Computing and Informatics, Vol. 24, 2005, 31–51

A SCALABLE NETWORK ARCHITECTURE FOR CLOSELY COUPLED COLLABORATION

Christoph Anthes

GUP Linz
Johannes Kepler University Linz
Altenbergerstraße 69
A-4040 Linz, Austria/Europe
e-mail: canthes@gup.jku.at

Adrian Haffegee

Centre for Advanced Computing and Emerging Technologies
The University of Reading
Reading, RG6 6AY
United Kingdom
e-mail: sir04amh@reading.ac.uk

Paul Heinzlreiter, Jens Volkert

GUP Linz
Johannes Kepler University Linz
Altenbergerstraße 69
A-4040 Linz, Austria/Europe

Manuscript received 16 December 2004

Abstract. This article describes the architecture and the network communication of a large-scale, networked virtual environment, which is designed to specifically support closely-coupled collaboration in highly interactive scenarios.

Its main goals are the maintenance of low latency during user interaction and fast multicasting of messages in order to fulfill consistency requirements. This is achieved by sophisticated message distribution techniques, peer-to-peer connections between

interacting clients and a global hierarchical communication topology. Scalability is realised through partitioning the virtual world.

Keywords: Distributed virtual environments, closely-coupled collaboration, scalability

1 INTRODUCTION

In recent years a growing interest can be observed in the area of large scale virtual environments (VEs). They have become typical Virtual Reality (VR) applications. These VEs can be identified by their huge size and complexity, as well as their large population numbers. In many cases the characters populating these environments represent human users which are connected to the environment over a network. An important example application domain of these VEs are Massively Multiplayer Online Roleplaying Games such as Everquest [12]. Other typical applications span the areas of training scenarios, military simulations [15], or virtual meetings [9].

For supporting such large environments, while also maintaining a tolerable latency during interaction within the VE such environments are commonly distributed over several computers. They are therefore identified as distributed virtual environments (DVEs) or networked virtual environments (NVEs). The clients use replicated databases and run their own simulations of the virtual world while being synchronised over the network. The biggest challenge in these environments is allowing many users to experience the world concurrently and to interact with it in real time. Geographically dislocated users should be able to be immersed to the degree that they really feel to be colocated within the environment.

Considering interaction its highest level is defined by [21] as truly concurrent object manipulation. This is what we understand as closely-coupled collaboration; the concurrent manipulation of the same attribute or attributes of an object by two or more users. In a cooperative scenario which is different from a collaborative scenario as defined by [5] users can share the same environment but they are only able to manipulate the world sequentially, as implemented in the most computer supported cooperative work (CSCW) applications. Closely-coupled collaboration needs very low latencies since more than one user are directly integrated in the simulation loop. The world is manipulated based on the perception of one user and the reaction of another collaborating participant. Enabling this type of interaction poses high requirements onto the systems network layer by requiring a maximum lag of 200 milliseconds during interaction [23].

The networking architecture described in this article addresses the issues of closely-coupled collaboration using peer-to-peer (p2p) connections between clients, while the issue of scalability is addressed by distributing messages of global concern hierarchically. This networking architecture is part of a framework which will allow the intuitive design of highly interactive NVE applications.

The article is structured as follows: After a discussion of related work in the area of NVEs Section 3 gives an overview over the networking topology of our system. Subsequently message types as well as their distribution mechanisms are discussed in Section 4. Section 5 addresses the issues of fault tolerance of the environment before Section 6 describes the testing environment with some test results being presented in Section 7. Finally a discussion of future work concludes the paper.

2 RELATED WORK

In the last few years research has been undertaken in the area of NVEs. Many different approaches have been suggested to address various issues. Good overviews of the field are given in [16, 19, 22, 27].

A very common approach is to partition the whole environment into subsections of different shapes and sizes. NPSNET [20] and SIMNET [8], which are focused on military applications distribute state updates using broadcast messages and dead reckoning to hide latencies. In NPSNET the world is partitioned into regions of hexagonal shape. Entities in this environment send their messages to the multicast group of the region they belong to as well as to the surrounding regions via their multicast groups. This approach scales well if the entities are distributed evenly over the groups. Participating entities in an NPSNET system communicate over the DIS [11] protocol, which enables integration of independently developed components.

Spline [4] splits the world into regions called 'Locales' which can be of arbitrary size and shape. The world can be designed to avoid the initial occurrence of too many clients in one region. The disadvantage in this approach is due to movement of the entities in the environment; the size and shape of the regions would have to change dynamically, which is computationally intense.

The RING system [14] takes the line of sight between two participants into account to judge the update rate between two clients. If two entities cannot see each other no real-time updates are needed, the latency requirements for updates are reduced significantly. This is achieved by a client-server system, where the server routes the messages according to the visibility relation between two clients. The visibility approach is also applied for spatial partitioning by identifying non-visible portions of the scene.

All of the above address the issue of scalability by partitioning NVEs. Other NVEs work with completely replicated databases or by using a p2p architecture. Another notable NVE system, MASSIVE, is described in [3]. It uses p2p connections for managing interaction between the VE participants and implements a spatial model of interaction by taking into account the aura of two avatars for judging the necessity of a p2p connection. MASSIVE guarantees a high responsiveness and low latencies using p2p connections between several clients once the connections are established. Another interesting aspect of MASSIVE is the use of different connections for different media types. Some approaches try to work only on p2p architectures, like the distributed VR chat application described in [17].

The DIVE system [13] can be used for building distributed VR applications across local and wide-area networks. It supports different types of devices like the CAVE [10] and desktop workstations and uses a replicated database which is updated concurrently using multicast network communication.

Virtual Life Network (VLNET) [24] is a collaborative, client-server based DVE, which offers a modular, extendible architecture as well as support for gestural and facial communication by a high-level virtual human representation.

Another huge field in the area of NVEs is the research on protocols which allow efficient communication between dislocated entities. The most important of these protocols are VRTP [7], DWTP [6] and the DIS protocol. An interesting approach in this field is Bamboo [28] which uses dynamical download and installation of networking protocols, to guarantee an optimal runtime performance in the communication between objects.

Considering the collaboration aspect of DVEs, not many approaches supporting closely-coupled collaboration have been undertaken so far. A scenario where two users are carrying a piano through a virtual scene is described in [26]. To minimize the effects of network latency, this system operates over a LAN, which results in smooth avatar movement. Within the scenario described in [25], the users collaboratively build a virtual gazebo. This scenario is built on top of DIVE.

In [18] an approach for collaborative virtual sculpting is elaborated, which is based on a hybrid client-server and a peer-to-peer architecture to allow real-time deformation and rendering of objects.

Considering the work discussed above, none of the network architectures is specifically addressing the aspects of closely-coupled collaboration in combination with large scalability. Our approach is to combine the concepts of hybrid client-server and p2p architectures with a replicated world database in order to guarantee latencies below 200 milliseconds and still provide high scalability.

3 NETWORK TOPOLOGY

3.1 Terminology

For supporting the understanding of some important terms within the network topology a short description of their meaning within the context of this paper is given here:

Client: A client is a computer which has joined the VE and is represented therein as an avatar.

Portal server: The portal server acts as a single entry point into the NVE for new clients.

Domain: A domain is a VE part of varying size which is managed by a single domain server.

Domain server: A domain server is a client, which has taken over the additional responsibility of managing the domain where it is located.

Segment: A segment is a part of a domain which has a fixed size and serves as an atomic element for domain splitting, merging, or resizing operations.

Cluster representative: A cluster representative routes global state update messages between the domain servers.

Area of interest: An area of interest (AoI) is a sphere around a client defining the part of the VE which can be directly influenced by it.

3.2 Structure

Several approaches of different NVEs are combined in our network architecture to enable closely-coupled collaboration in a scalable environment. The distributed system is composed of the following entities: clients, domain servers, cluster representatives, and the portal server.

The interrelation of these entities is illustrated in Figure 1. The cluster representatives are shown on top in dark grey, the domain servers are on the intermediate layer coloured in light grey, whereas the white clients are located at the bottom. The figure also illustrates the communication topology.

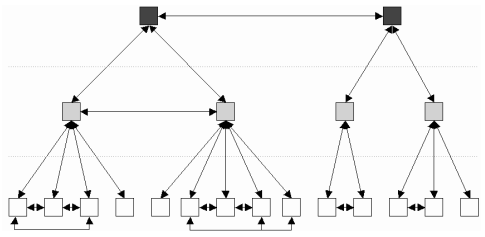


Fig. 1. Entity hierarchy

The main role of the portal server is that of a single entry point for new clients joining the system. It accepts client connections, does the initial synchronisation with the world database by providing the client with the scene information required and integrates it into the communication topology.

The clients are represented within the VE by their avatars. They have abilities to interact with each other and the environment and hold a replicated VE database.

A domain server is a client, which additionally acts as administrator of its domain and as a communication interface towards the other domain servers and the responsible cluster representative. This interface function is used for the distribution of messages of global concern, such as state updates. Thus all clients are directly connected to the domain server responsible for the domain in which they are currently located.

Domains can be split, which subsequently generates a new domain, by turning a client system into a new domain server. This takes place if the number of clients becomes too large for a single domain server. New clients joining the domain are then

connected to the new domain server. If the population count of a domain becomes too low to justify its existence or if the domain server loses its network connection the reverse operation takes place and the domain is merged with a neighbouring one.

A cluster representative embodies the next level up the hierarchy. It routes messages between the clients located within its cluster of domain servers and the outer world.

For starting up a new NVE only a portal server is needed. The first client connecting to the portal server becomes domain server and also a cluster representative. As a domain server it will arrange the connections between its future clients in order to provide maximum communication performance. A subsequently joining client connects to the portal server and receives the object states to initialize its replicated object database. After that it is passed on to the domain server, which is responsible for the domain where the client is currently located within the VE.

We identify inter-domain and intra-domain communication as different types of communication using their own types of logical network topologies. Position and orientation changes of objects and avatars are handled by intra-domain communication, while inter-domain communication deals with state updates and avatar migration. Domain splitting and merging are two operations which involve both types of communication. Although domain splitting is initiated within the intra-domain topology, it mainly affects the inter-domain topology. A domain merge is an operation initiated by inter-domain communication, while ending up in intra-domain communication when finished.

3.3 Inter-Domain Topology

The main task of the inter-domain network topology is the distribution of state updates between different domains. Each domain server is directly connected to its cluster representative and also to the directly adjacent domain servers, which are identified by their domains sharing a common border or corner.

The inter-domain topology is shown in Figure 2, where the domain servers are shown as light grey blocks, while the cluster representatives are represented as black boxes. The connections to the neighbouring domain servers are shown as thin arrows, while the connections to the cluster representatives are displayed by the dashed strong ones. The cluster representatives are interconnected with large grey arrows within the picture.

3.4 Intra-domain topology

The communication topology within a domain consists of the connections of the domain server towards all the clients and transient p2p connections between clients currently performing a closely-coupled collaboration. The establishment of such a p2p connection is initiated by the domain server, since it knows the location of all clients within its domain. If two avatars are approaching each other, their areas

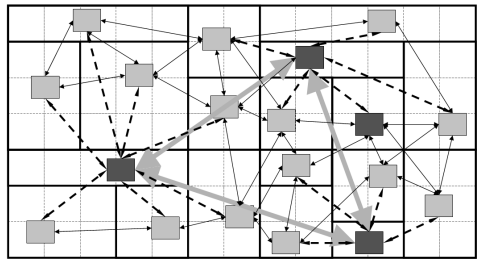


Fig. 2. Inter-domain topology

of interest (AoIs) are used by the domain server to decide the necessity for a p2p connection. In contrast to the approach used within the MASSIVE system [3] where the AoIs are referred to as auras, we define an AoI as a sphere around an object defining its area of possible influence and interaction. Figure 3 shows a domain server and several clients in a typical intra-domain topology. It shows different levels for the AoI as rings around each of the entities. These represent ranges that are used for influencing p2p connections, and the messages sent over them. The solid lines represent permanent connections between the DS and every client, whereas the dotted lines indicate the transient connections which are established based on inter-client distances.

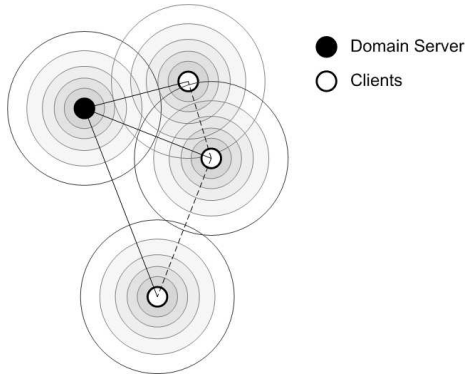


Fig. 3. Intra-domain topology

3.4.1 Segments

Each domain is split up into several segments. The segments have a fixed size and are used for geometrical structuring of the scene. For splitting and merging operations segments are treated as atomic. Another important use of segments is the addressing of positions in the VE. Each segment has its own local coordinate

system which is required to deliver the necessary precision while not limiting the size of the whole VE.

4 MESSAGE DISTRIBUTION

Actions within the VE initiated by clients such as movement and interaction result in messages which have to be distributed to maintain the consistency of the environment. One can identify tracking data, state updates, and synchronisation data as different categories of messages.

4.1 Tracking Data

Tracking data is automatically generated when using VR systems like Head Mounted Displays (HMDs) or CAVEs [10]. It represents the position and orientation of different sensors, whose number can differ according to the hardware which is available within the VR system. Typically two sensors are used which are tracking the head of the user as well as the input device located in the users hand. Each sensor is delivering position and orientation information, each represented as a vector of three floating point values.

The head trackers position and orientation as well as the controller are evaluated for positioning the avatar representing that user within the VE. The position values of the head are also used to measure the distance between two avatars within the VE. This distance information steers the level of detail which is used to represent the other avatar within the users field of view.

If two users are close to each other communication using postures is possible since the avatars head and hand within the VE are positioned according to the tracked position and orientation values. To represent the users motion in a correct way the avatar is positioned using inverse kinematics. However, to perceive postures correctly the users have to stay comparably close together. A minimum distance between two avatars must be maintained to avoid unrealistic visual representations. On the other hand, if the distance exceeds a certain limit awareness of the other avatars presence is in most cases sufficient, since the possibilities of interaction are limited.

Since the head's position is more important than orientation for representing the whole avatar within the VE, the position information is distributed at a higher rate than the orientation. For supporting this differentiation the AoI of the clients has been separated into five different levels which are distinguished based on the distance between two clients. Based on the AoI level between two clients (far, long, mid, near, and collaborating) the amount of data to be transmitted is selected as shown in Table 1. The number of asterisks indicates the relative frequency of updates (four being the most frequent).

The message format used for the transmission of tracking data is specified as shown in Figure 4. At first the user id identifies a specific client as the source of

Range	Visibility	Head Position	Head Orientation	Wand Position	Wand Orientation
Far	Out of sight	*	n/a	n/a	n/a
Long	Nearly visible	**	n/a	n/a	n/a
Mid	Visible	***	*	*	n/a
Near	Visible	****	**	**	**
Collaborating	Visible	****	****	****	****

Table 1. Update rates depending on client distance

the message before the position and orientation of the users head sensor are sent, followed by the same type of values for the wand. The position is represented by a (x, y, z) coordinate triple, while the orientation is encoded by heading, pitch, and roll which represent the rotation around the z, x, and y-axis respectively. The six degrees of freedom for each tracking value are sent at a 32 bit floating point resolution.

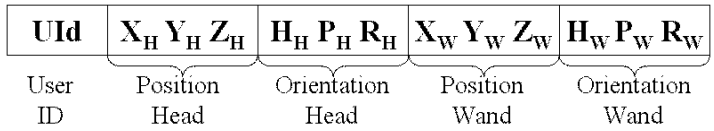


Fig. 4. Tracking message format

The avatar's position and orientation are also transferred as two floating point triplets. The position of the avatar within the world coordinate system of the VE is calculated by adding the tracking coordinates to the navigated coordinates of the VR device being used. The user's movement within the VE can be interpreted as moving the VR hardware being used through the VE. Within this model the navigated coordinates represent the position of the device within the VE.

Navigation can be performed in many different ways, which is not subject of the paper. For more details refer to [2].

4.2 State Updates

State updates are used for distributing changes to the world database and can be initiated by several entities within the VE but in most cases an avatar is the source of such an event. In addition to avatars, other objects and entities can also act as event sources. In general we can identify avatars, actors and interactive objects as different types of event sources.

Avatars represent users within the VE and act on their behalf while actors are active objects which are steered by background scripts as part of the simulation. Interactive objects are defined by their ability to interact with an avatar or an actor, but the interaction is not initiated by them.

Events generated by an avatar or an actor in combination with an interactive object can be classified into five different types:

- Touch
- Untouch
- Grab
- Release
- Use.

The *touch* and *untouch* events refer to the avatar touching the interactive object without changing its position, while *grab* and *release* delimit a time period of the avatar holding the object in its hand. The *use* event refers to the avatar holding an object and using it as a tool for manipulating other interactive objects.

The message format for state update messages is specified as shown in Figure 5, where MT defines the message type, UI is the user id, OI holds the object identifier, OT identifies the object type, SG contains the segment, while PS indicates the position and TI is the timestamp of the message.

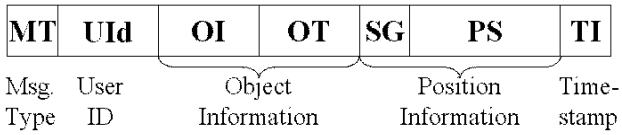


Fig. 5. State Update Message Format

4.2.1 State Update Distribution

The distribution of state update information aims at distributing the state update as fast as possible to all interested clients, but first providing the information to the nearby clients. Therefore a hierarchical message distribution scheme is applied, which involves different visibility levels. Messages are only distributed if the clients involved are visible and therefore aware of each other.

First the state update is done locally on the client. For the next step the update is distributed within the segment where the client is located, with first the initiating client sending the message to its peers. Then it passes the message upward to its domain server, which distributes the message to the clients in the same segment which have not been informed by the initiator. The required receivers of the update are found by the domain server based on the connectivity information which the domain server holds about its domain. Not only do the clients within the segment in question receive the message, but also the ones within the neighbouring segments.

If the message to be distributed is of importance for all the clients within the VE, global message distribution as the third visibility level is done by the domain server,

first by starting with updating the remaining clients within its domain before informing its peers to which it is connected, and which are assigned to the same cluster representative. After the cluster representative has received the message it subsequently multicasts the information to the other domain servers which have not been informed, before then distributing the message to its peers which proceed accordingly.

In the case where the source of the update message is a client currently migrating from one domain to another, and is therefore connected to two domain servers, both of them distribute the update within their domains.

4.2.2 Event Cascades

During state update distribution event cascades can occur. This is the case if an event occurs as a consequence of another. The initial event results from an avatar action or is triggered by system time. Within the architecture described the simulation runs independently on each client using a fully replicated world database and scene graph. This means that each state update must be broadcast to the other participants. In case of an event cascade where one event is causally determined by its predecessor only the initiating event needs to be distributed since the occurrence of the subsequent events is determined by the first event received as trigger and the contents of the world database.

If the exact time of occurrence of an event is needed, a message can also be distributed even if it encodes a subsequent event. Such an explicit distribution is only allowed if the messages are distributed by a unique source. Such a unique source is guaranteed by letting the initiating avatar of an event cascade act as the source of all messages within the cascade. In case of a collaborative object manipulation which involves more than one avatar, the avatar which first connects to the object acts as the initiator of the object manipulation and thus of all messages generated during the collaborative action. The direct synchronization between the involved avatars besides the state update messages which are of global interest is done over a p2p connection between the avatars involved.

4.2.3 State Update Distribution Example

A good example is given by the following scenario: A user picks up a key to unlock a door and afterwards grabs the door handle to open it. The messages in this scenario to be sent would be a pickup message for the key, since the key is an interactive object. The position updates of the key do not have to be distributed since all clients get a notification that the avatar carries a key, therefore they can calculate the key position according to tracking data which has been distributed for correct representation of that avatar. Once the key collides with the door and a use event is generated by the user the next state update is distributed. Through this use event every client is notified that the door is in a state in which it can be opened. The avatar drops the key and generates a drop event which is distributed.

The new position of the key is transmitted as well. To open the door the avatar grabs the door handle and gets attached to it leading to the generation of the next message. Now the user is able to open the door by changing his hand position. The axis of the door has to be connected to an orientation interpolator which will calculate the position of the door according to the tracking data. When the user finally releases the door handle a release message is generated, which will leave the door open at a position which is distributed via the release message. During this whole process five messages are generated which have to be distributed. All clients which are connected via p2p connect will be able to see a smooth interaction due to the transmission of the tracking data which is distributed to provide a smooth animation of the avatar.

4.3 Synchronization Data

Based on the hierarchical structure of the network topology changes to it have to be done in a synchronized manner. Messages carrying information about such changes are called synchronization messages and occur when a client migrates from one domain to another, during the merging and splitting of domains as well as due to the leaving of a domain server or cluster representative.

4.3.1 Moving Through Segments

For exploring the VE or completing certain tasks the user moves his avatar through the virtual world. To guarantee scalability only the actual area in which the avatar moves is loaded into memory at the client machine. The area consists of the segment where the avatar is located and the adjacent eight segments for supporting faster data fetching.

If it becomes likely that the client will change its segment, the adjacent segments of the segment to which the client might want to move are loaded. The probability of a segment change is measured by comparing the actual distance between the avatar and the segment border to a threshold distance.

In a scene graph representation using quadratic segment dimension a maximum of 14 segments have to be loaded at once, as shown in Figure 6. The central segment in which the user currently moves, and the eight surrounding ones, are loaded any time during client movement. In the worst case, if the client leaves a segment through a corner point five new segments have to be loaded. Using this method the client always has the data needed, even if the avatar is moving quickly through the NVE.

Splitting up the whole NVE as described above supports scalability but leads to a complex structure in world design. The coordinates of an object have to be enhanced with index values for the segment. Five coordinates are needed to define the exact position of an object in the data structure. Two coordinates are needed to identify the current segment of the object. The other three coordinates determine the

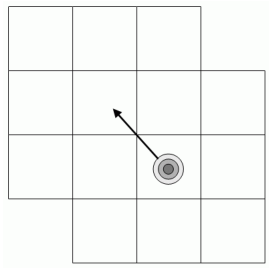


Fig. 6. Prefetched segments during client movement

position in space inside of the segment. As a consequence of domain segmentation a much higher coordinate resolution is available.

4.3.2 Client Join

Another important issue is providing a consistent and up to date view of the whole VE for clients joining the VE. Therefore the world data is provided by the domain server to which the client is assigned by the portal server.

If a new client joins the VE it initially connects to the portal server where a unique user id is assigned to it. The client is handed over to the domain server of the starting segment, which has been chosen by the user. The domain server holds the database of the segment being entered as well as the databases of the adjacent segments. If the databases are not available, they are downloaded automatically from the portal server. If the client connects to a domain server it is registered and can receive all changes within the domain compared to the database received, although its representation is not visible for the other clients. It updates its local view of the VE applying the data, which has been received in between onto the received database. After the client has current information about its current segment the same process is applied for the adjacent segments. The avatar of the client becomes visible within the VE after it holds the required data and is fully integrated into the communication topology.

4.3.3 Domain Splitting

If the split of a domain is considered reasonable all adjacent domain servers of the domain which is supposed to be split are notified. During the splitting operation the adjacent servers are not allowed to split their own domains. The splitting is done along borders of segments which serve as atomic units for a split operation. The best suited of the clients within the splitting domain is promoted to a domain server. The selection is done based on system performance, load and available network connectivity. The new domain server which results from the split operation establishes connections to the adjacent servers, which are identified by their connections to the old domain server and their position relative to the domain being splitted. The

clients inside the splitting domain remain connected to the old domain server, while some of them also establish connections to the new one. The selection of clients to be connected to the new domain server is done based on the location of the clients within the domain, with those nearest to the new server being the ones selected. To complete the splitting operation client connections to the old domain server are removed if they are not needed. Finally the adjacent domain servers and the cluster representative are notified that the split operation has completed.

4.3.4 Domain Merging and Resizing

The reverse operation for domain splitting is given by domain merging. We identify two reasons which make a domain merge necessary. The first reason is a significant amount of message traffic generated due to synchronization if clients frequently change between two adjacent domains. A reason for this could be the result of an inefficient split operation. Another reason would be a certain event which has taken place in the virtual world leading to an accumulation of avatars at a domain border.

The domain server sends a message to all adjacent domain servers that it is willing to merge or resize. The recipients check their load with special attention to the traffic at the domain borders. Even in case the load of two domain servers is high a merge operation could be beneficial, if the reason for their high load is border traffic generated by clients which frequently change between the two domains.

Resizing for the reduction of frequent domain change traffic can only be taken into account if the sidelength of the border, which is going to be moved, is equal to the length of the two involved domains on both sides. That can be judged locally because each domain server holds the size of the adjacent domain servers.

If a resize or a merge operation is decided between two adjacent domains, the adjacent domains of the resizing or merging ones are notified. The cluster representatives are informed about this domain change, but are not directly involved in it.

5 FAULT TOLERANCE

The issue of fault tolerance is a very important one within this system, where clients can be promoted to domain servers or even cluster representatives. The system has to cope with a node going off the network, independent of its current role.

If a pure client loses the connection to the VE no recovery mechanism is needed. In contrast, domain servers as well as cluster representatives need to be backed up by their peers. Since a domain server is used for connection establishment between clients, the data of which clients are in its domain has to be backed up by the adjacent domain servers.

In case a domain server leaves the environment unexpectedly this is noticed by the adjacent domain servers and the clients in the domain. The adjacent domain servers delegate a client in the domain to replace the domain server. Messages sent by the clients during the loss time have to be resent to the new domain server.

Cluster representatives are used mainly for the distribution of state updates within the VE. If a cluster representative leaves the VE unexpectedly, the domain servers belonging to the cluster as well as the peers of the cluster representative notice this. A new cluster representative is selected from the group of domain servers which were connected to the former cluster representative. Connections are re-established to the domain servers and the cluster representatives. In case of global message distribution during the time where no cluster representative was available, the messages have to be resent.

6 TESTING ENVIRONMENT

Testing and evaluating the network layer of a NVE system is a complicated but crucial task for judging the performance of the overall system. To prove the functionality of the described architecture, an application has been written that can be used for the evaluation of the architecture and the simulation of a large number of clients.

The implementation follows a systematic approach. First the low level functionality for establishing connections as well as sending and receiving messages was developed. This was then extended to create a process for handling connections, and for managing buffers containing transmit and receive data. From these it was possible to create the topology of networking entities that would support the proposed architecture. This last stage was developed in tandem with various test and simulation tools to validate the implementation.

6.1 Testclient

The testclient utility was created to allow the simulation of a single user instance. It is a wrapper around the client object implementation, and controls it imitating a human user.

Having successfully connected to a domain server through the portal server, the application generates a random initial position. It also establishes a desired destination, and a temporary mode of travel (walking, running or stationary).

A periodic timer, currently with a resolution of 50 ms, is started for effecting changes in the client's behaviour. For each iteration the application calls a heartbeat function in the client, allowing it to service any necessary networking tasks required by the topology. Next the testclient updates its position based on its travel mode and informs the client. If this new position is its destination it randomly selects a new one. Additionally it checks if it is time to change its mode of movement.

Basic wand (hand) usage support for the client is available, although currently just consisting of the client moving its wand up and down in a regular motion. This motion is reported to the client object for dissemination over the network as required.

The utility also gathers information on other clients, some of which is written to shared memory where it can be used for further debugging, simulation analysis

and visualization. In the future it could also be used as an input to the testclient's behaviour, for example reacting to the presence of other clients by moving towards other avatars and offering a handshake.

Configuration support has been provided for tuning the testclient. Some of it serves informational purposes such as the portal server network adress, but it can also be used for controlling the behaviour of the client.

6.2 Simtest

Simtest has been developed to test and stress the full network environment. It is configured to optionally create the portal server object, and any number of testclient client simulations each one using its own configuration.

Simulated entities can be mixed with real ones as desired. A real client connecting to the Simtest portal server would see multiple clients moving around within the environment, and acting as dictated by their individual configuration.

6.3 Visual representation

Full network simulation generates large quantities of data due to the constant sending of tracking updates. Interpretation of textual log files is trivial to interpret for the human user. Graphical representation can help, therefore a basic visualization tool, showsim, has been developed. It uses client information, stored in shared memory by the various test and simulation tools, to generate a birds-eye graphical representation of the VE. One of the main purposes of this tool is the detailed evaluation of client behaviour and the resulting network communication.

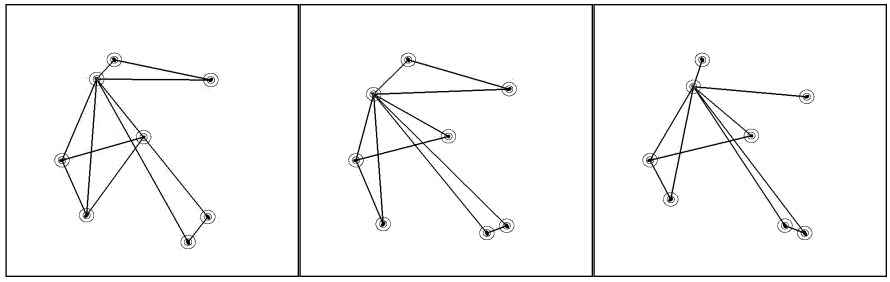


Fig. 7. Screenshots of showsim simulation application

Figure 7 shows a sample snapshot of the showsim output. It gives a bird's eye view of a single domain containing various clients and the domain server (just to the left of centre). Connections between clients are indicated by lines. It can be seen that all clients are connected to the domain server, but only those within their respective AoI ranges have connections to each other. The image is updated as the clients move, and one can dynamically see as connections are made and broken as the clients first move close, and then away.

Additionally showsim supports basic console input and output, allowing polling of state information of the environment, or any of the clients.

6.4 Monitoring Client Status

The topology objects have been designed to generate debug and status information on request. Such information is useful in determining the load on an entity, and in tuning parameters for obtaining optimal network performance.

A debug and monitoring utility, debmon, has been created that provides a method for remotely connecting to any client and then interacting with it through console input resulting in various actions. The types of interaction currently supported are as follows:

- Text messages can be sent to the remote client. These can then be displayed on the remote users console terminal, providing a basic textual communication medium.
- `ping` functionality has been implemented, which determines the round-trip delay for a simple message.
- Requesting networking statistics such as the number of bytes sent and received on a per connection basis.
- Requesting basic remote client status (e.g. name, topology role, etc.).
- Requesting networking process status, including information on network IPC buffer usage, which can be indicative of potential delays due to slow networks.

7 RESULTS

Using the test and simulation tools various results have been gathered. Currently these have just been acquired for the intra-domain case, however it is expected that the performance will also scale well when used in the inter-domain case.

The basic infrastructure ping was used to prove the functionality of the topology, and to get a baseline of network response time. On a moderately loaded simulation system the ping was observed not to take much more time than a regular ICMP ping. It was in the order of tens of milliseconds for international connections (averaging 38 ms for connections between the UK and Austria), and milliseconds between local sites. Obviously this could increase for heavily loaded sites, or for connections spanning multiple entities in the hierarchy, however the latency certainly won't be in the order of seconds as sometimes experienced using other architectures.

The quantity of intra-domain network traffic was also analysed while running the simulated environment. Many factors can impact these results, obviously if there are numerous users in a small domain their traffic will be much more due to their intersecting AoIs, than for a similar number of users in a larger domain. Through running simulations in which domain size, AoI ranges, and frequency of updates can

all be changed it is possible to start building up an overview of optimal networking parameters.

Table 2 demonstrates the effect that the number of clients in an intra domain topology can have on network traffic. It shows the number of bytes transmitted and received, by both domain server and client entities. The results are the averaged readings over multiple runs for each of the different user counts, and show the general traffic trends. To stimulate excessive message interaction, and to stress the system, these results were taken from a relatively small domain (units of $4000 \times 4000 \times 1000$) using clients with sizeable AoIs (awareness at 1000 units, disconnection at 1500, and with 800, 700, 600, 400 and 200 being far, long, mid, near and close ranges, respectively). The clients sent updates to the domain server every 300 ms, and to each other between 100 ms and 500 ms when at ranges between close and far.

No Users	DS Tx	DS Rx	Client Tx	Client Rx
5	1.6	250	230	180
15	12	700	250	200
25	26	1200	570	530

Table 2. Network traffic (bytes/second) for intra domain topology entities for differing numbers of users

While the pre-averaged figures for the domain server traffic over the multiple runs was fairly consistent, those for the clients' showed a higher variance. This was due to the clients' quasi-random motion having variable effects on the AoI intersections, and therefore the message quantity. As expected, the table shows that average client traffic increases exponentially with an increased number of domain users. This reflects the higher density of users creating a greater quantity of p2p messaging. From these figures it can be seen that acceptable throughput can be achieved at a user count up to 25 clients.

Without further study it is difficult to exactly determine the inter-domain scalability. However, since such messaging will not involve tracking information, but just the lightweight event traffic, it can be assumed that the scalability will surpass the 25 entity intra-domain level simulated here. Based on this assumption as a lower boundary full system scalability would exceed 15 000 users (25^3).

8 CONCLUSIONS

This article introduced an adaptive, scalable, highly responsive network architecture, which has been designed to support closely-coupled collaboration. The high responsiveness is realised by the use of p2p connections between interacting clients, whereas scalability is supported by a three level hierarchy. The message passing mechanisms as well as the different types of messages have been defined. A prototype application has been used for evaluation of the proposed architecture.

Future work will include further development of the network architecture, as well as a framework supporting highly interactive applications.

REFERENCES

[1] Anthes, C.—Heinzlreiter, P.—Haffegee, A.—Volkert, J.: Message Traffic in a Distributed Virtual Environment for Close-Coupled Collaboration. In: Proceedings of the 17th International Conference on Parallel and Distributed Computing Systems, PDCS '04, San Francisco, CA, USA, September 2004, pp. 484–490.

[2] Anthes, C.—Heinzlreiter, P.—Kurka, G.—Volkert, J.: Navigation Models for a Flexible, Multi-Mode VR Navigation Framework. In: Proceedings of the ACM SIGGRAPH International Conference on Virtual-Reality Continuum and its Applications in Industry, VRCAI '04, Singapore, June 2004, pp. 476–479.

[3] Greenhalgh, C.—Benford, S.: MASSIVE: A Distributed Virtual Reality System Incorporating Spatial Trading. In: Proceedings of IEEE 15th International Conference on Distributed Computing Systems, Vancouver, Canada, May 1995, pp. 27–35.

[4] Barrus, J.— Waters, R.C.— Anderson D.: Locales and Beacons: Efficient and Precise Support for Large Multi-User Virtual Environments. In: Proceedings of the IEEE Virtual Reality Annual International Symposium, Santa Clara, USA, 1996, pp. 204–213.

[5] Broll, W.: Interacting in Distributed Collaborative Virtual Environments. In: PIn: Proceedings of the IEEE Virtual Reality Annual International Symposium, Los Alamitos, March 1995, pp. 148–155.

[6] Broll, W.: DWTP – An Internet Protocol For Shared Virtual Environments. In: Proceedings of the Third Symposium on the Virtual Reality Modeling Language, Monterey 1998.

[7] Brutzman, D.—Zyda, M.—Watsen, K.—Macedonia, M.: Virtual Reality Transfer Protocol (VRTP) Design Rationale. In: Procedings of Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET-ICE): Sharing A Distributed Virtual Reality, June 1997, pp. 179–186.

[8] Calvin, J.—Dickens, A.—Gaines, B.—Metzger, P.—Miller, D.—Owen, D.: The SIMNET Virtual World Architecture. In: Proceedings of the IEEE Virtual Reality Annual International Symposium, September 1993, pp. 450–455.

[9] Childers, L.—Disz, T.—Olson, R.—Papka, M. E.—Stevens, R.—Udeshi, T.: Access Grid: Immersive Group-to-Group Collaborative Visualization. In: Proceedings of the 4th International Immersive Projection Technology Workshop, 2000.

[10] Cruz-Neira, C.—Sandin, D. J.—DeFanti, T. A.—Kenyon, R. V.—Hart, J. C.: The CAVE: Audio Visual Experience Automatic Virtual Environment. Communications of the ACM, Vol. 35, 1992, No. 6, pp. 64–72.

[11] Protocols for Distributed Interactive Simulation. ANSI/IEEE Standard 1278-1993, March 1993.

[12] Everquest, Sony Computer Entertainment America Inc., `http://www.everquest.com`, 1999.

[13] Carlsson, C.–Hagsand, O.: DIVE – A Platform for Multi-User Virtual Environments. Computers & Graphics, Vol. 17, Issue 6, 1993, pp. 663–669.

[14] Funkhouser, T. A.: RING: A Client-Server System for Multi-User Virtual Environments. In: Proceedings of the Symposium on Interactive 3D Graphics, 1995, pp. 85–92.

[15] High Level Architecture, Defense Modeling and Simulation Office. US Department of Defence, `https://www.dmso.mil/public/transition/hla`.

[16] Joslin, C.—Pandzic, I. S.—Magnenat-Thalmann, N: Trends in Networked Collaborative Virtual Environments. Computer Communications, Vol. 26, 2003, No. 5, pp. 430–437.

[17] Kawahara, Y.—Morikawa, H.—Aoyama,T.: A Peer-to-Peer Message Exchange Scheme for Large Scale Networked Virtual Environments. In: Proceedings of the 8th IEEE International Conference on Communications Systems, 2002.

[18] Li, F. W. B.—Lau, R. W. H. —Ng, F. F. C.: Collaborative Distributed Virtual Sculpting. In: Proceedings of the IEEE Virtual Reality 2001 Conference, 2001, pp. 217–224.

[19] Macedonia, M. R.—Zyda, M. J.: A Taxonomy for Networked Virtual Enviroments. IEEE Multimedia, Vol. 4, 1997, No. 1, pp. 48–56.

[20] Macedonia, M. R.—Zyda, M. J.—Pratt, D. R.—Barham, P. T.—Zestwitz, P. T.: NPSNET: A Network Software Architecture for Large-Scale Virtual Environments. Presence: Teleoperators and Virtual Environments, Vol. 3, 1994, No. 4.

[21] Margery, D. M.—Arnaldi, B.—Plouzeau, N: A General Framework for Cooperative Manipulation in Virtual Environments. In: Proceedings of the Eurographics Workshop on Virtual Environments, 1999, pp. 169–178.

[22] Matijasevic, M.: A Review of Networked Multi-User Virtual Environments. Technical Report TR97-8-1, Center for Advanced Computer Studies, Virtual Reality and Multimedia Laboratory. University of Southwestern Lousiana, USA, 1997.

[23] Park, K. S.—Kenyon, R. V.: Effects of Network Characteristics on Human Performance in a Collaborative Virtual Environment. In: Proceedings of the IEEE Virtual Reality Conference, 1999, pp. 104–111.

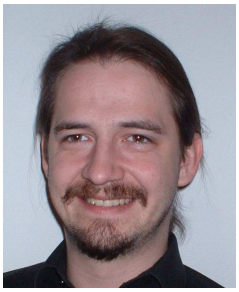
[24] Pandzic, I. S.—Chapin, T. K.—Magnenat-Thalmann, N.—Thalmann, D.: A Flexible Architecture for Virtual Humans in Networked Virtual Environments. In: Proceedings Eurographics '97, 1997.

[25] Roberts, D. J.—Wolff, R.—Otto, O.: Constructing a Gazebo: Supporting Team Work in a Tightly Coupled, Distributed Task in Virtual Reality. Presence: Teleoperators & Virtual Environments, Vol. 12, 2003, No. 6.

[26] Ruddle, R. A.—Savage, J. D. C.—Jones, D. M.: Symmetric and Asymmetric Action Integration During Cooperative Object Manipulation in Virtual Environments. ACM Transactions on Computer-Human Interaction, Vol. 9, 2002, No. 4, pp. 285–308.

[27] Singhal, S.—Zyda, M.: Networked Virtual Environments. ACM Press, New York, 1999.

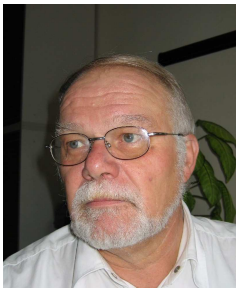
[28] Watsen, K.—Zyda M.: Bamboo – A Portable System for Dynamically Extensible, Real-Time, Networked, Virtual Environments. In: Proceedings of the IEEE Virtual Reality Annual International Symposium, Atlanta, 1998, pp. 252–259.



Christoph Anthes graduated as Dipl. Ing. (FH) at the University of Applied Sciences, Trier in 2002. Afterwards he went to Britain to successfully complete an MSc Course in network centered computing at the University of Reading. In 2003 he moved to Austria to work on his Ph.D. Currently he is working as a research assistant, lecturer and administrator in the field of VR at the Johannes Kepler University, Linz.

Adrian Haffegee is a Ph.D. student from the centre for Advanced Computing and Emerging Technologies (ACET) at the University of Reading. Prior to returning to university he spent 11 years in research and software development within the telecommunications industry. His current research interest involves virtual reality, specialising in collaborative virtual environments.

Paul Heinzlreiter received his masters degree in computer science from the Johannes Kepler University Linz in 2001. Since then he has been working as a scientific assistant and lecturer at the Institute of Graphics and Parallel Processing at the same university. His research interests include computer graphics, grid computing, virtual reality, networking as well as the interconnections between these areas.



Jens Volkert is Professor and Head of the Institute of Graphics and Parallel Processing (GUP) at the Johannes Kepler University Linz, Austria. His scientific interests combine advanced computer graphics (including virtual reality and collaborative virtual environments) and high-performance computing (with special attention to cluster and grid computing systems).