Seminar: Computer Game Technology - 2001

Massive Multi Player

Introduction to Multiplayer Network Technology

Thomas Lidy – 9825862

ABSTRACT

There is an increasing spreading of networked virtual environments and multiplayer games over the Internet. Reliable communication over a network is of great use in many different areas.
This paper gives an overview of the employed network technology, and the rise of Networked Virtual Environments, starting with the first simulations in military. Also, the first academic approaches and the first games in this field are introduced. The underlying protocols are presented and different kinds of networking issues are discussed.

1 INTRODUCTION

Network technology, widespread in today's Multiplayer Games, has its early roots in military and industrial simulations. The environments and abilities of interaction of 3D shooter games are quite similar to military training programmes. And also the network technology used in these games originated from software which was developed in military and industry first.

In this document, we will use the term "Networked Virtual Environment" (net-VE) as a common term for different scenarios. Networked virtual environments are being used for education, training, for distributed engineering and design, for commerce, and for games and entertainment – the range of applications is still growing. In all these areas, a reliable network communication technology is of great value.
What exactly is a networked virtual environment? In [1] a net-VE is defined as a software system in which multiple users interact with each other in real-time, even though they may be located all around the world. Users may participate in the net-VE with different machines (personal computers, consoles, scientific projection environments ...), but all these environments aim to provide users with a sense of realism by incorporating realistic 3D graphics and stereo (3D) sound to create an immersive experience. Networked virtual environments are distinguished by the following five common features:

- A shared sense of space: All participants have the illusion of being located in the same place (the same terrain, room or building, etc.). The shared space must present the same characteristics to all of the participants.
- A shared sense of presence: Each participant in a net-VE takes on a virtual person (called avatar), with a certain graphical representation. An avatar may be a human, an animal, a plant, an aircraft, or any other type of figure. In any case, when entering the net-VE, each participant can see the other avatars that are located in the shared space, and also those users can see the new participant's avatar. Similarly, when a user leaves the net-VE, other participants should see the avatar going away.
- A shared sense of time: Participants should be able to see each other's action as it occurs – that means a net-VE should enable real-time interaction to occur.
- A way to communicate: Though visualization forms the basis for an effective networked virtual environment, most net-VEs also enable some communication among the participants: by gesture, by typed text or by voice. This communication adds more realism to the net-VE, and it is a fundamental component of engineering or training systems.

● A way to share: A realistic net-VE enables users not only to interact with each other but also with the virtual environment itself. Users might collide with each other, or they need to shoot each other. Users should be able to pick up, move, and manipulate items, that exist in the environment, and they should be able to give them to other participants. It might even be possible to manipulate the environment itself by destroying buildings, building bunkers, etc.

The presence of multiple independent users differentiates net-VEs from standard virtual reality or gaming systems.
Networked virtual environments are most appropriate for applications that demand the illusion that other users are visible from remote locations. In these applications, users demand a sense of realism which otherwise only can be achieved by face-to-face contact.

A net-VE system consists of four basic components, which work together to provide the sense of immersion among users at different sites:

- graphics engines and displays
- communication and control devices
- processing systems
- and a data network.

One of the challenges facing net-VE designers is allocating processor time among the tasks required to support the user's net-VE presence.
As this document concentrates on network technology, we will focus on the Data Network. Participants in the net-VE rely on the network for the exchange of information. For example, as the user moves within the virtual environment, the computer must transmit positional updates over the network to the other users. Similarly, if a user picks up an object in the environment, other users need to be notified that the object is now being carried. The network is also used to synchronize the net-VE's shared state, for example time, terrain, and the weather. It also supports textual, audio and video communication among the users.

For many years, networked virtual environments could only be used at universities or at large military or industrial institutions with fast local area networks. Because networks had limited capacity and net-VE systems did not carefully manage how they used the network capacity, net-VEs could support only few simultaneous users. To support users from multiple sites, a net-VE needed to be deployed over a high-bandwidth private network. Internet capacity was not sufficient for the deployment of net-VEs.

However, the world of networking has changed dramatically over the past years. Modem speeds have quadrupled since 1993 from 14.4 Kbps to 56 Kbps, enabling users now to participate in a net-VE by connecting to an Internet Service Provider from home. With growing bandwidth there is also a potential growth of net-VEs and this leads to an increasing demand for Massive Multiplayer Games.
Still, there are certain challenges to solve with massive networking - these issues will be discussed in Section 3. But before that, we will go to the origins of Networked Virtual Environments.

2 ORIGINS

The first modern networked game / virtual environment can be dated as early as to "**Amaze"** [3] in 1984, but experimental net-VE systems have been around many years before ...
The **military heritage** of this field can be seen by the fact, that the US Department of Defense (DoD) is the largest developer of net-VEs for use as simulation systems, and it was one of the first to develop net-VEs with its SIMNET system.

SIMNET (simulator networking) is a distributed military virtual environment originally developed for DARPA. SIMNET was begun in 1983 and delivered to the US Army in 1990. The goal of the

SIMNET project was to develop a “low-cost” but high-quality net-VE for training small military units (tanks, helicopters, command posts, etc.) to fight as a team in a virtual battlefield.

The SIMNET network software architecture has 3 basic components:
1) An object-event architecture
2) A notion of autonomous simulation nodes
3) Predictive modeling algorithms – called “dead reckoning”

In the first version of SIMNET each change in the state of an object was reported immediately onto the network by sending an update packet. With small numbers of players, this was not a large worry, but it meant that for some objects packets were generated as fast as the simulation control program could generate them – that is, at frame rate. This rate of packet generation flooded the network and overloaded the CPUs. So the third component of SIMNET was introduced: a well-defined set of predictive modeling algorithms called “dead reckoning”. This “objects and ghosts paradigm” reduces the packet traffic: The other nodes in the system are maintaining “ghost” copies of the object, which are predicted on the basis of the last state of the object. Packets are only placed onto the network when the object’s home node determines that the other nodes on the network are no longer able to predict their state within a certain threshold amount.
This concept reduces the packet rate and therefore increases the amount of possible participants. In an exercise in March 1990, the SIMNET network software architecture proved scalable with some 850 objects (mostly semiautomated forces) at five sites. The objects averaged 1 packet per second.

The SIMNET protocol, defined in September 1989, did not define the network software architecture in a general form. It had not documented the packet formats and the architecture so that others could utilize it. This led to an attempt to formally generalize and extend the SIMNET protocol. The result was called the **DIS (Distributed Interactive Simulation)** protocol. The purpose was to allow participation in the DIS virtual environment by any type of player, on any type of machine. DIS was an enormous success and that success led net-VEs from being only of DoD concern to having more widespread availability and interest.

The DIS object-event architecture was designed to cover a broader spectrum of military simulation requirements than SIMNET. The core of the DIS network architecture is the protocol data unit (PDU). The DIS standard defines 27 different PDUs, only 4 of which are used by nodes to interact with the virtual environment:

- the entity state PDU (position, orientation, velocity)
- the fire PDU (a weapon has been fired)
- the detonation PDU (munition exploded, vehicle crashed or died)
- the collision PDU (vehicle collision detection)

The remainder of the defined PDUs are for simulation control, electronic emanations, and supporting actions; many DIS-compliant simulations do not implement these additional PDUs.
Each node in the virtual environment is responsible for issuing the appropriate PDUs, and each node also must maintain a state table for the other participants. Since packets in DIS are sent via unreliable UDP broadcast (see section 3), packets are sometimes lost. Hence, it is quite possible for displays and state tables to differ among different participating hosts. Also the implementation of collision detection is up to the individual node, and its accuracy often is quite dependent on the available CPU cycles. It is not uncommon for DIS-based VEs to have only partial collision detection or no collision detection at all.

The distributed and heterogeneous nature of DIS has been its success: In the DIS demonstrations everything from a large 20-processor SGI Onyx machine to a single Pentium PC were connected on the network. This has widened the scope of players in the world. However, there are some concerns with this variety of platforms, as the speed of processing may be an “advantage” for participants on low-end machines: slow polygon rendering may make opposing forces easily visible, or the worse collision detection may be a pleasure.

DIS was originally designed for small unit engagements having fewer than 300 participants; but the DoD wanted to be able to run real-time simulations with some 100,000 to 300,000 participants. There were several modifications of the DIS software architecture for these circumstances to achieve useful demonstrations. Some of the changes were delayed "heartbeat" packets (a host sends a "still alive" packet only every 30 to 60 seconds instead of every 5 seconds) and the use of packet compression.

In the area of networked games and simulations, there has been dramatic parallel development, also on low-end machines. We now present some of the most influential games and demos, those that have either developed new technologies or brought many people to realize the importance of net-VEs.

In the time period from 1984 to 1992, the demo program **Flight** was installed on every sold Silicon Graphics (SGI) workstation. Flight was inspired by the Blue Angels air shows at Moffett Field, which is directly across the street form SGI's original headquarters and was developed in the summer of 1983. In that program, the user selected one of the available airplanes and used the keyboard of the workstation to accelerate and steer. The primary challenge of Flight was to see if you were coordinated enough to land your plane. The purpose of the demonstration was to show the SGI workstation's capability to rapidly allow movement through small polygonally defined 3D virtual worlds. From 1984 on, networking was added to Flight. The initial networked version used a serial cable between two SGI workstations and ran at approx. 7 frames per second on a Motorola 68000 based workstation.

Some time after the release of the networked version of Flight (in 1985) SGI engineers modified the code of Flight to produce the demonstration program **Dogfight**. This modification dramatically upgraded the visibility of net-VEs, as players could now interact by shooting at each other. Everyone who bought an SGI workstation, had the two demonstration games, and many people were therefore inspired to develop their own net-VEs. SGI made the source code to Flight and Dogfight freely available, so that many people had the chance to learn about the development of networked virtual environments.

Later, also the PC world has taken the desire and interest for such connected worlds: In December 1993, **id Software** released its shareware game **Doom** – one of the first typical ego-shooters in a 3D space full of monsters and enemies. The shareware giveaway of the first level of Doom is probably singularly responsible for the rush of startups into the business of providing online gaming networks. Doom did no dead reckoning and flooded LANs with packets at frame rate. Hence, many network administrators were confronted with Doom, when their LANs started to go down. An estimated 15 million shareware copies of Doom have been downloaded around the world, passed from player to player by floppy disk, or online networks. The networked ability to blast in a believable 3D environment created enormous demand for further 3D networked games. id Software is still in the development of outraging 3D games; their first hits are presented on the id Software vintage page [4].

Many **other games** have served as inspirations to the net-VE community. On the Macintosh side, there is the 3D game **Marathon** (released in Dec 1994), played across Appletalk nets. Marathon has high-speed action in a 3D world, impressive graphics and spatial sound. Marathon won many awards and, like Doom, provided tools for editing and creating additional game worlds. Another networked Macintosh game worth mentioning is the tank game **Bolo**, played across Appletalk, and later over IP networks. Unlike most games of the time, Bolo avoids broadcast protocols. Instead, the participant hosts organize themselves into a virtual ring. Though latency proves to be a problem as the number of participants grows, Bolo had a cultlike success, because it did not flood the network and so there was no reason for system administrators to remove it.

Academic Networked Virtual Environments

On the Naval Postgraduate School, the **NPSNET** originated in the framework of a computer graphics class project taught in 1986. Two students developed a visual simulator for the FOG-M (fiber-optically guided missile) system, which simulated the launch and the steering of a missile which was

targeted on on-ground objects – a stopped tank in the first version. Because it was desired to be able to move the tank (or the target), the group began the development of the FOG-M follow-on system, which supported the movement of ground vehicles – through networking.
The NPSNET research group focused its efforts on developing virtual environment technology that can be useful for the DoD; by that way, it also learned about SIMNET. NPSNET-1 was demonstrated live at the SIGGRAPH '91 conference. Later, NPSNET was split in the development: NPSNET-2 and NPSNET-3 were utilized to explore better, faster ways to do graphics and to extend the size of the terrain databases. NPS-Stealth was spawned off from NPSNET-1 with the goal of a system being capable of interoperating with the SIMNET system.
NPSNET-IV [5], released in 1993, was DIS-compliant and did dead-reckoning. It had many capabilities which made it one of the most ambitious virtual environments of its days. For example, a player in NPSNET-IV can walk through buildings and up stairs and ladders... NPSNET-IV was the first net-VE that was Internet-capable, being able to play across the multicast backbone (MBone).

The **PARADISE** (Performance Architecture for Advanced Distributed Interactive Simulation Environments) project, initiated in 1993 at Stanford University, was explicitly addressed to networking issues facing environments with thousands of users. To reduce bandwidth, a hierarchy of area of interest (AOI) servers collects information subscriptions from each host (IP multicasting was used – see section 3). At the same time, the designers of PARADISE tried to correct several of the mistakes made by DIS. The value of the PARADISE system lies in its attempt to reconsider many of the "standard" assumptions of existing net-VE systems in the context of future network capabilities and future scalability requirements. This resulted in several advances in core simulation software technology.

At the National University of Singapore, a virtual environment toolkit that provides support for graphical, behavioral and network modeling of virtual worlds – called **BrickNet** – was started in 1991. The focus was to allow VEs to share individual objects rather than the entire content. BrickNet allows objects to be shared by multiple virtual worlds. It is primarily aimed at collaborative design environments, where a complete design task is distributed over multiple client workstations. But with the extension of behavior sharing, BrickNet was also used for games and applications. The primary contribution of BrickNet in the early net-VE arena is that it explored the client-server model and did not require each client to have a complete copy of the net-VE database. Unfortunately, in a dynamic world, servers are bottlenecks, increasing the latency of the net-VE.
After BrickNet, in 1996, the same institute began with the development of a new toolkit, called **NetEffect**. NetEffect is a software architecture for developing large, low-end virtual environments. It is targeted for PCs, with potentially several thousand geographically distributed, connected users.

Of course, there are several other net-VEs which are worth mentioning in this document, but with this overview of the origins of net-VEs you have at least an idea of the different approaches and directions the developers have gone.

3 NETWORK TECHNOLOGY

3.1 Overview

A network is obviously an essential feature of a networked virtual environment. In contrast to a virtual environment running on a single machine, which exchanges information between its different components instantaneously through a function call, the presence of a network brings up a number of issues that determine the design and implementation of a net-VE. The exchange of information over a network may take considerable time, and even worse, the data might get lost on the way to its destination. Thus, the designer of a net-VE has to choose carefully the technologies he employs in the net-VE, including the communication protocols he uses. We will describe the communication protocols and techniques which are in use on the Internet, in Section 3.3. First we will talk about the basic networking issues.

3.2 Networking Issues

When sending information over computer networks, there are some basic issues, which one need to know. A computer network is much like a complex maze of roads connecting cities together. To achieve an efficient flow of data, it is important to know about the following terms:

- Latency

The network latency, or delay, is the amount of time required to transfer a bit of data frome one point to another. When a net-VE application transmits a packet of data over the network, the network latency determines, when the application at the receiving host actually sees the transmitted data. This delay represents one of the biggest challenges to a net-VE designer. The latency directly impacts the realism of the net-VE experience because it determines how up-to-date the information which was received over the network is. Unfortunately, net-VE designers can do very little to reduce it. The latency is the sum of the time needed for transferring the data over the wire, for passing a modem and the network hardware and for travelling through the computer's operating system. Also, the delays of passing routers or even travelling up to satellites introduce an additional amount of latency.

- Bandwidth

The network bandwidth is the rate at which the network can deliver data from the source to the destination host. The available bandwidth is determined by the type of wire used to transport data, and it is also limited by the hardware used to transmit the data. For example, a modem can handle data up to the rate of 56 K bits per second, many long-distance leased lines can support between 1 and 2 Megabits per second. Ethernet typically handles the data with 10 or 100 Mbps. Note the distinction between network bandwidth and network latency: Latency is the delay of transfer, bandwidth is the rate of transfer. It measures how many bits can be transported per unit of time. Another way to think of it is how many bits would pass by you per second if you stood at a particular place on the wire.
Bandwidth and latency are not necessarily related.

- Reliability

Network reliability is a measure of how much data is lost by the network on its way from source to the destination host. Data might be lost on the network (e.g. due to electrical interference), or while passing routers, or it might be corrupted. Data corruption means, that the content of the packet has been changed during transmission so that the arriving data is basically useless to the destination host. In either case the data effectively does not arrive at the destination application in a usable form.
To address these issues, designers typically include sufficient redundancy to detect data corruption (CRC checks, error-correcting codes, etc.). To ensure, that the data arrived at the destination, the destination computer can send acknowledgements back to the source host. The acknowledgement can be either a special packet, or it may be included ("piggybacked") inside another data packet transmitted by the destination computer to the source host. These acknowledgements notify the source that the data has arrived successfully. If the source does not receive an acknowledgement with in a certain time after sending data, it assumes that the data was lost and attempts to send it again.

3.3 Network protocols

A network protocol describes the set of rules that two applications use to communicate with each other. There are very many network protocols in use today. Each one is specialized for a particular task, ranging from downloading documents from the WWW to exchanging real-time audio and video. Moreover, when two applications communicate with each other, they are typically using many protocols simultaneously. Transporting data from one host to another host is a complex task. The data passes several layers, each of which has a particular protocol. Additionally, there are protocols for communicating between a modem and a dial-up server and for exchanging information among routers.
TCP for example, is a protocol of the transport layer in the OSI (Open Systems Interconnection) reference model, whereas IP resides on the network layer. For details about the layers in the OSI reference model, see [2], p. 71. The most important protocols are described in the following sub-sections.

Sockets and Ports
On a particular host, several applications may be executed concurrently and multiple applications may use the network. These applications may be communicating with different destinations at once, or with different applications on a single destination host. They are likely to use several different protocols in communication. Hosts therefore must be able to keep track of all this communication. When an application sends a packet, the host must make sure, that it gets sent to the right destination, and the destination host must be able to deliver it to the correct application. To achieve this, most hosts on the Internet use the BSD (Berkeley Software Distribution) Sockets network architecture:
A socket is a software representation of the endpoint to a communication channel. All applications that use the network communicate over a socket. The Sockets architecture allows multiple networking applications to use the network simultaneously. A socket identifies several pieces of information about a communication channel:

- the protocol (TCP, UDP, ...)
- the destination host address
- the destination application ID (i.e. the port number; 16-bit)
- the source host address
- the local application ID (local port)

By including the local address and port ID in the packet, the source host ensures that the destination host will be able to send back reply packets. For each protocol, each of a host's sockets is assigned a different 16-bit integer by the operating system (see figure 1). Every application grabs a port number and by that means it makes sure that other applications can connect and send data to it.

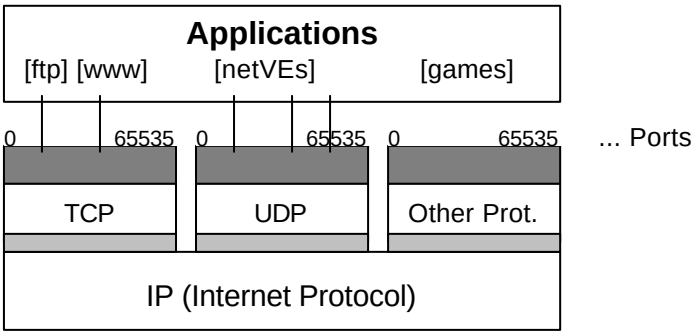

Figure 1: Sockets and Ports architecture

IP – The Internet Protocol
IP is a low-level protocol used by hosts and routers to ensure that the packets travel from the source host to the destination host. The Internet Protocol makes sure, that packets are split into small fragments, when they traverse network links that cannot support large packets and that they are reassembled on the other end. The IP header also includes a "Time-to-Live" (TTL) field that specifies how many network hops may transfer the packet. Each time the packet is transferred by a router, the TTL field is decremented, and if it reaches zero, the packet is discarded. This ensures, that packets are not routed in infinite loops around the Internet.
Applications almost never use the Internet Protocol directly. Instead, they use one of the protocols that are written on top of IP, which are described next.

The various protocols take advantage of the basic services provided by IP and extend those facilities to provide services that are suitable for applications to use for transmitting their own data. The protocols in common use today allow programmers to decide how to best balance the need for efficiency (low bandwidth, low latency), against the desire for good transmission semantics (reliability, sending to one or multiple destinations, etc.). In general, the more services that one desires from a protocol, the more it costs in terms of efficiency and scalability. When building a net-VE, the designer must take care to choose the right protocol.

TCP – Transmission Control Protocol
TCP (often referred to as TCP/IP because it is layered on top of IP) provides the application with the illusion of a simple point-to-point connection. It ensures reliability through automatically transmitting acknowledgments and checking data packets against a data checksum which is included in the packet header. A TCP connection can be regarded as a bidirectional reliable stream of bytes between two endpoints. TCP discards duplicate packets and automatically inserts the packets in the correct order into the byte stream the application reads. The price, however, is that it must transmit more information to describe data ordering, checksums to detect corruption, and acknowledgement / retransmission packets. Consequently, though TCP/IP is useful for a large variety of applications, it is not suitable for applications that do not necessarily need the strict ordering and consistency guarantees it provides.

UDP – User Datagram Protocol
The power of UDP lies in its simplicity. UDP differs from TCP in three respects: connectionless transmission, best-efforts delivery, and packet-based data semantics. UDP makes no attempt to guarantee that data is delivered reliably or in order. The sender must rely on other information (such as responses from the recipient) to determine, whether the destination is still alive and whether the data has arrived. The endpoints do not maintain any state information about the connection; UDP data is sent and received on a packet-by-packet basis. Because it does not include the overhead needed to ensure reliability and maintain the connection, UDP packets require considerably less processing at both the transmitting and receiving hosts. Packets can be transmitted as soon they are sent by the application. Thus, UDP/IP is more appropriate for large-scale distributed systems. A number of techniques have been adopted for detecting and dealing with lost data. In many cases, lost data is not even important.
However, UDP introduces a security problem for applications that do not robustly distinguish between expected and unexpected packets. Many network firewall administrators block UDP data and therefore, full Internet UDP connectivity cannot be assumed.

IP Broadcasting
When an application needs to send a packet to multiple destination, it could repeatedly call the sendto() function (in C) to send multiple copies of the packet. This approach, however, has two disadvantages: Because the packet is sent over the network multiple times, excessive network bandwidth is required, and each host must maintain an up-to-date list of all other application endpoints who are interested in the data.
IP broadcasting provides a partial solution to these issues by allowing a single transmission to be delivered to all hosts on a network. When a net-VE participant starts, it can simply start listening for data on the application's well-known port and start broadcasting its own data on that same port. Broadcasting can be simply realized by sending the data to the address 255.255.255.255. Packets sent to this address are guaranteed to be delivered to all hosts on the local LAN. The disadvantage of this is, that broadcasting is expensive because every host on the network must receive and process every broadcast packet, even if no local application is actually interested in receiving the data. Furthermore, broadcasting cannot be used for Internet-based net-VEs. For large net-VEs and massive multi player games, IP multicasting would be more appropriate.

IP Multicasting
IP multicasting is not only useful in a LAN, it is also appropriate for Internet use. It does not impose burdens on hosts that are not interested in receiving the multicast data.
To understand Multicasting, it is useful to consider a process for distributing a newspaper. Imagine that the newspaper needs to be distributed to subscribers around the world. The paper might be printed in London, and one copy would be sent to each of four major distribution sites – say Washington, Vienna, Moscow and Tokyo. Each of these distribution sites can take the copy that it receives and make copies for a set of subdistribution sites, as seen in the example in figure 2. Each of the subdistribution sites makes a copy for every subscriber and sends those copies. Alternatively, a subdistribution site may send a copy to an even smaller distributor who may have its own set of subscribers.

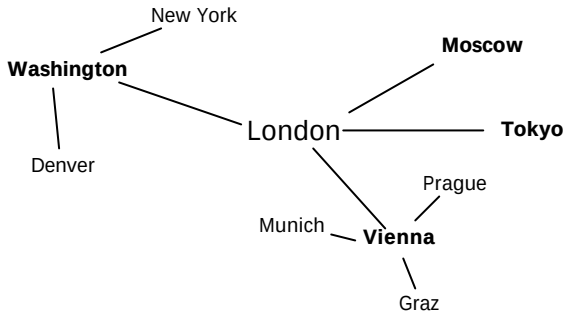

Figure 2: Distribution of a Newspaper – shows the distribution and sub-distribution scheme of Multicasting

This is the principle, how multicasting over networks is realized. The "distributors" are the multicast-capable network routers that are responsible for determining which destination networks should receive a particular multicast packet. Multicast distribution has several important properties:

- It is assured, that data is only sent to destinations, where it is wanted.
- Subscribers need to explicitly request the information from their local "distributors".
- Duplicate copies of the information are never sent down the same distribution path.
- The host, which issues the information, does not need to know the identities of all of the subscribers, just the address of the next subdistributor.

The multicast routers are responsible for making new copies of the data when it is necessary to send it down different distribution paths. In this respect, the distribution is quite efficient. It is called "receiver-controlled distribution". This scheme consumes considerably less overhead in communication than IP broadcasting.
When sending multicast data, an application can specify the IP Time-to-Live (TTL) field to control how far multicast packets should travel. The TTL field can take on values between 0 and 255; some values represent different ranges in the scope of the network (a value of 1 for example means delivery only on the local LAN).
To send data with IP multicasting, special addresses have to be used. Each multicast distribution tree is represented by a special pseudo-IP address called multicast address or class D address. IP addresses in the range 224.0.0.0 through 239.255.255.255 are designated as multicast addresses – but some address ranges are reserved (for details see [1], page 73). If the net-VE application does not require a permanent address, it can choose to use one temporarily. However, it is desirable to avoid having multiple applications using the same address at the same time. Such collisions might cause each application to unexpectedly receive packets from another application. These unexpected packets may be hard to distinguish from the packets that the application actually expects and they could additionally introduce a security problem.
If the net-VE will use a small number of multicast addresses on a regular basis, or will be deployed commercially, the net-VE developer may want to permanently reserve a set of multicast addresses through the IANA [6].

On the academic side, multicast is rapidly emerging as the recommended way to build large-scale net-VEs over the Internet. However, it does have some limitations, generally related to its infancy. Although an increasing number of routers are multicast-capable, many older routers are still not capable of handling multicast subscriptions. In the meantime, multicast routers communicate directly with each other, "tunneling" data past the routers that cannot handle multicast data. This Multicast Backbone (MBone) provides its participants with the illusion of a fully connected multicast-capable Internet. However, a host cannot participate in a multicast-based net-VE if its local router is not itself multicast-capable and connected to the MBone. Although multicast is rapidly becoming universally available, it is not quite there yet.

Additional information about Multicast Routing in internetworks can be found in [7], details about the Multicasting algorithm itself in [8].

4 CONCLUSION

Networking is an essential feature of networked virtual environments and networked multi player games. A designer cannot develop such applications without the understanding of sockets and ports and the protocols which are used for networking. One of the first decisions faced by a designer is to select an appropriate communication architecture. This communication architecture defines how data will flow among the participating hosts in a net-VE. As we have seen, the choice of communication architecture and of communication protocol go hand-in-hand. We have presented the common network protocols: unicast TCP/IP and UDP/IP, IP broadcasting (using UDP), and IP multicasting. Each communication technique provides a different level of reliability, functionality and scalability. When building net-VEs, one must be sure to select the most appropriate protocol for the application requirements.

For the implementation of large-scale networked virtual environments, the conclusion would be that the best choice is multicasting, because it provides desirable network efficiency while also allowing the application to partition different types of data through the use of different multicast addresses. Unfortunately, Multicasting is not yet available everywhere in the Internet. – In fact, most of the networked games today do not use Multicasting. They provide their users with communication through a sophisticated client/server architecture. For supporting many simultaneous users – without introducing a bottleneck – the developers consequently have to ensure, that they scale their hardware to be sufficient enough for processing a massive amount of players.

Multicasting will probably become universally available after the global introduction of IPv6, the next version of the Internet Protocol, which introduces several essential extensions to the current protocol [9].

REFERENCES

[1] S. Singhal, M. Zyda: Networked Virtual Environments; ACM Press/SIGGRAPH Series, 1999

[2] Coulouris, Dollimore, Kindberg: Distributed Systems – Concepts and Design, Second Edition, 1994

[3] E. J. Berglund and D. R. Cheriton. Amaze: A distributed multi-player game program using the distributed V kernel. In Proceedings of the Fourth International Conference on Distributed Systems. IEEE, May, 1985.

[4] id Software
http://www.idsoftware.com/killer/vintage.html

[5] NPSNET: A multi-player 3D virtual environment over the Internet;
Michael R. Macedonia, Donald P. Brutzman, Michael J. Zyda, David R. Pratt, Paul T. Barham, John Falby and John Locke; Proceedings of the 1995 symposium on Interactive 3D graphics, 1995, Page 93

[6] IANA – Internet Assigned Numbers Authority
web site: http://www.iana.org

[7] S. E. Deering; Multicast routing in internetworks and extended LANs;
Symposium proceedings on Communications architectures and protocols, 1988, Pages 55 - 64

[8] Anna Hac; Distributed multicasting algorithm in a wide area network;
Proceedings of the 1990 ACM annual conference on Cooperation, 1990, Pages 37 - 42

[9] IPv6 Information Page: http://www.ipv6.org