# Enabling Social and Distributed Interaction in the Future 3D Internet

Stefano Chessa, Stefano Lenzi, Francesco Furfari
Institute of Information Science and Technologies (ISTI –CNR)
Via Moruzzi, 1, 56124 Pisa, Italy
and
Department of Computer Science, University of Pisa
Largo B. Pontecorvo 3, 56127 Pisa, Italy
{stefano.chessa, stefano.lenzi, francesco.furfari}@isti.cnr.it

Raffaele Bolla, Riccardo Rapuzzi, Matteo Repetto
Dept. of Communication, Computer and System Science (DIST) – University of Genoa
Via all'Opera Pia, 13
16145 Genova, Italy
{raffaele.bolla, riccardo.rapuzzi, matteo.repetto}@unige.it

**Abstract**

Current trends in 3D visualization merged with internet applications and sensor technologies will soon break the barriers to the widespread acceptance of 3D Internet, and they will enable a full user immersion in the virtual world. However this evolution will impact dramatically on the existing infrastructures since the fruition of 3D content in real time by a huge number of people poses new issues related to the system responsiveness and to the ability of managing context information. In this paper we observe that actions related to the acquisition of context from the physical and from the virtual worlds, as well as actuation in both worlds can be considered as a special case of context management and actuation. For this reason we propose a architecture for the management of such aspects within a unique framework, and an experimental testbed that validates the architecture.

1 Work funded in part by the EU FP6 project INTERMEDIA (contract N.38419)

## Introduction

After having been confined to stand alone applications for long time, the use of 3D visualization technologies in the representation of reality is now mature enough to merge with internet applications [1]. This trend can already be observed in some popular applications such as Second Life [2] or World of Warcraft [3] and it is progressively changing the way in which people will experience the future Internet (or, as often it is called, *3D Internet*). Furthermore, recent technologies such as networks of embedded sensors and actuators immersed in the physical environment appear now mature enough to overcome another limit to the widespread acceptance of 3D Internet in the everyday life, which has been the lack of suitable, non invasive interfaces letting the user to interact with the virtual world as if she was immersed in the physical environment. For these reasons, although the use of 3D is currently limited to a few applications (that however are of interest for million of people [2]), it can be expected that most future internet applications, such as e-mail, web browsing, VoIP, virtual shops, file sharing, social networking etc… will rely on this technology.

Enabling the interaction between physical and 3D virtual worlds of billions of users is an extreme challenge to the current technologies. For this reason this (r)evolution will not come for free and will impact radically on the existing infrastructures (including home networks or Personal Communication Networks), which do not appear ready to face this challenge. In fact, on the one hand the fruition of 3D content in real time by a huge number of people with an even higher number of applications will pose to the networks issues related to the available bandwidth, security, latency and real-time communication constraints; on the other hand it will be necessary to encapsulate information from a huge number of sensors and to drive actuators (in other words, to manage the user's context) to bring the user's experience beyond the possibilities offered by the nowadays virtual worlds.

All these issues naturally fit the concepts of Quality of Service (QoS) and Context Awareness (CA) that traditionally have been considered as separate problems. However in 3D Internet the model changes completely. In fact the user has an identity both in the real environment and in the virtual environment in form of avatars. From the point of view of the avatars the network is the glue of its virtual environment, and all the actions related to the network status (that normally are considered QoS-related) now become actions on the avatar's context. Under this respect aspects such as the management of QoS are just a part of the more general problem of context management of the user (intended as both physical and virtual).

In this paper we present a architecture for the management of such aspects within a unique framework, where, for example, network probes monitoring the network are a special kind of sensors, and mechanisms and policies for the QoS management or 3D graphic engines are a special kind of actuators on the virtual world. In particular we define network probes able to monitor low level flows on the network, and a virtualization layer that can filter, aggregate, and present the information coming from the network probes and from physical sensor networks in order to make this information suitable for high level decisions of the QoS manager. Differently than the conventional approach, with our framework the context management system can gather and treat uniformly all information related to the physical and virtual environment of the user.

We propose a scenario and a case study to discuss how the proposed architecture enables the integration of physical and virtual information, namely the user's position (physical information) and the network status (virtual information), and how it can result in better and more accurate decisions of the 3D applications. Finally we present an experimental testbed aimed at validating the proposed architecture.

## Related Works

Several works approach the problem of attaining the desired QoS in virtual worlds, either at the user, application, or at the network level [13]. At the user level, the QoS metrics are features experienced by the user during active participation in the virtual system; they are mostly qualitative, like immersion, presence, comfort, learning time, overall satisfaction. At the application level, the QoS takes into account the performance of the connection between users and applications (for instance interactivity), or simulation performance metrics (3D and acoustic rendering, tactile feedback etc,). Finally, at the network layer, QoS is expressed in terms of bandwidth, latency, throughput etc.

Until now, QoS in virtual worlds has mainly been approached at the user or at the application level, by building architectures where the transmission of information is adapted in order to meet predefined goals based on the user's interest for certain objects [14, 15], or based on *partitioning* schemes [16], i.e. schemes for the assignment of avatars to users, and only in a few cases the QoS at the network level is considered and mapped from the other levels [13]. However none of these approaches consider the dynamic nature of the network, whose status may change quickly during a 3D session since the network itself surrounds the user and it is part of its context.

## Background Concepts

Context-aware systems provide services which enable other applications to adapt their behaviour upon the current context. At the begging the context was mostly identified by the user location. Nowadays user localization continues to be one of most important context information, but now the word context is used to refer any environmental parameter where the meaning of the term "environmental" is as broad as possible. Environmental parameters may refer to user physiological/emotional data, user actions/movements, user identity, status of the surrounding environment, location, time, profiles, agendas and data referable to the user, and even presence and context of other users [10]. A reference architecture for context-aware system includes the following layers: sensors, raw data retrieval, pre-processing, storage/management, and application. In particular, the sensor layer includes: the hardware sensors (physical sensors), virtual sensors which offer data available from applications or services (e.g., data deducted by a specific use of a browser by the user), and logical sensors which combine information obtained from physical and virtual sensors (e.g., the location of the user associated to an action on a browser). In [10] the authors suggest a classification of context-aware systems based on the way the contextual information is collected and shared among the context-aware applications. The proposed categories, depending on the context-aware system implementation, are: direct sensor access, middleware infrastructures, or context servers.

Quality of service (QoS for short) is a generic term used to indicate methods and infrastructures for granting a given service level to selected traffic. The service level can be related to parameters such as, for example, bandwidth or latency.

Quality of Service mechanisms can be implemented locally or end-to-end. Local implementations manage the access to the physical channel and reside at the data link layer. Currently, each link-layer technology has its own mechanisms. For example, IEEE 802.11e [4] defines a limited number of access categories for queuing and scheduling of packets in WiFi networks. The presence of different mechanisms for each networking technology prevents the possibility of a uniform end-to-end service level when multiple heterogeneous networks are traversed. Thus, higher layers represent a possible interoperability solution aiming at offering a uniform vision of QoS towards users and applications; such role is held by the network layer.

Dealing with QoS at the network layer has significant advantages: traffic flows can be effectively distinguished (for both hosts and applications) and the same mechanisms can be used across heterogeneous networks. There are two different approaches from IETF, namely the Integrated Services (IntServ) [5] and the Differentiated Services (DiffServ) [6]. In Intserv the applications and the network negotiate and agree a service level for each traffic flow, while Diffserv uses only a given set of classes to differentiate the treatment of each single packet. Diffserv is simpler and more popular than Intserv, and can satisfy most QoS needs, but lacks end-to-end semantics, error signalling and admission control. Local and end-to-end QoS mechanisms can interoperate by mapping policies within the network layer into lower layer mechanisms.

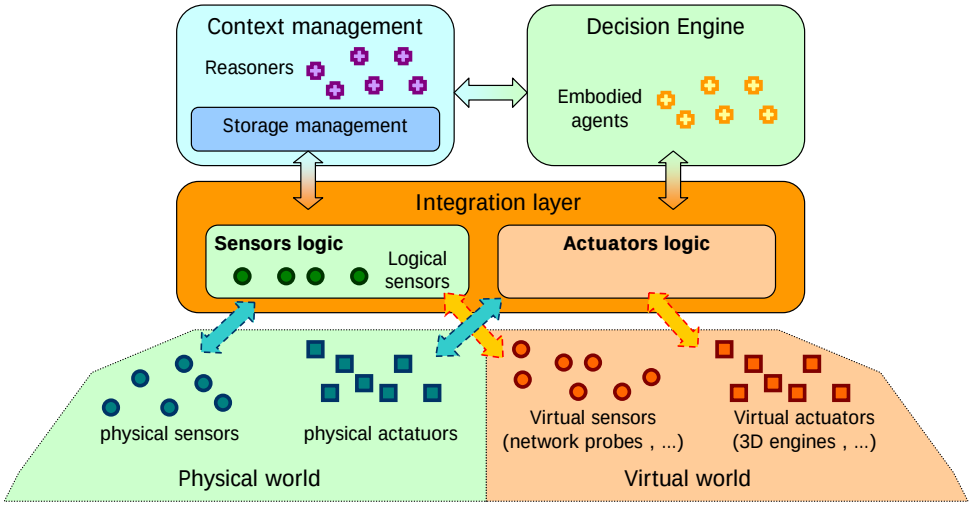


Figure 1: A conceptual architecture

## A Conceptual Architecture

The structure of a virtual world application must take into account the interaction between the real and the virtual world. Sometimes virtual worlds require some kind of sensing of the real word, especially when transmissions and mixed reality are involved. For example, user localization as well as movement and gesture recognition are useful to drive a synthetic model based on the actual user activity; on the other hand,

acquisition of environmental data can be used to bring sensory effects to the user in order to increase the overall experience of the user.

In this section we describe a conceptual architecture that can enable such interaction. The conceptual architecture (shown in Figure 1) is composed by three entities: an Integration Layer, a Context Management layer, and a Decision Engine that is a collection of distributed embodied agent able to interact with the physical and virtual worlds.

The **Integration Layer** is in charge of abstracting from the complexity of specific sensor and actuator technologies. In fact, the interaction with sensors or actuators requires interfacing with a number of heterogeneous technologies, such as ZigBee [7] for sensor networks and SNMP [8] or other specific protocols for network monitoring; each of these technologies requires specific interfaces and protocols. For this reason we propose a unifying layer called Integration Layer that enables a consistent and common way of controlling sensors and actuators. To this purpose this layer embeds two components called *Actuator* and *Sensor*. The Sensor component provides a common representation of a number of physical, logical, or virtual sensors. Example of physical sensors are accelerometers, thermometers, infra-red detectors, cameras, etc., while virtual sensors provide context information from the virtual world, as it is the case of network probes, for example. Logical sensors combine information obtained from physical and virtual sensors (e.g., the location of the user associated to an action on a browser). The Actuator component is responsible for the management of the physical and virtual actuators: we can think at physical actuators as leds, step-by-step engines or robots, while examples of virtual actuators are QoS managers or 3D engines. In the latter case, the actuator may provide functions for the user's avatar animation.

The **Context Management** layer interacts with the Integration Layer to obtain the raw data from the sensors and provides refined information to the Decision Engine. The Context Management is composed by a set of Reasoners, each of which is an intelligent entity in charge of inferring behavioural information from raw data or from the context inferred by other Reasoners. Consider for example a user that wears accelerometers on her arms. The Reasoners acquire the data produced by the accelerometers from the Integration Layer. A first Reasoner collects the data measured by the three accelerometers, and infers that the arm is moving. A second Reasoner collects the arm movement and inspecting the historical data (kept by the Storage component), it recognizes that the user is raising her arm, thus the user's context is now updated with this information.

The **Decision Engine** is a collection of distributed embodied agent that exploit the refined information from the Context Management layer to react with suitable actions by controlling the Actuators; in some sense, it completes the chain "sensing–contextualization–decision–action". The Decision Engine may be thought as the business logic that implements the real application. In the previous example, when the user raises her arm and the Context Management becomes aware of a change in her context, the Decision Engine reacts to the change in the context by requesting some actions in the physical and/or virtual world to the Actuator components.

## Case Study and Use Cases: A 3D conference

This section sketches a reference scenario for the analysis of a use case of a 3D Internet application.

**A 3D scenario**: Let us consider the case of a conference, for example a university class, that takes place in a 3D environment, where the third dimension is a requirement to overcome the limitations of the current 2D technologies in terms of lack of deep immersion in the system. The conference room is a virtual room suitable for the conference. In the 3D conference there is no 2D video, but only a synthetic model for each person, which is reproduced in the right place of the virtual room. Each user chooses the framing to view the scene, perhaps from her location inside the virtual environment. In the real world the user may enrich her experience by means of sensors that monitor her physical activity. The monitoring of the user helps her to fully plunge into the virtual conference, for example, if she turns her head the angle of vision follows her movement, or when she starts talking the system requires the network to enable a high quality audio streaming.

**Use case**: Consider in the above scenario the case where Jennifer is leading a 3D conference and she is showing her slides. In the real world she is in a room wearing her augmented reality visor, so that she can look at all the other participants as if they were in the same room. Jennifer wears also some sensors: a compass on the head and accelerometers on her arms. During the Jennifer presentation Tom raises his arm in the real world, thus his avatar in the virtual world also raises its arm. Hence Jennifer points her arm in the real world to the Tom's avatar in the virtual world to let him talk. These actions reserve network resources to begin a multicast audio streaming to enable Tom make his question, and at the same time the Tom's microphone gets enabled.

**Use case analysis**: The physical sensors on Jennifer and Tom are interfaced to the Context Management that embeds motion reasoners to infer when they are raising or pointing their hands. These movements are communicated to the Decision Engine that replicates them in the virtual world by means of the 3D actuators. Another Reasoner, for example a Real2VirtualWorld Reasoner joins the "point to direction" with the "3D world model" to infer that Jenifer is pointing to Tom and communicates this information to the Decision Engine.

At this point the Decision Engine realizes that Jennifer wants Tom to make his question, thus it acts on the QoS managers to reserve bandwidth for the audio streaming, and drives another actuator for the activation of the Tom's microphone.

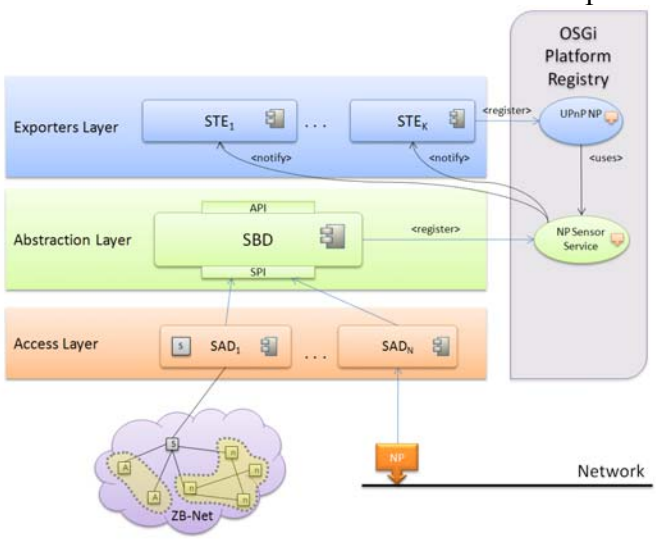


Figure 2. The SAIL layers.

## The Concrete Architecture

The role of the conceptual architecture is to illustrate the whole picture in which our work fits. In the current stage of our work, only the part of this architecture related to the acquisition of context information from sensors and control of physical actuators (called SAIL, Sensors and Actuators Integration Layer), and a simple Reasoner have been refined and implemented.

**The Sensors and Actuators Integration Layer** – SAIL enables applications to acquire context information independently of the nature of the information: physical variables (e.g., temperature, light, and humidity), network status (e.g., bandwidth, latency, quality of service), user motion (e.g., localization, gesture) and user profiles (e.g., preferences, personal data). Each source of information is modelled as a sensor and exported by an UPnP-based interface; applications become aware of available sensors (either physical, virtual or logical) through the description and publication methods of UPnP and gather information by querying the sensors. In the following we describe the SAIL architecture together with the QoS management that, we believe, will play a crucial role in the future 3D Internet.

SAIL is organized in three layers, namely the Access, Abstraction, and Exporters Layers, constructed over an OSGi framework [11] and shown in Figure 2. The SAIL Access Layer defines a minimal set of functionalities that any sensor should provide, either on its own or by means of an application adapter. This layer interacts directly with the sensors/actuators to implement the Sensor Node or the Actuator Node. To this end, it comprises a set of components called Sensor Actuator Drivers (SADs), each of which drives a sensor or an actuator. The SAD exports the sensors and actuators functionalities in terms of interfaces specified by the Abstraction Layer.

The Abstraction Layer is implemented by a single component called SAIL Base Driver (SBD). The SBD defines the interface that must be implemented by the SAD. The SBD can be thought of as a high-level driver which registers the SAD in the OSGi framework. The SBD also implements an API that is used by the Integration Layer.

The Exporters Layer exports the OSGi services registered by the Abstraction Layer to client applications. To this purpose it encapsulates different exporters, called Sensor Technology Exporters (STEs), suitable to provide access to the OSGi services using different technologies such as UPnP, SIP, web services etc...

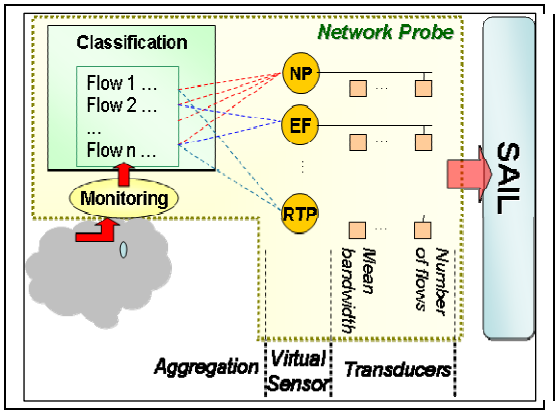


Figure 3. The Network Probes and SAIL.

**The QoS Management** – The QoS management functions are distributed into several components. In particular we identify a set of sensors aimed at network monitoring

(called Network Probes), a QoS Reasoner, and a set of QoS managers. With respect to the conceptual architecture, a Network Probe is a virtual sensor that monitors and categorizes the network traffic around the user. The Network Probe makes a flow-based classification at the application layer and keeps track of the flows above the transport layer. In particular, for each flow the probes measure all parameters defined by Cisco NetFlow, the de-facto standard in the IP traffic monitoring: total number of packets/bytes, packets' QoS class, average and current bandwidth, etc. To this purpose the probes employ a classification engine that determines the application protocol of each flow detected. Other information can be retrieved from aggregated data, such as number of active and terminated flows, statistical properties of any given class of flows (mean and instantaneous arrival rate and flow duration), bandwidth for different QoS classes. The classification engine is built upon the *Click Modular Router* [17], and specific modules have been developed in order to recognize the application protocol generating the flow. In the current testbed, the classification module is based on a patter-matching schema derived from L7-filter [9], but other filters are being considered for implementation.

The result of the activity of the Network Probe is acquired as context information handled by the QoS Reasoner, that in the current stage of development exploit simple heuristics to infer new context information. For instance the QoS Reasoner can infer the maximum available bandwidth between two peers that are located in different networks and that are monitored by two different Network Probes. The QoS managers grant access to the QoS facilities (if any) offered by the lower communication layers. The information obtained by the Network Probes fits the virtual sensor model of SAIL, where a different sensor is created for different aggregations of the traffic. In particular a Network Probe is a virtual sensor associated to the following measurements: mean and current bandwidth, number of active flows, mean and current arrival/departure rates, and number of bytes/packets arrived. Such measures are available for different levels of aggregation, namely per-L7 protocols and per-QoS classes, thus providing the Decision Engine Manager with a powerful set of information and the ability to distinguish among different classes of service, protocols and applications. Figure 3 depicts the virtual sensor abstraction in a Network Probe and shows its relationship with SAIL.

## The Experimental Testbed

In order to test and validate the proposed architecture we set up a preliminary test bed based on a set of Iris sensors [12] equipped with accelerometer, light, temperature, and audio transducers, and three PCs connected in a wireless LAN: one for the management of the sensors, one for network monitoring and the QoS managers, and one for the Context Management and the Decision Engine. A set of 5 Iris form a body network to capture the user's arms and legs movements by means of the accelerometers (as shown in Figure 4). Another set of 10 sensors are used to locate the user's position in the environment. The PC controlling the sensors is equipped with SAIL and with an exporter that provides information from the accelerometers and the location of the user by means of UPnP (since the experiments were physically limited to a room). The second PC runs the network probes and the QoS managers; it also runs SAIL to export the information obtained from the network probes. The monitoring information provided by the network probes are updated dynamically according to a configurable period. The data can be accessed according both to push or pull models. In the push

model, the Sensor Node automatically sends the data to the upper layers, while in the pull model the data has to be explicitly requested. The latter method is particularly useful when the decision engine needs more details about the network status.

The third PC runs the Reasoners and the Decision Engine. The Reasoners exploits UPnP to interconnect with the network probes, to the sensors and to the QoS managers, and it provides the information to the Decision Engine. The latter implements a few sample applications that activate different kind of streams directed to different monitors depending on the user' position and on her actions detected by the body sensors.

This testbed was also used to evaluate the overall architecture against its performance and effectiveness. In these preliminary experiments we evaluated the reactivity (in terms of latency) to the queries issued by the Decision Engine to the sensors and network probes and the overhead induced by the queries. In particular, the latency measures the time elapsed between the instant of time in which the query is issued and the instant of time in which the answer from the sensors is received. Table 1 summarizes the results obtained both for light and heavy traffic, for different levels of aggregation. "Single protocol" refers to a query inspecting the network probes sensors about all flows belonging to the same application protocols, while "All protocols" refers to a query about all active flows. With the current implementation the answer delay is below half a second on average. The network overhead for these queries was 1490 bytes on the average, with a standard deviation of 210.

*Table 1. Latency of UPnP queries to the sensors.*

| | **Light load** | | **Heavy load** | |
|---|---|---|---|---|
| | *Single protocol* | *All protocols* | *Single protocol* | *All protocol* |
| **Mean (sec)** | 0.217409 | 0.476036 | 0.4702 | 0.7391 |
| **Standard deviation** | 0.009774 | 0.052382 | 0.2423 | 0.3945 |

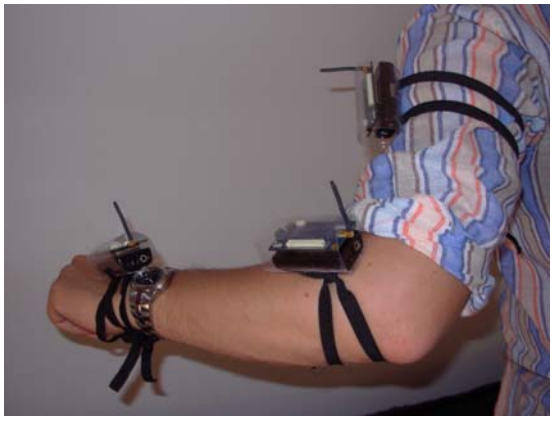

Figure 4. A detail of the body area sensor network.

## Conclusions and Future Work

We presented an architecture for enabling seamless interaction between virtual and physical worlds. The architecture aim at overcome the complexity of the interaction between these worlds by encapsulating different kind of sensors related to both world into a unified Context Management. In particular this server can manage both physical

sensors and virtual sensors. The architecture has been validated by means of a testbed that include a set of Reasoners, and set of actuators and a simple Decision Engine acting as an embodied agent. A first set of preliminary experiments prove that this architecture is suitable to support the coexistence of such sensors and that it can offer a reasonable response time.

We are currently working to encapsulate other kind of sensors and actuators, in particular to include 3D engines as actuators, and to improve the overall system performance. In the future we also plan extensive simulations and tests in order to improve the accuracy of the detection of events in both physical and virtual worlds and to evaluate the performance of the system.

## References


[1] A. Alpean, C. Bauckhage, E. Kostovinos, "Towards 3D Internet: Why, What, and How?", Inte. Conf. on Cyberworlds, Hannover Germany, 2007 pp.95-99.

[2] Second Life website: http://secondlife.com/.

[3] World of Warcraft website: http://www.worldofwarcraft.com/index.xml.

[4] IEEE 802.11 WG. Std 802.11e, IEEE Standard – Specific requirements Part 11 Amendment 8: Medium Access Control (MAC) Quality of Service Enhancements, 2005.

[5] R. Braden, D. Clark , S. Shenker, "Integrated Services in the Internet Architecture: an Overview". RFC 1633, June 1994.

[6] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang and W. Weiss, "An Architecture for Differentiated Services". RFC 2475, December 1998.

[7] ZigBee Alliance, ZigBee Specifications, version 1.0, April 2005.

[8] J. Case, M. Fedor, M. Schoffstall, J. Davin. Simple Network Management Protocol (SNMP). IETF RFC 1157, http://www.ietf.org/rfc/rfc1157.txt.

[9] Application Layer Packet Classifier for Linux. http://l7-filter.sourceforge.net.

[10] Matthias Baldauf, Schahram Dustdar, and Florian Rosenberg, "A survey on context-aware systems", Int. J. Ad Hoc and Ubiquitous Computing, 2 (4) 2007.

[11] OSGi Service Platform Release 4, Version 4.1, May 2007 (http://www.osgi.org).

[12] CrossBow Iris, http://www.xbow.com

[13] Y. Zhou and D. Gračanin. "User Level Quality of Service Mapping for Network Virtual Environments", 7th Int. Conf on Telecommunications, 2003, pp. 367-374.

[14] C. Greenhalgh, S. Benford and G. Reynard "A QoS Architecture for Collaborative Virtual Environments" ACM Int. Conf. on Multimedia, Orlando USA, 1999, pp. 121–130.

[15] S. Oh, D. Kado, K. Fujikawa, T. Matsuura, S. Shimojo and M. Arikawa, "QoS mapping for networked Virtual Reality System" SPIE Conf. on Performance and Control of Network Systems, Dallas, USA, 1997, pp. 18–26.

[16] P. Morillo, J. M. Orduña, M. Fernández and J. Duato, "A Method for Providing QoS in Distributed Virtual Environments", 13th Euromicro Conference on Parallel, Distributed and Network-Based Processing, 2005, pp. 152–159.

[17] E. Kohler, R.Morris, B.Chen, J.Jannotti, and M. Kaashoek, "The Click modular router", ACM Transactions on Computer Systems, 18(3):263-297, Aug. 2000

[18] Cisco Systems Inc., "Introduction to Cisco IOS NetFlow - A Technical Overview", 2007, http://www.cisco.com/en/US/products/ps6601/prod_white_papers_list.html