DOCTORAATSPROEFSCHRIFT

2007 | School voor Informatietechnologie
Kennistechnologie, Informatica, Wiskunde, ICT

An Architecture for Large-scale Virtual Interactive Communities

Proefschrift voorgelegd tot het behalen van de graad van
Doctor in de Wetenschappen: Informatica, te verdedigen door:

Peter QUAX

Promotor: prof. dr. Wim Lamotte
Copromotor: prof. dr. Frank Van Reeth

Universiteit Maastricht

universiteit hasselt

D/2007/2451/27

Acknowledgments

To provide an exhaustive list of all people I've worked with throughout these years is a near impossible task, so I'll try and list those that have contributed in some way or another to the realization of this text.

First and foremost, thanks to prof. dr. Wim Lamotte and prof. dr. Frank Van Reeth for their expert guidance. Whilst searching for a topic for my PhD they came up with the idea of starting research on Networked Virtual Environments – something that combined my interests in computer networks and multimedia in general. Throughout the years, they have continued to provide me with valuable input for scientific publications and novel directions to take. At the same time, the input of the senior research staff at EDM, prof. dr. Eddy Flerackers, prof. dr. Karin Coninx, prof. dr. Chris Raymaekers and prof.dr. Philippe Bekaert allowed for interesting discussions.

Thanks also go out to my friends and colleagues prof. dr. Kris Luyten, dr. Tom Van Laerhoven and Jori Liesenborgs, whom I've known since the start of my studies at the former LUC. Kris and Tom for their support in all things related to our everyday tasks at EDM and their off-duty friendship, Jori for his help in generating the initial ideas on ALVIC and design of the software components.

From its (relatively modest) start, the NVE group at EDM has expanded over the years to include several people I enjoy(ed) working with : Tom Jehaes, Pieter Jorissen, Maarten Wijnants, Patrick Monsieurs, Stijn Agten, Bjorn Geuns, Jeroen Dierckx and Bart Cornelissen. Thanks also to Erik Hubo, William Van Haevre, prof. dr. Fabian di Fiore, dr. Koen Beets and Panagiotis Issaris.

I'd also like to extend my thanks to Peter Vandoren, Ingrid Konings and Roger Claes for taking care of the formalities associated with the successful completion of a PhD.

Thanks to Danny De Vleesschauwer and Natalie Degrande of Alcatel-

Lucent for their cooperation in the IWT MOVE project, which resulted in the findings described in chapter 4.

Finally, of course, thanks to my parents for supporting me during these years of study.

Diepenbeek, July 2007.

Abstract

Following the popularity of text-based Internet chat channels and instant messaging applications, virtual interactive communities are often advertised as 'revolutionary' applications that allow large groups of people to share ideas, experiences and make new friends. What is largely unknown by the general public is that these applications share the underlying technology with networked computer games, and were founded in research for military purposes.

We start this dissertation by analyzing the behavior (in terms of bandwidth usage) of applications based on networked virtual environments technology, such as networked games and virtual interactive communities. Also, the impact of common network-related problems, in this case delay and jitter, on user performance and perceived quality is quantified.

Afterwards, we discuss the development of ALVIC, an Architecture for Large-scale Virtual Interactive Communities. The unique feature of ALVIC is certainly its scalability, enabling both large amounts of simultaneous users to interact, as well as allowing them to experience the vastness of a virtual world.

Several building blocks make up the ALVIC framework, starting with the subsystem that enables users to view the three dimensional world on computing devices located all over the world. It is of prime importance that each of the participants is, at any one time, looking at the same world. We will describe how messages are exchanged between users, using advanced network techniques that enable efficient distribution to large groups of people. Also, a mechanism for the simulation of large audiences is described using software that emulates behavior of human users using elementary artificial intelligence techniques.

Furthermore, the ability for people to interact directly, through means of video communication, is integrated into the ALVIC architecture. Especially when considering large audiences, the requirement of controlling bandwidth

is stringent. Also, due to limitations in resource availability on a computer, optimizations are included to increase the subjective quality of experience.

Finally, as ALVIC was designed to be used as foundation for the deployment of a range of applications, from games to teleconferencing systems, we will look at the upcoming market of mobile applications. We will show that it is feasible to deploy applications based on the ALVIC architecture on a range of devices, taking into account their individual limitations.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

When we started our research on the topic of networked virtual environments, it was an uphill struggle to explain to users what the research topic was all about. At that time, the only (remotely) related well-known applications were networked first person shooter games (on local area networks) and Internet chat services (such as the IRC network). Nowadays, if anyone asks about this particular topic of research, one just needs to mention applications like Second Life or World of Warcraft, which have practically become household names. Because of all the media attention these applications have drawn, they immediately provide anyone (with at least a passing interest in computer science) with a rough idea of what networked virtual environments actually are.

Over time, several formal definitions were developed for networked virtual environments, a lot of them using non-trivial terms such as 'user embodiment' or 'immersion'. However, it is quite feasible to describe what they actually are by simply dissecting the term into its components.

First of all, the 'networked' part references the fact that users need not be co-located to use these applications, but can be distributed over various locations, as long as a network connection is available to exchange vital data between them. It also shows that these environments are typically targeted towards larger groups of users than just a single home user sitting behind his/her PC and his/her direct neighbor.

The 'virtual' adjective is added to suggest that the application and everything it contains is entirely computer-generated, and in fact only exists as long as it is being supported by a computer infrastructure. While links may exist

between the application and reality, the 'virtual world' is in fact self-sustaining and exists entirely for the sake of the application.

Finally, the 'environments' section of the research topic references the fact that everything is visually represented in three dimensions, enabling users to interact as if they are actually present in the virtual world themselves. This is also referred to by the term 'immersion'.

The combination of these three parts should provide a basic overview of what the topic is all about: three-dimensional computer-generated environments in which a possibly large number of users can interact. While this effectively summarizes the general context, we should immediately point out that several sub-topics can be identified to warrant further study. For example, one may wish to study the ways in which people interact in these environments, the underlying network aspects or how the virtual world is visualized in greater detail. This indicates that networked virtual environments as a topic of research is really an aggregation of several existing domains of research, combining techniques from computer networks, computer graphics and human-computer interaction into a single application.

In this text, the focus is definitely on underlying network technology needed to support these applications. However, it simply does not suffice to have a state-of-the-art network architecture, without a means of visualizing the environment or allowing actual user to interact and test the application. This is why, at several points in this text, references will be made to research carried out by other members of the research group at EDM. It is only through collaboration between researchers who specialize in the three main identified subtopics (networking, representation and interaction) that a successful application can be devised, implemented and deployed.

1.1 Research contributions

Several research projects have focused on the topic of networked virtual environments. We will provide an overview of these in the next section, but will first summarize the essential research contributions contained in this text.

In chronological order, we have investigated and will discuss:

- the impact of currently existing NVE applications on the underlying network in terms of bandwidth.
- the impact the network has on existing NVE applications because of several imperfections.

- the development of a basic supporting architecture to synchronize the view of the virtual world on a large number of end-user devices, using state-of-the-art network technologies.
- ways of determining the actual scalability of an application using our supporting architecture.
- the extension of the architecture to support inter-person communication with a focus on video.
- the ability for clients to adjust the amount of incoming network data, optimizing the (limited) resources available.
- the changes required in terms of network architecture when deploying these types of applications on mobile devices.
- the supporting infrastructure needed for remote visualization of complex environments in order to target the heterogeneous world of mobile devices.

1.2 Overview

In part I, we will first show the bandwidth usage of a number of representative networked virtual environment applications. The figures presented in this chapter were obtained using a customized test setup, and analysis results through partial reverse-engineering of client/server connectivity behavior. Afterwards, we will discuss the impact of network deficiencies – more specifically delay and jitter – on a first person shooter game. Both objective as well as subjective studies were performed, resulting in a so-called ping-threshold for the type of game studied. Results obtained here are indicative of similar boundaries for other types of applications.

In Part II, the basic architecture that is used in the dissertation is presented. It includes a state synchronization that is specifically developed with large numbers of simultaneous users in mind. Multicast is used as the underlying distribution method, as it is a technology that has several advantages in this context. Next, a novel method for testing scalability of the architecture is described. This methodology is subsequently used to derive actual figures on bandwidth usage and prove the scalability up to thousands of users using only a single server.

Part III focuses on the extensions needed in the architecture to support various forms of communication. More specifically, we will look at real-time

video support through an integrated means of visualization for lots of simultaneous users, the optimal distribution method (at network level) and some issues and solutions when deploying the architecture on current-generation networks. Client-controllable bandwidth usage is the main topic of discussion here. The technique discussed in the previous part is re-used to confirm the scalability of the proposed scheme.

Finally, in part IV we discuss the migration of networked virtual environment applications towards mobile devices and networks. First, the impact of mobility on the network architecture is described, both in terms of test results of actual mobile networks, and a theoretical discussion on possible extensions for use on various types of mobile networks. In the second half, we will look at the integration of remote visualization of environments in the architecture using video streams, a technique that is especially useful for those devices that lack the processing power for local rending.

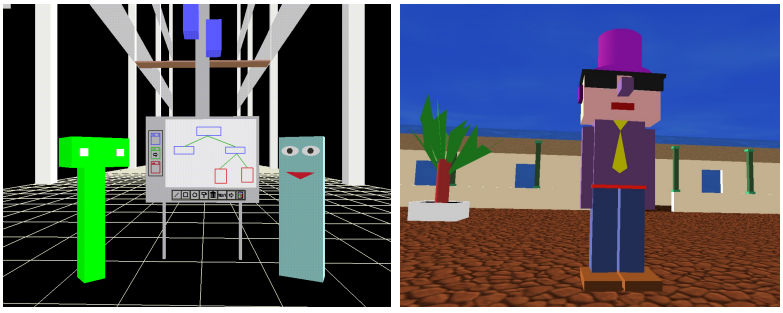
Chapter 2

Historical overview

The earliest examples of networked virtual environments can be traced back to the early 1980s. The DARPA (Defense Advanced Research Projects Agency) of the US military had, for a few years leading up to that moment, been using computers to simulate various aspects of warfare. These simulations, run on individual computers, were used successfully to train commanders to take tactical decisions and to distribute troops and equipment in an efficient way on the virtual battlefield. However, those responsible for the simulation program quickly determined that an important aspect was missing from the simulation, namely the interaction with human opponents or allies. In 1983, the SIMNET [Calvin 93] project was launched, with as its main goal the interconnection of the various types of simulators in use at the time (see figure 2.1). It quickly became clear that, for the project to be successful, a standardized network protocol would be required to transport essential data between simulation end-stations. This resulted in the first version of the DIS protocol (Distributed Interactive Simulation), of which several versions were developed over the years and ratified as IEEE standards. In fact, it is still being extended today. The information transmitted by the DIS protocol is referred to as *state*. In its simplest form, state describes the relevant information about entities that is required by other users to be able to visualize or simulate them. Examples of state information in general are position, orientation and action information; in the SIMNET architecture these are obviously military-inspired: e.g. collision, fire, minefield and resupply. in SIMNET, state is encapsulated within PDUs (Protocol Data Units), which define the syntax and parameters of each message transmitted over the network. *Syn-*

(a) SIMNET aircraft simulator (b) SIMNET environment

Figure 2.1: SIMNET



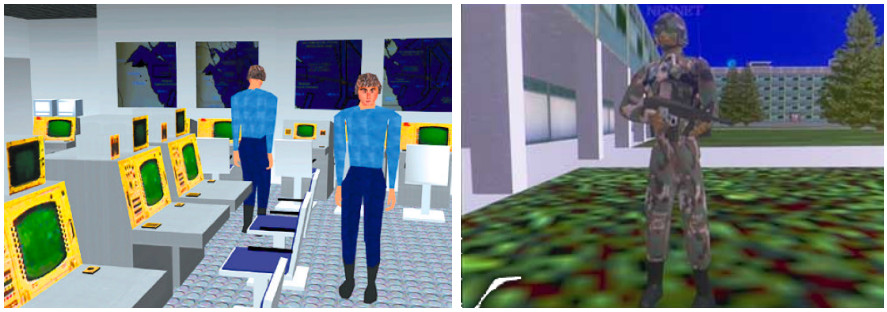
(a) Sample DIVE scene (b) DIVE blockie avatar

Figure 2.2: DIVE

chronization refers to the process of ensuring that the visual representation of the virtual world on all connected systems is kept (more or less) the same. If the latter is the case, then one can say that the state is *consistent*. For example, if, in SIMNET, a tank fires a missile, it is critical that the launch is visualized at the same time on every other computer that takes part in the simulation. Consistency is achieved by exchanging (synchronizing) state information and calculating parameters relevant to the simulation process (for example: adjusting clocks to the same time, according to a central clock or: determining actual state from previously received state updates). Depending on the application, synchronization algorithms and consistency requirements may be more or less relaxed. Several years after the initial versions of SIMNET, the software architecture supporting interoperability between and reuse of simulation was 'standardized' as the High-Level Architecture (HLA).

In the latter half of the eighties, little interest was shown by the research community to further develop the ideas presented in SIMNET into more general-purpose applications. It wasn't until 1991, when the Swedish Institute of Computer Science released the first version of the DIVE [Frécon 98] software (see figure 2.2(a)), that the first real non-military networked virtual environment was developed. In fact, unlike SIMNET which was designed to run on tightly-controlled and dedicated networks (witness the fact that it used broadcasting), DIVE was developed with deployment on the Internet in mind. The creators of DIVE investigated several means of visualizing the presence of other people in the environment. This virtual representation of the user is referred to as an *avatar* [Benford 95], coming from Indian philosophy, where the term references the incarnation of a divine being on earth In one of the applications based on the DIVE platform, these avatars were called 'blockies' (see figure 2.2(b)), after their visual appearance (based on regular cubes and blocks). Though simple, they were clearly effective enough for the simple forms of interaction featured in the early versions of the DIVE platform. Also supported by DIVE was virtual *collaboration*: a specialized form of interaction, whereby multiple users join effort to achieve a common goal. Usually, collaboration puts higher demands on the underlying synchronization system than other, simpler, forms of interaction. This is due, for example, to the fact that manipulations often take place on the same object, and the system needs to make sure that the order in which manipulations are carried out is the same as the input of the various participating users. In early versions of DIVE, the entire *world database* (a data source that contains the state of all objects present in the virtual world) was maintained (*replicated*) at all connected stations. Each time a change was required, all local copies of the database had to be synchronized, clearly a resource-intensive task. To ensure that databases did not go out of sync, reliable multicast was used to distribute change messages. These early versions did not scale well beyond 10 simultaneous users on a LAN. Through various enhancements, such as the improvement of the replication mechanism and a better reliable multicast protocol, later versions would scale to 20 users.

Based on the findings of the SIMNET project, and using the DIS protocol developed in this context, the Naval Postgraduate School of Monterey, California started work on NPSNET [Macedonia 95b, Macedonia 94, Macedonia 95a], a networked vehicle simulator (see figure 2.3). A first version was described in 1994. The most important advantage over the earlier SIMNET simulations, besides the improved visualization and increased number of supported objects/entities) was that NPSNET was able to be deployed on the Internet using



(a) NPSNET environment (b) NPSNET avatar

Figure 2.3: NPSNET

multicast through the MBone. More details on MBone will be discussed in part II of this text, but suffice to say that it provides a way of sending multicast messages over a (mainly) multicast-agnostic network, except for small 'islands'. NPSNET was also one of the first to introduce the notion of an *area of interest* or *AOI*, a virtual 'bubble' which encompasses those objects directly relevant to the end-user (mostly because they were visible) in order to minimize processing load and network bandwidth usage. Several versions of NPSNET were released over the years, leading up to the current one (number 5) [Capps 00].

The MASSIVE [Greenhalgh 95] environment (Model, Architecture and System for Spatial Interaction in Virtual Environments) extended upon this notion of an area of interest to include a more detailed specification of what was important to an end-user in terms of other objects in the vicinity. The *aura*-concept was developed here, representing the distance (originating from the location of an object) to which interaction with other objects is possible. It is only if the auras of two objects intersect that interactions can take place, a condition that can be monitored by a central entity. If this condition is met, peer-to-peer connections can be set up to exchange information. MASSIVE also introduced *awareness*, which represents the relevance that is attributed to other objects (which may be a subjective or objective measure). Both concepts are of primary importance and they are present, in one form or another, in nearly all current networked virtual environments. The underlying network distribution method used in MASSIVE deviated from previous work, in that it uses peer-to-peer connections (unicast) rather than multi-or broadcast. Possibly for that precise reason, the architecture was not as scalable as it was made out to be, supporting a relatively low number of 10 simultaneous users over

(a) Diamond park environment (b) SPLINE-based application

Figure 2.4: Diamond Park and Spline

the Internet.

BrickNet [Singh 95] (developed in 1995) uses yet another architecture, based on client/server connectivity. BrickNet servers maintain state of objects and handle client requests for the state of the objects under their control. The servers also keep track of open connections with clients to exchange information. The real power of BrickNet is that it allows not only for the geometry of objects to be exchanged, but also their behavior (through a general purpose programming language). Using centralized servers to manage state enables strict enforcement of *locking* and *floor control* (determining who has the right to alter the state of an object). Another advantage is that BrickNet applications are immediately usable over the Internet, as the communication channels are unicast-based.

In the same year as BrickNet, the Mitsubishi Electric Research Laboratory - or MERL for short- demonstrated Diamond Park [Waters 96a] (see figure 2.4(a)), a 'social virtual reality system' based on the SPLINE platform [Waters 96b]. SPLINE (figure 2.4(b)) abstracts from the notion of an object being an element that can be visualized, but rather uses an object-oriented database to store all types of information (including audio and autonomous behavior definitions). This database is replicated at client-side to improve interactivity. However, as the SPLINE platform was developed with scalability to several thousands of users in mind, the consistency requirements are relaxed. In practice, this means that the various copies of the database in use throughout the set of connected computers do not have to be exactly synchronized at all times. Although such an approach may introduce consistency problems when two clients are altering the same object at the same time, the developers of SPLINE felt that the 'sense' of interactivity, determined by the

reaction speed of the application, was more important. SPLINE also uses a system of *spatial subdivision* (called locales [Barrus 96]) to split the (possibly large) virtual world in manageable chunks. Information is only exchanged between entities in the same locale. Although the first versions of SPLINE used only peer-to-peer traffic, later versions introduced servers (with limited responsibilities) to optimize traffic flows and to keep track of objects between locales. The distribution method for the bulk of information (server-to-server and between high-capacity-peers) is multicast.

In 1996, a paper on MASSIVE-2 [Greenhalgh 96] was published, in which several enhancements were discussed to solve the scalability problems that plagued version 1. Through an intelligent mechanism that was able to 'group' several objects, the spatial scope could be detailed further, thereby limiting the propagation of state messages over the network. The second version also uses IP multicast to distribute various types of media among clients.

Similarly, version 4 [Zyda 97] of the NPSNET architecture improves on the original by optimizing the data flow in and between multicast groups. It also introduces some novel features, such as the integration of video communication and (at least in theory) support for mobile networks.

After NPSNET-IV, it seems like the research community lost interest in the topic of networked virtual environments, instead focussing on the various components that make up such a system (such as better visualization, AI support etc). The entertainment industry, on the contrary, was only just beginning to show interest in the possible applications of the technology. Based on their experience with Multi-User Dungeons (MUDs) - a type of game traditionally played by communities on Bulletin Board Systems -, the logical next step was to migrate these types of games to a three-dimensional environment. Of primary importance to this type of applications was the *persistence* of the virtual world, as it is not desirable for the world to simply vanish when the active user count has (temporarily) dropped to zero. *Persistence* means just that: a world that keeps on 'running', even though no users may currently be connected. It requires specific features in the network architecture that can keep working autonomously, without user intervention.

While research on the architectures behind these systems slowed down, the US military joined forces with the entertainment industry. For example, the US Marines developed a special version of DOOM II, called Marine Doom, for tactical training purposes in 1996. At the same time, products were developed that enabled ordinary users to experience a career in the army. Among these,the most prominent example was America's Army. The game, based on the classical genre of first-person-shooters features a life-like gaming world and

puts the user in charge for making tactical decisions. Released in 2002 and updated in 2006, it was originally seen as a recruitment tool, but later on created controversy due to the diminishing line between war and entertainment.

An excellent overview and detailed comparison between the architectures described in this chapter is presented in [Matijasevic 97] and [Macedonia 97]. [Roehl 95] presents a discussion based on the DIS protocol. Several issues associated with large-scale virtual environments are presented in [So 97], as well as information on spatial scoping techniques.

Part I

Network-level requirements

Introduction

In an ideal world, this entire part of the dissertation would be superfluous and any discussion on the topic of resiliency against network deficiencies of networked virtual environment applications would be irrelevant. Unfortunately, current generation networks are in fact quite far from the ideal of a zero-delay, limitless bandwidth-providing and error-free data channel. While the backbone of the Internet, as it exists in its current implementation, is based on optical communication (which, in theory, enables data to travel at near light-speed) the processing equipment and the heterogeneous nature of the network limits the speed at which data can be transmitted from one end of the earth to the other. Typically, these delay figures range from microseconds – on local area networks – to several seconds – in case of multi-hop satellite transmissions. There is an analogy to this situation to be drawn for the availability of bandwidth to individual users. While the typical home user nowadays has a broadband connection at his/her disposal, there is mostly a limit to the amount of data that can be sent in the upstream direction of a network – as seen from the end-user point of view. This presents serious challenges for the development of large-scale virtual environment applications, as the amount of data that is to be transmitted does not grow in a linear fashion with regard to the number of users present in the virtual world.

In this part of the dissertation, we will discuss the two most common elements that pose a limit to the actual performance that can be achieved on current-generation networks: throughput and delay. Besides the study of typical applications, we will also discuss the influence of the network deficiencies on user performance and experience.

Chapter 3

Bandwidth utilization of NVE applications

Networked virtual environments or – more in particular – virtual interactive communities have made a transformation from being mainly unknown to reaching head-line news in just a short period of time. With all publicity regarding the 'Second Life' world, it is obvious that there is some interest among the general public for this type of application. Besides Second Life [Linden Labs 03] however, there have historically been a lot of applications that were based on networked virtual environment technology. As was demonstrated before, military simulation and massively multiplayer on-line games share a lot of the technology that is at their foundations. It is therefore a natural step to take a look at the bandwidth use of these types of applications to get a grasp on some typical usage examples. Unfortunately, game developers and entertainment companies alike, are hesitant to give away any information regarding the underlying technologies that they program into their applications. It is therefore necessary to partially 'reverse engineer' the network traffic generated by these applications to identify the architectural components and to determine the reasons behind the bandwidth distribution. All charts presented in the following sections were created using real-life network traces and manual stream analysis.

3.1 Networked games

In this section, three popular types of games are analyzed to try and determine on the one hand the components that are used in the architecture (such as

servers) and on the other hand the distribution methods for synchronizing state at the end-points.

3.1.1 MMORPG

The age of the popular MMORPGs (massively multiplayer on-line role playing games) began with the introduction of Sony's EverQuest [SOE 97]. This fantasy-genre role playing game is based around the concept of 'quests' that need to be completed by the player in order to achieve skill points and advance the abilities of their character. The game was originally released in 1999, and was therefore not developed with broadband network access in mind. The game, even at time of writing of this text, still features an active community, although a large number of players has currently migrated to more modern counterparts of the original game, including EverQuest II and World of Warcraft[Blizzard].

The gaming world (comprising all 3D meshes, textures, scripting information,...) is updated at the start of each session through a system of patches. Although such a system has a clear negative impact on the start-up time of a session, it also eliminates the need for high-capacity network connections to stream these types of information at run-time. Besides these patches, data that is exchanged between client- and server-side consists mainly of authentication information – a monthly fee is required to be able to join the virtual world- as well as state information.

Figure 3.1 shows a traffic capture chart that demonstrates the typical bandwidth usage of a player who is active in a session (once logged in and the patch cycle completed). It is clear that the game is targeted towards dial-up users, since the maximum amount of data that is transmitted is around 5 kBps. Typical numbers range from 0 to around 1 kBps. This is consistent with the bandwidth capacity of V.90 type modems, which were popular at the time the game was released. The V.90 standards calls for a maximum throughput of 56 kbps which in practice yields around 5 kBps of effective data capacity.

Because of the massive nature of the playing world, it was clearly necessary for the world to be divided into a number of regions, which were called 'zones' in EverQuest. Each player would be situated in a single zone at any one given time, making it possible to limit the data to be sent to each participant. The entire EverQuest world (at launch) consisted of around 400 zones. At startup, players can choose among a number of servers to connect with, with their preference based on the mean ping time (more on this subject in chapter 4). Each of the servers ran a separate copy of the virtual world, and supported

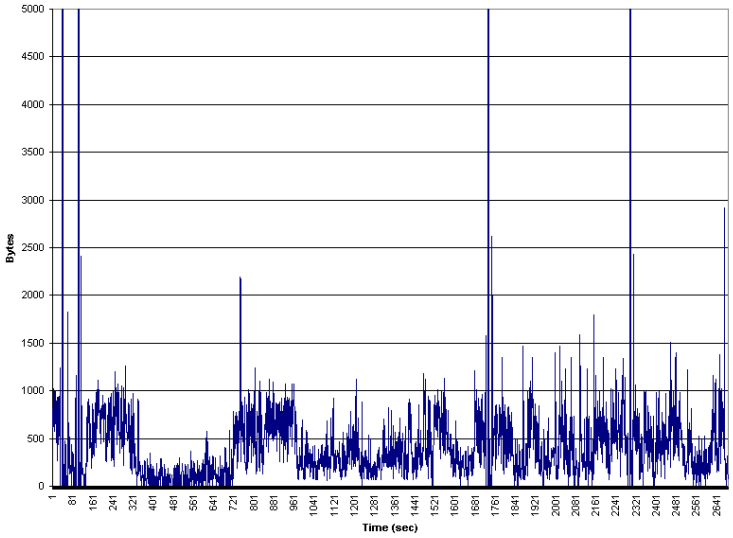
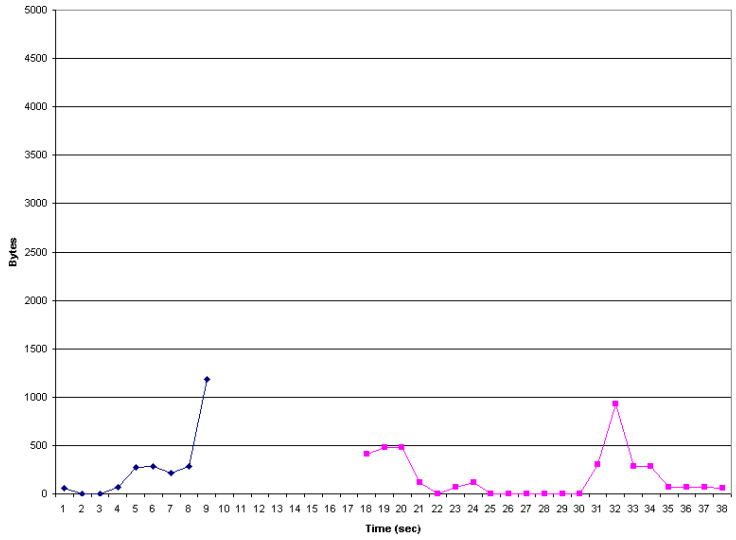


Figure 3.1: Captured network traffic of an Everquest gaming session. Total traffic (up- and downstream) is shown.

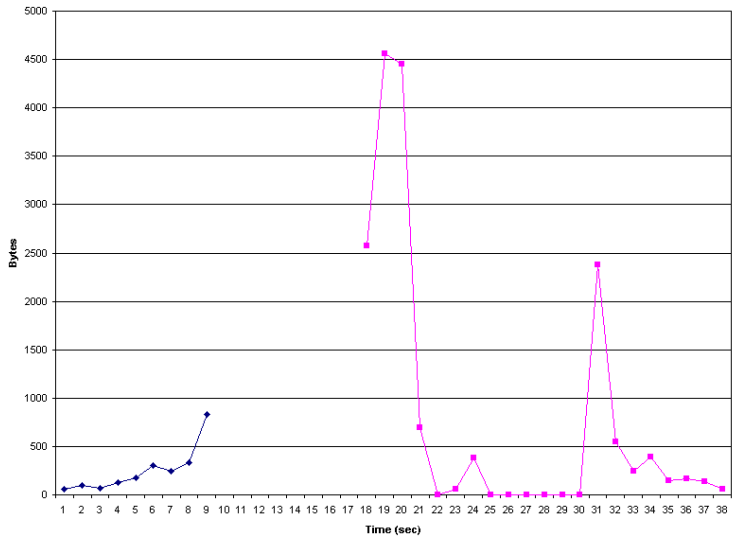
around 3000 simultaneous users. Given an equal distribution of players over the world, this resulted in about 10 players per zone. In practice however, the action was concentrated in a limited number of zones, which leads to the assumption that there are (on average) about 50 to 100 players active in the most frequently visited zones. The transition between zones was cleverly disguised by making use of dark tunnels and/or teleportation rooms, eliminating the need for cross-zone network traffic.

The effect of this zoning optimization is demonstrated in figure 3.2, where two individual network streams can be detected, each associated with a single EverQuest zone. Transitions between zones are not smooth, as can be derived from the lack of network traffic (associated with state updates) for around 10 seconds. Although the techniques in EverQuest are considered to be outdated when compared with modern MMORPGs, it is clear that the game introduced some optimizations that remain viable even for today's alternatives.

EverQuest was not the only game of its kind (although it was one of the more popular examples). Figure 3.3 shows the bandwidth associated with the game "Dark Age of Camelot" [Mythic 96], which was similar in setup. It too features similar requirements, with the bandwidth used averaging around 600

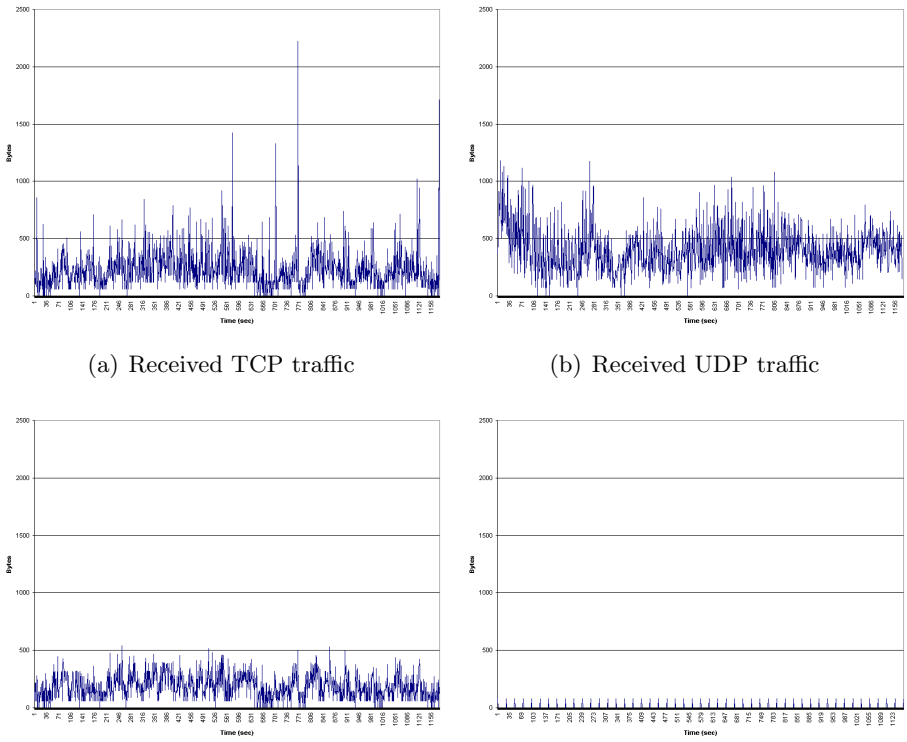


(a) Sent UDP traffic



(b) Received UDP traffic

Figure 3.2: Captured network traffic of an Everquest gaming session. UDP traffic is shown.



(a) Received TCP traffic

(b) Received UDP traffic

(c) Sent TCP traffic

(d) Sent UDP traffic

Figure 3.3: Captured network traffic of a Dark Age of Camelot gaming session. Sent and received TCP and UDP traffic is shown.

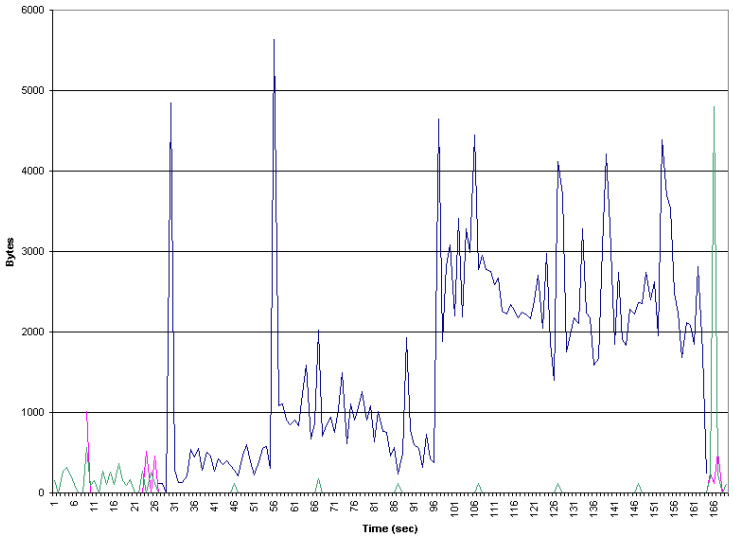
bps, with extremes around 1100 bps. While DAoC and Everquest required the use of a fee-based service to enable the users to connect to the virtual world, games such as Diablo and Warcraft III made it possible for players to connect ad-hoc to a server and create sessions of a shorter duration (even on a LAN). Traffic from these games is depicted in figures 3.4 and 3.5. Overall, these charts are very much similar to the ones associated with EQ and DAoC. They are easily recognizable by the lack of burstiness in traffic and the overall low throughput numbers.

3.1.2 MMOFPS

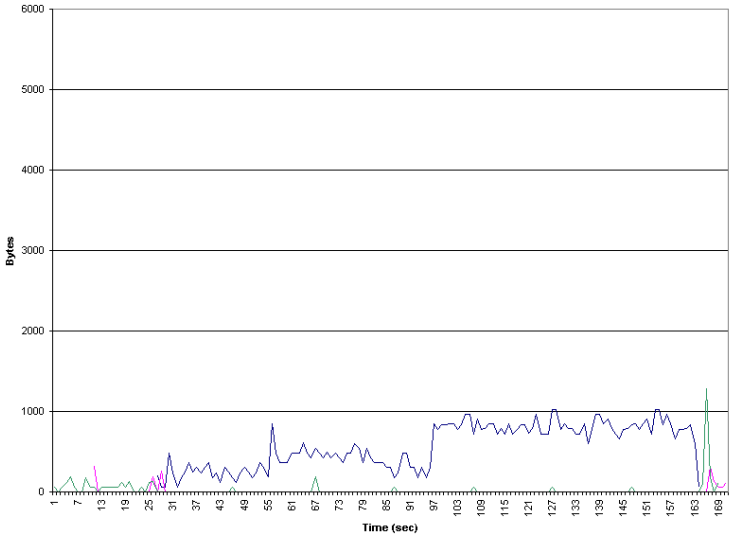
The archetypical First Person Shooter (FPS), which also received major public attention due to the graphical nature of violence, is certainly 'DOOM' by ID software. Released in 1993, it featured network play based on the, at that time, popular IPX/SPX network protocol stack used by the Novell Netware network operating system. Up to 8 players were able to join a session of the game, with all data being broadcast over the LAN. This behavior lead to the eventual banning of the game by system administrators who were concerned with the increased traffic on their networks. While the game has spawned a legacy of similar games (including the Quake, Unreal and Half-Life series), the typical number of users in a single session has remained limited to less than 50. Partially due to the nature of the virtual world that is used (limited in size), the degree of interactivity and dependance on time-critical updates has made the genre difficult to adapt to large scale audiences.

In 2003, Sony launched PlanetSide, one of the first examples of a Massively Multiplayer On-line First Person Shooter). Like its MMORPG counterpart EverQuest, PlanetSide also featured fee-based Internet play for thousands of simultaneous users. Players connect to one of 3 servers (in practice, these are probably clusters of servers, but the information regarding the server setup is never released by game manufacturers/producers), which are running separate copies of the game. While that game could be played over a narrowband (analogue modem) connection, it featured some functionality that required broadband access (mainly real-time sound related features).

When looking at the traffic charts of PlanetSide in figures 3.6(a) and 3.6(b), it can be observed that (over a short period of time) much of the network traffic associated with game status is situated in the region around 5 kBps (similar to EverQuest). When using a broadband connection, there are peaks to around 30 kBps that are related to the sudden switching between different parts of the world. The game featured a concept of zones that was similar to the

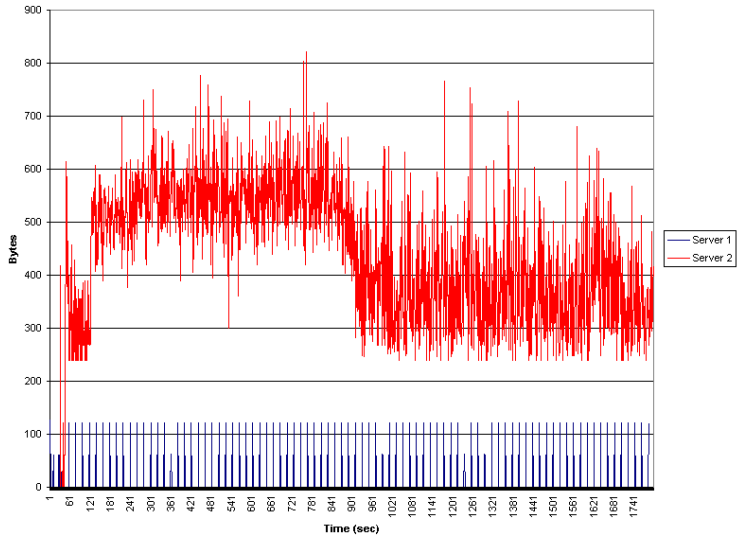


(a) Received TCP traffic

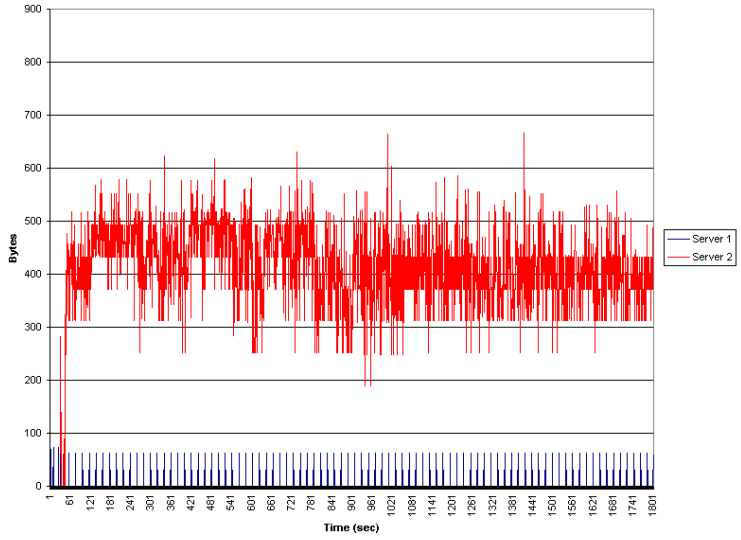


(b) Sent TCP traffic

Figure 3.4: Captured network traffic of a Diablo II gaming session. TCP traffic is shown.

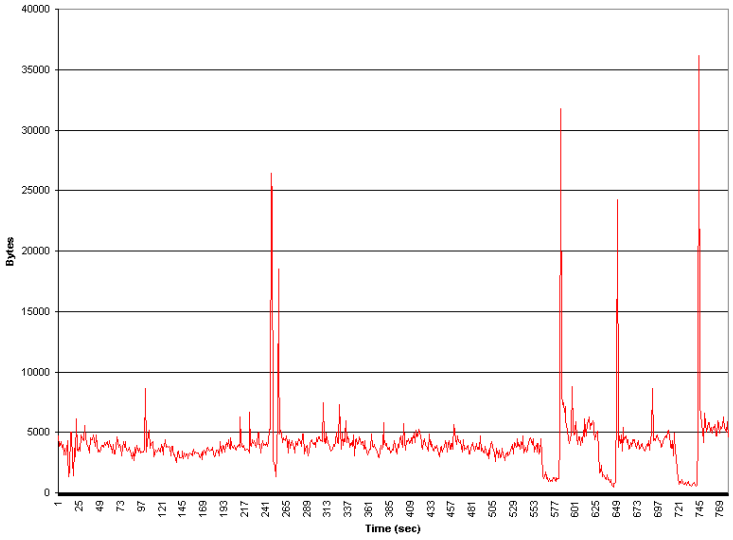


(a) Received TCP traffic

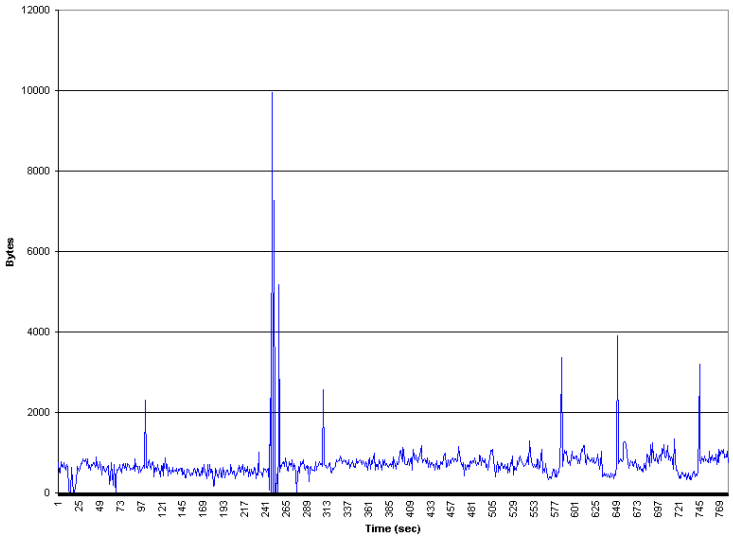


(b) Sent TCP traffic

Figure 3.5: Captured network traffic of a Warcraft III gaming session. TCP traffic is shown.

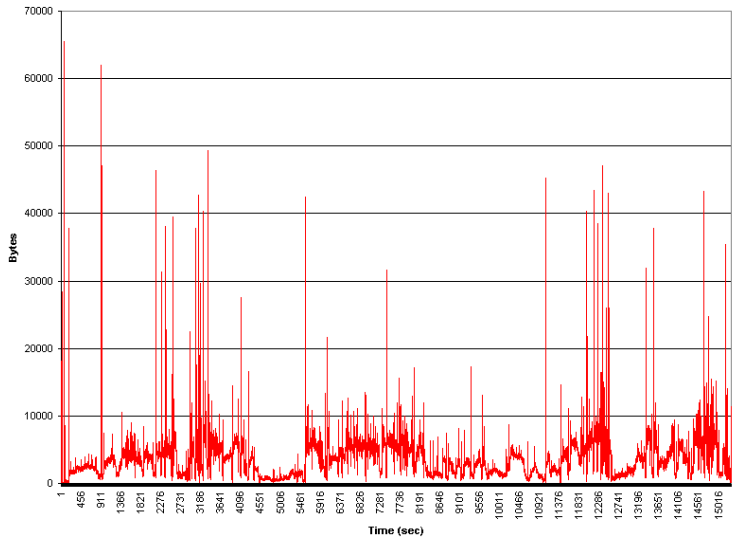


(a) Received traffic

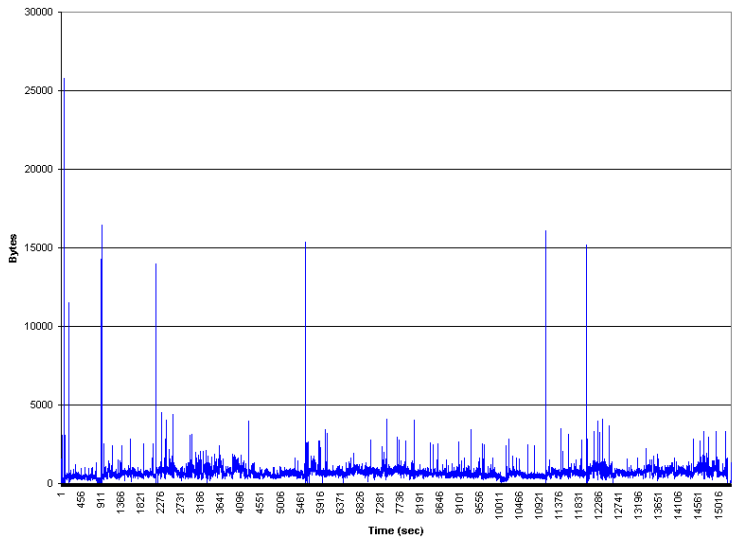


(b) Sent traffic

Figure 3.6: Captured network traffic of a PlanetSide gaming session over approximately 10 minutes.



(a) Received traffic



(b) Sent traffic

Figure 3.7: Captured network traffic of a PlanetSide gaming session over a long period.

one used in EverQuest, however in PlanetSide different zones could only be reached through a teleportation mechanism as they were located on different virtual continents. When looking over a longer period of time, as in figure 3.7, it is clear that some features of the game were best enjoyed using a broadband connection, although not strictly necessary.

3.1.3 Xbox Live

Launched in 2002, the Xbox Live architecture was the original on-line platform for Microsoft's first game console. It enabled users to join smaller sessions of multiplayer games and real-time voice transmission. The service has required a broadband network connection ever since launch. The unique feature of Xbox Live, in contrast to offerings from other vendors and publishers such as Sony and Nintendo, was the facilitation of matchmaking. The community that formed around Xbox Live featured virtual identities for participants (in the form of unique GamerTags). Through a generic 'portal', players could seek for and join sessions of ongoing games. Each game was required to incorporate Microsoft's standard GUI for the on-line part in order to deliver a recognizable platform. Although the service was fee-based, costs were low and free subscriptions were often included with the purchase of the game console, which lead to the enormous popularity of the community. Nowadays, the Xbox Live community has grown to include features such as the Market-Place (game-related downloads) and connectivity to the PC platform as part of Microsoft's Live services platform.

The early Xbox Live architecture is interesting to study because it provides insight into the ways in which a large scale deployment of NVE related applications can take place on existing networks. The first generation of Xbox Live games featured both client/server and peer-to-peer traffic flows and was touted as the next generation of on-line gaming because of the superior integration of in-game voice chat. Traffic was prioritized, which in practice meant that the voice channel was disabled if the throughput on the network dropped. The use of peer-to-peer traffic however resulted in a growing number of users complaining about the inability to initiate sessions, due to firewall filtering and the use of NAT (network address translation) routers.

There are two basic communication scenario's for setting up an Xbox Live game, either through peer-to-peer connections or through a centralized client/server architecture. In the peer-to-peer case, a single peer is chosen as the host – each user can decide on his/her own whether to host a specific game or not. Microsoft's server infrastructure is used solely for authentication pur-

poses and to register the 'state' of the user at each time (i.e. 'playing game abc' or 'waiting for other users to join session'). Once other players decide to join an ongoing game session, the Live infrastructure lists the sessions that are reachable through low-latency links. Most of the traffic is routed through the game host, which means in practice that this machine should have a link with relatively high upstream capacity. This is reflected in the official Xbox Live system requirements, which state that a minimum of 64kbps bandwidth capacity between consoles is required.

In figure 3.8, we show the results of an experiment that was set up using the 'Moto GP' game, a first-generation Xbox Live title. In this setup, we hosted several session for a total duration of over an hour. During this time, several players joined our sessions, and in the chart the total amount of traffic that was sent/received is visible along with their IP address. It is clear from this chart that the matchmaking process is able to filter out the 'closest' hosts quite effectively, as all connected users are located in neighboring countries (even though location information was not visible to end-users in first generation Xbox Live titles). In figure 3.9, traffic for a single session of the game is shown, starting with the authentication procedure (first few seconds) through the matchmaking process (seconds 25 through 199) and finally the session itself (200 untill the end). Note that MotoGP is a fully peer-to-peer game. We should also point out that the voice chat feature was enabled and used in this session. At peak times, total bandwidth averaged about 6 kBps in each direction, clearly outnumbering the available bandwidth on analogue dial-up connections. The bulk of this data is used for transmission of the audio streams, distinguishable by port number. We have also performed test on the influence of the traffic flows on delay, on several types of access networks. Results have shown that especially for ADSL networks, delay is heavily dependent on the saturation of the channel. We also refer to chapter 4 for more details.

3.2 3D Virtual Interactive Communities

Besides networked games, which we studied in the previous paragraphs, virtual interactive community applications are rapidly taking over from 'traditional' messaging communities (chat-rooms) like the –currently little used – IRC network (Internet Relay Chat) or instant messengers like Yahoo Messenger and MSN messenger. Users of these applications like the ability to immerse one's personality in a virtual world where the presence of other people is represented through their avatars. The ability to personalize the avatar and 'shape' the

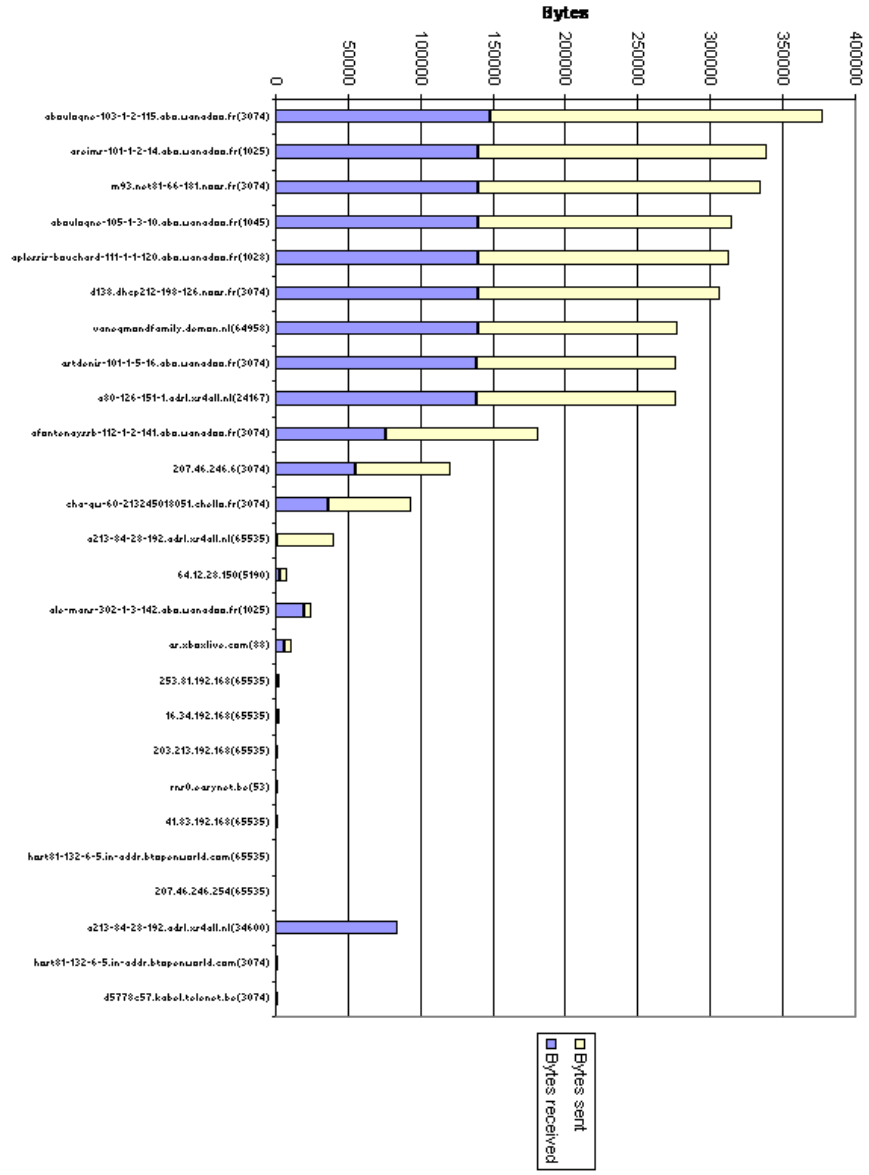
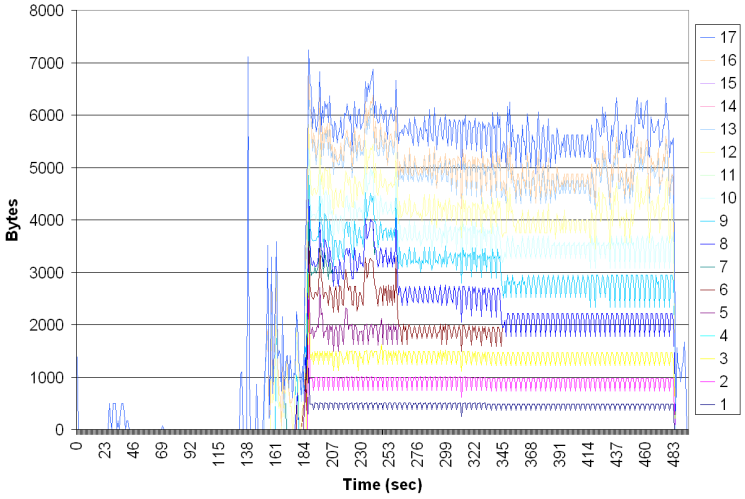
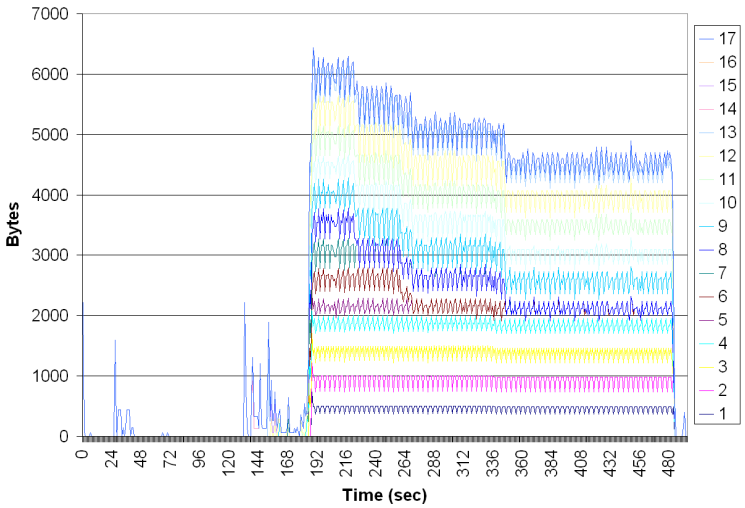


Figure 3.8: Captured network traffic of several hosted Moto GP sessions. Total traffic per IP address is shown.



(a) Downstream traffic



(b) Upstream traffic

Figure 3.9: Captured network traffic of a single Moto GP session. Sent and received traffic per host is shown in a stacked line chart.

virtual world according to individual wishes is something that was not feasible using earlier interactive community technologies (such as forums, multi-user-dungeons etc). In this section, we will closely examine some early and current examples of virtual interactive community applications to create a better understanding of the network issues that they introduce and need to overcome.

3.2.1 ActiveWorlds

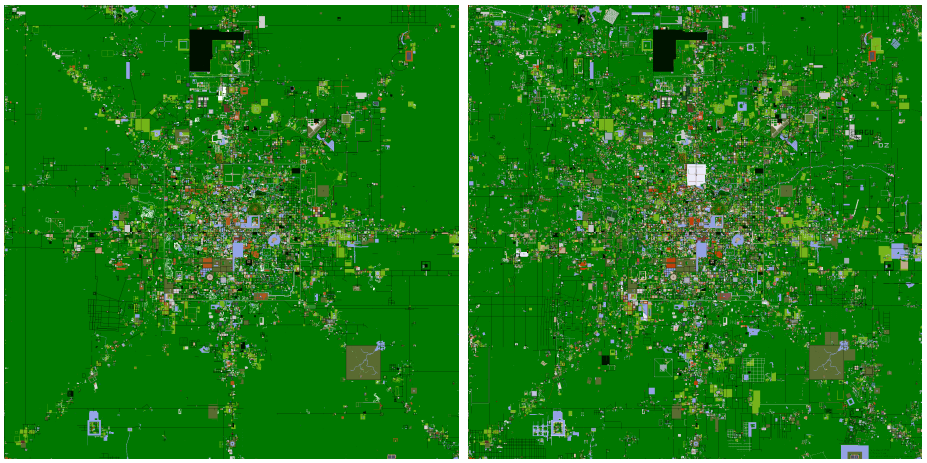
One of the first commercially available 3D virtual interactive community applications is 'ActiveWorlds' by ActiveWorlds Inc. Started in 1995, the community still exists but has been superseded by other, more popular and graphically interesting, alternatives. However, it is still interesting to look at some of the concepts the developers introduced, many of which live on in today's VICs.

ActiveWorld users were, ever since the earliest versions, able to add objects and buildings to the virtual world that enabled personalization of their surroundings. While guest (non-paying) users were able to roam the entire virtual world, customization was a fee-based service, with the price depending on the virtual surface of the area that was to be customized. The client/server based architecture enabled easy management (administration) of the virtual world, as well as the ability to monitor and moderate user actions. The entire system integrated with other WWW services such as linking to web pages and FTP servers. Although the company boasted figures of several thousands of simultaneous users, it is doubtful that these numbers were actually representative. The reasoning behind this is probably that ActiveWorlds was ahead of its time, with users clueless about the added value of 3D and social networking. The fact that many of today's alternatives have incorporated a large number of features from ActiveWorlds is testimony that they were technically valid – although maybe not commercially at that time.

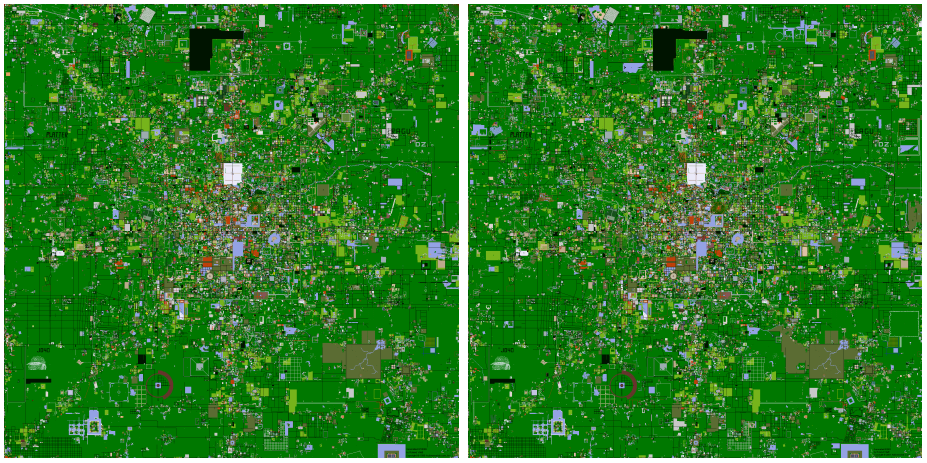
Figure 3.10 shows an overview of the AlphaWorld, as the main community of ActiveWorlds was called. These snapshots or virtual satellite images were taken in 1996, 1998, 1999 and 2001.

3.2.2 Second Life

Second Life by Linden Labs has seen an enormous increase in user numbers since a few large corporations decided the time was ripe for a virtual presence in this community. Some of these included Reuters that opened a virtual press agency, large car manufacturers that provide virtual models of their vehicles (GM and Toyota) and PC equipment manufacturers that sell their products on-line (Dell and Cisco). The entire Second Life community consists of about



(a) 1996 (b) 1998



(c) 1999 (d) 2001

Figure 3.10: Virtual satellite images of the AlphaWorld virtual interactive community over several years.

4 million residents at time of writing (early 2007) and is growing at a dramatic rate. Many of the features that Second Life provides were already available in ActiveWorlds, but Second Life added a more visually pleasing representation, as well as the concept of virtual 'Linden Dollars'. This currency has a real-world exchange rate to major currencies such as euros or USD. Users are able to set up small businesses selling virtual items to gain virtual Linden Dollars and exchange these afterwards for real-world cash.

Technologically, Second Life uses a client/server architecture with incremental object streaming. In practice, this means that users only need to download a 'viewer application' that enables them to participate in the virtual world. All content is streamed in real-time when necessary from an array of servers provided by Linden Labs. Objects can be edited through an in-world creation process (integrated 3D editor) that even allows for collaboration between users working on a single object. Scripting can be added to virtual objects to determine their behaviour. Avatar personalization is made possible through the manipulation of over 150 different parameters with each of them able to take on approximately 100 distinct values. Object data is streamed incrementally, meaning that a course representation is downloaded first, followed by a more detailed version when necessary (for objects nearby or large objects). All of these features result in a system that is highly adaptive but still manageable from a moderation point of view and that takes into account bandwidth limitations at client-side.

The underlying architecture is depicted in figure 3.11, obtained by analyzing the traffic flows generated by the application. At login time, a connection is made with the authentication server at marie.lindenlabs.com. Once login is completed, communication with this server is discontinued. While the session is active, a control channel is maintained with data.agni.lindenlabs.com using UDP. It appears that data sent over this channel is used to switch between simulators and for security reasons (session stealing etc). The world itself is divided into a set of distinct regions, each of them managed by a simulator or content server. At any given time, a user is receiving data from at least one of these simulators (or multiple if moving closely to region boundaries). As said before, content is streamed when requested by the user, and probably triggered by a distance calculation. In practice, the continuous use of streaming introduces visible level-of-detail artifacts, such as the sudden apparition of new objects (pop-up) and obtrusive changes in texture qualities. Obviously, some data exchange between content servers or simulators is necessary, but as we were not able to capture these data flows, no detailed information is given here.

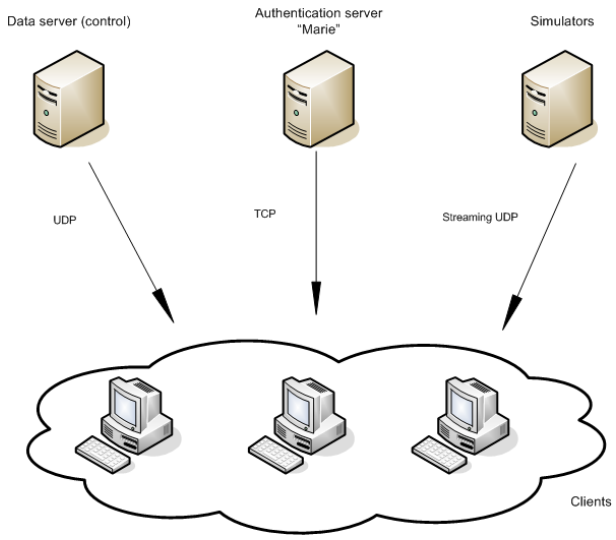


Figure 3.11: Network architecture of Second Life as derived from packet analysis.

Figures 3.12 and 3.13 provide an overview of the network traffic associated with a single session of Second Life. For reasons of clarity, we have indicated all user activity on the chart. It can clearly be seen that different servers are contacted when various activities are performed. As long as the user remains in a single position in the virtual world, nearly all data is streamed from a single simulator. After approximately 30 minutes, the user starts exploring the virtual world through the fly-over mechanism built into the viewer application. It is then that a rapidly changing number of servers is contacted for their content. When interactivity between users is important (as in the 'playing a game of sumo' in the scenario, it can be observed that a dedicated server is used to perform the calculations associated with the simulation. It should also be obvious that Second Life is a broadband-only application, given the higher throughput requirements when compared to earlier examples in this chapter.

3.2.3 There.com

The virtual world of There.com [There] is comparable to Second Life, although at network level it uses about 600 'connections' to various servers, both using UDP and TCP sockets. The major addition (at least in the first versions)

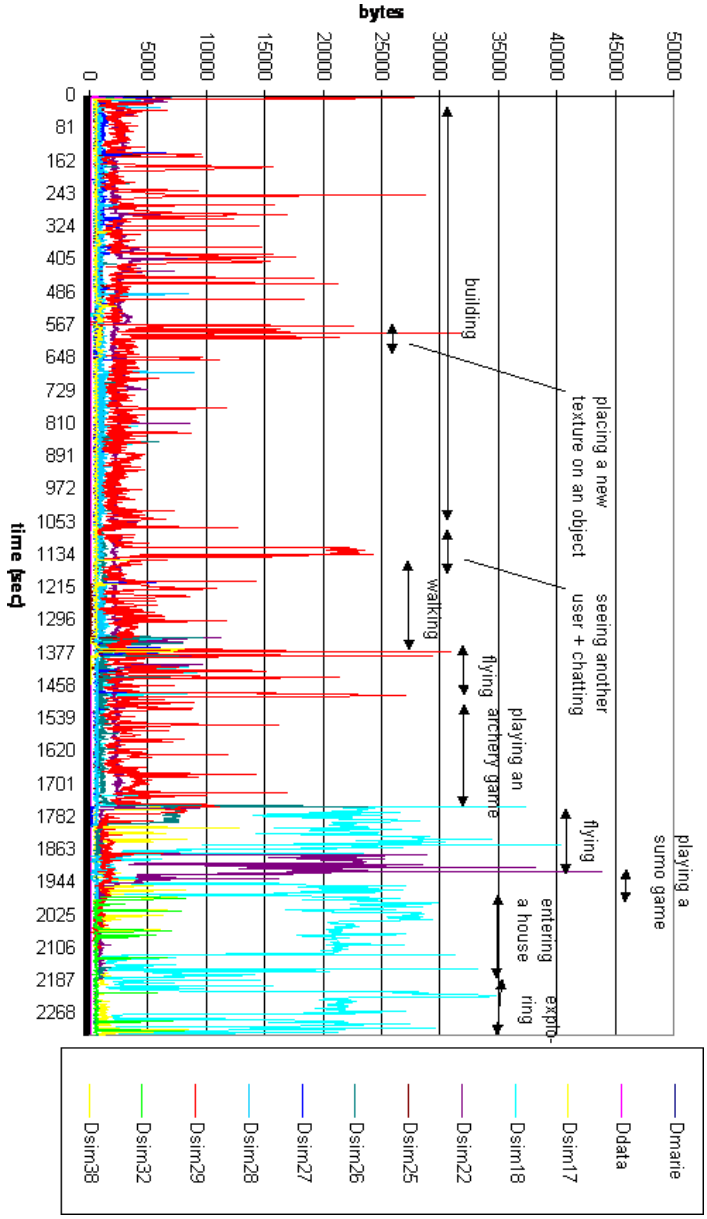


Figure 3.12: Captured network traffic of a Second Life session. Downstream traffic is shown along with server ID.

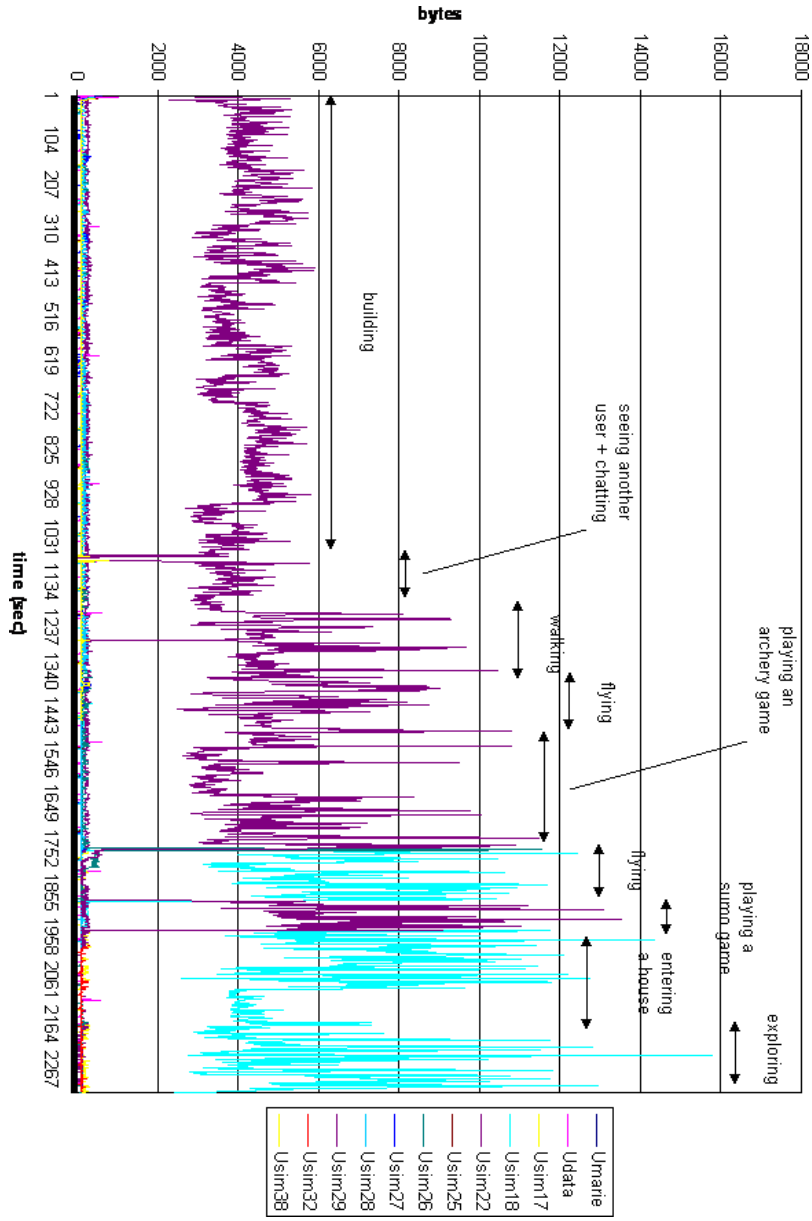


Figure 3.13: Captured network traffic of a Second Life session. Upstream traffic is shown along with server ID.

of There.com is the integration of voice chat for communication. Figure 3.14 shows the bandwidth usage of a There.com session. Actions performed during this session were walking, flying and teleporting. It is obvious that the addition of sound multiplies the amount of bandwidth needed by a factor of at least five. There.com also uses the concept of 3D audio, although this is not reflected in the network traffic, probably meaning that there is only a single quality audio sent by each participant.

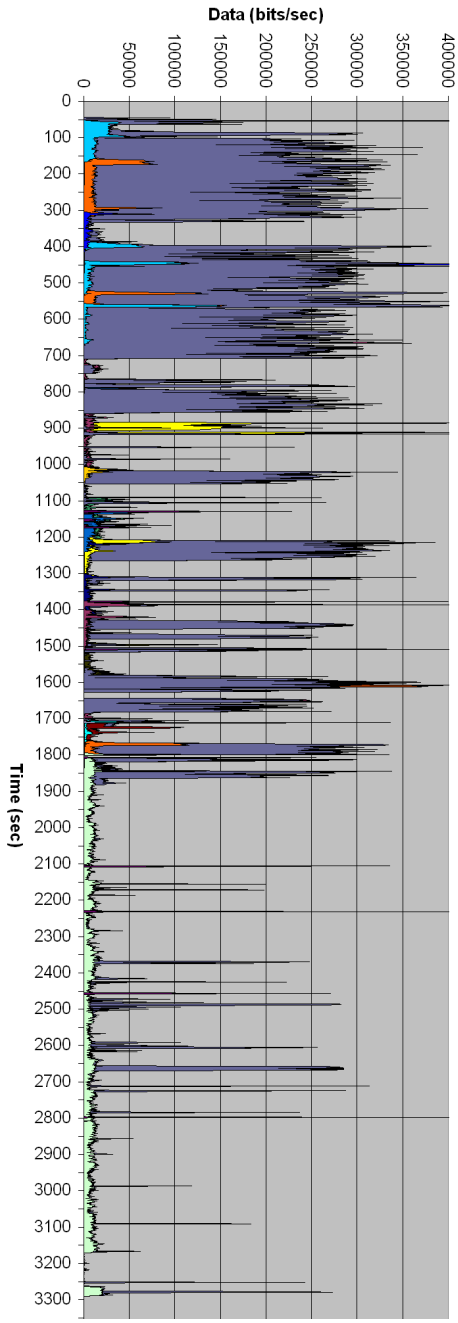


Figure 3.14: Captured network traffic of a There.com session. Downstream traffic is shown.

Chapter 4

Impact of delay and jitter

Besides the limitation in available bandwidth, there are other problems associated with data transmission on a computer network. The deficiency that is most relevant to the NVE-like applications is the presence of delay in transmission. As was stated in the introduction of this chapter, the time it takes a packet to go from the workstation of one end-user to the other - *the network delay* - is clearly not negligible. To make things worse, this transmission delay is not constant, introducing yet another factor to be reckoned with: *jitter*.

Typical delay values on a local area network are situated in the microsecond range. However, for a wide area network that relies on satellite communications, typical delay values can increase to several seconds (worst case scenario). For a typical Internet user however, the practical range of delay values encountered is situated between 5 and 200 ms. Although the total amount of delay is accumulated along the entire path from sender to receiver, the access network is the first step in which a substantial latency factor is introduced. In [Jehaes 03], a comparison is made between several types of access network. It is concluded that for a dial-up connection, average round-trip-time values for packet sizes ranging between 100 and 1600 bytes start at 150ms up to a maximum of about 450ms. Considering an ADSL line, figures range between 15 ms and 90ms (delay values increase a linear fashion with regards to packet size). On cable networks, delay values are (on average) relatively constant at 75ms, regardless of packet size.

It is vital to get a clear idea of the influence of precisely these factors on the 'performance' of a typical end-user. Because of their increasing popularity, we have opted to conduct the actual test using a popular First-Person

Shooter Game. It is often mentioned by gamers that their performance can be attributed to the presence of delay on a network, however little work has been done to actually quantify the effect that can be measured objectively. Furthermore, a more subjective analysis - inquiring about user experience - is even more lacking.

We can summarize the issues to be determined or quantified using the following three questions:

- Can a player effectively determine whether his/her connection is influenced by lag without consulting diagnostic tools, i.e. purely based on his/her perceived game quality and/or performance?
- Is there a bound below which the influence of delay and jitter on the players' performance is minimal or even non-detectable?
- Will small amounts of delay and jitter influence the score on modern first person shooter games that were developed for use over the Internet?

4.1 General description of chosen test case parameters

Relatively modern games that are deployed on wide area networks, such as the Internet can be expected to have better performance than older games when faced with adverse network conditions. Because of this, the game that was chosen to study in this context was Unreal Tournament 2003 [Epic Games 03]. This game was considered to be one of the most popular first person shooter (FPS) games that was both playable on a Local Area Network as well as on the Internet at the time the experiment was conducted (early 2004). This game genre is characterized by its high level of interactivity, which makes it especially suitable for determining the actual influence of network degradation on performance factors. In the first person shooter type of game, players run around in a virtual environment and attempt to hit other players with various kinds of weapons. The objective of the game is to kill as many of the other players as possible, while avoiding getting killed oneself. When a player is killed, he/she is returned to the environment after a short delay, called the respawn time. During typical gameplay in a FPS, multiple hits are needed for a player to be killed. Using the InstaGib modification of the original game, a single hit suffices to achieve the same effect. This modification also facilitates recording each player's activity, as only kills (or frags) can be logged server-side. The type of game used in the experiments was so-called 'deathmatch',

		Scenario																				
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	
PC	1	1	1	1	1					1		1	1	1	1					1		10
	2	1	1	1					1	1		1	1	1					1	1		10
	3	1	1		1			1			1	1	1		1			1			1	10
	4	1	1					1	1	1		1	1					1	1	1		10
	5	1		1	1		1			1		1		1	1		1			1		10
	6	1		1			1		1		1	1		1			1		1		1	10
	7		1		1	1		1		1			1		1	1		1		1		10
	8		1			1		1	1	1			1			1		1	1	1		10
	9			1	1	1	1				1			1	1	1	1				1	10
	10			1		1	1		1		1			1		1	1		1		1	10
	11				1	1	1	1			1				1	1	1	1			1	10
	12				1	1	1	1			1					1	1	1	1		1	10
	13	1	1	1				1	1		1			1		1	1		1			10
	14				1	1	1			1		1	1		1			1		1	1	10
		7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	
	Delay	40	40	40	80	100	60	60	80	100	60	100	100	80	80	10	100	100	20	80	60	
	Jitter	20	35	5	50	80	35	50	20	5	5	65	20	35	65	95	35	50	5	5	20	
	Map	1	2	3	4	5	2	4	1	5	3	1	3	4	2	5	3	2	4	1	5	

Table 4.1: PCs affected by delay and jitter. PCs 2 and 14 were not used.

which means that players engage each other in one-to-one combat, instead of playing in teams.

The participating players recorded their perceptions after every session while playing the game. During almost every session, the network conditions of about half of the participants were degraded with respect to the other players. In the following sections, these players will be designated as being 'impaired'.

Our original experiment was set up for 14 players – with varying experience in playing first-person shooter games – to participate, each using an identical Pentium IV 2.6 GHz computer with 512 MB RAM. A dedicated Unreal Tournament server was installed on an additional machine. All computers on this dedicated LAN were connected through a managed 100Mb switch. To simulate delay and jitter on the network connection, a router was placed between the switch and the dedicated server machine. On this router, the software NistNet [NIST a] was used to introduce delay and jitter on specific network streams. Finally, a machine was connected to the switch that captured the traffic sent to and from the server using port mirroring on the switch. The complete network layout is depicted in figure 4.1. The dedicated server was configured using the default settings, the most important being the server tick rate, which was set at 25, and the InstaGib mutator. The server tick rate represents the rate at which the dedicated server recalculates the state of the entire world (i.e. player positions, object properties,...) and distributes this state to the connected clients. In case of a tick rate of 25, the server updates the internal state 25 times per second, which is consistent with a standard frame rate for video playback. This way, a delay below 40ms should have little influence.

To measure the effect of delay and jitter, it is necessary to simulate various

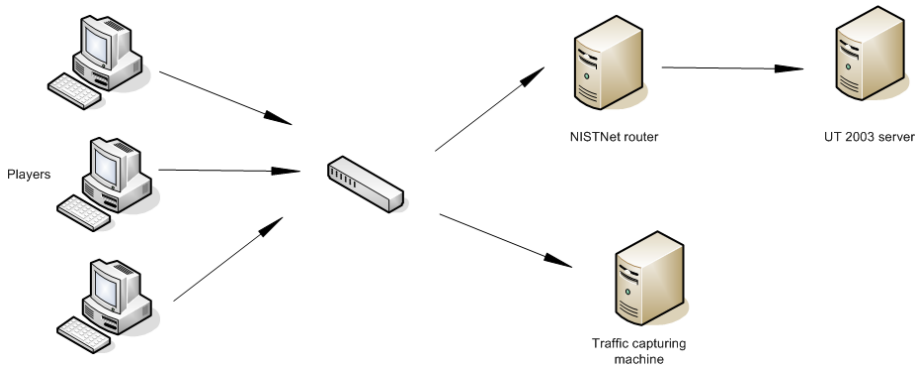


Figure 4.1: Unreal Tournament network setup.

different network conditions. In this case, these conditions range from a negligible round-trip delay and jitter of 20 ms +/- 5 ms to a maximum of 100 ms +/- 95 ms. Figures were derived from related work already described at the start of this chapter. A total of 20 different settings of delay and jitter were selected, each called a 'scenario'. NistNet was configured to split the delay and jitter equally over upstream and downstream traffic towards a particular (impaired) user. Every scenario was tested in a session that lasted for 7 minutes. During every scenario, one set of players experienced the introduced lag and jitter, while the other players were unaffected, i.e. their settings were set to the minimum amounts (20ms +/-5 ms). We opted to subject the non-impaired group of players to these minimal amounts in order to simulate the minimal delay that is always present on a typical access network. The set of affected players changed after every configuration. Table 4.1 shows the delay and jitter used in each configuration, and the players that were subjected to this delay and jitter (marked by the number 1). Note that in scenario 18 all players experience the minimum amount of delay and jitter, so the marking with 1 has no meaning. Also, the ID of the Unreal Tournament map (or world) used in each setup is depicted. Originally, 14 players were scheduled to participate in the test. Unfortunately, 2 players were unable to attend, and as a result, PCs 2 and 14 were not used. It is unfortunate that PC2 was not used, because in the tests on some sessions 5 or 7 players are impaired, instead of consistently having 6 impaired players. This setback slightly increased the complexity of processing the results but does not influence the results and conclusions obtained from the tests.

In order to obtain answers to the initial questions that are stated at the

PC \ Scenario	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	9	9	9	6	8	7	8	8	8	9	7	7	9	8	8	7	7	8	8	9
3	6	6	8	3	5	7	5	7	5	3	7	7	7	6	5	3	7	9	4	6
4	5	6	6	7	8	7	4	5	6	8	3	4	8	8	6	7	5	6	5	6
5	7	6	3	5	7	3	5	7	3	3	3	5	4	5	7	2	6	8	5	8
6	6	7	4	6	4	4	7	7	3	2	3	4	4	6	5	2	6	8	4	8
7	8	8	8	6	6	7	7	7	4	7	8	3	8	4	3	7	5	7	3	7
8	9	8	7	9	7	7	8	6	8	6	9	6	9	9	7	6	5	6	7	9
9	9	9	8	8	6	8	9	8	9	7	9	9	6	7	5	6	5	9	8	6
10	9	8	6	9	8	8	9	8	9	6	9	8	8	8	8	6	8	8	9	6
11	8	7	9	8	7	9	9	8	9	6	8	7	9	9	9	7	8	9	8	9
12	9	9	8	9	9	8	9	9	7	7	9	6	6	7	9	2	6	9	9	9
13	9	9	9	8	7	8	8	8	9	6	8	8	7	9	5	4	9	9	7	8

Table 4.2: Results of question 1: 'Rate the quality of the network'. Colored background indicates impairment.

beginning of this chapter, the impact of delay and jitter was measured using two approaches. First of all, the number of times a player effectively killed another player ('kills'), and the times he was killed himself ('killed') were logged at the server. This provides us with an objective measurement of the quality of the game play, and enables comparison between different sessions. Second, after every session, the participants had to fill out a short questionnaire regarding the network quality they experienced. The following questions had to be answered after every session:

1. Rate the quality of the network (0(=worst)-1-2-...-8-9-10(=best))
2. How much did the quality of the network influence your gameplay? (0(=not at all)0-1-2-...-8-9-10(=very much))
3. Do you think the quality of the network influenced your score? (Yes/No)
4. Remarks on this scenario

For questions 1 and 2, the participants were not allowed to select the values 0 and 10 (as described in ITU-T Recommendation P.800.1). The questionnaire also included some general information, such as name, age, sex and amount of experience with FPS games(none-little-much-very much).

The results of the objective and subjective measurements are detailed in the next section.

4.2 Objective observations and subjective ratings

This section presents the objective and subjective measurements of the experiment for every player during every scenario. Table 4.5 shows the number

		Scenario																			
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
PC	1	1	1	1	5	1	2	3	2	2	1	3	2	1	1	2	2	3	2	2	2
	3	4	8	5	9	7	4	7	3	5	8	3	7	2	2	2	6	2	9	6	4
	4	8	2	3	1	1	3	7	3	1	1	8	5	1	1	1	1	3	1	3	2
	5	3	4	8	5	3	8	4	3	7	7	8	5	5	4	4	7	3	2	6	2
	6	4	5	6	4	8	7	5	4	7	8	8	8	6	5	7	9	4	3	5	3
	7	7	7	7	6	7	6	6	6	8	6	6	8	6	8	9	6	7	6	8	8
	8	1	2	2	1	2	3	2	3	2	3	1	3	1	1	3	3	3	3	3	1
	9	7	7	8	7	8	7	9	8	8	7	9	9	8	8	8	7	8	9	8	6
	10	1	1	5	1	3	2	1	2	1	3	1	2	2	2	2	5	2	2	1	3
	11	2	2	1	2	4	2	2	2	2	3	2	2	1	1	1	3	2	1	2	1
	12	1	1	2	1	1	1	1	1	4	3	2	3	6	1	1	8	4	1	1	1
	13	3	2	1	2	4	1	2	3	1	5	2	1	3	1	5	6	1	1	2	1

Table 4.3: Results of question 2: 'How much did the quality of the network influence your gameplay ?'. Colored background indicates impairment.

		Scenario																			
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
PC	1	0	0	0	1	0	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0
	3	1	1	0	1	1	0	1	0	0	1	0	1	0	0	0	1	0	1	1	0
	4	1	0	1	1	0	1	1	1	1	0	1	1	0	0	0	0	1	1	1	1
	5	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	6	1	1	1	0	1	1	0	0	1	1	1	1	1	0	1	1	0	0	1	0
	7	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	8	0	1	0	0	0	1	0	1	1	1	0	1	0	0	0	1	1	1	1	0
	9	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	10	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1
	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	12	0	0	1	0	0	0	0	0	1	1	1	1	0	1	1	1	1	0	1	0
	13	0	0	1	0	1	0	0	0	0	1	0	0	1	0	1	1	0	0	1	0

Table 4.4: Results of question 3: 'Do you think the quality of the network influenced your score ?'. Colored background indicates impairment.

		Scenario																			
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
PC	1	18	16	29	21	19	12	23	13	10	24	12	18	26	9	10	20	10	20	22	20
	3	12	20	26	22	16	11	18	20	11	15	5	25	31	10	32	21	10	30	17	21
	4	19	21	30	34	31	26	30	19	18	45	22	29	50	22	25	39	17	33	27	28
	5	19	22	30	40	25	12	38	28	18	34	25	36	33	20	31	31	19	42	24	37
	6	14	11	17	23	11	5	25	14	20	19	10	22	23	7	20	27	12	33	17	16
	7	35	27	39	36	32	31	63	49	22	60	41	35	58	27	22	56	26	57	28	40
	8	26	16	27	56	26	19	41	28	29	38	24	29	51	24	35	40	26	39	25	28
	9	35	24	27	45	26	25	44	33	39	40	40	50	40	30	20	33	25	58	40	21
	10	22	16	18	31	15	12	30	11	25	34	26	32	22	12	18	16	14	30	27	14
	11	22	16	24	20	10	12	26	16	12	23	19	28	26	14	20	15	8	26	17	16
	12	18	14	33	40	15	12	27	19	26	28	19	29	20	8	24	16	8	29	23	22
	13	34	28	56	57	34	33	43	31	49	50	34	49	57	39	30	30	37	56	35	49

Table 4.5: Kills per player in each scenario. Colored background indicates impairment.

PC \ Scenario	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	27	20	36	45	24	23	39	32	29	35	28	45	47	23	27	37	21	52	30	28
3	24	18	38	39	23	17	48	22	27	49	32	38	48	17	27	36	17	52	27	28
4	36	28	40	44	24	23	49	34	34	46	27	38	48	30	28	37	29	53	32	28
5	23	23	33	35	21	20	24	26	22	35	20	30	41	17	16	29	18	32	27	27
6	20	19	28	40	23	19	38	29	25	47	21	37	35	21	27	33	18	43	30	29
7	19	16	24	28	16	10	15	10	18	23	15	32	26	19	20	20	16	31	22	19
8	21	20	28	31	19	18	31	22	23	25	25	22	29	16	22	13	16	34	26	25
9	22	16	22	28	24	19	30	18	14	24	20	24	30	19	28	23	15	32	15	26
10	22	16	30	33	27	15	24	22	21	28	21	23	27	18	23	31	17	24	24	28
11	19	17	26	42	22	14	35	21	21	30	17	24	33	15	23	26	15	37	21	28
12	24	20	33	34	18	17	47	24	29	41	28	41	43	15	22	32	19	33	28	28
13	17	18	18	26	19	15	28	21	16	27	23	28	30	12	24	27	11	30	20	18

Table 4.6: Number of times a player was killed in each scenario. Colored background indicates impairment.

of kills each player made during every scenario of the experiment. Table 4.6 depicts the number of times each player was killed during each scenario. These objective results were recorded by the dedicated server. The next set of tables indicate the results of the questionnaire that was filled out by the participants. Table 4.2 shows the answers to question 1 and gives a subjective score of the quality of the network. Table 4.3 displays the results of question 2 and presents a subjective measurement of the quality of play regarding the network conditions. Finally, table 4.4 shows the answers to question 3: '1' meaning the player thinks the network influenced his score and '0' otherwise. For every measurement, a cell with colored background indicates that a participant was impaired during that session. Again for scenario 18, the colored background has no meaning. For a definition of impairment, we refer to section 4.1.

4.3 Traffic capture

One PC in the network setup was used to capture the data sent to and from the server. To avoid any possible influence of the capture process on the performance of the server or router PC, we used port mirroring to capture the traffic on a separate computer. Game traffic was captured using a custom-built program, which is based on the WinPcap library. The bandwidth usage of network traffic sent to and from the clients is shown in figure 4.2. The individual sessions can easily be distinguished in this graph because of the gaps of activity between the sessions. Because port mirroring was only enabled during the first session, some traffic of this session is missing. There was also a 22-minute break halfway through the experiment. Some observations can be made from this data. First, the amount of traffic sent by the clients remains fairly constant over all the sessions. On the other hand, the amount of traffic

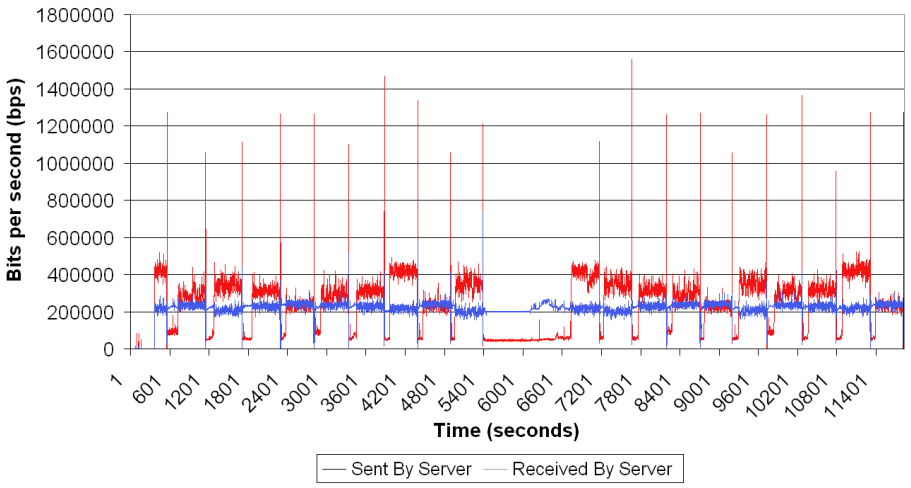


Figure 4.2: Captured network traffic. Order of the maps is: 1-2-3-4-5-2-4-1-5-3/1-3-4-2-5-3-2-4-1-5.

sent to the clients by the server varies depending on the map that was used. Map 1, which is a large exterior environment, generates most traffic. Map 5, an interior map, generates the least amount of traffic. This can be explained by taking into account the line of sight in those maps. Exterior environments have a larger line of sight, and therefore more objects (including opponents) are visible compared to interior maps with small rooms. Apparently, the server only transmits location updates of visible opponents, which reduces the amount of traffic in confined environments.

4.4 Measurement analysis

4.4.1 Objective observations

In this section, objective measurements are analyzed in order to investigate whether a degraded network quality, i.e. delay and jitter in this context, has an impact on the performance of the players. For that purpose, an objective measure of performance will be defined first. Next, the performance of the players in scenarios where they do not experience impairment will be compared to their performance in scenarios where they do experience impairment. And finally, it will be verified whether the performance of players that are not

directly subjected to an impairment condition, is affected by the impairment experienced by other players in the same session.

As explained in section 4.2, the objective measurements consist of statistics about the number of times each player has been killed and/or has killed other players during each of the 20 scenarios (see tables 4.5 and 4.6). In order to analyze these objective results, the difference between the number of kills and the number of times being killed for each player p and in each scenario c, is used as a measure of score:

$$S(p,c) = \#\text{kills}(p,c) - \#\text{killed}(p,c) \qquad \forall p \in P, \forall c \in C$$

with P the set of 12 players and C the set of 20 scenarios. A positive score indicates that a player has killed more other players than that he was killed himself. A negative score on the other hand signifies the player has been killed more frequently than that he has killed other players. As such, the score as defined above is an objective measure of the performance of each player in each scenario.

In figure 4.3 the mean score for each player over all scenarios where he is not affected by impairment and the mean score over all scenarios where he is subjected to impairment, is shown. The mean score over all 20 scenarios is also shown. The vertical error bars indicate the standard deviation of the latter. This figure shows the trend that the mean score of the impaired games is consistently lower than the mean score of the unimpaired games. This is an indication that network impairment, as defined in section 4.1, does have a negative influence on the affected players' performance.

The question can be asked whether players that are not directly subjected to impairment, are hampered when other players in the game session are subjected to high impairment conditions. In other words: can, at high impairment levels, all players be considered as being hampered, no matter if they are directly subjected to the impairment or not. In order to get a notion of that, all 20 scenarios are divided in 2 categories: hypothetical impaired and hypothetical unimpaired scenarios, where the hypothetical impaired scenarios are the 13 scenarios where the delay and jitter values are at least 60 ms and 50 ms, respectively. The hypothetical unimpaired scenarios are the other 7 scenarios. Now for each player, his mean score over the 13 hypothetical impaired scenarios, irrespective of really being exposed to the impairment or not, is calculated, as well as his mean score over the 7 hypothetical unimpaired scenarios. If in general these two mean scores do not show significant differences compared to the differences between real impaired and real unimpaired scenarios as presented in figure 4.3, one can not state that all players in hypothetical

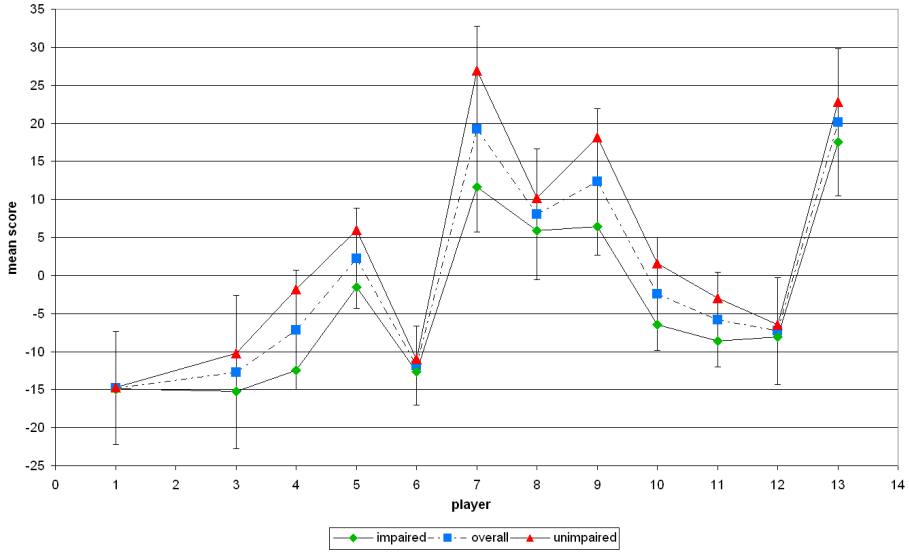


Figure 4.3: Mean score for all players: over all 20 scenarios, only over the impaired and only over the unimpaired scenarios, respectively.

impaired scenarios (irrespective of directly being exposed to the impairment) can be considered as being impaired. In figure 4.4, diamonds and triangles, connected with solid lines represent the mean scores of the impaired and the non-impaired scenarios of all players as already shown in figure 4.3. The circles and squares, connected with dashed lines represent the mean scores for all players for the hypothetical impaired and unimpaired scenarios. It is clear from this figure that in general the difference between the hypothetical impaired and unimpaired scenarios is inferior to the difference between the real impaired and unimpaired scenarios. As such, players that are not directly subjected to impairment, are not hampered when other players in the game session are subjected to high impairment conditions.

4.4.2 User perspective

In order to evaluate the subjective experience of network impairment, the results of the questionnaire as detailed in section 4.2 are used. First, it will be verified whether players that are affected by impairment conditions, do feel hampered. Based on the findings from this first step, the question is asked whether they can make a distinction between mild and severe impairment

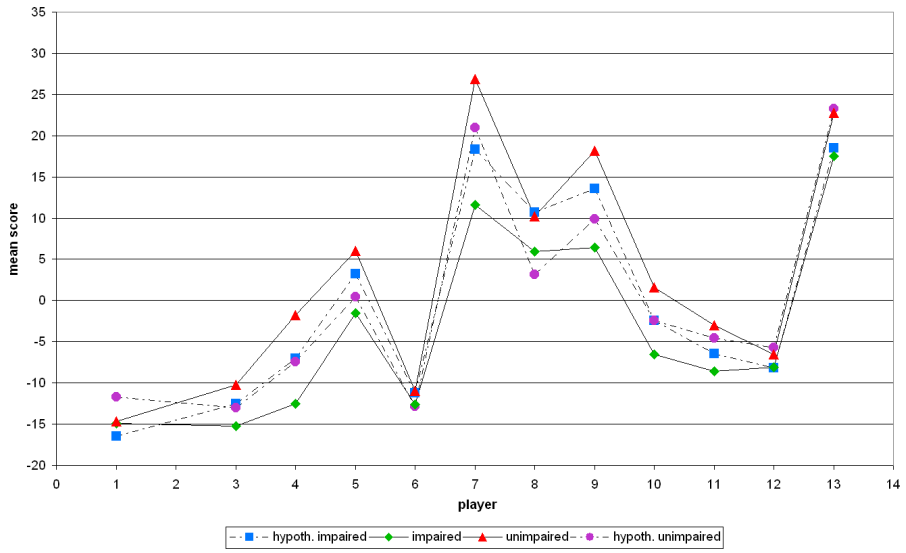


Figure 4.4: Mean scores for all players for the real impaired and unimpaired, and the hypothetical impaired and unimpaired scenarios, respectively.

conditions. Finally, an attempt is made to determine impairment bounds above which players experience the network quality as inferior.

Overall rating

The answers given to the first question 'Rate the quality of the network', summarized in table 4.2, are used to see whether in general players experience impairment as degrading network quality. A distinction is made between players that are exposed to impairment (and are marked with 1 in table 4.1) and players that are not subjected to impairment. For each scenario, the mean rating of the network quality is calculated for the non-impaired and for the impaired players, respectively. This is shown in figure 4.5. The mean rating of all players is also given, as well as the standard deviation. The figure indicates that overall, players that are subjected to increased delay and jitter rate the network quality poorer than the other players. In other words: increased delay and jitter are experienced as degrading network quality.

Note that in scenario 18 a distinction is made between impaired and unimpaired players, following table 4.1, although for this scenario all players should be considered as unimpaired. As explained in section 4.1 a delay of 20 ms and

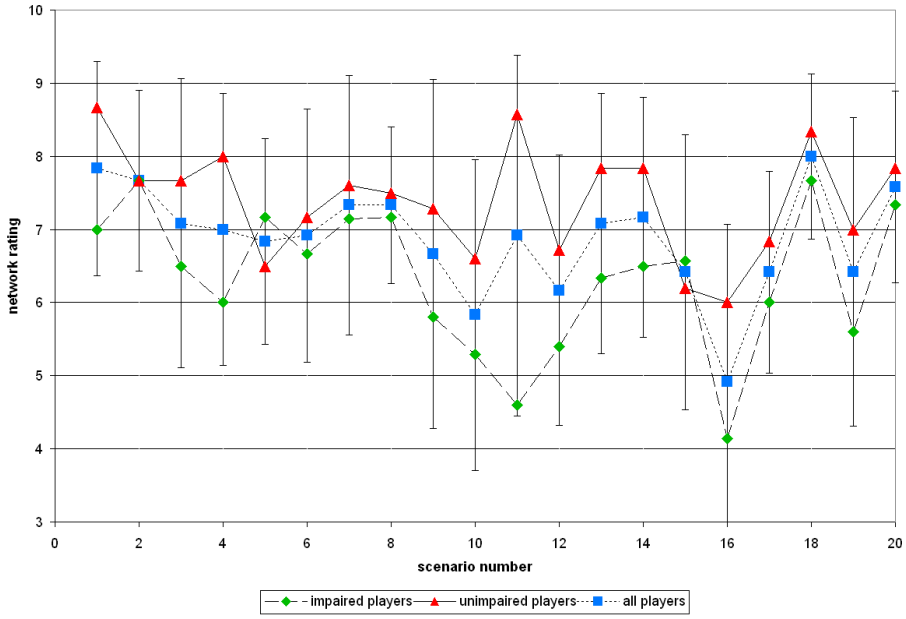


Figure 4.5: Mean rating for all unimpaired, all impaired and all 20 players, respectively, as a function of scenario number.

a jitter of +/-5 ms is considered a non-impaired scenario.

Besides this general tendency, the above figure provides more interesting information when looking at scenarios 1, 5, 11 and 15. For that purpose, the players are divided in two categories: the 'optimists' and the 'complainers'. The selection of both is done based on their network rating behavior and is explained in the following. For all 20 scenarios, the mean rate given by all 12 players is determined, irrespective of being impaired or not. Next, for all players it is checked in each scenario whether they rate this scenario better or worse than this mean rate. The more scenarios a player rates better, the higher his degree of 'optimism'. Finally, the mean degree of 'optimism' over all 12 players is determined. Players with a degree of optimism, lower than this overall mean degree, are labelled 'complainers', the players with a degree higher than this overall mean degree, are labelled 'optimists'. This method classifies players 3 to 7 as 'complainers' and all other 7 players as 'optimists'. Note that applying the above methodology to only the impaired scenarios, leads to the same classification of players. Table 4.7 summarizes for scenarios

		Players														
		1	3	4	5	6	7	8	9	10	11	12	13	Delay	Jitter	Map
Scenario	1	X	X	X	X	X							X	40	20	1
	5						X	X	X	X	X	X		100	80	5
	11	X	X	X	X	X								100	65	1
	15						X	X	X	X	X	X	X	100	95	5
	Optimism	18	6	6	3	1	8	15	14	17	19	15	17			
			Complainers													

Table 4.7: Summary of impairment settings for scenarios 1, 5, 11 and 15; the degree of optimism is also given.

1, 5, 11 and 15 which players are subjected to impairment, indicated with an X on a colored background, as well as the corresponding impairment settings. Their degree of optimism is also given.

As can be seen in figure 4.5, for scenarios 5 and 15, respectively, the players that are exposed to impairment rate the network quality slightly better than the players that are not exposed to impairment. Examining both scenarios more in detail explains what happens here. On the one hand both scenarios are quasi-identical: severe impairment conditions, same map, and identical sets of impaired and unimpaired players (except for player 13). On the other hand the top four of the complainers are concentrated in the group of unimpaired players for these two scenarios (see table 4.7). This forces the mean rating of the unimpaired players for these two scenarios downwards compared to the mean rating of the impaired players.

Looking at table 4.7 one sees that, regarding the configuration of the complainers' impairments, scenarios 1 and 11 are exactly the opposite from scenarios 5 and 15: the top 4 of the complainers now belongs to the group of impaired players. Following the above reasoning, the difference between the mean rating of the unimpaired and the impaired players, respectively, should blow up. For scenario 11 this is obvious in figure 4.5. For scenario 1, this effect is less pronounced (but still present). Taking into account the very mild impairment conditions in scenario 1 and the severe impairment conditions in scenario 11, one can conclude from this that players are able to distinguish to some level between different degrees of impairment.

Worst and best rated scenarios

While the above figures give a global impression of the effect of impairment, more detailed information is obtained in the following.

Since the rating scales of different players might be different and there is no real means to hallmark them, only the best rated and the worst rated settings

		Player											
		1	3	4	5	6	7	8	9	10	11	12	13
Delay	20 ms	2	1	4	2	1	4	6	7	6	4	6	4
	30 ms	3					1						3
	60 ms					1					3	2	
	80 ms	1									1	1	
	100 ms										1	2	

Table 4.8: Overview of occurrence of best rated delay values for all players.

		Player											
		1	3	4	5	6	7	8	9	10	11	12	13
Delay	20 ms	1/10	1/10	1/11	1/10	1/11	1/10	1/11	1/10	1/11	1/10	1/11	1/11
	40 ms	1/3	1/2	1/2	1/2	1/2	1	1	1	1	0	0	1/3
	60 ms	0	1/3	1	1/1	1/3	1	1	1/3	1/3	1/4	1/4	1/2
	80 ms	1/4	1/2	1/2	1/4	1/2	1/3	1/2	1/3	1/2	1/2	1	1/2
	100 ms	1/3	1/3	1/4	1/3	1/2	1/5	1/5	1/3	1/3	1/4	1/4	1/2

Table 4.9: Overview of first step correction factors.

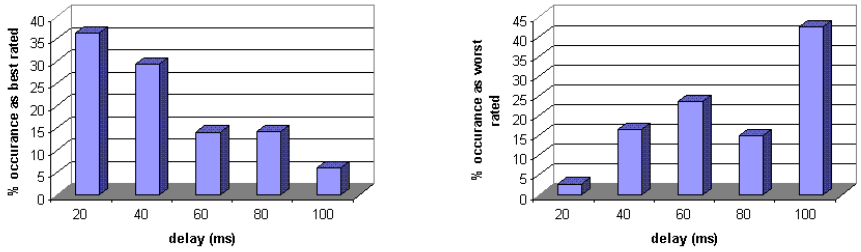
of all players will be used in the following analysis. The main idea is: the more a certain impairment scenario has been given the best(worst) rating, the lesser(more) a negative impact on the network quality is experienced. Note that many players have given their best/worst rating to more than 1 scenario (see table 4.2), as such 1 player can cause different scenarios to be present in the analysis. First we will only consider network delay, in a second step the jitter will be examined.

Table 4.8 summarizes for all players the number of times the different delay values (ignoring the jitter values) have been rated as best. For practical reasons, it was impossible to cover all possible scenarios in the experiment. As such, as can be seen in table 4.1, not all players have been exposed the same number of times to all different delay values. This implies that, even if the players had to choose randomly the ratings of the different scenarios, not all scenarios would have equal chance to get e.g. a best rating. As such, the figures of table 4.8 have to be corrected for this discrepancy. This is done in two steps : first the figures are divided by the occurrence of the corresponding impairment (i.e. delay value) for each player. Table 4.9 gives an overview of these correction factors. Note that, when a certain player did never experience a certain delay value, the correction factor is set to 0 at this point. Adding up all best occurrences of the different delay values, corrected as described above, results in table 4.10.

The second step in the correction accounts for the fact that some players never experienced a certain delay value. This has been marked with a '0' in table 4.9. Since 2 players out of 12 never experienced 40 ms delay, the results

	Delay				
	20 ms	40 ms	60 ms	80 ms	100 ms
Occurrence Best Rated	4.45	3	1.58	1.75	0.75

Table 4.10: First step corrected occurrence of best rated delay values.



(a) Best rated delay values

(b) Worst rated delay values

Figure 4.6: Occurrence of delay values and their ratings.

for 40 ms delay are weighted with a factor 12/10, 1 player out of the 12 never experienced 60 ms delay, so the occurrence of 60 ms delay is weighted with a factor 12/11.

The final corrected figures, normalized to 100, for the number of times (a scenario with) a certain delay values has been rated as best are shown in figure 4.6(a). In an analogous way, results are obtained for the number of times a (scenario with) a certain delay value has been rated as worst. They are shown in figure 4.6(b).

The trend that can be observed from both figures is that it is less likely that a higher imposed delay will receive a best quality rating, and on the other hand that it is more likely that the network will receive a worst quality rating.

An analogue analysis for the imposed jitter values did not yield useful information. The statistical relevancy however for this analysis was very low (see table 4.1) since almost no jitter values were present in more than two best/worst rated scenarios.

Indication for delay and jitter bound

Based on the cases where the players give the 'best' or 'worst' rating to the network, we concluded that the players are able to predict whether or not they are hampered. Next we try to estimate from which level of delay and

jitter they are able to predict that they are hampered. In contrast to the previous paragraph we consider all experiments in this analysis (not just the ones where the players give either their worst or best score) and consider all three questions on the questionnaire.

We want to test the hypothesis, if the players are able to predict that they feel congestion for a delay and jitter pair (d, j) imposed in scenario c. If they feel hampered for this pair (d, j), then the players that experience this kind of impairment, should give a low value to question 1, a high value to question 2 and answer 'yes' to question 3.

In order to translate the subjective rating given by the players to question 1 in binary values (0 meaning that the player indicates that he does not feel hampered, 1 meaning that he esteems he does), we translate the ratings given by a player into 0 if the rating given is larger than the average value given by this particular player and 1 otherwise. For question 2, we set the value to 0 if the rating given is smaller than the average given by the player, and to 1 otherwise. So based on the three questions we have a table (one for each question) with an indication whether or not player p feels hampered in scenario c: $Q_i(p, c)$ indicates (is 1) if player p feels hampered in scenario c, according to his answer to question i.

To perform the hypothesis test, we predict whether or not player p is impaired in scenario c. Player p is impaired in scenario c, if his network path was affected (see table 4.1) and if for the delay and jitter value imposed in scenario c, the player is able to feel impairment, the latter of which is exactly the hypothesis we want to test. So, we define $R(p, c; H(c)) = \text{AND}(T(p, c), H(c))$, where $T(p, c)$ are the values specified in table 4.1, $H(c)$ is the hypothesis that the delay and jitter pair (d, j) used in scenario c is felt by the user as hampering his performance and AND(.,.) is the boolean and-function.

If $Q_i(p, c) = R(p, c; 0)$ this supports the fact that the players do not experience impairment in scenario c (more precisely, for the delay and jitter pair (d, j) used in scenario c) and similarly $Q_i(p, c) = R(p, c; 1)$ endorses the hypothesis that the players do feel impairment in scenario c. So, the hypothesis test consists of determining for each scenario c, which value of $H(c)$ maximizes

$$\sum_{p \in P} \text{EQ}(Q_i(p, c), R(p, c; H(c)))$$

where EQ(.,.) is the boolean equality-function.

Table 4.11 gives the results of this hypothesis test for the three questions in the questionnaire. These tables indicate that a delay below 60ms is not felt as an impairment and that the jitter does not play a prominent role in whether

		Question 1						
Delay	20 ms	0						
	30 ms	0	0	0				
	60 ms	1	0	0	0			
	80 ms	1	0	1	1	1		
	100 ms	1	1	1	1	1	1	1
		5	20	35	50	65	80	95
		Jitter						

		Question 2						
Delay	20 ms	0						
	30 ms	1	0	0				
	60 ms	1	0	0	0			
	80 ms	1	1	1	1	0		
	100 ms	1	1	1	1	1	1	1
		5	20	35	50	65	80	95
		Jitter						

		Question 3						
Delay	20 ms	0						
	30 ms	1	0	0				
	60 ms	1	0	0	0			
	80 ms	1	0	1	1	1		
	100 ms	1	1	1	1	1	0	1
		5	20	35	50	65	80	95
		Jitter						

Table 4.11: Results of the hypothesis tests.

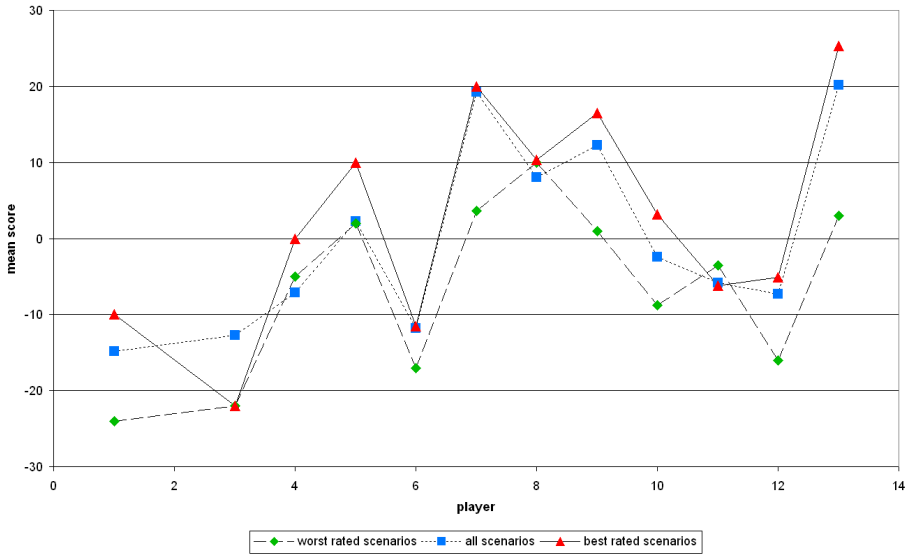


Figure 4.7: Mean score for all players: over all scenarios, over his best and over his worst rated scenarios, respectively.

or not the player feels hampered for the particular FPS game considered in this paper. These findings are supported by figure 4.6(a) where a large gap from 40 to 60 ms delay is present.

4.4.3 Objective versus subjective observations

In order to get an impression whether the perceived network quality is reflected in the scores of the players, again only the data of the best and the worst rated scenarios are taken into account. For each player, the mean score obtained in his best rated scenarios is compared to the mean score in his worst rated scenarios. Both are plotted in figure 4.7. As a reference, the overall mean score for each player is also presented. As can be seen, in general the best rated scenarios yield better scores than the worst rated scenarios.

4.5 Comparison to other studies

To validate the results obtained from our tests, we will compare our findings to related studies. In [Armitage 03] and [Armitage 01], two (identical)

dedicated Quake 3 servers were placed on different continents. A study was carried out to decide which players stayed connected to the server over longer periods, or returned to the same server more than once. These data were subsequently correlated with their mean 'ping' time. It was shown that most 'active' players had a mean ping time under a threshold of about 130 to 200 milliseconds (depending on the amount of activity required for a player to be labeled 'active'). By comparing the data from the two servers, which yielded comparable results, the ping threshold for a server to be preferred by players was determined to be around 150 to 180 milliseconds.

The authors of [Sheldon 03] opted to study the effects of latency on a completely different type of computer game, the real-time strategy game of Warcraft III. Several scenarios were isolated, in which users had to complete a pre-determined set of tasks to achieve a set goal. The main interaction types in this class of computer games were determined to be building, exploring and combat. (Artificially introduced) latency figures ranged from 0 to 4000 milliseconds. After study of the test results, it was determined that for the building scenario, the impact of latency on the total time taken to complete the test was negligible. The same is true for the combat scenario, although for the exploration scenario a correlation was found between time to complete the test and the introduced latency factor. The (at first sight) surprising results can certainly be attributed to the nature of this type of networked game. Interaction is often turn-based or consists of a low number of actions per time frame. Although very little attention was paid to the user experience, a relation between latency and user experience was found to be present.

[Pantel 02] presents remotely similar work to our own, but is based on racing games instead of a first-person shooter. Also, the delay is introduced client-side and is related to the presentation, i.e. measured between the moment of action (key press) and the visualization of that action (car steering, accelerating,...). Delays in presentation range from 50 milliseconds up to 500 ms. Objective measurements are carried out by timing the user as he drives once around the track. The familiarity of the user with racing games/simulators is taken into account. It is determined that for beginner and intermediate players, delay increases (almost) linearly with the amount of delay. On average, lap times of 7 seconds under zero-delay circumstances increase to about 14 seconds under 250 milliseconds of latency. Also, the frequency at which players depart the set course (due to an uncontrollable car) vary from 0.1 to 1.5 per lap (with latency ranging from 0 to 250ms). Querying the users on their subjective experience learned that at 50 ms, no influence is noticed. A latency of 100ms is said to be noticeable through a delay in reaction speed,

but it is not visually apparent. At 200 ms, players report that the delay is observable, but that controlling the car is still possible if the driving style is adapted.

In [Henderson 03], the question is posed whether QoS provisions on networks would yield sufficiently improved experience for users to pay an additional fee on top of their basic Internet access price. Again, two servers for a first-person-shooter game were set up on the Internet, and, alternating between the two, additional delay was introduced. It is determined that, when 50 milliseconds of additional latency is added, the number of users that decide to join sessions on that server drops remarkably (e.g. from 15 to 10). Secondly, an experiment was setup in which delay of 25 to 250 ms was introduced during a session, but only for a short duration (10 minutes). The amount of users that leave the session during this 10-minute impairment period was calculated and found to be marginal for those players that were already active for a long period of time (on average around 45 minutes) - the degree of involvement may be higher for those players. Although not many results are shown, the authors claim that there was an influence on the score - which is consistent with our findings.

Other related work is discussed in [Borella 00] Comparing the results from this related work to our own findings, we see that first-person-shooter games clearly have a much lower tolerance for latency than RTS and racing games. However, under non-optimal conditions (centralized servers on the Internet), players are willing to deal with delay values up to 150 milliseconds, although we have shown that there is a definite influence on the scores under these circumstances (which may be mitigated by the fact that all players are experiencing similar latencies). The degree of involvement in the virtual world may also be a contributing factor.

Chapter 5

Discussion - Part I

From the results described in this part, a number of conclusions can be drawn.

For the first generation of massively multiplayer applications, such as EverQuest and Dark Age of Camelot, a broadband connection is clearly not a necessity, for a number of reasons. First of all, these games rely heavily on a patching system, which eliminates the need for run-time download of game content trough a streaming mechanism. These games also feature minimalistic interaction types (a small pre-defined set) and do not incorporate real-time multimedia streams such as Voice over IP.

In contrast, the second generation of NVE applications do provide a number of additional features that require the additional bandwidth capacity of broadband networks. Architectures such as Microsoft's Xbox Live, with its built-in audio chat features, require real-time streaming of multimedia information. Virtual Interactive Community applications such as Second Life and There are heavily dependent on a continuous stream of geometry information to be able to display the highly dynamic virtual world at a large number of end-stations.

We have demonstrated, using real-life data stream captures, the effective bandwidth usage for a number of these applications. Besides raw numerical data on network utilization, this information also gives insight into the inner workings of the applications, such as the chosen network architecture and distribution mechanisms used, as well as possible 'quality of service'-like provisions (such as the preferential treatment of some data streams over others). Throughout our work on these analyses, this information has proven to be impossible to obtain from developers directly - for mostly obvious reasons.

It does however provide us with valuable information for designing our own architecture, which will be discussed in the following parts.

Using a test setup of a first-person shooter game, we were able to determine the validity of the claim of multiplayer games all over the world that ping times have a direct influence on their performance. From our experiments, we were able to deduce the fact that 'network impairment', consisting of both delay and jitter, does have a negative influence on the affected players' perceived game quality and performance. A second interesting conclusion to be drawn is that players that are not directly subjected to impairment, are not hampered by possible impairment present for other players in the gaming session. From a user perspective, it has been shown that the players' perception of the quality of the game is dependent on the size of the delay that is present in the network (with indications of a boundary value of about 60 ms). This perception is correlated with the actual performance of a player in a particular session.

In the next part, we will look into the actual design of a large-scale NVE framework, built from the bottom up with next-generation network features in mind.

Part II

Architectural considerations

Introduction

In this part, we will take a closer look at the intricacies of the supporting network architecture behind a massive virtual environment. It should be pointed out that, besides the large amount of simultaneous users, large-scale environments are also to be considered 'massive' in the sense that they occupy vast amounts of virtual land space. Although the architectures used in some of the examples shown in the previous part were already (briefly) explained, this part will introduce the ALVIC architecture using a bottom-up approach. While the existing deployments are clearly able to scale up to the required number of users, it should be pointed out that – especially from a research point of view – more optimal alternatives can be devised, as one is not directly hindered by practical issues such as content management, security and moderation.

It is vital for an application that deals with (possibly) huge amounts of incoming data to be able to influence the data flow that is received. Instead of relying on an intricate server-based design, the ALVIC architecture employs the implicit scalability enhancements that a multicast-based design provides.

Besides the architectural design of ALVIC, this part will also introduce the scalability testing methodology which is employed to attest that the claimed scalability is achievable in practice. Instead of abstracting and modeling the traffic flows associated with a networked virtual environment, the ALVIC software platform is used to effectively deploy 1000+ simultaneously connected agents on a local area network.

Chapter 6

Design of the multicast-based ALVIC framework

It should be obvious that, for a virtual environment to be displayed in a consistent state on a multitude of connected end-points, large amounts of data needs to be shifted around the network. In particular, once a user decides to undertake a specific action, the details about the maneuver need to be distributed to either a server (for further processing and/or distribution) or other clients that should visualize the action. While both pure client/server and peer-to-peer based approaches have their pros and cons, a combination of both is able to provide both protection from breakdown due to server failure, as well as facilitation of scaling of the architecture towards many thousands of users.

For peer-to-peer systems to be applied in this context however, a lot of uplink bandwidth is needed, as the uplink bandwidth usage is directly linked to the amount of users that are present in any one given part of the virtual environment. This is clearly in contrast with the fact that most home broadband connections are asymmetric in nature, meaning that they offer much more throughput in downstream direction than upstream. These peer-to-peer systems are also hindered by the fact that the upstream throughput cannot be controlled by the end-user, as he/she is directly dependent on the amount of 'interested' people in the vicinity that need to receive state updates.

We have therefore opted to base our framework design on the concept of multicasting, in which a sender of a datagram can reach a number of other users through a single 'send' action, consuming only a single 'unit' of bandwidth in the upstream direction. While some limitations exist, on which we will digress in a later section, it will be shown to be a powerful alternative to

client/server only systems, as scaling a multicast-based virtual environment proves to be an almost trivial task, not hindered by single points of failure or processing capacity. Throughout this text, we will refer to the framework as 'ALVIC', an acronym for 'Architecture for Large-scale Virtual Interactive Communities'.

6.1 Justification behind the choice for multicast

6.1.1 Relation to identified deficiencies

When looking at the results gathered from all experiments discussed in part I of this text, we can see that two main architectural options are currently deployed in existing (commercial) applications. MMORPGs such as EverQuest, Dark Age of Camelot and World of Warcraft use a client/server-only architecture. This provides the application providers with two advantages: on the one hand, they are in complete control of all that goes on in the virtual world at any given moment in time, as all actions performed by the end-users need to be transmitted to the server. Moderation becomes easier, as the service and content provider (SCP) can intervene and moderate all activity at these central points in the network. At the same time, eavesdropping by other users becomes less probable and access to the virtual environment is restricted by requiring client authentication. The main advantage of this approach from the customer's point of view is that all traffic can be routed through the server infrastructure, eliminating the need for multiple transmissions of packets in the upstream direction. While these advantages are clearly advantageous from an SCP's point of view, the downside is something that cannot be measured by looking at data streams. We have also mentioned this fact in the discussion in part I, as server-to-server traffic cannot be intercepted at client-side. It should however be obvious that there is need for extensive synchronization traffic between servers in the architecture in order to maintain consistency. This is doubled by the fact that each server is, in theory, a single point of failure for the whole application, demonstrating the fact that there is need for fail-over in the form of redundant servers and (most likely expensive) storage capacity. There is also the issue of bandwidth consumption, as all traffic needed to keep the world in a consistent state at user side needs to be transmitted by the server infrastructure. As bandwidth is a relatively expensive commodity for SCP's, it is also one of the reasons for the fee-based subscriptions needed for current massively multiplayer games.

The Xbox Live architecture takes a different approach, as it combines a

client/server architecture with peer-to-peer state exchange. The Microsoft server infrastructure is used for matchmaking and authentication purposes, but the bulk of data needed to synchronize state is exchanged either directly between clients or through a client that has dynamically been assigned the role of 'server'. This is obvious from the example we discussed for the MotoGP game. Sessions of these types of games are typically short in duration and are non-persistent in nature. They are therefore ideal candidates for deployment in peer-to-peer architectures. From the SCP point of view, this architectural setup has the added advantage of requiring only a small amount of expensive server-originated traffic, making it easier to provide a basic feature list for free (but still controlled) and to easily scale the amount of users using a relatively small server farm. However, from a client point of view, things become a lot more difficult. The dynamically assigned 'server' for each session needs to process and transport all data streams for active users in the session. Besides the obvious throughput restrictions this poses, mainly in the already limited upstream direction, it is also a non-trivial task to configure home NAT routers and firewalls in order to forward all necessary traffic to the internal game console or PC.

6.1.2 Application in current- and next-generation networks

Based on the problems described above, the application of multicasting in these scenarios seems like the ideal solution, as it would facilitate the combination of using client/server based architectures with peer-to-peer features, without the major drawback of explosive upstream bandwidth usage at client-side.

Multicasting is, in fact, being used in current access networks for a variety of applications. Looking at, for example, digital TV over IP transmissions, it should be obvious that ability to easily transport a stream to a dynamically changeable number of end-users is a powerful feature. The absolute gain on the overall bandwidth consumption from the ISP/TVSP point of view turns out to be especially advantageous when multicast is deployed on a shared medium LAN such as cable networks, as all end-stations are, by definition, capable of receiving all signals and data physically present on the wire (need only be filtered by individual receivers). Multicast is currently employed for near video-on-demand and for interactive TV presentations by a number of digital TV providers. Even for non-shared medium networks, the fact that data that originates at a specific location in the network is only replicated at the edge of the access network has a clear positive impact on the overall

throughput requirements of the backbone network.

While multicasting was a feature already present in the earliest designs of the IPv4 protocol - witness the reservation of a relatively moderate amount of address space for this purpose - the uptake and effective implementation of multicast in network equipment at the end of the previous century was truly minimal. At design time, possible applications for extensive use of multicast could not be envisioned, and as such, hardware developers were reluctant to include the features in their equipment. The supporting control protocol behind IP multicast, the IGMP (Internet Group Management Protocol) has gone through a number of revisions, starting with version 0, defined in RFC 966 in 1985, already 4 years after the RFC for IPv4 was released. The latest version, v3, defined in RFC 3376, was presented in 2002 and includes additions for much-wanted features such as source filtering.

For LAN applications, multicast support does not present a real challenge, mainly because of the fact that Ethernet was developed as a shared medium network technology. Deployment of multicast features on a local basis is a viable option because of built-in support in the lower layer protocols. For example, in an ethernet address, the low-order bit of the high-order octet is used to make a distinction between unicast and multicast addresses. A value of 0 signifies a unicast address, while a 1 in this position indicates a multicas address. At higher protocol stack levels, and especially in larger (inter)networks, routing issues become prevalent and create a host of problems. Because of the lack of support for multicast features in the backbone infrastructure of the Internet (especially in the eighties and nineties), workarounds have been developed such as the MBone initiative, which provided an experimental backbone specifically for multicast. Several research projects made use of the MBone infrastructure, for example for demonstrations of multi-party video conferencing and shared whiteboard applications. With increased support for multicast in current- and of course next-generation networks, the MBone initiative is rapidly becoming superfluous, making it feasible to run multicast applications on the Internet, at least in the backbone and for service providers. Typical end-users are still unable to send multicast datagrams in the upstream direction, as ISPs are afraid of an explosive growth in traffic, flooding their own access networks and the (expensive) international peering links with other service providers. With this impediment in mind, the CastGate[CastGate] initiative was developed in order for end-users to make direct use of the multicast-enabled backbone of a WAN, without themselves being able to send multicast packets. The software routes the multicast datagrams transparently between the host and several CastGate servers located within the multicast-enabled backbone. We

will discuss the features and impact of this software more specifically in section 10.3.3.

6.1.3 Inherent limitations and problems

Using a peer-to-peer state exchange system using multicast presents some inherent problems that are difficult to overcome because of the way the mechanism is implemented in today's Internet Protocol. As we are looking at developing a multicast-based architecture that can also be deployed on next-generation networks however, these issues can (partly) be solved by a better design (partly) present in future revisions of the protocol stack. We will present some of these limitations here, mainly to create a better understanding of the lack of use of multicast in currently deployed networked virtual environment applications.

Multicast groups are 'open' by design, meaning that anyone is free to join any group (as long as the scope is defined over the Internet) without any regulating parties. In practice, this means that anyone is able to eavesdrop on data transmission being sent to any one multicast address. Also, considering the other direction, anyone is able to send data to any given multicast group address, possibly disturbing other applications that were using that same address. It is because of these issues that cheating and security issues in general pose difficulties in a multicast-only scenario. While in a client/server scenario, data flow (and consequently visibility) is regulated by the server, multicast directly exposes all traffic sent to the multicast address. It would clearly be easy for a malicious user to inject malformed or modified packets into the multicast group, in order to exploit vulnerabilities and/or design flaws in the client software.

There is also the added complexity of synchronizing state among a large number of peers in a peer-to-peer only system. As their is no single governing authority that determines a common 'state' for all connected hosts, clients are responsible for tasks such as collision detection, floor control and persistence of the world as a whole. The topic of maintaining consistency in peer-to-peer based systems is a subject on its own, and we refer to [Singhal 99] and [Vaghi 99] for further discussion.

Clearly, a pure multicast-based system is not desirable, due to the reasons mentioned above, as well as the management of such a system. In the next sections, we will present the multicast-based framework developed, which combines the strong points of both architectures into a single design.

6.2 Alleviating server load

While, in theory, it is entirely possible to design a networked virtual environment architecture using only multicast traffic, we have opted to include a set of governing servers into the architecture. Their purpose is threefold: authentication, network resource management (e.g. multicast addresses) and server resource management. The minimal load on these servers in the architecture allows for a large number of clients to be simultaneously connected to a single server (for more details, see section 7.4.1) and facilitates the distribution of load over several physical machines.

6.2.1 Identification of remaining roles

The server infrastructure in our design requires both a limited amount of administrative servers and a larger number of dedicated gameservers, depending on the targeted number of simultaneously connected users (figure 6.1). Each of the administrative servers has a publicly known named Internet address, which can be pre-programmed into any client software. They provide both an easy point of reference and an excellent opportunity to introduce load-balancing among servers that are hidden 'behind' these publicly visible machines. One of the key responsibilities of these administrative servers is to handle all login requests. Every client that wants to participate in the on-line world needs to go through an authentication phase in which their current account information is checked and updated (figure 6.1 A). Once authentication is successfully completed, a redirect is done to one of the 'game servers', which are responsible for handling session-specific traffic (see figure 6.1 B, detailed in section 6.3.2). Afterwards, clients may choose to disconnect from the administrative server.

The administrative servers may also be implemented to handle other responsibilities besides client authentication. A prime example of this is billing, as is used in most of the current on-line games, which, as was said before, require a periodically charged subscription fee. The administrative servers can keep track of these payments through a back-office application (and payment infrastructure) and deny access to the world if requirements have not been fulfilled. Because of the length of some sessions, clients will (ideally) periodically be requested to re-affirm their subscription information to prevent session stealing. The entire back-office application that is needed for payment tracking, subscription information and logging is not detailed in this architecture, as these are readily available from commercial vendors. Through the

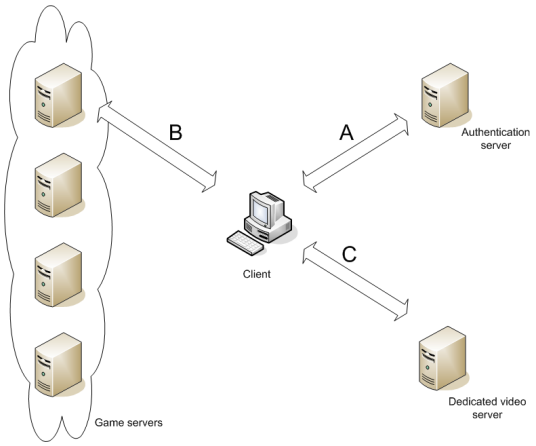


Figure 6.1: Server connections.

back-office, a link can also be made to in-game events, such as buying virtual goods and/or expansion packs: these are quite trivial and are not discussed further in the context of the framework.

Game servers are dedicated to providing the clients with data that is necessary to position themselves, interact and communicate in the virtual world (figure 6.1 B). Examples of these tasks are notification of multicast addresses and handing out unique session identifiers. It should again be noted that these responsibilities are truly minimal, therefore allowing a large amount of users to be connected to the same game server. Specifically how large these numbers are will be demonstrated in section 7.4.1. In the first version of the architecture, a third type of server was envisioned that could optionally be included: the Dedicated Video Server (figure 6.1 C). More on this subject in section 10.4. A more detailed discussion on the main responsibility for the game servers is deferred until section 6.3.2.

6.2.2 Topology of a virtual world

The architecture behind our virtual environment framework should be adaptable to several usage scenario's, ranging from games to virtual interactive communities. Each of these applications should be able to be deployed on the same architecture, preferably even concurrently. However, in practice, this means that each end-user can, at a given time, only be present in one specific world. Because of the extensive size of the virtual world, we followed an approach similar to that in [Waters 96a], in which each virtual world, run-

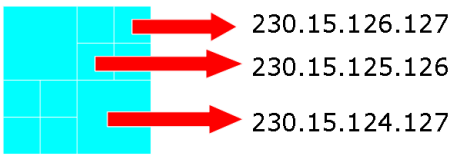


Figure 6.2: Example assignment of multicast addresses to regions.

ning on the server infrastructure, is divided into a number of square regions, whose size depend on the estimated number of active clients in that region and on the type of region. For example, a region that represents a small room inside a building would most likely be scaled to equal the dimensions of the room. Clients that move around the world dynamically enter and leave regions depending on their position.

The reasoning behind this subdivision of the world is to effectively link the physical properties of the virtual world (geographic location) with the underlying network architecture. The relation between the two entities is strong because of the fact that data propagation can easily be coupled to visibility. If an object is invisible to the end-user, there is no need for any data to be received. Furthermore, by assigning a distinctive multicast address to each of the regions defined before, we can reduce unnecessary network traffic. This is illustrated in figure 6.2.

In fact, event information, origination from a single end-user should only be sent to the multicast address of the region from which the event originated. When a client enters a region, a simple subscription to the multicast group assigned to that specific region suffices to start receiving state information on all objects present in the region. As all members of a region send their generated events to the same multicast address, it should be clear that they will also receive all events from other members in the same group without the need for an explicit distribution mechanism through a dedicated or ad-hoc defined server.

We have already explained that a mapping of these (geographical) regions onto multicast groups is an efficient way of distributing data. There is no need to maintain open connections with a (number of) server(s) to receive state information. Neither is there need for determining where to send data, as the current location is always known by a client. The key to the entire system is the fact that data distribution within a multicast group is done implicitly. However, as pure IP multicasting is done over the unreliable UDP protocol,

this may present some new problems, given the fact that some or all data sent between systems may be relying on the correct reception of events of other users in the area of interest. A number of ways have already been investigated to create a reliable multicast protocol for use in different situations. An excellent overview is given in [Obraczka 98].

The way in which normal reliable transport protocols operate, using a system of positive acknowledgements when packets have been received in the correct order and without errors, does not appy to multicast transmissions. If all receiving stations were to send their acknowledgements to the sender, an explosion of ACKS would ensue and the sender's transmission channel would be swamped with this control information. Also, because of the wide range of applications to be supported and the specific requirements for each of them makes it practically impossible to design a single and universal transport protocol such as TCP for unicast transmissions. One of the first implementations, the Multicast Transport Protocol or MTP [RFC 1301] assigns 'masters' that control admission of receivers to multicast groups. Through a system of tokens which are required before transmission by a sender, reliable communication is obtained. MTP also uses a system of negative acknowledgements, in which a NACK is only sent when a packet is tagged as 'lost' by one of the receivers. The subsequent retransmission is sent to the entire multicast group. The Reliable Multicast Protocol or RDP [Whetten 94] implements congestion control through the same algorithms as TCP (based on a sliding window). If a NACK is received (NACKS are multicast to the entire group in this protocol), the windows size is adjusted at sender-side. Another interesting example is the Scalable Reliable Multicast transport protocol [Floyd 97], which allows for any member of a group to answer the request for retransmissions (signalled again by NACKS). The protocol has successfully been used to set up a shared whiteboard environment. Other work is presented in [Liu 95].

In case of state transmission in virtual environments, minimal overhead and scalability are crucial factors. Looking at actual NVE applications such as FPS games, it can be seen that some resiliency to packet loss is easy to obtain[Bernier 01], depending on the rate at which update packets are received. When considering typical packet loss figures, it becomes unnecessary to bother with the added overhead of a reliable multicast protocol (including the time taken for retransmissions). Instead, this time is better spent focussing on the correction needed to hide the effects of lost packets for the end-user.

Another major problem which is mainly due to the inefficient integration of multicast support in the IPv4 protocol stack definition: the limited number of available multicast groups. At design time, IANA reserved the range

of addresses from 224.0.0.0 to 239.255.255.255 for multicast purposes. Unfortunately, many subnets defined in this range have a local-only or otherwise limited scope, and are only therefore usable only on singled-out parts of the Internet (the time to live (TTL) value in the IP header was originally meant to determine the scope of the multicast transmissions, e.g. between institutions, international or intercontinental). In practice, this means that there should be a database that keeps record of the multicast addresses currently in use, and that is responsible for assigning new ones from the pool.

To cope with some of the inherent issues concerning pure multicast issues, it may be advisable to consider using data encryption for all transmissions of state information. However, as this is a CPU-load- and time-consuming process, it may or may not be an option in any situation. The topic of encryption is not crucial to the development of the architecture and will not be discussed in further detail.

6.2.3 Comparison to other spatial subdivison methods

The idea of using spatial subdivision for a larger virtual world is certainly not unique. In this section, we will compare our approach to those of some famous examples (which used multicast) that came before.

MASSIVE2[Greenhalgh 96] introduced effective spatial subdivision into the architecture. A global overview of the world is provided by a world group, with limited detail, containing only top-level artefacts (most important objects in the world) The subgroups are identified within this world group as separate artefacts with an associated multicast address. When a client connects to the virtual world, the top-level artefacts are first downloaded from the world group. If the focus changes, the client is up to date on which subgroup it has to subscribe to for receiving the appropriate information. Membership of artefacts (objects) to groups is controlled by the group masters, which send out join invitations. In turn, when an artefact leaves a sub-group, it sends a leave message to the group master. This system is fundamentally different from ALVIC in the way that it assigns responsibilities for specific groups to several master objects. In ALVIC, information about group boundaries is present client-side, and multicast group address assignment is done server-side. While the MASSIVE-2 approach limits server responsibilities even further, scaling the approach to a vast virtual world containing thousands of objects or subgroup is clearly difficult as the global information of the world is downloaded at once from the world group.

SPLINE[Barrus 96, Waters 96a] defines ‘locales’ as the chunks that make

up the entire virtual world. Information relevant to the locale is only sent to a group of users that is likely to be interested. A separate multicast address is assigned to each locale, to make it possible for clients to determine what information to receive. As can be seen, the spatial subvision technique is very much similar to the one used in ALVIC. However, the creators of SPLINE felt it necessary to have a 'caching' server for each of the locales, to enable users to get the initial information from the locale more rapidly than through the normal event mechanism. This is due to the fact that objects in SPLINE do not announce their state as often as is the case in ALVIC. The chosen event mechanism is clearly more beneficial for worlds with a lot of static objects, but increases the complexity of the architecture by requiring locale-specific caching servers.

The authors of [Macedonia 95b] hint at the possibility of using spatial subdivision in NPSNET-IV to lower network and processing load, but no details or test results are provided. Other related work is presented in [Lea 97], [Pryce 97] and [Roehl 97].

6.3 Enhanced client responsibilities

The fact that server responsibilities are relaxed means, in turn, that clients will need to fulfill additional tasks. However, this is certainly not necessarily a bad thing, as we will show that clients are often able to make more intelligent decisions regarding their bandwidth usage than a server can, without too many extra processing requirements.

6.3.1 Area of interest management

A first trivial function to be carried out by clients was already described in a previous section, namely the tracking of the region and associated multicast group he or she is located in/subscribed to at any one given time.

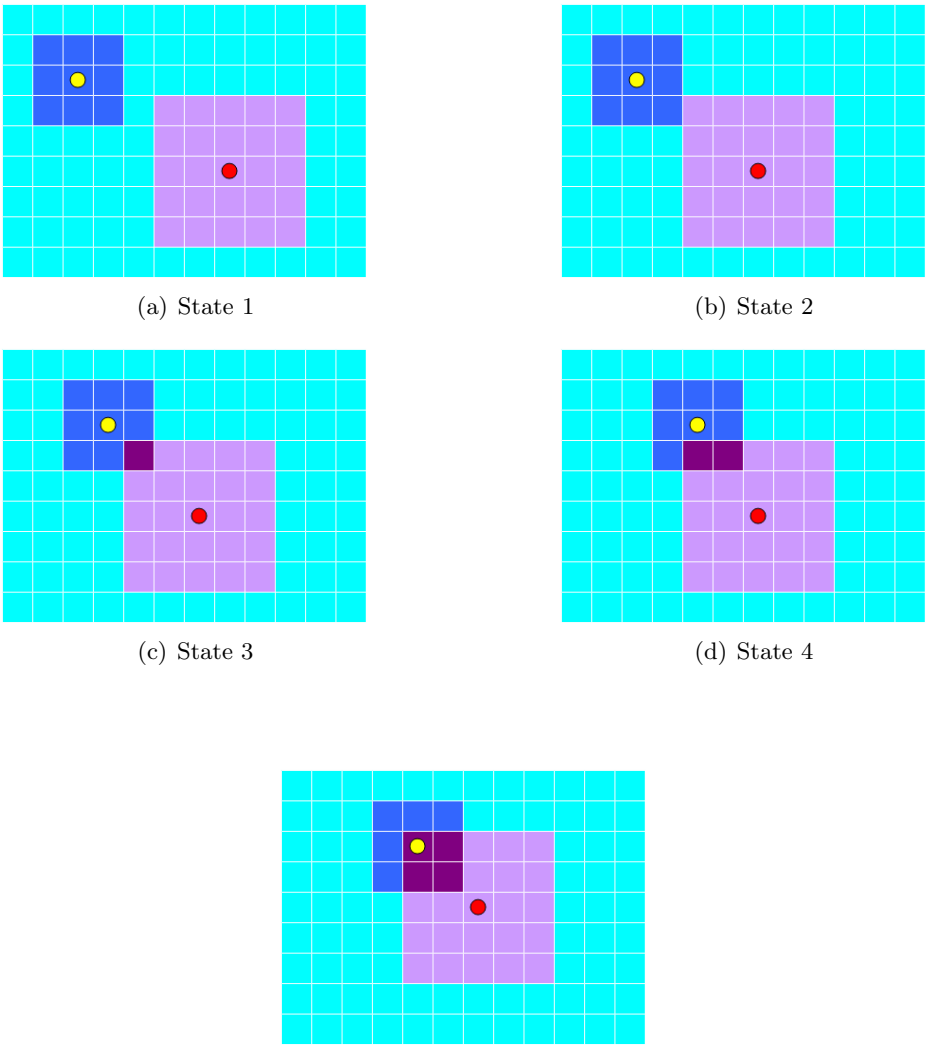
Besides this first trivial task, each client is responsible for managing its own 'area of interest' or AOI, analogous to e.g. [Morse 96]. It is of vital importance to note (as stated before) that there is a coupling between geographical regions and their associated multicast addresses. It can clearly be seen that at a specific moment in time, a limited number of other regions will be located in the view frustum of a client. It is therefore only necessary for a client to subscribe to exactly those regions. The view frustum is entirely decided upon at client-side, and can be adapted dynamically to either expand or shrink depending on several factors, such as available bandwidth or processing

power. We point out here that a large view frustum does not have any impact whatsoever on the upstream traffic needed for sending out state information, as this data only needs to be sent to the local multicast address.

Consider the scenario as shown in figure 6.3, where two players are present in a virtual world consisting of about 12 x 9 regions. Present in the world are 2 clients, denoted by a yellow dot (player X) and a red dot (player Y). In figure 6.3(a), we demonstrate the fact that the size of the area of interest depends on the properties of the client and may change dynamically at run-time. Figure 6.3(b) shows that the AOI is moved along with the position of the client in the world. In practice, this means that at region boundaries, new areas are subscribed to and regions no longer required are left. In figure 6.3(c), it is shown that regions may overlap, but this will not necessarily cause players to be visible to one another. In fact, only the objects in the overlapping region will be visible to both users (also shown in figure 6.3(d)). In figure 6.3(e), we demonstrate that in fact it is entirely possible that one player can see the other but not vice versa. In this example, player Y can observe the actions undertaken by player X, but this fact need not be known to client X (and will, subsequently, not have an impact on the upstream traffic of client X).

The dynamic nature of AOI management in the architecture, made possible through the linking of the definition of the virtual world to multicast groups, is key to the scalability of the environment. Through this mechanism, we get the ability of throttling bandwidth usage in downstream direction almost 'for free', while not impacting the throughput in upstream direction in any way. It is up to the individual client to monitor their available resources (which may be defined in both network and processing terms) and to adapt their AOI at run-time. In case availability of one of these resources is exceeded, it is easy to fall back to a more minimalistic view of the environment. We will refer to the combination of determining factors for the size of the AOI as the subscription policy.

In fact, the chosen client subscription policy is closely related to the visualization of the game environment. Our system includes several views of the environment such as the classical first-person and third-person perspective. Based on the position of the viewpoint, subscriptions are first made to the regions that are adjacent to the region that contains the viewpoint. If the client software decides that it has sufficient bandwidth at its disposal, it can autonomously increase this area of interest by subscribing to regions that are further away, taking care not to include regions that are too distant to have any useful effect on the visualization. We should however also take into account the fact that a larger region of interest means that more time is needed



(a) State 1

(b) State 2

(c) State 3

(d) State 4

(e) State 5

Figure 6.3: Example of changing area of interest.

for the rendering stage, depending on the availability of hardware acceleration at client-side. Many level-of-detail solutions currently exist that can be used to resolve this problem in a satisfactory way. These topics are however outside the scope of this work, so we will assume this problem is handled by the system and extra regions are only requested if the current frame rate allows it.

6.3.2 Game server role and distribution

Now that the AOI assignment has been discussed, we will look into the remaining responsibilities for the game servers in the architecture. We mentioned before that multicast groups are a scarce resource because of the limited scope of addresses that has been pre-defined in the IPv4 protocol. It is therefore important to assign them dynamically, based on the presence of clients in parts of the virtual world.

At each time a new region is entered, a poll is made to the game server responsible for that particular part of the world to request the currently assigned multicast address for the region. In case the client is the first one to enter a specific region, a new address is sourced from the pool of available groups and the address is assigned to that region. A counter will keep track of the number of active clients in a region. If this counter reaches zero, the association between the region and the multicast group is released and the address is added to the pool.

To optimize the underlying protocol, a number of regions can be requested at the same time, instead of sending individual requests for each of the regions in the client's AOI. The fact that requests are made to the game servers, also allows for some tracking of user activity from a management point of view. It is possible to gain an overview of the distribution of clients throughout the virtual world, and therefore also to change the definition of the multicast regions to a more fine-grained grid or to combine several regions into a single unity.

Regarding possible distribution of game servers among a farm of machines, we should point out that, depending on the size of the virtual world, one may decide to have a number of servers, all of them managing the same world. In this case, all that is needed for inter-server synchronization is for the servers to access a common database with, amongst others, available and reserved multicast addresses. The servers can operate largely independent of one another and the infrastructure will be freed from the single point of failure formed by having a single game server. On the other hand, it is also possible to have

a number of machines, each one responsible for a part of the world (defined as a group of regions). In this case, servers only need to synchronize the boundaries between the parts of the world that define each one's responsibility. While synchronization is facilitated and there is less need for a common data storage facility, the fail-over rate is clearly diminished as data will be lost when a server (responsible for a part of the world) goes down. A combination of both approaches can therefore be considered to be the optimal solution for a truly large-scale deployment with fail-over provisions.

6.4 Event systems and associated problems

Up until now, we have referred to the data that is exchanged between end-users simply as state management information. In fact, a number of solutions exist to transmit state between entities to be able to synchronize points-of-view. The most trivial of these is the use of absolute state updates, in which packets are exchanged that simply consist of a vector of positional and orientation information. Each time such a packet is received, the location of the associated object is updated locally. It should be obvious that this trivial way of distributing state does not provide satisfactory results in real-life scenarios. Some of the observed artifacts are non-fluent movements and jumpiness under presence of packet loss, not to mention the enormous amounts of data needed when combined with high update and frame rates.

While the focus of research is clearly on the development of the underlying scalable architecture for NVE applications, there is also need for the development of test applications on top of this architecture to ensure the principles we envisaged would work out in practice. It is because of this fact that a few (non-essential) optimizations to the synchronization mechanism are implemented to generate more visually pleasing results as well as to limit the network traffic needed. We will discuss these optimizations in the next subsection.

6.4.1 Event synchronization

Any movement, rotation or other action initiated by a client is referred to as an event. The challenge is to synchronize the stream of incoming events at client-side under varying network conditions (mainly latency and jitter related issues). Synchronizing clocks of interconnected systems can easily be accomplished through the use of the Network Time Protocol (NTP) [RFC 1305]. Each event that is being sent should be accompanied by a timestamp and is interpreted at the receiver side according to this timestamp. As all incoming

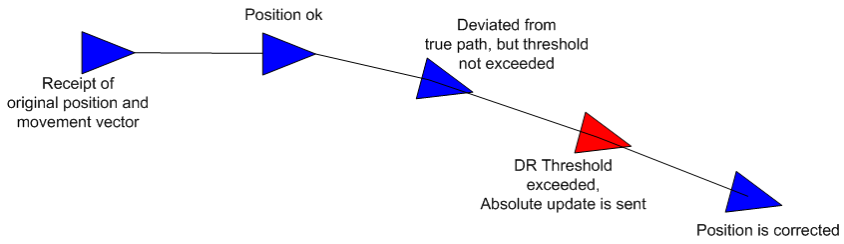


Figure 6.4: Example dead reckoning scenario.

events are kept in a queueing system, the rendering system can use data from all previously received events to determine the state of the virtual world at every given time, up to the length of the queue. When continuously deriving the state and position of objects from a high-level description, it is likely that these calculations will, at a given time, start diverging from the actual situation due to rounding errors or network delay, introducing inconsistencies amongst clients. We therefore allow for all clients to send periodical updates of absolute states, such as positions and orientations. By converging calculated object status to this exact information we are able to mask a number of these issues from the user. Depending on the interval that is chosen between these absolute state updates, the perceived quality at client side can be optimized, but with an adverse effect on bandwidth consumption.

6.4.2 Extension of the concept of dead reckoning

A technique that is often used in virtual interactive communities, on-line games and the like is dead-reckoning. Originally developed for navigation purposes and first implemented for use in NVEs in SIMNET, it uses the combination of a previously known position, along with heading and speed information to derive the location of an object at a specific time. An adaptation of the original system is commonly used to determine the state of objects without the need for constant updates to be sent. The process is depicted in figure 6.4, where an object is moving along a path. As the original position is known to the receiving end, a combination of this information with speed and orientation vectors can be employed to calculate the likely position of the object. As the path followed by an object is never straight, it should be clear that at some time the actual position will have diverged from the dead-reckoned state beyond a certain threshold. It is at this time that an absolute update will need be transmitted, necessary to correct the error.

Dead-reckoning works on the lowest level description of events, namely position and orientation information. In our framework, we have abstracted these to a higher level. Examples of such high-level events are 'start walking' and 'make wave gesture'. Using these types of events allows for an integration of other types of information into the protocol, i.e. information needed for animation purposes or for triggering of external events can be easily embedded into the data stream. Another advantage of using such a high-level protocol is that each individual client can perform different actions, based on the same protocol information. For example, a client that is connected to the mobile phone, can opt not to display avatar animations, but will still be able to display movement. At the same time, the exact same data stream can be interpreted by a desktop PC client with full-fledged hardware-accelerated rendering and multimedia capabilities. Also, the effect of these events can be defined for entire classes of devices in advance and distributed using a patching mechanism. Of course, in its simplest form, the protocol will assume the classical form of a vector containing position, orientation and speed information, on which the dead-reckoning technique can directly be applied.

6.4.3 Concealment of transmission problems

Because of the obvious presence of deficiencies in the form of delay, jitter and packet loss in current-generation wide area networks (as discussed extensively in part I of this text), we cannot conclude this chapter without discussing some ways of concealing these errors.

The first optimization presented here is the adjustment of the timestamp of actions that are transmitted. By increasing the timestamp by a factor δ, we can create events that take place in the future (when considered from the senders' point of view). If the δ factor is kept below a certain threshold, no negative impact is observable by the sender. At the receiving end however, it should be obvious that when the δ factor is defined lower than the actual network delay, the events will be played out in a synchronized way as they can still be processed in time.

The following example will clarify this. Suppose that at a given time a user instructs its client to move forward. The client will then build a packet containing the *start walking* message. However, instead of setting the timestamp to the current time, the timestamp is set to a time in the future, e.g. 150 ms later. Locally, this will cause the action to take place after a small but hardly noticeable delay. When the action is distributed, this increased timestamp value makes sure that network delays below 150 ms do not affect

the synchronization between clients.

Unfortunately, the δ factor that may be chosen is dependent on the type of application that is being run on the architecture. We have demonstrated in detail in the previous part that for a first person shooter, an appropriate δ value would have to be below 60 milliseconds, in order not to influence the quality of experience. We wish to reiterate the results already cited before in section 4.5 and [Pantel 02] which state that delay factors up to 200 milliseconds are acceptable for other types of games. For other less time-critical applications, this solution can prove to be a workaround for the most common delay values experienced on the Internet today.

The second workaround we implemented is to 'mask' the actual error correction caused by the receipt of an absolute state update. These error corrections may be sent by the dead-reckoning algorithm when the pre-defined error threshold is exceeded. The 'masking' effect is achieved by providing a smooth transition from one state to the other, implemented for example by an increase in velocity of a moving object. Note that the high-level event mechanism again comes into play, as it can be used to direct the use of specific animations or transitions (to be displayed by various clients).

Regarding the issue of packet loss, it should be obvious that the impact this has on a system that relies heavily on an event-based synchronization mechanism is quite significant. However, solutions exist to perform reliable transmissions over non-reliable networks, even for multicast schemes. We refer to the discussion in section 6.2.2 for some pointers to related work and possible solutions. However, as typical packet loss figures are low (at least under normal circumstances) and since we are discussing non-essential optimizations, we have not implemented these into the architecture.

6.5 ALVIC software design

The original software design behind ALVIC was developed in such a way that new ideas and the basic technology could rapidly be integrated into a working setup. This original design is also the one used for the scalability evaluation and the addition of communication streams (which will both be described in other chapters of this dissertation). As the framework matured, it became apparent that the design was too rigid to enable a wide range of applications to be easily mapped onto the software components. This lead to the refactoring of the software components in ALVIC, extending them towards a more generic interface. As the new software design is the one that is used in all new developments regarding ALVIC, we will focus on the latest version only.

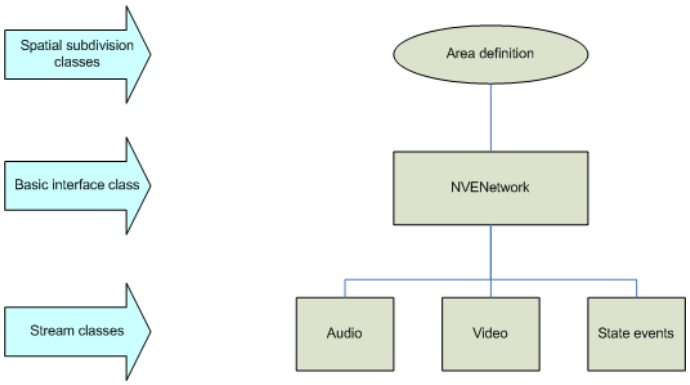


Figure 6.5: ALVIC high-level software design.

A high level overview of the software architecture is shown in figure 6.5

As can be seen, the spatial subdivision technique is integrated in such a way that several implementations can co-exist. A basic interface between these components and the remainder of the architecture is defined in the abstract base class. While the original ALVIC design only allowed for a single spatial subdivision method to be used, it was shown in several tests that it might be desirable (due to the wide range of applications) to be able to switch between various implementations. Central in the implementation is the notion of 'areas', which may take any form or shape, as long as they can clearly be distinguished from one another.

Also apparent in the new software design is the increased abstraction of data streams to generic bit pipes, independent of the contents. Several basic functions are defined, which are common to any stream used in a networked virtual environment, such as subscription management, transmission functions etc.

The basic distribution classes are designed in such a way that several distribution schemes are easily interchangeable (see figure 6.6). This includes, for example, a multicast scheme that is not dependent on a spatial subdivision method, a scheme encompassing the original ALVIC optimizations and a unicast scheme, using ENet to provide reliable packet transmission over UDP.

We will refrain from performing an in-depth analysis of the software architecture as it does not attribute to the understanding of the workings of ALVIC. However, they have proven to be a powerful way to test and integrate additional features required by next-generation applications.

In the next chapter, we will focus on the ways in which scalability of the

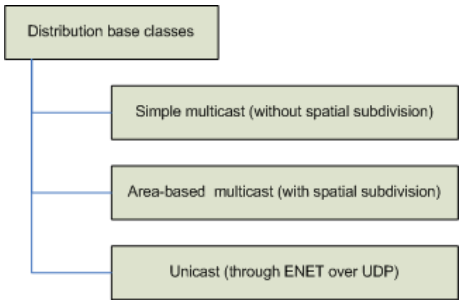


Figure 6.6: ALVIC software design - distribution.

ALVIC solution was attested.

Chapter 7

ALVIC scalability evaluation

In the previous chapters, we described the fundamental building blocks that make up ALVIC. We also stated that the extensive use of multicast to distribute state amongst clients would guarantee a scalable solution, with the added benefit of a minimal investment in server capacity. To support these claims however, we would have to deploy the architecture on a vast scale, which is clearly impossible to to for an academic research project. On the other hand, supporting our statements using simple extrapolations of captured traffic of a very small amount of clients is not advisable. Such an approach runs the risk of not exposing some of the intricacies and flaws in the architectural design that rear their head only when several hundreds or thousands of clients are connected at the same time. We have therefore opted to design a test-bed to effectively simulate the presence of a (nearly) unlimited amount of connected users. This enables us to determine the load on both client- and server side, and has the added benefit of generating actual data flows which can be measured using traditional packet analysis software. Our goal is to simulate as many of these concurrent users on a single system as possible. Combining these machines into a single physical network provides a cheap and practical alternative to real-life large-group testing. We have chosen not to investigate the scalability of the architecture using network simulation software such as NS2 because of two reasons. First, the nature of ALVIC, being a networked application targeted towards large audiences, makes it difficult to trace all bugs in software that may only appear when large volumes of data are transmitted over the network. Using the actual software implementation in the scalability tests allows for effective software-component testing in a realistic

environment, resulting in more stable end-user applications. Second, when multiple networked applications are deployed, they are bound to influence each other, not only at the network level (which is easily simulated by, for example, NS2), but also at application level. This influence will be demonstrated and detailed in following sections, but, in general, we can state that the integration of application-level induced stream alteration is not easily achieved using simulation tools.

7.1 General description of autonomous avatars

Our test-bed introduces autonomous avatars to simulate the behavior of actual users of an NVE application, loosely based on the observations of massively multiplayer on-line role playing games such as EverQuest, Dark Age of Camelot and virtual interactive communities such as There and Second Life.

Through observation of several gaming sessions, we noticed that it is characteristic for these types of massive environments that users spend a lot of their time exploring the environment, interacting with other users through several communication channels or exchanging information with computer-controlled characters. Some areas in the environment, such as spawning areas, locations of merchants or easy killing grounds, will prove to be more of interest to the average user than other areas. As a result, these areas will be more densely populated. In summary: to simulate the traffic of these kinds of applications, we have found that the behavior of our autonomous agents must fulfill several conditions:

- The users are distributed over all the areas of the environment.
- Some areas are more densely populated than other areas.
- Most of the time, the users are moving around in the environment.
- Sometimes users are inactive for a period of time.
- Not all users move at the same speed.

Due to the need to be able to simulate large numbers of agents on a limited number of computers, there is one additional constraint, namely that the behavior of the simulated users must be simple enough to limit the processing power needed for the calculations. Some other work has been done on the determination of ‘crowded’ areas in virtual environments, for which we refer to [Chittaro 04].

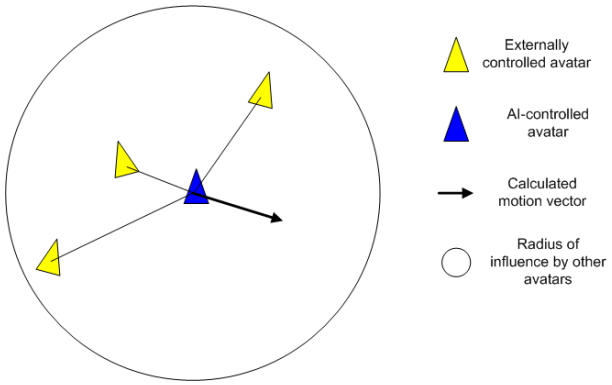


Figure 7.1: Separation behavior.

The implementation of the behaviors of the autonomous avatars is inspired by the work of Reynolds [Reynolds 87]. In his work, Reynolds implements flocking behavior of groups of virtual creatures such as birds or fish. Flocking behavior is implemented through the combination of three simple behaviors that are executed by every individual creature. The first behavior (7.1) is 'separation', which makes sure that some minimal distance is kept from other entities present in the flock. Cohesion is responsible for the movement towards the averaged position of a number of other entities in the vicinity. Finally, the alignment behavior ensures that the orientation of movement is kept (on average) in line with those of the closest other entities.

When programmed with some or all of these elementary behaviors, autonomous avatars can be made to move in a representative way, comparable to an average user-controlled avatars. Besides these behavioral constraints, it is vital for testing purposes that both autonomous and user-controlled avatars have an AOI which determines the regions from which they receive positional information of other avatars. In both cases, the selection of AOIs is based on the current position of the avatar by subscribing to the region associated with the current position and (possibly) all adjacent regions. For representative testing results, autonomous avatars should clearly also use the same network protocol and elementary state messages as user-controlled avatars.

7.2 Application of autonomous avatars for scalability testing

In our test setup, we have opted to include a subset of the flocking behaviors. On the one hand this enables us to to limit processing requirements, and on the other hand it provides a more life-like result, as including all flocking behaviors would result in a 'follow the leader' effect. However, all avatars in the world are controlled by the same, single, set of reactive behaviors. These behaviors control the avatar based on both internal and external events, resulting in a slightly different behavior of each avatar, depending on the environmental factors. Autonomous avatars, just like their human-controlled counterparts, are capable of receiving state updates from other avatars in their AOI. It is these updates that are used to calculate the behavioral moments. The events received from other avatars can be regarded as the sensors of the avatar. Besides these external events, internal events are used to generate some randomness in the behavior of individual avatars. These are triggered by using a random number generator.

In practice, we noticed that using a separation behavior to avoid collisions between the avatars in the environment yield satisfactory results in order to simulate (rudimentary) movements of large groups of participants. As explained before, this behavior uses the positions of nearby avatars, and continuously calculates a vector that points away from other entities in the vicinity (see figure 7.1).

Using only this separation behavior would have the avatars spreading out over (part of) the entire world seemingly without goal. This is clearly in contrast with the fact that some areas need to be more populated than others and that the entire world should be visited over time by all avatars (see earlier in this section for details). To make sure that this is simulated by the autonomous avatars, we have added a MoveTo behavior. Based on the setup of the entire world, a randomly selected target is defined, located in any of the regions that make up the virtual world. The MoveTo behavior generates a vector that is pointing towards one of these goals for a specified amount of time. The vector calculated by this behavior is subsequently added to the vector generated by the separation behavior, resulting in the true movement direction for the avatar. After a random time threshold has been exceeded, the MoveTo behavior will choose a new target in the list of potential goals to move towards. In case the final position has been reached before the threshold has been reached, the MoveTo behavior will remain inactive, with the avatar being controlled only by the separation behavior.

Because the target positions for the MoveTo behavior are chosen uniformly over the entire environment, the areas in the center of the world will be traversed more often than the outer regions. This ensures that these areas will be more crowded, thereby satisfying the conditions specified above.

Network traffic is generated by transmitting positional information, just like a human-controlled avatar would do when moving towards the target position. The information transmitted information includes both the position and the orientation of the avatar, with the latter set to match the current movement direction of the avatar. We should point out here that we do not utilize any of the optimizations discussed in the previous chapter (such as an event based state distribution scheme or dead-reckoning techniques), as we wish to determine a set of worst-case figures in terms of bandwidth consumption for a large scale virtual environment. In practice, one would certainly opt to include one or more of these optimizations into an event synchronization system, to make sure (as stated before) that movements are displayed in a smooth fashion or that simulation corrections do not introduce visual artifacts. In essence, these optimizations are clearly non-essential, and as the choice between these optimizations may vary along with the type of application that is to be deployed, we have opted to leave them out of the equation and present the reader with a worst case scenario.

7.3 Test setup description

In this section, the setup of the different hardware components used in the experiment will be discussed.

A total of 8 systems were used:

- 6 nodes of a cluster setup. 5 of these nodes contain a dual Intel Xeon processor running at 2.4 GHz with 2GB RAM. These nodes were used to run 180 agents each. One node is equipped with a single Xeon processor running at 2.4 GHz. This PC was used to run another 100 agents.
- 2 single processor PC's running at 1.7 GHz with 512 MB RAM. One of these PC's was used to run the master server and game server, and the other was used to run either a non-autonomous client to observe the world, or a single autonomous avatar. This PC was also used to capture the traffic of a single human-controlled or autonomous client.

All PCs were interconnected through a dedicated gigabit network. Network traffic was captured live using Ethereal (nowadays called 'Wireshark'

[Ethereal.com]). As it proved impossible to run the packet capturing software along with the 180 autonomous avatars on a single machine, a separate node was dedicated to running a single instance of the application and to make traffic measurements.

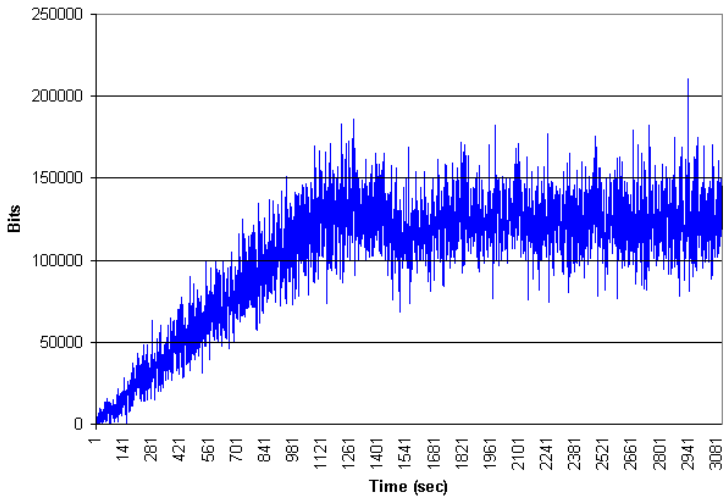
When an autonomous avatar is added to the environment, it spawns at a single specific location in the environment (the origin). If all avatars in the simulation would be added to the environment at the same time, they would all be added to the same region and multicast group, which would not be representative of true behavior, nullify (at least for some time) the measurements and increase the processing load to an extreme degree. Because of these reasons, agents are added to the environment gradually, allowing them time to move away from the spawning point. In this test setup, over a period of 20 minutes, 1000 agents were added to the environment.

7.4 Results of scalability testing

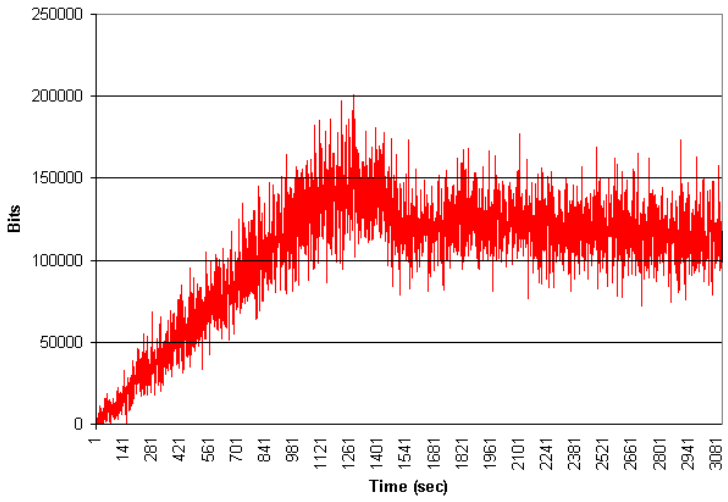
The following section presents the results of the captured traffic by the server, an autonomous avatar and a user-controlled avatar. This traffic includes all protocol overhead of the transmission, such as ethernet/IP/UDP/TCP headers. Traffic is summed every second and displayed in the charts.

7.4.1 Server traffic

Figure 7.2 shows the traffic that was transmitted and received by the server. This communication is used to request and distribute the addresses of multicast groups to clients when they move to different regions. The amount of sent and received traffic are almost equal. Capturing starts when the first client is added to the system. The amount of traffic gradually rises as more avatars join the environment and levels off when all 1000 avatars have joined. There is a slightly higher amount of traffic when an avatar initially joins the environment, because at that time it requests the addresses of all the multicast groups in its AOI. When the avatar moves around in the environment, only the addresses of the new multicast groups are requested. Note that the total amount of server traffic is low, considering the number of users that are present in the world, particularly when compared to pure client/server based systems.

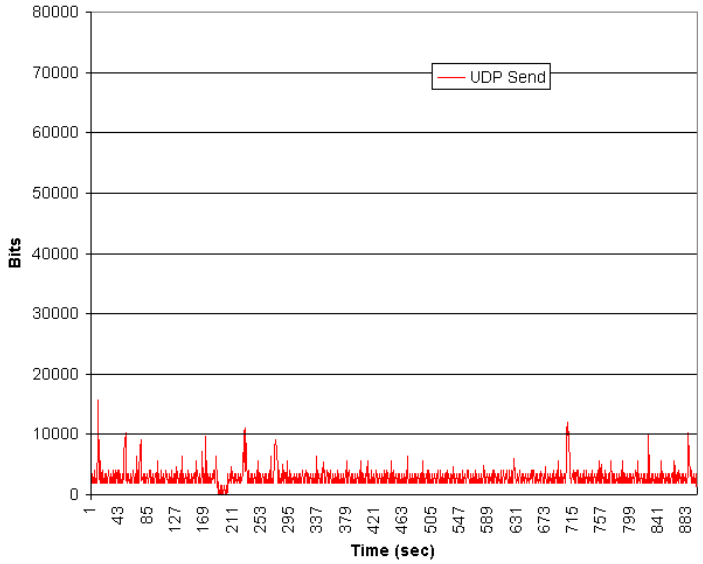


(a) Sent TCP Traffic.

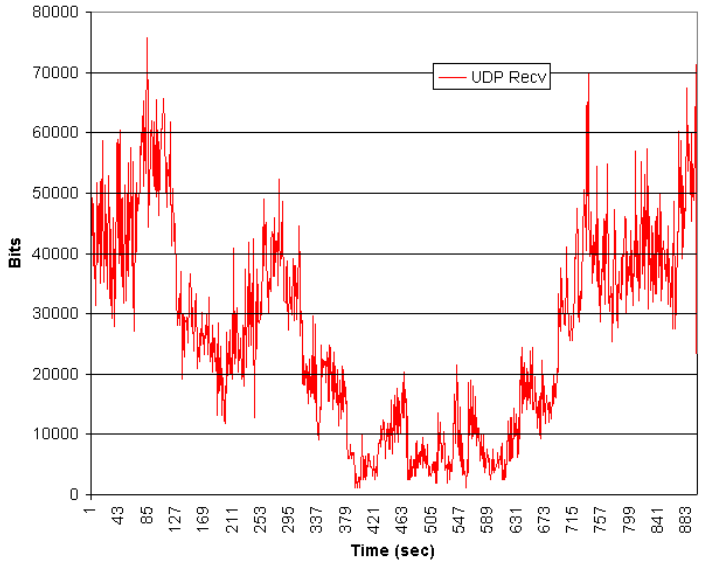


(b) Received TCP Traffic.

Figure 7.2: TCP traffic sent and received by the game server.

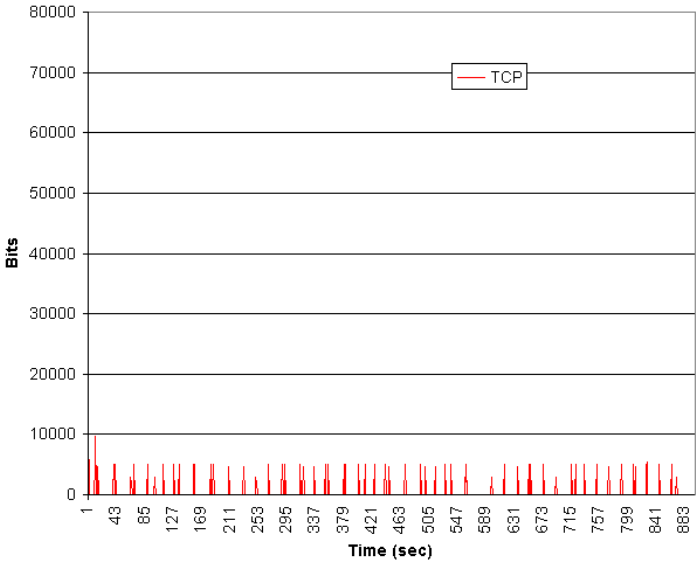


(a) Sent UDP Traffic.

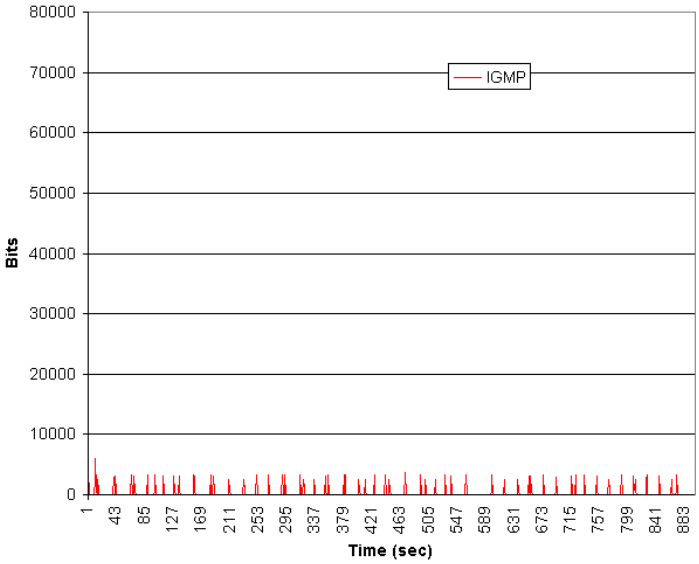


(b) Received UDP Traffic.

Figure 7.3: UDP traffic sent and received by an autonomous avatar with 9 regions in AOI.



(a) Sent and Received TCP Traffic.



(b) Sent and Received IGMP Traffic.

Figure 7.4: TCP and IGMP traffic sent and received by an autonomous avatar with 9 regions in AOI.

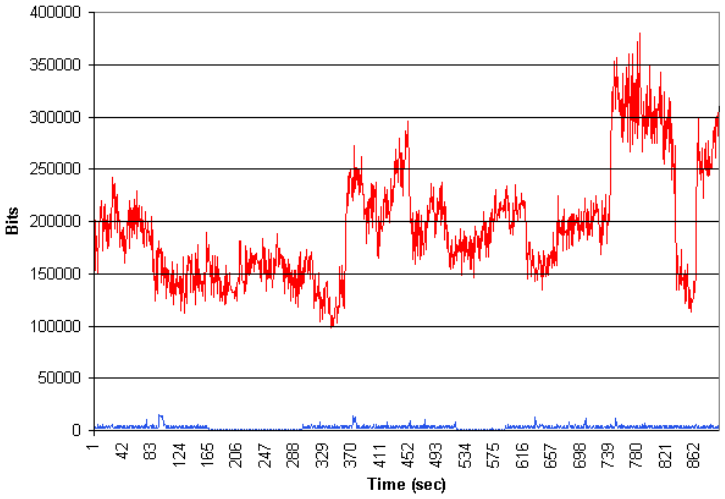
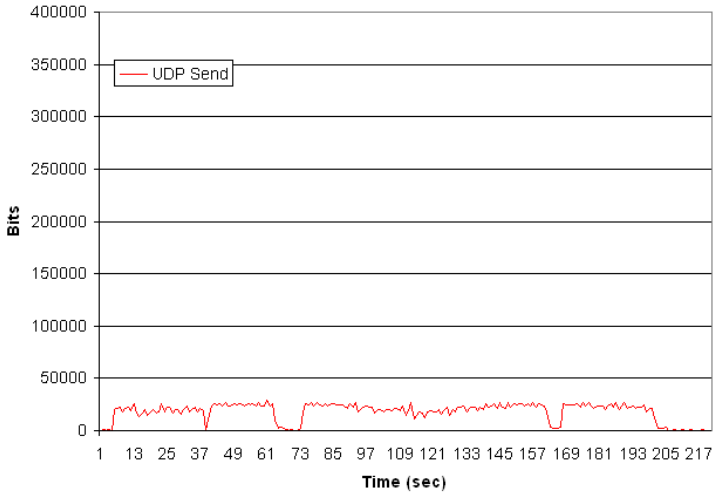


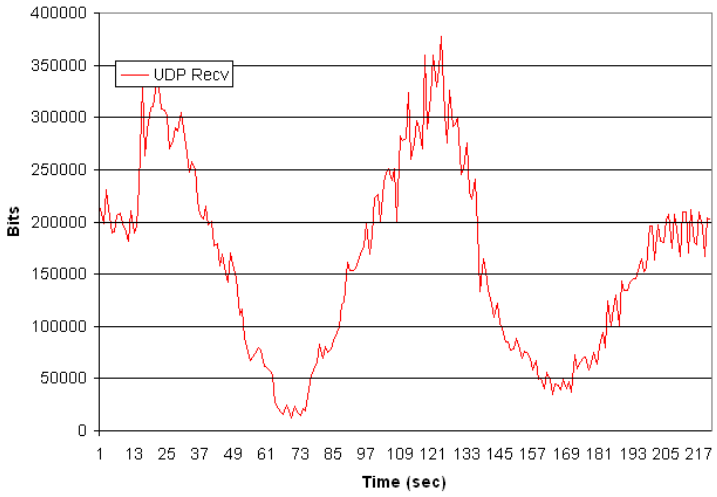
Figure 7.5: UDP traffic sent and received by an autonomous avatar with 25 regions in AOI.

7.4.2 Autonomous avatar traffic

Figure 7.3 shows the UDP traffic of an autonomous avatar while it is moving around in the environment. It is spawned in a world already containing 1000 avatars and consisting of 576 distinct regions. The areas are defined in such a way that a typical autonomous avatar with the pre-defined speed settings can traverse them in approximately 15 seconds. As explained before, most of the avatars tend to stay relatively close to the center of the world. The autonomous avatar here is configured to always have 9 regions in its area of interest, and therefore is always subscribed to 9 multicast groups (surrounding its own area). Fig 7.3(a) and Fig 7.3(b) show the sent and received UDP traffic, which is used to transmit positional data about the avatars. Again, it should be noted that optimizations such as dead reckoning are not used in this experiment. As a result, these positional updates are transmitted every time the agent moves, according to the update rate of the calculations of the behavior. When near other avatars, the update rate is increased to be able to avoid collisions. We should also point out that orientation changes are transmitted with a much higher frequency (and more precision) than simple movements, as a change in orientation is needed before the avatar can move in a new direction. Even when an agent doesn't move, it still sends a positional

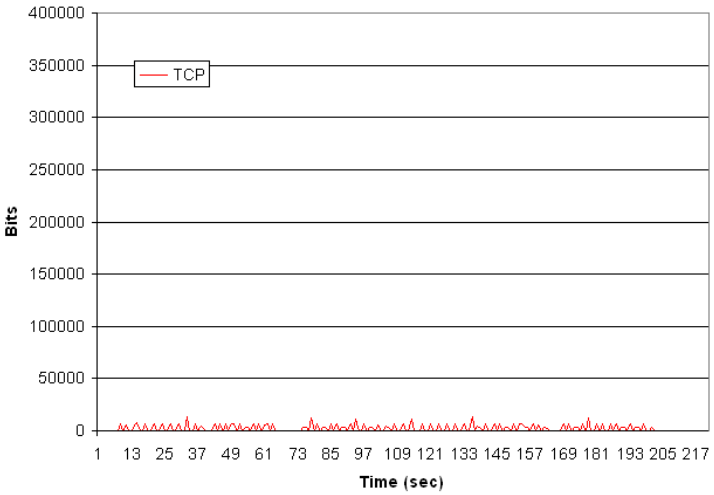


(a) Sent UDP Traffic.

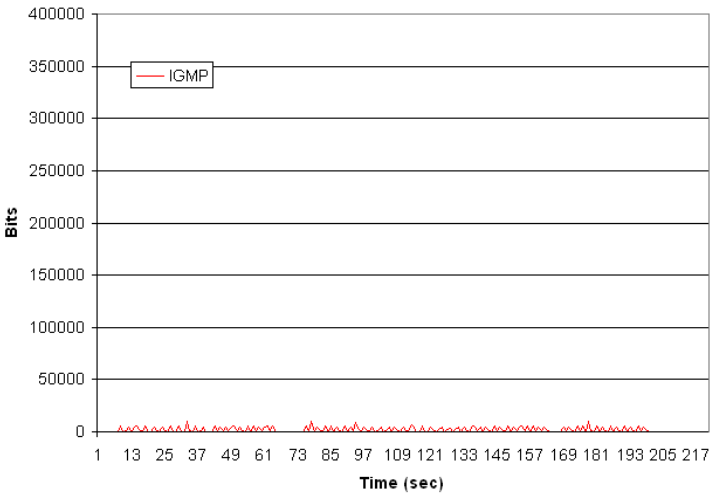


(b) Received UDP Traffic.

Figure 7.6: UDP traffic sent and received by a user-controlled avatar.



(a) Sent and Received TCP Traffic.



(b) Sent and Received IGMP Traffic.

Figure 7.7: TCP and IGMP traffic sent and received by a user-controlled avatar.

update every few seconds. Figure 7.3(b) shows the received updates of other avatars in the client's AOI. Initially, the client is located in a relatively busy part of the world near the spawning point, which results in a large amount of traffic. As the agent moves around, the traffic decreases and increases as the avatar joins and leaves multicast groups. Figure 7.3(a) shows the transmitted positional updates of the autonomous avatar. This traffic is relatively constant most of the time, but occasionally a spike occurs. This happens when the autonomous avatar comes near another avatar, and must maneuver to avoid this avatar (because of the separation behavior described in section 7.1). We already explained that the update rate is dependent on the relative distance to other avatars. Also, the separation behavior results in a lot of orientation information needing to be transmitted, thereby increasing the total amount of traffic required. After about three minutes, when the avatar has reached its random target, the traffic drops to near zero. At this time, only the periodic positional updates are transmitted. After some time, a new random target is selected and the avatar starts to move again. Figure 7.4(a) shows the sent and received TCP traffic, which is used to request the addresses of new multicast groups. When the new multicast addresses are received, the avatar joins these multicast groups using an IGMP message. This traffic is shown in figure 7.4(b). It can be observed that the amount of this traffic is relatively low compared to the traffic of the positional updates. This is useful for the scalability of the servers, as these only handle the TCP traffic. Figure 7.5 shows the UDP traffic that is sent and received by an autonomous avatar that is spawned in the same world but has 25 regions in its area of interest. Note that the amount of traffic transmitted is the same as in the case with 9 regions in AOI, but received traffic is higher than the previous case. It is therefore clear that the downlink traffic can be throttled by adjusting the extent of the AOI.

7.4.3 Human-controlled avatar traffic

Figures 7.6 and 7.7 show the traffic captured by a non-autonomous avatar. This avatar is spawned at approximately the same time as the autonomous avatar described above, and under the same world conditions. The traffic is therefore expected to be comparable to the traffic of the autonomous avatar. The user-controlled avatar has 25 regions in its AOI and therefore is subscribed to 25 multicast groups, comparable to the setup of the autonomous avatar with 25 regions in its AOI (see Fig 7.5). This results in an almost equal amount of received positional updates (mean value), see fig 7.6(b). The sent data (fig 7.6(a)) is higher in volume because of the speed of the user client. This

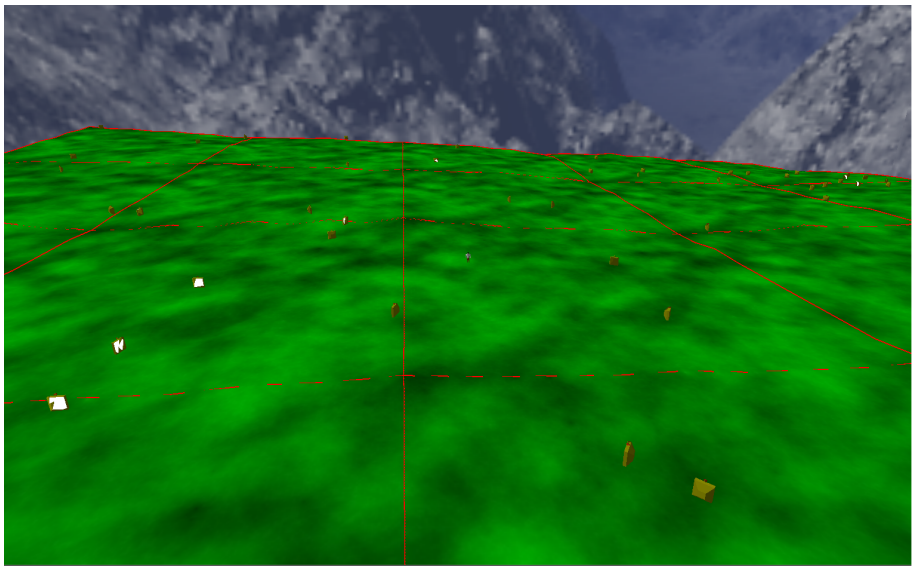


Figure 7.8: Screenshot of an active session with around 70 autonomous avatars.

avatar can move at considerably higher speeds throughout the world than the autonomous avatars. The results are faster changing of regions (more TCP/IGMP traffic) and more positional updates (sent UDP traffic).

Initially, the avatar remains stationary near the spawning point in the environment. After approximately 200 seconds, the avatar starts moving towards one of the outer regions of the environment that is sparsely populated by autonomous avatars. As a result, the amount of received positional updates, shown in figure 7.6(b), drops drastically. Subsequently, the avatar moves towards the other outer end of the environment (passing through the densely populated center), showing an increase and again a drop in the amount of received positional data. The amount of TCP and IGMP traffic, shown in Fig 7.7(a) and Fig 7.7(b), is comparable to the traffic of the autonomous agent. A screenshot of the application with around 70 agents and 25 regions in a user's AOI can be seen in figure 7.8, while a 2D overview is shown in figure 7.9 .

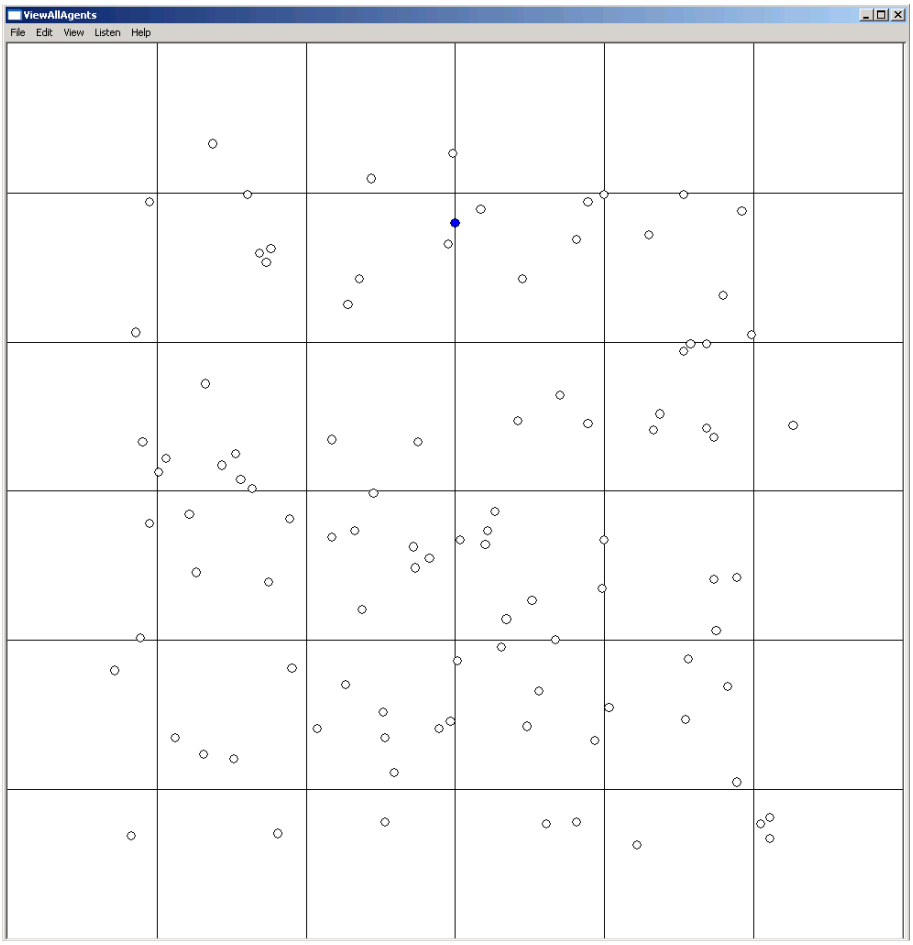


Figure 7.9: 2D overview of an active session.

Chapter 8

Discussion - Part II

In this part we have introduced the design of the architecture that is used as the common basis for the work described in subsequent parts of this dissertation. The ALVIC architecture focusses heavily on the use of multicast as primary distribution mechanism in order to achieve scalability. Through a spatial subdivision system that is tightly coupled to the distribution mechanism (by mapping virtual world space onto multicast groups), the vast virtual world is split into manageable chunks, both in terms of bandwidth usage and number of clients. Additionally, through an area of interest management system based on this spatial subdivision methodology, efficient client-controlled bandwidth throttling can easily be achieved.

Scalability of ALVIC has been demonstrated using a custom-made test setup using autonomous avatars that effectively simulates the behavior of large groups of human users. By deploying the simulation onto a number of interconnected computers of a commodity PC cluster, we were able to attest that the system lives up to the requirements that were set, both in terms of bandwidth adaptability as well as in scalability with regards to server load. In practice, we have shown that it is clearly feasible to support at least one thousand simultaneous users on a single server, with the total amount of clients limited only by the availability of multicast addresses and processing capacity of the cluster on which the setup was deployed.

In the next part, we will look at some extensions to the basic ALVIC architecture, more in particular towards support for multimedia streams.

Part III

Communication

Introduction

Up until now, the only data exchanged between clients that has been considered was state information, e.g. movement updates, interaction information and control data. Besides these essential ingredients, a modern NVE framework cannot really be considered state-of-the-art without the addition of some multimedia elements.

In the context of the applications considered here (mainly virtual interactive community-related), it seems natural to focus on the addition of sound and video transmission. Both are at the same time similar and dissimilar from data flows studied before. On the one hand, at the network level data needs to be exchanged in a way very much comparable to state information, but the absolute amounts of data needed to be channeled are an order of magnitude larger.

In this chapter, we will first take a look at the impact of the addition of audio and video on applications in general. After this general discussion, the addition of these multimedia capabilities to the ALVIC framework is examined in detail, as well as several optimizations to manage bandwidth usage at client-side. To prove that the proposed solution does indeed scale as claimed, the test setup as described in the previous part is adapted to include the additional features required for video transmission.

Chapter 9

Audio/video communication in NVEs

Early examples of NVE-like applications supported chat functionality through text messages. A well-known example is DOOM and its 'console mode' where players were able to enter text messages that would appear on the other participants' screen during gameplay. This was afterwards extended in games such as half-life to the concept of 'taunts', which were in fact no more than simple identifiers sent over the network that triggered remote sound playback events. None of these scenario's required the real-time transmission of multimedia information over the network. This can be attributed to a number of reasons, first of all the processing power available to the end-user was consumed entirely by the game rendering engine, due to the lack of dedicated graphics hardware. On the other hand, real-time voice transmission (VOIP), as we said before, consumes a lot more bandwidth than simple state update mechanisms, which simply wasn't in abundance at the time these games were developed.

9.1 Current-generation implementations

A majority of the current virtual interactive community applications, such as There, as well as several on-line FPS games support voice communications either natively or through a third-party add-on. Others, like Second Life, have announced the integration of audio as a feature to be added in the near future. However, it has not always been like that. In the launch year of Microsofts Xbox Live architecture the voice chat feature was being promoted

as revolutionary and dedicated hardware - a headset that plugged into the controller) was provided for precisely this purpose. Most people thought up until then that it would be prohibitively expensive, both in terms of network capacity and processing power, to use multi-party voice chat on a game console during gameplay. Although a lot of issues were apparent in the first version of the Live architecture (jittery playback, problematic routing of voice traffic,...), it was a feature clearly of interest to the community. Furthermore, not all issues were of a technical nature, as some people abused the system by swearing or spreading undesirable content. Clearly, Microsoft had foreseen these issues and was able to ban these malicious users from communicating with others by disabling their voice capabilities remotely.

9.1.1 Audio

One cannot discuss the use of real-time audio without dedicating some space to the best-known VOIP application currently in existence: Skype. Developed by the same people that brought you 'Kazaa', Skype was one of the first free applications to transmit Voice over IP over the Internet. Some other options existed, but they were either cumbersome to install, requiring difficult reconfiguring of NAT routers and firewalls, or were purely client/server based and required monthly subscription fees. Using a custom made protocol and architecture, Skype is able to run on networks behind a variety of firewalls, without requiring the user to configure advanced settings. It achieves this by dynamically routing the voice traffic through a number of supernodes, which are in fact ordinary users that have the Skype application running and are not behind a strict firewall. The fact that the protocol is not standardized and the complete absence of information on the encryption said to be used by the developers has not prevented Skype from becoming the market leader in free VOIP transmissions. Nowadays, the Skype network offers advanced features such as SkypeOut, which enables users to make phone calls to ordinary PSTN connected telephones, and SkypeIn, which assigns a real phone number to a Skype ID for other people to call using the PSTN network. Skype is not directly used in any of the currently popular VIC applications, but can be indirectly linked by web services that, for example, display presence status for avatars in the virtual world.

Another application that is effectively being used by a large amount of players of current-generation on-line games is the TeamSpeak application, developed by TeamSpeak systems. This platform-independent application is free for non-commercial use and consists of a server and client module. Server

capacity can either be rented from commercial vendors, or can be installed locally, for example for use in LAN parties. What sets TeamSpeak apart from the rest of the VOIP offerings is that it can be integrated as a plug-in into several popular games (such as BF1942, CounterStrike etc).

The Vivox system, which will be used as technology platform for the upcoming support of voice chat in Second Life, is said to be specifically designed with the application of virtual worlds in mind. Besides voice communication, other means of communication are supported such as presence management, video and instant messaging. Vivox provides an API for software developers to interface with the supporting server infrastructure, which is maintained and provided by Vivox themselves (and is touted as being immensely scalable, although no further information is provided).

Several virtual interactive communities have their own built-in support for voice chat, but as little details are known about the distribution method, we will not discuss these any further.

9.1.2 Video

Video support is a feature currently supported by few VIC applications, and even less so for games. While one-on-one chat is supported through instant messenger applications such as Microsoft MSN messenger and Apple iChat, conferencing between multiple parties is a feature available only in commercial (and feature-dedicated) software.

Video communication in virtual environments nonetheless provides interesting alternatives to more 'traditional' means of chatting. It is, for example, much easier and certainly more natural to convey emotions through a video image than, for example, by using emoticons. VIC applications such as Second Life provide means to indicate emotional state by altering the facial expressions of the avatars, but this is a labor-intensive task for content creators, as the wide range of emotions that can be expressed by the human face need to be geometrically modeled. Displaying life-like emotions on a computer-generated model has proven to be extremely difficult because of the high number of variables that need to be tweaked. Using a video image instead of these artificial models has the potential for solving these problems, the catch being the high bandwidth requirements for transmitting multi-party video streams.

In a few research projects, limited video support was integrated in the architecture. For example, in the Virtual Life Network (VLNET [Joslin 00]), video is integrated to complement the other multimedia streams such as audio and object animation information. Face information is extracted to super-

impose on a virtual head model. The VLNET was deployed on an ATM network linking several universities, so scalability was really of interest to the developers. In fact, even some of the earliest examples, such as NPSNET IV [Macedonia 95b] and versions of the DIVE architecture [Frécon 98] included rudimentary forms of video integration (although again not scalable in any way).

9.2 Problems associated with video transmission

To gain a better insight into the intricacies of video transmission, one needs to compare video stream traffic to the other data streams present in a virtual environment. This will be done by comparing actual figures with the results obtained in part I of this text.

9.2.1 Bandwidth usage explosion

To reiterate some relevant results from part I, we have shown that downstream traffic for a classic MMORPG game like EverQuest averaged about 10kbps. This includes the state information for a number of other avatars and all control data required to keep the applications running. More recent examples like PlanetSide had a total downstream traffic of about 40kbps. The Xbox Live game of MotoGP with voice support consumed an average of 70kbps.

While several parameters can be tweaked and adjusted, it is generally known that, using a state-of-the-art codec like MPEG4 or H263, at least 100kbps is necessary to transmit a relatively decent quality CIF video stream over a network (not including protocol overhead). As a side note: the most common video frame sizes are derived from standard television resolutions (i.e. in the 4:3 aspect ratio), and are chosen so they can be converted easily to PAL or NTSC size. Commonly used are CIF (352x288) and QCIF (176x144). For higher quality levels, but still comparable to video conferencing quality, the bit rate can increase to around 300 kpbs. A discussion can be found in [Kuhne 99]. It should immediately be clear that these figures for a single stream are disproportionally large when compared to the total amount of traffic needed to synchronize an entire virtual environment at client-side. The situation becomes even worse when multiple video streams are displayed simultaneously. It is therefore a non-trivial task to design an architecture that supports a scalable distribution of these video streams to and from a large amount of clients.

Res.	FPS	Mean Encoding Time (ms)	bit rate	Mean Decoding Time 1 (ms)
CIF	25	4.2	110	3.25
CIF	15	5.38	50	3.31
QCIF	25	2.92	90	0.92
QCIF	15	1.46	25	1.07
Total		13.96	275	8.55

Table 9.1: H. 263 video timings and measurements on a 1.7 GHz system.

Res.	FPS	Mean Encoding Time (ms)	bit rate	Mean Decoding Time 1 (ms)
CIF	25	3.15	110	2.29
CIF	15	3.5	50	2.21
QCIF	25	2.12	90	0.73
QCIF	15	1.37	25	0.67
Total		13.96	10.14	275

Table 9.2: H. 263 video timings and measurements on a 2.3 GHz system.

In tables 9.1 and 9.2, we provide some actual figures on encoding times for real-time video, associated bit rates for decent quality playback and mean decoding times. These results were achieved using a 1,7GHz(1) and a 2.3GHz(2) system.

9.2.2 Prioritization of data flows / QoE and QoS

Video communication is certainly an interesting feature to add to any on-line experience, but not when it causes the actual gameplay or feeling of interactivity to degrade. It is therefore of vital importance that prioritization is added to such as system, in fact thereby providing some manner of Quality of Service control and influencing the subjective Quality of Experience of the end-user.

In the Xbox live architecture, this QoS provisioning is essentially achieved by probing the bandwidth availability between peers in the system and/or between the consoles and the server infrastructure. In case the probing process reveals that sufficient bandwidth capacity is available for voice communication, the feature is enabled for a specific console; in case the link capacity is barely capable of keeping the game state in sync, the feature will be remotely disabled. While this binary decision on whether to enable or disable voice communication is definitely a bold one to make, it has shown to be beneficial to the end-user in the sense that the gaming experience itself is not degraded

by having a lot of extra features in the architecture.

The concept of this prioritization is an important one to take into consideration when upgrading our architecture with video capabilities. However, a binary system where video is either supported or not is not recommended, as it might exclude some participants from joining in the experience because their throughput specifications are at the borderline of the proposed system requirements .

Chapter 10

Extension of ALVIC for video

In this chapter, we will discuss the practical implementation of video transport capabilities in ALVIC (described in the previous part). The usage context is explained, as well as several real-world test results that prove the scalability of the solution.

10.1 Description of the video avatar concept

Various ways to integrate video into an NVE-like application can be envisioned. For example, in [Insley 97], a video recording is made of a person turning 360 degrees in front of a camera setup. Each time the avatar needs to be displayed in virtual reality, a set of two images is selected from the obtained sequence. The use of two distinct images is needed for stereo visualization in the CAVE environment. Of course, the images being displayed are static, and only their position in the scene is subject to change. The authors of [Ogi 00] also use an immersive display technology, and use the positional data gathered from an electromagnetic tracking device to place the avatar in virtual space. Stereo-image video cameras are placed in the corners of the immersive display setup to capturemoving images of the user together with depth information, in sync with the captured motion information. All data is subsequently transmitted to the other clients, which choose those frames from the sequence that were captured with the stereo camera that matches best with the position of the avatar from their viewpoint. While the system yields good results, it is only applicable in small-scale deployment. In [Yura 99], (full body) video

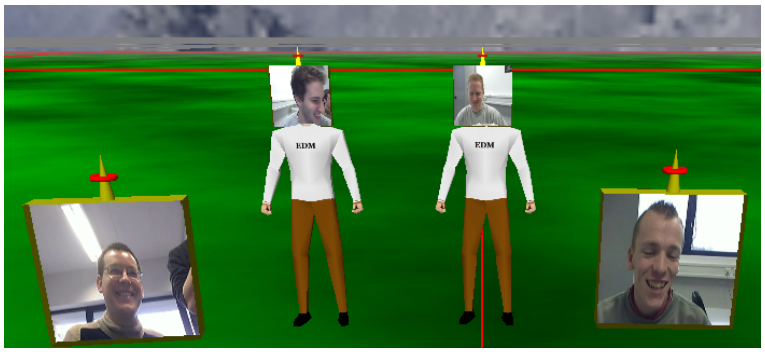


Figure 10.1: Video avatars.

captured from a camera is applied to a three-dimensional mesh of a human figure. Some enhancements are made to provide additional features beside pure video (comments and gestures). The system described is used mainly for guiding users through vast virtual spaces. Finally, [Rajan 02] uses an offline reconstructed head model to project real-time video onto. Again, the setup used is an immersive display setup and obtains positional data from tracking devices. Segmentation of the video images is facilitated as only information on the user's head is relevant. As the system was designed for use in a controlled local area network environment, no encoding and/or decoding of video sequences was needed, optimizing the quality obtained. Other examples of related work are presented in [Liu 01] and [Wang 97].

The technique we implemented in ALVIC will be referred to as the video avatar, in analogy with the related work described above. Figure 10.1 shows an example usage scenario. Unlike other systems, where the video streams are displayed in other windows or in a separate part of the GUI, we have opted to integrate them into the 3D environment. They can, furthermore, either be used as an independent avatar form or coupled to a 3D mesh model, integrating the video frames as textures on specific surfaces. The concept of video avatars is especially suitable when considering multiple concurrent video streams active in a single viewpoint, as displaying them separately (in an independent window for example) would distract from the actual interactions in the environment. The application of the video image as a texture on an existing avatar form enables a natural way of conveying emotions, which was referred to in the previous chapter.

10.2 Extension of existing architecture

To provide an overview for the following sections, we will start by summarizing our proposed solution without going into great detail. Once the general idea behind the solution is made clear, a detailed description of the impact on the various elements of the architecture will be provided.

10.2.1 Overview

We should first point out that the concepts described in this subsection are not meant to be fully understood at first glance. Rather, we will come back to each of them individually in a subsequent section of this chapter. To create a better understanding of now how the pieces fit together however, it is essential to give a high-level overview at the start of discussion.

To enable scalable transmission of video streams, we envisage a system that does not impact the existing architecture for the distribution of state information. The extensions are in fact by large analogous to the way in which the other information is transmitted. The virtual world is again divided into a number of distinct regions, each with one or more multicast addresses associated with them. These addresses are distinct from the addresses used to transmit state information, to enable any client to perform a simple form of QoS: distinguishing essential from non-essential information (such as video streams). Scalability in our setup follows from the definition of a number of multicast groups that are associated with a single region in the world. Each of these groups has a pre-defined quality setting associated with it. Examples of these qualities are provided in table 10.1. Each client is responsible for sending its video data to each multicast region with the quality parameters as defined for that multicast region. If one or more of the required quality streams are unavailable (due to, for example, lower quality input streams), a lower quality stream is sent to that specific multicast group. All packets sent within a specific multicast region are tagged with a quality parameter that defines the quality of the video stream that is contained within the packets. The lowest-quality setting will most likely consist of a single still frame that is retransmitted every few seconds.

10.2.2 Client-side - video area of interest determination

We have already discussed, in the previous part, the concept of the AOI as it is being used in the architecture. By altering this AOI, a client is independently able to adapt the incoming flow of data to ‘fit’ in the available bandwidth

Quality	Resolution	FPS	Bitrate
High	CIF	20	110000
Medium	QCIF	15	80000
Low	SQCIF	10	30000
Minimal	SQCIF	5	15000

Table 10.1: Sample video quality parameters.

capacity. It should be obvious from the discussion above, that a similar concept is necessary for video transmissions, as these too are associated with separate multicast addresses. Because of the similarity between the two, we will refer to this as the Video Area of Interest (VAOI). In its simplest form and if bandwidth capacity is not an issue, the VAOI will coincide with the AOI. It is however very unlikely that sufficient bandwidth is available to display large amounts of video streams concurrently. Even besides this technical issue, it would be pointless to try and visualize video streams of avatars that are located at the edge of the view frustum, as this will not attribute to the recognizability. Through the adaptation of the VAOI size, we achieve the much-desired non-binary QoS system, in which users are independently able to throttle the bandwidth usage of the application.

We should point out at this time that it is certainly not necessary for the division of the world into regions, originally designed for the distribution of state events, to be copied and be used for video transmission purposed. In fact, it would be beneficial to the end-user if the regions used for video transmission (and associated multicast groups) would be defined so as to contain less entities than the ones used for state distribution. This would enable a much more fine-grained selection of the streams to displayed and even further enhance the quality of experience.

10.2.3 Client-side - quality selection strategies

A client is also able to dynamically alter the video stream quality of a region already in its VAOI, depending on, for example, the distance between the user and that region. Switching between quality levels is achieved simply by joining the multicast group that contains the lower quality video streams of a specified region in the world. (see figure 10.2). Determination of the VAOI can be done according to a number of factors. A client that has large downstream capacity may choose a VAOI like Fig 10.2.A, in which all video in the subscribed regions is streamed in the highest quality setting. Users that

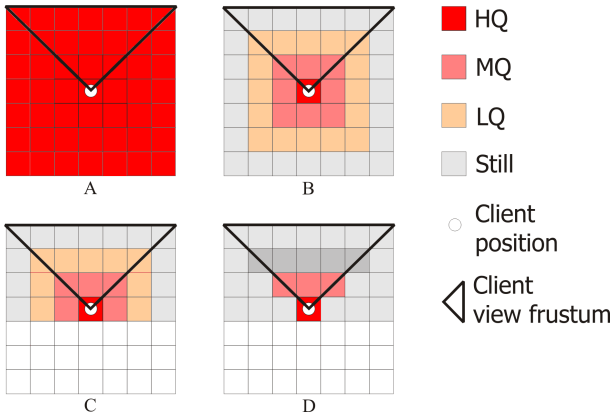


Figure 10.2: Video area of interest selection.

have a more limited throughput capacity at their disposal might adopt another selection mechanism, where the quality is gradually degraded for those regions located further away from the viewpoint. This is visualized in figure 10.2.B. In practice, switching between multicast groups is a process that is associated with a certain delay factor, because of the way the IGMP protocol interacts with the routers in a WAN. However, when fast switching of multicast groups is possible, such as in LAN environments, the quality selection strategy may be even further optimized as in figure 10.2.C, where regions located behind the avatar are not subscribed to. When taking into account the actual view frustum, the strategy depicted in Fig 10.2.D may be used to achieve near-optimal use of downstream bandwidth.

10.2.4 Server-side responsibilities and impact

The servers in the proposed architecture need not perform many additional tasks compared to the non-video setup. The required extra functionality can therefore be integrated into the same servers as in the non-video based setup. When a client connects to a game server, it is assigned a multicast group address for sending its data, depending on its starting position in the virtual world. When changing position or expanding/extending the VAOI, clients request the multicast addresses of the regions concerned from the appropriate (assigned) game server.

It should be clear that the additional impact of having to hand out multicast addresses will not increase the processing/network load on the game server by a large factor. Scalability can be demonstrated in the same way as

with thenon-video based setup, using a setup that simulates avatar movement and video stream distribution. Server capacity will be clearly be affected in a linear fashion, and given the fact that the standard server setup scales easily to 1000+ users on a single machine, it is unlikely that the additional tasks presented here will have a major negative impact on server scalability.

10.3 Identified issues and optimizations

While we will present actual scalability test results in the next chapter, a number of issues can be readily identified without testing in real-life. Some of them are discussed here, along with a few possible optimizations and interesting additions.

10.3.1 Issue: client-side processing and network scalability

In theory, it is perfectly possible for any type of codec to be used for a specific quality setting. This way, one may select the optimal combination of bit rate and frame rate / frame size parameters that is achievable in highest quality using a specific codec. However, a use of multiple codecs would increase the memory footprint of the application and may introduce extra processing load due to the required initialization of the required framework. In practice, we have had good results using codec frameworks such as FFMPEG or Microsoft's DirectShow ©, as they support multiple codecs through a single interface and are optimized to make use of hardware extensions, available in modern CPUs. Using such a system certainly speeds up the required processing to encode streams in a number of different qualities.

Possible problems regarding client-side processing are largely due to the fact that the outgoing video stream has to be encoded multiple times, depending on the required quality levels. Should processing power become a bottleneck however, a lower-quality stream may be sent to a higher-quality defined multicast group, as long as the codec remains the same. Provisions are often made in decoders to be able to handle incoming video streams of different sizes. Decoding a video frame from a stream typically takes less time than encoding, but in case this should become problematic due to the high number of streams, shrinking the VAOI size provides an easy way out.

Of course, the same that was stated for processing power is also valid for bandwidth requirements, as multiple streams will need to be uploaded to the appropriate multicast groups through the (often) limited uplink channel. We

will show in the next chapter however that it is in fact feasible to do this for at least 3 different video qualities using a 'standard' asymmetric DSL connection.

10.3.2 Issue: deployment on non-shared-medium networks

As we stated many times before, allowing individual clients to multicast large amounts of data is a policy that is seldom adopted by ISP's at this time, mainly due to possible explosive growth of bandwidth usage. Specifically in the case of xDSL networks (and contrary to cable networks), the access network is not a shared-medium network, and traffic effectively needs to be duplicated on individual lines to reach groups of end-users. It is therefore interesting to see what provisions can be made in these types of access networks to support the type of application under discussion.

Generally, in an xDSL network, the DSLAM (Digital Subscriber Line Access Multiplexer) is located located at the edge of the access network. We propose the introduction of video-servers at DSLAM level that enable unicast to multicast conversion. Being located where it is in the network topology, the DSLAM is the first location in the hierarchy where 'intelligence' can be added in the form of processing nodes.

Each client that wishes to send data to a multicast group unicasts the data to the dedicated video server. F1 in Fig 10.3 shows the video stream on the client's private point-to-point connection, while F2 denotes the stream on the shared network at DSLAM level. The server's responsibility in turn is to multicast this data to the desired multicast group (F3 in Fig 10.3). As this server can be located in the xDSL infrastructure itself it is very likely that this kind of multicasting will be allowed at ISP level. Multicasting at DSLAM level (mostly ATM) is currently employed for broadcast quality video stream distribution for digital interactive television. With IP-based DSLAM's, support for other (third party) applications is very likely to be enabled, although probably still limited to the backbone network.

In case positioning of servers at the edge of the multicast-enabled network is not feasible, for example for small-scale deployments, servers may be co-located in an ISP data center. Clients would still be able to communicate directly with these servers through unicast connections, with the server in turn distributing the streams in a peer-to-peer fashion. One might envision the use of transcoding to be helpful, as only a single stream would have to be exchanged between servers. Once a client requests a data stream with specific parameters, the server would be responsible for transcoding the stream on-the-fly. Given the fact that transcoding is a very processor-intensive task however,

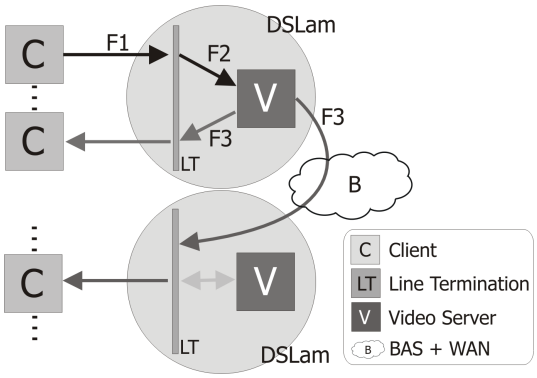


Figure 10.3: Video servers in the access network.

such a system will not scale easily to hundreds of users. We will provide a better solution in section 10.4.

10.3.3 Optimization : use of the CastGate project

During the course of the project in which the research was conducted, a test bed was deployed between three sites: two that were directly connected to the multicast-enabled BELNET network and one through an ADSL link. The configuration is shown in figure 10.4. As native multicasting was not available between the internal network of the three sites and the backbone, a remedy was found in the application of the CastGate project[CastGate]. The architecture of CastGate consists of a software router service that is to be installed on the internal network segment that is to be connected to a multicast-enabled backbone network. The CastGate tunnel server is located in the backbone network itself. The CastGate projects enables transmission of multicast traffic through a unicast connection with this tunnel server.

With the same video parameter setup as will be described in section 11.3.1, we were successfully able to interconnect the two sites that had a direct uplink to the backbone network. For the ADSL connect site, the codec parameters had to be altered so that less bandwidth was required for the uplink stream, so as not to choke the downstream channel. After these alterations were made, all three sites were successfully able to interconnect and exchange video streams.

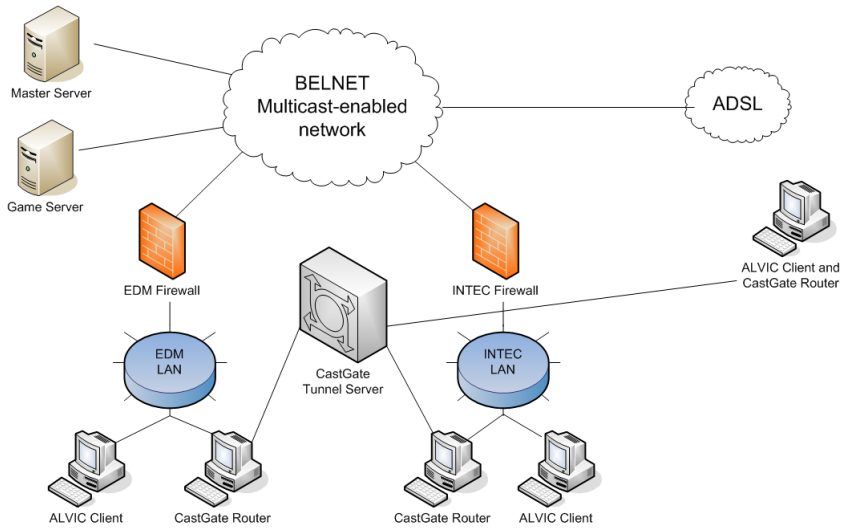


Figure 10.4: CastGate test case setup.

10.3.4 Optimization: application of scalable codecs

At first sight, the methodology used in the proposed architecture may seem incompatible with scalable video codecs, or at the least make them superfluous. In fact, exactly the opposite is the case, as the two technologies can be perfectly combined. If it were not for the lack of available practical implementations of these codecs, these would have improved the scalability test results (to be discussed in chapter 11) even further.

We will make a short digression to explain the general principles behind scalable coding; for a more detailed explanation, we refer to e.g. [Cohen 00] and [Horn 99]. The enhanced coding technique was developed to stream the highest possible video quality to a multitude of users, each with their own requirements and capabilities. Obviously, doing so requires an encoder that is designed specifically to take into account the fact that real-time processing by the streaming server should be kept to a minimum to be able to output as many streams as possible (the encoding is separated from the streaming process). The encoder is instructed to encode video with a target range of bandwidth figures in mind, such that the streaming component is able to quickly adapt the information provided by the encoder to multiple output streams. At the same time, the decoding process of such a scalable video stream should be of minimal complexity, to allow devices with low processing power to deliver real-time output. This is achieved by having the encoder split

the output into (at least) two components: a base layer that contains basic information needed for all output qualities, and (possibly several) enhancement layers that, when combined with information of the base layer, provide higher quality output. By dynamically switching between enhancement layers, the decoding process can at the same time adapt the stream to the desired quality level and preserve, possible scarce, resources, as the process of combining base layer and enhancement layer(s) is a relatively straightforward operation. Note that, from the conception, this process was designed with streaming over multicast-enabled networks in mind, which we will illustrate further on in this section

In general, when considering for example MPEG-4 FGS [Radha 01] or scalable streams as defined in H.263 Annex O [ITU.T], selection of the desired output quality is performed by the receiving client by discarding the scaling information that is irrelevant to the specified quality setting. At that time however, the harm has been done in terms of bandwidth usage, as the total required link capacity in either uplink or downlink direction is not affected by that client's selection, when used in a pure peer-to-peer way. In our proposed system, incoming bandwidth is continuously changing as new regions are entered/left and as the number of (video) avatars in the subscribed regions changes. To throttle the bandwidth in downstream direction it suffices to adjust either the size of the VAOI or to switch to lower quality groups for some or all regions. Upstream bandwidth of any given client is never influenced by the number of other users that have the client in their VAOI.

To combine the quality selection mechanisms in our framework with the capabilities of scalable codecs, we would have to separate the basic information needed for the lowest quality setting from the higher detail levels. In turn, each of these 'parts' that make up the scalable video stream would be sent to a different multicast group. By subscribing to either the most basic quality group or combining this with the information of the group that contains the scalability information related to the desired quality level, an increased reduction of bandwidth is achieved. The added advantage of this solution versus the uploading of 3 distinct video streams is that there is no need for duplication of the basic quality level information.

10.3.5 Optimization: broadcast video

Broadcasting video in NVE applications presents a new opportunity for TV stations and information providers in general to distribute their programs and information. Sharing the experience of watching a TV show, with the

added benefit of direct interaction between viewers may increase the number of viewers (and associated revenues). Encouraging comments from national TV stations that witnessed actual demonstrations of this feature has convinced us that this is no longer a far-fetched idea but may become reality in a relatively short time frame.

Broadcast-quality video streams, compressed with MPEG2 codec, consume several megabits per second of bandwidth. In context of NVE applications, it will not always be necessary for viewers to receive a stream in the greatest detail, as the screen space of the broadcast transmission will probably be relatively low when compared to a dedicated application. These streams are therefor ideal candidates to be offered in several quality levels and distributed the same way as client-to-client communication streams. Servers that distribute the streams in the proposed architecture are ideally situated at ISP level, and as such are highly likely to be able to take advantage of multicast capabilities. It is only at the edge of the access network that traffic needs to be unicast to interested parties. However, as the same is currently being done for distributing digital TV over DSL links, this is not a practical issue anymore.

10.4 Integration of a supporting proxy infrastructure

As we will show in the next chapter, it is feasible to deploy the ALVIC framework with video on connections with typical amounts of available bandwidth. However, several ideas were formulated to further reduce the required bandwidth in upstream direction. The initial suggestion was that dedicated video servers had to be introduced into the network that would accept single high-quality streams coming from clients. These streams would subsequently be re-encoded in a number of lower-quality streams and be multicast on the network. While this preliminary solution would certainly work, the concept was later extended into a more generic proxy architecture that complements the framework.

The first application of the proxy architecture was to be a multicast-to-unicast conversion, enabling sites with no multicast capabilities to use the ALVIC framework. As it resides in the backbone network (preferably in the access network, close to the end-users), the proxy is able to accept incoming unicast transmissions and distribute them over the backbone network. It differs from the CastGate solution as it does not 'eavesdrop' on the local network to trace multicast traffic, instead, the proxy is directly adressed by applications

that require its' services.

Secondly, the proxy is equipped with network and application intelligence. In practice, this means that decisions on bandwidth management are made by the proxy, but in cooperation with the client. The proxy is able to dynamically adapt the bandwidth usage of traffic going to each individual end-user by taking intelligent decisions based on application-level) knowledge. For example, by taking into account the virtual position of the avatars in the world, video streams are either forwarded in a specific quality level or blocked.

The software architecture of the proxy is built such that it is easy to plug in extra functionality. In a later chapter, we will look at its possible application for the mobile scenario (see 14.2.3 and 14.1.2). The general setup of the proxy is described in [Wijnants 05a] and [Monsieurs 05]. For a more generic discussion on proxies and their applications, we refer to [Amir 95], [Shen 04] and [Lei 03].

Chapter 11

Evaluation

In this chapter, we will look at the scalability of the proposed solution using real-world data. The methodology used is similar to the one used for the basic architecture, and uses autonomous avatars to generate a lifelike environment.

11.1 Hardware and software setup description

The test scenario is run on off-the-shelf PC hardware, including a number of desktop systems with processors ranging from 1.7 GHz up to 2.4 GHz. All systems are interconnected using a gigabit LAN ethernet switch. On the software side, the implementation uses the JRTPLIB [Liesenborgs 01] library for transmissions of the real-time video streams. Compression and decompression are taken care of by codecs from the FFMPEG avcodec library.

The timing results shown earlier are reiterated in table 11.1. As can be seen, for a high quality video stream in CIF resolution, encoding a single frame requires 3.15 milliseconds on the 2.4GHz system. Given the fact that we require at least 25 frames per second (or 40 milliseconds per frame), this means that in practice, we hit a limit of around 4 high quality streams that can be encoded at the same time (consuming 12 milliseconds out of 40 available). The remainder of the time slot is divided equally over the decoding for incoming video frames (decoding time ranges from 2/3 to 1/3 of encoding time) and the graphical rendering of the environment. By adjusting the frame rate of the video streams, we can increase the processing time available to the other parts of the application, as more time slots become available. A combination

Res.	FPS	MET 1(ms)	MET 2(ms)	kbps	MDT 1(ms)	MDT 2(ms)
CIF	25	4.2	3.15	110	3.25	2.29
CIF	15	5.38	3.5	50	3.31	2.21
QCIF	25	2.92	2.12	90	0.92	0.73
QCIF	15	1.46	1.37	25	1.07	0.67
Total		13.96	10.14	275	8.55	5.9

Table 11.1: Video timings and measurements.

of 3 different qualities to be encoded is therefore perfectly attainable using commodity hardware. It should also be clear from these results that one can fit 4 streams of the qualities described in the table into the uplink channel of a typical broadband home connection (which range around 384kbps).

Practical tests have shown that the P4 1,7 Ghz system is capable of real-time encoding of 4 quality streams and decoding streams from 20 clients using the quality settings as described in table 11.1. It should be obvious that faster machines are able to decode either more streams or higher quality streams.

11.2 Description of methodology used for evaluating scalability

11.2.1 Applicability of autonomous avatars

To evaluate the impact of the addition of video to the scalability of the architecture, we adopted the same technique as described in chapter 7. This enables for true trace analyses to be performed on actual data sent over the network.

The system is adapted in such as way that each of the autonomous avatars now transmits 'dummy' video streams that display the same network traffic pattern as the actual video streams that would be transmitted in a human-controlled test setup. This way, we can effectively simulate large amounts of users in the virtual world on a limited number of actual computer systems.

For the test setup, we have chosen to disable video stream decoding by the autonomous avatars. Although it may be argued that we hereby deviated from the original goal of simulating real-life behavior of a human user, several reasons exist why this choice is warranted. First of all, autonomous avatars do not output any visual information but rather communicate their behavior through positional (and orientation) updates only. Secondly, the effective con-

tent of the video streams does not influence the movement of the autonomous avatars in any way (content is not analyzed). The final and most important reason is that having to decode all video streams would drastically reduce the number of autonomous avatars that can be supported on a single simulation machine because of processing power consumption.

11.2.2 Quality selection strategies used

To obtain realistic results, we have measured the processing load and bandwidth consumption on a system running a human-controlled avatar under three different selection strategies. The first quality selection strategy will select only the high quality streams from the regions that are in the Video Area of Interest. An optimization is to include the distance between the avatar and the other regions in a calculation to determine the high-, medium-and low-quality areas. This mechanism is used in our second selection strategy. Finally, we can also make use of the view frustum of the user in order to further limit the number of subscribed multicast groups. This is done in our third quality selection approach. The strategies are comparable to those depicted in figure 10.2 A,B and D, shown earlier in this part. We have used three separate PCs that were connected to the same virtual environment to obtain comparable results under the three quality selection strategies. Measurement results are detailed in the next section.

11.3 Test results

11.3.1 Dummy stream characteristics

First of all, we will describe the quality parameters that we adjusted to obtain the three different streams used in our test setup. These are detailed in table 11.2 and depicted in figure 11.1. The values for the target bitrate are chosen so as to deliver acceptable qualities for the typical situation they are used in (i.e. depending on the average distance of the region to the avatar that is receiving the video streams). The bit rates can, of course, be adjusted to suit individual needs.

In this test, we used a total of 3 PCs to generate the autonomous avatars with dummy video streams. The maximum number of avatars used in this test is limited to 125. They were added to the simulation in batches of 25, over a period of 260 seconds. After this time, the simulation continued with the 125 avatars for approximately another 400 seconds. Avatars are all spawned at the

Stream ID	Resolution	FPS	Target Bitrate
High Quality	CIF	25	80 kbps
Medium Quality	CIF	15	40 kbps
Low Quality	CIF	15	25 kbps

Table 11.2: Stream quality definitions.

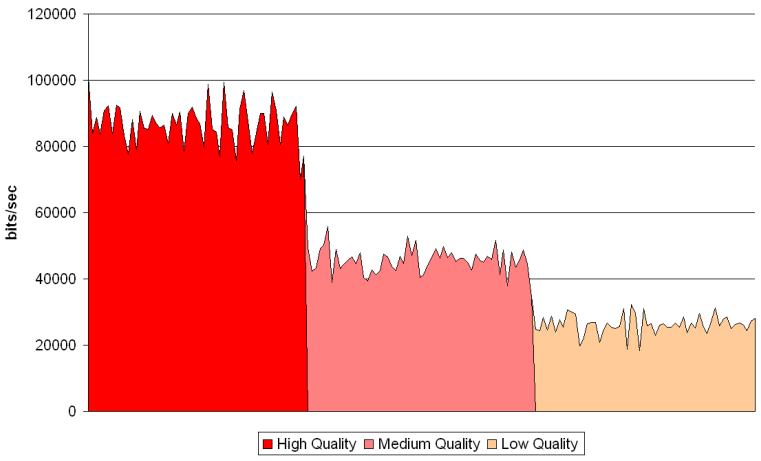


Figure 11.1: Stream qualities.

same position, which is the same as the initial location of the human-controlled avatars.

The PC on which the measurements were made were running the virtual environment application with a human-controlled avatar with nine regions in the Video Area of Interest. In figures 11.3, 11.5, 11.7, two-dimensional overviews of the Video Areas of Interest are displayed. The selected quality for a specific region is indicated by the background color of that region. The red regions are regions from which high-quality streams are received. The pink regions indicate medium-quality streams and the light pink regions indicate the lowest quality setting. Gray regions indicate regions from which no streams are currently selected, and subsequently no data is received. Also note that the yellow square and cone depict the position and view frustum of the human-controlled avatar. Autonomous avatars are represented by blue squares.

The two-dimensional overviews should be related to the preceding figures, where the traffic charts for the three quality selection strategies are depicted. Each of the charts contains vertical lines which indicate the timestamp at which the 2D overviews were captured. For example, position (a) in chart 11.2 corresponds with figure 11.3(a).

11.3.2 Detailed discussion

At around the 88th second, the system is in the state depicted by the marker 'a' in figures 11.2,11.4 and 11.6. The second batch of 25 avatars is introduced in the simulation, increasing their total number to 50. Because of the fact that the autonomous avatars are spawned at the position of the human-controlled avatar, a peak in high-quality stream traffic can be noticed. After a short while, the avatars are either dispersed into lower quality regions, or have moved out of the users's Video Area of Interest. In case there is no actual quality selection (figure 11.2), it can be observed that, as could be expected, the total downstream traffic on the human-controlled avatar is much higher than the other two strategies. The traffic on the system that employs the strategy without frustum culling is still higher than the traffic on the system with frustum culling, due to the incoming traffic of the autonomous avatars that are located behind the human-controlled avatar.

Around the 260th second, the final 25 avatars were introduced in the simulation. It can clearly be seen that the traffic peaks at around this timestamp, again due to the fact that these new avatars are located in the high-quality region of the human-controlled avatars. Overviews of the three strategies are presented in figures 11.3(b), 11.5(b) and 11.7(b).

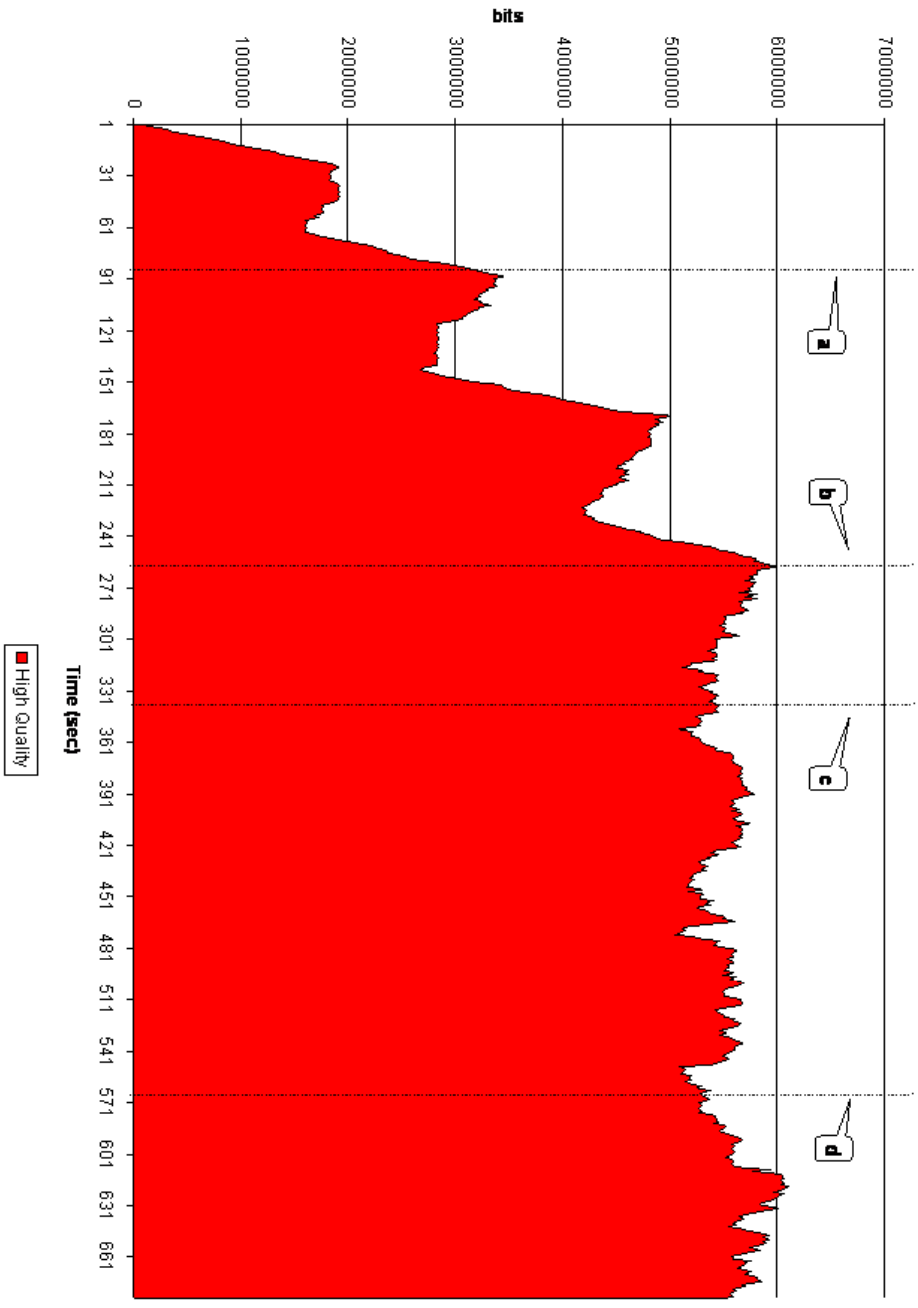
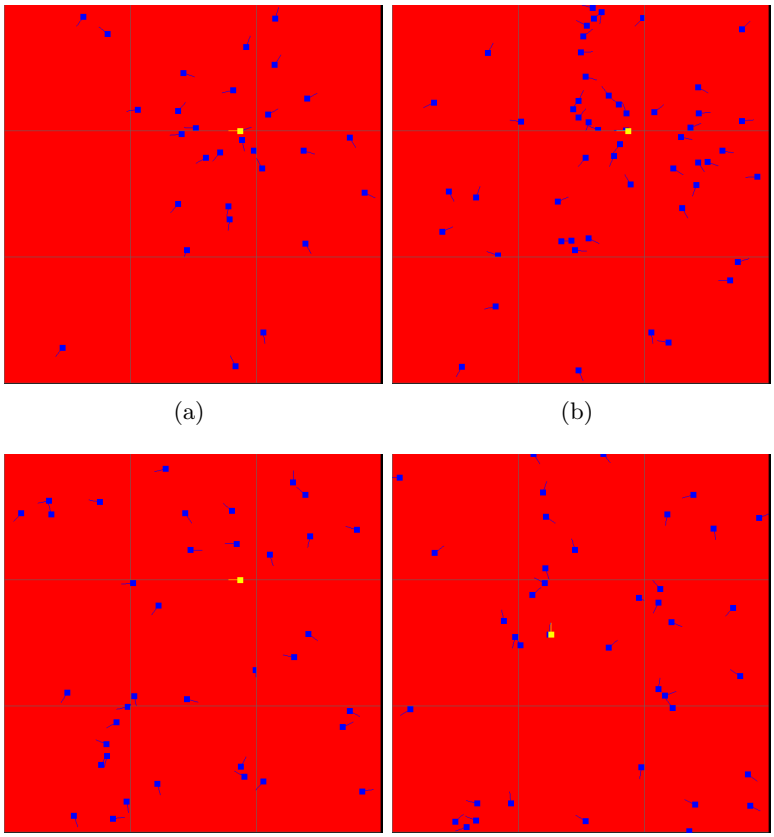


Figure 11.2: Results without quality selection.



(a) (b)

(c) (d)

Figure 11.3: Quality selection strategy 1 (highest quality only).

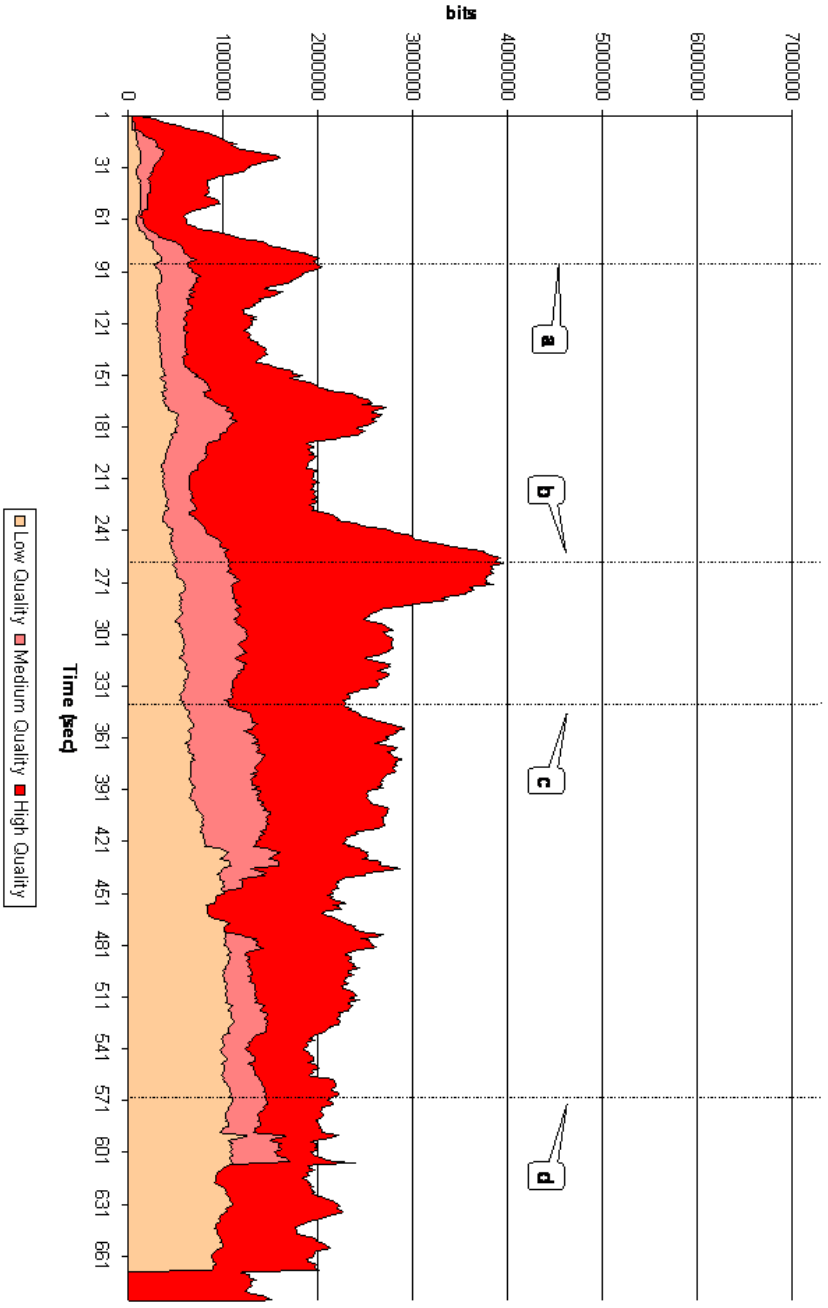
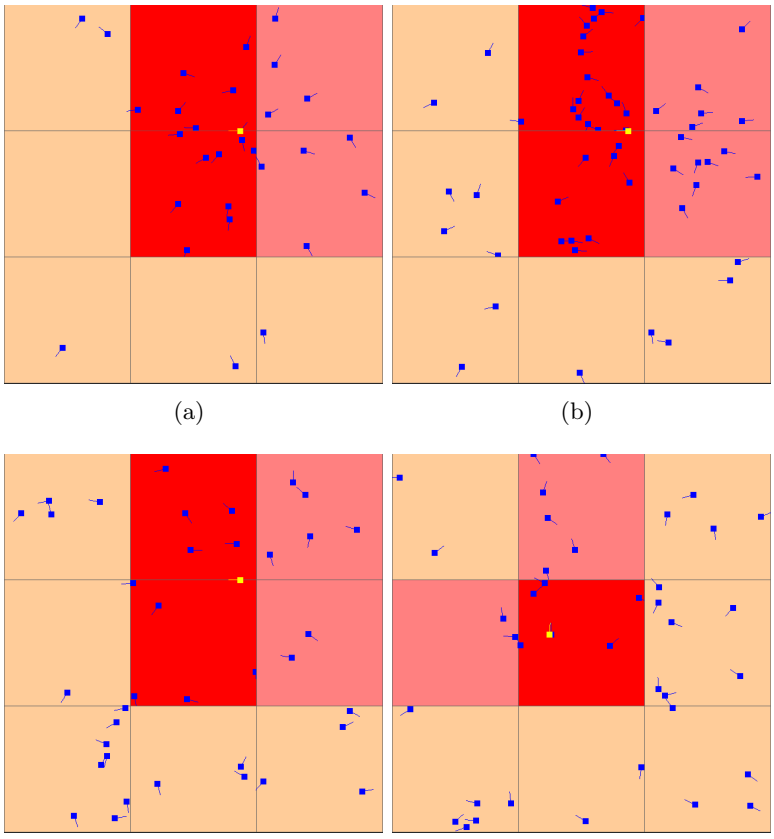


Figure 11.4: Results with quality selection, but without frustum culling.



(a) (b)

(c) (d)

Figure 11.5: Quality selection strategy 2 (without dependency on view frustum).

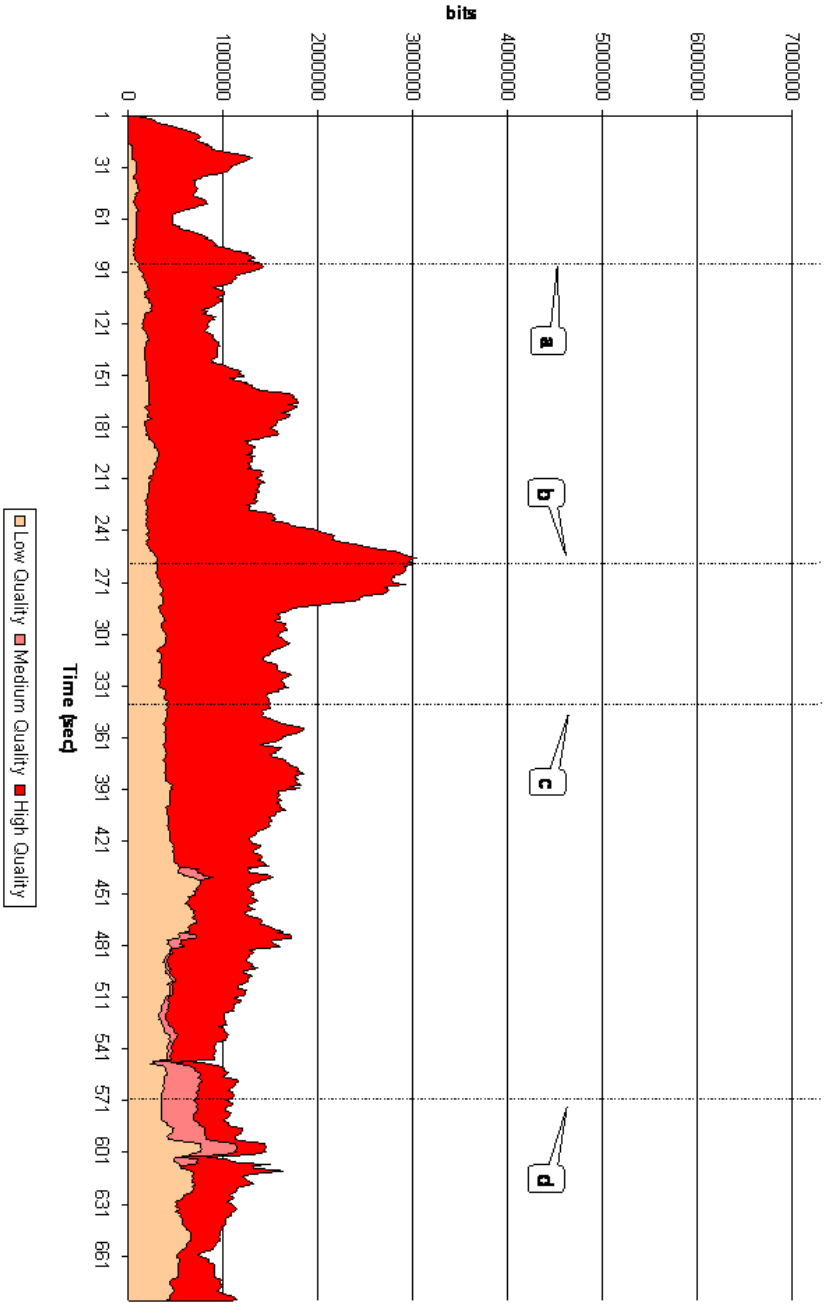
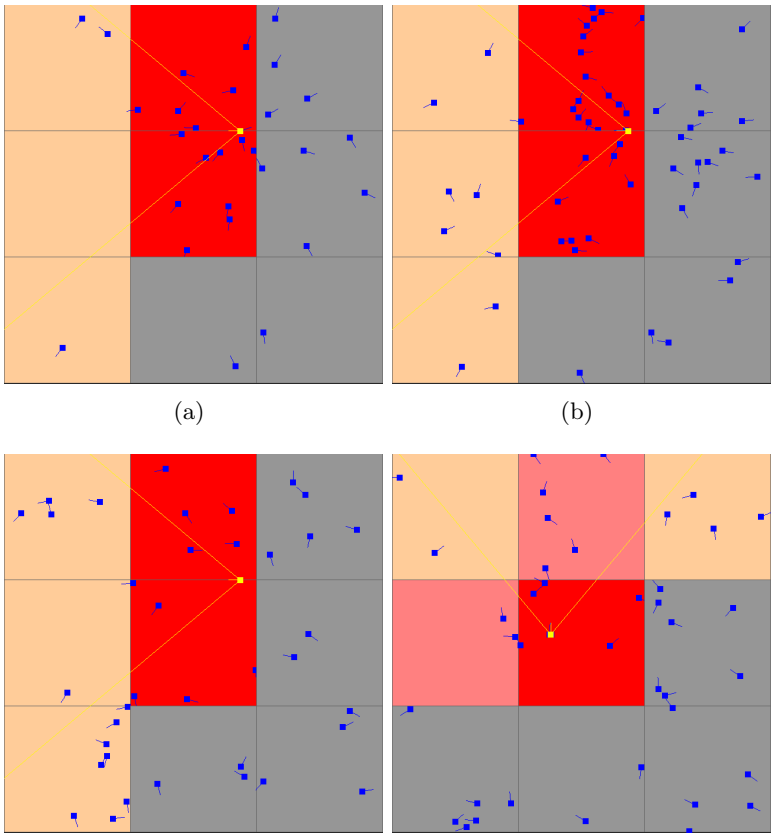


Figure 11.6: Results with quality selection and frustum culling.



(a) (b)

(c) (d)

Figure 11.7: Quality selection strategy 3 (with dependency on view frustum).

The simulation stabilizes some time after, and figures 11.3(c), 11.5(c) and 11.7(c) depict the state of the world at that time. It should be clear from the traffic charts that the strategy with frustum culling results in the least downstream traffic for the human-controlled avatars. Bandwidth usage comparison between the three strategies results in a ratio of approximately 1:1.6:3.3 (best to worst order).

When moving around the world, the selection of qualities of the regions changes dynamically. At the same time, new regions may enter the user's Video Area of Interest. This is depicted in figures 11.3(d), 11.5(d) and 11.7(d), which represent the state at around the 571th second. It should be clear that there are now also medium quality regions selected by the human-controlled avatar that takes into account the view frustum. Total downstream traffic for the strategy without quality selection is now around 5.2 megabit per second. For the second strategy, without taking into account the view frustum, we obtain a bandwidth usage of about 2 megabit per second. Finally, when taking into account the user's view frustum, bandwidth usage is further reduced to about 1 megabit per second, which yields ratios of 1:2:5.2 (best to worst order).

In general, we can see that in case no quality selection is made, bandwidth does not decrease dramatically after a batch of new avatars is introduced in the simulation, except for a small amount which is due to avatars that have moved out of the human-controlled avatar's Video Area of Interest. The desired behavior can however be observed when making use of quality selection. When autonomous avatars move into lower-quality regions, their associated individual downstream bandwidth usage decreases, which in turn makes for a decrease in total traffic consumption. We should note that, while the strategy which takes into account the view frustum yields the best results, it may not in fact be suitable for use when switching between multicast groups takes a long time. When turning around quickly, new groups have to be subscribed, and the time it takes for the data to be propagated to the new subscriber may lead to undesired latency effects. In this case, the strategy depicted in figure 10.2c would yield much better results.

Chapter 12

Discussion - Part III

In this part, we have presented an extension of the basic ALVIC setup to include multimedia stream distribution. Visually, the inclusion of video communication is achieved by incorporating a system of video texturing on the 3D meshes of the avatars – resulting in the creation of so-called Video Avatars. The advantage of this approach is that video quality – and subsequently the bandwidth consumption associated with individual video streams – can be adapted according to the distance between the avatars without a major impact on the subjective quality of experience.

At network-level, the system distributes video streams in a number of qualities, each of them generated client-side through multiple encoding steps. While this has a negative impact on system performance, this disadvantage is clearly outweighed by the added advantage of easy bandwidth usage throttling – a technique not dissimilar to the one presented in the previous part. Deciding on the number of incoming streams and/or their quality settings is entirely controlled client-side and can be determined according to an arbitrary number of factors – which is designated as the Quality Selection Strategy. The resulting system does not rely on the (commercial) availability of MPEG-4 FGS-alike codecs that include scalable transmission but can – as an advanced optimization – be adapted to incorporate them.

The scalability test setup that was developed for the basic ALVIC architecture was adapted in such a way that dummy video streams were integrated to simulate actual traffic flows originating from real-world users. By combining these new data streams with the AI-techniques that were already present in the setup, we were able to validate the concept. Practical tests have shown

that the application of an efficient quality selection strategy can result in a bandwidth saving of a factor 5 when compared to the non-optimized scenario.

In the next and final part, the deployment of ALVIC on mobile networks will be discussed.

Part IV

Mobility

Introduction

In the previous chapters of this text, all attention has been focused on the PC as primary platform for application deployment. Mobile devices, being part of the ubiquitous computing experience, are rapidly becoming more powerful in terms of computing resources. At the same time, universal access to the Internet trough a variety of technologies such as GPRS, UMTS and WLAN is available to the masses (although sometimes with a hefty price tag).

Although the initial impression might be that the migration towards the mobile scenario is a trivial case, as mobile networks superficially act like any other network offering IP services, the reality is that the heterogeneous market of mobile hardware restricts the availability of resources in terms of bandwidth, computing power and screen size availability. All three limitation need to be taken into account for a successful deployment of networked virtual environment applications on these devices.

It would definitely be short sighted not to investigate possible future (and potentially entirely novel) applications of NVE technology when deployed on mobile hardware, so as a final part of this text, we discuss the migration of the architecture to the mobile scenario. Besides the architectural issues, the effective deployment of an ALVIC-based application on mobile networks is analyzed, taking into account the aforementioned restriction on computing resource availability.

Chapter 13

Complications concerning mobile access to NVEs

Mobile (cell phone) networks have gone through a number of generations of development. While the first generations were focussed entirely on voice communication (using little bandwidth), 4G networks are touted as being fully IP-based and capable of offering at 100Mbps of bandwidth per channel to mobile end-users, enabling for example the delivery of high-definition video on mobile devices. In practice however, at least in Europe, telecom operators are only now starting to push the third generation of mobile networks, offering IP services over UMTS (the Universal Mobile Telecommunications Systems). Unfortunately, high costs, attributed to the gigantic prices paid at the time of auctioning of the radio spectrum for these services, are clearly a hindrance to the mass-market uptake of these new services. Also, mobile network operators are reluctant to open up their networks to allow just any sort of IP application to be deployed, as the use of free VoIP software would see their revenues from the classic telephony service rapidly vanish. This is even more true for the case of SMS (short messaging service), which charges an exorbitant price per byte transmitted over the network: providing free alternatives would be similar to killing the goose that laid golden eggs. The few applications that are currently offered (such as mobile TV) are deployed under strict monitoring of the network operators and are charged either by the minute, data unit or per session.

In this chapter we will abstract from these economic and business-model issues and focus on the technical issues associated with currently existing mobile (data) networks.

13.1 Network and device issues

The main challenges for deploying NVE applications on mobile devices are twofold: at the one hand there is the dynamic nature of the (non-permanent) connection. Also, advanced features available on fixed networks may be unavailable in their mobile counteparts. On the other hand, the form factor of mobile devices and their computing resources are limited, triggering adaptation of applications to take these into account.

13.1.1 Mobility and network access

There are, roughly speaking, two sides to the mobility challenge : on the hand (from a technical network point-of-view) technology needs to be developed that ensures that elementary data packets can be sent and received at all times while the user is moving. On the other, applications need to be made aware of the fact that the user is mobile and may have different expectations of and different ways of interacting with an application, compared to the desktop scenario. In case of ALVIC, we are mainly concerned with the higher level (application-oriented) issues, but as the underlying network imperfections may impact the applications themselves, we will first provide a short and non-exhaustive overview of some solutions for providing an 'always-on' network connection.

Although one could analyze the mobility problem in networks at a very low level (starting with radio spectrum issues and signal strengths), we will abstract from these and assume that the lowest levels of the network stack are taken care of. Specifically, issues at level three (network) of the TCP/IP reference model provide useful insight into the repercussions on the application-levels that may be expected. Although a draft of version 6 of the Internet Protocol has been in existence since 1998, the proliferation of IPv4 deployment has meant that a lot of effort has gone into devising extensions for this version of the protocol. Specifically, the proposal for Mobile IP [RFC 2002] includes provisions for maintaining the same IP address during an entire communication session. In practice, this is achieved by providing a mobile host with two addresses, one on the so-called 'home network' and one on the 'foreign network'. Specialized agents (which may be implemented in network hardware such as routers) are responsible for tracking the association between the two address spaces. Although the location (and address) of the mobile user on the foreign network may change due to roaming requirements, the home address remains the same. By routing and tunneling messages through the home

agent, mobile users can always be reached through the same home address.

Specifically in the case of ALVIC, the support for multicast in these mobile networks is a required feature, which, as it turns out, is not extendable from the fixed network case in a trivial way due to routing issues. Although provisions for multicast in mobile networks are defined in the Mobile IP proposal (through subscription renewal at every change in the foreign network or through bi-directional tunneling), a more scalable and dynamic solution is presented in [Chikarmane 99]. The authors propose to place the root of the multicast routing tree always at the home network, using a tunnel (if required) to reach the mobile host. In this way, packets sent by the mobile host will always seem to be originating from the home network (thereby eliminating downstream discarding of apparently spoofed IP packets). Another benefit of the proposed solution is that the multicast capabilities of foreign networks can be exploited, in contrast to the standard Mobile IP solution in which tunnels to individual mobile hosts are required. Another solution is presented in [Acharya 96], which includes an extension of the distance vector multicast routing protocol for mobile hosts. While we will not go into the technical details, the fundamental ideas are compliant with those proposed in the Mobile IP draft. However, this solution also features the benefit of being able to use link-level multicast on the foreign networks, something which is not possible in the original draft.

Version 6 of the Internet Protocol also includes extensions for mobility (see [RFC 3775]), many of which are based on the earlier version 4. There are however several advantages to the use of the latest version, including the removal of the requirement of having hard- and software support in routers used as foreign agents. Also, several routing enhancements make it easier to manage traffic sent to and from mobile hosts. Although the new protocol also has enhanced provisions for multicasting, the mobile scenario still makes it difficult to design a scalable and highly dynamic solution (see, for example, [Zhang 06] and [Garyfalos 05]).

13.1.2 Throughput issues

In this section, we will discuss results obtained from real-world tests with a GPRS [World] network, available through Proximus, the Belgian cellular phone network provider. The tests were carried out on a notebook, equipped with a PC Card, which is used as a modem. This allowed to eliminate variations in signal strength as much as possible.

In these tests, the NUTTCP application was used to measure throughput under varying packet lengths. As the overhead generated by the underlying

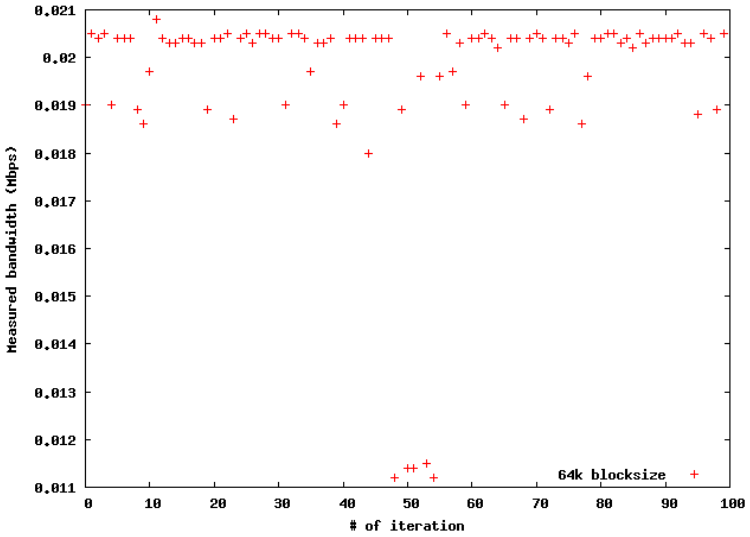
network protocol may influence the total througput (as is the case with wired networks such as ethernet), a relatively high packet length will produce optimal results.

In figure 13.1, results are shown for the GPRS network. We can conclude that the optimal throughput rate is around 20kbps. For the 3G network (see figure 13.2, capacity fluctuates between 290 and 300 kbps.

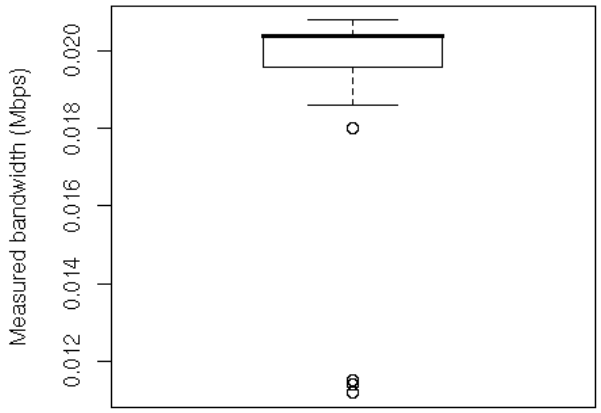
Although no quantitative tests on delay were performed, we have obtained good results with the currently deployed UMTS network. There is empirical evidence that, on average, delay values are below one second.

13.1.3 Device-related issues

Due to the small form factor, inherent problems plague mobile devices. Battery technology is still at that stage in development where size determines capacity. With lower battery capacity comes lower processing power and reduced screen sizes. To break this vicious circle, either the processing units used in mobile devices need to be made more energy-efficient or we would have to wait for a revolution in battery technology. In the remainder of this part, we will assume that these problems can either be resolved through hardware adaptations or that they are here to stay (e.g. limited screen size).

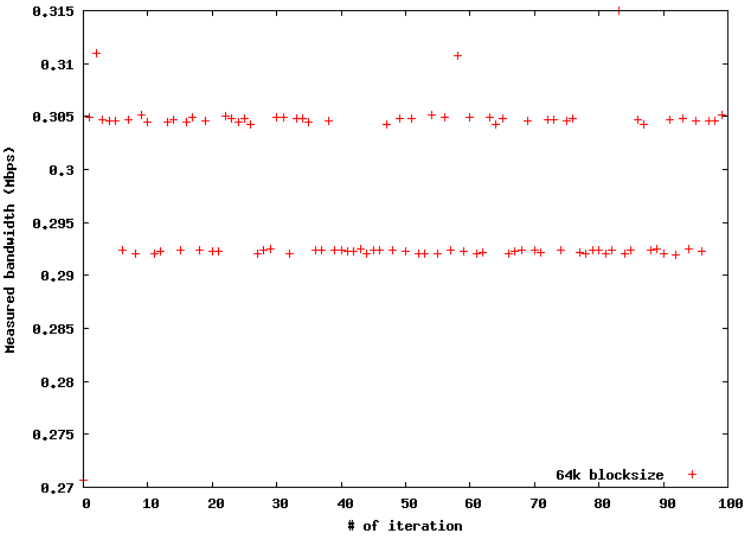


(a) GPRS throughput plot

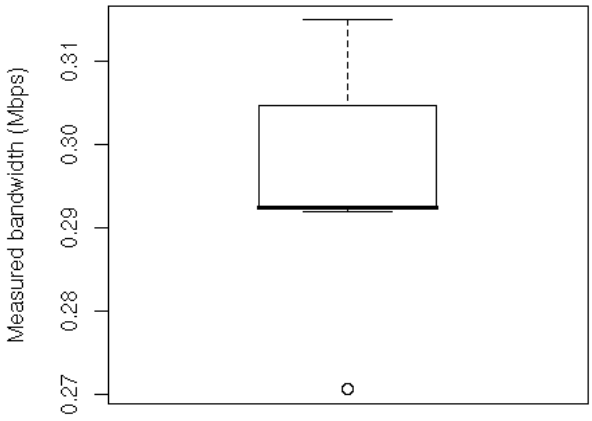


(b) GPRS throughput variance

Figure 13.1: Optimal throughput of a GPRS connection with 64kB block size.



(a) 3G throughput plot



(b) 3G throughput variance

Figure 13.2: Optimal throughput of a 3G connection with 64kB block size.

Chapter 14

Combining mobile and fixed access to NVEs

While providing access to NVE-like applications through (a variety of) mobile networks is a difficult subject in itself, the combination of supporting wired and wireless clients simultaneously makes a complex situation even worse. In this chapter, we will look at possible extensions to the proposed multicast-based architecture of part II to include precisely this support.

14.1 Mobility extensions of ALVIC

To recapitulate the general overview of the multicast-based architecture, we refer to figure 14.1.

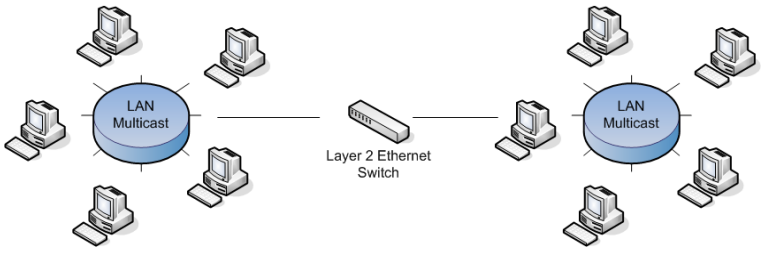


Figure 14.1: Existing architecture for wired NVE access.

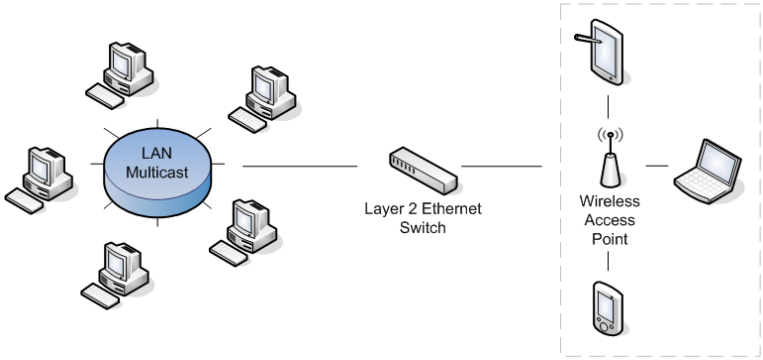


Figure 14.2: Extended architecture for short range mobile access.

14.1.1 Short-range LAN connectivity

The most obvious way of extending the existing architecture towards mobile access is depicted in figure 14.2. A number of wireless access points, each connected to the LAN, forward traffic over a wireless connection to the mobile devices. While this seems an obvious approach to tackle the connectivity problems, there are some issues to be resolved. First of all, bandwidth, and more specifically throughput, over a wireless (radio) link is limited when compared to a Fast Ethernet LAN. This leads to problems when dealing with an NVE in which a large number of users are employing video-based avatars. These issues can, up to a certain degree, be dealt with in a relatively straightforward fashion. As the framework makes use of multicast for distributing both event and audio/video data, we can already filter out a great deal of unnecessary traffic at protocol layer 2 (link layer). The majority of manageable ethernet switches nowadays are able to 'snoop' multicast traffic that is sent over the wire. This is accomplished by eavesdropping on the IGMP traffic that is communicated between clients. By intercepting both multicast 'join' and 'leave' messages, a switch is able to determine what groups should be forwarded over each of its links. If such a multicast-enabled switch were to be used to connect each of the access points to the wired LAN, a large number of users may be connected directly to the multicast-enabled LAN. The major downside of this approach is the limited range of these WLANs [IEEE].

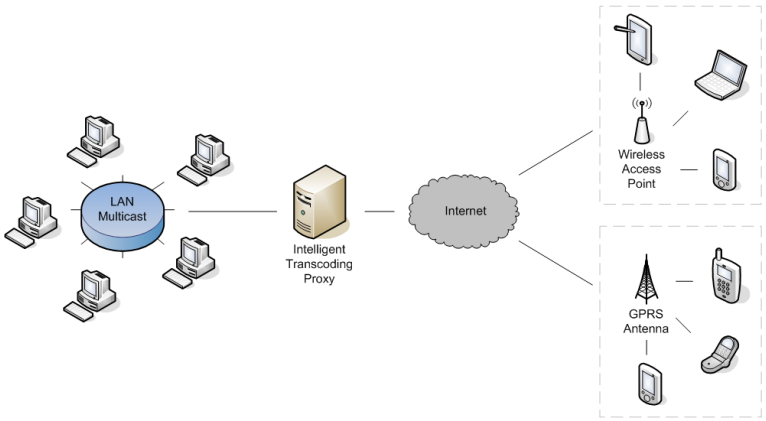


Figure 14.3: Proposed architecture for long range mobile access.

14.1.2 Long-range access through mobile networks

To enable mobile access to NVE applications over truly large distances, a number of connection methods should be considered. At short range, WLAN 802.11 would be an obvious candidate if such an infrastructure is at hand. In other cases, a GPRS or similar connection may be the only way for a mobile device to connect to the virtual environment. It should however be clear that in these cases, it is exceedingly difficult to provide users with a view of the virtual environment comparable to the one that is available when connected directly to the multicast-enabled LAN.

This is why we propose to insert the intelligent proxy as described in section 10.4 at the edge of the LAN which provides a number of services specifically targeted towards mobile users. The resulting architecture is depicted in figure 14.1.2. The proxy needs to fulfill two major tasks. First of all, the proxy should act as a multicast-to-unicast gateway for remote mobile clients. As there is very little support for multicasting in the Internet at present, traffic being sent on the multicast-enabled LAN should be forwarded towards each remote mobile client through unicast connections. However, simply duplicating all multicast network traffic that resides on the LAN does not scale well with a growing number of connected users, neither is it suited for low-bandwidth connections such as GPRS. For these reasons, we believe the proxy should have an extensive knowledge of both the application it is serving as well as the network infrastructure. Based on its compound application and network awareness, the proxy can then intelligently decide which multicast

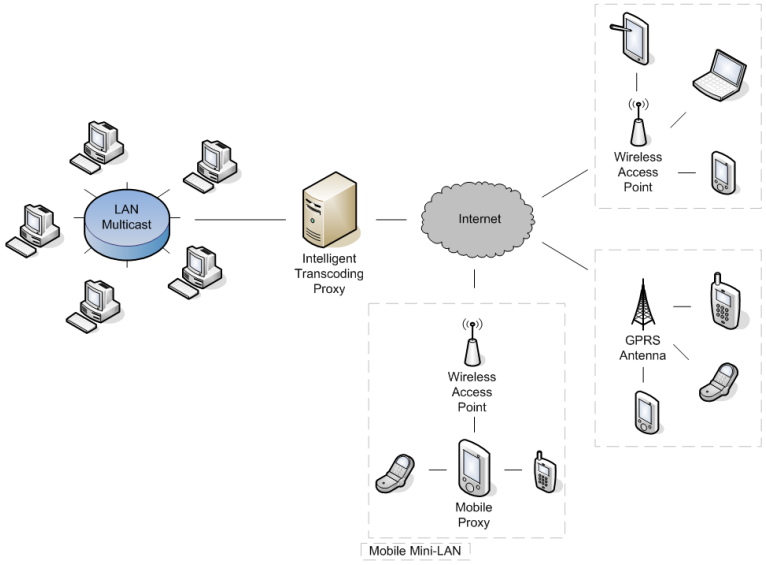


Figure 14.4: Proposed architecture including Bluetooth-based short-range mini-LAN.

traffic should be unicasted to each individual remote client.

Secondly, we envision that the proxy will also be acting as a transcoder of audio and video streams sent out by wired users. More details of this case are given in section 14.2.3.

14.1.3 Ad-hoc LANS

As detailed above, placing an intelligent proxy at the edge of the multicast-enabled LAN, combined with WLAN and GPRS mobile connectivity, enables access to the NVE application for a large number of devices. However, there are still some that may not be covered due to, for example, lack of availability of long-range radio functionality in devices. Our third proposed extension to the architecture tries to solve some of these issues by using ad-hoc network technology (see discussion in [NIST b]) between mobile devices in close neighborhood, forming local meshes.

In this scenario, a powerful mobile device will perform part of the proxy functionality that was described in section 14.1.2. By maintaining BlueTooth connections between this device and each of the clients in its vicinity, we will be able to transmit the necessary data over these short-range wireless channels,

forming a local mesh or virtual mini-LAN (see fig 14.4), in case of BlueTooth referred to as a PicoNet. The 'mobile proxy' is responsible for providing the uplink to the rest of the network, either through a GPRS or WLAN connection. In the case of a GPRS connection, the available bandwidth is severely limited, so the mobile proxy should instruct the fixed proxy to discard all video streams at the edge of the multicast-enabled LAN, leaving only the positional and state information available for the mobile clients in range of the mobile proxy.

Given the roadmap for BlueTooth development presented in [SIG], we may even consider passing video streams along the ad-hoc BlueTooth network when a high-bandwidth WLAN uplink is available. This can be achieved because multicasting data along BlueTooth connected hosts will be supported for up to 7 devices. Some caution is needed however, as the achievable data rates are currently unknown due to lack of available hardware.

14.2 Porting of the existing framework onto mobile devices

To get acquainted with the mobile device platform and its limitations in terms of processing power, connectivity issues and graphics capabilities, we opted to make a straight port of the existing NVE application, including the underlying multicast-based architecture, to the Windows Mobile platform. In this section, we will discuss some of the techniques and features that had to be altered or enhanced for use on the mobile platform. The results shown here are indicative of what is currently possible using state-of-the-art hardware and universal (IP-based) wireless access to broadband networks. For a more generic discussion on service deployment on mobile (ubiquitous) platforms, we refer to [Ardaiz 01].

14.2.1 3D Interface

Our mobile NVE extensions target users on the road using a PDA or Smart-Phone or similar device. The general idea is to provide an opportunity to stay in touch with the NVE using a 3D interface comparable to the one they are used to on their desktop systems. An example of this can be seen in figure 14.5. The two images on the left display screenshots of our client application running on a PDA while the image on the right shows the corresponding view from the desktop client. As can be seen, both the environment and the avatars are represented similarly on both platforms.

We have also provided mobile users with the possibility to view the video streams transmitted by other users of the NVE, just as it was the case for

the architecture described in part II. To indicate which clients are sending out video, we have placed a multi-colored 3D arrow above the client avatar. The mobile user can indicate he wants to view a client video stream by tapping on the corresponding avatar with the stylus. The 3D arrow of the selected video avatar will then start rotating, indicating that the currently visible video originates from that client. Please note that, due to processing limitations (and screen size), the implementation is limited to visualizing only one specific video stream at a time.

14.2.2 Alternative visualization

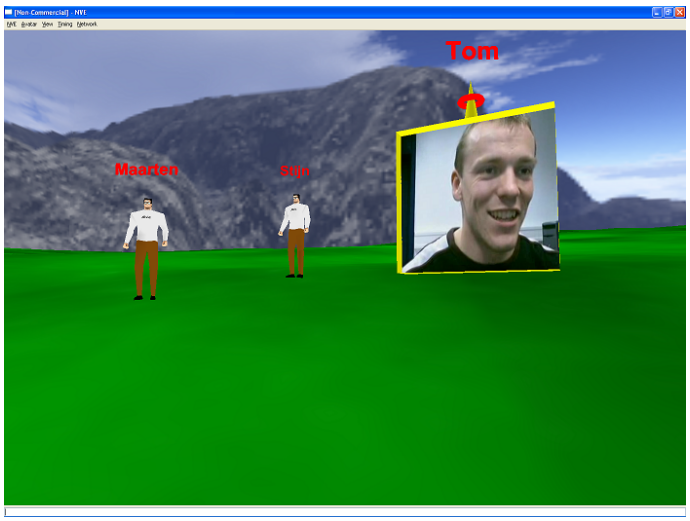
While a 3D interface is very intuitive to use, not all mobile devices will be capable of displaying the 3D environment at a sufficiently interactive framerate. For that reason, we have provided an alternative in the form of a simple 2D interface, as can be seen in figure 14.6. Again, we have provided the possibility to view the video stream sent out by one of the other clients. We have used different colors to indicate whether or not a client provides a video stream and to indicate which client's video stream is currently being displayed. For instance, a video client is indicated in green, while the selected video client is colored brown. Other clients are colored blue and the client's own avatar is indicated in red. This 2D interface can be used on a wider range of devices because it is very lightweight and automatically scales to the resolution of the device output screen.

14.2.3 Video communication

Mobile devices in particular can benefit from tailor-made video streams. This way, the already limited bandwidth that is available is used to the maximum of its capacity. However, directly applying the video quality selection strategy detailed before is not an adequate solution. For a mobile device to encode a number of video streams concurrently would leave no room for other processing tasks. Also, as the screen size is inherently limited, high-quality streams (from a desktop machine point of view) are not required. We therefore foresee that the proxy architecture as described in section 10.4 will be of great importance for the mobile scenario.

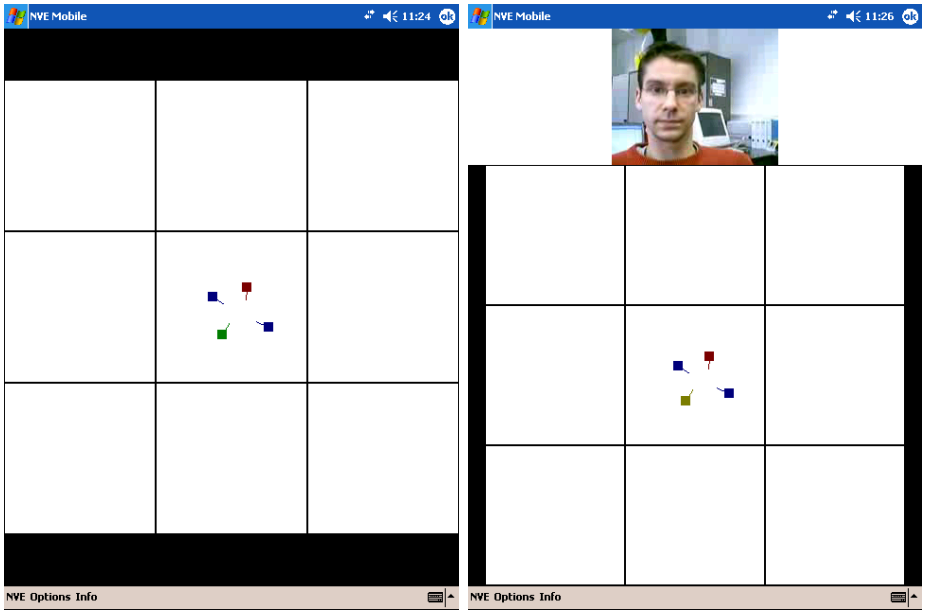
The intelligent proxy is extended with video transcoding functionality. This results in the network setup shown in figure 14.4. The intelligent proxy acts as a unicast-to-multicast and multicast-to-unicast gateway for its connected clients. This means mobile clients on the one hand can unicast their

(a) (b)



(c)

Figure 14.5: Rendering the NVE on a mobile device. (a) screenshot from the mobile client. (b) viewing one client video stream. (c) the same view from the desktop application.



(a) (b)

(c)

Figure 14.6: 2D overview of the environment. (a) 2D only on mobile device. (b) combined with video display on mobile device. (c) corresponding 3D view on PC.

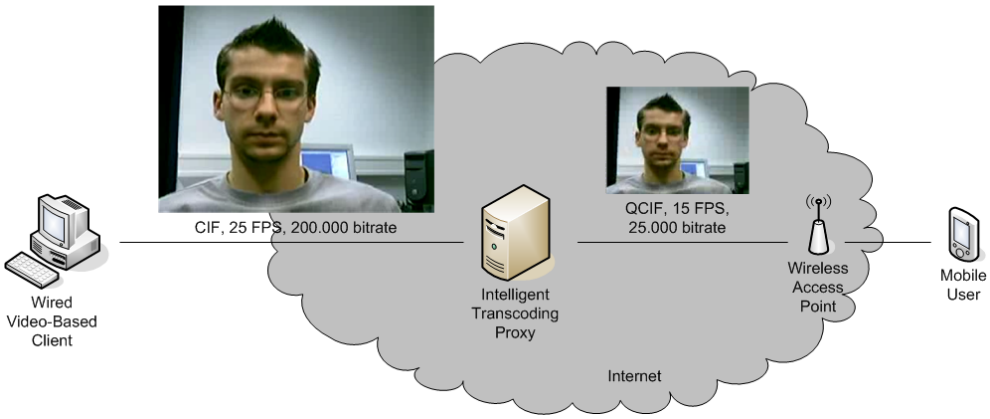


Figure 14.7: On-the-fly proxy transcoding of video streams with client-specified quality parameters.

positional and state information to the proxy, which will subsequently disseminate this information to all interested wired clients by sending it to the correct multicast group. On the other hand, the intelligent proxy subscribes to multicast groups on behalf of connected mobile clients, and subsequently unicasts the relevant information distributed in these communication channels to them. Furthermore, mobile users can also indicate at which quality they want to receive video streams. If the requested video quality does not match the quality of the video stream as sent out by the selected video-based client, the intelligent proxy transcodes the original video stream to the requested format before forwarding it to the mobile client. This is illustrated in figure 14.7.

A general discussion of the implementation of our intelligent proxy can be found in [Wijnants 05a], while the proxy's video transcoding functionality is described in full detail in [Wijnants 05b].

Chapter 15

Remote visualization for mobile access

While a straight port of a 3D environment onto a mobile device (as described in the previous chapter) would be ideal from the application developer point of view, in reality several issues exist, specifically due to the inherent limitations of less-powerful classes of devices than the ones we used. We have already established a number of these issues in section 13.1, but the rendering problems in particular have proven a limiting factor for devices without hardware acceleration. At the same time, for those classes of devices that do support native 3D hardware-assisted rendering, device specific tweaking of the rendering engine settings is required and/or the lack of support for advanced vertex shaders increases rendering times. Although it is highly likely that these open issues will be resolved in the future, application developers wanting to deploy NVE applications on existing hardware should be provided with a feasible fall-back scenario for the time being. We will discuss an alternative to native 3D rendering on mobile devices that has been developed, specifically for those classes of devices that are capable of at least decoding an incoming video stream, preferably through dedicated hardware.

This topic of research is part of a larger project targeting the use of portable devices as mobile guides to enhance the experience of visiting a city. As one cannot realistically expect for all users of such as system to use similar hardware, a fallback scenario was required for the visualization of the 3D world. To make it clear why a three-dimensional representation was chosen, we will first discuss a typical usage scenario.

15.1 Usage scenario and context

The application to be developed is, as stated before, a mobile city guide, which runs on off-the-shelf mobile hardware that is handed out to tourists visiting a city (through the visitor information centers) and/or hardware that is owned by inhabitants of the city. The system provides a 3D environment representing the city center, as shown in figure 15.1. Through use of active sensing devices, in this case GPS sensors and/or the available wireless network connection, the location of each of the devices and users, active in the city, is known at all times. This information is subsequently transmitted to all other users that have shown an interest in this type of information, allowing people to track one-another while walking through the city. Evidently, the tracking features of the system can easily and selectively be turned off for e.g. privacy reasons. Specifically in the 3D city context, the generation of new information and accompanying metadata is a crucial contribution. In a typical usage scenario, pictures and/or video fragments will be shot using the built-in camera of the mobile device the user is carrying, and subsequently annotated through the use of speech-to-text analysis. At the same time, the location the picture was taken at will be added to the asset as meta-data. The picture is instantaneously uploaded to the main database through the device's wireless network link and shared with other users of the system. This type of user-generated content will be visualized through the use of tags in the 3D environment, representing the particular type of data that is available at that particular location. Besides the rather trivial means of representing pictures as tags, the system also allows users to actively participate in extending and embellishing the 3D model of the city, by contributing their pictures to serve as textures to be placed on the facades of the various buildings in the city. Tags that are placed in various location throughout the city may be commented on by other users, e.g. to provide reviews on restaurants, museums etc.

For related work that shows some similarities to our own, we refer to [Mitchell 03]. In this paper, 'real tournament' is introduced, a mobile multi-player game that uses, amongst others, a GPS receiver and electronic compass to receive information on the environment. The game itself is played in the environment of a public park, equipped with WLAN access providing IPv6 communication channels. Through this IPv6 network, group communication (voice chat) is enabled using the push-to-talk principe. While the interactivity requirements are more or less the same when compared to the scenario described above, the fact that the game is being played in a restricted environment and that no interaction with fixed devices is required sets it apart.

(a) (b)

Figure 15.1: Screenshots of system concept.

In [Burigat 05], the authors present LAMP3D (Location-Aware Mobile Presentation of 3D content), a system that is used to develop a mobile guide that can display 3D models on a PDA, based on VRML descriptions. The navigation through the world is achieved using a GPS module, similar to the setup we propose. Besides the use of VRML as description language, the main difference is the fact that the setup proposed in this paper is not truly multi-user, in the sense that there is no direct interaction between users.

15.2 Remote rendering in general

Remote rendering in general is a technique in which computing power, available in a centralized machine, is used to calculate several independent visualizations, that are subsequently sent to a number of connected clients. This way, the clients themselves need not be equipped with high-end graphics ca-

pabilities and processing load in general is reduced. All that remains is for these clients to display the incoming data stream, which may come in the form of an encoded video stream or a sequence of still images. As should be clear, the main challenge lays in the addition of interactivity in such a system, as for any type of interaction to take place, the user input information should be sent back to the remote render server (or another server in case of a more complex architecture).

A parallel can clearly be drawn between the application of remote rendering and thin client systems. The latter is based on the deployment of terminals, popular in the 1980s with systems such as IBMs AS/400. Thin clients ideally only provide a ways of getting user input into the systems and output on the screen. All processing should (in theory) be done on a server infrastructure, although this requirement is often relaxed in favor of interactivity. Advantages of such an approach are clear: lower IT maintenance cost, optimal use of computing resources and flexibility for end-users. However, the disadvantages sometimes prohibit the application of thin clients: additional interactivity delay, network throughput requirements and the reliance on a single resource (the server).

Remote rendering is discussed in several scientific publications, but in this context we will focus on those that are relevant for the mobile scenario, as well as some other related work.

In [Humphreys 02], the authors describe the CHROMIUM framework, which uses clusters of computers equipped with commodity graphics accelerators to generate 3D visualizations. The interface for the application programmer consists of the well-known OpenGL API. Commands are sent over the network to a series of Chromium servers that are responsible for processing the commands (either process them locally or dispatch them for further processing. The main advantage of this approach is that applications need not be altered (drastically) to include remote rendering support, as the underlying OpenGL API remains the same. The authors show that the system applies to, among others, volume rendering and stylized drawing and has provisions to guarantee interactivity. However, the framework is targeted specifically towards large displays (which cannot easily be powered by a single graphics card), and it requires a high capacity network to retransmit the rendered parts back to the original station. It is therefore not really suitable for the mobile scenario.

The authors of [Brachtl 01] have implemented a system in the context of a museum visit. People interacting with the provided PDA's indicate their starting and end point, information which is subsequently transmitted to the

server infrastructure responsible for generation the virtual walkthrough. In the scenario envisaged, the system is being used to generate on-demand tours through the museum, focussing on user-determined points of interest. Internally, a walkthrough is generated each time a request is handled, using a cache which is maintained for frequently used sequences. The generated walkthrough is sent back to the end-user in a custom format, designed specifically for monochrome low-resolution PDA screens. Although the system is shown to be effective for the walkthrough scenarios, it is not generic enough to be directly applicable to a range of applications.

A hybrid system can be found in [Diepstraten 04]. In this system, the processing power available both at client-side and server-side is leveraged to provide the end-user with a basic visualization. At the same time, bandwidth usage and memory requirements are kept under conrol. The authors have achieved this by using the server to generate an internal 3D view. This view is subsequently processed in such a way that only the 'feature lines' are retained, e.g. boundaries and silhouettes. These feature lines, which make up a 2D representation of the view on the 3D object, are subsequently transmitted to the client, which is responsible for drawing them on screen. The system is applicable for mobile devices, as 2D rendering is often quite efficient due to the (already present) need for rapid GUI visualization. The method presented is clearly usable for individual objects, but less so for a complete virtual environment.

The system described in [Cohen-Or 02] renders the 3D scene entirely server-side and generated MPEG-4 video streams that are to be decoded client-side. Additionally, the authors have opted to make use of the available depth information to encode the information into several layers, each with a distinct quantization factor, determining the quality and size of the stream. For background layers, the quality is reduced, and vice-versa for the frontmost layers. In practice, this reduces the total amount of traffic to be sent over the network. However, no details on the impact on interactivity are provided.

For some other pointers to related work, we refer to [Yoon 00]

15.3 Modified architecture

We have already gone into great detail to point out the advantages of using multicasting to build a scalable networked virtual environment. Unfortunately in this case, we had to come up with a compromise due to the fact that multicast features were not enabled in the already deployed city network the application had to be connected to. However, given the fact that the system

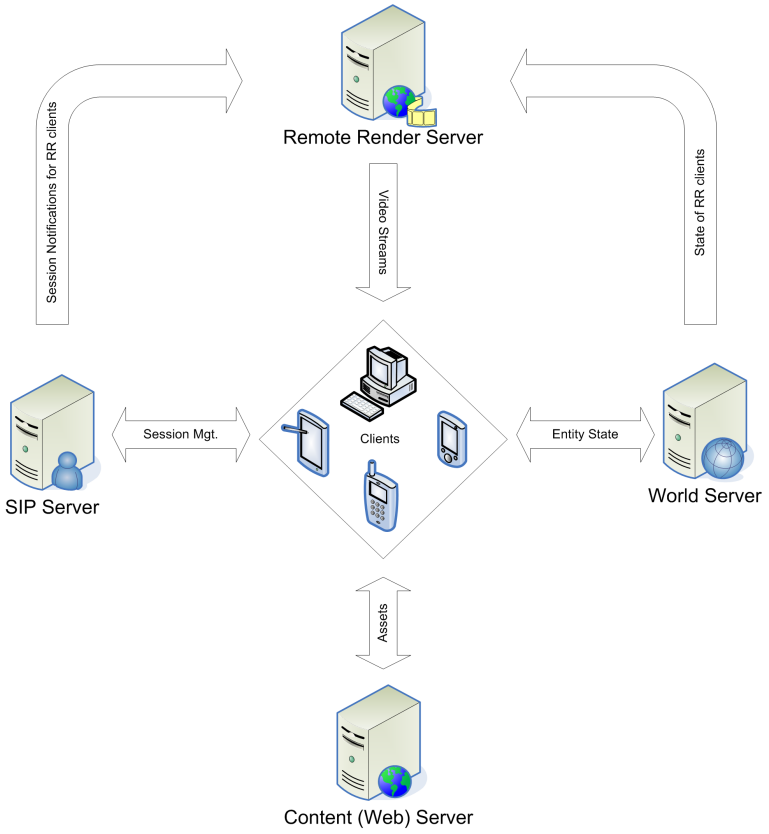


Figure 15.2: System architecture including remote rendering services.

was not required to scale up to the massive numbers of users envisioned in our earlier work but rather to about 250 simultaneous users, the network architecture was based on the software framework described in section 6.5. This provides a system that reiterates the ideas of spatial subdivision and area of interest management, but implements them through unicast connections rather than multicast groups.

Also, other features that we already identified in earlier chapters are of primary importance to the application and service provider. They include the ease of management, the ability to provide access to a persistent world and the ability to moderate user-generated content and actions, all of which are easier to implement in a pure client/server based architecture.

A third important factor behind the choice for a client/server based architecture for this specific application was that it would need to be able to interface with an existing management system (for accounting/billing purposes, inter-person communication and storage of content). It was therefore much easier to design a unicast-based world server setup than to convert the existing infrastructure to comply with our own standards.

Because of the fact that the remote rendering infrastructure outputs video streams for visualization, we opted to include support for the SIP protocol (session initiation protocol) in order to be able to make use of existing client-side applications, running on a variety of devices and software platforms. Surely, the SIP protocol is used in a growing number of (open standards-based) voice/video over IP applications for session control purposes, so including support for this protocol allows direct use of existing client software for visualization of the video streams.

The resulting system architecture including all main (server) components and their connections is shown in figure 15.2.

As should be clear from this picture, the server infrastructure consists of four main parts, namely the SIP server, the World server, the Content server and finally the topic of this chapter: the Remote Render server. Clients that wish to join the virtual world will do so by first connecting to the SIP server and, after authentication and session setup is finalized, will be assigned a World Server to connect to. It should be obvious that these elements are functionally compatible with the 'management servers' and 'game servers' of the proposed architecture in part I. However, all state conversations (positions, orientations, camera angles,...) happen between clients and the world server, without the need for extra inter-client connectivity. Only when audio/video communication is explicitly requested is the SIP server queried for the current location (IP address) of the other party, after which a direct connection is

established for transmission of RTP data. The responsibility of the Content server is the management of all multimedia assets and associated metadata that is present in the virtual world.

We should point out at this time that – taken into consideration all elements in the architecture described above – there is no distinction between clients that perform their rendering locally and those that use the remote rendering infrastructure. There is in fact no direct data flow originating from the mobile clients towards the remote render server (as is made clear through the unidirectional arrow in the diagram). All information that is necessary for performing the rendering by the server can be retrieved from the World server. While this slightly increases the processing load on the clients that wish to use remote rendering, we feel that this is an acceptable compromise, given the fact that it greatly simplifies the communication flow between all types of devices in the architecture.

15.3.1 Remote rendering stages

The Remote Render Server consists of two separate processes. The first process is responsible for generating the graphical content. In this case the graphical content is the view of a client on the 3D virtual city. When a client wishes to use the facilities offered by the Remote Render infrastructure, a notification is sent to both processes of the server, see figure 15.3.

In the first process (Render process) this notification will result in the allocation of an available viewport to the client. In the second process (Customization process) a worker thread will be started containing a video encoder that will process the incoming views into a video stream that gets sent back to the client which in turn decodes the stream to get a view on the virtual world. This view can correspond to the user's view in the real world when GPS is used but it can also deviate from the real world when the user chooses to use keypad/stylus input. In a typical usage scenario, the mobile client will automatically transmit positional data using a GPS module, indirectly telling the first process where to position the client in the virtual world. The user can also choose to send commands telling the World server to make his/her avatar move in a certain direction or to make his/her avatar turn around to get a better view on the surroundings. We should stress again that this data does not get sent directly to the remote render server, but rather to the World server which will, in turn, deliver this information to the Remote Render server. In every render pass the 3D engine renders the scene in every viewport using the corresponding personalized virtual camera.

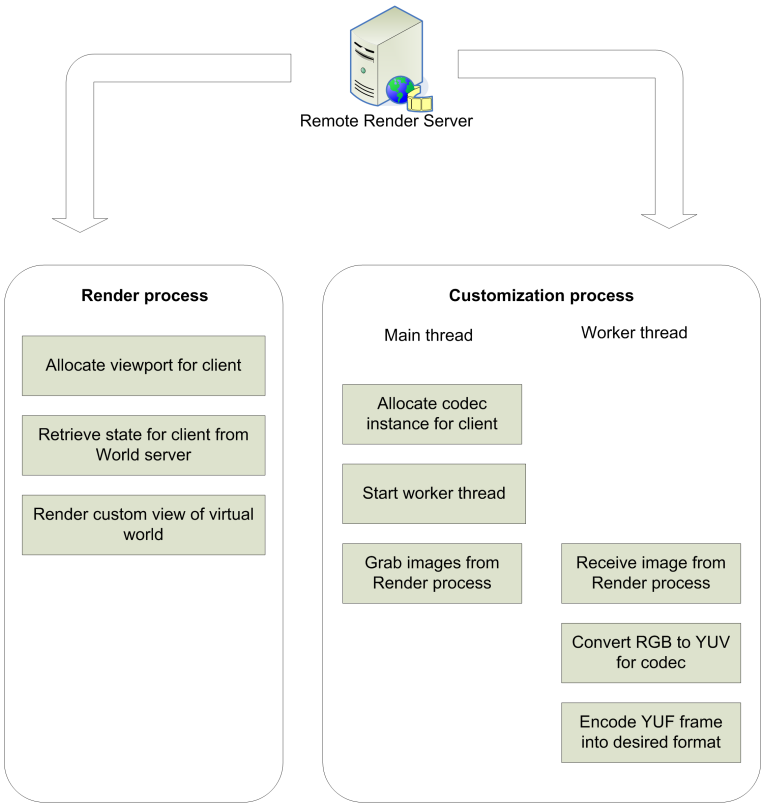


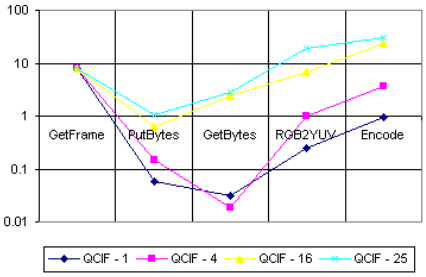
Figure 15.3: Remote rendering stages.

The second process (Customization) is responsible for grabbing the graphical content generated by the Render process. When a client connects to the virtual environment, the Customization process in the Remote Render server gets notified and starts a worker thread. Every worker thread corresponds to a client and contains an RGB buffer, a YUV buffer and an encoder. After every pass of the Render process, the main thread of the second process will grab the backbuffer of the Render process containing all the viewports. To efficiently grab the backbuffer of the first process, the Customization process installs hooks in the graphical library (OpenGL or DirectX) used by the engine in the Render process. Every time the render engine in the Render process calls swapbuffer, a function in the Customization process gets triggered to first copy the backbuffer to main memory before the actual swapping begins. The second process then has the backbuffer in main memory and can start processing the data contained in this buffer. First the buffer gets sliced into segments. Each segment contains the data corresponding to one viewport. These segments get passed to a framebuffer manager. The main thread then notifies all the waiting worker threads that a new frame has arrived and is ready to be processed. Every worker thread retrieves the corresponding framebuffer and converts the data in this buffer from RGB to YUV. This step is necessary because nearly all codecs rely on the input frames being in YUV format. The encoder then takes this YUV frame and adds it to the video stream. When a worker thread is finished with that frame it puts itself on hold until it gets notified by the main thread. The resulting stream is sent back to the client which in turn decodes the video stream to get a view on the 3D virtual city. Every worker thread's encoder can be customized to best fit the client's needs. Parameters such as frames per second, bit rate, resolution and codec can be adapted. This way the stream can be tailored to best fit the receiving device.

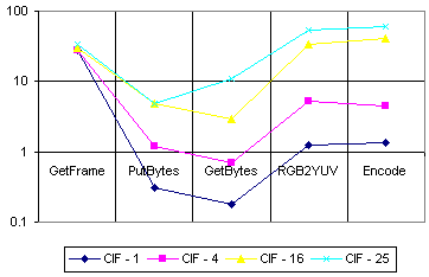
15.3.2 Test results

In order to get accurate readings from our trials, that can later be extrapolated, we have opted to use a single trial server, with following specifications: an Intel dual core 3GHz CPU, 1GB of main memory, an NVIDIA Geforce 7200 LE graphics accelerator and a standard installation of windows XP. Given the fact that all of this hardware is commonplace, the inclusion of such an infrastructure will remain cost-effective for service providers.

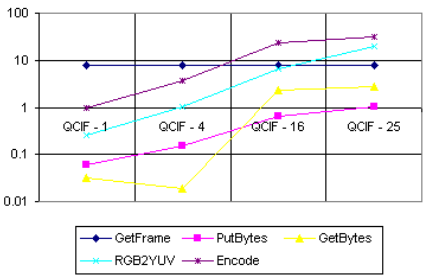
Several video codecs were considered for use in the architecture, however given the limitations of mobile devices in terms of network throughput and screen resolution, the choice fell on the H. 263 and mpeg-4 codecs. These



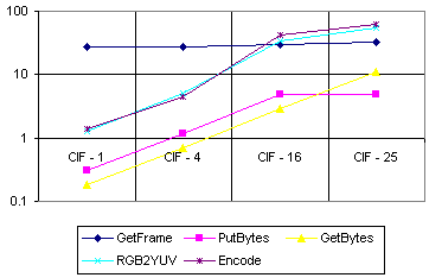
(a) H. 263 QCIF. X axis shows stage in customization process.



(b) H. 263 CIF. X axis shows stage in customization process.

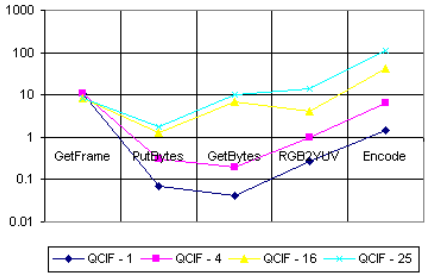


(c) H. 263 QCIF. X axis denotes number of clients.

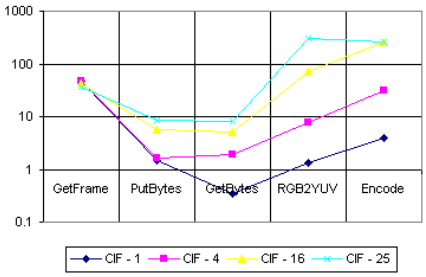


(d) H. 263 CIF. X axis denotes number of clients.

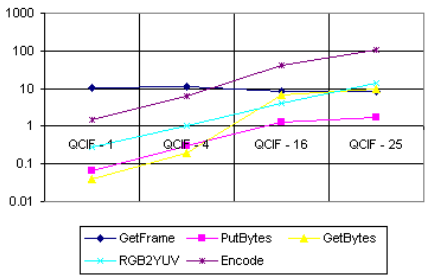
Figure 15.4: H. 263 timing results in msec.



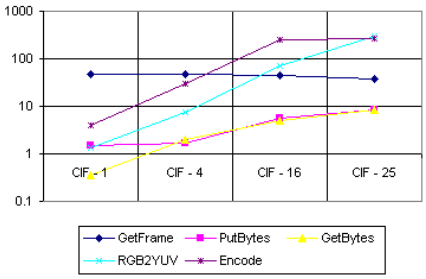
(a) MPEG-4 QCIF. X axis shows stage in customization process.



(b) MPEG-4 CIF. X axis shows stage in customization process.



(c) MPEG-4 QCIF. X axis denotes number of clients.



(d) MPEG-4 CIF. X axis denotes number of clients.

Figure 15.5: MPEG-4 timing results in msec.

are often supported by media processors that may be present in 3G mobile devices. In this section we will provide test results for both H. 263 and mpeg-4. Both codecs used are part of the LibAVCodec library, which in turn is part of the FFMPEG project [FFMPEG]. It would have been interesting to be able to include H264 in these tests, however, due to license restrictions on the available encoders, this was not an option at the time of writing.

The first stage in the remote rendering process is the actual rendering of the virtual world for all assigned clients, based on the state as received from the World server. However, as the performance of this stage depends heavily on the complexity of the scene that is being rendered and the available graphics accelerator, this stage is not included in the measurements. The second stage can roughly be divided into five functions: GetFrame, PutBytes, GetBytes, RGB2YUV and Encode. The GetFrame function comprises grabbing the backbuffer via the hooks in the graphical library and transferring this backbuffer from graphical memory into main memory. In the function PutBytes, the main thread slices the backbuffer into separate viewports and copies them into framebuffer objects contained in a thread-safe framebuffer manager. Every worker thread in turn calls GetBytes to retrieve the framebuffer from the framebuffer manager. Next, the RGB data contained within the framebuffer object is transformed into YUV data in RGB2YUV. Finally this YUV data is passed on to the encoder that adds a new frame to its output stream.

We have performed measurements on these 5 functions using the H. 263 and MPEG-4 codecs. For both codecs we measured results using QCIF(176 x 144) and CIF(352 x 288) resolution viewports. Every resolution was measured running 1, 4, 16 and 25 clients.

The results show that Getframe remains constant for all tests using CIF and also using QCIF (see e.g. figures 15.4(c) and 15.4(d). The resolution of the backbuffer used with CIF viewports was 1920x1440 and 1024x768 with QCIF viewports. The cost of GetFrame is defined by the cost of copying data from graphical memory to main memory and as the amount of data remained constant for all tests with CIF and all tests with QCIF the time spent in this function is constant. The time spent in the function PutBytes increases with the number of clients. This function's initial cost is that of waiting for a lock on the framebuffer manager. Each client adds an additional cost in terms of copying the viewport data into a framebuffer object assigned to that client. Every worker thread has to wait for a lock on the framebuffer manager to get the content of the corresponding framebuffer object to copy the framebuffer data into an RGB buffer. This RGB buffer will be converted by RGB2YUV

into a YUV buffer. The cost of the functions PutBytes, GetBytes, RGB2YUV and Encode is determined by the amount of data per viewport and the number of clients. We aim for a video stream of 15 frames per second. The H. 263 codec produces satisfactory visual results for limited screen resolutions at 15 frames per second.

Our measurements have resulted in the following figures for QCIF resolution with a load of 25 clients and H. 263 encoding: GetFrame 7.88 msec, PutBytes 0.042 msec, GetBytes 0.112 msec, RGB2YUV 0.749 msec and Encode 1.221 msec. This results in a total time of 10.09 msec per client per frame or 61.48 msec per frame for a total 25 clients. This translates into a frame rate of approximately 15 FPS. Using CIF resolution, the totals change to 38.372 msec per client per frame or 164 msec per frame for all 25 clients. This translates into approximately 6 FPS. The main bottleneck however are the GetFrame function and the RGB to YUV conversion, both of which will be subject to optimization (see the section on future work).

In summary, we can say that by using QCIF resolution for the viewports and the H. 263 codec for encoding the tests show that this is feasible for 25 simultaneous clients. Using CIF resolution approximately half that number can be supported. It is also clear from the differences between figures 15.4 and 15.5 that the H. 263 codec is able to compress the frames faster when compared to the MPEG-4 codec used.

Chapter 16

Discussion - Part IV

The migration of ALVIC on mobile devices was the focus of this part. The architectural changes required were discussed first, based on three example scenarios that encompass a range of typical access technologies. At all times, the interconnection of mobile networks and fixed networks needed to be taken into account, which is required for easy migration of sessions between network types and/or communication between users connected through various access networks.

Besides the optimal scenario, where a high-capacity wireless LAN is available, the less-ideal cases of cellular networks and ad-hoc networks were studied. While in the first case, a hardware-only change was required to allow for mobile access, the second and third scenario required an adaptation in software in order to be practically viable. Most importantly, the inclusion of an intelligent proxy in the network enabled the adaptation of data flows, especially required for mobile devices, due to the limited availability of bandwidth and computing resources.

Besides the obvious issue of bandwidth restrictions, the limited capabilities of mobile devices in terms of graphical rendering were identified as a major hurdle for the deployment of 3D applications. In this part, we therefore also investigated the possible application of a remote rendering infrastructure to mitigate this major issue. The ALVIC architecture was adapted in such a way that the use of the remote rendering infrastructure was 'invisible', both to the end-user and to other devices. Unlike some previous examples of remote rendering, the additional resource availability of modern mobile devices was put to use to process the interaction information originating from the devices

themselves (for example, collision detection and picking). It was shown that, using a software-based approach to the problem, a single off-the-shelf PC was able to serve up to twenty-five simultaneous users with a video stream of sufficient quality. At the same time, it was shown that the integration of multiple remote render servers (from an architectural point of view) would not be causing additional scalability issues.

Chapter 17

Overall conclusions and future research directions

At the end of this dissertation, it is time to take a look back at the research goals as defined in section 1.1, and to see whether the requirements were met.

In part I, through various measurement setups, on which practical examples of networked virtual environment related applications were deployed, we were able to determine some ballpark figures on bandwidth consumption. The results have shown that, for a system that is relatively static – meaning that the content is changed infrequently- and that relies on a powerful server infrastructure, a low bandwidth connection usually suffices. However, as soon as either communication channels (like audio and video), peer-to-peer traffic (dynamic game hosting) or user-generated content streaming is integrated, bandwidth requirements surge and a broadband connection is of vital importance.

A test environment was installed in laboratory conditions, in which the impact of network delay and jitter on a popular first person shooter game was determined. This was achieved by taking both objective and subjective measurements in a set of controlled gaming sessions in which 14 players participated. The conclusions of this experiment were that an impact on both the players' score and their subjective quality rating was apparent from a threshold of about 60 milliseconds. Also, there was no impact on the performance of players that were not subject to delay by the others that were in the same session. Finally, a correlation between the objective performance and subjective experience was found to be present. The analysis of the influence of jitter on performance however provided inconclusive results.

In part II, a basic architecture to support networked virtual environments

was developed. The main feature of this architecture is the ease of scalability – both in terms of the vastness of the virtual world as well as the amount of simultaneous participants. The resulting architecture is called 'ALVIC', the Architecture for Large-Scale Virtual Interactive Communities. ALVIC relies heavily on the use of IP multicast as network distribution mechanism. Through a system of spatial subdivision and linking of this partitioning to multicast groups, combined with an appropriate area of interest management technique, bandwidth is easily controllable by individual end-users.

The ALVIC framework was subsequently put to the test by designing a scalability testing environment. To be able to gather real world test data, autonomous avatars were developed that roam the virtual world in a way similar to their human counterparts. By combining a number of test machines into a cluster setup and distributing a high number of instances of the autonomous avatar software over these, we were able to attest that the architecture performs as claimed. On the one hand, server load is reduced so that at least 1000 simultaneous users can be supported on a single machine, on the other it is shown that adaptation of downstream bandwidth is attainable by adjusting individual areas of interest.

The addition of multimedia streams to an NVE-like application boosts the bandwidth requirements. Ensuring scalability under these conditions is therefore a non-trivial assignment. Specifically in ALVIC the addition of video was studied and described in part III, reusing the concepts already implemented in the first version of the architecture. Using a run-time quality selection strategy, individual clients can adapt the quality of incoming video streams depending on available resources, both in terms of bandwidth and processing load. The system was tested for its scalability using the same autonomous avatar approach that was discussed before. It was proven to be capable of real-time encoding and decoding a sufficient number of streams at several bit rates, making the solution practically viable.

Part IV focused on the migration of NVE applications to mobile platforms. In particular, three possible architectural extensions were proposed, that allow for both mobile and fixed clients to participate in the same virtual world. Access to the network can be ensured through a variety of network technologies, ranging from WLAN to Bluetooth. It was also shown that the ALVIC framework is directly portable to mobile devices, if sufficient bandwidth and processing power is available on mobile devices. Adaptations for fall-back scenarios on less capable devices were discussed, including alternative visualizations and a video transcoding proxy architecture. The applicability of server-based visualization through remote rendering was discussed and found

to be scalable to about 25 simultaneous clients on a single rendering server.

The research described in this dissertation is currently being extended in three ways. First, the design of the basic ALVIC framework is being adapted to better integrate with currently deployed access networks. Specifically the combination of multicast-enabled networks and unicast-only deployments is a challenging area of research. In case the architecture can be extended in such a way that multicast-to- unicast conversions can be achieved using minimal overhead, using for example an extended version of the intelligent proxy described in this text, scalability of the architecture on real-life networks can be envisaged to several tens of thousands of users.

Secondly, the intelligent proxy in its role as adaptation node for multimedia data is being developed into a more generic component to be included in a variety of scenarios, not necessarily limited to networked virtual environments. The addition of several generic features, such as on-the-fly transcoding of video streams and intelligent flow selection based on bandwidth availability will enable the setup to be used as part of an overlay network providing services to a plethora of end-user applications.

Finally, the deployment of applications on mobile devices can be seen as the major growth area for networked virtual environments. Although we have provided some insight into possible extensions for the ALVIC framework using current-generation mobile networks, new types of mobile access technologies are constantly being developed, some of them offering specialized services that may be employed to improve the end-user experience. It is especially challenging to investigate the impact these next-generation networks and their services may have on networked virtual environment applications. However, one should not loose sight of the fact that the integration of these novel networks should not come at the cost of a diminished compatibility with previous (and possibly legacy) access technologies.

Appendices

Appendix A

Scientific contributions and publications

The following list of publications, presented at scientific international conferences, contains work that is part of this dissertation:

[**Liesenborgs 02a**] *Jori Liesenborgs, Peter Quax, Wim Lamotte & Frank Van Reeth.* Designing a Virtual Environment for Large Audiences. *Lecture Notes in Computer Science, vol. Information Networking, Wireless Communication Technologies and Network Applications, pages 585–595, 2002*

[**Liesenborgs 02b**] *Jori Liesenborgs, Peter Quax, Wim Lamotte & Frank Van Reeth.* Designing a Virtual Environment for Large Audiences. *In Proceedings of the 16th International Conference on Information Networking (ICOIN), pages 3A–2.1–3A–2.10, 2002*

[**Quax 03**] *Peter Quax, Tom Jehaes, Pieter Jorissen & Wim Lamotte.* A multi-user framework supporting video-based avatars. *In Proceedings of the 2nd workshop on Network and system support for games, pages 137–147. ACM Press, 2003*

[**Quax 04a**] *Peter Quax, Chris Flerackers, Tom Jehaes & Wim Lamotte.* Scalable Transmission of Avatar Video Streams in Virtual Environments. *In Proceedings of the 2004 IEEE International Conference on Multimedia and Expo (ICME). IEEE, 2004*

[**Quax 04b**] *Peter Quax, Patrick Monsieurs, Tom Jehaes & Wim Lamotte.* Performance Evaluation of Client-Side Video Stream Quality Selection using Autonomous Avatars. *In Proceedings of the 2004 International Conference on Advances in Computer Entertainment Technology (ACE 2004), pages 251–256. ACM Press, 2004*

[**Quax 04c**] *Peter Quax, Patrick Monsieurs, Tom Jehaes & Wim Lamotte.* Using Autonomous Avatars to Simulate a Large-Scale Multi-User Networked Virtual

Environment. *In Proceedings of the 2004 International Conference on Virtual-Reality Continuum and its Applications in Industry (VRCAI2004), pages 88–94. ACM Press, 2004*

[**Quax 04d**] *Peter Quax, Patrick Monsieurs, Wim Lamotte, Danny De Vleeschauwer & Natalie Degrande.* Objective and Subjective Evaluation of the Influence of Small Amounts of Delay and Jitter on a Recent First Person Shooter Game. *In Proceedings of the 2004 Workshop on Network and System Support for Games (NETGAMES 04), 2004*

[**Quax 05b**] *Peter Quax, Maarten Wijnants, Tom Jehaes & Wim Lamotte.* Bridging the Gap between Fixed and Mobile Access to an Audio/Video-Enabled Large-Scale NVE. *In Proceedings of the International Conference on Web Technologies, Applications and Services (WTAS). Electronic proceedings. Acta Press, 2005*

[**Quax 05a**] *Peter Quax, Tom Jehaes, Maarten Wijnants & Wim Lamotte.* Mobile Extensions for a Multi-User Framework Supporting Video-Based Avatars. *In Proceedings of the International Conference on Internet and Multimedia Systems, and Applications (IMSA). Electronic proceedings. Acta Press, 2005*

[**Quax 06**] *Peter Quax, Bjorn Geuns, Tom Jehaes, Wim Lamotte & Gert Vansichem.* On the Applicability of Remote Rendering of Networked Virtual Environments on Mobile Devices. *In Proceedings of the International Conference on Systems and Network Communication. Electronic proceedings. IEEE, 2006*

A technical report contains additional information regarding [Quax 04d]:

[**Quax 04e**] *Peter Quax, Patrick Monsieurs, Wim Lamotte, Danny De Vleeschauwer & Natalie Degrande.* Objective and Subjective Evaluation of the Influence of Small Amounts of Delay and Jitter on a Recent First Person Shooter Game. *Technical report MOVE D2.3, Limburgs Universitair Centrum - Alcatel Bell NV., 2004*

The following papers are not part of this dissertation:

[**Jehaes 04b**] *Tom Jehaes, Peter Quax, Patrick Monsieurs & Wim Lamotte.* Hybrid representations to improve both streaming and rendering of dynamic networked virtual environments. *In Proceedings of the 2004 International Conference on Virtual-Reality Continuum and its Applications in Industry (VRCAI2004), pages 26–32. ACM Press, 2004*

[**Jehaes 04a**] *Tom Jehaes, Peter Quax & Wim Lamotte.* Analysis of Scalable Data Streams for Representations in Networked Virtual Environments. *In Proceedings of the 2004 Workshop on Network and System Support for Games (NETGAMES 04), 2004*

[**Jehaes 05**] *Tom Jehaes, Peter Quax & Wim Lamotte.* Adapting a Large Scale Networked Virtual Environment for Display on a PDA. *In Proceedings of the International Conference on Advances in Computer Entertainment, pages 217–220. ACM Press, 2005*

[**Wijnants 05b**] *Maarten Wijnants, Patrick Monsieurs, Peter Quax & Wim Lamotte.* Exploiting Transcoding to Increase the User Quality of Experience in Networked Applications. *In Proceedings of the International Workshop on Advanced Architectures and Algorithms for Internet DElivery and Applications (AAA-IDEA 2005), 2005*

Appendix B

Samenvatting (Dutch summary)

De laatste jaren zijn de zogenaamde 'Genetwerkte Virtuele Omgevingen' of 'Networked Virtual Environments' aan een sterke opmars bezig. Denk hierbij maar aan voorbeelden zoals 'Second Life' of 'World of Warcraft', die zelfs geregeld vermeld worden in het journaal. De (al dan niet gegronde) hype die wordt gecreëerd rond dit soort toepassingen zorgt ervoor dat de meeste mensen zich een beeld kunnen vormen van het onderzoeksonderwerp dat behandeld wordt in deze tekst. Wat velen echter niet weten is dat dit soort toepassingen vaak het resultaat zijn van onderzoek dat oorspronkelijk bedoeld was voor militaire doeleinden, met name voor het verbinden van meerdere simulatoren. Later werd de technologie opgepikt door de entertainmentindustrie en werden de eerste spellen gelanceerd waaraan meerdere personen (door middel van een computernetwerk) deel konden nemen. Genetwerkte virtuele omgevingen combineren onderzoekstopics uit meerdere disciplines : computer graphics, mens-machine interactie en computernetwerken. De elementen uit deze laatste categorie vormen de hoofdmoot van deze tekst.

Allereerst geeft deze thesis een overzicht van een aantal bestaande toepassingen die, grosso modo, kunnen ingedeeld worden in twee categorieën : de genetwerkte games en de 'Virtual Interactive Comunities'. Door het bandbreedtegebruik en de netwerkarchitectuur van deze toepassingen te bestuderen, verbetert het inzicht in de componenten die nodig zijn om een nieuw raamwerk op te bouwen. Daarnaast is het van belang een correct idee te vormen van de impact van de imperfecties van de huidige generatie computernetwerken op de subjectieve en objectieve performantie van gebruikers. Hiertoe wordt gebruik gemaakt van een uitgewerkte testscenario waarbij de kwaliteitservaring

en doeltreffendheid van een groep spelers van een computerspel voortdurend onder de loep wordt genomen.

Deze thesis behandelt vervolgens de ontwikkeling van ALVIC, een acroniem voor 'Architecture for Large-scale Virtual Interactive Communities', oftewel een architectuur voor grootschalige genetwerkte virtuele omgevingen. Bijzonder aan ALVIC is zonder meer de schaalbaarheid, waardoor zeer grote aantallen gebruikers kunnen interageren en in staat zijn een uitgestrekte virtuele wereld te doorkruisen. Uiteraard zijn er in een dergelijke architectuur meerdere componenten noodzakelijk, waaronder een systeem dat er voor zorgt dat gebruikers die zich, vanuit een geografisch standpunt gezien, eender waar bevinden, toegang kunnen krijgen tot één en dezelfde virtuele wereld. Daarnaast moet er rekening gehouden worden met het feit dat de wereld zeer dynamisch van aard is : de gebruikers wijzigen de 'status' van de wereld voortdurend (al dan niet bewust) door de acties die zij ondernemen. Het resultaat van deze acties moet vervolgens consistent worden getoond aan de andere, met het systeem verbonden, toeschouwers. ALVIC bepaalt hoe de berichten die nodig zijn voor het uitwisselen van deze 'status' informatie opgebouwd moeten worden en hoe zij verdeeld worden onder alle participanten. Hierbij wordt specifiek rekening gehouden het grote aantal (gelijktijdige) ontvangers door middel van optimalisatie van distributiemechanismen en een intelligente opsplitsing van de uitgestrekte virtuele wereld.

Om aan te tonen dat de voorgestelde architectur inderdaad schaalbaar is, stellen we een mechanisme voor waardoor grote aantallen gebruikers efficiënt kunnen gesimuleerd worden. Het verzamelen van een groep gebruikers van dergelijke omvang (meer dan duizend) is immers niet haalbaar om elke kleine wijziging en optimalisatie te evalueren. Door gebruik te maken van enkele basistechnieken uit de artificiële intelligentie en een cluster van (standaard) computers kan aangetoond worden dat de voorgestelde oplossing een haalbare kaart is.

Naast de basisvereiste dat de virtuele wereld op verschillende machines consistent wordt gerepresenteerd zijn er, zeker voor toepassingen van de huidige generatie, eisen die de gebruikers stellen aan een interessante applicatie van dit type. Eén van deze is de integratie van video-communicatie, waardoor de interactie bevorderd wordt. Hoewel deze toevoeging vanuit een applicatiestandpunt mogelijk eenvoudig lijkt, schuilen er veel problemen op technisch vlak. Het versturen van videobeelden over een computernetwerk vereist immers grote hoeveelheden bandbreedte, hetgeen nog versterkt wordt wanneer toepassingen beschouwd worden waarbij meerdere gebruikers tegelijkertijd naar dezelfde videosequenties kijken. ALVIC heeft daarom voorzieningen om het

mogelijk te maken op efficiënte wijze videobeelden naar een groot aantal gebruikers tegelijkertijd te sturen, alsmede om het mogelijk te maken dat een individuele gebruiker materiaal ontvangt van meerdere bronnen tegelijk. Het is hierbij essentieel dat er rekening wordt gehouden met prioriteiten, waardoor aan sommige videobeelden een (relatief) groter belang zal toegekend worden dan andere. Bij dit alles speelt de subjectieve gebruikerservaring uiteraard een ook grote rol.

Als laatste wordt aangetoond dat ALVIC geschikt is als fundament voor een groot aantal verschillende toepassingen, die elk op hun beurt kunnen gebruikt worden op een verscheidenheid aan apparaten. Naast de klassieke Personal Computer zullen draagbare apparaten immers het volgende platform zijn waarop deze toepassingen worden gelanceerd. Gezien het grote aantal technologieën dat reeds bestaat om draadloze verbindingen mogelijk te maken, worden deze in deze thesis geabstraheerd door middel van drie scenario's die representatief zijn voor de meest voorkomende types. Uiteraard is de beschikbaarheid van een netwerkverbinding niet de enige hinderpaal bij het lanceren van nieuwe toepassingen op mobiele apparaten, ook de beperkte rekenkracht en schermgrootte spelen een rol. De voorzieningen binnen ALVIC voor visualisatie op afstand zijn daarom van belang, zeker op het mobiele platform.

De technologie achter ALVIC wordt momenteel verder ontwikkeld, met name om de schaalbaarheid op bestaande netwerken te garanderen en nog grotere aantallen gelijktijdige gebruikers te ondersteunen. Daarnaast is ook de mobiele component onderwerp voor verdere studie.

Bibliography

[Acharya 96] Arup Acharya. *IP Multicast Extensions for Mobile Internetworking.* In Proceedings of the Fifteenth Annual Joint Conference of the IEEE Computer and Communications Societies, 1996.

[Amir 95] Elan Amir, Steve McCanne & Hui Zhang. *An Application Level Video Gateway.* In Proc. 3rd ACM International Conference on Multimedia, pages 255–265, San Francisco, California, November 1995.

[Ardaiz 01] O. Ardaiz, F. Freitag & L. Navarro. *On Service Deployment in Ubiquitous Computing.* In Proceedings of the 2nd International Workshop on Ubiquitous Computing and Communications, 2001.

[Armitage 01] Grenville Armitage. *Sensitivity of Quake3 players to network latency.* In ACM SIGCOMM Internet Measurement Workshop 2001, Berkeley, CA, USA, Nov. 2001.

[Armitage 03] Grenville Armitage. *An experimental Estimation of Latency Sensitivity in Multiplayer Quake 3.* In 11th IEEE Int. Conf. on Networks (ICON 2003), Sydney, Australia, 2003.

[Barrus 96] J. Barrus, R. Waters & D. Anderson. *Locales and Beacons: Precise and Efficient Support for Large Multi-User Virtual Environments.* Proceedings of VRAIS'96, Santa Clara CA, pages 204–213, 1996.

[Benford 95] Steve Benford, John Bowers, Lennart E. Fahlén, Chris Greenhalgh & Dave Snowdon. *User Embodiment in Collaborative Virtual Environments.* In Proc. ACM Conf. Hu-

man Factors in Computing Systems, CHI, volume 1, pages 242–249, 1995.

[Bernier 01] Yahn W. Bernier. *Latency Compensation Methods in Client/Server In-game Protocol Design and Optimization.* In Proc. of Game Developers Conference'01, 2001.

[Blizzard] Blizzard. World of warcraft. World Wide Web, `http://www.worldofwarcraft.com`.

[Borella 00] M. Borella. *Source Models of Network Game Traffic.* In Computer Communications, vol. 23, no. 4, pages 403–410, 2000.

[Brachtl 01] M. Brachtl, J. Slajs & P. Slavk. *PDA based navigation system for a 3D environment.* Computers and Graphics, vol. 25, no. 4, pages 627–634, August 2001.

[Burigat 05] S. Burigat & L. Chittaro. *Location-aware Visualization of VRML Models in GPS-based Mobile Guides.* In Proceedings of Web3D 2005: 10th International Conference on 3D Web Technology, pages 57–64. ACM Press, 2005.

[Calvin 93] J. Calvin, A. Dicken, B. Gaines, P. Metzger, D. Miller & D. Owen. *The SIMNET virtual world architecture.* Proceedings of VRAIS'93, pages 450–455, 1993.

[Capps 00] Michael V. Capps, Don McGregor, Donald P. Brutzman & Michael Zyda. *NPSNET-V: A New Beginning for Dynamically Extensible Virtual Environments.* IEEE Computer Graphics and Applications, vol. 20, no. 5, pages 12–15, 2000.

[CastGate] CastGate. Castgate. VUB ETRO-TELE, `http://www.castgate.net/`.

[Chikarmane 99] Vineet Chikarmane, Carey L. Williamson, Richard B. Bunt & Wayne Mackrell. *Multicast Support for Mobile Hosts Using Mobile IP : Design Issues and Proposed Architecture.* Mobile Networks and Applications, vol. 3, no. 4, pages 365–379, 1999.

[Chittaro 04] L. Chittaro & Ieronutti L. *A Visual Tool for Tracing Behaviors of Users in Virtual Environments.* In Proceedings of

AVI 2004: 6th International Conference on Advanced Visual Interfaces. ACM Press, 2004.

[Cohen-Or 02] Daniel Cohen-Or, Yuval Noimark & Tali Zvi. *A server-based interactive remote walkthrough.* In Proceedings of the sixth Eurographics workshop on Multimedia 2001, pages 75–86, New York, NY, USA, 2002. Springer-Verlag New York, Inc.

[Cohen 00] R. Cohen & H. Radha. *Streaming fine-grained scalable video over packet-based networks.* In Global Telecommunications Conference, pages 288–292. IEEE, 2000.

[Diepstraten 04] Joachim Diepstraten & Thomas Ertl. *Remote Line Rendering for Mobile Devices.* In Proceedings of Computer Graphics International 2004, pages 454–461. IEEE, 2004.

[Epic Games 03] Atari Epic Games. Unreal tournament 2003. 2003.

[Ethereal.com] Ethereal.com. Ethereal. World Wide Web, `http://www.ethereal.com`.

[FFMPEG] FFMPEG. `http://www.ffmpeg.org`.

[Floyd 97] S. Floyd, V. Jacobson, C. Liu, S. McCanne & L. Zhang. *A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing.* IEEE/ACM Transactions on Networking, vol. 5, no. 6, pages 784–803, 1997.

[Frécon 98] Emmanuel Frécon & Marten Stenius. *DIVE: A Scaleable network architecture for distributed virtual environments.* Distributed Systems Engineering Journal, vol. 5, no. 3, pages 91–100, 1998.

[Garyfalos 05] A. Garyfalos & K.C. Almeroth. *A flexible overlay architecture for mobile IPv6 multicast.* IEEE Journal on Selected Areas in Communications, vol. 23, no. 11, pages 2194–2205, 2005.

[Greenhalgh 95] Chris Greenhalgh & Steve Benford. *MASSIVE: A Collaborative Virtual Environment for Teleconferencing.* ACM Transactions on Computer-Human Interaction, vol. 2, no. 3, pages 239–261, 1995.

[Greenhalgh 96] Chris Greenhalgh. *Dynamic, embodied multicast groups in MASSIVE-2.* Technical report NOTTCS-TR-96-8, Department of Computer Science, The University of Nottingham, UK., 1996.

[Henderson 03] Tristan Henderson & Saleem Bhatti. *Networked games: a QoS-sensitive application for QoS-insensitive users?* In Proc. of the ACM SIGCOMM workshop on Revisiting IP QoS, pages 141–147. ACM Press, 2003.

[Horn 99] U. Horn, K. W. Stuhlmller, M. Link & B. Girod. *Robust Internet Video Transmission Based on Scalable Coding and Unequal Error Protection.* Image Communication, vol. Special Issue on Real-time Video over the Internet, no. 15(1-2), pages 77–94–595, 1999.

[Humphreys 02] Greg Humphreys, Mike Houston, Ren Ng, Randall Frank, Sean Ahern, Peter D. Kirchner & James T. Klosowski. *Chromium: a stream-processing framework for interactive rendering on clusters.* ACM Trans. Graph., vol. 21, no. 3, pages 693–702, 2002.

[IEEE] IEEE. 802.11 standard. World Wide Web, `http://standards.ieee.org/getieee802/`.

[Insley 97] Joseph Insley, Daniel Sandin & Thomas DeFanti. *Using Video to Create Avatars in Virtual Reality.* In Visual Proceedings of the 1997 SIGGRAPH Conference, page 128, Los Angeles CA, 1997.

[ITU.T] ITU.T. H.263 : Video coding for low bit rate communication. International Telecommunication Union, `http://www.itu.int/rec/T-REC-H.263/en`.

[Jehaes 03] Tom Jehaes, Danny De Vleeschauwer, Toon Coppens, Bart Van Doorselaer, Eva Deckers, W. Naudts, K. Spruyt & R. Smets. *Access network delay in networked games.* In Proc. of the 2nd workshop on Network and system support for games, pages 63–71, 2003.

[Jehaes 04a] Tom Jehaes, Peter Quax & Wim Lamotte. *Analysis of Scalable Data Streams for Representations in Networked Virtual*

Environments. In Proceedings of the 2004 Workshop on Network and System Support for Games (NETGAMES 04), 2004.

[Jehaes 04b] Tom Jehaes, Peter Quax, Patrick Monsieurs & Wim Lamotte. *Hybrid representations to improve both streaming and rendering of dynamic networked virtual environments*. In Proceedings of the 2004 International Conference on Virtual-Reality Continuum and its Applications in Industry (VRCAI2004), pages 26–32. ACM Press, 2004.

[Jehaes 05] Tom Jehaes, Peter Quax & Wim Lamotte. *Adapting a Large Scale Networked Virtual Environment for Display on a PDA*. In Proceedings of the International Conference on Advances in Computer Entertainment, pages 217–220. ACM Press, 2005.

[Joslin 00] Chris Joslin, Hyewon Seo, Christian Lefevre1, Wonsook Lee, Nadia Magnenat-Thalmann, Maja Jovovic, Stephane Rougeot, Joaquim Esmerado & Daniel Thalmann. Distance communication using networked virtual collaborative environments. Swiss Priority Programme for Information and Communications Structures, 2000.

[Kuhne 99] Gerald Kuhne & Christoph Kuhmunch. *Transmitting MPEG-4 video streams over the Internet: problems and solutions*. In MULTIMEDIA '99: Proceedings of the seventh ACM international conference on Multimedia (Part 2), pages 135–138, New York, NY, USA, 1999. ACM Press.

[Lea 97] Rodger Lea, Yasuaki Honda, Kouichi Matsuda, Olof Hagsand & Mrten Stenius. *Issues in the design of a scalable shared virtual environment for the Internet*. In Proceedings of HICSS'97, January 1997.

[Lei 03] Zhijun Lei & Nicolas Georganas. *Video Transcoding Gateway for Wireless Video Access*. In Proc. IEEE Canadian Conference on Electrical and Computing Engineering (CCECE'03), Montreal, May 2003.

[Liesenborgs 01] Jori Liesenborgs, Wim Lamotte & Frank Van Reeth. *Voice over IP with JVOIPLIB and JRTPLIB*. In Proceedings of

the 26th Annual IEEE Conference on Local Computer Networks, pages 346–347, 2001.

[Liesenborgs 02a] Jori Liesenborgs, Peter Quax, Wim Lamotte & Frank Van Reeth. *Designing a Virtual Environment for Large Audiences.* Lecture Notes in Computer Science, vol. Information Networking, Wireless Communication Technologies and Network Applications, pages 585–595, 2002.

[Liesenborgs 02b] Jori Liesenborgs, Peter Quax, Wim Lamotte & Frank Van Reeth. *Designing a Virtual Environment for Large Audiences.* In Proceedings of the 16th International Conference on Information Networking (ICOIN), pages 3A–2.1–3A–2.10, 2002.

[Linden Labs 03] Linden Labs. Second Life. World Wide Web, `http://www.secondlife.com`, 2003.

[Liu 95] Cging-Gung Liu. A scalable reliable multicast protocol. PhD Dissertation Proposal, 1995.

[Liu 01] Zicheng Liu, Zhengyou Zhang, Chuck Jacobs & Michael Cohen. *Rapid modeling of animated faces from video.* The Journal of Visualization and Computer Animation, vol. 12, no. 4, pages 227–240, 2001.

[Macedonia 94] M. R. Macedonia, M. J. Zyda, D. R. Pratt, P. T. Barham & S. Zeswitz. *NPSNET: A Network Software Architecture for Large-Scale Virtual Environment.* Presence, vol. 3, no. 4, pages 265–287, 1994.

[Macedonia 95a] M. Macedonia, M. Zyda, D. Pratt, D. Brutzman & P. Barham. *Exploiting Reality with Multicast Groups: A Network Architecture for Large-scale Virtual Environments.* IEEE Computer Graphics and Applications, vol. 15, no. 5, pages 38–45, 1995.

[Macedonia 95b] Michael R. Macedonia, Donald P. Brutzman, Michael Zyda, David R. Pratt, Paul T. Barham, John Falby & John Locke. *NPSNET: A Multi-Player 3D Virtual Environment over the Internet.* In Proceedings of the Symposium on Interactive 3D Graphics, pages 93–94, 210, 1995.

[Macedonia 97] Michael R. Macedonia & Michael J. Zyda. *A Taxonomy for Networked Virtual Environments*. IEEE MultiMedia, vol. 4, no. 1, pages 48–56, – 1997.

[Matijasevic 97] Maja Matijasevic. *A Review of Networked Multi-User Virtual Environments*. Technical report TR97-8-1, The Center for Advanced Computer Studies. Virtual Reality an Multimedia Laboratory. The University of Southwestern Louisiana. Lafayette, LA, USA., 1997.

[Mitchell 03] Keith Mitchell, Duncan McCaffery, George Metaxas, Joe Finney, Stefan Schmid & Andrew Scott. *Six in the city: introducing Real Tournament : a mobile Ipv6 based context-aware multiplayer game*. In Proceedings of the 2nd Workshop on Network and System Support for Games, 2003.

[Monsieurs 05] Patrick Monsieurs, Maarten Wijnants & Wim Lamotte. *Client-Controlled QoS Management in Networked Virtual Environments*. In Proceedings of the Fourth International Conference on Networking, pages 268–276, 2005.

[Morse 96] Katherine L. Morse. *Interest Management in Large-Scale Distributed Simulations*. Technical report Irvine Technical Report TR 96-27, 1996.

[Mythic 96] Mythic. Dark age of camelot. World Wide Web, `http://www.everquest.com`, 1996.

[NIST a] NIST. Nistnet. National Institute of Standards and Technology, `http://snad.ncsl.nist.gov/itg/nistnet`.

[NIST b] NIST. Wireless ad-hoc networks bibliography. National Institute of Standards and Technology, `http://w3.antd.nist.gov/wctg/manet/manet_bibliog.html`.

[Obraczka 98] K. Obraczka. *Multicast transport protocols: a survey and taxonomy*, 1998.

[Ogi 00] Tetsuro Ogi, Toshio Yamada, Ken Tamagawa & Michitaka Hirose. *Video Avatar Communication in a Networked Virtual Environment*. In Proceedings of the 10th Annual Internet Society Conference, volume Electronic edition, 2000.

[Pantel 02] Lothar Pantel & Lars C. Wolf. *On the impact of delay on real-time multiplayer games.* In Proc. of the 12th int. workshop on network and operating systems support for digital audio and video, pages 23–29. ACM Press, 2002.

[Pryce 97] Nat Pryce. *Group Management and Quality of Service Adaptation in Distributed Virtual Environments.* In Proceedings of the Fourth UK VR-SIG Conference, Brunel University, Uxbridge, UK, 1997.

[Quax 03] Peter Quax, Tom Jehaes, Pieter Jorissen & Wim Lamotte. *A multi-user framework supporting video-based avatars.* In Proceedings of the 2nd workshop on Network and system support for games, pages 137–147. ACM Press, 2003.

[Quax 04a] Peter Quax, Chris Flerackers, Tom Jehaes & Wim Lamotte. *Scalable Transmission of Avatar Video Streams in Virtual Environments.* In Proceedings of the 2004 IEEE International Conference on Multimedia and Expo (ICME). IEEE, 2004.

[Quax 04b] Peter Quax, Patrick Monsieurs, Tom Jehaes & Wim Lamotte. *Performance Evaluation of Client-Side Video Stream Quality Selection using Autonomous Avatars.* In Proceedings of the 2004 International Conference on Advances in Computer Entertainment Technology (ACE 2004), pages 251–256. ACM Press, 2004.

[Quax 04c] Peter Quax, Patrick Monsieurs, Tom Jehaes & Wim Lamotte. *Using Autonomous Avatars to Simulate a Large-Scale Multi-User Networked Virtual Environment.* In Proceedings of the 2004 International Conference on Virtual-Reality Continuum and its Applications in Industry (VRCAI2004), pages 88–94. ACM Press, 2004.

[Quax 04d] Peter Quax, Patrick Monsieurs, Wim Lamotte, Danny De Vleeschauwer & Natalie Degrande. *Objective and Subjective Evaluation of the Influence of Small Amounts of Delay and Jitter on a Recent First Person Shooter Game.* In Proceedings of the 2004 Workshop on Network and System Support for Games (NETGAMES 04), 2004.

[Quax 04e] Peter Quax, Patrick Monsieurs, Wim Lamotte, Danny De Vleeschauwer & Natalie Degrande. *Objective and Subjective Evaluation of the Influence of Small Amounts of Delay and Jitter on a Recent First Person Shooter Game.* Technical report MOVE D2.3, Limburgs Universitair Centrum - Alcatel Bell NV., 2004.

[Quax 05a] Peter Quax, Tom Jehaes, Maarten Wijnants & Wim Lamotte. *Mobile Extensions for a Multi-User Framework Supporting Video-Based Avatars.* In Proceedings of the International Conference on Internet and Multimedia Systems, and Applications (IMSA). Electronic proceedings. Acta Press, 2005.

[Quax 05b] Peter Quax, Maarten Wijnants, Tom Jehaes & Wim Lamotte. *Bridging the Gap between Fixed and Mobile Access to an Audio/Video-Enabled Large-Scale NVE.* In Proceedings of the International Conference on Web Technologies, Applications and Services (WTAS). Electronic proceedings. Acta Press, 2005.

[Quax 06] Peter Quax, Bjorn Geuns, Tom Jehaes, Wim Lamotte & Gert Vansichem. *On the Applicability of Remote Rendering of Networked Virtual Environments on Mobile Devices.* In Proceedings of the International Conference on Systems and Network Communication. Electronic proceedings. IEEE, 2006.

[Radha 01] H. Radha, M. Van der Schaar & Y. Chen. *The MPEG-4 Fine Graned Scalable Video Coding Method for Multimedia Streaming Over IP.* IEEE/ACM Transactions on Multimedia, vol. 3, no. 1, pages 53–67, march 2001.

[Rajan 02] Vivek Rajan, Satheesh Subramanian, Damin Keenan, Andrew Johnson, Daniel Sandin & Thomas Defanti. *A Realistic Video Avatar System for Networked Virtual Environments.* In Proceedings of IPT 2002, Orlando, FL, 2002.

[Reynolds 87] C. W. Reynolds. *Flocks, Herds, and Schools: A Distributed Behavioral Model.* In Computer Graphics, 21(4) (SIGGRAPH '87 Conference Proceedings), pages 25–34, 1987.

[RFC 1301] RFC 1301. Multicast transport protocol. World Wide Web, http://www.eecis.udel.edu/~mills/database/rfc/rfc1301/.

[RFC 1305] RFC 1305. Network time protocol. World Wide Web, http://www.eecis.udel.edu/~mills/database/rfc/rfc1305/.

[RFC 2002] RFC 2002. Ip mobility support. World Wide Web, http://www.ietf.org/rfc/rfc2002.txt.

[RFC 3775] RFC 3775. Mobility support in ipv6. World Wide Web, http://www.ietf.org/rfc/rfc3775.txt.

[Roehl 95] Bernie Roehl. *Distributed Virtual Reality – An Overview*. World Wide Web, http://ece.uwaterloo.ca/~broehl/distrib.html, 1995.

[Roehl 97] Bernie Roehl. *Channeling the Data Flood*. IEEE Spectrum, vol. 34, no. 3, pages 32–38, 1997.

[Sheldon 03] Nathan Sheldon*et al.* *The effect of latency on user performance in Warcraft 3*. In Proc. of the 2nd workshop on Network and system support for games, pages 3–14. ACM Press, 2003.

[Shen 04] Bo Shen, Sung-Ju Lee & Sujoy Basu. *Caching Strategies in Transcoding-enabled Proxy Systems for Streaming Media Distribution Networks*. IEEE Transactions on Multimedia, Special Issue on Streaming Media, vol. 6, no. 2, pages 375–386, April 2004.

[SIG] Bluetooth SIG. Bluetooth roadmap. World Wide Web, http://www.bluetooth.com/news/sigreleases.asp?A=2&PID=1438&ARC=1&ofs=.

[Singh 95] G. Singh, L. Serra, W. PNG, A. Wong & H. Ng. *BrickNet: sharing object behaviors on the Net*. In Proceedings of the Annual International Virtual Reality Symposium, pages 365–379, 1995.

[Singhal 99] Sandeep Singhal & Michael Zyda. Networked virtual environmnets: Design and implementation. Addison-Wesley Professional, 1999.

[So 97] Oldfield K.Y. So & John C.S. Lui. *Issues in a Very Large Scale Distributed Virtual Environment.* Technical report RM1026-TR97-1205, Department Of Computer Science and Engineering, The Chinese University of Hong Kong, 1997.

[SOE 97] SOE. Everquest. World Wide Web, `http://www.everquest.com`, 1997.

[There] There. There.com. World Wide Web, `http://www.there.com`.

[Vaghi 99] Ivan Vaghi, Chris Greenhalgh & Steve Benford. *Coping with Inconsistency due to Network Delays in Collaborative Virtual Environments.* In Proceedings of the ACM symposium on Virtual reality software and technology, pages 42–49, 1999.

[Wang 97] R.S. Wang & Y. Wang. *Facial feature Extraction and tracking in video sequences.* In IEEE International Workshop on Multimedia Signal Processing, pages 223–238, 1997.

[Waters 96a] R. Waters, D. Anderson & J. Barrus. *Diamond Park and Spline: A Social Virtual Reality System with 3D Animation, Spoken Interaction, and Runtime Modifiability.* TR 2, MERL, 1996.

[Waters 96b] Richard C. Waters. *Time Synchronization in Spline.* Technical report TR1996-009, Mitsubishi Electric Research Laboratories, USA., 1996.

[Whetten 94] B. Whetten, T. Montgomery & S. Kaplan. *A High-Performance Totally Ordered Multicast Protocol.* Lecture Notes in Computer Science, vol. 938, 1994.

[Wijnants 05a] Maarten Wijnants, Patrick Monsieurs & Wim Lamotte. *Improving the User Quality of Experience by Incorporating Intelligent Proxies in the Network.* Technical report TR-LUC-EDM-0605, Expertise Centre for Digital Media (EDM); http://research.edm.luc.ac.be/mwijnants/, April 2005.

[Wijnants 05b] Maarten Wijnants, Patrick Monsieurs, Peter Quax & Wim Lamotte. *Exploiting Transcoding to Increase the User Qual-*

ity of Experience in Networked Applications. In Proceedings of the International Workshop on Advanced Architectures and Algorithms for Internet DElivery and Applications (AAA-IDEA 2005), 2005.

[World] GSM World. Gprs standard. World Wide Web, `http://www.gsmworld.com/technology/gprs/intro.shtml`.

[Yoon 00] I. Yoon & U. Neumann. *Web-Based Remote Rendering with IBRAC (Image-Based Rendering Acceleration and Compression).* Computer Graphics Forum, vol. 19, no. 3, 2000.

[Yura 99] S. Yura, T. Usaka & K. Sakamura. *Video avatar: Embedded video for collaborative virtual environment.* In Proceedings of the IEEE International Conference on Multimedia Computing and Systems, volume 2, page 433, 1999.

[Zhang 06] Hong-Ke Zhang, Bing-Yi Zhang & Bo Shen. *An efficient dynamic multicast agent approach for mobile IPv6 multicast.* Academic Open Internet Journal, vol. 17, 2006.

[Zyda 97] Michael Zyda. *NPSNET-IV: inserting the human into the networked synthetic environment.* In SIGGRAPH '97: ACM SIGGRAPH 97 Visual Proceedings: The art and interdisciplinary programs of SIGGRAPH '97, page 280, New York, NY, USA, 1997. ACM Press.