Multimed Tools Appl (2009) 45:215–245
DOI 10.1007/s11042-009-0305-9

The H-N2N framework: towards providing interperception in massive applications

Aquiles M. F. Burlamaqui · Samuel O. Azevedo · Rummenigge Rudson Dantas · Claudio A. Schneider · Josivan S. Xavier · Julio C. P. Melo · Luiz M. G. Gonçalves · Guido L. S. Filho · Jauvane C. de Oliveira

Published online: 23 May 2009

Abstract We propose a framework with a flexible architecture that have been designed and implemented for collaborative interaction of users, to be applied in massive applications through the Web. We introduce the concept of interperception and use technologies as massive virtual environments and teleoperation for the creation of environments (mixing virtual and real ones) in order to promote accessibility and transparency in the interaction between people, and between people and animate devices (such as robots) through the Web. Experiments with massive games, with interactive applications in digital television, with users and robots interacting in virtual and real versions of museums and cultural centers are presented to validate our proposal.

Keywords Massive interaction · Multi-user gaming · Accessibility · Interperception

1 Introduction

We propose the H-N2N framework including system and architecture that has been used in our Lab for the design and implementation of several massive applications through the Web. Our architecture is developed to enrich the interaction between

A. M. F. Burlamaqui (✉) · S. O. Azevedo · R. R. Dantas · C. A. Schneider · J. S. Xavier · J. C. P. Melo · L. M. G. Gonçalves
Universidade Federal do Rio Grande do Norte ECT-UFRN, Campus Universitario, Lagoa Nova, 59.072-970, Natal, Rio Grande do Norte, Brazil
e-mail: aquilesburlamaqui@gmail.com

G. L. S. Filho
Universidade Federal da Paraiba LAVID-DI-UFPB, Campus Universitario, 58059-900, Joao Pessoa, Paraíba, Brazil

J. C. de Oliveira
National Laboratory for Scientific Computing (LNCC) CCC/LNCC, Av. Getulio Vargas, 333, 25651-075, Petropolis, Rio de Janeiro, Brazil

users in a transparent way. Robust and transparent communication is provided between the users that can exchange messages as command data, text, audio and video.

The main idea consists of separating clients (individuals) in groups and of organizing these groups in a hierarchical structure. That way, the H-N2N has a Client/Server hierarchical architecture in which, for each client group, a server there exists at its disposal. The difference between the H-N2N architecture and other architectures that intend to solve the problem of wide scale environments is the possibility of leaving all clients indirectly interlinked. By applying filters in the messages going from a group to another, traffic is reduced and consequently the processing needs in the clients side.

Our system allows generic, animated devices (like robots or other animated objects) to be remotely controlled through the use of keyboard, joysticks, data glove, and voice commands. Yet, our system treats all connected entities as a common perceived user, that is, as a person, independently of its nature. So, no matter the representation inside the system, any user (including software agents or physical devices such as objects and robots, or other live beings) can be perceived as a human being in our virtual and real environments. With that, we provide flexible interfaces for people to interact to each other, also people with robots, and robots with robots themselves. Our system allows users with different resources to share the same environment in a seldom way, even if the users are connected using different graphical interfaces (1D, 2D or 3D). One or several users can communicate with one or more users (or animated devices) and also receive information from them. Also, a user can perceive other users through a virtual environment and, for example, receive video from a location where there is a robot with a camera.

We introduce a new paradigm where people can cooperate with each other and also with animate entities. In this way, the development of applications can rely on this framework in order to speed up its design and implementation. In practice, we have developed Web based applications on top of this framework at the Natalnet Laboratory, including the Hyperpresence system used for robot control through the Internet and for teaching Robotics for very young children at elementary schools [29], an Internet Cultural Space (IC-Space) [28], a system for creation and edition of virtual museums (http://www.natalnet.br/gtmv), somewhat similar as the Second Life (http://www.secondlife.com/) tool, a Table Soccer game [4], and virtual cheering application for Digital Television [30], amongst others.

So the main contribution of this work is the framework itself that allows massive applications to be implemented in a very easy fashion. The interperception paradigm provides ways of guaranteeing accessibility to users, indistinctly, dealing somewhat with fairness in games and other applications. The main characteristics of the H-N2N framework and the interperception architecture developed on top of it are discussed in the paper and experiments with theoretical and practical results using this technology are also presented, validating our proposal.

2 Avatars and multi-user environments

The need of visually representing a user in a synthetic, virtual environment has led to the design of avatars [5]. This is currently a well known term, introduced in 1965

by Sutherland, which had the idea of inserting people, in fact their representations, in synthetic environments. The implementation of this simple idea has enhanced Virtual Reality applications, making them more attractive to users. This has made possible a new kind of application technology named multi-user collaborative systems that are mostly known as Collaborative Virtual Environments (CVE). More than a synthetic world, CVE can be understood as a technology that has been widely used as a very nice tool to assist communication through the Web [29]. This technology can be used, for example, to create a three-dimensional virtual museum where people can have their visual representations inside it and can interact in a friendly way (http://www.natalnet.br/gtmv) or as a basis for a multi-user game.

However an avatar is not, merely, a visual representation. More than that, it provides a powerful tool for applications that run through the Internet. In order for an environment to make available this tool, it is necessary to provide multi-user services, that is, services that are able to control and manipulate the interaction between several users in the same application. These services are generally offered through the implementation of a distributed system that must provide functionalities such as user authentication, event synchronization amongst users in the 3D environment, user action monitoring, and the control of objects inserted in the virtual world. Thus, it is possible to guarantee the ability of user perception in the environment, given that all users have a visual representation that can be viewed by other users in the same environment. Another advantage is the involvement of the user in the world, since he or she is transported into the application. Once there, a user can move and interact with the system through a proper representation, an avatar.

Another issue is related to the environment representation in the computer itself. According to the application, the environment that we perceive can be classified in four types [17]: of reality, of virtuality, of augmented reality and of augment virtuality. Virtualization is the process by which a human observer interprets a sensory impression molded to be an extended object in an environment different of that one in which it physically does exist [2]. There are three levels of virtualization: virtual space, virtual image, and virtual environment. A virtual world can be defined as a virtual space periodically visited by people that have at least some issues in common, as activities, goals, specialized language, or some specific ethic linkage. A multi-user virtual environment refers to environments characterized by systems that support synthetic three-dimensional spaces that are shared by multiple users, remotely located.

By using Internet in these systems, we can have large-scale and heterogeneous environments, allowing people (and devices) all around the world to participate in the system in a collaborative way. So the term massive refers to very many users accessing the environments, some times millions of them. The need of a massive Internet application able to support a high number of users connected at the same time is one of the challenges here that the H-N2N architecture aims at solving.

When talking about web-based systems, another important challenge is accessibility, which deals with systems that can be used by any person, either with physical impairments or technological restrictions. The degree of accessibility of a system depends on how many people with heterogeneous capabilities and resources can use it. Thus, if we want systems that allow users through the web to have access to them in a democratic way, they must be accessible to all users.

3 Related works

Some of the first known virtual worlds are the MUDs and Habitat that is noticed as the first virtual world in which a person could have an avatar. Nowadays, there are several applications devoted to virtual world communities, which allow one to build his/her own virtual environments, e.g. those in Second Life (http://secondlife.com/). Some 3D game environments can also be mentioned as Doom (http://www.doom3.com/), Warcraft (www.warcraft.com) and Myst (http://www.mystworlds.com). The on-line Myst (http://www.mystonline.com/), Runescape (http://www.runescape.com/), and World of Warcraft (www.worldofwarcraft.com) are examples of massive versions of the games. Scalability is a great challenge in all of these applications.

One of the main concerns regarding the so called Large-Scale Collaborative Virtual Environments (LS-CVE) is the exchange of update messages, which include user id, localization in the virtual worlds and current status of the user, amongst other necessary information. Several communication architectures have been proposed considering the LS-CVE problem. One can mention NPSNET-4, DIVE, MR. TOOLKIT, BRICKNET, MASSIVE, and VELVET [7, 11, 16, 20, 25, 26] amongst others. Comparison of several of such architectures has been performed based on aspects related to visualization [6], information, process and user distribution [6], and network communication [15].

Regarding network communication, the client-server model suffices when the number of participants is small, i.e. the total number of participants is limited to the network and processing power at each station. Problems certainly arise when this model is used for large-scale CVEs [21] where the number of users is expected to be very high. The client-server model, hence, has the server as a bottleneck for the LS-CVE case, as the flow of data and processing power requirement at the server will eventually be insufficient, no matter what is the server capacity. Since LS-CVEs require a high level of scalability, also considering distributed interactive systems, solutions based on the peer-to-peer (P2P) model are often more appropriated choices for our problem as defined here. Murilo et al [19] claim that peer-to-peer LS-CVE architectures lead to a high complexity for the development and management of such systems. Other problems which also need to be addressed include data security, interoperability and network control [19].

The LS-CVE architectures mentioned herein address scalability through the deployment of an information distribution and filtering scheme which aims at allowing each user to receive only a subset of all events, hence reducing networking and processing power requirements. A common technique, employed in space-based architectures, is the geographical partitioning of the virtual world in chunks often called locales. Often, a given user only receives events which take place in its own locale as well as in those immediate neighboring locales. Other architectures have some other means of defining an Area of Interest (AoI) around a user, which defines his scope of interaction.

Such filtering schemes allow some scalability; however one can notice that each user is completely blind to events that happen outside of his/her AoI. Some events which are external to one's AoI may be relevant, however. If a given individual shots up with a firearm, only those in its surrounding areas would be aware of that event; however, in a real-life situation all participants are often aware of such an event. A situation like this screams for a solution that provides scalability, but still allows that

different groups of users (locales for instance) can exchange information about some important events, i.e. a solution that solves the *shooting in a crowd* problem.

Another important aspect treated in our work refers to the Human–Machine Interface (HMI) that means to study ways used by users to provide inputs for computer systems. Through years, HMI has been hugely improved and today it is possible to use devices such as data gloves, phantoms and head trackers to interact with machines. Kromker [13] discuss strategies to solve interaction problems created by Interactive Digital TV. But in many of these use cases, the interaction paradigm is very simple, where a human starts the interaction and the machine answers with some kind of fixed behavior as, for example, when a user turns the TV on, changes the channel or the volume. This paradigm, however, has suffered many changes and now it is accepted that it is not necessary for the user to be conscious that he is interacting with the device. The main characteristics of this new paradigm are: omnipresence, adaptation, distributed systems and new relationship between human and machines [27].

Some authors extends the HMI paradigm developing frameworks for interaction between human and domestics robots [14, 24]. Lee et al. [14] define three main modules in their architecture. One module is destined to multi-modal interaction, which provides many interaction ways between the users; the other one is a task oriented module, known as a cognitive module; and finally a module that deals with the emotions in the interaction with human. Schrempf [24] allows the planning of the robots actions even if information about the human intentions are uncertain, introducing the concept of pro-active execution of tasks. The idea of bridging the real world to the virtual one is not new. Some works have presented solutions where it is possible to connect people and animate devices through their virtual representations as well as through remote operation. Dragone et al. [9] consider a framework composed of two modules: one responsible by the treatment of social matters between the participants, that they call "Social Robot Architecture"; and the other, "Virtual Robotic Workbench", responsible for promoting the transparency on the integration between the participants in a mixed reality environment, which can be human, avatar, and robot participants.

We can notice that these works deal with similar questions, however the authors do not treat situations such as a massive environment with interaction between robots and people, both with different needs. Although the use of multi-modal interaction, cited in the systems above, they do not guarantee accessibility for their users.

So the approach introduced in the current work is also novel in these aspects. The differential of our architecture is in using interperceptional environments [1], which are environments with accessible and multi-modal interfaces, in order to provide accessibility in HMI systems. We remark that in this work we have neither approached emotion generated by the users, nor social aspects between the involved parts in these processes, as these are not, currently, related topics to our research.

4 The H-N2N framework

The H-N2N framework has been developed aiming at simplifying the construction of distributed, collaborative and massive systems, without losses in the *Quality of Service* (QoS) [22]. To better describe the framework, let us use Fig. 1, which shows

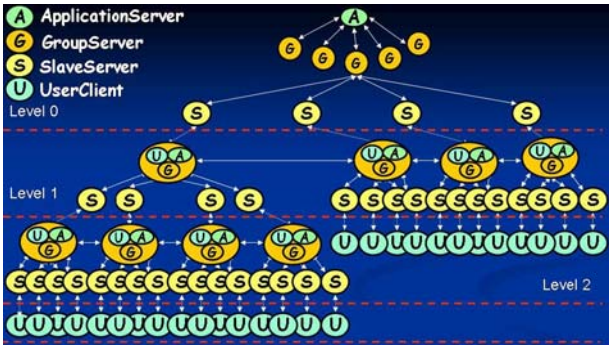


Fig. 1 Example of a possible configuration of the H-N2N architecture

an example of a possible configuration of the H-N2N architecture. In the example, we have 37 clients (UC—UserClient) connected, and 8 of those clients are also making the role of servers (GS—GroupServer).

The following architecture components are used at the server side:

- **ApplicationServer:** the main server of the application, responsible for greeting connections.
- **GroupServer:** the server of groups that contain slave servers belonging to a specified environment.
- **SlaveServer:** created to assist a client; there exists a slave for each connected client.
- **UserClient:** represents the clients connected to the *GroupServer*.

Upon the start of an H-N2N session there will be a single server, representing the first group (Group 1) of users. This server (S_0) is said to be at level 0 in the architecture hierarchy. As more users join in, the server S_0 will eventually be overloaded, what would eventually degrade the quality of the service provided to the users. Once a given server reaches its maximum capacity, a new group is created and a new server is elected. More specifically, when the server S_0 reaches its limit a new Group 2, will be created and a server S_1 (for example) will be elected. Server S_1 will be positioned in level 1, which is under level 0, and will keep a connection established to server S_0.

So often, clients may be regrouped in the servers, which allows server load-balancing. In this case, the users may be regrouped between servers S_0 and S_1, which are in levels 0 and 1 as described above. Here, there are two important aspects which must be addressed: 1) how the grouping/regrouping is to be performed and 2) how such users shall exchange messages. In the following sections we define specific functions to address these grouping/regrouping and filtering and merging of messages issues. Such functions help in controlling the information flow in the CVE.

Another important issue that needs attention is related to the way that the events propagate amongst the connected clients. In order to prevent system bottlenecks in some nodes of the hierarchy, we can apply filters that reduce the information flow. Some strategies must be drawn in order to answer some questions. The first is to what extent a given event shall be perceived in the CVE. The second is who shall perceive

a given event and the third is how the events from one group will be forwarded to other groups.

Our proposed solution is based on the location of a given user within the Virtual World, also considering its proximity to each other section of the world. In order to properly process this information, we make use of a localization matrix, as well as a graph to interconnect the various world sections. The matrix allows a given server to keep track of the locations of its users as well as the proximity of those to other users.

Figure 2 shows an example of an environment (a house), which is partitioned in 7 rooms (S0, S1, ..., S6). Rooms S1 and S4 have a single user (avatar A and D respectively). Rooms S0, S2 and S5 have no users. Room S3 has two users (avatars B and C). Finally, room S6 contains four users (avatars E, F, G and H).

Let us now suppose that the user A creates an event such as that of a bomb explosion. Such event would be sent to the server which is responsible for room S1. Such server would then decide whether or not such event should be forwarded to the immediate higher level. In our example all rooms should receive the explosion event, however, not all events need to reach all users. Some events may be unimportant enough to reach just the immediate neighborhood or none at all.

In order to decide where to send a given event to, a server needs to know about neighborhoods. For instance, if user A knocks the door that leads to room S4, an event should be generated and sent to those users within room S4.

The position of avatars can be retrieved out of a position matrix. A graph is used to store neighborhood information, with each link storing information about an immediate neighbor. The weight stored in each link of the graph provides information on how easy it is for a given information from one area to pass to the neighbor (if the rooms are linked by a glass door we should have easier transmission than if there is a bullet-proof steel door between the rooms. Figure 3 shows the graph used for the rooms shown in Fig. 2. The graph nodes represent the rooms, while the links store neighborhood information, with weight 1 for rooms with doors between them and weight 5 for those who are neighbors, but with no doors between them. The main H-N2N components are the ApplicationServer component that awaits client connections, GroupServer component that manages the message exchange amongst clients, SlaveServer component which is responsible for the communication with the user and UserClient component that represents the user.

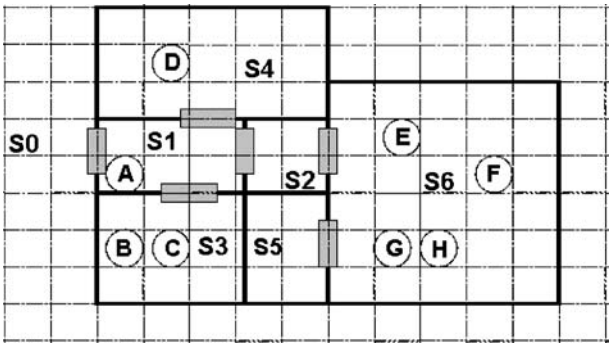


Fig. 2 Example of an environment with rooms s0 to s6 and avatars A to H

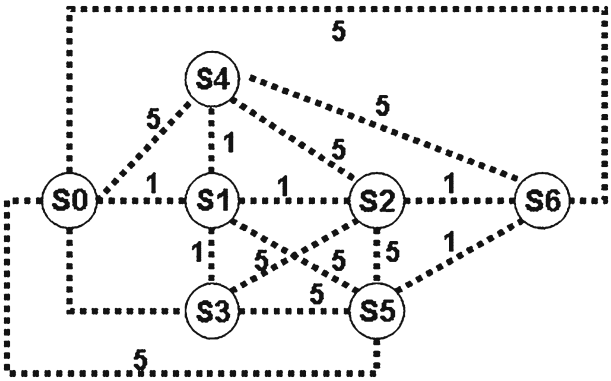


Fig. 3 Graph linking the rooms

4.1 Grouping functions

The client grouping and regrouping is performed taking into account several factors such as the processing power of the servers and clients, network characteristics (bandwidth, delay, jitter etc.), and the common objectives of the clients within the CVE. Other characteristics such as affinities and level of interest in messages are also used to define the individuals that are grouped together and, in a higher level in the hierarchy, to define groups that are grouped together. A function receives the above factors as parameters and whenever there is an event indicating that groupings or regroups are called for, the server activates this function aiming at performing any eventual group rearrangement. The function may be activated based on various types of events: temporal, new users joining in or decision based on data traffic analysis.

Let G_{n+1} be a group of clients connected to a server of level $n+1$. Let I_n be an individual from level n (that is, a user connected to a server in level $n-1$). Notice that I_n may be client, if we consider it connected to level $n-1$, or a server, if we consider that it may have children from level $n+1$. The operation C_α can thus be defined as a combination of two operations, $C_{\alpha 1}$ and $C_{\alpha 2}$, according to Eq. 1:

$$C_\alpha = C_{\alpha 1} + C_{\alpha 2} \tag{1}$$

$C_{\alpha 1} : G'_n \subset G_n \rightarrow Gn+1$ is a function that maps (transfers) a subgroup of individuals at level n a group at level $n+1$ and $C_{\alpha 2} : G_n \rightarrow I_n$ maps (collapses) a group at level n to a subgroup at level n.

Analogously, there will be a de-grouping function (C) that may be used when a client exits or disconnects from the system. It is defined as $C : G_n \rightarrow G'_{n-1} \subset G_{n-1}$ and makes a group at level n to become a subgroup at level $n-1$. This function has as objective avoiding resource misuse, which may happen when we keep a server with very few clients.

4.2 Event filtering functions

As seen earlier, several messages may be exchanged within a CVE. Considering the client grouping, each group containing its own server, we have an architecture that

makes message exchange of users of the same group quite straight forward; however, it may sound complicated to allow exchange of messages by users from different groups. In the proposed architecture clients from different groups may exchange messages that, although eventually filtered, allow the submission of at least part of the needed information.

In order to enable such inter-group communication procedure, whenever there is a group at a given level (let us assume level 0 for example), a new group is created at a lower level (level 1 in our example) to accommodate new users. In this case, the newly created group gets to be treated by the architecture as an individual in its father group. Therefore, the groups at level n are treated as individuals in the groups of level n-1.

Figure 4 illustrates the concept introduced here, where individuals of a group in fact represent groups of individuals at lower levels (and this may go on recursively). One of the children of server A, at level 0, that is an individual at this level, in fact represents a group in the lower level (1). In the specific case, the group at level 1 is then linked to a server B, which manages the group of individuals connected to it. More precisely, in Fig. 4 we have an individual at level 0, who is connected to the server A, which is a representation of the individuals that are in the group controlled by server B.

In the HN2N model, the individual at level 0 (above) representing the group at level 1 (below) will be responsible for the submission of messages between the two levels. Furthermore, in a Large Scale CVE it is not advisable to forward all messages between different levels, since that could overload the system, specially the forwarding node. One shall notice that some kind of processing is called for aiming at reducing the message passing at least towards the root of the distribution tree. To address this issue we apply filtering techniques (selection and join) in the messages which are to cross level borders. With selection we reduce the number of messages due to the fact that we select which of all messages are to be forwarded. The Join technique (mixing) allows us to merge several messages into a single one (For example one can consider mixing various audio streams into a single stream).

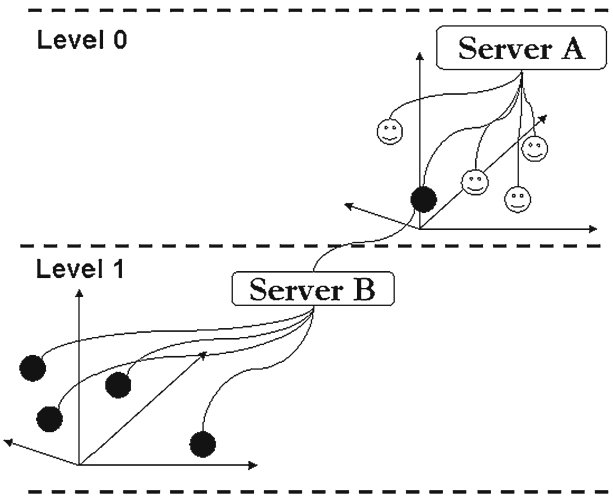


Fig. 4 Group as an individual of higher-level group

Messages exchanged between clients contain information about events in the system. The servers can apply filtering or mixing functions over these messages depending on the occurred events. The functions related to the treatment of events generated by the system may be performed between different levels as well as within a same level of the hierarchy. With this in mind, these functions may be defined as follows:

Filters between different levels:

$g_\alpha : E_n \rightarrow E_{n+1}$: event filtering function from a higher level to a lower level.

$g_\alpha : E_{n+1} \rightarrow E_n$: event filtering and mixing function from a lower level to a higher level.

Filters within the same level:

$i_\alpha : G_n \rightarrow I_n$: joining and filtering function applied in the messages generated by the group and forwarded to a single individual of the same group.

$i_\alpha : I_n \rightarrow G_n$: joining and filtering function applied in the messages generated by an individual and forwarded to the whole group.

4.3 Architecture robustness

Once defined the basic structure of the architecture, where we basically have to deal with joining of more users as well as the rules for the exchange of messages between users, we further discuss about the users (natural or forceful) exit and how the system behaves in such situations. The procedure to be performed upon a user exit depends on the way the exit occurs that can be normal or forceful.

A normal exit happens when a user wishes to leave the system through a logout procedure, the user warns all others about his/her intention to leave through an exit message. The actions to be performed in this case depend greatly of the function and location of the user within the system tree. Some problems that may occur and its solutions are described next.

Exit of a node in a leave: since the only connection in this case is that of the exiting node with its parent, all that is called for is the removal of the exiting node from the fathers list.

Exit of a parent node: when this occurs the parent will elect the best of its descendants (children, grandchildren, etc.) to replace it, considering available resources. Once out, all the orphan children will get connected to the new parent. Such procedure also holds for the root of the tree.

A forced exit happens when a node leaves the three abruptly, i.e. without sending normal exiting messages to its peers. Similarly as in the normal exit case, the actions to be performed in this case depend on the function and location of the exiting node. Possible problems that arise and their solutions follow. Failure in a leave node: normal exit, without major problems. Failure in a parent node: when a node has various children and it gets disconnected abruptly, its children, once detecting the problem, will send again a connection request to the root of the system. Failure in the main server (root node): when this happens, the children will not have parent anymore; in this case, the children will then elect the child with more resources to take the position of the parent. Once that happens, all children get connected to it. In order for allowing this, all level-0 children must know its brothers.

4.4 Communication protocol

The implementation of the H-N2N framework uses a communication protocol composed by several messages organized in categories like Control, Communication, Object/Scene Description and Real-time Data. The main messages can be viewed in Table 1. In order to better understand this protocol we describe next how the current implementation of H-N2N framework determines when a server has reached its maximum capacity.

4.4.1 Maximum capacity and load balancing

When a server reaches its maximum limit, the H-N2N has to provide load balancing in the structure of servers and clients. Let us use the scenario of Fig. 5 where we have one server, represented by circle 7 and four clients (circles 3,4,5,6) connected on it. The first message related to this scenario is the QoS (Quality of Service) message. This message can be send by a client, at any moment when it detects a QoS problem. The H-N2N server, responsible for this group, keeps a table (QoS Table) updated based on this kind of message. Additionally, this table includes information about the QoS attributes required by the running application (e.g maximum delay allowed), and about the server queue length for request packets (LRP) metric [18].

When a new client (circle 10) tries to connect by sending MSG 1 (Fig. 5B) the server checks its QoS Table. Case the acceptance of this client does not compromise

Table 1 H-N2N communication protocol

Id	Category	Type	Description
MSG 1	Control	MJoin	Client asking for connection
MSG 2	Control	MAnswer	Server accepts client connection
MSG 5	Control	MBusy	Server busy sends list of clients
MSG 6	Control	MPing	A Ping message to clients
MSG 7	Control	MPong	A Pong message response to Ping
MSG 11	Control	MExit	Client sends Exit message (leaving)
MSG 12	Control	MSon	Search clients able to be server.
MSG 13	Control	MQoS	Clients informing about QoS.
MSG 8	Communication	MText	Textual message
MSG 3	Object/Scene Description	MChanges	Coming from clients
MSG 4	Object/Scene Description	MChanges	Coming from server
MSG 9	Real time Data	MAudio	Audio messages
MSG 10	Real time Data	MVideo	Video messages

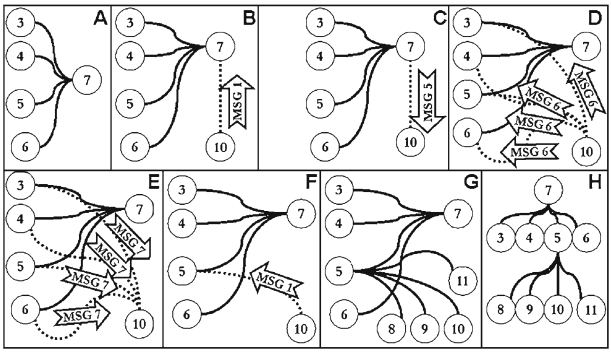


Fig. 5 New server discovery

the QoS, the connection is made naturally, that is, the server sends MSG 2 and the client joins the group. Case the server has reached a level such that a new client could compromises the QoS of the group the new client is not accepted. In this case, the server sends to the new client a message (MSG 5) with a list composed by the address of all others clients (Fig. 5C) already connected (only the clients that are possible to be new servers are added to this list). In some cases, clients could not become a server (e.g. it does not have permission to open related ports of the system) or it is already a server that has reached his maximum capacity. After the new client receives MSG 5 he sends a MPing message (MSG 6) to all other clients (Fig. 5D) that answer with a MPong message (MSG 7) (Fig. 5E). The new client receives these answers and calculates the response delay for each one. The client with lower delay becomes a server and the new client tries to connect to it (Fig. 5F).

By using the above load balancing approach, the H-N2N tree can become unbalanced. We can clear perceive this in a case where we have many users from a local network and only a few from another local network that is far away (based on the delay response between them). For example, let us suppose 100 Brazilian users (from Natalnet network) and 5 Chinese ones from another network with a lower bandwidth connected in our system. In this situation most part of the users will connect only to a subtree of our architecture, in this case the Brazilian subtree of servers. This will possibly creates a tree with two subtrees, one with 100 users and another with 5.

As a consequence of this configuration we can highlight two possibilities. If the users connected are all personal computers with normal hardware resources, the number of levels will be huge, as normal computers, depending on the application, could not provide QoS to many users forcing the creation of several new servers. If we have high performance servers on the top of the subtree, the number of levels will be reduced in our hierarchy, as a powerful server can provide good QoS to a bigger number of users such as a cluster approach. We note that other architectures also have this problem, but it is minimized here, yet allowing users to connect to the network, while keeping QoS.

5 An H-N2N based architecture for interpercetive interaction

In order to enrich user interactions, and to validate the H-N2N framework, we have devised an application architecture which has as its main characteristics

interperceptuality, accessibility, massive architecture, collaborative interaction, heterogeneous users and devices, and transparent perception. By *transparent perception* we mean that, by using the avatar, any user, including a robot, can be represented as a person. In this manner, the first barriers of communication (as resistance) that may exist in some users in order to interact with not human users are minimized.

We remark that we do not approach ways of doing IA here, in order for a robot to pretend a person. The main goal is to provide an architecture for allowing this, not the IA aspects themselves. That is matter of a parallel, ongoing work, in our lab.

The main functionalities of the proposed system (or application) architecture, which is divided in layers, can be seen in Fig. 6. Starting from the bottom, we have the H-N2N layer, responsible for the massive communication and group formation in virtual environments. Following is the VixnuServer layer that is an implementation of the H-N2N for providing services as communication, interperception, collaboration, control, and others to clients of the system. The next layer, VixnuClient, is responsible for the implementation of the clients who will make the interfaces for the users of the system that may be both people or robots, which are represented at the top of the architecture. We yet define four modules for the interaction with animate devices (like robots): Perception, Communication, Autonomy and Control. In the following Section, we better describe each layer and modules, presenting their details and showing how they provide each of the characteristics related to this main application architecture.

5.1 VixnuServer

The Vixnu 2.0 is implemented following the framework H-N2N (inherits its mains characteristics) and using the standard, Java2D e JAVA3D API (that allow the execution of the 1D, 2D e 3D interfaces in other platforms). This is currently a new version of this server constructed due to some limitations identified in the previous

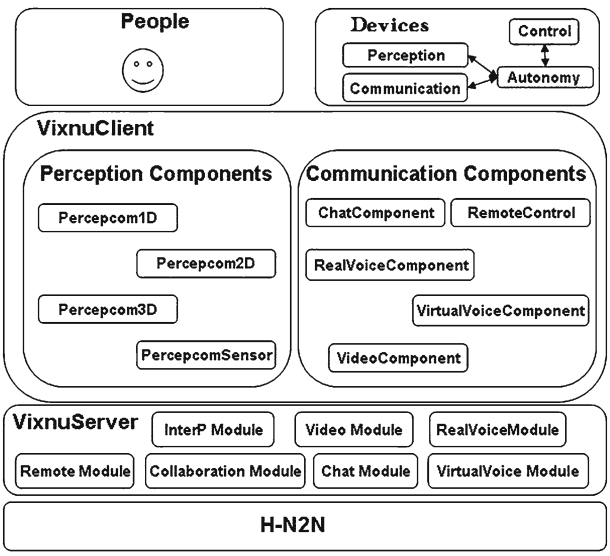


Fig. 6 InterP architecture, for interaction of people and animate devices

version[3]. A possible configuration of VixnuServer can be seen in the diagram shown in Fig. 7. We have the Server component managing the connections with the Client components. These clients use the UserClient component to set up connection with the Server. The Server creates one Slave for each connected UserClient and organizes them into an EnvironmentServer (implementation of a GroupServer) component. The Video, Voice (Real and Virtual), Chat, InterP, and Control Modules are responsible for sending and receiving video, voice, text, perception, and control messages between the clients, respectively. In the example shown in Fig. 7, there are three clients: a person interacting through a 3D representation of the environment, a robot using a 2D representation, and a blind person using a 1D representation with speech synthesis and voice recognition. The InterPModule at the server side creates the virtual environment and implements the avatar concept in a transparent way for all users. The RemoteControl component is used to send control messages to robots connected in the system.

5.2 The InterP module

The use of different interfaces can promote accessibility of our system to users who may have limitations. Such limitations may be for instance a computer that does not attend hardware or software requirements to run a more complex 3D interface; or a person whose vision is limited can access the environment by the textual interface using a speech synthesis tool. In the same way, an impaired listener can participate of audio conversations through a voice to text recognition tool. So, we can have clients sharing the same environment transparently by using one of these interfaces: 3D mix (merge video on 3D models), 3D, 2D, textual with voice recognition and speech synthesis, and purely textual. Independently of the chosen interface, all users can perceive and interact with each other (even the ones using a different interface). The creation of this kind of interperceptive system is done by following some rules, as correspondence between environments, abstraction of messages, and adequacy of interfaces. We applied these rules in the framework

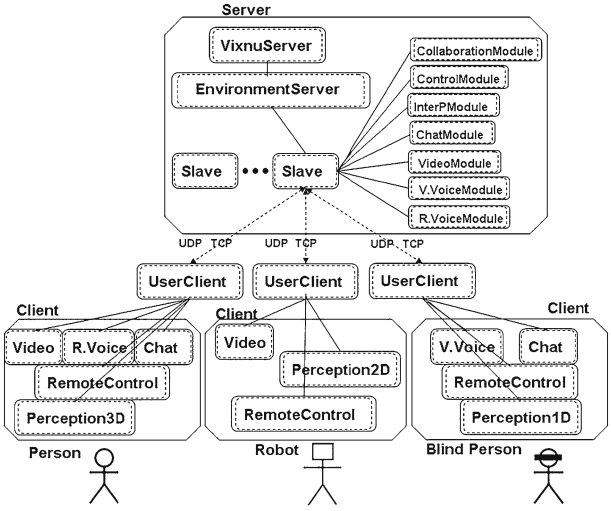


Fig. 7 Possible configuration of Vixnu 2.0 architecture

H-N2N, on the server side, including a component that translates the messages in the server to adequate messages for the corresponding client interfaces. This component, called PORTAL, transforms messages that arrive on the server from a dimension, or perception, to another.

5.3 Portal: implementing the H-N2N based interperceptive architecture

The InterP architecture creates an environment that can have support to a huge number of clients with many possible profiles. It is guaranteed access to the system for all clients, since the H-N2N allows connection to all of them. The Portal component, which is better described here, implements InterP paradigm (hence the H-N2N).

Basically, the Portal component implements the rules defined above for interperception in the framework H-N2N, on the server side. Messages that arrive on the server side are mapped (transformed) to convey with the perceptions of all client profiles, depending on the kind of connected users. After, the message returns to the client, as it can be seen in Fig. 8 that shows the interperceptional architecture, called InterP architecture. Note that the PORTAL component is the main component of the InterP Module, and is implemented on the top of H-N2N.

As an example of its use, suppose that a user is connected by the 3D interface and that he/she/it moves in the environment. This user generates a flux of messages of the MTransform type. This message type contains information about the user identification and the coordinates x, y, and z of the corresponding position. When these messages arrive at the server, the PORTAL verifies the dimension of the destiny interface. Then, the PORTAL applies the appropriated transforms and sends the final message to the clients. The PORTAL is built following the class diagram shown in Fig. 9. It is designed and constructed following the well known Design Pattern Strategy [10].

Actually, the majority of messages, like MConnect, MNickExists, MChangeRoom, and MChat, does not need to be translated. The main translation is done when the server indicates which environment should be loaded by the client interfaces. This message contains a field that indicates the URL of the corresponding model of the environment. So, a table that indicates all the equivalent models, for a same abstract

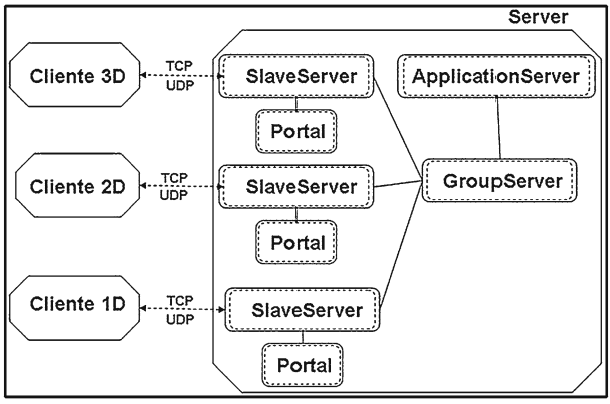


Fig. 8 InterP architecture

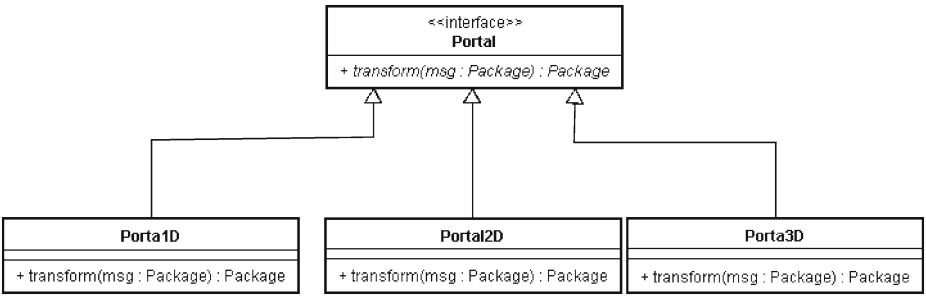


Fig. 9 PORTAL class diagram

environment to each different kind of interfaces, must be created, in order to allow the proper translation.

5.4 Allowing animate devices

People and animate devices are different, when providing components for perceiving the environment and also for communicating with other users (robots or people). People do not need any other tool, but robots need to be prepared in order to use information available in the environment. Thus, in this section, we provide details of the architecture used when a robot connects to our interperceptive architecture, using the H-N2N. Our architecture allows devices like robots to interact with people and also with other robots in a multi-modal fashion. When talking about multi-modal interaction, people refer to providing device capabilities to be programmed, to be operated or to perceive the environment (specially human actions) in two or more different ways. For example, a robot can be programmed using hand gestures or spontaneous speech [12], both input ways can be used by a person that introduces objects and places to a robot [23]. A robot can also be remotely operated using human visual, auditory and haptic senses [8]. Because devices and people can connect to the VixnuServer from any place in the world, our architecture also allows remote operation of these devices. This remote operation can be made using a keyboard, joysticks, data gloves or voice commands.

Because the supported capabilities explained above, a device connected to it has to be prepared to understand information obtained through the VixnuClient. Also, it has to be prepared to send information. Animate devices can perceive the environment in four different ways (textual, 2D video, 3D video or sensorial information). Besides, it can share information using text, voice or video, and finally robots can be controlled by a human user or else to control another robot.

An interesting characteristic of using our architecture is that any device can perceive the environment using sensors embarked or not on it. For example if a robot only has sonar sensors in front of it, it can sense the environment as if it has a complete sonar ring, all around it. It means that sonar information in the back and in the right/left sides of the robot can be simulated from a copy of the virtual environment, by the VixnuClient, and provided to the robot. Sonar information can also be computed (simulated) from other sensor data (as from stereo vision). It is important that data is fused in such a way that the robot system uses

information provided by its embarked sensors and by the VixnuClient (or other source), transparently. Note the generality of this system, which, from the robot thinking (processing), does not make distinction between simulated and real sensor information.

Animate devices need some degree of autonomy, in order to interact with users in a better fashion. Autonomy can be implemented in several ways, but the most important to think is that the robot has to change automatically between autonomous and controlled modes when a user takes/releases its control. Our architecture was designed to allow several users to control an animate device in a collaborative way, thus, the device has to have certain autonomy to decide what command it can execute. For example, this autonomy allows a device to avoid executing dangerous commands (e.g. a command that can break it, hitting a wall for instance).

Commands received through remote control component in the VixnuClient are high level commands (e.g. walk ahead), then, a device has to translate it into a group of low level commands and execute them. The last means that these devices have to have their own low level controllers (our architecture does not provide these).

Our Interperceptional architecture based on the H-N2N allows connection of any kind of person and animate device to the system. The transparency of this capability is guaranteed by implementing components in the VixnuClient or Server in a generic way. An XML file is used to describe the device characteristics. This file is specific to each device and when it connects to the VixnuClient, the file has to be sent to the client and to the server to be interpreted.

6 Experiments and results

We have made several experiments in order to validate our work that includes the H-N2N proposal itself and the interperceptive architecture. Firstly, validation of the H-N2N for further use was done using a simulation environment. Next, we implemented the interperceptive architecture on the top of it, which was used in several, massive applications that will be described here.

6.1 Validating the H-N2N framework

An implementation of H-N2N was performed in the application layer of a network simulator J-SIM [15]. A simulation of a CVE with 100 participants using our architecture was performed. In such simulation, users first establish connection with a server and then keep sending information through the established connection. This information is regarding to their positions (world coordinates X and Y) and orientations in the environment, simulating a constant motion for all of them. The users are distributed as shown in Fig. 10. One can notice that we have 6 routers: 0, 1, 2, 12, 13 and 14, the remaining nodes are standard workstations.

In a first simulation we used just 26 active nodes, with node 7 performing the task of initial H-N2N server (system root). We initialize processing in the server (7) and, after that, in the other stations. Figure 11a shows the throughput for nodes 7, 6, 10 and 25. The simulation maintains a stable throughput, as shown in Fig. 11.

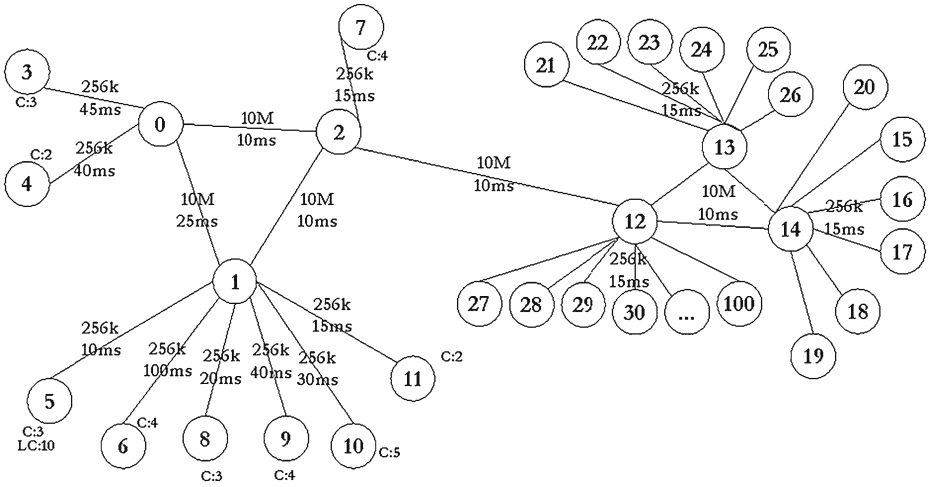


Fig. 10 Simulation topology

In a second simulation we performed the same procedure, but using an ad-hoc server instead of the H-N2N server. Such ad-hoc server has all users connected to it, which allows us to compare the throughput with H-N2Ns. At the simulation startup all 26 clients connect to the server (node 7). Figure 11b shows the throughput of the server. We can notice that the throughput is considerably higher, making the node 7 to become the bottleneck of the system.

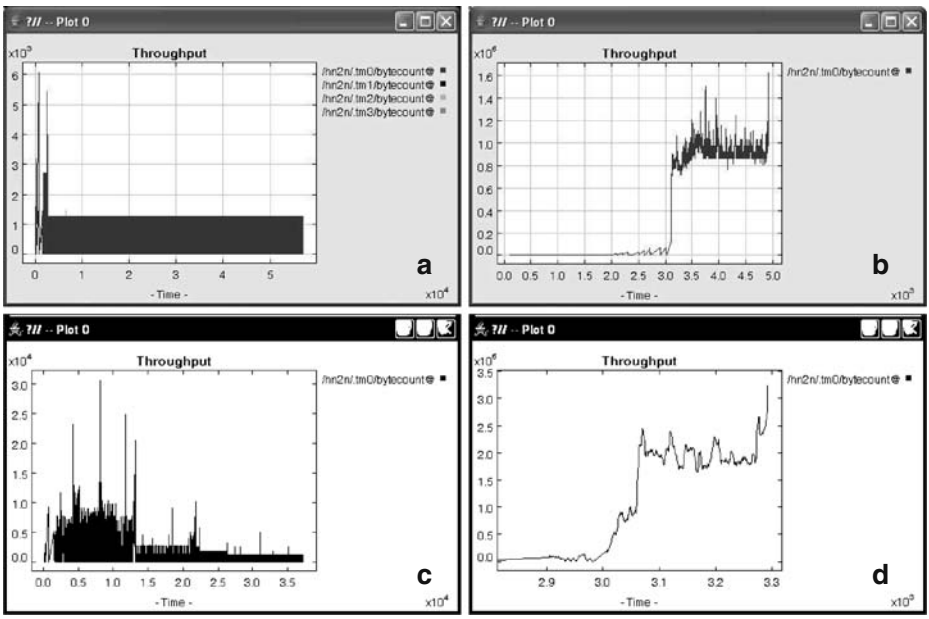


Fig. 11 **a** Throughputs for **a** H-N2N with 7, 6, 10 and 25, **b** Ad-Hoc solution, **c** node 7 (root H-N2N), and **d** an Ad-Hoc server

A third simulation was performed using all the 100 nodes of the topology adopted. An H-N2N Server (ApplicationServer) was added to node 7, whilst the other nodes were standard clients. We performed simulations with clients joining in both sequentially and simultaneously. Figure 11c shows the throughput of node 7, with a rather constant behavior, even tough several users were joining in. Such behavior indicates the efficiency and scalability of the architecture. The peak at the startup is due to the simultaneous entries performed. Finally, also using the 100 nodes available, we simulated an ad-hoc setup, which is shown in Fig. 11d, which clearly indicates the higher message flow rising as more clients join the system.

We can then observe that whilst using the proposed H-N2N architecture the throughput stabilized even tough the amount of clients connected arises. On the other hand the ad-hoc setup shows an increase in throughput as the client base arises. This simulation, therefore, indicates that as more users join the system the throughput in the H-N2N architecture remains basically constant.

Yet to test the framework, we have made an experiment with a 3 levels H-N2N configuration (two users per level) using UDP protocol. In this run, we noticed that, for each new level, the message delay increases by 5 ms. By using the TCP protocol in the same kind experiment, a 200 ms delay is added in the delivery time of messages. Based on these results, in order to setup the H-N2N servers for a better use of the framework, the application must be taken into account. If the application has hard real-time requirements between all groups of users the better approach is to use clusters as members of the top of H-N2N tree (e.g Virtual Cheering, seen below). But if the application only has real-time requirements between users in their AoI, normal computers can be used, as our architecture is designed to guarantee QoS inside the group without isolating the groups.

This shows the flexibility of the proposed architecture, as it allows developers with no resources to create a scalable and interperceptive game, or a multimedia application over the internet. At the same time it allows developers owning high performance computers to insert his/her high performances servers in the H-N2N architecture in order to improve the service quality.

Our model has also been implemented in several real case scenarios, including the virtual cheering [30], a virtual museum (http://www.natalnet.br/gtmv) and the IC-Space project [28], which will be discussed next.

6.2 Applications using the H-N2N and Interperception paradigm

Several applications were implemented in order to validate the proposed architecture and its tools (the whole implementation time and tests lasted 5 years). Here, we relate main experiments done in a cultural space, in an elementary school, in a project developed for interaction of people and robots through the web, and in a project for creation and edition of virtual museums, comparing main characteristics of each one.

6.2.1 Hyperpresence

Hyperpresence is a mixed reality environment developed by the Natalnet Lab. The idea is to have, simultaneously, two worlds representing the same situation, that is, a real space and a virtual copy of it. Figure 12 illustrates an example in which a robot is inside an environment, interacting with people. In a first roll (arrow 1), we have a

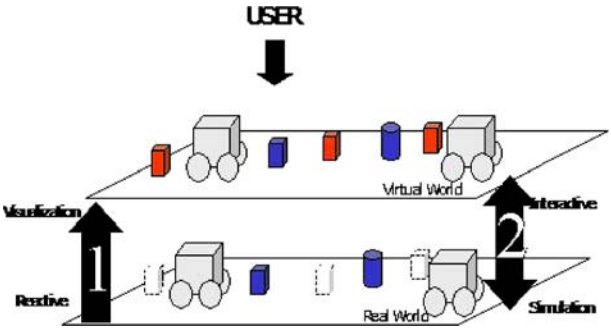


Fig. 12 Instance for the hyperpresence system

one-directional flow coming from the real plane in direction to the virtual one. The virtual plane acts as a kind of visualization filter that is responsible for exhibiting exactly what is happening in the real plane, in three-dimensional format. At a second moment, (arrow 2 in the figure), it is proposed an alternative for promoting simulation and interaction inside this platform already established in the first roll, that is, control is transferred to the virtual plane. Thus, the real plane has to simulate the events generated by the virtual plane and vice-versa. It would be possible to insert an obstacle in the virtual plane and this obstacle be perceived by the real robot that would perform an obstacle deviation motion. This even if the obstacle does not exist in the real world, being thus a ghost obstacle (or an avatar of a remotely connected user, for example).

To test the Hyperpresence, we have finalized the hardware and software setup for a Pioneer robot, kindly named Galateia. Next, we have created a new 3D model of the Conviv'art gallery, which is a small museum gallery managed by NAC (Culture and Art Center) at Federal University of Rio Grande do Norte, Brazil. As an experiment, we put a guide robot inside this gallery. 3D virtual copies of the NAC and the robot were created, coming up with a virtual environment. So, people could be visiting the gallery physically and/or remotely through our collaborative virtual environment. People remotely connected could interact with the robot through the created, mixed reality environment, while the guide robot could interact with the users really present and also with system users through the web. Figure 13 shows Galateia performing in the museum, being controlled remotely via a Remote Control module/interface.

Figure 14 shows an interface built with four available used components of this framework: perception component (top left), chat communication (bottom left), remote control buttons component (bottom right) and video I/O component (top right) and the virtual version of the environment. The perception component shows the virtual representation of the art gallery (NAC). When a user connects to the system, he/she can see the avatars of the other users and the avatar of the robot. The video transmission/reception component shows the video from a camera mounted on the robot. With the remote control, the user can move the guide robot through the art gallery. In this first experiment, only one user could control the robot: the first one asking for the robot control. With the chat module, users who are connected to the system can talk via a text communication. In this manner, users can interact via the virtual environment and via the chat module as well. In this experiment, we identified three kind of participative elements: Virtual Visitors (VV), Real Visitors

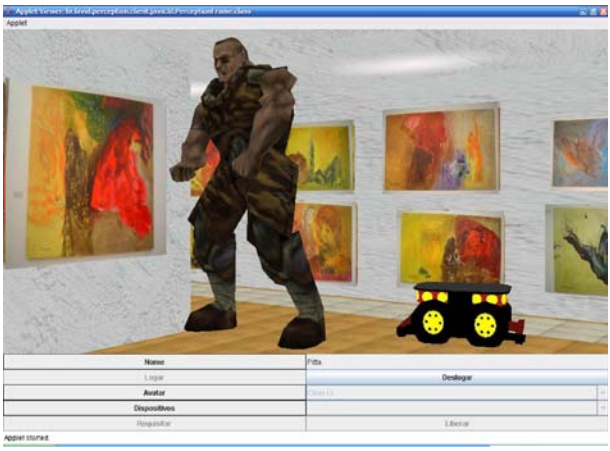


Fig. 13 Galateia robot walking around the Conviv'art. Virtual version is shown

(RV), and the robot (RB). By using the proposed H-N2N framework, the system allows interaction between all participants in several ways, as shown in Table 2.

As the last test for the Hyperpresence system, we have modeled the Station Convention Center located at Curitiba, south of Brazil, where an evaluation test for a project named GT-VR, from National Research Network, was done. This is distant some 3 thousand kilometers from Natal, RN, which is the place from which she (Galateia) and other small robots would be controlled. People in Natal could interact with Galateia and also with two other robots (LandMine and a LEGO built one), using protocols that are specific to those robots, all in a unique server (located at Rio de Janeiro). That is, the robots were made available for control at the server, and people all around would connect to this server and request a robot for operation.

Fig. 14 Vixnu interface

Table 2 Summary of the interaction media between participative elements Virtual Visitors (VV), Real Visitors (RV), and robot (RB)

Interaction	Media
VV with VV	Chat, audio and video.
VV with RV	Incarnating the Robot.
VV with RB	Voice, Data Glove, Joystick, etc.
RV with RV	Voice, gestures, etc.
RV with VV	Incarnating the Robot.
RV with RB	Voice recognition
RB with VV	Voice, text, audio, video synthesis
RB with RV	Voice synthesis
RB with RB	Web messages, voice synthesis, and recognition.

As a result, all robots could be controlled in the same environment, from different computers, and their avatars could be shown in the same virtual environment, seen at different locations, by different users.

6.2.2 The virtual cheering

This application provides tools for users (watchers) to watch transmission of sport events "together" with other watchers, for example, a soccer game. Each watcher has the possibility of sharing the acoustic space as well as the physical space of a virtual stadium, used as front-end of the application, with other virtual watchers. Figure 15 shows the schema for the acoustic sharing, which is implemented over the H-N2N with the abovementioned event filtering functions applied to mixing of the sound captured in the surround where the users are located in the field. A user's cheering during the commemoration of a score can be listened by the other users that are part of the same virtual group. A more general sound, for example including the whole stadium, can be generated through the mixing of the sound of several groups. The final mixed sound can be transmitted in real time to the place where the sport event is happening and shared with the sound of the real people present at the stadium.

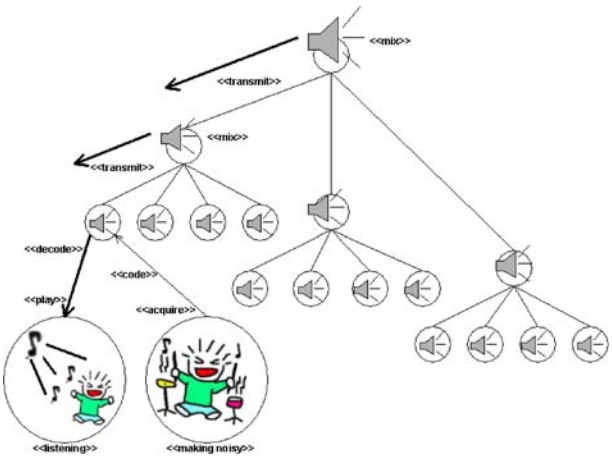


Fig. 15 Schema for mixing sound coming from several virtual users

Figure 16 shows the system running for a Digital Television version, tested during a project for the Brazilian System for Digital Television (SBTVD) model definition. This application has pointed out the appearance of a new scope of interactive TV programs that we denominate Immersive TV, once they give sensations of immersion in the real field, together with all other people. With the inclusion of 3D elements in this class of programs, the virtual version more frequently gets confused with the real one.

6.2.3 The GT-MV project

The GT-MV is a project that has being developed for collaborative creation and edition of virtual museums through the Web. The idea is to have the museum curator doing this, since he/she is the most capable person for this job (definition of the places of artworks). In certain cases, collaborative edition of virtual versions of museums is necessary, for example if two curators want to put their two museums, fair away, in the same virtual environment, linking them, one to another. So the H-N2N framework is once again used for providing this tool to all museums in Brazil (around 25 hundreds of them). Such an example of museum is again the Conviv'art, which we used as the first test-bed for our project. Figure 17 shows the museum NAC as viewed by users in the net, after its creation. There are 4 users (including the front viewer) visiting the museum.

6.2.4 Massive table soccer game for digital television

In this setup, a DTV program (see Fig. 18) has been broadcasted with videos of table soccer game matches. Games are played every day and, by the end of the month, the best players have started the finals in a dedicated channel. All final matches can be watched by supporters that are users whose only objective is to cheer, against or in favor of a team. There are two ways of implementing the client associated to this component as it is intended to be used in two different environments: on the web using Java 3D and on Digital TV following the SBTVD-T (Brazilian System of

Fig. 16 Virtual cheering interface

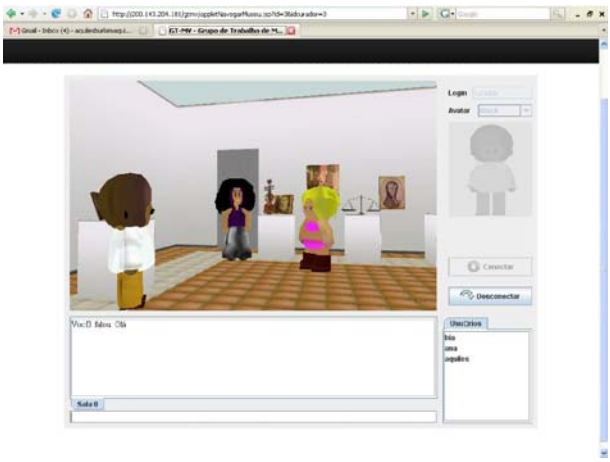


Fig. 17 People connected to the virtual version of NAC museum

Terrestrial Digital Television) standard, in this case using the API JavaTV available for this kind of environment.

Thanks to the interperception paradigm, these environments can communicate to each other in an easy and transparent way. Users can express their cheering using web cam, microphone audio from PCs, built-in cell-phones cameras, accelerometers or other devices. A strategy used to measure the degree of enthusiasm of a cheering user uses his/her body movement. The Nokia N95 has been used to capture the hand movement, using a Python script to gain access to sensor values. If the user moves the device very fast, the enthusiasm factor will be very high (varying from 0 to 10). Another way of detecting the user enthusiasm is by watching motion captured by a camera. However, this requires image processing algorithms that are known to be very expensive to run with the available hardware (cell phone). We use a simpler algorithm that uses the differences between captured images to know when the users are moving, and try to measure the intensity of the movement.

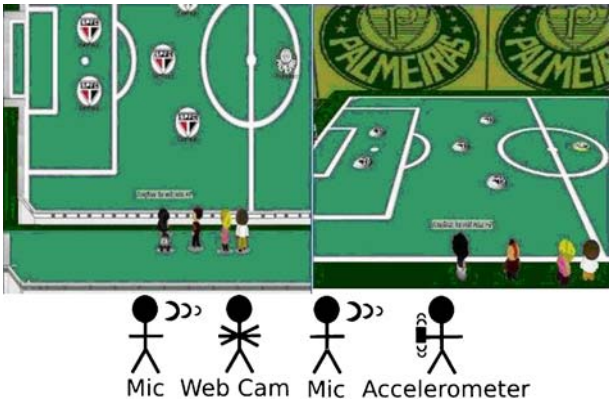


Fig. 18 Table soccer environment developed for gaming in digital television

7 Conclusion and future works

This work has proposed a framework, with which a new paradigm for interperception could be built, allowing collaborative interaction among people and animate devices on the Internet through use of massive, virtual environments. To make possible this kind of interaction, several tools are proposed and constructed, using different platforms and languages. These tools allow participants (people and animate devices) to communicate to each other, people interacting indirectly to each other through the Internet, people interacting directly to devices, and devices interacting with both.

We have shown details of the architecture, including the framework, so that users with some background in "frameworks" can easily get access to information useful for its implementation in other case studies. Thus, the framework can be used in applications that require interaction among robots and users through the Internet, including usual commands, video streaming, chat, text, sound, and voice commands.

By mixing pervasive, large scale collaborative environments, interactive digital television programs and interperception concepts, a new possibility of interaction in a game has been created. The proposed solution allows different users, using different devices, in different places to participate, in an active way, of a game, and in some cases completely change the outcome of a match.

So, as the main contribution, we have proposed an approach for users to interact through a transparent way. As an innovation, human users can either perceive animate devices, as a robot, as it is or else to understand it as a paired being, which can do some tasks, through request, sometimes forbidden tasks, would lead to responses such as "this is not allowed". Also, by using the proposed framework and related architectures, for example, several users in the same environment would control different robots or else several users would compete to control the same one (in this case, a strategy would make Galateia do what the "majority" tells her to do). The developed framework allows massive interaction.

As future work, we intend to generalize our proposal to several, different devices, as for example to have the same application being accessed by cell phones, PDA, Digital TV and computers. Also, another promising research work is to find ways to allow game fairness. This may be unpractical in some situations, but we could develop paradigms to make games more fair played. The current set of possible devices that access an application can be used as a grid, improving the processing capability in some hard consuming applications. In this case, permission would have to be allowed to the application that could run in a pervasive way.

In relation to animate devices, more sensing capabilities are to be finalized, including general features for recognition capabilities and better navigation strategies. These features will be used, in order to have a better interactivity, in a more autonomous way. We conjecture that low-level capabilities can be connected directly to IA strategies, allowing it yet to be available through the Internet, by performing reduction and abstraction of data.

Acknowledgements We acknowledge financial support from CNPq and FAPERJ, Brazilian Research Sponsoring Agencies, and from RNP, the National Research Network, sponsor of the GT-MV Project.

References

1. Azevedo S, Burlamaqui AMF, Dantas RR, Schneider CA, Gomes RB, Melo JC, Xavier J, Gonçalves LMG (2006) Interperception on shared virtual environments. In: IEEE international conference on virtual environments, human-computer interfaces, and measurement systems, VECIMS, 2006, La Coruña
2. Burdea G (1996) Force and touch feedback for virtual reality. Wiley Professional Computing, New York
3. Burlamaqui A, Tavares TA, Filho GLS, (2002) Vixnu - um servidor multi-usuário com suporte a comunicação em ambientes virtuais colaborativos. In: SBMIDIA, Fortaleza
4. Burlamaqui A, Azevedo S, Melo J, Dantas R, Schneider C, Xavier J, Gonçalves L (2008) Indirect group interaction in interpeceptive virtual environments. In: Proceedings of IEEE international conference on virtual environments, human-computer interfaces, and measurement systems (VECIMS) 2008, Istanbul
5. Capin TK, Pandzic IS, Thalmann NM, Thalmann D (1998) Realistic avatars and autonomous virtual humans. In: Earnshaw R, Vince J (eds) Virtual worlds in the internet. IEEE Computer Society, Silver Spring, pp 157–174
6. Capin T, Thalmann D, Pandzic I (1999) Taxonomy of networked virtual environments. In: Avatars in networked virtual environments. Wiley, New York, pp 15–58
7. Carlsson C, Hagsand O (1993) DIVE - a platform for multi-user virtual environments. Comput Graph 17(6):663–669
8. Dongseok R, Sungchul K, Munsang K, Song J-B (2004) Multi-modal user interface for teleoperation of robhaz-dt2 field robot system. In: Proceedings of IEEE/RSJ international conference on intelligent robots and systems (IROS) 2004, vol 1, September/October 2004, pp 168–173
9. Dragone M, Duffy BR, O'Hare GMP (2005) Social interaction between robots, avatars & humans. In: Proceedings of IEEE international workshop on robot and human interactive communication, 2005, Roman, pp 24–29
10. Gamma E, Helm R, Johnson R, Vlissides J (1994) Design patterns: elements of reusable object-oriented software. Professional computing series. Addison-Wesley, New York
11. Greenhalgh C, Benford S (1995) MASSIVE: a virtual reality system for tele-conferencing. ACM Trans Comput Hum Interfaces (TOCHI) 2(3):239261
12. Iba S, Paredis CJJ, Khosla PK (2002) Interactive multi-modal robot programming. In: Robotics and automation, 2002. Proceedings. ICRA 02. IEEE international conference, vol 1. IEEE, Piscataway, pp 161–168
13. Kunert T, Kromker H (2006) Proven interaction design solutions for accessing and viewing interactive TV content Items. In: Proceedings of 4th European conference on interactive television, 2006, Athens, pp 242–250
14. Lee KW, Kim H-R, Yoon WC, Yoon Y-S, Kwon D-S (2005) Designing a human-robot interaction framework for home service robot. In: Proceedings of IEEE international workshop on robot and human interactive communication, 2005, Roman, pp 286–293
15. Macedonia MR, Zyda MJ (1997) A taxonomy for networked virtual environments. IEEE Multimed 4(1):48–56
16. Macednia M, Zyda M, Pratt D, Barham P, Zeswitz S (1994) NPSNET: a network software architecture for large scale virtual environments. Presence Teleoperators Virtual Environ 3(4):265287
17. Milgram P et al (1994) Augmented reality: a class of displays on the reality-virtuality continuum. In Proc. of SPIE, vol 2351: telemanipulator and telepresence technologies
18. Min-Jang S et al (2008) An efficient load balancing mechanism in distributed virtual environments. ETRI J 30:618
19. Morillo P, Fernandez M, Pelechano N (2003) A grid representation for distributed virtual environments. In: Proceedings of 1st European across grids conference, vol 2970. Springer LNCS, Santiago de Compostela, pp 182–189
20. Oliveira JC, Georganas ND (2003) VELVET: an adaptive hybrid architecture for very large virtual environments. Presence Teleoperators Virtual Environ 12(6):555–580
21. Oliveira MAMS, Todesco G, Araujo RB (1999) The limitations of interactive multiuser 3D environments in the WWW, dexa. In: 10th international workshop on database & expert systems applications, p 279
22. Rfc rtp (2006) RTP: a transport protocol for real-time applications. http://www.ietf.org/rfc/rfc3550.txt. Accessed January 2006

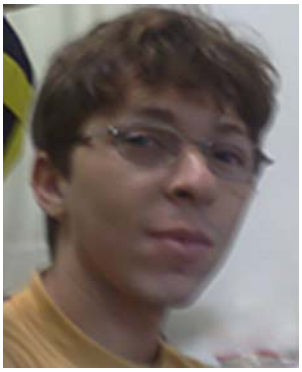
23. Saito H, Ishimura K, Hattori M, Takamori T (2002) Multi-modal human robot interaction for map generation. In: SICE 2002. Proceedings of the 41st SICE annual conference, vol 5, pp 2721–2724
24. Schrempf OC, Hanebeck UD, Schmid AJ, Worn H (2005) A novel approach to proactive human-robot cooperation. In: Robot and human interactive communication, 2005. ROMAN 2005. IEEE international workshop. IEEE, Piscataway, pp 555–560
25. Shaw C, Green M (1993) The MR toolkit peers package and experiment. In: Proceedings of IEEE virtual reality annual international symposium, pp 463–469
26. Singh G, Serra L, Png W, Wong A, Ng H (1995) BrickNet: Sharing object behaviors on the net. In: Proceedings of IEEE virtual reality annual international sumposium, 19–25 March 1995
27. Souza F (2004) Frameworks and middlewares for ubiquitous computing (in Portuguese). Campus, So Paulo
28. Tavares TA, Araujo AS, Souza Filho, GL (2001) ICSpace the artists place on the net. In: AMT2001, 2001, Hong Kong. Springer-Verlag lectures notes in computer science, vol 2252. Springer, Hong Kong
29. Tavares T, Tavares D, Gonçalves L, Burlamaqui A, Lemos G (2003) Hyperpresence an application environment for control of multi-user agents in mixed reality spaces. In: 36th annual simulation symposium, 2003, Orlando. Proceedings 36th annual simulation symposium (ANSS-36 2003), pp 351–358
30. Tavares T et al (2004) Sharing virtual acoustic spaces over interactive TV programs—presenting "virtual cheering" application. In: 2004 IEEE international conference on multimedia and expo-ICME

Aquiles M. F. Burlamaqui holds a PhD in Electrical and Computing Engineering from Federal University of Rio Grande do Norte, Brazil. He is currently an Associate Professor at Science and Technology School of Federal University of Rio Grande do Norte, Brazil. He has done researches and published many papers in the fields of Multimedia, Virtual Reality, Electronic Games and Interactive Digital Television and related topics.

Samuel O. Azevedo holds a MSc. degree in Systems and Computing from Federal University of Rio Grande do Norte, Brazil. He is currently a PhD student at Electrical and Computing Engineering Program at UFRN. His main experiences are in the fields of Artificial Intelligence and Software Engineering, including research topics as Cryptology, Collaborative Virtual Environments, Digital Image Processing, Multiagents Systems, Educational Robotics and Computing Games.

Rummenigge Rudson Dantas holds an MSc degree in Electrical and Computing Engineering from Federal University of Rio Grande do Norte, Brazil, where he is currently a PhD student. He has done research in the fields of Software Engineering, Collaborative Virtual Environments, Digital Image Processing, and Games Computing.

Claudio A. Schneider holds a B.Sc. in Computer Science from Federal University of Rio Grande do Norte, where he is currently a graduate (MSc) student. His main research interests are in the fields of Shared Virtual Environments as Virtual Museums, Massive Multiplayer Games, and Digital Television.



Josivan S. Xavier holds a B.Sc. in Computer Science from Federal University of Rio Grande do Norte, where he is currently a M. Sc. student. His main interests are in the fields of Shared Virtual Environments, Massive Multiplayer Games, and Digital Television.

Julio C. P. Melo holds a B.Sc. degree in Computer Engineering from Federal University of Rio Grande do Norte where he is currently a M. Sc. Student. His main interests are in the fields of Digital Television, Shared Virtual Environments, Massive and Multiplayer Games.

Luiz M. G. Gonçalves holds a PhD in Systems and Computing Engineering from Federal University of Rio de Janeiro, Brazil. He is an associate professor at Universidade Federal do Rio Grande do Norte, Brazil. His main interests are in the field of Graphics Processing, including topics as Computer Vision, Robotics, and Multimedia Applications. He is a Member of IEEE and Brazilian Computing Society.

Guido L. S. Filho holds a Ph.D. in Computer Science from Pontifícial Catholic University of Rio de Janeiro. His main interests are in the fields of Multimedia, including topics as Virtual reality, Distributed Multimedia Applications, and Digital Television.

Jauvane C. Oliveira holds a Ph.D. in Electrical and Computer Engineering from University of Ottawa. He is currently a researcher at the National Laboratory for Scientific Computation, Brazil. His research interests include Collaborative Virtual Environments, Multimedia, Virtual Reality, and Computer Networking. He is a Senior Member of the ACM, as well as a member of the Brazilian Computing Society (SBC).