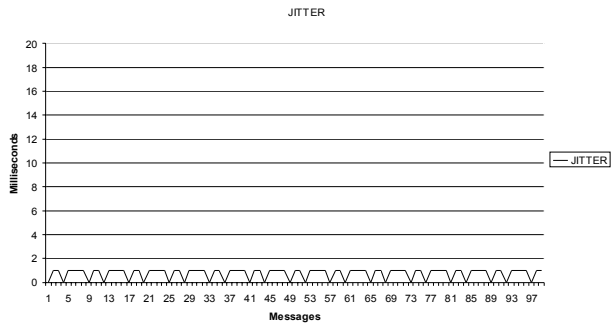

Figure 8 - Jitter in milliseconds from messages received at the actuator from the main controller game device

10.2 Common APIs

We have also tested the persistence sub-package on its own in order to estimate network bandwidth usage. For this, we designed a program to compare the savings made in relation to java Serialization and Externalization mechanisms and to allow us to calculate the size of a message for a common case. This common case was chosen to be the transmission of a position update (position sensor, for example). Then we ran a program to evaluate the size of the message that is produced applying each of the three methods, and printed the results. The obtained results were 92 bytes for Serialization, 83 bytes for externalization and only 22 bytes for SENSACT persistence.

10.3 Large Scale Network Level APIs

The Sixrm protocol API was subject to various functional and stress tests, with various profile settings. It was demonstrated that, depending on the profile settings, one can have many instances of Sixrm running on the same computer (over 10 instances, if the settings are right). It was also demonstrated that, depending on the settings, many more instances could be run on different computers. This is because they will not consume all CPU time, like we did on these stress tests, but only a fraction of CPU time, as the sources will try to adapt to the slowest computer and weakest link (because of nak throughput adaptation, and error throughput adaptation, within the limits set forth in the profile). We made tests on two computers connected by a 100 Mbps full duplex switch configured in an IPv6 network, and tests in a single computer configured to run multiple virtual nodes in an IPv6 network. The test results presented here are extracts from the tests with one computer, because they show the delay and jitter introduced solely by the Sixrm protocol machine. We present here graphics for delay and jitter for one of the nodes (received at the first node) of a Sixrm network which achieved twelve nodes in the same computer without errors. When the thirteen's node was added, there were briefly some errors that were handed to the test application by the Sixrm protocol, in a period when the computer's processor was at a peak load of 100% and availability was reaching at its physical limits, which we believed were the causes of the inability of the node to handle processing data. More details on these tests can be found on [26].

Figure 9 represents delay for the second node on the system as received on the first node. Figure 10 represents jitter for the same situation.

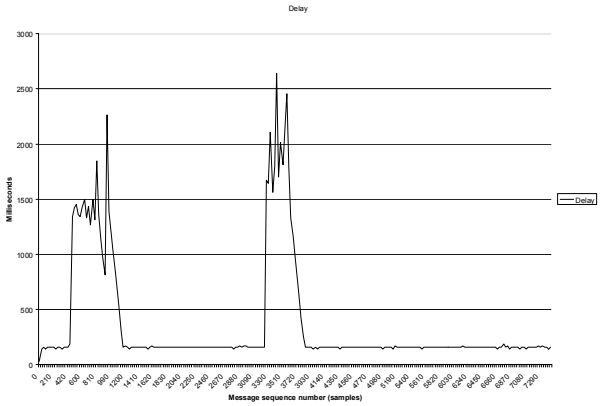

Figure 9 - Delay from second node on first node

We do not show here results for more nodes because the results are in all cases similar to these.

From these results we can conclude that the delay is normally close to 150 milliseconds and that the jitter normally varies between 0 and approximately 17 milliseconds. This happens consistently except some cases, which occur exactly on the moments where we were adding nodes to the sixrm network and the network was adjusting itself. We can see that in these moments the delay and jitter briefly increase and then stabilise in acceptable values again. In fact, they stabilize around the steady-state values, in spite of the increase in the number of nodes. The results clearly show that the steady-state values of delay and jitter are adequate for most interactive delay sensitive applications. The results also show that the complexity and overhead of the proposed Sixrm protocol are low, making it adequate for the support of the intended applications.

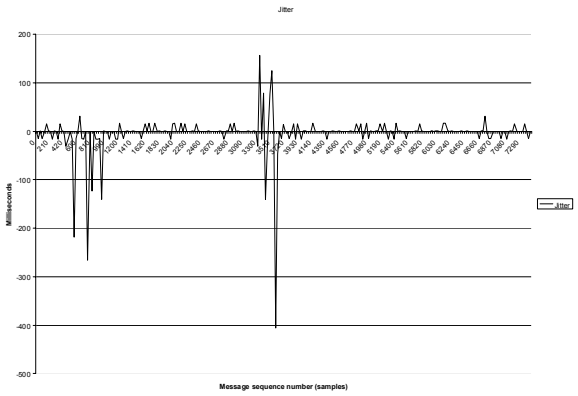

Figure 10 - Jitter from second node on first node

11. Conclusions

In this paper we have presented architectural and implementation aspects of a network middleware for the support of wireless and pervasive large-scale augmented reality systems. The middleware was implemented and extensively tested. Results obtained for the large scale part of the system (reliable multicast protocol), for the personal area network part of the system and for the common part of the APIs that handle persistence (transmission of messages on the network and/or its storage on disk – affecting bandwidth), lead us to conclude that both delay and jitter are adequate for this kind of applications. In particular, jitter has a very good behavior, which, as the main factor to consider, leads us to conclude that the solution is indeed adequate for pervasive large scale augmented reality games in a wireless 3GPP environment.

Further work will address complementary aspects such as enhanced mobility support, security and system/service manageability, as they constitute challenging research topics and are fundamental to achieve "release ready" state.

12. 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] M. Pullen, M. Myjack, C. Bouwens, "Limitations of Internet Protocol Suite for Distributed Simulation in the Large Multicast Environment", RFC 2502, IETF, February 1999

[12] Reliable Multicast Transport (IETF Working group), http://www.ietf.org/html.charters/rmt-charter.html, Acessed April 2006

[13] T. Speakman, J. Crowcroft, J. Gemmell, D. Farinacci, S. Lin, D. Leshchiner, M. Luby, T. Montgomery, L. Rizzo, A. Tweedly, N. Bhaskar, R. Edmonstone, R. Sumanasekera, L. Vicisano, "PGM Reliable Transport Protocol Specification", RFC 3208, IETF, December 2001

[14] Eila Niemelä, Teemu Vaskivuo, Agile Middleware of Pervasive Computing Environments, Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops (PERCOMW'04), 2004, IEEE

[15] P. Calhoun, J. Loughney, E. Guttman, G. Zorn, J. Arkko, "Diameter Base Protocol", RFC 3588, September 2003

[16] 3GPP TS 29.229 v7.0.0 , "3rd Generation Partnership Project; Technical Specification Group Core Networks and Terminals; Cx and Dx interfaces based on Diameter protocol; Protocol details (Release 7)", January 2006

[17] 3GPP TS 29.329 V7.0.0, "3RD generation Partnership Project; Technical Specification Group Core Network and Terminals; Sh interface based on the Diameter protocol; Protocol details (Release 7)", December 2005

[18] L.Ong., J. Yoakum, "An Introduction to the Stream Control Transmission Protocol (SCTP)", RFC 3286, May 2002

[19] Stewart, R., Xie, Q., Morneault, K. Sharp, C., Shwarzbauer, H., Taylor, T., Rytina, I., Kalla, M., Zhang, L. and V. Paxson, "Stream Control Transmission Protocol", RFC 2960, October 2008

[20] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, January 1999.

[21] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002

[22] M. Handley, V. Jacobson, "SDP: Session Description Protocol", RFC 2327, April 1998

[23] SDP API, http://jcp.org/en/jsr/detail?id=141

[24] JAIN SIP, https://jain-sip.dev.java.net/

[25] Location API for J2ME, http://jcp.org/en/jsr/detail?id=82

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

[27] CLDC – Common Limited Device Configuration 1.1, http://jcp.org/en/jsr/detail?id=139

[28] MIDP – Mobile Information Device Profile 2.0, http://jcp.org/en/jsr/detail?id=118

[29] Java APIs for Bluetooth, http://jcp.org/en/jsr/detail?id=82

[30] SIP API for J2ME, http://jcp.org/en/jsr/detail?id=180

[31] Mobile 3D Graphics API for J2ME, http://jcp.org/en/jsr/detail?id=184

[32] Mobile Media API, http://jcp.org/en/jsr/detail?id=135

[33] 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

[34] 3GPP TS 23.002, "3rd Generation Partnership Project; Technical Specification Group Services and Systems Aspects; Network architecture (Release 7)"

[35] 3GPP TS22.228, "3rd Generation Partenrship Project; Technical Specification Group Services and System Aspects; Service requirements for Internet Protocol (IP) multimedia core network subsystem; Stage 1 (Release 8)"

[36] 3GPP TS 23.228, "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; IP Multimedia Subsystem (IMS); Stage 2 (Release 7)"