Issues in Large Scale Collaborative Virtual Environments

By

© Jauvane Cavalcante de Oliveira

A thesis submitted to the University of Ottawa in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Ottawa-Carleton Institute of Electrical and Computer Engineering
School of Information Technology and Engineering
University of Ottawa

Ottawa, Ontario, Canada

October 2001

Abstract

Collaborative Virtual Environments (CVE) are virtual reality spaces that enable participants to collaborate and share objects, as if physically present in the same place. CVE concepts have been used in many systems in the past few years. Applications of such technology range from military combat simulations to various civilian commercial applications. These collaboration spaces have strict performance requirements. Today, there are many such systems developed specifically for collaboration. At the same time, some relatively new standards that address multiuser virtual environments and shared spaces have become available; however, most of these standards have been developed assuming that a small number of users would be interacting at a given time. The architectures available today provide support for a modest number of users but they fail if too many users are "present" together in a small "space" in the Virtual World. In this work, we first evaluate the currently available standards for the case of a very large number of users. An Adaptive Hybrid Architecture for VEry Large Virtual EnvironmenTs (VELVET) is then introduced. VELVET allows a large number of users to interact in a CVE. It also supports small groups of users, but it is in the large environment case that shows its greatest potential. VELVET introduces a novel adaptive area of interest management, which supports heterogeneity amongst the various participants. That allows users in a supercomputer with a high-speed networking connection to successfully collaborate with others in not-so-powerful systems behind a slow dial-up connection.

In order to make a Collaborative Virtual Environment more interesting to users, it is possible to "stitch" together copies of areas which users may have interest in from one Virtual World into another. This procedure augments the physical size of a Virtual World, and creates a potentially larger number of users within the World, first because of the "embedding" and second since the added attractions may work as an incentive for some more users to join the "embedded" World. On the other hand this procedure brings up a series of problems related with consistency, which are also addressed in the thesis. We introduce a methodology which ensures that all copies of a given area of a World are kept consistent among them, as well as with the original world. We also apply this methodology in VELVET, as well as in other Architectures. Additionally, we introduce other approaches to be used when a less strict consistency model is sufficient.

Acknowledgments

First and foremost, I'd like to thank Dr. Nicolas D. Georganas, who has been the perfect supervisor, creating a great work environment at our lab and also allowing me to work at home in my unconventional time shift. He is always available for discussions and has been a true source of inspiration. More than a supervisor, Dr. Georganas has been a great friend, along with his wife Jacynthe and sons Nikita and Emmanuel;

I would also like to thank my wife Carla, for the support, patience, affection and understanding during our lengthy stay in Ottawa. She never complained much when I was awake through the night with the noisy computers at work;

I must also thank Dr. Luiz Fernando Gomes Soares, my Masters supervisor, who gave me great encouragement to pursue advanced studies abroad;

I would like to express my gratitude to my parents Mrs. Jaudelice and Mr. Santos de Oliveira, who have always encouraged us to follow through challenges. I am also indebted to my relatives, specially my sister Jaudelice who visited us while in Ottawa twice and my brother Jáuber who helped me through the application process;

I would like to thank all members of the MCRLab family, for the great and relaxed work environment and precious discussions. Special thanks to Dr. Seok Jong Yu, for the helpful discussions at the last stages of this work and François Malric for the always friendly support. I would like to extend that to the staff of the University of Ottawa for the always friendly and relaxed environment.

I would like to express my gratitude to a number of friends, specially Fernando Carvalho and Rossana Andrade, who helped me preparing my documentation to apply for academic positions in Brazil (the next step). I would like to extend that to a number of friends (too many to list) from Ottawa, Rio de Janeiro and Fortaleza for all good moments and relaxed chat during the last few years;

Last but not least, I would like to thank CAPES, the Brazilian Ministry of Education Agency, for the financial support provided through a comprehensive scholarship. I would also like to thank the Ontario Research and Development Challenge Fund for the financial support provided. Without these financial support this work wouldn't have ever started. From CAPES I'd like to thank specially Mrs. Silvia Velho for the trust deposited on me and Mrs. Gláucia Gusmão for an efficient and friendly support.

Index

List of Figures

List of Tables

List of Acronyms

AICI / AICO – Area of Interest Check-In / Area of Interest Check-Out

AoI – Area of Interest

CI / CO – Check-In / Check-Out

COR – Collision of Ownership Requests

CVE / DVE – Collaborative Virtual Environment / Distributed Virtual Environment

DIVE – Distributed Interactive Virtual Environment

DIS – Distributed Interactive Simulation

HLA – High Level Architecture

ISTP – Interactive Sharing Transfer Protocol

JNI – Java Native Interface

LSVE – Large Scale Virtual Environments

LTC – Locale Transmission Channel

LVW – Local Virtual World

LW – Living Worlds

MASSIVE – Model, Architecture and System for Spatial Interaction in Virtual Environments

OC – Open Community

OMS – Ownership Management Subsystem

OTC – Object Transmission Channel

PDU – Protocol Data Unit

PVW – Parallel Virtual World

RTI – Runtime Infrastructure

RTP – Real Time Protocol

RVW – Remote Virtual World

RVWA – Remote Virtual World Agent

SIMNET – Simulation Network

SPLINE – Scalable Platform for Large Interactive Networked Environments

VRML – Virtual Reality Markup Language

VW – Virtual Worlds

Chapter I

1. Introduction

Human tele-interaction has become a very important feature in the last few years, as people around the world need to collaborate. Such issue has been historically addressed by Computer Supported Cooperative Work (CSCW) systems, such as multimedia conferencing [Oliv96]. The term “Virtual Reality” (VR) was initially introduced in the late 80’s. At the time other terms also used were “Artificial Reality” and “Cyberspace” [Beie01]. VR denotes an artificial representation of a world. A user may interact with such world through his/her “avatar”. An avatar is a graphical representation of a user in a Virtual World. An avatar allows the transmission of non-verbal cues which are common to human beings. Virtual Reality has been evolving through exploitation of interaction of a single user with a virtual environment. A few years ago, with the advent of commonly available hardware being able to handle the load of such systems, some research institutes started exploiting Collaborative Virtual Environments (CVE), where one user interacts, through a virtual world, with other users via their avatars. A CVE is a special case of a VR system where the emphasis is more on collaboration between users rather than on stand-alone simulation. CVEs are used for applications such as collaborative design, training [OlSG00b, OHSC00], telepresence, tele-robotics and many others.

Many of the applications may have potentially a very large number of users at a time and that can easily overload a fast network, as well impose huge processing requirements at the user stations. As computing resources are limited, there are obvious problems which arise once the number of users in a simulation raises beyond a certain limit. In fact, if no special mechanisms are provided, one may expect a simulation to produce undesirable effects such as choppy rendering, loss of interactivity and alike, due to lack of processing power to handle the ever increasing load. Another problem, which a CVE faces, is that of heterogeneity of hardware available for end users. That also imposes some limitations to a CVE as users in very powerful systems and fast networks would need to collaborate with others with very limited hardware and networking. It is obvious that the second type of systems should not be required to deal with the same load as the first. The easiest way to control such problems is either by deploying the simulation based on the slowest and weakest system or by setting up a minimum requirement and simply denying access to non-conforming systems. The first approach, while guaranteeing functionality and availability, would greatly under-utilize better systems; even worse, more limited systems may join the session at any point in time. The second approach denies access to a potentially large number of users and may still lead to under-utilization of some systems.

We have designed and implemented a number of CVEs for industrial training and electronic commerce [OlSG00b, OHSC00]. Such CVEs, while allowing rich collaboration, did not provide means for handling a large number of users. In such prototypes all users are aware of (and receive updates from) every other object in the virtual world, which situation does not scale well.

In this thesis, we present and evaluate VELVET, an Adaptive Hybrid Architecture for VEry Large Virtual EnvironmenTs. VELVET addresses the issues mentioned above, allowing a virtually unlimited number of users to participate in a CVE while allowing users, with heterogeneous hardware and available networking, to collaborate the best they can. We also present a number of approaches to keep consistency amongst "embedded" Virtual Worlds, where copies of sections of other Worlds are used.

Chapter 2 presents background information, describing some available standards/prototypes, which are used to deploy CVE. Chapter 3 describes prototypes developed for Very Large CVEs. Chapter 4 describes VELVET, our proposed hybrid, and adaptive architecture supporting very large CVEs. Chapter 5 presents a method which would allow a Large-Scale Virtual Environment (LSVE) through embedding of content from existing CVEs into other CVEs. A number of novel approaches for keeping total or partial consistency amongst the various copies of such Worlds are also introduced. Chapter 6 concludes the thesis.

The main contributions of this thesis are:

- Proposal of VELVET, an Architecture to handle VEry Large Virtual EnvironmenTs. This architecture has been submitted for publication in [OlGe02].
    - VELVET gracefully supports Heterogeneous Systems
        - Degree of Blindness Concept
        - Adaptive mechanism which allows each participant to receive as much as possible (or requested) of the virtual environment.

  - VELVET defines a novel Area of Interest Management approach, through the concept of Parallel Virtual World.

- Parallel Virtual World concept, where the Virtual World is organized based on metric, rather than Euclidean distance, allowing one to see what matters the most.

- Definition of parameters ∂ and ρ, as well as relations $\subset$ and $\supset$.

- Proposal of two schemes for Consistency Control in Embedded Worlds. A methodology for Full Consistency control, as well as Partial Consistency control of Embedded Worlds. Such novel Consistency Control schemes are introduced in our paper [OlYG02] submitted for publication. They are built upon common features of existing CVE architectures, so that the results are also applicable to other architectures. We further discuss a Multiarchitectural Consistency Control scheme.

Research publications during this doctoral study:

- Oliveira, J. C., Yu, S. J., & Georganas, N. D. (2002) Enabling Embedded Worlds in Very Large Virtual Environments, *IEEE Computer Graphics and Applications* (Submitted).

- Oliveira, J. C., Hosseini, M., Shirmohammadi, S., Malric, F., Nourian, S., & Georganas, N. D. (2002) Java Multimedia Telecollaboration, *ACM Multimedia Magazine* (Submitted).

- Oliveira, J. C. & Georganas, N. D. (2002) VELVET: An Adaptive Hybrid Architecture for VEry Large Virtual EnvironmenTs, *Presence: Teleoperators and Virtual Environments* (Submitted).

- Oliveira, J. C., Malric, F., Yang, D., Nourian, S., & Georganas, N. D. (2001) Java Multimedia Telecollaboration, *Technical Demo at ACM Multimedia 2001*, Ottawa, ON, Canada.

- Oliveira, J. C., Shen, X., & Georganas, N. D. (2000) Collaborative Virtual Environment for Industrial Training and e-Commerce, *Invited Paper, IEEE VRTS'2000 (Globecom'2000 Conference's Workshop on Application of Virtual Reality Technologies for Future Telecommunication Systems)*, San Francisco, CA, USA.

- Oliveira, J. C., Hosseini, M., Shirmohammadi, S., Cordea, M., Petriu, E., Petriu, D. & Georganas, N. D. (2000) VIRTUAL THEATER for Industrial Training: A Collaborative Virtual Environment, *Proc. 4th WORLD MULTICONFERENCE on Circuits, Systems, Communications & Computers (CSCC 2000)*, Vouliagmeni, Greece.

- Oliveira, J. C., Shirmohammadi, S., & Georganas, N. D. (2000) A Collaborative Virtual Environment for Industrial Training, *IEEE Virtual Reality 2000*, New Brunswick, USA.

- Oliveira, J. C., Shirmohammadi, S., & Georganas, N. D. (1999) Collaborative Virtual Environment Standards: A Performance Evaluation, *IEEE DiS-RT'99 (Distributed Interactive Simulation and Real Time Applications)*, Greenbelt, MD, USA.

- Shirmohammadi, S., Oliveira, J. C., & Georganas, N. D. (1998) Applet-Based Multimedia Telecollaboration: A Network-Centric Approach, *IEEE Multimedia Magazine*, 5(2).

- Shirmohammadi, S., Oliveira, J. C., & Georganas, N. D. (1998) Implementation and Management of Web-Based Collaboration Using Java, *Canadian Conference on Broadband Research (CCBR'98)*, Ottawa, ON, Canada.

- Shirmohammadi, S., Oliveira, J. C., & Georganas, N. D. (1998) Java-Based Multimedia Collaboration: Approaches and Issues, *Invited Paper, International Conference on Telecommunications (ICT'98)*, Porto Carraras, Greece.

Chapter II

2. Background

In the early 90's, Virtual Reality (VR) was considered as a leading edge technology which required very expensive stations and special peripherals, such as data-gloves, Head Mounted Displays (HMD), Shutter Glasses, etc. At that time VR was mostly used to allow a user (alone) to interact with a virtual world. After this stage was reasonably accomplished, some research groups started looking at collaboration aspects, i.e. how should users be allowed to interact not only with the world but also among themselves in the Virtual world. Since then, many prototypes and/or standards have been developed. In this section we will have a look at these technologies, which include rendering facilities such as:

- OpenGL;
- DirectX;
- Virtual Reality Modeling Language (VRML);
- Java3D

We will then discuss standards (and proposed standards) for CVE, including:

- Living Worlds;
- DIS;

- Open Community;
- High Level Architecture (HLA)

2.1 Virtual Reality Renderers

2.1.1. OpenGL

OpenGL™ is a graphical library introduced by SGI (formerly Silicon Graphics) in 1992 to allow developers to write code once, based on the OpenGL API, which is supposed to run in various platforms (as long as they have an OpenGL library implementation). Since its inception, OpenGL has been controlled by an Architectural Review Board [OpenGL] whose representatives are from the following companies: 3DLabs, Compaq, Evans & Sutherland, Hewlett-Packard, IBM, Intel, Intergraph, NVIDIA, Microsoft, and SGI.

Most of the Computer Graphics research (and implementations) broadly uses OpenGL, which has become a de facto standard. Virtual Reality is no exception to this rule. The main goals of OpenGL are:

- Open Standard: OpenGL is now driven by an independent consortium, the OpenGL Architecture Review Board previously mentioned, with broad industrial support, which ensures a vendor-neutral, multiplatform, graphics development of the standard.
- Reliability and Portability: All OpenGL implementations allow a consistent presentation of graphics in multiple platforms, from consumer electronics to PCs, workstations, and supercomputers.

OpenGL provides many graphic functions in its API, grouped in the following categories: Accumulation buffer, Alpha blending, Anti-aliasing, Automatic rescaling of vertex normals, BGRA pixel formats and packed pixel formats [OpenGL], Color-index mode, Display list, Double buffering, Feedback, Gouraud shading, Immediate mode, Level of detail control, Materials lighting and shading, Pixel operations, Polynomial evaluators, Geometric Primitives, Raster primitives, RGBA mode (an image format [OpenGL]), Selection and picking, Specular Highlights, Stencil planes, Texture coordinate edge clamping, Texture mapping, Three Dimensional Texturing, Transformation, Vertex array enhancements and Z-buffering.

There are many options available for hardware acceleration of OpenGL based applications. The idea is that some complex operations may be performed by specific hardware (an OpenGL accelerated video card such as those based on 3DLab's Oxygen series or nVidia's GeForce chipset for instance) instead of the general CPU which is not optimized for such operations. Such acceleration allows low-end workstations to perform quite well yet at a low cost.

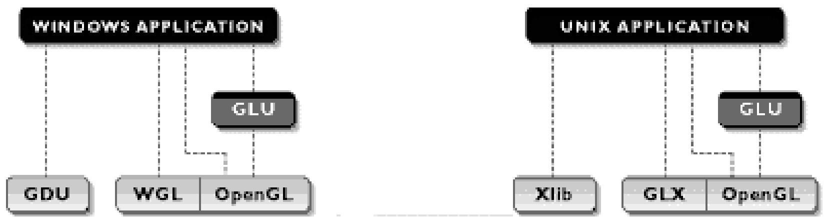

Figure 1: OpenGL API Hierarchy

Figure 1 shows how OpenGL interacts with native applications under Windows and Unix.

We will see that such is the importance of OpenGL that both VRML and Java3D (described below) are built on top of it, i.e. if a given workstation has hardware support for OpenGL, the VRML browser and Java3D, discussed below, will also benefit from it.

2.1.2. DirectX

Microsoft DirectX® [DirectX] is a group of technologies designed by Microsoft to allow Windows-based computers to run and display applications rich in multimedia elements such as full-color graphics, video, 3-D animation, and surround sound. DirectX is an integral part of Windows 98 and Windows 2000, as well as Microsoft® Internet Explorer 4.0 and higher. DirectX components may also be installed in Windows 95 as an optional package.

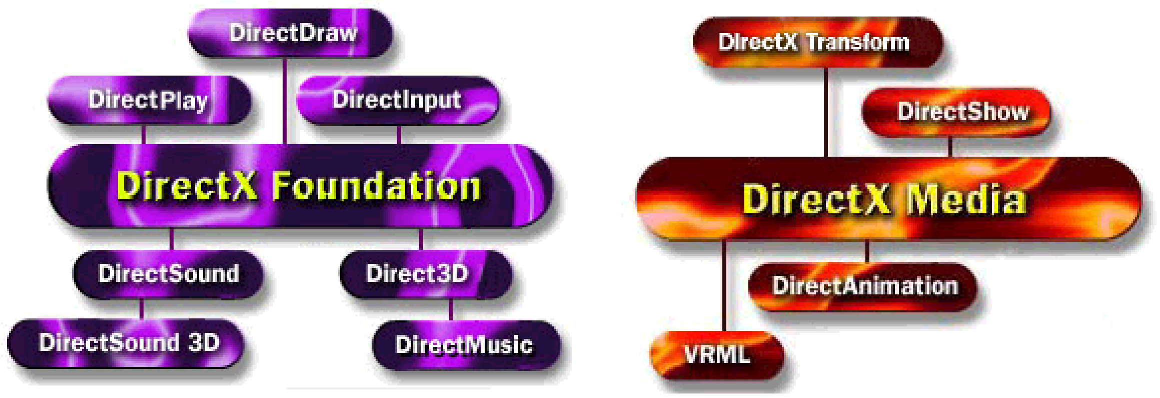

Figure 2: Two of Microsoft's DirectX Layers

DirectX allows a compliant application to run in any Windows based system, independent of particularities of the hardware of each system. In some sense, it seems similar to OpenGL however there is a logical limitation in availability, as it is a Windows specific component. DirectX accomplishes its task via a multilayered structure as shown in Figure 2. The Foundation layer is responsible for resolving any hardware dependent issue. DirectX also allows developers to deploy the creation and playback of multimedia content via DirectX's Media layer. A third layer, Component, completes the high-level protocol layer stack.

We will see that some VRML browsers also provide a Direct3D based version (as well as the common OpenGL). A good example is blaxxun's Contact 4.0[1].

2.1.3. VRML

2.1.3.1 Historic

In 1994, during the First International Conference on the World Wide Web, some concerns about the specification of a 3D-scene description language were raised after the presentation of Labyrinth, a prototype three-dimensional interface to the Web.

Just after the conference, work on the VRML specification began. An electronic mailing list was set, aiming at facilitating discussion of the specification for VRML. The list members quickly agreed upon the following set of requirements for VRML [VRML]:

- Platform independence;
- Extensibility; and
- Ability to work over low-bandwidth connections.

[1] http://www.blaxxun.com/products/contact/

SGI offered its Open Inventor ASCII file format as the basis for VRML. The Inventor file format supported complete descriptions of 3D scenes with geometry, lighting, materials, 3D user interface widgets, and viewers. It had all of the features that developers needed to create highly interactive 3D applications, as well as an existing tool base with a wide installed presence. A subset of the Inventor File Format, with extensions to support networking, was then the basis of development of VRML 1.0.

In 1995, some experts on the VRML mail list formed the VRML Architecture Group (VAG), which aimed to reach consensus in the VRML community in order to develop a scalable, fully interactive standard for 3D shared worlds.

VAG issued a Request-For-Proposals (RFP) for VRML 2.0. Moving Worlds, one of the proposals, was unanimously agreed as the working document for VRML 2.0.

On August 4, 1996, the official VRML 2.0 Specification was released at SIGGraph 96 in New Orleans.

The VRML97 International Standard was developed by the Joint Technical Committee 1 (JTC 1) of ISO/IEC in partnership with the VRML Consortium. The formal processing of VRML97 to become an International Standard began in June 1996 with the Consortium's VRML 2.0 draft specification. VRML has been approved as an International Standard in a record 18 months period.

Nowadays there are many VRML browsers available, almost all of them implemented via OpenGL; however some also allow one to use a Direct3D (part of MS DirectX) version. SGI had created the Cosmo Player (a free VRML browser based on OpenGL) as well as other products such as Cosmo Worlds, which support the creation of Virtual Worlds that are VRML compliant. A few other companies provide today VRML based solutions, such as blaxxun with its blaxxunContact (a VRML browser available in OpenGL and Direct3D versions) as well as server side technology for collaboration. Even though there are several VRML browsers for Windows based workstations, lack of such applications in some other platforms, such as Linux and some other flavours of Unix and Apple's Macintosh, has been noticed.

2.1.2.2 Basics

The Virtual Reality Modeling Language is the standard modeling scheme for 3D multimedia and shared virtual worlds on the Internet. The file format consists of specification of the geometry and behaviour of a 3D scene.

At the current stage, VRML has been released in its initial version 1.0 and 2.0 (also known as VRML'97). The major characteristics of each one are:

VRML 1.0:

- Standard objects (cube, sphere, cone, cylinder, text);
- Arbitrary objects (surfaces, linesets, pointsets);
- Ability to fly through, walk through, examine scenes Lights;
- Cameras (viewpoints);

- Textures on objects;
- Clickable links;
- Define and reuse objects.

VRML 2.0:

- All VRML 1.0 features plus:
- Animated objects;
- Switches;
- Sensors;
- Scripts (Java or JavaScript) for behaviours;
- Interpolators (colour, position, orientation, etc.);
- Extrusions;
- Background colours and textures;
- Sound (.wav and MIDI);
- Animated textures;
- Event routing;
- Define and reuse objects and behaviours and effectively add new nodes to the language with PROTO and EXTERNPROTO.

We will focus our discussion to VRML 2.0, as this is the version which introduces behaviours through Scripts as well as accepts extensions through PROTO and EXTERNPROTO.

Figure 3 shows a pretty simple scene, consisting of the logo of the VRML Consortium. Such scene consists of basic geometry elements, namely a cube a sphere and a cone, using respectively the colours red, green and blue.

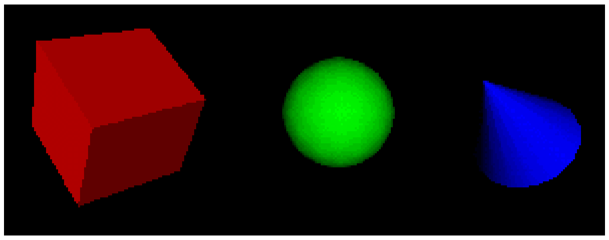

Figure 3: A Simple Scene (VRML Consortium Logo)

The above scene may be created by the following VRML code (Figure 4). It is easy to check the above-mentioned standard objects support.

```
#VRML V2.0 utf8
Transform {
   children [
      Transform {
         translation -3.5 0 1
         rotation 1 0 1 1.8
         children [
            Shape {
               geometry Box {}
               appearance Appearance {
                  material Material { diffuseColor 1 0 0 }
               }
            }
         ]
      }
      Transform {
         translation 0.5 0 1
         children [
            Shape {
               geometry Sphere {}
               appearance Appearance {
                  material Material { diffuseColor 0 1 0 }
               }
            }
         ]
      }
      Transform {
         translation 3.5 0 1
         rotation 1 0 1 1
         children [
            Shape {
               geometry Cone {}
               appearance Appearance {
                  material Material { diffuseColor 0 0 1 }
               }
            }
         ]
      }
   ]
}
```

Figure 4: Source Code for the VRML Consortium Logo

2.1.2.3 Advanced Features

The following shows an example where a ball falls on the floor and bounces back to its original position when the user clicks on it. This shows both interactivity and animated objects, as listed above. Figure 5 shows such world with the described animation.

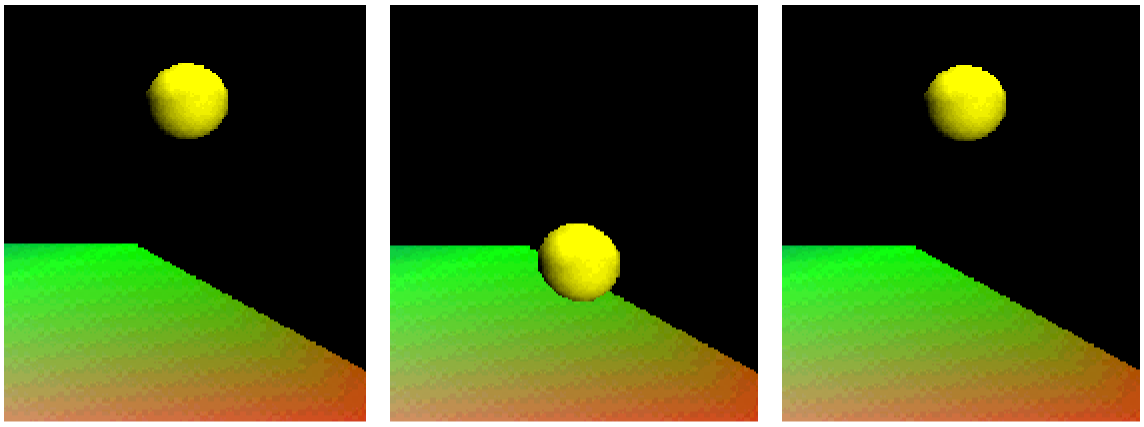

Figure 5: A more complex scene (Bouncing Ball)

The user's click action is captured through a TouchSensor, which is an example of one more element of VRML. Sensors are nodes that generate events from non-VRML inputs, such as a user interaction or the tick of time. The VRML 2.0 sensors are:

- CylinderSensor: Translates user input into a cylindrical motion;
- PlaneSensor: Translates user input into a motion along the X-Y plane;
- ProximitySensor: Detects when the user comes within an area around the given object;
- SphereSensor: Translates user input into a spherical shape;
- TimeSensor: Detects time and gives the scene a sense of time passing (animations);
- TouchSensor: Detects when the user touches a given object;
- VisibilitySensor: Detects if an object is currently visible to the user.

These sensors are the basis for user interaction, as they are the only way to create new events from outside the world. For instance, the proximity event may implement a self-opening door, which opens when the user's avatar is close enough to such door. Any "avatar" is a representation of a user in the Virtual World. It should be controlled by the user's actions. Another example of usability of sensors would be the use of the visibility sensor to reduce processing at the client station. For instance, a lake with fish and ducks could be processed only when the user is looking at it.

This scene also uses one of the main elements of VRML 2.0, named ROUTE, which is used to pass information between nodes in the scene. Unlikely programming languages, VRML does not have function call capabilities. This is why ROUTES are so important. In the bouncing ball example, a message is passed when the user clicks over the ball (TouchSensor), as well as during the motion of the ball. This is shown by the three ROUTE commands at the end of the code (Figure 6). When an object receives a message, it may take the appropriate action, including rerouting the message to some other node. One may point out that this functionality follows the object-oriented paradigm.

```
#VRML V2.0 utf8

Group {
   children  [
      DirectionalLight { direction  0 -1 0 },
      Viewpoint {
         position 0 0 20
         description "Bouncing Ball"
      },
      Shape {
         appearance  Appearance {
            material  Material { }
         }
         geometry IndexedFaceSet {
            coord Coordinate {
              point [10 -5.5 10, 10 -5.5 -10, -10 -5.5 -10, -10 -5.5 10]
            }
            coordIndex [ 0, 1, 2, 3, -1 ]
            color Color { color [ 1 0 0, 0 1 0, 0 0 1, 1 1 1 ] }
            solid FALSE
         }
      },
      Transform {
         translation 8 0 0
         children [
            DEF touchBall Transform {
               children [
                  DEF TOUCH TouchSensor { },
                  Shape {
                     appearance Appearance {
                        material DEF COLOR Material {diffuseColor 1 1 0}
                     }
                     geometry Sphere {}
                  }
               ]
            }
         ]
      },
      DEF TIME TimeSensor {cycleInterval 2.0 },
      DEF ball_pos PositionInterpolator {
         key [ 0, .1, .2, .3, .4, .46, .5, .54, .6, .7, .8, .9, 1 ]
         keyValue [ 0 0 0, 0 -.196 0, 0 -.784 0, 0 -1.764 0, 0 -3.136 0,
                    0 -4.14 0, 0 -4.8 0, 0 -4.14 0, 0 -3.136 0,
                    0 -1.764 0, 0 -.784 0, 0 -.196 0, 0 0 0 ]
      },
   ]

   ROUTE TOUCH.touchTime TO TIME.set_startTime
   ROUTE TIME.fraction_changed TO ball_pos.set_fraction
   ROUTE ball_pos.value_changed TO touchBall.set_translation
}
```

Figure 6: Source code for the Bouncing Ball

If the programmer needs even more control over the world, VRML supports Script, which is another VRML node. Script nodes allow VRML programmers to create new behaviours through scripts written in Java, JavaScript or VRMLScript, which is a subset of JavaScript. When the programmer uses such node, he or she needs to write the node script (in VRML) and the script itself in one of the previously mentioned languages.

2.1.2.4. Extensibility

A language which doesn't allow extensibility is usually not very successful. For instance Java allows the C/C++ extensibility through the Java Native Interface (JNI). VRML, as expected, also allows extensibility through PROTO (prototype) nodes. PROTO defines a new customized node. Once created, a PROTOtype can be used just like any standard VRML node.

In addition to prototypes and scripting, VRML provides the External Authoring Interface (EAI). EAI is a means of interfacing a VRML browser with an external program. So far, a Java API is defined, so it is possible to control the behaviour of the World (running within a VRML Browser, such as Cosmo Player or blaxxunContact) from a Java Applet via EAI. Through EAI, it is possible to establish a connection from the Java Applet to a server, which may facilitate sharing. Such connection may be established either through Java's built-in features or through JNI, i.e. C/C++. Such approach is used in the industrial training prototype described in [OlSG00].

2.1.3 Java 3D

The Java 3D API is a set of classes for writing three-dimensional graphics applications and 3D applets. It gives developers high level constructs for creating and manipulating 3D geometry and for constructing the structures used in rendering that geometry. Like most of the VRML browsers, there are OpenGL and DirectX versions of Java3D, which should be chosen accordingly to the configuration of the workstation. That is, if there is OpenGL hardware acceleration, it is preferable to use the OpenGL version. Otherwise the ActiveX is advisable (of course, this discussion makes sense only for stations where ActiveX is an option, namely Windows. WinNT only supports ActiveX 3, which is pretty much obsolete taking into account the just released version 8 for Win9x, etc.). It is worth to mention that Windows 2000 brought DirectX 7 and over into NT category systems.

Sun Microsystems (the creator of Java and Java3D), along with the Web3D consortium has formed the Java 3D and VRML Working Group, whose goal aims at improving the Java3D←→VRML interoperation as well as providing a Java3D implementation of a VRML browser [VJ3D].

Java3D introduces high-quality graphics support into Java, which makes it quite promising due to its platform independence. Java3D seems to be set to play a key role in the future.

2.2 CVE Middleware

Section 2.1 presented the most important rendering standards available for 3D graphic presentations. As previously mentioned, some of the engines have been built on top of others, namely OpenGL and ActiveX are usually the basis for the others. We are now going to comment on current CVE standards (and proposals for standards) for the communication infrastructure. It will be seen that implementations compliant with each middleware will use one or more of the former rendering engines to display the content of the world to the various users.

2.2.1 Living Worlds

2.2.1.1 Historic

Even before VRML 2.0 was formally launched, a number of new working groups had emerged to push it forward [LW]. One of these, working under the name Living Worlds (LW), brought together a small group of developers who have already implemented proprietary systems for inserting dynamic objects - human-controlled avatars and autonomous bots - into VRML 1.0 scenes and distributing the results in real time over consumer-grade Internet hookups. The goal of the LW group is to distill the experience gained from these systems into a proposal for a VRML 2.0 application framework that would enable VRML developers to populate and share their "Moving Worlds."

The charter of the LW group was to distill its experience with avatar-based interaction in VRML 1.0 into a proposed standard for distributed object interaction in VRML 2.0. The LW effort aimed at a single, narrow, well-defined goal: to define a set of VRML 2.0 conventions that support applications which are both interpersonal and interoperable.

By interpersonal, we mean VRML applications which support the virtual presence of many people in a single scene at the same time: people who can interact both with objects in the scene and with each other. By interoperable, we mean that such applications can be assembled from libraries of components developed independently by multiple suppliers, and visited by client systems which have nothing more in common than their adherence to the VRML 2.0 standard.

Draft 1.0 of the LW specification, dated February 24th, 1997, includes interfaces for:

- ♦ coordinating the position and state of shared objects (including avatars);
- ♦ information exchange between objects in a scene;
- ♦ personal and system security in VRML applications;
- ♦ a (small) library of utilities, and some workarounds for VRML 2.0 limitations;
- ♦ Identifying and integrating at run-time interaction capabilities implemented outside of VRML and its scripts.

Interoperability is a requirement posed by all participants in shared worlds:

- ♦ Users want to be able to take with them their Avatars, and their favourite interactive objects, to different worlds;
- ♦ Component developers want to focus their efforts on building single components (e.g. avatars or portable multi-purpose components, such as dice or playing cards), for which there will be a larger market, as more and more worlds become interoperable;
- ♦ Application developers (i.e., the authors of worlds and complete applications, such as the board game) can combine best-of-breed components to yield richer results with less effort.

2.2.1.2 Development

The VRML consortium has been developing a new proposal which aims to achieve collaboration (sharing) among users of VRML browsers. A first draft of such proposal was released on February 17, 1997, while the second release of the draft is still under development. The idea behind Living Worlds is the definition of VRML 3.0, which would include support for sharing and collaboration.

In the scope of Living Worlds, VRML 2.0 is extended by defining a new set of nodes, through PROTO and EXTERNPROTO (similar to PROTO). The LW basic principles are [VRML, LW]:

- ♦ Build on VRML 2.0: It is built entirely with existing VRML 2.0 mechanisms, defining as "implementation specific" anything that cannot be implemented inside the current standard;

- ♦ Standards, not designs: The focus of the LW effort is to identify the minimum set of system features required to enable shared environments, and to define the minimum set of standard interfaces required to enable application developers to combine feature-sets from multiple suppliers;

- ♦ Architectural Agnosticism: Although all of the LW initiators have based their solutions on architectures that exploit a central server, the standard should not assume any particular architecture.

Work in the Living Worlds consortium has been quite slow.

2.2.2. Open Community

2.2.2.1 Historic

Open Community (OC) [BaWA96b] is a proposal of a standard for multiuser enabling technologies from Mitsubishi Electric Research Laboratories. SPLINE (Scalable Platform for Large Interactive Networked Environments) is an implementation compliant with OC which provides development APIs. Such libraries provide very detailed and essential services for real-time multi-user cooperative applications. SPLINE implementations are available in ANSI C or Java. OC handles issues related to networking, real-time audio transport, application-independent transport of large and complex objects such as VRML models as well as filtering which augments the performance at a user' station.

For its communication, SPLINE uses the Interactive Sharing Transfer Protocol (ISTP) [WaAS97], which is a hybrid protocol supporting many modes of transport for VR data and information through 5 subprotocols, which are described in section 2.3.2.7.

SPLINE partitions the World Model in *Locales* which may have any shape. *Locales* are detailed in section 2.3.2.4.

2.2.2.2 Architecture

OC defines a middleware solution for sharing virtual worlds amongst users. An OC application has the general architecture shown in Figure 7. For the application, everything works just as if all the databases were available locally, as OC provides the entire communication infrastructure which allows an efficient distribution of data (the box in Figure 7 shows the components which are part of Open Community) [OC].

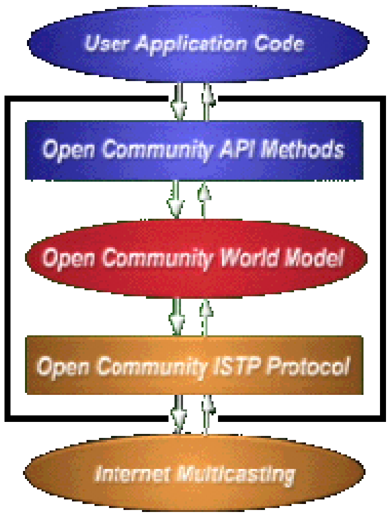
User Application Code
Open Community API Methods
Open Community World Model
Open Community ISTP Protocol
Internet Multicasting

Figure 7: OC Architecture

The user runs an OC application which makes calls on the OC library, exchanging data with other OC processes to create a local version of the World. The dynamic parts of the World are transmitted directly though the network, while parts which rarely change are fetched through HTTP. Once the World Model is complete, it is sent to the rendering software, which is implementation specific (Raw OpenGL, VRML browser, etc.). OC also makes callbacks into the user's code to inform about various events.

2.2.2.3 The World Model

The Open Community's world model contains many objects with diverse properties such as Permanent Objects (e.g., buildings), objects with motion (e.g., an Avatar) and seasonal objects (like an audio clip explaining characteristics of a given object "touched" by a user), amongst others.

An OC object may have subparts organized in a hierarchy in such a way that when the top-level object moves, the full hierarchy moves as well.

Each object in an OC world has three references:

- Parent: This is another OC object, which is above the object in the hierarchical graph of the World Model. When the parent moves, its children move too. The top-level object is the *locale* whose parent reference is null. It is then possible to find out in which *locale* a given object is located by examining its parents (see the *locale* section below).
- Owner: The owner of a given object is a process, possibly remote. Note that it is not another object. The object's owner represents the user who is allowed to modify a given object (see the ownership section below).
- Class: Described by an OC class descriptor, which contains some info about the objects instantiated from it. The class of a method defines which callbacks should be triggered for a given action.

An OC application almost inevitably has to have a loop calling spWMUpdate, which is the function that allows OC to update and control the sharing of worlds. Such function is typically called every 50 to 100 ms.

OC implements automatic cooperation among stations with different update frame rates, hence the developer doesn't need to worry about synchronization in the case of less powerful stations. In fact he or she should try to optimize the frame rate to the local hardware.

2.2.2.4 *Locales*

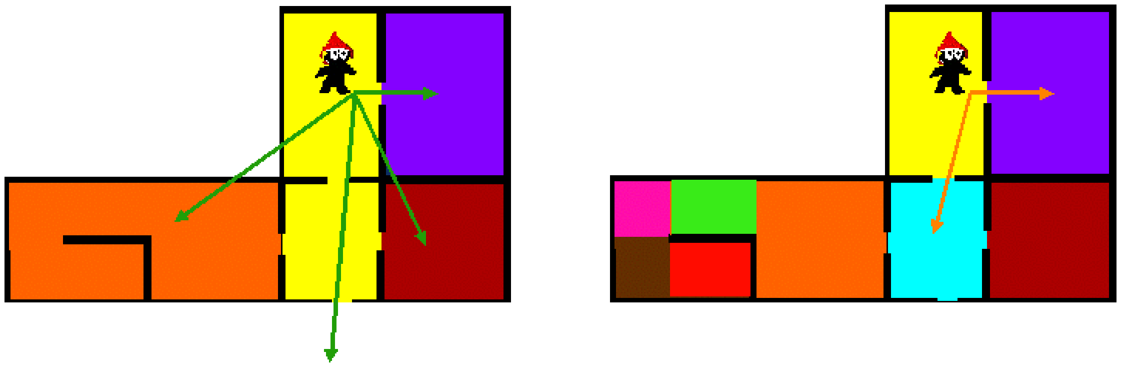

Figure 8: Partitioning a World Model in *Locales*

An Open Community world is partitioned into *Locales*. The general idea behind *Locales* is as follows: If a user is in a *Locale*, only the objects present in that *Locale* and in the directly adjacent ones are visible. As a comparison let us consider a student in a classroom. Using a *Locale*-based architecture, such student would be able to see objects present in the classroom itself, as well as neighbouring areas (e.g. the corridor besides it). In this example the student is not able to see objects in a gymnasium, if that doesn't have direct relationship (neighbourhood) with the classroom *Locale*. The designer of a Virtual World should take this into account while developing such a world. A well-developed world should significantly improve the performance of its visitors, as less data will be required to be processed. This increases the scalability to unlimited size worlds. In order to increase the performance even further, OC uses the built-in hardware of Ethernet cards to instruct them to only interrupt the processing if some relevant message arrives (from some *Locale*, or multicast group, the user is interested on). SPLINE will behave differently depending on how the World Model is partitioned in *Locales*. Figure 8 shows a World Model partitioned in two different ways and one can see that the partition to the right allows a user to receive information from relevant *Locales* only.

Each *Locale* has its own coordinate system, which facilitates scalability as many pieces may be put together without the need for modifications in the coordinate system of each component. When an object moves from one *Locale* to another the developer has two options: Move the object directly into that *Locale* or let OC decide which *Locales* such object best belongs to.

As seen above, Open Community uses the general rule that only the current or adjacent *Locales* are viewable for a given user; however, using spObserver it is possible to be "present" in more than one *Locale* at a time. spObserver is nothing but an indication that a given user is interested in a certain *Locale* and its neighborhood. Internally, a spObserver makes the local OC engine to listen to the multicast group of that *locale* (and the neighbourhood).

2.2.2.5 Visual and Aural Information

The way a user looks at a Virtual World is controlled by a spVisualObserver object. Such object specifies a camera viewpoint, i.e. the user's Point of View of the world.

According to the Open Community *Locales* rule, only objects present into the current *locale*, as well as in the neighborhood, are visible. This is shown in Figure 9.

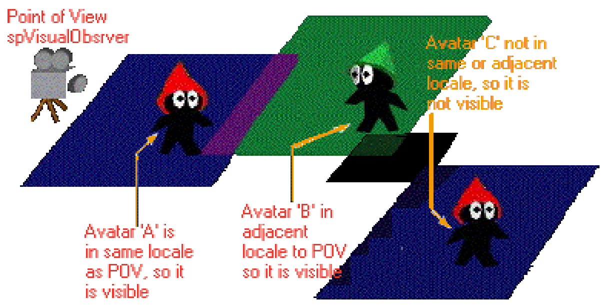
Point of View
spVisualObsrver
Avatar 'C' not in same or adjacent locale, so it is not visible
Avatar 'A' is in same locale as POV, so it is visible
Avatar 'B' in adjacent locale to POV so it is visible

Figure 9: Visibility from a Pont of View

The object spVisualObserver does not have a graphical representation, but it still has attributes such as parent and position, hence it is possible to bundle such object as the child of a given avatar. With such an approach, spVisualObserver will follow the avatar and allow an easy rendering of visual information according to the information retrieved at each spWMUpdate call. It is also easy to conclude that a bird's eye view of the world, or a third-person-over-the-shoulder view, may be accomplished by placing the Visual Observer detached from a user's avatar, respectively, in a position above the scene or a bit behind and higher than the user's avatar. It is also possible for a given user to have "God's eye", when the spVisualObserver would be respectively in a fix location at the *Locale*.

Audio information in Open Community is dealt with as any other object, i.e., it has attributes such as location, parent and so on. A sound source is then placed in a certain location in the world and a URL for an audio file (if not real-time) is also given. A user receives such information according to the location of its spHearing object, when the various audio sources are mixed on the fly and presented to the user taking into account fade due to distance. There are three different kinds of audio sources in a World: Continuously looping sound; Foley sound and real-time sound. The first is usually some background music or other sort of continuous sound. Foley sounds are pretty much understood as special effect sounds, such as the noise of the collision of an avatar with some object in the scene. Finally, real-time sources are sounds usually emitted by users of a virtual world (this may certainly be the audio track of a virtual movie as well).

2.2.2.6 Ownership Control

One of the attributes of an object refers to ownership, which indicates which user (process) is allowed to move or manipulate such object. Any manipulation is performed by modifying some of the attributes of the owned object (such as position). Every single object in Open Community has to have an owner. Objects which don't comply with this rule are simply removed from the world. The ownership of an object may be transferred from one user (process) to another. If a given process wishes to get ownership of an object it may call the spRequestOwnershipTransfer, which will throw a callback to the owner process. If this one agrees with the transfer of ownership, it just calls spTransferOwnership and the object has a new owner. An object may also be transferred to another process, even though the latest one didn't require such ownership. This is useful when one wishes to leave a given object in a world even though he or she will leave that world.

2.2.2.7 Interactive Sharing Transfer Protocol (ISTP)

ISTP is the protocol behind SPLINE (Scalable Platform for Large Interactive Networked Environments), which is an implementation of Open Community. ISTP is not particularly attached to SPLINE and vice-versa, i.e. ISTP may certainly be used for any other Virtual Environment architecture (such as RTI, described in HLA below). Also, SPLINE could use other set of protocols to accomplish its task (Sharing of Virtual Environments). ISTP has certainly many features which resemble SPLINE, such as support for Beacons and *Locales* [WaAS97].

ISTP's goal is to communicate information about objects in a shared world model underlying a DVE. Such a goal must include features for handling the various objects present in a Virtual World. The various objects that SPLINE must support carry quite different characteristics, ranging from very large objects which rarely change, such as a background image, to real-time objects such as streamed audio. To accomplish such broad range of characteristics, ISTP contains five subprotocols, namely:

- 1-1 Connection Subprotocol: Used to establish and maintain a TCP connection between two ISTP processes;
- Object State Transmission Subprotocol: Used to communicate the state of objects from one ISTP process to another. Such updates may be sent via 1-1 Connection or UDP Multicast;
- Streaming Audio Subprotocol: Used to stream audio data via RTP;
- *Locale*-Based Communications Subprotocol: This is the core of ISTP and supports the sharing of information about objects in the world model; and
- Content-Based Communication Subprotocol: Supports central server style communication of beacon information.

The last two subprotocols are built upon the other three. ISTP does not provide video-streaming capability to date; however such support could be provided by extending ISTP with an appropriate subprotocol.

A key feature of ISTP is that it is fully distributed, i.e. a ISTP process plays the role of client and server for other processes.

ISTP is a hybrid protocol that builds on top of four underlying protocols: TCP, UDP, RTP, and HTTP.

- TCP is used for the reliable communication of control information.
- UDP and RTP are used for the communication of time critical information. (UDP and RTP messages are both sent via multicast whenever possible.)
- HTTP is used for the distribution of large or complex pieces of data.

By this hybrid approach, ISTP addresses the communications needed for different sizes and types of objects in a fairly efficient manner, and provides the reliability and error correction necessary for operations across various types of network configurations.

Collaborating ISTP Processes keep a local copy of the data they share. Such data, called World Model in ISTP terminology, are limited by the notion of *locales* as described above, i.e. each process only handle objects within the *locale* it is in, as well as any other *locale* in the neighbourhood. This feature allows scalability, while reducing required processing power in each participant station, as discussed above.

2.2.2.7.1 1-1 Connection Subprotocol

1-1 Connection subprotocol is implemented through a TCP connection. The process which is requesting the channel opens a TCP connection to the other ISTP process and sends an HTTP Get message asking for connection. If the connection is accepted, an ISTP Connection Status reply will be sent and the connection will be opened. In case the connection is denied, the reply consists of a Not Found HTTP reply. At any time either process may close this connection and keep-alive messages are sent in a regular basis to inform each other that the processes are still alive. At any time one of the processes may re-open the connection. In such case, the other party behaves just like if a new connection was just started. This feature is useful to reinitialize data in a corrupt World Model.

2.2.2.7.2 Object State Transmission

The variables in a given World Model which need to be shared are communicated by ISTP through Object State messages. Such messages are sometimes dispatched within the 1-1 Connection using TCP and sometimes sent using multicast UDP. When an Object State message is received by another process, it is used to create/update/remove the copy of the object it refers to. There are basically three different kinds of descriptions of an object: full description, differential description and link data differential descriptions. The first contains the values of all shared variables; the second contains only the variables which have changed while the last specifies how the data associated with a link has changed. Similarly to ISO MPEG I and P images [ISO11172-2], the full description images are much larger than differential description ones, but the latter relies on the existence of all other values at the destination, reason why full descriptions need to be send sometimes (also periodically, just like I frames).

Each object in ISTP is identified by a Globally Unique ID (GUID), which includes the IP address to avoid server dependency. For this reason the GUID is quite large, containing a 32 bit Internet address as well as a 64 bit intra-machine unique value. It is easy to conclude that ISTP need to suffer major changes to accommodate IPv6, reason why SPLINE does not run over IPv6 as per now.

In order to allow precise specification of timing information, ISTP implements methods for adjusting the time values among machines, which allows the use of a compact absolute time stamp. The current version of SPLINE does not implement such service. Thus, it is necessary to use some external NTP application to synchronize clocks.

2.2.2.7.3 Streaming Audio Subprotocol

This subprotocol is based directly on the Real Time Protocol (RTP) [RTP].

2.2.2.7.4 *Locale*-Based Communication

The *Locale*-Based Communication subprotocol communicates information about changes in the world model through UDP multicast backed by TCP unicast when UDP messages are lost. A process indicates which *locales* it is interested in obtaining information about by creating a spObserving object as described previously. Each *Locale* has a *Locale*-Based Communication server associated with it, with a given server possibly supporting multiple *Locales*. Each *Locale* receives two Multicast addresses, one to communicate changes in the objects in the *locale* to its "subscribers" and another used for streaming audio communication. When multicast isn't available, ISTP uses simulated multicast via a 1-1 Connections. A hybrid approach may be used, i.e. if 9 out of 10 processes are within a multicast enabled network, these will be served by a unique multicast address while the one process with no multicast capability will receive updates through a 1-1 Connection to the *Locale* server. This allows scalability, while serving all clients. Such process is transparent to the process in the sense that, other than an augmented delay, it can't be noticed.

2.2.2.7.5 Content -Based Communication

Content-Based Communication supports content addressable connections between ISTP processes. The subprotocol aims at controlling the storage and retrieval of beacons with URL-like tags. After creating an object, a given process may publish it by creating a beacon object and registering with some Content-Based Server. At this point, this beacon is visible by any other process, which may retrieve such information at any time. Whenever a process creates or modifies a beacon it opens a 1-1 Connection to the server specified by the tag of the beacon and performs any update that is required [BaWA96, BaWA96b].

2.2.3. Distributed Interactive Simulation (DIS)

Even though not considered a middleware [Mace95], since there is no component mediating the communication, DIS still has some importance for our discussion, reason why it is included in this section. DIS (Distributed Interactive Simulation) is a standard [IEEE1278] which focuses on military simulations and has been created as an improvement of SIMNET. SIMNET (Simulator Network) has been one of the very first standards developed for military simulations. It was developed by ARPA and the US Army by Bold Beranek and Newman, Perceptronics, and Delta Graphics [Mace95]. SIMNET has been developed to take full advantage of Ethernet hardware, when the broadcasting was heavily used, reducing software selection of packets; however that also brought undesirable dependency of Ethernet facilities which are only fit for a LAN, which makes easy to determine its limitations for large simulations. In SIMNET there is no central object repository, i.e. each host is responsible for maintaining its own copy of the objects participating of the simulation. Stations participating in a simulation only exchange state messages with the others and dead reckoning is used to reduce communication requirements, which will be discussed shortly.

DIS (IEEE 1278 standard) has been developed in an attempt to overcome SIMNET limitations. It is a group of standards developed by the US Department of Defense and industry. DIS uses similar Protocol Data Units (PDU) as SIMNET, as well as its terminology and some of its functionality, such as Dead Reckoning. Dead Reckoning is implemented by the idea of player and ghost [Mace95], when each object is controlled by a unique station (its owner) and by a player object. Such an object is present on all other stations as a ghost object. The ghost object is supposed to mirror the actions of the player in each station; however, no state updates are sent all the time, instead the ghost tries to predict the motion of the player. The player also calculates such prediction using the same algorithm. Only when there is an error greater than a pre-defined threshold, the player sends an update which is used to update the ghost in every station. Such messages are used to correct its position/state. This approach diminishes the scalability of DIS since all objects are mirrored in every station. For example, in a 1,000 participant environment, each station would have to process 1,000 ghost objects (the player's ghost has to be processed locally as well) and a player (minimum).

As DIS does not provide a central server or similar component, there are also problems related with latecomers, e.g. how is a latecomer supposed to get an updated status of the simulation in an efficient way? In order to solve this problem, DIS requires all objects to send periodic update messages even for the objects which do not move, such as a bridge, or are "dead". The newcomer then has just to listen for a while until the next refresh cycle happens. There are some problems as to the selection of an appropriated refresh cycle, as the shorter it is the higher the load on the network. On the other hand, if it is too long, a newcomer will have to wait too much before getting information about the simulation. Thus, each player sends periodically updates, even if the prediction is smoothly matching its actions (the same thing happens for stationary objects). Moreover, DIS requires that the multiplexing of the various media be performed at the application layer, which prevents the model from benefiting from appropriate protocols, such as RTP for real-time audio.

DIS has been broadly used by the American Department of Defense but has been dropped since its Master Plan was released in 1995. DIS still has historical importance, along with SIMNET, as they have been some of the very first standards to support collaboration amongst various users in a Synthetic World. DIS is also known as IEEE 1278, as it is an IEEE standard.

2.2.4. High Level Architecture (HLA)

2.2.4.1 Historic

In accordance with the US Department of Defense (DoD) Modeling and Simulation (M&S) Master Plan, the Defense Modeling and Simulation Office (DMSO) is leading a DoD-wide effort to establish a common technical framework to facilitate the interoperability of all types of models and simulations among themselves, as well as to facilitate the reuse of M&S components. This Common Technical Framework includes the High Level Architecture (HLA). Initial definition of the M&S High Level Architecture was accomplished under the sponsorship of the Defense Advanced Research Projects Agency (DARPA) Advanced Distributed Simulation (ADS) program. Central to this task was the development of a set of prototypes which addressed critical issues in the HLA. The HLA Baseline Definition was completed on August 21, 1996. It was approved by the Under Secretary of Defense for Acquisition and Technology (USD (A&T)) as the standard technical architecture for all DoD simulations on September 10, 1996. In December 1997, HLA was accepted as a draft IEEE standard to be supported by the Simulation Interoperability Standards Organization (SISO). In February 1998, an Evolved, Stable Specification of HLA was released by DMSO [Weat98, CaWe96, HLA]. In late 2000, it was approved as IEEE standard 1516 [IEEE1516].

The DoD has defined a set of compliance milestones, known as the "No can" dates:

- "No Can Pay": From the beginning of 1999 on, no funds toward development or modification of non-HLA simulations will be admitted;
- "No Can Play": From the beginning of 2001 on, all non-HLA compliant simulations will be completely dropped by the DoD.

Only upon approval of USD (A&T) a non-HLA compliant project may be waived of the HLA policy.

2.2.4.2 HLA Terminology

- Federation: a set of simulations, a common federation object model, and supporting interoperation facility, which are used together to form a larger model of simulation.
- Federate: a member of a federation; one point of attachment to the infrastructure.
- Federation Execution: a session of a federation executing together.

2.2.4.3 Architecture

The High Level Architecture is composed of three components [Weat98]:

- Object Model Template: Defines a basis for the exchange of data and events between simulations. It defines the set of objects defined to represent the real world in the simulation, the attributes and interactions of such objects as well as the level of details of each object (including spatial and temporal resolution).
- Runtime Infrastructure (RTI): Consists of a collection of software that provides the required services to simulation systems. Such services are available through a programming API which should be available for C++, Java, Ada'95 and CORBA IDL;
- HLA compliance rules: Consists of a set of rules that must be accomplished by a simulation to be considered HLA compliant. Example of such rules includes: Federations must have an HLA Federation Object model (FOM); All Federates must have an HLA Simulation Object Model (SOM); All object representation occurs in the Federates, not in the Runtime Infrastructure.

2.2.4.4 Run Time Infrastructure

RTI provides a set of common services that are useful across multiple simulation domains. These common services fall into six categories, namely:

- Federation Management: Includes 20 services which aims at creating and deleting federation executions; joining and resigning federation executions, as well as control checkpoint/pause/resume/restart operations.
- Declaration Management: Establishes intent to publish and subscribe to object attributes and interactions.
- Object Management: Creates and deletes object instances; controls attributes and interaction publications as well as creates and deletes object reflections.
- Ownership Management: Transfers ownership of object attributes.
- Time Management: Coordinates the advance of logical time and its relationship to real time.
- Data Distribution Management: Supports efficient routing of data.

Chapter III

3. Very Large Virtual Environments

A CVE is a system through which users may interact with each other and collaborate through a virtual synthetic world. As computing resources are limited, there are obvious problems which rise once the number of users in a simulation raises beyond a certain limit. In fact, if no special mechanisms are provided, one may expect a simulation to produce undesirable effects such as choppy rendering, loss of interactivity and alike. Systems such as DIS, which are known not to scale well, suffer from such problems. An example is given in section 2.2.3 where, due to the heavy dead reckoning employed, each station has a "ghost" of every other player, which requires each station to process all ghosts information in addition to the usual processing of the user's actions. For a sufficiently large number of users, even a powerful station will be overloaded and either the whole system will get slower or such station will present an incoherent world to the user. In the former example, it is assumed that a few objects will be presented in the wrong position due to the inability to dead-reckon their new location. The increased amount of message exchanges required might easily overload a network, requiring retransmissions, which makes things even worse. One may also consider delays and jitter, which has strong impact on collaboration coordination [Leig99]. To address these issues research has been conducted to optimize network and computing resources. In this chapter, we will present some work developed aiming at enabling a larger number of users to be supported in a CVE system. In the following chapter, we will introduce our own hybrid adaptive architecture.

A number of standards and prototypes have addressed the issue of allowing a larger number of users to collaborate through a CVE. In this section, we will discuss briefly several of them, namely NPSNET-IV, DIVE, SPLINE, MASSIVE-2 and SCORE, which somewhat represent the other models as well.

3.1 NPSNET-IV

NPSNET-IV [MZPB94] is a prototype developed at the Department of Computer Science at the Naval Postgraduate School in Monterrey, CA, USA. NPSNET I and II were based on Ethernet technology, which limited their use to LANs, hence with few stations. An enhanced version called NPSNETStealth was developed complying with SIMNET and later on NPSNET IV was designed to comply with DIS 2.0.3. NPSNET IV included enhancements such as the use of IP Multicast with dynamic multicast groups reflecting a hexagonal partition of the Virtual World. This guarantees a constant number (three) of hexagons to be added and deleted when an object moves from one hexagon to an adjacent one [Mace95, MZPB94]. Figure 10 shows such add/remove characteristic. NPSNET IV also implements the player/ghost paradigm detailed in section 2.2.3 and, as DIS, NPSNET's communication PDU include military oriented packets, such as Fire PDU, Detonate PDU and alike, which makes it somewhat unfit for civilian applications.

NPSNET-IV takes advantage of SGI Onyx with Reality Engine2 graphics' heterogeneous parallelism and multiprocessor architecture; however this imposes a restriction regarding the compatibility of hardware, i.e. it requires SGI IRIX based stations.

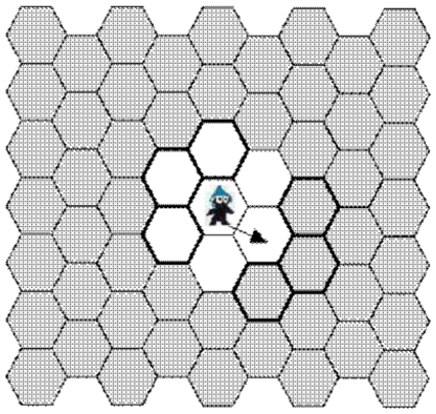

Figure 10: Moving Through Hexagons in NPSNET-IV

NPSNET-IV achieves better results, compared with DIS, mostly due to the lack of keep-alive heartbeat messages, as well as the partitioning of the world which ensures that at a given time only a few hexagons (7 at a time) need to be dealt with. Each hexagon has its own multicast group and data is sent through the multicast address of the hexagon where the source of such data flow is located. Figure 10 shows the hexagonal partitioning of the World with an Avatar moving from one area to the neighbourhood. The avatar will unsubscribe the multicast groups for the three hexagons to its left and subscribe the new set of three multicast groups defined by the three hexagons to the right. Since each hexagon has a fix size, the amount of objects in each hexagon may be large. The hexagonal partition of the world may be thought as a limited kind of *Locale* as defined in SPLINE, Section 2.2.2. NPSNET-IV works well for the Military application it has been designed for.

More recently NPSNET-V has been introduced in [CMBZ00, NPSNET-V]. NPSNET-V is being designed to become a platform for research on infrastructure for Large Scale CVE. It defines a dynamic partitioning of the World similar to SCORE, with regions being split like SCORE cells. NPSNET-V implements Ownership Management.

3.2 DIVE

DIVE, Distributed Interactive Virtual Environments, is a prototype developed at the Swedish Institute of Computer Science (SICS), Sweden.

DIVE [FrSt98, Hags96] implements a shared, distributed database, which is partially replicated at each client. Modifications are first realized in the local copy of (part of) the world and sent to the other clients through the network. Each client updates its local copy of the world once an incoming update arrives. By using this approach, DIVE allows copies of the world to be slightly inconsistent to each other. It also implements dead reckoning in an attempt to further reduce network load.

DIVE divides the world into sub-hierarchies, which are similar to SPLINE's *Locales*. A multicast group, called lightweight group, is assigned to each sub-hierarchy. Only the clients interested in such sub-hierarchy would then join its lightweight group.

Database changes are distributed using a negative-acknowledgement based reliable multicast communication scheme, which ensures the eventual consistency of the world. Continuous media streams, on the other hand, are sent using unreliable multicast.

Lightweight groups define a hierarchical tree which is used by the communication engine to select the appropriate multicast group through which a given message should be sent. It is important to note that the world itself contains a multicast group attached to it. All entities are required to listen to such multicast group. When a message is to be sent, the communication engine climbs the hierarchy tree looking for the first available multicast group.

3.3 MASSIVE-2

MASSIVE-2 (Model, Architecture and System for Spatial Interaction in Virtual Environments) is a prototype developed at the Computer Science Department at the University of Nottingham, U.K [Gree97].

For the sake of clarity on the description of MASSIVE-2, its terminology follows [Gree97, GrBe97]:

- World: Place where all interactions occur. A CVE may contain a set of disjoint worlds;
- Artifact: Tangible objects which populate the VE. This include any object present in the world;
- Aura: Area to which a given artifact if offered or requested. Used for spatial trading;
- Group: Coordinate the use of multicast communication. Used to group artifacts;
- Focus: The area within the world (around a participant) which is visible by the user. Describes the observer's centre of attention, or which other artifacts interest~~s~~ a given artifact;

- Nimbus: Describes the observed object's availability, or which other artifacts can see (or hear, etc.) the artifact;
- Regions: Subdivision of the World; it is somewhat similar to a *locale* in Open Community;
- Participants: Representation of the user, usually represented by an artifact.

The major contribution of MASSIVE-2 is the introduction of the Third-Party Objects, which allows a hierarchical dynamic space-based embodiment of multicast groups [Gree96]. The idea behind third-party objects is to allow a group of artifacts (called crowd) to be represented as a unique object which is seen by others. Only when an artifact gets into a crowd boundary, it will receive information regarding individuals within the crowd. This model allows an elaborate hierarchy of groups, as a crowd may be composed of artifacts and other crowds, recursively. Such an approach requires that media mixing be performed in order to provide a single audio channel, for instance, representative of the whole group. This processing may be prohibitive by itself, if we consider that each nested crowd would require a station to process its media (not only audio) to give a representation of the group.

Focus and Nimbus concern awareness. It is said that "the more you are in my focus, the more I am aware of you and the more you are within my nimbus, the more you are aware of me" [GrBe97].

MASSIVE-2 runs in SGI workstations running over IP.

MASSIVE-3 has been introduced more recently [GrPS0]. MASSIVE-3 is based on *Locales* like in SPLINE, but extended to include subdivisions of *Locales*, called aspects. Massive-3 also implements a number of Ownership Control policies.

3.4 SCORE

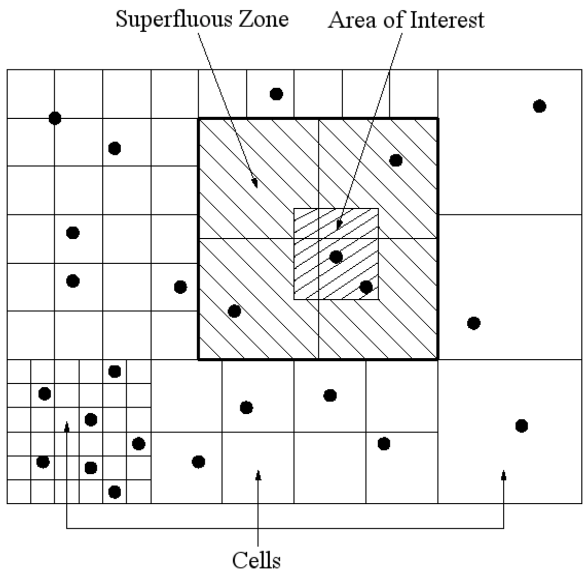

Figure 11: Cells in SCORE

SCORE [LeTB99, Lety00] has recently been introduced. It was developed at INRIA – Sophia Antipolis, France in 2000. It is based on the division of the World in Cells as suggested in [HoRC94]. A user interacts with those cells, which fall, at least partially, within an area of interest. The latter is defined as a square region around a user's avatar. Each cell has its own multicast group (MG) and an avatar then subscribes for that set of MGs. SCORE allows for two policies regarding determination of cell-size: pre-calculation of a fixed cell size and dynamic re-estimation of the cell-size during the session. The dynamic estimation may be performed based on some pre-defined parameters, such as number of MGs available, density of participants, etc. Furthermore SCORE allows the partitioning of the World to have cells of different size, which allows one to have a fine grid at highly populated areas and a sparse grid of cells in unpopulated areas, as shown in Figure 11.

Chapter IV

4 The Proposed VELVET Architecture

VELVET (a Hybrid Adaptive Architecture for VEry Large Virtual EnvironmenTs) is our Adaptive Hybrid Architecture.

The approaches previously described, while addressing the issues of a Large-Scale Virtual Environment (LSVE), have been known to fail under certain conditions. In section 4.1, we will describe such failure potentials. In section 4.2, we will present the VELVET approach in a high-level overview, whereas section 4.3 will bring details about how the functionality described in the previous section actually works. We will show that VELVET clearly performs better under the circumstances discussed in section 4.1.

4.1 Limitation of Existing Models

We may assume that SPLINE is representative of a class of models based on geographic partitioning of the Virtual World, such as NPSNET-IV and others. In both cases the Virtual World is partitioned and each user is supposed to receive data from objects which are located in a well-defined subset of the partition (or *Locales* for SPLINE).

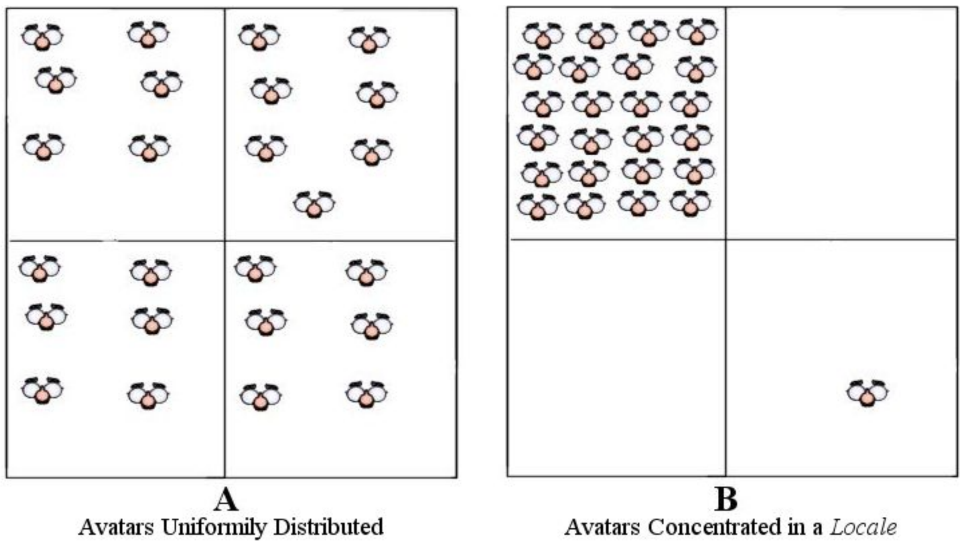
A
Avatars Uniformily Distributed
B
Avatars Concentrated in a *Locale*

Figure 12: Space Based Solutions and their Limitations

Locale-based models assume that users are somewhat uniformly dispersed in the Virtual World (Figure 12.A). That is, the idea of reducing the amount of data which each station must deal with is addressed by reducing the area which is “seen” by each participant. That would assume that by reducing the area dealt with, the number of visible users would equally be reduced. If, however, most (or all) users are packed together in a small area of the LSVE, the number of objects a given user must deal with may still be too large (Figure 12.B). Suppose we have a Virtual Museum where some dozens of thousands of users are visiting. If all of them decide to see “The Mona Lisa”, one may notice that all stations would have to deal with all the dozen of thousands of data flows, as all of them would be in the same *Locale* (or in the neighbourhood). The *Locale*-based approach would hence fail in the task of reducing the amount of data each host must deal with in such a case. Other examples would be: a Virtual City, if the mayor calls for a meeting in a central park, or if many users wish to watch a soccer game in a Virtual Stadium.

Another limitation not quite addressed by the existing architectures is that of heterogeneity. Let us consider a group of 300 hosts participating in a CVE. Yet, let us assume that 290 of such systems are quite powerful, meaning that they can easily deal with the load, even if messages from every station successfully arrive. Such systems could also have very good networking connections which could support such a load gracefully. The remaining 10 stations, however, are assumed to be weak enough not to be able to deal with the load. Space-based solutions tend to assign an equivalent load to all systems populating the same neighbourhood. That would work well only if all systems are able to deal with the same load, which is not true in our example. The workaround in this case is that either all systems would have to meet a minimum or all systems would have to reduce data transmitted so that the weakest of the systems could support the load. Both solutions are somewhat inadequate as the first prevents some users from joining the CVE session while the second would under-use resources.

4.2 VELVET's Architecture

VELVET aims at allowing each and every user to interact with the Virtual World to the maximum extent possible (or optionally as much as paid for). We will first introduce VELVET's terminology:

- World: The whole set of objects.
- Area or *Locale*: A subdivision of the World.
- Object: An object which is located within the World.
- Avatar: Special kind of object which represents a user.
- Bot: Active object which is not an Avatar.

- Artifact: An object which is neither Avatar nor Bot.
- Area of Interest (AoI): Radius a user is able to view and directly interact with.
- Check-In: Operation which brings an object into a user's AoI.
- Check-Out: Operation which removes an object from a user's AoI.

VELVET is a CVE architecture which allows real time adaptation, according to the local client needs. At any point in time, a given user may elect to unilaterally reduce or increase his/her own view of the World. VELVET gracefully supports heterogeneous collaboration. Such feature allows systems with different processing power and networking facilities to still collaborate efficiently, since each one may elect to see just as much as possible (or as much as paid for).

In section 4.2.1, we will discuss about Area of Interest in general terms. That is followed by the Double Layered Boundaries, the Parallel Virtual World (PVW) and finally the support to heterogeneity of VELVET. Using the knowledge about the PVW, we will revisit Area of Interest Management on VELVET, as that can be better defined in terms of the PVW.

4.2.1 Area of Interest Management

The idea behind VELVET is that each avatar will be able to "see" whatever is located within its AoI. The AoI of a given avatar does not depend on another avatar's AoI. Such behaviour allows for each station to manage how large its own AoI is, hence how much of the World can be seen at a time. The AoI can be enlarged and reduced dynamically so that upon increase in load, by a higher density of objects around an avatar for instance, one can automatically reduce the AoI so that the load can be kept within a treatable range. Figure 13 shows a group of avatars in a room with one avatar, indicated by the arrow (A). One can then see a larger number of avatars (B) which makes the AoI shrink (C) as needed. Only the objects which are within the AoI are visible and only information from those objects is received.

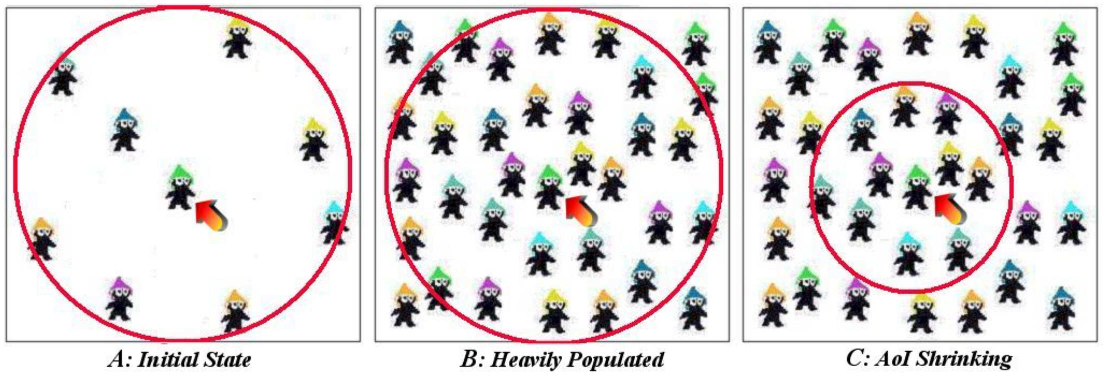

Figure 13: VELVET's Area of Interest Shrinking

Whenever the number of objects decreases, that same avatar may have its AoI expanded so that more objects may again be visible. That is shown in Figure 14.

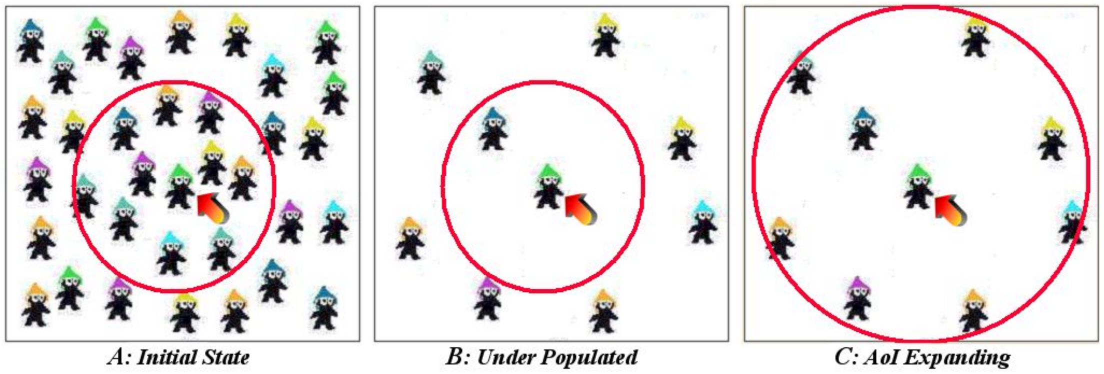

Figure 14: VELVET's Area of Interest Expanding

If we let B be the average traffic intensity transmitted by a participant, P_A be the number of participants a user is aware of and P_I be the number of participants a user is actually interested on, we can express incoming traffic for Space based solutions as $B \times P_A$ and for

VELVET as $B \times P_I$, where $0 \le P_I \le P_A$. A more formal description of AoI Management is shown in section 4.2.5.

4.2.2 Double Layered Boundaries of VELVET's AoI

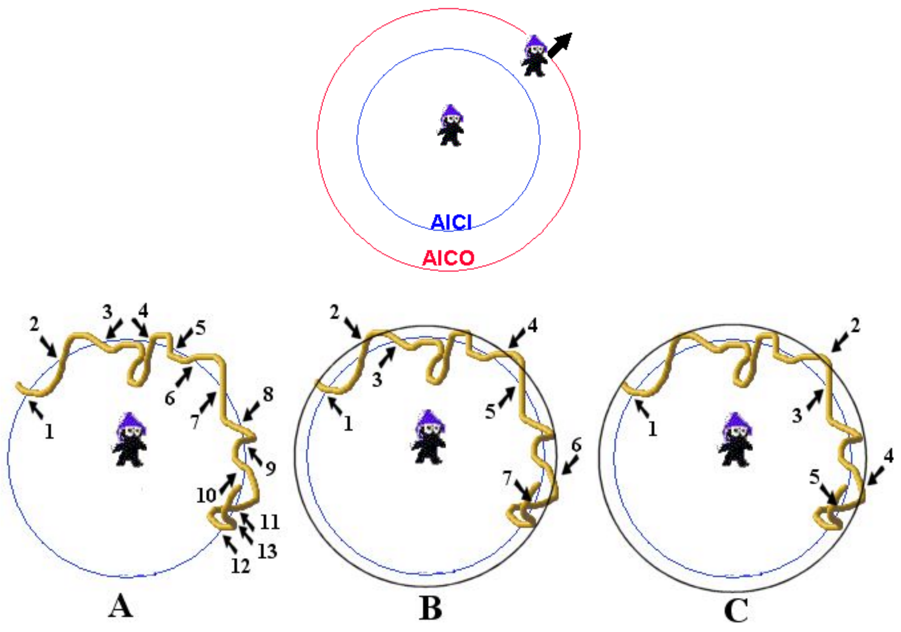

Figure 15: VELVET's AICI/AICO

A – Single Layer (13); B – Double Layer (7); C – Wider Double Layer (5)

When an object crosses the border of the AoI, it will Check-In (CI) or Check-Out (CO), depending on the direction being respectively towards the AoI or leaving the AoI. In order to avoid multiple Check-In/Check-Out operations, VELVET in fact defines two borders named Area or Interest Check-In (AICI) and Area of Interest Check-Out (AICO), so that only objects crossing AICI will Check-In and those crossing AICO will Check-Out. The "distance" between AICI and AICO is also variable and may be used to control the number of CI/CO operations. Figure 15 (top) shows an example of use of AICI/AICO. Figure 15 A, B and C shows respectively an AoI with a single border (or AICI and AICO with distance zero), and two different distances between AICI and AICO. One can notice the arrows displaying 13, 7 and 5 CI/CO operations for the same path.

If we define the following parameters:

- p_i = the radius of inner boundary of AoI_i;
- q_i = the radius of outer boundary of $\text{AoI}_{i;}$
- $q_{i,\ max}$ = the max of $q_{i;}$
- $q_{i,\ init}$ = the initial size of q_i;
- $s = q_i\text{-}p_{i;}$
- n = the allowable max number for checking of boundary collision;
- t = a limited time slice for checking of boundary collision;
- c = number of boundary collision during t.

We can define the procedure for AICI/AICO management as follows:

```
while (1)
{
        if ((c > n) && (qi ≤ $q_{i,max}$) && timeout)       increase $q_i$;
        if ((c ≤ n) && (qi > $q_{i,\ init}$) && timeout)       decrease $q_i$;
}
```

Where *timeout* is the minimum time which shall elapse before changes in the AICI/AICO, which prevents too many changes to happen in a small time interval. Such time interval can be set, for instance, at 1 second. Note that the procedure above is to be performed locally, with absolutely no dependency on any other participant.

As per the AoI's rule for expansion/shrinking, we shall add that it is not necessarily based on virtual space (Euclidean distance) from the avatar but rather on a pre-defined metric in use by that user's VELVET management subsystem. That is, one can see what is more important to him/her rather than what is geometrically closer, as the Figures 13 and 14 indicate. We can assume that the positioning of the avatars in Figure 13 and 14 is based on such metric, rather than the virtual distance to the avatar. Of course the metric itself could be that of virtual distance.

4.2.3 Parallel Virtual World of VELVET

The metric defines a Parallel Virtual World (PVW) in which objects are placed according to the metric chosen by each participant. Figure 16 shows the PVW.

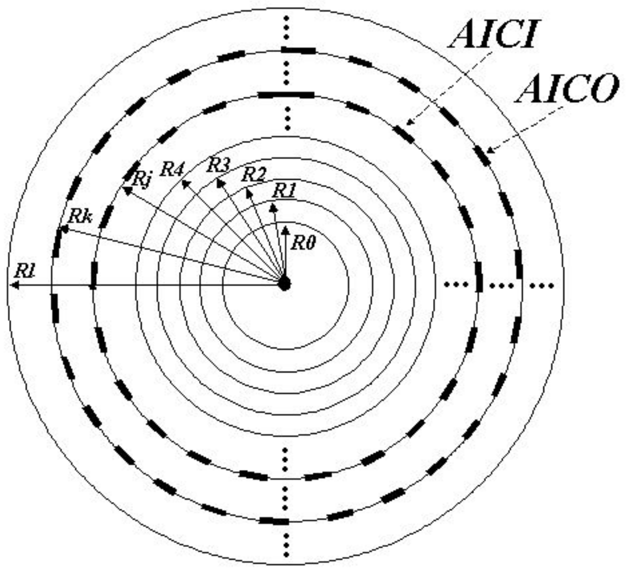

Figure 16: VELVET's Parallel Virtual World

The rings define levels in the metric oriented PVW. Each avatar has its own PVW and the management subsystem decides how many of the rings shown in Figure 16 will be subscribed for. Note that as each Avatar has its own PVW, the rings can be particularly arranged based on each participant's interest, for instance each ring may have a single object or a collection of those.

Let MS be a set of metrics, $MS = \{M_0, M_1, \ldots, M_m\}$, where,

M_0 = Metric 1, e.g. Number of Users;

M_1 = Metric 2, e.g. Network Throughput;

M_2 = Metric 3, e.g. Distance in the Virtual World;

M_3 = Metric 4, e.g. Distance in the Network (Number of Hops);

M_4 = Metric 5, e.g. Distance in Number of *Locale* Hops;

M_5 = Metric 6, e.g. Average End-to-End Delay;

M_6 = Metric 7, e.g. A mix of the above, such as $M_4 \times 10000 + M_1$;

$\vdots$

M_m = Metric m+1, e.g. Others.

Assume that in VELVET, an avatar A_i has its own parallel virtual world (PVW$_i$) with a metric M_γ at a given time t.

$PVW_i\ (M_\gamma) = \{R_0, R_1, \ldots, R_{l-1}\}$

Where $M_\gamma \in MS,$

R_k is (k+1)th level of the metric M_γ, $0 \leq \gamma \leq m$, and

l is the number of levels in the current PVW_i,

where R_{l-1} is the maximum level of M_γ.

The AoI will be such that it will include R_k, $0 < k < l$, so that $\sum_{n=0}^{j} \xi(R_n) \le T \le \sum_{n=0}^{k} \xi(R_n)$; where $\xi(R_n)$ is a function which gives the cost associated with the level R_n, e.g. the number of participants in level n for a metric considering the number of users. T is a value which will be optimized according to a pre-defined target value for a given metric. For instance, if one is behind a 56K modem connection and the metric considered is Total Throughput, the Target could be something like 48Kbps. T would be maximized considering that it must remain below such target. See section 4.3 for further details on the implementation of PVW in VELVET.

4.2.4 Degree of Blindness & Support to Heterogeneous Systems of VELVET

Since participants unilaterally decide which objects to subscribe for, based on the PVW, one may notice that such behaviour can lead to inconsistencies. Such inconsistency shows up when a given avatar A "sees" an avatar B even though B can't "see" A. That would happen if A's AoI is expanded enough so that B is enclosed while B's AoI is shrunk enough not to enclose A. This is called in VELVET's terminology "degree of blindness", which limits one's vision. That is perfectly legal in VELVET. In fact, it is the very reason why VELVET graciously supports collaboration among users in heterogeneous systems, when system collaborate the best they can. Figure 17 illustrates such behaviour, which defines the Degree of Blindness in VELVET. The smaller the AoI, the greater is the Degree of Blindness. Figure 17 shows users A and B adopting similar metrics; the area around each avatar is that based on the PVW rather than the Euclidean space.

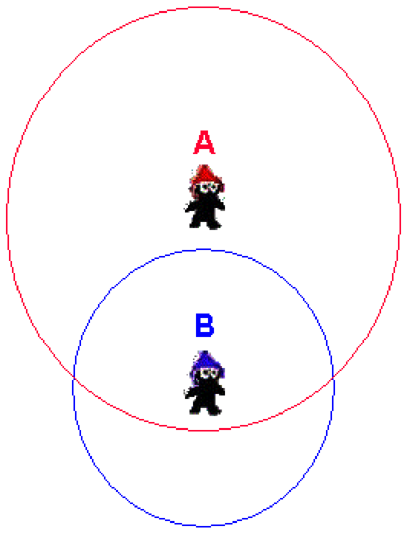

Figure 17: VELVET and Degree of Blindness

Define $\partial(A_i, A_j) \equiv \partial_{ij}$ as the distance between participant A_i and A_j in the PVW of participant A_i and ρ_i the radius of the AoI in participant A_i's PVW, as shown in Figure 18. We can define that $A_i \subset A_j \Leftrightarrow \partial_{ji} \leq \rho_j$, i.e. if A_i is within A_j's Area of Interest. Similarly we can define that $A_i \supset A_j \Leftrightarrow \partial_{ij} \leq \rho_i$. In other words $A_i \supset A_j \Leftrightarrow A_j$ is within the A_i's AoI, i.e. if A_i can "see" A_j. Considering space based solutions, for two avatars A_i and A_j within the same Locale L_n, $A_i \supset A_j \Leftrightarrow A_j \supset A_i$ holds, because $\rho_i = \rho_j; \forall i, j$ and $\partial_{ij} = \partial_{ji}; \forall i, j$. In VELVET $\partial_{ij} = \partial_{ji}$ may not necessarily hold, as after all A_i and A_j may be using completely different metrics. Furthermore, in VELVET $A_i \supset A_j$ does not lead to $A_j \supset A_i$ because the metrics can be different, and, in the event that both participants use the same metric, $\rho_i \neq \rho_j$ may hold as well, which is the reason for the existence of different Degree of Blindness for each user. Degree of Blindness allows heterogeneous systems to collaborate the best they can, for instance, if one is participating in a VELVET session with a processor-weak system or behind a dial-up 56k modem, it would still be possible to interact with a limited number of participants (high degree of blindness). At the same time another user joining a VELVET session with a supercomputer with a very fast networking connection would be able to interact with a comprehensive view of the World (if desired).

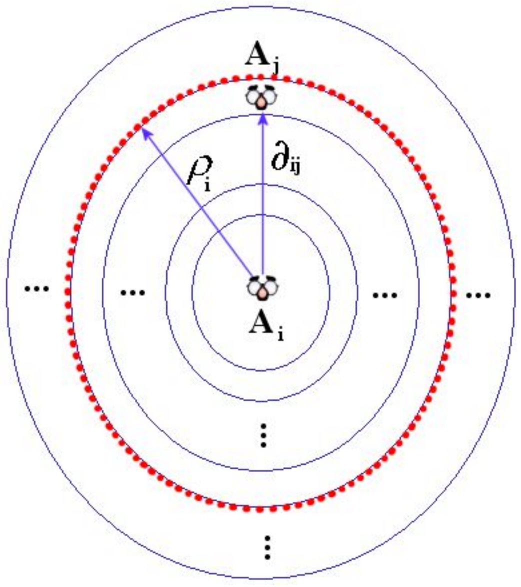

Figure 18: Parameters ∂ and ρ

Table 1 shows some $\subset$ and the equivalent $\supset$ relations of VELVET in the event that both A_j and A_i use the same metric M_δ.

Table 1. VELVET's $\subset$ and $\supset$ relations for objects with similar metrics.

VELVET	$\partial_{ji} \le \rho_j$	$\partial_{ji} > \rho_j$
$\partial_{ij} \le \rho_i$	$A_j \subset A_i$, $A_i \subset A_j$	$A_j \subset A_i$, $A_i \not\subset A_j$
$\partial_{ij} > \rho_i$	$A_j \not\subset A_i$, $A_i \subset A_j$	$A_j \not\subset A_i$, $A_i \not\subset A_j$

From Table 1 we can deduct relations such as if $A_i \subset A_j$ and $A_j \not\subset A_i$, then $\rho_j > \rho_i$.

Table 2 below shows several relations supported by VELVET, with the double grid box showing the scope of Space Based Solutions. This table assumes that that both A_j and A_i use the same metric M_k.

Table 2. Scope of VELVET and Space Based solutions regarding $\subset$ and $\supset$ relations.

VELVET	$\rho_i > \rho_j$	$\rho_i = \rho_j$	$\rho_i < \rho_j$
$\partial_{ij} > \partial_{ji}$	N/A	$A_j \subset A_i \Rightarrow A_i \subset A_j$	$A_i \subset A_j \Rightarrow A_j \subset A_i$
$\partial_{ij} = \partial_{ji}$	$A_j \subset A_i \Rightarrow A_i \subset A_j$	$\boxed{A_j \subset A_i \Leftrightarrow A_i \subset A_j}$	$A_i \subset A_j \Rightarrow A_j \subset A_i$
$\partial_{ij} < \partial_{ji}$	$A_j \subset A_i \Rightarrow A_i \subset A_j$	$A_j \subset A_i \Rightarrow A_i \subset A_j$	N/A

4.2.5 Area of Interest Management Revisited

In this section, we will detail how the AoI management works, now in terms of the PVW. Section 4.2.1 brought a high-level overview of AoI, which could shrink and expand. We will now define how that happens based on the PVW of VELVET.

AoI management basically consists of managing the AICO and AICI, which automatically defines participants to be subscribed for or dropped by the AoI management thread in a VELVET compliant system.

The AoI management in VELVET greatly depends on the PVW, which was not clear in section 4.2.1. Figure 16 shows a generic PVW with levels R_0 to R_l, being R_k and R_j respectively the levels of AICO and AICI, with $0 \leq j \leq k \leq l$. Let us assume that the PVW in question is that of participant A_i (PVW_i), without loss of generality. At any moment, if $\partial_{i\beta} > \rho_K$, A_β will check-out (CO) from the PVW_i, in the next cycle of the AoI management thread (if it was checked in previously). That happens because A_β is by definition outside of the Area of Interest of A_i (see section 4.2.3). On the other hand, if $\partial_{i\beta} < \rho_J$ at a given time, A_β would check-in (CI) into the AoI of A_i, in the next AoI management cycle (assuming that it was not checked-in already).

Additionally, $\forall\alpha; \rho_j \le \partial_{i\alpha} \le \rho_k$, if $A_\alpha \subset A_i$ at a given time *t+1* $\Leftrightarrow$ $A_\alpha \subset A_i$ at time *t*. Similarly, $\forall\alpha; \rho_j \le \partial_{i\alpha} \le \rho_k$, if $A_\alpha \not\subset A_i$ at a given time *t+1* $\Leftrightarrow$ $A_\alpha \not\subset A_i$ at time *t*, i.e. if the object A_α is located between the AICI and the AICO it will keep the same CI/CO status it had before.

If the AoI management detects too many CI/CO operations, it will act as follows:

- If the CI count is high, the AICI will get closer to A_i (i.e. *j=j-1*) so that ∂_{jk} will increase and an object will have to move further towards $A_{i,}$ in the PVW_i, before it checks-into the AoI of A_i. ∂_{jk} is defined as the distance between the AICI at level *j* and the AICO at level *k*, or $\rho_j - \rho_k$.
- If the CO count is high, the AICO will move further away from A_i (i.e. *k=k+1*), so that ∂_{jk} will increase and an object will have to move further away from A_i before checking-out of the AoI of A_i.

Similarly, if the CI/CO count is low the inverse procedure may be used so that ∂_{JK} decreases.

If a user experiences overload based on the metric (or other parameters, such as incoming traffic, processing power and any other bottleneck), both levels *j* and *k* (namely AICI and AICO) will be reduced, e.g. *k'=k-1* and *j'=j-1* (Figure 19.A). That will immediately lead to CO operations $\forall A_\alpha$ so that $\partial_{i\alpha} > \rho_{k'}$ and $\partial_{i\alpha} < \rho_k$, where *k* is the former level of the AICO and *k'* is the new shrunk level of the AICO. That is how the AoI shrinks based on the PVW.

Similarly, if the AoI management decides to expand the AoI that is accomplished by increasing both AICI and AICO levels (Figure 19.B). That will lead to CI operations for those objects which were not within the AoI before the expansion and which get enclosed after such expansion. More formally there will be CI operations $\forall A_{\alpha}$ so that $\partial_{i\alpha} < \rho_{j'}$ and $\partial_{i\alpha} > \rho_{j}$, where *j* is the former level of the AICI and *k'* is the new shrunk level of the AICI.

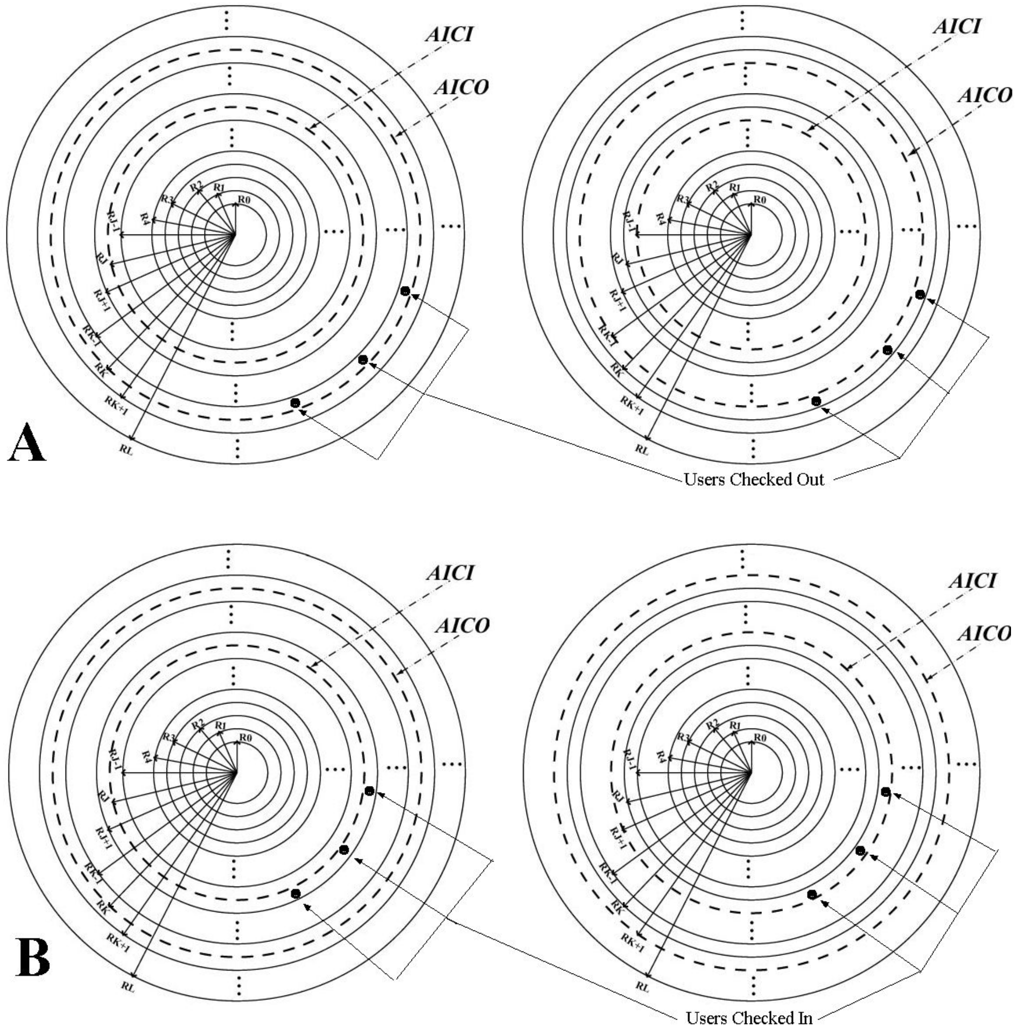

Figure 19: AoI Shrinking (A) and Expanding (B) based on the PVW

It is important to notice that the distances based on the function ∂ are distances in the PVW of a given avatar, it is not a distance in the Euclidean World. Remember that a PVW is built based on a chosen metric, as defined in section 4.2.3.

4.3 Internal Structures and Functionality of VELVET

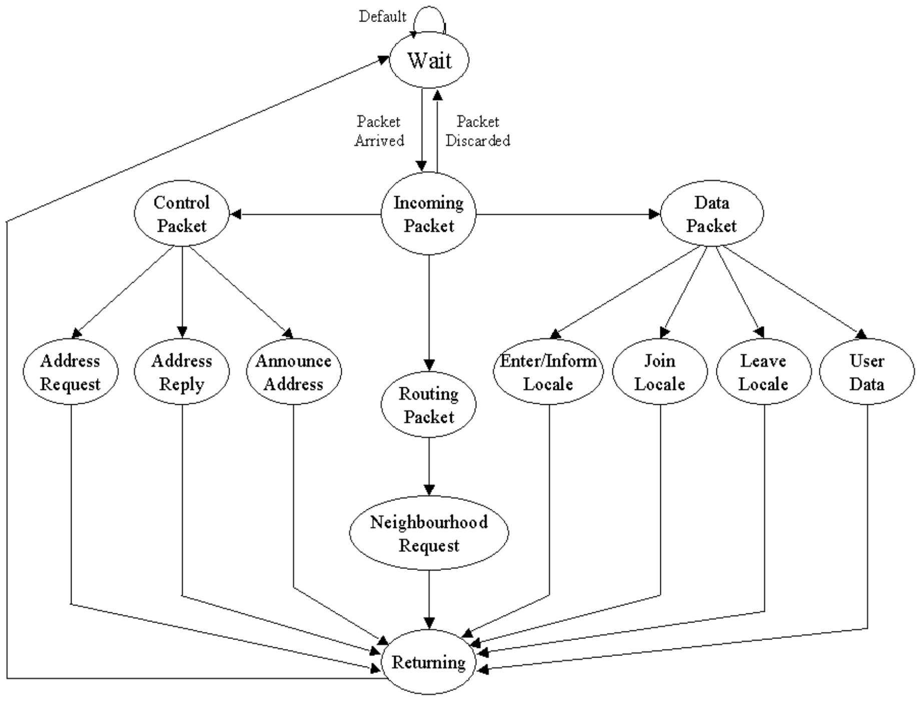

Figure 20: Receiver's Behaviour

In VELVET, the World is partitioned into Areas and each Area has a multicast address like in SPLINE and NPSNET-IV. VELVET accomplishes the flexible functionality described in the previous section by assigning a multicast group for each object which generates a flow of data (such as avatars and bots). That defines an Object Transmission Channel (OTC). Each client has three threads running in parallel. The first thread is the one responsible for sending data through the network, the second receives and acts upon arriving packets while a third thread performs management of AoI, joining and leaving Areas and OTCs as appropriated. Figure 20 shows the way the receiver thread works. Each *Locale* has similarly its own *Locale* Transmission Channel (LTC).

4.3.1 Awareness Control

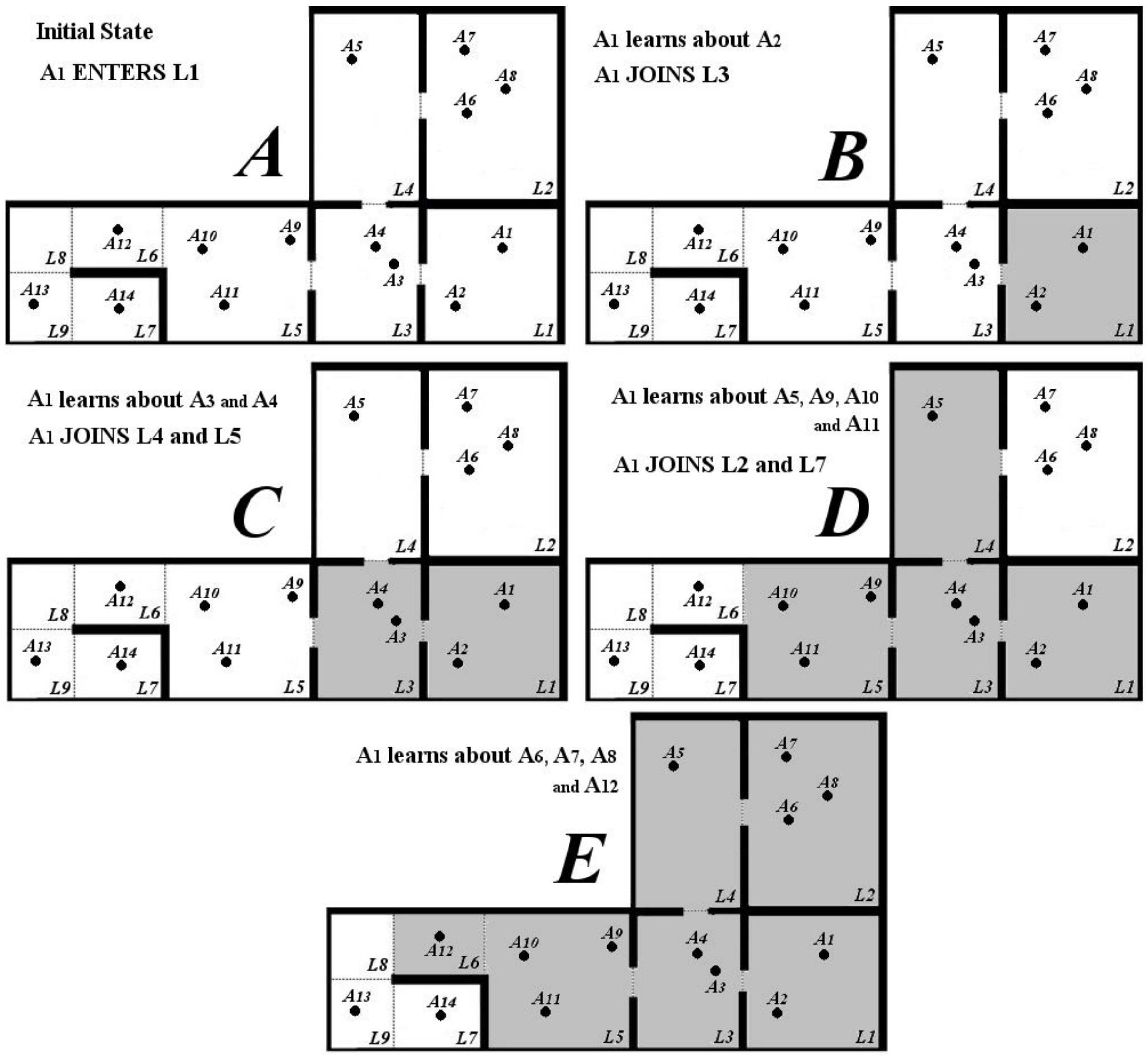

Figure 21: Awareness control in VELVET

In VELVET, a given avatar A_i may enter or join *Locales* $L_1, L_2, \ldots, L_n$. A_i enters a *Locale,* if it goes physically into that *Locale*. A_i joins a *Locale,* if it happens to have interest in such *Locale,* even though it is not physically in it. Once A_i enters or joins a *Locale*, it sends an appropriated message to the communication channel of that *Locale*. Every other active object which is located within the *Locale* sends an INFORMLOCALE message back to A_i, which learns in this way about the other objects. A_i can then consider these objects in the next cycle of its AoI management thread. On the other hand, the other avatars receiving an ENTERLOCALE message from A_i will become aware about it and will consider it in the next cycle of their local AoI management thread as well. An Avatar sends a LEAVELOCALE message once it is leaving a *Locale*. This message allows the others to remove it from the list of participants of such *Locale*.

The way an avatar A_1 becomes aware of other active objects is through the procedure depicted in Figure 21. The shaded area represents the areas which the local user has awareness about. Initially A_1 enters a *Locale*, such as *L1*, Figure 21.A. At this stage A_1 sends an ENTERLOCALE message in *L1*'s communication channel. This message is received by the active objects located in *L1*, which send a INFORMLOCALE message back to A_1, which learns in turn about them. Similarly A_2 has learnt that A_1 has just entered *L1*. Let us supposed that A_1 is following metric M_1 to subscribe to no more than 10 avatar's communication channels. At stage B in Figure 21 only 1 avatar is known to A_1 (namely A_2). As A_1's metric target is larger than 1 it joins the next level (the first in this case) of neighbouring *Locales*, namely *L3* in Figure 21.B and 21.C. A_1 then sends JOINLOCALE messages to *L3* and receives INFORMLOCALE replies from A_3 and A_4. Only 3 objects – A_2, A_3 and A_4 – are known, which is still under A_1's metric M_1. A_1 goes on and subscribes to the next level of neighbouring *Locales*, namely *L4* and *L5* (Figure 21.C and 21.D). Such procedure goes on one step further until the metric M_1 has been met or a sufficient large number of *Locales* has been subscribed for. At the stage E in Figure 21, A_1 has awareness of 11 active users, which are then filtered through the AoI management thread, so that the parameter *T* described in section 4.2.3 is kept below the target selected, e.g. 10 users in the example above.

For the case of a very large number of users, an avatar will likely have awareness of a single *Locale*, with further filtering provided by the AoI management.

4.3.2 Data Transmission Control

Each object is supposed to send data only on its own OTC and only those who have explicitly signed for that channel will receive such data. That ensures that no host will ever receive unsolicited user data. Moreover each VELVET system can pinpoint exactly which users should be receiving data from, based on a given metric. Table 3 shows the amount of superfluous data received by participants in various architectures.

Table 3. Superfluous Data

MASSIVE-2	Transmission from objects in the appropriate areas, which are of no interest to the local participant.
SPLINE NPSNET-IV	Transmission from objects in the appropriate areas, which are of no interest to the local participant.
SCORE	Transmission from objects in the appropriate areas, which are of no interest to the local participant, as well as objects which, even though not in the Area of Interest, are located in cells which fall partially within the AoI.
VELVET	Minimum superfluous data, based on the metric used.

VELVET is adaptive because each system may choose to sign for a larger or smaller number of groups on the fly. For instance, if a given system is connected to a VELVET World and experiences network overload, it may simply unilaterally shrink its own AoI (reducing its parameter *T*), which immediately reduces the flow of data arriving at that end. Additionally VELVET allows a heterogeneous set of hosts to successfully collaborate, since each user can have his/her AoI reduced as much as necessary to make it treatable, while increasing its Degree of Blindness. That would allow people in a supercomputer, as well as very limited systems, to collaborate in a CVE session.

4.3.3 Data Structures

Figure 22 shows how data are organized in a VELVET client. All starts with a linked list of *Locales*, each of which contains a list of boundaries, i.e. pointers to other *Locales* which are neighbouring each *Locale*, as well as a list of Active Objects (e.g. Avatars), which are located within that *Locale*. There is also a list of known avatars, i.e. those which the local user has awareness of through the procedure described above (Figure 21). Each known avatar has a list of metrics which are initialized with values provided by each active object and updated with corrected statistical values (such as average delays, jitter, etc.).

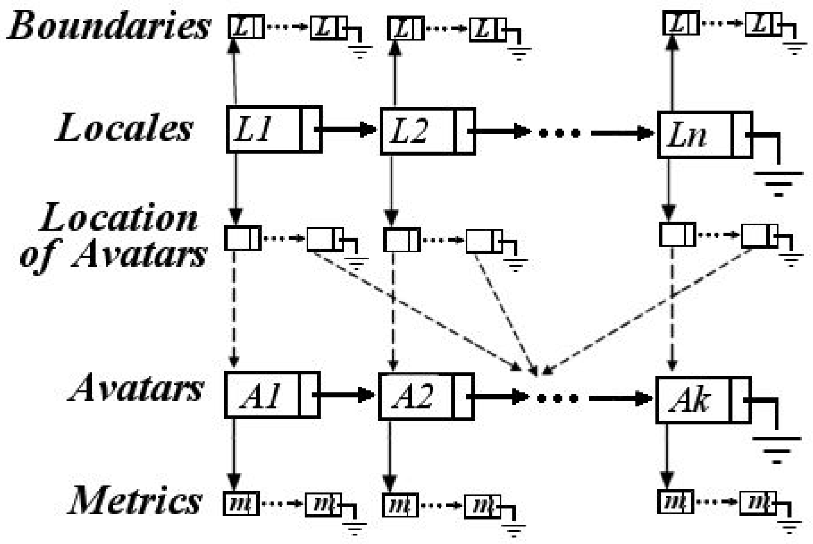

Figure 22: Internal Data Structures

The list of Avatars shown in Figure 22 is ordered according to the metric currently in use by the local user. One should note that all the information stored in these structures is dynamically gathered, unilaterally, by the local user.

4.3.4 Parallel Virtual World and Filtering of Messages

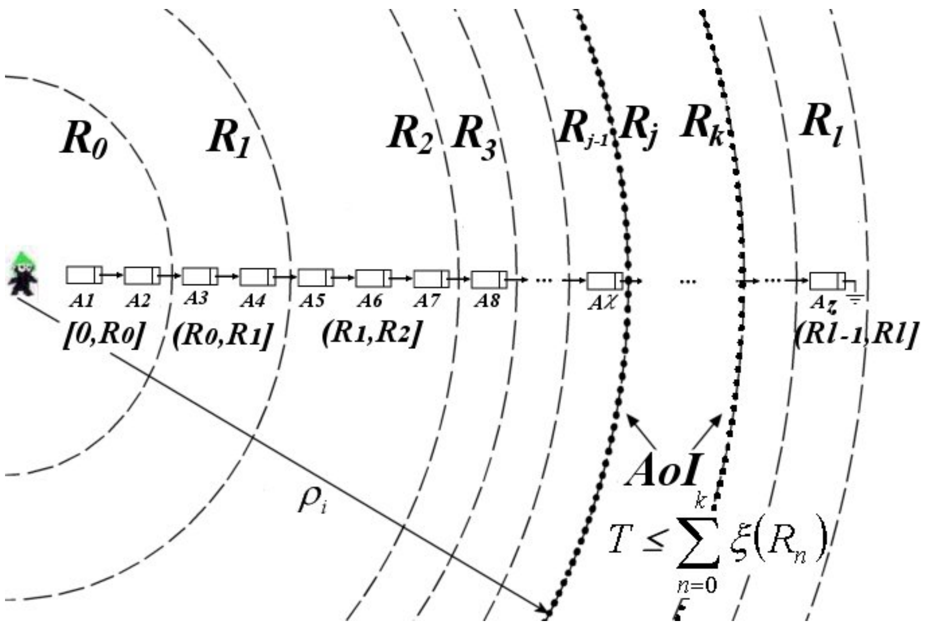

Figure 23: PVW in the scope of Internal Structures of VELVET

Figure 23 shows how the PVW actually works in the light of the structures shown in Figure 22. The list shown in Figure 23 is just the “Avatars” list from Figure 22. The idea as that as such list is ordered based on the metric chosen. The levels R_0, R_1, …, R_l, are defined based on the metric values known at the moment. For instance, if we consider that the metric chosen is the traffic required by a given avatar, the levels in the PVW could be set as 1Kbps, 2Kbps, …, α Kbps, where α would represent the maximum transmission rate known at the moment. The active objects which have transmission rates within the [0, R_0] interval, namely [0kbps, 1Kbps] would be in the first level of the PVW and so on. The AoI would be defined as the levels R_j and R_k, respectively for AICI and AICO. j and k are chosen so that *T*, $\sum_{n=0}^{j}\xi(R_n) \leq T \leq \sum_{n=0}^{k}\xi(R_n)$, is optimized and kept under the target maximum metric as defined in section 4.2.2.

VELVET is a hybrid protocol because it builds on top of existing *Locale* based systems. The Area's multicast group (LTC) is used to send/receive only management packets, such as ENTERLOCALE, JOIN LOCALE and LEAVELOCALE (i.e. packets which are related with Awareness management). The VELVET' Session Management entity will frequently analyze the structures shown in Figure 22 and 23 based on their profile and will choose which ones to sign up for/drop.

4.4 Modeling and Simulation

VELVET has been modeled using OPNET Modeler 6.0 PL12. In such modeling, we created a multicast enabled router which allowed VELVET to perform exactly as described in section 4.3, with all procedures and data structures as described in sections 4.3.1 and 4.3.3. Figure 24 (A) shows the Node Model for a VELVET station. The three main threads of the VELVET architecture (namely transmitter, receiver and management) are represented as three independent nodes. Figure 24 (B) shows the process model for the Management Node. Such model consists of a startup procedure required due to the DHCP server mentioned above followed by an infinite loop where the thread wakes up every once in a while (TIMEOUT) to analyze the current content of the structures presented in Figure 24 and perform changes that may be required. Such changes include subscribing/dropping OTCs, joining/leaving *Locales*, etc.

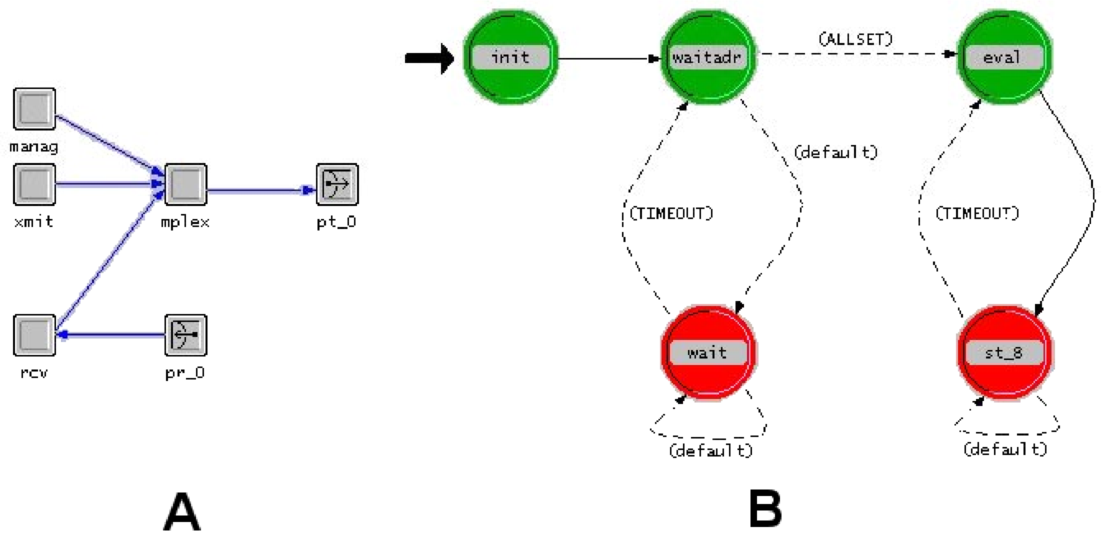

Figure 24: VELVET's Host Node/Process Models

The modeling of VELVET uses external configuration files which allow one to select the statistical behaviour of each station along with the geography of the VELVET World. For most simulations, we used the World shown in Figure 25, where 9 *Locales* exist with well-defined neighbourhoods. Several stations were placed and different profiles were chosen.

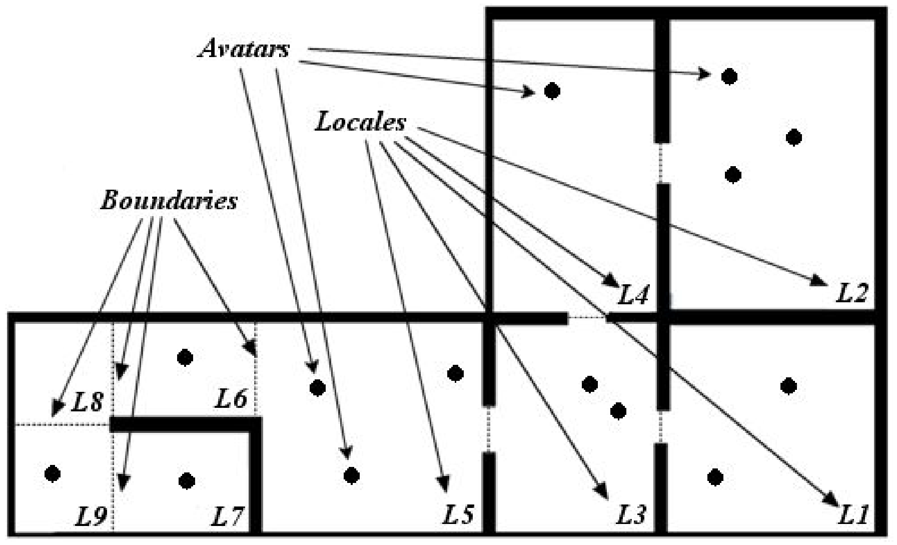

Figure 25: Virtual World

In order to make the setup of the simulation yet more convenient, we have introduced a DHCP server in the simulation. Each station requests this server to assign an IP address for it at start up. This allows easy reconfiguration since it is possible to add many stations to the simulation with no need for manual configuration. This DHCP setup leads to unnaturally high packet traffic at startup, but such high traffic only occurs approximately in the first 500 ms of the simulation. All packet exchanges after this time interval are those of VELVET management, along with data packets sent by the many stations.

Figure 26 shows the Average Number of Packets received by four stations selected in the network. Those are four stations with profiles to view 1, 5, 9 and 12 stations respectively. It is obvious to see that the lines well separated in the graph represent a clear relationship of the number of users a station chooses to join and the number of packets received. The graph does not show a station changing the metric on the fly but, if the station with metric target set at 12 reduces its metric target to 9 stations, the AoI manager is led to drop 4 stations. The number of packets would then fall from 110 to the 75 packets/second average experienced by the station with metric target set at 9.

Figure 27 shows results from 13 stations. One can see 12 somewhat well-defined data flow levels, which reflect the fact that the hosts were set to "see" from 1 to 12 stations. For this simulation all stations were sending packets according to an exponential distribution with a mean rate of 0.125 seconds (an average of 8 packets per second). The probability of an avatar changing a *Locale* was set to 0.15.

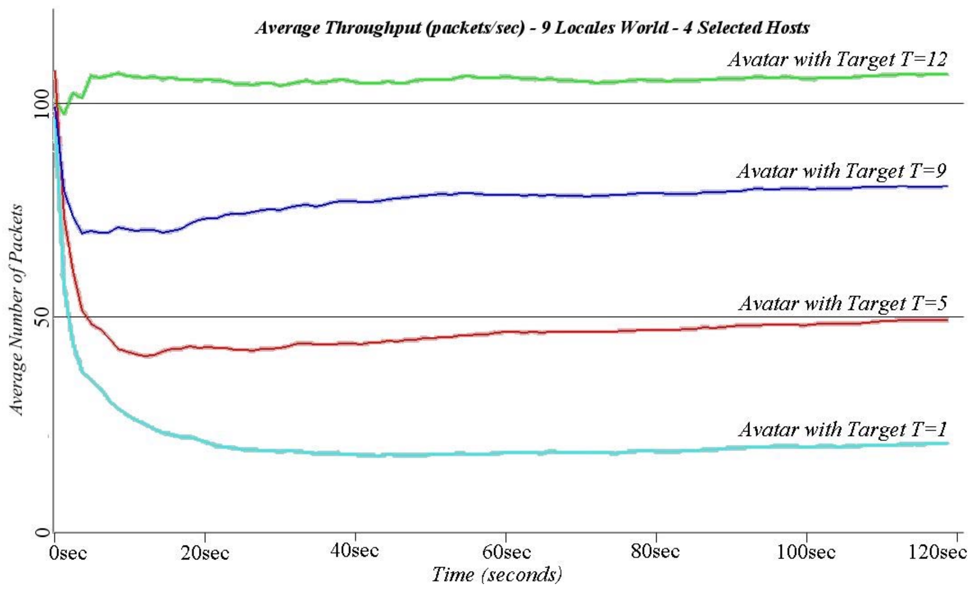

Figure 26: Incoming Traffic – Average Throughput in packets/second

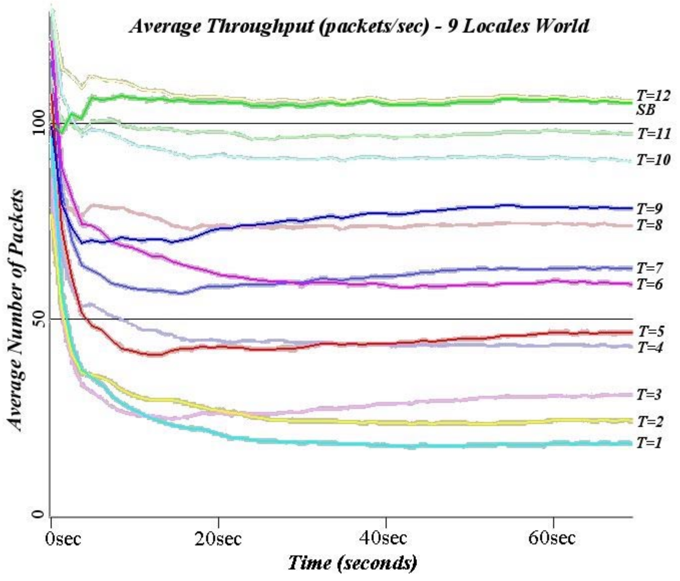

Figure 27: Incoming Traffic – Average Throughput in packets/second

The graphs of Figure 26 and 27 show that VELVET supports heterogeneity well, since each station may select a different level of service. This can be extended for Quality of Service purposes where a user could stay in the level most adequate according to paid fees (or load in the network). Such levels could be for instance those with metric targets set at 1, 5, 9 or 12 in Figure 26.

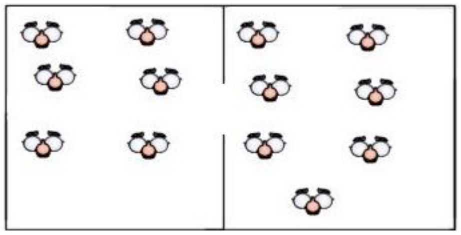

Figure 28: A Two-*Locale* World

The simulation has also been executed in a two Area World (Figure 28). In such a World, SPLINE would lead to the rendering of all users populating the World, since all objects in the current *Locale*, as well as the immediate neighbour *Locales*, are subscribed for. A given user would, hence, receive packets from every active object in this specific World. Figure 29 shows the performance of such simulation.

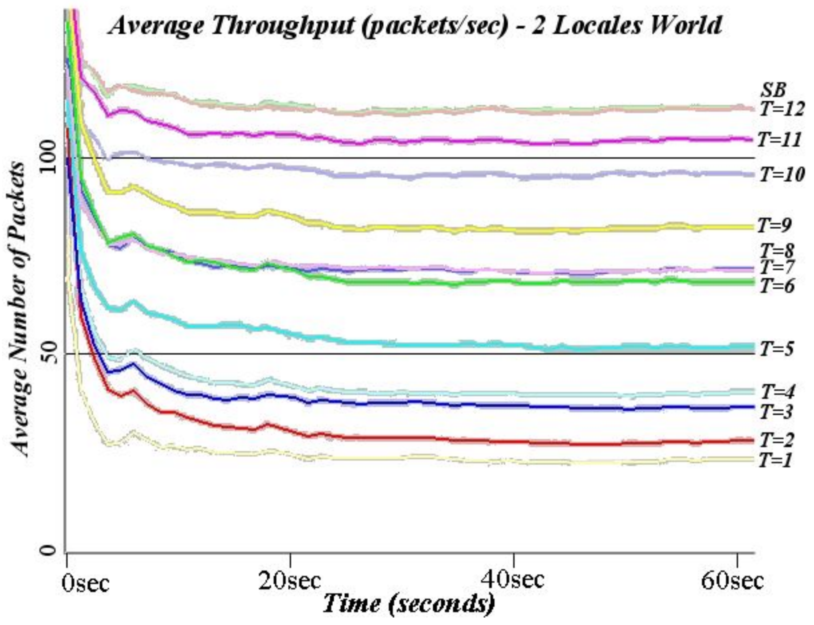

Figure 29: Incoming Traffic – Average Throughput in packets/second

The station behaving like SPLINE matches with that of the VELVET station receiving information from all hosts, averaging 115 packets per second, as shown in Figure 29. In our 2 *Locales* World, SPLINE would receive packets from absolutely all objects within the World. VELVET allows for filtering even within a single *Locale*, since the World is seen through the PVW for the AoI Management protocol of VELVET. If a given avatar chooses to expand both AICI and AICO so that they would coincide with the last level known, then all objects would check into the AoI of the avatar. More formally, if j=k=l so that $\partial_{jk} = 0 \Rightarrow \partial_{i\alpha} \leq \rho_j, \forall \alpha$, for an avatar A_i. In this special case, the avatar A_i in VELVET would receive packets from every object as well. For the other avatars, different values of ρ_j were chosen, leading to the various levels shown in Figure 29.

It is worthy to mention that the high traffic experienced at the first 500 ms is initially due to the use of a DHCP server, which has been introduced into the in order to allow fast automatic configuration of the stations. At the beginning of the simulation, all stations request IP addresses from the DHCP server through the use of broadcast packets. The response from the server is also sent via broadcast, which increases the number of incoming packets at startup.

This section has shown advantages of VELVET compared with *Locale* based architectures such as SPLINE and NPSNET-IV. In fact, if one defines a metric in VELVET of subscribing to all objects in the avatar's *Locale* as well as the immediate neighbours, VELVET would behave just like SPLINE; hence they can be thought as a subset of VELVET.

Figure 30 shows results of a simulation with up to 209 users and 25 users increments. The line shown at the top varying from about 280 incoming packets per second (when 34 users are in the Virtual World) up to about 1600 incoming packets per second (when 209 users are within the Virtual World) shows how Space Based Solutions (e.g. SPLINE) behave when the user population grows. VELVET on the other hand stays stable between 70 and 100 incoming packets per second.

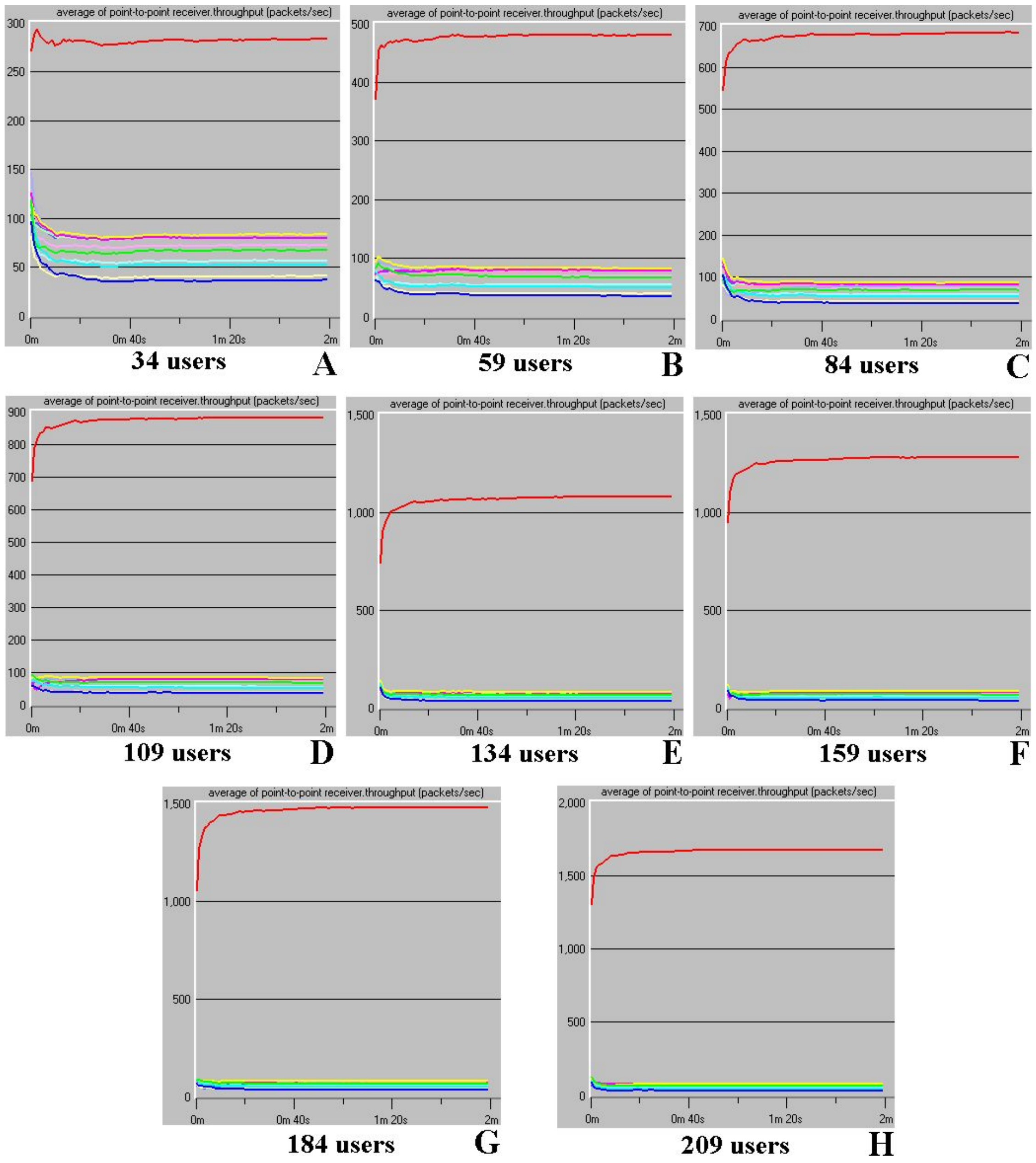

Figure 30: Incoming Traffic – Average Throughput in packets/second

Comparison of VELVET (12 lower lines) vs. Space Based Solutions (upper line)

With 34 users, VELVET stations had incoming traffic between 40 and 70 packets per second while space based solutions (SBS) had over 275 incoming packets per second (Figure 30.A). When 59 users were in the World, the results stayed almost identical for VELVET at 45 to 75 packets per second, while SBS achieved over 475 packets per second (Figure 30.B). SBS went further to about 690 packets per second with 84 users (Figure 30.C), 890 packets per second with about 109 users (Figure 30.D), etc., achieving over 1600 packets per second with 209 users (Figure 30.H). In contrast VELVET stations were stable at around 100 packets per second (Figure 30.A-H).

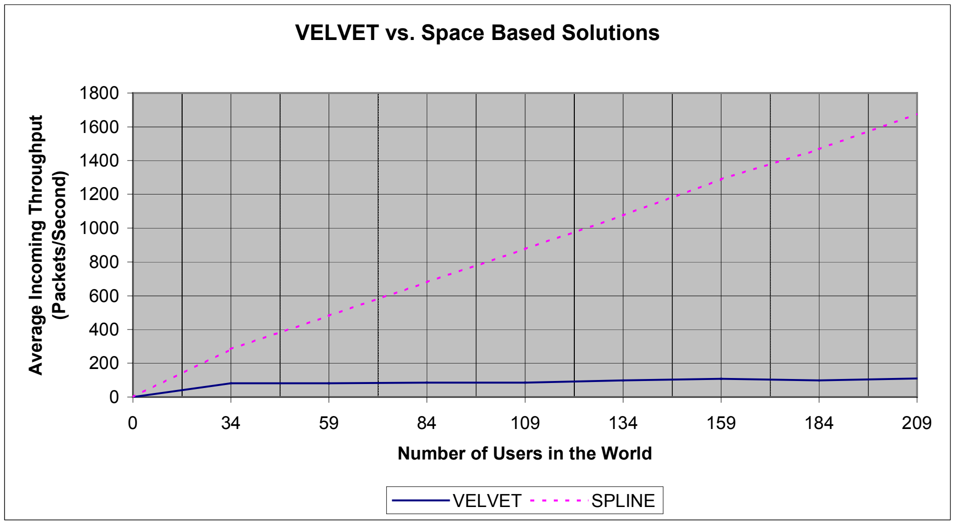

Figure 31: Comparison of VELVET vs. Space Based Solutions

Figure 31 shows a better comparison of VELVET vs. Space Based Solutions, using data gathered from simulation results shown in Figure 30. In this graph, we selected the VELVET station with the highest Incoming Throughput (i.e. that with the AoI more expanded amongst the various VELVET stations in Figure 30). We can see that this VELVET station keeps a relatively stable Average Incoming packet count, while Space Based Solutions grow steadily.

These simulation results shows how VELVET behaves when compared with *Locale* based architectures. Such results are somewhat obvious, since VELVET, being a superset, can behave exactly like SPLINE, as well as allow a more aggressive filtering of incoming data, hence reducing incoming traffic. It is important to mention that a VELVET station could have its AoI expanded enough so that it could achieve the same packet count shown above for Space Based Solutions. For example, assume a user A_i whose metric allows subscription to up to 250 other users. One advantage of VELVET is exactly this flexibility, allowing each user to unilaterally choose a metric appropriate to its own needs.

4.5 Comparison with Other Architectures

In this section we'll be comparing VELVET with other architectures, namely MASSIVE-2 and SCORE.

Greenhalgh presents studies [Gree97] of the total throughput of MASSIVE-2 as a means of comparison with MASSIVE-1. In such studies, the following parameters are used:

- I_A: number of artifacts in the World.
- I_P: number of participants in the Scope of Interest.
- S: average size of the state of an average object (in bytes).
- M: number of times an avatar changes its scope of interest.
- T_S: average amount of time spent by a user in the Virtual Environment (in seconds).
- B_P: average traffic generated by a participant (in bytes per second).

Traffic is given as the number of packets sent by each object in a given time and $\frac{SI_P M}{T_S}$ accounts for multicast announcement/state transfer of mobile users [Gree97]. When an Avatar changes its scope of interest (e.g. when entering in a different room with new content for instance), all objects will need to be loaded; hence we should expect $\frac{S(I_A + I_P)M}{T_S}$ average traffic due to the change of scope of interest. In addition to that, we also expect $I_P B_P$ which accounts for data transferred by the various participants.

Using this approach regarding incoming traffic, we can see that MASSIVE-2's incoming traffic is all together given by:

$$(I_A + 2I_P)\frac{SM}{T_S} + I_P B_P = \frac{I_A SM}{TS} + \left(\frac{2SM}{TS} + B_P\right)I_P \qquad \text{(Eq. 1)}$$

VELVET's incoming traffic is similarly given by:

$$(I_A + 2I_P')\frac{SM}{T_S} + I_P' B_P = \frac{I_A SM}{TS} + \left(\frac{2SM}{TS} + B_P\right)I_P' \qquad \text{(Eq. 2)}$$

Where $0 \leq I_P' \leq I_P$, i.e. VELVET has the second term of the left hand side half of Equation 2 which is responsible for incoming data such as audio, video, motion, etc., in a range from zero all the way up to the same I_P that MASSIVE-2 receives. If we assume that on the average each participant is interested in about 50% of objects he/she is aware of, it would lead VELVET to receive about 50% less data than MASSIVE-2. In other words, VELVET's users will not need to receive data from 50% of the objects they are not interested in.

Additionally, MASSIVE-2 has quite an overhead, not disclosed in the expressions above, which has to do with the management of Third Party Objects (see section 3.3). It should be remembered that some entity will be responsible for the non-trivial task of mixing the media of the objects, which are part of a third party object so that the mixed medium is transmitted at the level where the third party object is located.

SCORE's division of the World in cells brings heavy management overhead. Imagine that a given area of the World has a large number of users. SCORE should try to reduce traffic by increasing the number of the cells into a fine grid so that one can have a smaller Area of Interest. It happens that if the cells are small and a given user is moving around, that user will need to insistently subscribe for the new cell's multicast groups as well as leaving a number of them. Considering that more than one user would be moving at a given point in time, the number of join/leave multicast messages would grow to a significant amount of traffic, somewhat bringing back the traffic problem. In VELVET, it is of no relevance, in this regard, if users are moving around since the multicast groups one subscribes to are associated to the metric used and not the physical location of the objects. Only when a user crosses the borders of a *Locale*, will it send an enter/leave *Locale* message (Figure 20).

Also when SCORE decides to change sizes of cells, one may expect a peak of multicast group management messages sent by all users in the affected area.

Considering the situation where SCORE has a fix sized cell grid defined at start up leads to something similar to a *Locale* based World where each *Locale* is smaller. Similar problems found in those Worlds would be observed and the extra multicast group join/leave issue would lead to higher management traffic than standard *Locale* based solutions where *Locales* are usually larger.

4.6 Collision Detection

Collision detection brings some extra challenges to VELVET. It may happen that due to the allowed flexibility it is possible to have users A and B, where A is geographically aware of B while B has no geographical awareness of A as shown in Figure 17. If users A and B were to physically engage in collision, in the Euclidean World, only one of the parties would be aware of such collision. Remember that each user selects the objects that will be checked into the AoI based on its own PVW, which is not necessarily related to the Euclidean World. The collision issue discussed above is optionally resolved through Emergency Collision Packets (ECP) sent from the party who is able to detect collision to the party blind to it. In our example, shown in Figure 17, A would send such packet to B letting it know about the collision. B can create a representation of the object it is colliding with, so that it would be aware of such an event. As disclosed in section 3.3, each VELVET process signs up for some Areas (similar to SPLINE's *Locales*) and by doing so becomes aware about users who are in those Areas. An Avatar A_i may compare the list of users who are in a given area with the list of users which have subscribed for the local transmission channel of A_i. Such comparison allows an avatar A_i to determine whether or not another given entity A_j is geographically aware of A_i, allowing it to know if an ECP needs to be sent.

Some other collision detection schemes [Sing96] suggest the use of a server which would know about position and orientation of every object so that it would be able to inform objects about collisions. VELVET, aiming at an environment with a very large number of users, tries to avoid any kind of server; the reason why the collision detection scheme discussed above is advantageous. A disadvantage of this scheme is that the collisions between two objects which are not geographically aware of each other are not detected. Let us assume that object C is aware and has subscribed for both A and B. Object C may see A and B passing through each other, if A and B are not aware of one another, i.e. if $A \subset C$ and $B \subset C$ but $A \not\subset B$ and $B \not\subset A$ as defined in section 3.2.4. The ECP scheme can be extended to allow third party collision detection in which case C could send an ECP to both A and B letting them know about the collision. This extension would still allow collisions to occur, if both parties are unaware of each other and no other entity is aware of both A and B simultaneously. It is possible to choose a mixed metric which also takes into account position so that one object is more likely to "see" another one which is in the neighbourhood, if desired.

4.7 Recovery from Failures

Distributed systems should be able to recover as smoothly as possible from isolated failures. Systems that rely on a server for instance are challenged by the problem that arises when a failure happens at the Server station. In some cases that means that the session is immediately compromised and possibly aborted. VELVET does not use any server for transmission of data or management of the session. If one participant gets disconnected due to local failure, all that happens is that no more updates from that participant will be multicasted. VELVET processes which had subscribed for such resource will eventually drop it from their lists, after a certain time interval has elapsed, without communication. The procedure is not different from what happens when a client actually chooses to leave the Virtual Environment. The major difference is that before leaving an entity may let other interested parties know about it.

4.8 Summary

In this chapter we have introduced VELVET, a novel Adaptive, Hybrid Architecture for VEry Large Virtual EnvironmenTs. VELVET is one of the very first architectures to support heterogeneous collaboration, i.e. allowing participants in powerful computers and fast networks to successfully collaborate with others in more modest systems and 56K dial-up connectivity. VELVET also allows each participant to filter incoming data based on a metric chosen by that user alone. At the same time, we introduced the concept of Parallel Virtual World, which consists on a different arrangement of a Virtual World based on a user's metric. There are several Parallel Virtual Worlds, one for each metric. PVWs are built independently and transparently by each participant's Area of Interest management module of VELVET.

VELVET is a powerful architecture, which allows a larger number of users to interact, since limitations of space-based architectures are worked out in VELVET, which is shown in the simulation results presented in this chapter.

Chapter V

5 Enabling and Controlling Embedded Worlds

Collaborative Virtual Environment (CVE) designers have long relied on reuse of content while creating other CVEs. Figure 39 shows just one such example of reuse from SIGGGRAPH'1997 [WaAC97b, OC-SG97]. CVE content reuse has been historically performed manually, when a CVE designer would need to extract a portion of code of a CVE and reattach it to a new project, mostly saving just the repetitive description of the geometry of such environment. It has not been possible to copy sections of a CVE within another one in real time, i.e. without reprogramming.

Some recent work [KaSh00] has introduced the concept of real-time copying of sections of a given CVE into another one. That allows for quick creation of a large scale CVE, as a large number of pieces individually designed may be brought together into a single larger CVE.

This Chapter introduces a novel distributed approach which allows copies of a given section of a CVE to be kept consistent amongst all copies, as well as the original CVE area. Such approach is built upon existing functionality of CVEs, which allows its deployment without much interference on the way a CVE is designed. Additionally, this thesis introduces another approach to be used when less strict consistency is sufficient.

Section 5.1 presents background information. Section 5.2 introduces an approach for Full Consistency Control, as well as an analysis of compatibility with existing architectures.

Section 5.3 introduces some approaches for a less strict Consistency Control, and is followed by a conclusion.

5.1 Background

5.1.1. Hyperlinking and Embedding of Virtual Worlds

The World Wide Web (WWW) is a complete hypertext system. In the web, the unit of a page shown is a single HTML document. The hyperlinking mechanism enables the Web to be easily expanded and intuitively accessible as depicted in Figure 32.A. This may be one reason of the success of today's Internet. Hyperlinks also allow users to browse the same pages under different perspectives, depending on which page they came from before finding a given page (Figure 32.B). In CVEs, the function of portals resembles that of hyperlinking in the WWW. Even though current portal mechanisms have several problems, they are obviously an essential element for this kind of large scalable systems.

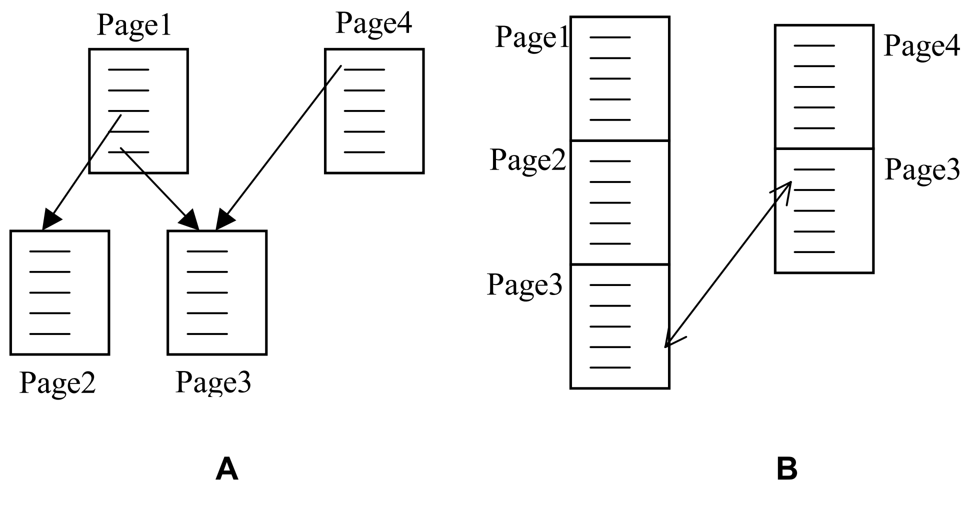

Figure 32: Expansion of WWW: A – Hyperlinking, B – Various Views.

This thesis presents a useful mechanism which is new in the CVE area, but common to other existing systems, such as hypermedia systems. Such mechanism is the idea of embedding objects.

5.1.1.1 Hyperlinking

Hyperlinking has similar characteristics as those of traditional portals in CVE. With regard to Hyperlinking, the conceptual object (anchor) is replaced with the real visual content when the linked area is activated. On the other hand, an embedded world is visible within the world in which it is inserted. The greatest difference between the two methods is whether or not a participant can see and interact with the expanded world and the current world simultaneously.

Similarly, portals are used as a method to overcome spatial limitations in CVEs. Besides a pure portal mechanism and *Locales* partitioning [Brol97, BaWA96b], some additional features are required to expand the scalability of CVEs.

Limitations of the Portal mechanism

- It performs world replacement to expand a CVE.
- It causes an inevitable navigation delay due to the world replacement.

Figure 33 describes common features in WWW and CVE regarding hyperlinking and embedding. A Hyperlink in an HTML document is analogous to a Portal of a CVE in that they both provide a way of expanding the local world view and teleporting to another virtual space. There was no such a mechanism in a CVE like embedding an object in an HTML document. It is possible to embed a world unit in CVEs as in the WWW. This gives new advantages but requires more considerations in CVEs.

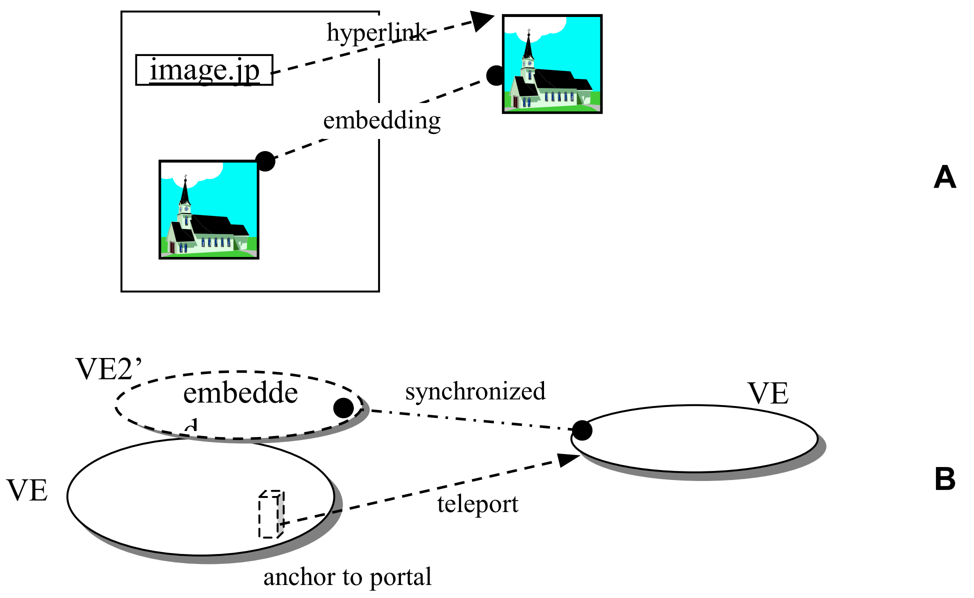

Figure 33: Hyperlinkng and Embedding in the WWW (A) and a CVE (B)

5.1.1.2 How to Embed a World Unit

Let us consider the following scenario, illustrated in Figure 34: Suppose that a participant P is navigating in a CVE world which supports World Embedding. When P finds a nice building and wants to use it in his/her own virtual world, P may simply copy this object's region and stitch it together with his/her own local World Space. If allowed, after embedding a region of the virtual world, P may also customize the shape and color of the embedded building, as well as to replace some components in the newly copied area.

Provided that the CVE supports certain consistency mechanisms for embedded worlds, the user P can see other participants moving within the newly copied object or, optionally, just the building itself.

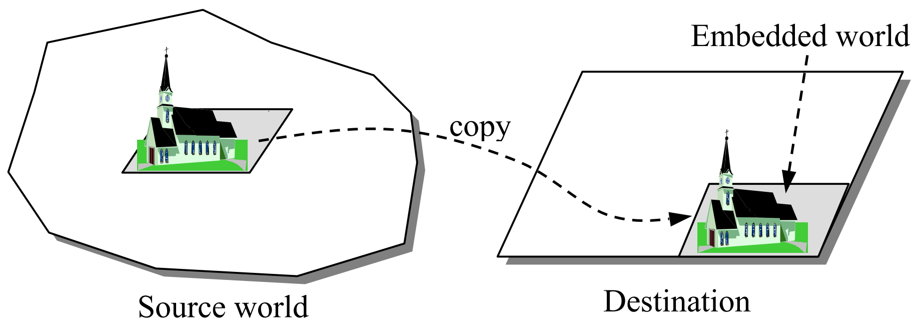

Figure 34: Copying a World Unit

Advantages of Embedded Worlds:

- Expandability: World Embeddeding may be used as a means of easily building a very large virtual world which can be seamlessly seen by several participants simultaneously.
- Navigation Delay Avoidance: When the Portal Paradigm is used, there is a delay when crossing the border between two worlds due to loading time. On the other hand, Embedded Worlds have no such additional delay when a participant enters the Embedded World. After all that is nothing but a part of the CVE in which it is embedded.
- Customizability: A user may freely customize its own world, depending on the consistency model in use.

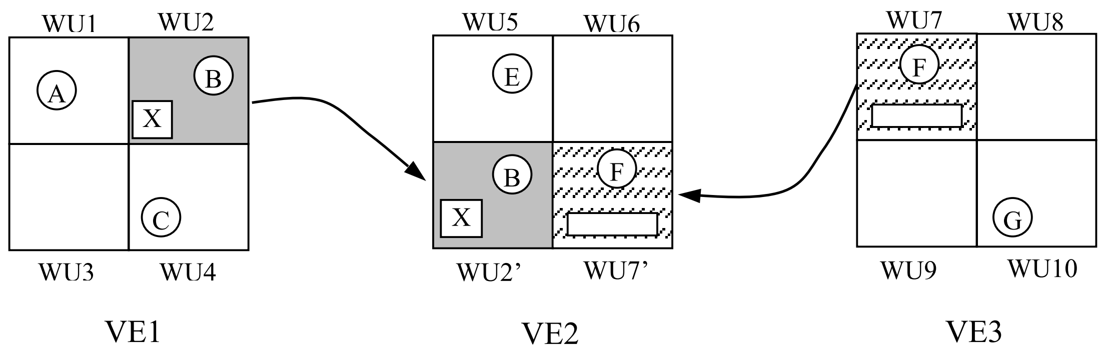

Figure 35: Expansion of VE through Embedding World Units

Disadvantages of embedded virtual worlds:

- It is not trivial to maintain consistency among multiple copies of an original world area.
- A mechanism to coordinate the relationship among the several copies of each section of a world needs to be defined.

Table 4 summarizes the characteristics of both Hyperlinking and Embedding mechanisms of CVEs. From such comparison, it is possible to notice that World Embedding requires some procedures for consistency maintenance among multiple copies of the world areas. Some approaches for consistency are introduced in sections 3 and 4.

Table 4. Portal vs. World Embedding

	Portal	**World Embedding**
How to Expand	Hyperlinking	Importing (copying) a world unit.
Number of Copies	One	Number of times the world was copied plus one.
Characteristics	- Current world and linked world are not shown at the same time. - Current world must be replaced when navigation to a linked world is activated. - World navigation delay.	- Without any replacement process, the whole world~~s~~ is seamlessly seen, and one can freely navigate through several different world areas. - No navigation delay after the initial pre-loading process.
Consistency	- Doesn't need to keep consistency because there is only one copy of the World.	- Must keep consistency because there are many copies of the World.

5.1.2 Related Work

Previous work [KaSh00] has suggested the idea of allowing various Virtual Worlds (VW) to be hyperlinked. The basic idea is to allow a user to copy a section of a CVE to a local "personal cyber-space world" [KaSh00], so that it can be used as a means of increasing the number of users in such local World (which is now "more abundant" [KaSh00]). That would easily lead to a Large-Scale Virtual Environment (LSVE), with a potentially large number of users. Such expansion may be implemented via portals, which are a path from one World to another. The use of such portals would require both Worlds to support such notion; after all there would be a new area neighbouring the area where the portal is placed (and both Worlds would need to know what to do, if a user gets through the Portal).

One other option for such hyperlinks is suggested at [KaSh00], where a local copy of the remote World, or part of it, should be made so that such VW would in fact become part of the local one. This, for example, would facilitate the duplication of the Desert House of Figure 39. Such approach brings as an advantage a reduced complexity and as disadvantage some consistency control concerns. The complexity is reduced by the fact that participants in the original remote World do not need to be represented locally, and a VW designer wouldn't need to create content from the scratch, as useful areas in existing Worlds could be imported into others. Consistency issues lie in the fact that modifications made to a copy of the World should be reflected in the original VW, and possibly in the other copies of it. This introduces some challenges not covered by [KaSh00], one of which is the need of a mechanism to keep consistency amongst the various copies of an Area of a World.

5.1.3 Concurrency Control in Collaborative Virtual Environments

In a Collaborative Virtual Environment (CVE), when several participants are going to access a shared object simultaneously, a certain control mechanism is needed to coordinate such concurrent actions. Awareness, responsiveness and consistency are considered as the principal requirements that CVEs have to satisfy. In particular, the trade-off between responsiveness and consistency causes a dilemma in the design of a CVE. According to the priority between consistency and responsiveness, concurrency control approaches are categorized as pessimistic and optimistic ones [SuYW99].

The followings are the related terminology when controlling concurrency of actions in a virtual environment:

- Requester: A participant who needs a token to manipulate an object
- Facilitator: A single central server that coordinates the request competition by message serialization.
- Token: A permit that allows a participant to change the state of an object
- Object Ownership Database: A database that manages the object token ownership
- Version Number: A number updated whenever there is a modification in the state of an object.

5.1.3.1 Pessimistic Concurrency Control

If a system guarantees that there is no inconsistency caused by concurrent operations at any time, it is called a pessimistic concurrency control. Early versions of DIVE (Distributed Interactive Virtual Environment) [Hags96] that was developed at the Swedish Institute of Computer Science employed a locking mechanism. CoCAD, a collaborative CAD system, has a central communication server, called the facilitator, which serializes all updates [GiSa94]. These are appropriate for systems that require a strong consistency, for example, database systems. The biggest problem of the pessimistic approach is a slow responsiveness. It was reported that this approach did not scale to more than 10 peers on a local area network, partly because the locking mechanism did not scale well [Hags96]. Such approach is especially unfit for the case where not many users wish to manipulate the same object at a given time (for instance, consider a CVE with few participants).

5.1.3.1.1 Locking by Token Passing

A locking mechanism is a concurrency control method that a node has to acquire the lock (token) before each manipulation to the virtual environment and then sent the update message to its peers. A token is assigned to each object that must be obtained before manipulation of the object.

Figure 36 shows a scheme of token passing, in which case a requester (participant) must, in order to move an object, check the object for availability through the Object Ownership Database (1). If such object is available, the requester gets a token and is able to manipulate the object (2). Otherwise, the requester is notified who the current token owner is. The requester, then, asks the token owner for the token (3), and the owner, eventually, sends the token to the requester (4).

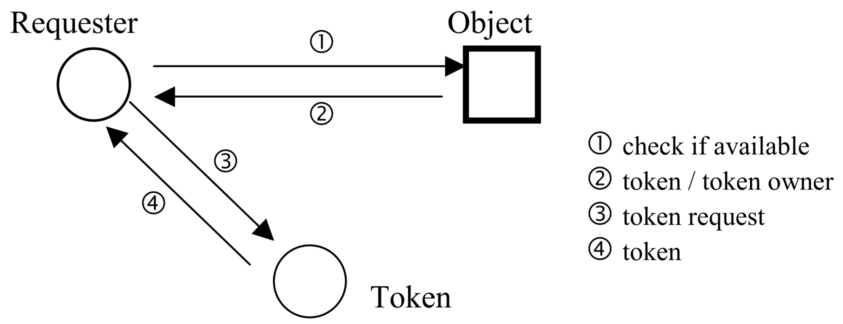

Figure 36: Token Passing

5.1.3.1.2 Serialization

A facilitator is a central communication server which coordinates all actions of each participant. To manipulate an object, the participant must send an update request to the facilitator (1), which serializes the update requests by a certain ordering policy (2). The consistency of the environment is easily kept because all updates are processed in a single server. However, there can be a bottleneck in a central facilitator when the number of participants increases.

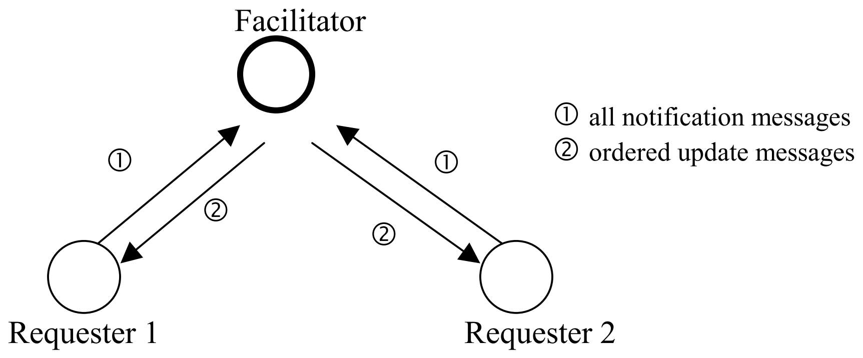

Figure 37: Serialization

5.1.3.2 Optimistic Concurrency Control

The concept of optimistic concurrency control has been used by various systems [ElGi89, ElGi91, GrMa94]. Figure 38 shows such control [SuYW99]. In the optimistic approach, a participant can operate an object without any waiting, and send a notification message containing the object identifier, the version number, and its action (1). The token owner receives the message, and verifies whether the version number is valid. If it is valid, then the owner validates the action by sending a message that the ownership is transferred to the node that started the manipulation (2). This validation message can cancel other concurrent manipulation of the same object. Compared to others, the advantage of this approach is an instant responsiveness to the virtual object.

The Optimistic Concurrency Control has disadvantages in the case where many users are consistently requesting ownership of a given object in parallel, since many will send update requests just to have them refused. One such example of CVE with potential problems is that with a large number of users, which raises the probability of more than one user expecting to manipulate the same object at a given time.

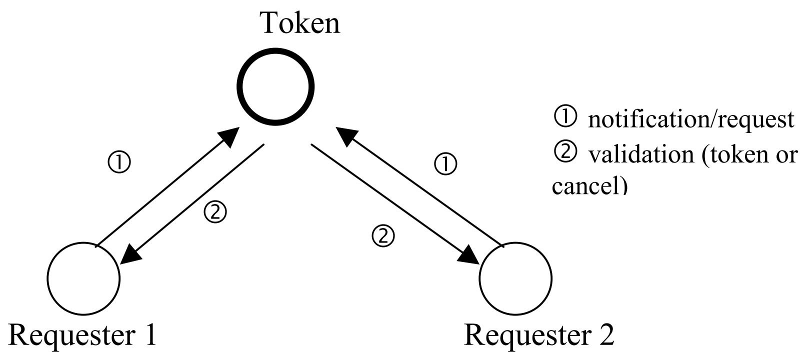

Figure 38: Optimistic Concurrency Control

5.1.3.3 Multi-Level Consistency Management for Embedded Worlds

A system supporting world embedding should provide various consistency options for world embedding.

Such a system has the following choices to keep the consistency among the embedded worlds.

- No consistency policy after copying a world.
- Partial consistency policy.
- Full consistency policy.

The full consistency policy covers all copied nodes and supports strict concurrency control. The partial consistency policy is useful when there is interest in keeping just some of the nodes consistent, providing a flexible concurrency control. The no consistency policy is the simplest scheme, which may be used when no consistency after embedding is required.

Each node can choose an appropriate consistency policy and degree from other nodes according to its own maximum throughput.

The first policy is nothing but a trivial copy of content with no consistency control. That could be seen as a set of one-nodded trees in a forest as shown in section 5.3. The second policy is covered in section 5.3, while the last policy is discussed in the section 5.2.

5.2 Full Consistency Control

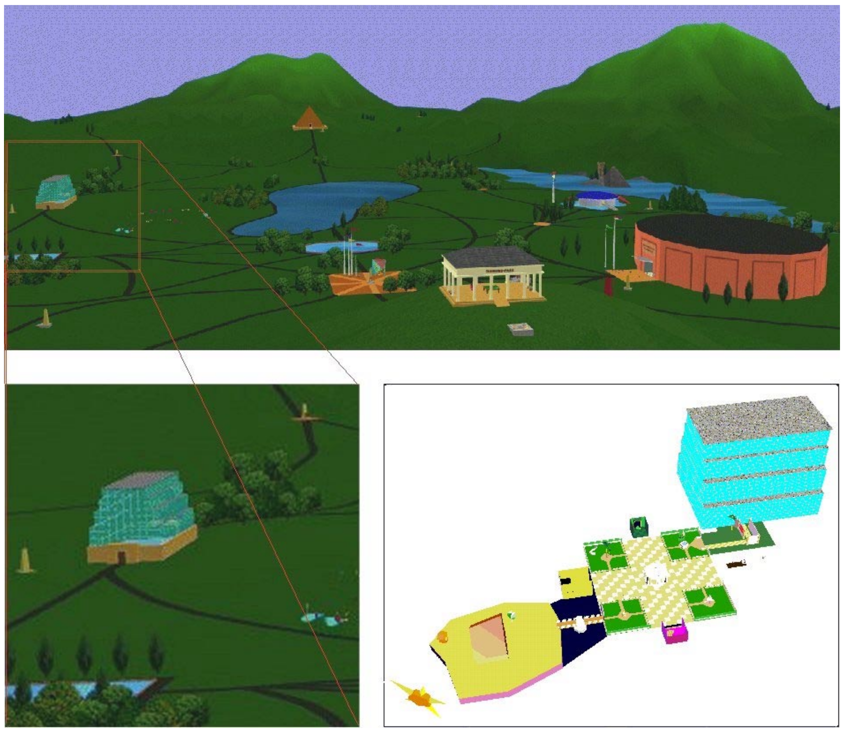

Figure 39: Desert House in Diamond Park (top) & SIGGRAPH'97 Demo (right).

When a designer is creating a new CVE, it is useful to have some parts of some previously designed CVEs used. As an example, we can see that the SPLINE demo presented at SIGGRAPH'97 [OC-SG97] by the MERL team has used the "Desert House" which was originally designed within the Diamond Park [WaAC97b], as shown in Figure 39.

Reuse of parts of a CVE within others, even though convenient, has always been a rather labourious procedure. The CVE designer needs to manually extract a section of a given CVE geometry definition and include it within the target CVE. No provision exists to keep consistency among a number of copies of sections of a CVE.

Subsequent sections present our contributions. Section 5.2.1 introduces a new approach, aiming at keeping consistency among the various copies of sections of a CVE, whose Ownership Management is presented in section 5.2.2. Section 5.2.3 presents some analysis of the approach introduced, while section 5.2.4 comments on limitations of such algorithm. Finally, section 5.2.5 comments on multi-platform consistency control and section 5.2.6 introduces a methodology which may be used to achieve better performance.

5.2.1 Consistency Control

The copies of sections of a CVE build up a tree, where the original VW is located in the root and the copies (as well as copies of copies etc.) are branches and leaves. In order to copy a section of a CVE a user would just need to know the location of such section and copy it from there. It is important to notice that the location a user is copying from may already be a copy of some other CVEs (and recursively a copy of a copy etc.). Figure 40 shows such a tree structure, in which W is the original world (root), with first level copies W_1, W_2, …, W_n. A copy W_i may have copies made out of it as well, which is shown in Figure 40 as $W_{i.1}$, $W_{i.2}$, etc.

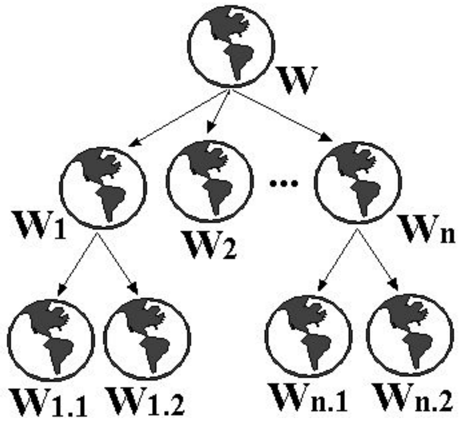

Figure 40: Virtual World Copy Tree Structure

We can assume that copies are made based on *Locales* [Bawa96b, BaWa96], being possible to copy a number η, $0<\eta<\infty$, of *Locales* from a given CVE. At the same time one may choose to copy sections from various CVEs in parallel, where several trees (one for each *Locale* is created).

As long as a given user is able to join a CVE, the procedure described in this section allows consistency control to maintain all copies consistent. VELVET [OlGe02] supports the above-mentioned idea of Hyperlinked Worlds. In VELVET, when a copy of a *Locale* $W_{i\ldots j.k}$ is made, an object Θ_k is placed in the original *Locale* $W_{i\ldots j}$, which is the parent in the tree structure. The terminology in VELVET defines the copy as Local Virtual World (LVW), the original *Locale* as Remote Virtual World (RVW), and the object Θ_k as RVW Agent (RVWA). The RVWA will act on behalf of the whole community which is located in the corresponding LVW. When a modification is performed in the RVM, the RVWA performs such change in the LVW, so that consistency is kept. When a user in the LVW wishes to perform a modification in the LVW, the RVWA will attempt to get ownership of the objects involved in the RVW. Upon successful ownership transfer the RVWA allows the local user to make the desired change, which is then performed in the RVW by the RVWA. This procedure ensures consistency between a LVW and its parent in the tree – the equivalent RVA.

Consistency with siblings is available automatically through the procedure mentioned above. When a RVWA Θ_k makes changes in the RVA it is located at, other RVWA Θ_l, Θ_m, etc. from siblings which similarly reside on the same RVW, will notice such modifications and perform them in their own LVW, hence ensuring consistency among siblings. A similar procedure will allow changes to reach every node in the tree.

As an example, let us consider an avatar which pushes a chair away from a table in a LVW. The appropriate RVWA will perform such operation in the RVW, so that both Worlds are kept consistent. Moreover, such change will be forwarded to parents and other children in the tree structure, again keeping all copies consistent.

5.2.2 Ownership Management

The Distributed Ownership Control Mechanism requires that an ownership request be successfully forwarded upwards, with successful ownership transfer granted at the root node, i.e. if a given user wishes to handle an object, an ownership request would be sent to the parent node and further up in the tree. If no one else wishes to handle the same object at the moment, the ownership request will be granted once it hits the root of the tree successfully. If, however, such request reaches a node whose other child has been granted the ownership of the object in question, the original ownership request fails and such request is not sent any further, just the denial is sent back to the requester. Figure 41 shows such procedure, where the shaded area indicates nodes where an ownership request is outstanding. Let us suppose that some avatar residing the RVW $W_{1.1}$ wishes to modify some object $\alpha_{1.1}$ (Figure 41.A). Such action leads the RVWA which resides at W_1 to try to get ownership of the object in that level (Figure 41.B). Such request keeps on going up on the tree, until it reaches the root (Figure 41.C). If that's the only request for the object α in question the ownership is successfully transferred and the avatar residing in $W_{1.1}$ may finally handle the object.

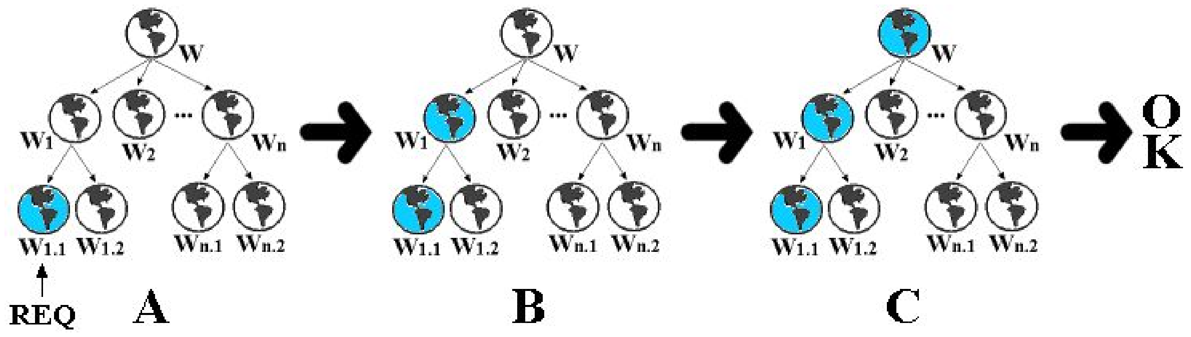

Figure 41: Ownership Management

In the event that more than one avatar tries to handle the same object β in various RVW, the procedure is depicted on Figure 42. Let us suppose, without loss of generality, that, at the same time, an avatar residing on the RVW $W_{1.1}$ and another avatar residing in another RVW $W_{n.2}$ wish to handle the same object β, represented as $\beta_{1.1}$ and $\beta_{n.2}$ respectively in the $W_{1.1}$ and $W_{n.2}$ RVWs (Figure 42.A). Such a request will go up the tree (Figure 42.B, until, if successful, it reaches the root node (Figure 42.C). At this stage, the Ownership Management Subsystem (OMS) will verify that more than one avatar wishes to handle the same object β. At this stage, the OMS will use some <u>metric</u> to choose which request should be granted and which should be refused. Such metric may be as simple as “the first request that arrives” or as elaborated as the request which comes from the smallest depth in the tree or according to some pre-defined priority scheme. The use of a metric ensures that several methodologies may be employed, depending on the choice made by the VW designer.

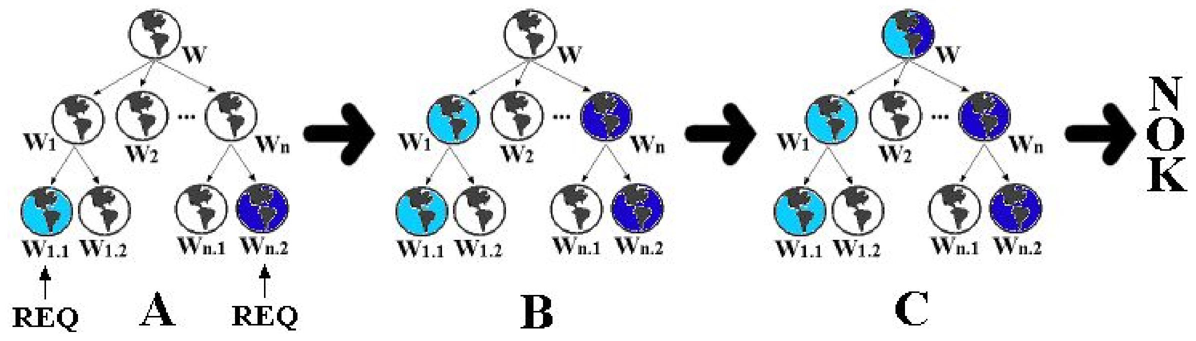

Figure 42: Collision of Ownership Requests at Root Node

It is important to notice that a Collision of Ownership Request (COR) doesn’t necessarily happen at the root level. Figure 43 shows an example where the collisions happen at some lower level. In such an example, avatars residing on the RVW $W_{1.1.1}$ and RVW $W_{1.n.2}$ request ownership of an object δ at the same time (Figure 43.A). Such object is represented in the RVWs respectively as $\delta_{1.1.1}$ and $\delta_{1.n.2}$. Such request goes up the tree as usual (Figure 43.B), until a level where a COR happens, which in our example happens on level W_1 (Figure 43.C). Such requests are not sent any further and the OMS of W_1 will elect the request which should continue the path upwards and those which should be denied right away, according to the metric in use. Such behaviour releases the load from the root node (the original copy of the VW), as descendents may cut off several requests, forwarding a single request per branch (in the worst case).

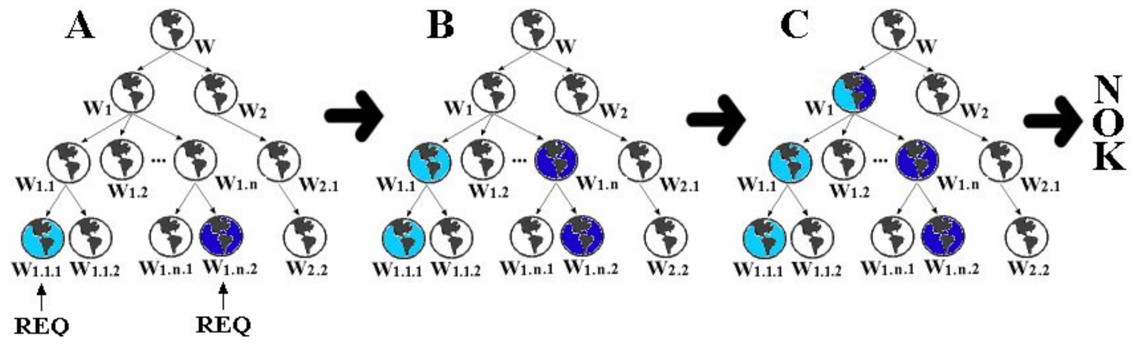

Figure 43: Collision of Ownership Requests at Intermediate Node

A request will need to travel no more than the depth of the tree, which has an average at O(lg n) for a balanced tree, with n being the number of copies of the original VW (nodes in the tree). That means that as the number of RVWs grows, the delays, due to forwarding of ownership requests, grow slowly. One may also expect a user to make a copy of a given area of a CVE from a known repository which is relatively close (geographically wise) to such user. For instance, suppose that each country hosts a copy of the Museum of Louvre. A second level copy is more likely to come from the local copy in a given country than from the original VW. Assuming that such behaviour holds true, the networking delay from a node to its descendents is expected to be small, which allows minimization and approval of Ownership Transfer Requests to be processed quickly and efficiently.

The support for Hyperlinked Worlds in VELVET is built upon the simple ownership locking mechanism, which is a built-in feature of most CVE architectures. The procedure mentioned above could be used by other architectures as well. Both token based and serialized architectures are possible. In the case of serialization, one can notice that each node works as a facilitator of its children. Such architecture avoids the bottleneck which a central facilitator, as mentioned in section 2.2.1.2. Such facilitator is in fact distributed amongst the various branch nodes in the trees shown in this section.

5.2.3 Performance Evaluation

In this section we'll compare the approach introduced by this thesis with an ad-hoc solution in which consistency is maintained through a single server, which may as well be understood as the original CVE. In other words, such ad-hoc solution would have all nodes being children of the same (root) node, as depicted in Figure 44.

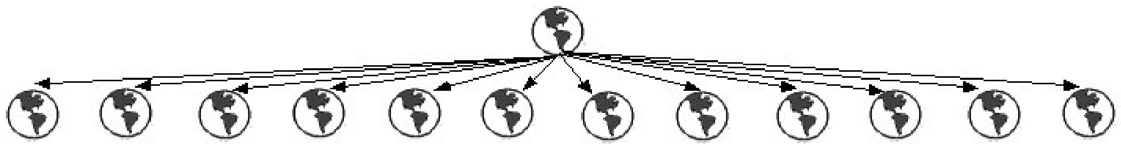

Figure 44: Ad-hoc solution with centralized consistency control

If we define:

- α : The average number of children a node has in the tree, which is also referred to as "degree of node" [CoLR90];
- β : The depth (height) of the tree;
- η : The Number of nodes in the tree (number of copies of the CVE section).
- δ : The average node-to-node networking delay between nodes in the tree; and

We can conclude initially that $1+\sum_{i=0}^{\beta-1}\alpha^i \le \eta \le \sum_{i=0}^{\beta}\alpha^i$, while $\beta = \mathrm{O}(\log\eta)$ [Knut97, CoLR90], assuming that such tree is balanced (see section 4.7 for the case in which the tree is not balanced). More specifically for a binary tree, $\beta = \lceil \log_\alpha(\eta+1)\rceil - 1$ and $\alpha^\beta \le \eta \le \alpha^{\beta+1}-1$.

The ad-hoc solution, as shown in Figure 44, also builds up a tree, with height β=*1* and degree of node $\alpha = \eta - 1$ (all nodes, except the server node – root in the tree).

The maximum delay for an ownership request to be granted will be $2\beta\delta$, or $2\delta[\mathrm{O}(\log\eta)]$ for our approach and 2δ (fix). It is important to add that for the analysis in this section we assume that processing time for an ownership request is negligible, for the sake of simplicity (one can also assume that such cost is built into δ).

We can further define λ the average rate of ownership requests for a given object σ. The expected number of requests, related with the object σ alone, reaching the root of the tree would be expressed by $\alpha\lambda$, being expressed as $(\eta - 1)\lambda$ for the ad-hoc solution shown in Figure 44. It is worth to mention that η grows steadily, as it is number of nodes in the tree, while α may be constant. In fact, for a balanced tree in the worst case $(\eta - 1)\lambda$ will lead *to* $\left[\left(\sum_{i=0}^{\beta}\alpha^{i}\right)-1\right]\lambda$, or $\mathrm{O}(\alpha^{\beta})\lambda$ to simplify. That shows the advantage of the approach introduced in this thesis, especially for a Large-Scale CVE, with many objects, participants and, therefore, many ownership requests being sent. In short, the delay in the new approach grows slowly, with a complexity of $\mathrm{O}(\log\eta)$, while the number of requests grows quickly with a complexity of $\mathrm{O}(\eta)$ for the ad-hoc solution.

For an example with numbers, let us assume that a given CVE has been copied 14 times, and that such copying builds up a balanced tree as shown in Figure 45.

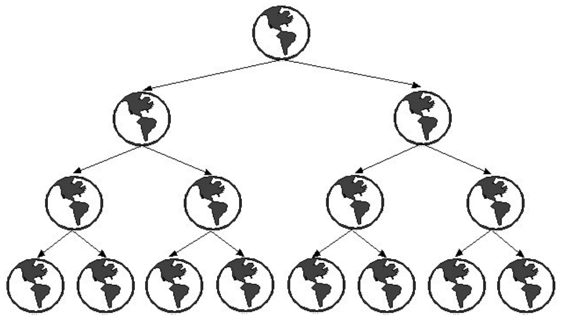

Figure 45: Test Case A

One can notice that in this tree $\alpha = 2$, $\eta = 15$ and $\beta = 3$. That means that the expected number of requests arriving at the root is expressed by 2λ, while the ad-hoc solution would have 14λ requests arriving per unit of time, just for one of the objects in the CVE. On the other hand, the delay is expected to be 6δ (or lower) for the new approach and 2δ for the ad-hoc approach.

For a Test Case B with 31 nodes in the tree (a full balanced tree with depth 4 and 2 children per node), the results would be as follows:

- The original CVE receiving the same 2λ requests per unit of time, while the ad-hoc solution would jump to 30λ such requests per unit of time.
- The maximum delay would rise to 8δ for our new approach and stay at 2δ for the ad-hoc solution.

5.2.4 Limitation on Other Architectures

The architecture above makes very few assumptions about a CVE in order to enable consistency control over copies of CVE areas, but there are still a few issues which may introduce limitations on its usage with other architectures. In this section, we'll discuss briefly how the Consistency Control algorithm described in section 5.2.2 and 5.2.3 could be implemented in CVEs based on SPLINE, NPSNET-IV, MASSIVE-2 and SCORE.

5.2.4.1 SPLINE

Sharing of *Locales* in SPLINE [BaWA96] would work as described in section 5.2.2 and 5.2.3 with no need for modifications. SPLINE implements the spatial division of the CVE in areas, called *Locales*, which would be the natural unit of copying/sharing for which a tree as shown in Figure 40. Additionally, SPLINE provides Ownership Management. Exactly one object at a time has ownership of a given object. Before being able to manipulate an object which is owned by another object, one must request ownership through a pair of spOwnershipRequest() and spOwnershipRequestGrant() calls.

5.2.4.2 NPSNET-IV

NPSNET-IV [Mace95, MZPB94] also has the CVE partitioned in well-defined hexagonal areas. Such areas would also work well as a unit to be considered for copying/sharing in the tree structure described above. Regarding Ownership Management, NPSNET-IV… NPSNET was designed to comply with DIS 2.0.3 [Mace95], adding multicasting and the lack of keep-alive heartbeat messages. Regarding Ownership control, since NPSNET-IV is build upon DIS, such feature is not fully supported, as entities are not expected to interact with each other (but shoot them). Section 3.4.5 discusses options for the event that ownership control is not fully implemented. The newly introduced NPSNET-V implements full Ownership Control.

5.2.4.3 MASSIVE-2

MASSIVE-2 [Gree97, GrBe97] implements Ownership Control to some extent since “each artifact is (and remains) under the control of a single object”, as disclosed on [Gree97]. As per partitioning of the CVE, MASSIVE-2 does not implement the idea of Locales, but rather that of third party objects. Third party objects recursively groups a number of objects and other third-party objects. At a first level a MASSIVE-2 may have third party objects defining what would resemble a Locale. These first level third party objects would make a good partitioning mechanism candidate, i.e. the unit to be copied as described above.

MASSIVE-3 implements both partitioning and ownership control through the notion of Locale and a comprehensive Ownership Control mechanism.

5.2.4.4 SCORE

SCORE [Lety00, LeTB99] brings some challenges with regard to the partitioning of the CVE. It happens that SCORE divides the world in small cells, as suggested on [HoRC94]. There are basically two methods of cell partitioning: static and dynamic. In the static partitioning all cells are pre-defined and hard coded in the CVE. Such cells could work as the unit for copying sections of a SCORE CVE; however, as such cells may be rather small, this may lead to an undesirable large number of units (cells) to be managed through copy trees as presented above. To make things worse, SCORE also allows for dynamic cell allocation, in which case such cells may be merged and split according to the concentration of objects in the CVE. With such merge/split option, it becomes impossible to keep track of a well-defined area in the CVE, which hence prevents the use of the copying scheme presented above.

5.2.4.5 Other Architectures

In general, there are two simple requirements for the mechanism defined in sections 5.2.2 and 5.2.3 to be usable:

- The CVE has some well defined, and fixed, partitioning, which may be used as a basis for users to copy individual areas, which are generally called *Locales*, as introduced by SPLINE.
- The CVE provides some means to allow a single user at a time to manipulate a given object.

There are alternatives for CVEs where no ownership management control is offered, i.e. CVEs which permit more than one user at a time to handle a single object. Such work around may consist of allowing the RVWA to manipulate a given object on behalf of the child (descendent to be more precise) and, upon multiple manipulation, forward the resulting position/shape of the object as a result of the action of the various users. For instance, if a descendent object is pushing a chair southwards with intensity 5, and another object is pushing the same object in the opposite direction with intensity 3, what the RWMA will forward to all descendents is the resulting motion of both forces applied simultaneously. In our example a motion southwards of intensity 2.

In short, as long as it is possible to have a well-defined area of a CVE copied, with ownership control being beneficial, the procedure described in sections 5.2.2 and 5.2.3 would allow for consistency control amongst several copies of such area of a CVE.

One example of other architecture not mentioned above would be the High-Level Architecture (HLA) [ZhGe01], which has been developed by the Department of Defence of the US to replace DIS. HLA became an OMG and IEEE standard [IEEE1516]. HLA provides both ownership management through the Ownership Management Service. An object must acquire the ownership of another object prior to being allowed to modify it. The two requirements mentioned above are, hence, available, which means that the approach presented in this thesis would also apply to HLA compliant CVEs.

5.2.5 Multiarchitectural Consistency Control

The above procedure is obvious in the case where all copies of a given CVE area are running under the same architecture. For instance, if all nodes shown in the trees of Figures 40 through 16 are all VELVET compliant CVEs, one can notice that no potential problems exist. Similarly, if all nodes are based on SPLINE, all should be fine as well. If, however, some of those copies are built into VELVET compliant CVEs, while others are based on SPLINE, for instance, it is not obvious that it would all work smoothly.

Table 5 shows a case-by-case analysis, indicating those combinations which would work smoothly, as well as those which would work partially or not at all.

Table 5. Multiarchitectural Consistency Control

	VELVET	SPLINE	NPSNET	MASSIVE	SCORE	HLA
VELVET	OK	OK	OK♠	OK*	Partially	OK
SPLINE	OK	OK	OK♠	OK*	Partially	OK
NPSNET-IV	OK♠	OK♠	OK♠	OK*♠	Partially♣♠	OK♠
MASSIVE-2	OK*	OK*	OK*♠	OK*	Limited*♣	OK*
SCORE	Partially	Partially	Partially♣♠	Limited*♣	Partially	Partially
HLA	OK	OK	OK♠	OK*	Partially	OK

* Assuming that, in MASSIVE-2, the first level Third Party Object is fix and can hence be used as a unit for copying. MASSIVE-3 implements the idea of *Locale*, which is a natural choice for partitioning, in which case no problems exist.

♣ It is rather problematic, but if the SCORE world has fixed sized cells it could still be possible.

♠ NPSNET-IV provides a well-defined partitioning scheme, which is the basic requirement for consistency control. There are some potential issues regarding Ownership Control, but that is manageable as mentioned in section 5.2.5.5. NPSNET-V provides full Ownership Control.

Additionally one must expect to render content from other architectures as well. Most architectures mostly provide means for communication and consistency control. It is usually up to each implementation to render each object in the required position. For instance, SPLINE has the rendering implemented in the spVisualBody() function. By default SPLINE uses OpenGL to render the interface to the user; however such function could be rewritten to use other technologies, such as Java3D, ActiveX, etc. Integration of various architectures must allow a user to receive the content of a world and allow such content to be rendered independently. One would be able to render the content of a VELVET Area into a SPLINE World. Most architectures are friendly in this regard.

5.2.6 Balancing of Trees

The approach presented in this thesis works quite well if the tree resulting of copies is balanced, i.e. if $\forall$ external node ω_i in the tree, the depth of the branch of the tree which reaches such node is either β-*1* or β, where β is the depth of the tree. It is easy to see why such feature is beneficial, since the maximum delay associated with an ownership transfer is upper bounded by the depth of the tree and a balanced tree has the minimum depth for each degree of node α [CoLR90], as defined in section 4.4. In this section a procedure to be employed when a tree is not balanced will be presented. Such procedure will allow a non-balanced tree to become balanced.

5.2.6.1 Tree Rotation and Child Adoption

As performed in data structures, when a tree is not balanced and one desires to transform it into a balanced tree a number of "rotations" is required [CoLR90]. A simplification of such mechanism consist in simply allowing the adoption of a child node by another branch which is at a lower depth and which still has some missing child. A branch node has a missing child if the number of children of such node is smaller than the degree of the nodes of the tree α [CoLR90].

In the case of our CVE area copying tree, an adoption of a child would mean that a CVE which has originally been copied from a given location will end up transferred to have another location as a parent. In other words, a node $\omega_{i.n}$ which was originally keeping consistency through a branch node ω_i, will be required to keep consistency through another branch node ω_j. Such node adoption may be performed in such a way to minimize delay. I.e. amongst a set of candidate adopters (ω_j, ω_k, ω_l, etc.) of a node $\omega_{i.n}$, the system may elect the candidate adopter node ω_α so that the delay between the adoptee and the chosen candidate adopter $\delta(\omega_{i.n}, \omega_\alpha)$ is minimized.

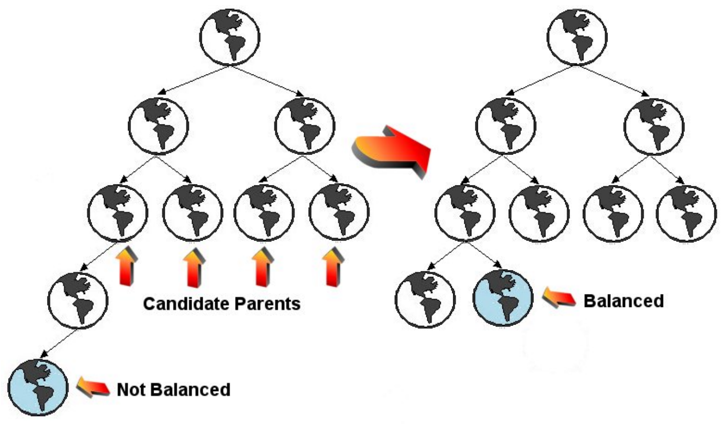

Figure 46: Child Adoption and Balanced Trees

This procedure allows a tree to eventually become balanced, which is the optimum case for the approach presented in this thesis. Figure 46 shows an unbalanced tree (left) with a series of candidate parents to adopt the marked node. Once the adoption takes place the tree is balanced (right). Of course this procedure doesn't need to be enforced continuously. In fact, it is beneficial to allow the tree to be unbalanced to some degree in order to avoid unnecessary node adoptions when additional nodes may be added and naturally bring the tree to balance. The adoption procedure may be performed just when the tree unbalance is greater than some pre-defined threshold, for instance if the difference in depth of two leaves is found to be greater than 3.

5.3 Partial Consistency Control

In this section, we introduce a more relaxed consistency control mechanism, in cases where full consistency control is not required (or desired). This mechanism provides a greater flexibility at the price of lack of full consistency.

The approach we introduce for partial consistency control is based on the full consistency control mechanism presented in section 5.2. The idea is that instead of having a single tree for each section of a Virtual World, we may have a forest, with consistency kept just within each tree of such forest. The basic idea is that a user would have two options while copying a World:

- keep consistency with the parent: the new node becomes a leaf of the tree where the VW area was copied from.

- not keep consistency with the parent: the new node becomes the root of a new tree in the forest related to a given VW area.

Figure 47 shows an example of a forest, with a couple of single nodes (single-node trees – left) as well three trees (right). The first would be examples of an individual who copied a World and decided not to keep it consistent with any other copies of that world, while the latter have consistency control at each tree level. It is important to notice that at some point some other user may decide to copy from such a single-nodded tree, hence creating a branch in such tree.

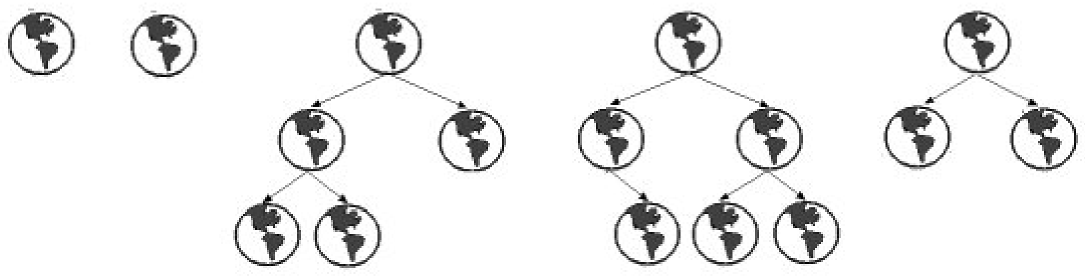

Figure 47: A Forest – Partial Consistency Based on Full Consistency

All that was commented in Section 5.2 applies to each tree of a forest, such as the ones shown in Figure 47. In fact, Section 4 discloses a special case in which all nodes have chosen to keep consistency with their "parents". Each section of a Virtual World which is eventually copied will define a forest with one or more trees in it.

5.4 Summary

This chapter has presented the notion of Embedded Worlds, which is a useful mean to create Large-Scale Virtual Environments through the replication of sections of existing CVEs within other CVEs. We have introduced a number of approaches for Full Consistency Control and

Partial Consistency Control, as well as presented an analysis of usability of some of the Consistency Control mechanisms proposed with existing CVE architectures.

The idea of Embedded Worlds is attractive, since it allows a Large-Scale Virtual Environment to be built quickly. Some of the consistency control mechanism may also be used to allow multiplatform collaboration. As they only depend on existing partitioning and ownership control management, such techniques may be deployed in today's CVE Worlds.

Chapter VI

6. Conclusion

In this thesis, we have studied a number of issues on Very Large Collaborative Virtual Environments. That includes a mechanism which may be used to create Large Scale Collaborative Virtual Environments, through the idea of embedding world areas within other worlds. We introduce two new methods for consistency control among the various copies of a given world area, namely full consistency and partial consistency control among the various copies.

We also presented VELVET, a novel Adaptive Hybrid Architecture for Large Scale Virtual EnvironmenTs (LSVE). VELVET has been shown to be flexible, allowing a broad range of LSVE, which would otherwise fail, to work gracefully. VELVET allows heterogeneous systems to successfully collaborate, i.e. users in supercomputers and very fast network connections to successfully collaborate with other users in not-so-powerful stations behind a 56K modem connection. That has been recently suggested as a desirable feature in CVEs [NPSNET-V, GrPS00]. VELVET also allows a greater level of adaptation, which translates to a greater number of users being able to collaborate through a CVE.

The combination of Embedded Worlds and the VELVET architecture shows great potential for Large Scale Collaborative Virtual Environments.

One disadvantage over other architectures is that VELVET makes use of a potentially larger number of Multicast Addresses, which leads to a potentially large number of entries in routing tables of various routers. More specifically, VELVET uses a Multicast Group (MG) for each Area, as well as for each non-artifact object, hence the cost regarding MGs in VELVET has an upper bound O(M+P), with M being the number of Areas in the Word (*Locales*) and P the number of participants. SPLINE's lower bound MG is O(M) since each *Locale* has a multicast group. MASSIVE-2 has an upper bound at O(M+P) as well, when considering the case where every couple of objects creates a third party object. SCORE has a lower bound MG usage at O(C), where C is the number of cells, which is much larger than the number of Areas (*Locales*) M.

It is worth mentioning that, even though VELVET has an O(M+ P) number of MGs, each router only needs to deal with those subscribed for stations whose traffic goes through it. Part of the metric of a given station could also include some cost measurements of the load in routers in the neighbourhood. That would allow a VELVET-based system to drop MGs, if some multicast table in some router on the way to that station is overloaded. The adaptability of VELVET and asymmetric presentation of the World for the various participants allows such features without much overhead, since all changes can be performed unilaterally by each participant.

It is also worthy of mention that, as technology evolves, routers get faster and more powerful and such limitations tend to be diminished as time goes by. Regarding the number of multicast addresses available, IPv6 is increasing their number to the same range of the total IPv4 unicast addresses available today.

One other way of seeing VELVET's advantage is to consider a precision parameter P_r, defined as the amount of desired data divided by the amount of data received. In other words, the rate of "unsolicited" traffic compared with the traffic which is of interest for a given user. In the ideal case P_r would be 1. We discuss below a case-by-case comparison of the various architectures:

- In an ad-hoc scheme where every user receives updates from all objects (such as SIMNET), P_r would reduce as the number of objects rise, approaching zero at the extreme case.
- *Locale* based solutions would have P_r decreasing, as a greater number of users join a single *Locale*; This parameter would approach zero in the extreme case as well.
- MASSIVE-2's case would be similar to SPLINE's, potentially having P_r approaching zero as well. Additionally one could have special interest in an object which is part of a (potentially recursive) third-party object. That would mean that such user would not receive updates from that specific object unless the third-party object boundary is crossed. This could bring P_r to a value greater than 1.
- SCORE, also being space based, would suffer from similar degradation of P_r since unsolicited data from objects which happen to be outside of the AoI may still be "subscribed" for, if the cell where they are located is partially within the AoI.

VELVET on the other hand would allow one to subscribe only to those objects one is really interested in. Some objects of interest may be dropped due to lack of resources rather than the inability of subscribing for them. Additionally objects that get dropped will be those of lower priority among all objects that one is aware of. VELVET would hence optimize P_r to be as close to 1 as possible, taking into account the user's processing power, networking connection and routing as well as the metric chosen by that user.

Suggestions for further research:

It would be beneficial to allow a participant to choose amongst several levels of detail of objects within a given Virtual World. That would allow a user with a slow speed connection and not as powerful station to choose a lower quality level, filtering out yet more incoming data. Such feature could be included in VELVET to enable such enhanced filtering mechanism.

Security in CVEs is an issue not broadly addressed. An architecture could provide encryption and authentication services in order to protect sensitive content, as LSVE systems are expected to be used through the Internet Worldwide. We also consider developing such features within the VELVET architecture.

Results from Hysteresis curves should be inspected as a methodology for AICI/AICO control.

As per Embedded Worlds, we expect to develop new consistency control mechanisms which would allow a more flexible and fast control.

More work is needed to allow truly collaborative Virtual Environments, i.e. those in which several users can manipulate a single object simultaneously. Some enhanced Ownership Control mechanism is required to enable such feature, such as that presented in [GrPS00].

References

[BaWA96] Barrus, J.W.; Waters, R.C. and Anderson, D.B. (1996) "*Locales* and Beacons: Efficient and Precise Support for Large Multi-User Virtual Environments", *Mitsubishi Electric Research Laboratory TR-95-016a.*

[BaWA96b] Barrus, J. W., Waters, R. C., & Anderson, D. B. (1996). *Locales*: Supporting Large Multi-User Virtual Environments. *IEEE Computer Graphics and Applications*, 16(6), 50-57.

[BeGa92] Bertsekas, D., & Gallager, R. (1992) Data Networks, Second Edition, *Prentice Hall*.

[Brol97] Broll, W. (1997) Populating the Internet: Supporting Multiple Users and Shared Applications with VRML. *Proceedings of the VRML'97 Symposium, Monterey, CA, ACM SIGGRAPH*, 87-94.

[CaWe96] Calvin, J.; Weatherly, R. (1996) An Introduction to the High Level Architecture (HLA) Runtime Infrastructure (RTI), *Technical Report DMSO/Mitre Corporation, 1996.*

[CMBZ00] Capps, M., McGregor, D., Brutzman, D., & Zyda, M. (2000) NPSNET-V: A New Beginning for Dynamically Extensible Virtual Environments, *IEEE Computer Graphics and Applications, 20(5), pp. 12-15.*

[CoLR90] Cormen, T., Leiserson, C., & Rivest, R. (1990). Introduction to Algorithms. *MIT Press*.

[DiGa99] Diot, C., & Gautier, L. (1999). A Distributed Architecture for Multiplayer Interactive Applications on the Internet. *IEEE Network (July/August).*

[DuFo98] Durbach, C.; Fourneau, J-M (1998) Performance Evaluation of a Dead Reckoning Mechanism, *IEEE DIS-RT 1998.*

[ElGi89] Ellis, C. A., and Gibbs, S. J. (1989) Concurrency Control in Groupware Systems, *ACM SIGMOD International Conference on the Management on Data, pp. 399-407.*

[ElGi91] Ellis, C. A., and Gibbs, S. J. (1991) Groupware Some Issues and Experience, *Communications of the ACM, 34(1), pp. 38-58.*

[FaTo98] Farcet, N. & Torguet, P. (1998) Space-Scale Structure for Information rejection in Large-Scale Distributed Virtual Environment, *IEEE Virtual Reality Annual International Symposium (VRAIS'98), Atlanta, GA.*

[FrSt98] Frécon, E., & Stenius, M. (1998). DIVE: A Scalable Network Architecture for Distributed Virtual Environments. *Distributed Systems Engineering Journal*, 5(3), 91-100.

[GiSa94] Gisi, M. A., and Sacchi, C. (1994) Co-Cad: A collaborative Mechanical CAD System, *Presence, MIT Press, Vol. 3, No. 4, pp. 341-350.*

[GrBe97] Greenhalgh, C., & Benford, S. (1997). A Multicast Network Architecture for Large Scale Collaborative Virtual Environments. *Multimedia Aplications, Services and Techniques – ECMAST'97, Lecture Notes in Computer Science*, Vol. 1202, ISBN 3-540-63078-3, Springer-Verlag, 1997.

[GrMa94] Greenberg, S., and Marwood, D. (1994) Real Time Groupware as a Distributed System: Concurrency Control and its Effect on the Interface, *ACM CSCW 1994, pp. 207-217.*

[Gree96] Greenhalgh, C. (1996). Dynamic Embodied Multicast Groups in MASSIVE-2. *Technical Report NOTTCS-TT-96-8, Department of Computer Science, The University of Nottingham, UK.*

[Gree97] Greenhalgh, C. (1997) Large Scale Collaborative Virtual Environments. *Ph.D. Thesis, Computer Science Department, University of Nottingham, UK.*

[GrPS00] Greenhalg, C., Purbrick, J., & Snowdon, D. (2000) Inside MASSIVE-3: Flexible Support for Data Consistency and World Structuring, *Proceedings of the third international conference on Collaborative Virtual Environments*, 119-127.

[Hags96] Hagsand, O. (1996) Interactive Multi-user Virtual Environments in the DIVE System. *IEEE Multimedia* (Spring), 30-39.

[HoRC94] Hook, D. J. V., Rak, S. J., & Calvin, J. O. (1994) Approaches to Relevance Filtering. *Proceedings of 11th DIS Workshop*.

[IEEE1278] Institute of Electrical and Electronics Engineers, International Standard, ANSI/IEEE Standard 1278-1993 (1993) Standard for Information Technology, Protocols for Distributed Interactive Simulation.

[IEEE1516] Institute of Electrical and Electronics Engineers, International Standard 1516 (2000) IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Framework and Rules.

[ISO11172-2] ISO/IEC 11172-2 1993 - Information technology - Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s -- Part 2: Video.

[KaSh00] Kato, K., & Shimamura, K. (2000) Reproductive Cyberspace for User Proliferation, *Globecom'2000 VRTS Workshop, San Francisco, CA, USA.*

[Knut97] Knuth, D. E. (1997) The Art of Computer Programming, Volume 1: Fundamental Algorithms. *Third Edition, Addison Wesley*.

[Leig99] Leigh, J. at al., (1999) A Review of Tele-Immersive Applications in the CAVE Research Network, *IEEE Virtual Reality – VR'99, Houston, TX, 1999.*

[LeTB99] Lety, E., Turletti, T., & Bacelli, F. (1999). Cell-based Multicast Grouping in Large-Scale Virtual Environments. *Techinical Report 3729 INRIA*.

[Lety00] Lety, E. (2000). Une Architecture de Comunication pour Environnements Virtuels Distribues à Grande-Echelle sur l'Internet. *Ph.D. Thesis, L'Universite de Nice-Sophia Antipolis, France*.

[Lora92] Loral Systems Company, (1992) Straeman Distributed Interactive Simulation Architecture Description Document Volume 1, *Technical Report, Advanced Distributed Simulation Technology Program Office, Orlando, FL March 1992.*

[LZGS84] Lazowska, E. D.; Zahorjan, J.; Graham, G. S. and Sevcik, K. (1984) Quantitative System Performance: Computer System Analisys Using Queueing Network Models, *Prentice Hall, 1984.*

[Mace95] Macedonia, M. (1995). A Network Software Architecture for Large-Scale Virtual Environments. *Ph.D. Thesis, Computer Science Department, Naval Postgraduate School, Monterey, CA, USA.*

[MIL3] Mil3 Inc. OPNET Modeler – Simulation Kernel Manual. Vol. 1 and 2.

[MZPB94] Macedonia, M., Zyda, M., Pratt, D., Barham, P., & Zeswitz, S. (1994). NPSNET: A Network Software Architecture for Large Scale Virtual Environments. *Presence*, 3(4), 265-287.

[Oliv96] Oliveira, J.C. (1996) TVS - Um Sistema de Videoconferência, *M.Sc. Thesis, Computer Science Department, Pontifical Catholic University of Rio de Janeiro.*

[OlGe02] Oliveira, J., & Georganas, N. (2002) VELVET: An Adaptive, Hybrid Architecture for Very Large Virtual EnvironmenTs. *Submitted for publication*.

[OlSG00] Oliveira, J. C., Shen, X., & Georganas, N. D. (2000). Collaborative Virtual Environment for Industrial Training and e-Commerce. *IEEE VRTS'2000 (Globecom'2000 Conference's Workshop on Application of Virtual Reality Technologies for Future Telecommunication Systems), San Francisco, CA, USA.*

[OHSC00] Oliveira, J. C., Hosseini, M., Shirmohammadi, S., Cordea, M., Petriu, E., Petriu, D., & Georganas, N. D. (2000). VIRTUAL THEATER for Industrial Training: A Collaborative Virtual Environment. *Proc. 4th World Multi-Conference on Circuits, Systems, Communications & Computers (CSCC 2000), Vouliagmeni, Greece*.

[OlSG99] Oliveira, J. C., Shirmohammadi, S., & Georganas, N. D. (1999). Collaborative Virtual Environments Standards: A Performance Evaluation. *IEEE DiS-RT'99, Greenbelt, MD, USA*.

[OlSG00] Oliveira, J. C.; Shirmohammadi, S. and Georganas, N. D. (2000) A Collaborative Virtual Environment for Industrial Training, *IEEE Virtual Reality 2000 (VR'2000).*

[OlSG00b] Oliveira, J. C., Shen, X., & Georganas, N. D. (2000). Collaborative Virtual Environment for Industrial Training and e-Commerce. *IEEE VRTS'2000 (Globecom'2000 Conference's Workshop on Application of Virtual Reality Technologies for Future Telecommunication Systems), San Francisco, CA, USA*.

[OlYG02] Oliveira, J. C., Yu, S. J., & Georganas, N. D. (2002) Enabling Embedded Worlds in Very Large Virtual Environments, *Submitted for publication.*

[Sing96] Singhal, S. (1996). Effective Remote Modeling in Large-Scale Distributed Simulation and Visualization Environments. *Ph.D. Thesis, Computer Science Department, Stanford University, CA, USA*.

[SiZy99] Singhal, S., & Zyda, M. (1999). Networked Virtual Environments, Design and Implementation. *ACM Press/Addison Wesley*.

[SuYW99] Sung, U. J., Yang, J. H. and Wohn, K. Y. (1999) Concurrency Control in CIAO, *IEEE VR'99, pages 22-28, Houston, Texas USA, March 1999.*

[Tane97] Tanembaum, A. S. (1997) Operating Systems: Design and Implementation, *Prentice-Hall International, 1997*

[WaAS97] Waters, R. C., Anderson, D. B., & Schwenke, D. L. (1997). The Interactive Sharing Transfer Protocol Version 1.0", *MERL Technical Report TR-97-10.*

[WaAC97b] Waters, R., Anderson, D., Casey, M., Yerazunis, & Sterns, I. B. (1997) Diamond Park and Spline: A Social Virtual Reality System with 3D Animation, Spoken Interaction, and Runtime Modifiability, *Presence, MIT Press, Vol. 6, No. 4, pp. 461-480.*

[Weat98] Weatherly, R. (1998) DoD High Level Architecture for Modeling and Simulation, *Software Engineering and Economics Conference.*

[Yu00] Yu, S. J. (2000). Dynamic Update Message Filtering Schemes for Distributed Virtual Environments. *Ph.D. Thesis, Computer Science Department, Yonsei University, South Korea*.

[ZBDM97] Zyda, M., Brutzman, D., Darken, R., McGhee, R., Falby, J., Bachmann, E., Watsen, K., Kavanagh, B. & Storms, R. (1997) NPSNET - Large-Scale Virtual Environment Technology Testbed, *Proceedings of the International Conference on Artificial Reality and Tele-Existence, Tokyo, Japan, pp.18-26.*

[ZhGe01]	Zhao, H., & Georganas, N. D. (2001) Collaborative Virtual Environments: Managing the Shared Spaces, *Networking and Information Systems Journal , Vol. 3, No.2*.
[Zyda96]	Zyda, M. (1996) Networking Large-Scale Virtual Environments, *Proceedings of Computer Animation '96, 3-4 June 1996, Geneva, Switzerland, IEEE Computer Society Press, pp. 1-4.*
[Beie01]	Beier, K.-P. (2001) Virtual Reality: A Short Introduction http://www-vrl.umich.edu/intro/
[COVEN]	COVEN – http://chinon.thomson-csf.fr/projects/coven/
[DirectX]	Microsoft DirectX – http://www.microsoft.com/directx/
[HLA]	HLA Web-Site – http://hla.dmso.mil
[LW]	Living Worlds draft specification http://www.vrml.org/WorkingGroups/living-worlds/draft_2/index.htm
[NPSNET-V]	NPSNET-V Research Group – http://movesinstitute.org/~npsnet/v/
[OC]	High Level Overview of Open Community – http://www.opencommunity.com/ov.html
[OC-C]	Open Community ANSI-C API Specification http://www.opencommunity.com/opencom-c-api.htm
[OC-SG97]	Open Community Exhibit at Siggraph '97 http://www.opencommunity.com/sig.html

[OpenGL] OpenGL Architecture Review Board (http://www.opengl.org).

[RTP] http://www.ietf.org/rfc/rfc1889.txt

[VJ3D] VRML-Java3D Working Group

http://www.web3d.org/WorkingGroups/vrml-java3d/

[VRML] VRML Web Site – http://www.vrml.org