Jauvane C. de Oliveira
jauvane@acm.org
University of Ottawa
School of Information Technology and Engineering
DIStributed & COllaborative Virtual Environments Research (DISCOVER) Laboratory
800 King Edward Ave. Room 5-077
Ottawa, ON K1N 6N5
Canada
Military Institute of Engineering
Department of Systems Engineering
Pça. General Tibúrcio, 80
Rio de Janeiro, RJ 22290-270
Brazil
National Laboratory for Scientific Computation
Computer Science Department
ComCiDis Research Group
Av. Getúlio Vargas, 333
Petrópolis, RJ 25651-075
Brazil

Nicolas D. Georganas
georgana@discover.uottawa.ca
University of Ottawa
School of Information Technology and Engineering
DIStributed & COllaborative Virtual Environments Research (DISCOVER) Laboratory
800 King Edward Ave. Room 5-077
Ottawa, ON K1N 6N5
Canada

VELVET: An Adaptive Hybrid Architecture for VEry Large Virtual EnvironmenTs

Abstract

Collaborative virtual environment (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. The architectures available today provide support for a number of users, but they fail if too many users are together in a small "space" in the virtual world. This paper introduces VELVET, an adaptive hybrid architecture that allows a greater number of users to interact through a CVE. This is accomplished through an adaptive filtering scheme based on multicasting. VELVET also supports small groups of users, but its use in large environments shows the greatest potential, better handling local concentrations of activity than region-, cell-, or *locale*-based approaches. VELVET introduces a novel adaptive area of interest management that supports heterogeneity amongst the various participants. This allows users in a supercomputer with high-speed networking to successfully collaborate with others in not-so-powerful systems behind a slow dial-up connection.

1 Introduction

Over the past few years, a number of interactive virtual reality (VR) systems have been developed. A collaborative virtual environment (CVE) is a special case of a VR system wherein the emphasis is more on collaboration among users rather than on simulation. CVEs are used for applications such as collaborative design (Daily et al., 2000; Hindmarsh, Fraser, Heath, Benford, & Greenhalgh, 2000; Wang, Wong, Shen, & Lang, 2002), training (Oliveira, Shen, & Georganas, 2000; Oliveira, Hosseini, et al., 2000; Kirner et al., 2001), software engineering (Fernando, Murray, Tan, & Wilmalartne, 1999; Doppke, Heimbigner, & Wolf, 1998), and telepresence, and many more applications are showing up daily.

Many of the applications may potentially have a very large number of users at a time, and that can easily overload a fast network as well as impose huge processing requirements at the stations. As computing resources are limited, obvious problems arise once the number of users in a simulation increases 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 and loss of interactivity. Another problem that a CVE faces is that of heterogeneity of hardware available for users. This also imposes some limitations to a CVE as users in very powerful systems and fast networks would need to collab-

Presence, *Vol. 12, No. 6, December 2003,* 555–580

orate with others with very limited hardware and slower networking. It is obvious that the second type of system should not be required to deal with the same load as the first. The easiest way to control such problems is by deploying the simulation based on the slowest and weakest system or setting up a minimum requirement and simply denying access to nonconforming systems. The first approach, although guaranteeing functionality and availability, would greatly underutilize better systems; yet worse, more-limited systems may join the session at any time. The second approach denies access to a potentially large number of users and may still lead to underutilization of some systems.

We have designed and implemented a number of CVEs for industrial training and electronic commerce (Oliveira, Shen, et al., 2000; Oliveira, Hosseini, et al., 2000; Oliveira, Shirmohammadi, & Georganas, 1999). Such CVEs, although allowing rich collaboration, did not provide the 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, a situation that does not scale well.

In this paper, we present VELVET, an adaptive hybrid architecture for VEry Large Virtual EnvironmenTs. VELVET addresses the aforementioned issues, allowing a very large number of users to participate in a CVE while allowing users with heterogeneous hardware and available networking to collaborate in a best-effort approach. Such adaptive filtering is performed through multicasting.

In section 2, we introduce related work. Section 3 introduces VELVET, its design, and its functionality. Section 4 presents simulation results, and section 5 a performance comparison with other architectures, followed by a conclusion.

2 Related Work

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 introduce several of them, namely SIMNET, DIS, NPSNET-IV, SPLINE, MASSIVE-2, SCORE, and the architecture proposed by Abrams (Abrams, Watsen, & Zyda, 1998; Abrams, 1999), which somewhat represent the other models as well.

2.1 DIS and SIMNET

DIS (distributed interactive simulation) is a standard (IEEE, 1993) that 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 U.S. Army by Bolt Beranek and Newman, Perceptronics, and Delta Graphics (Macedonia, 1995). SIMNET has been developed to take full advantage of ethernet hardware when broadcasting is heavily used, reducing software selection of packets; however, this also brings undesirable dependency on ethernet facilities that are fit for only a LAN. SIMNET has no central object repository; that is, each host is responsible for maintaining its own copy of the objects participating in the simulation. Stations participating in a simulation exchange only state messages with the others, and dead reckoning is used to reduce communication requirements, which will be discussed shortly.

DIS (IEEE 1278) has been developed in an attempt to overcome SIMNET limitations. It is a group of standards developed by the U.S. Department of Defense and industry. DIS uses similar protocol data units (PDUs) 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 (Macedonia, 1995), 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, but instead the ghost tries to predict the motion of the player. The player also calculates such predictions using the same algorithm. When there is an error greater than a predefined threshold, the player sends an update that 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 because all objects are mirrored in every station. For example, in a 1,000-participant environment, each station would have to process at least 1,000 ghost objects (the player's ghost has to be processed locally as well) and a player.

DIS requires all objects to send periodic update messages even for the objects that do not move or are "dead."

2.2 NPSNET-IV

NPSNET-IV (Macedonia, Zyda, Pratt, Barham, & Zeswitz, 1994) is a prototype developed in 1995 at the Department of Computer Science at the Naval Postgraduate School in Monterey, California. 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 (Macedonia, 1995; Macedonia et al., 1994). NPSNET-IV also implements the player/ghost paradigm detailed in subsection 2.1 and, as with DIS, NPSNET's communication PDUs include military-oriented packets, such as Fire PDU, Detonate PDU, and the like, which makes it somewhat unfit for civilian applications.

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 (seven) need to be dealt with. Each hexagon has its own multicast group, and data are sent through the multicast address of the hexagon where the source of such data flow is located. Figure 1 shows the hexagonal partitioning of the world with an avatar moving from one area to the neighborhood. 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.

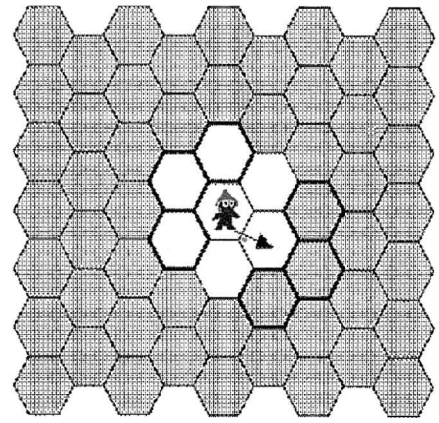

Figure 1. *Moving through hexagons in NPSNET-IV.*

2.3 Open Community and SPLINE

Open Community (OC) (Barrus, Waters, & Anderson, 1996) 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 OC-compliant implementation that provides development APIs. Such libraries provide very detailed and essential services for real-time multiuser cooperative applications. For its communication, SPLINE uses the interactive sharing transfer protocol (ISTP) (Waters, Anderson, & Schwenke, 1997), which is a hybrid protocol supporting many modes of transport for VR data and information through five subprotocols, namely 1-1 connection; object state transmission, streaming audio, *locale*-based communications, and content-based communication subprotocols.

SPLINE partitions the world model in *locales,* which may have any shape. In this sense, NPSNET's partition could be considered as a particular case of *locales* where the division follows the rule of hexagonal partitions. Once a user joins a given *locale*, everything that is located in that *locale*, as well as the *locales* within its immediate neighborhood, would be visible. It is possible to be "present" in more than one *locale* by using spObserver objects. Internally, a spObserver object makes the local OC engine listen to the multicast group of that *locale* (and its neighborhood).

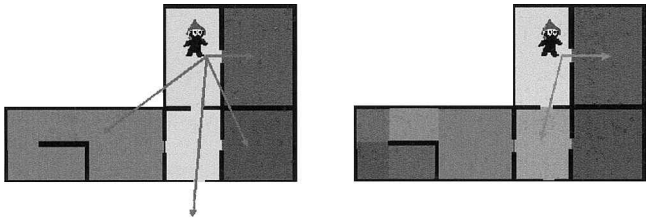

Figure 2. *Partitioning a world model in* locales.

SPLINE will behave differently depending on how the world model is partitioned in *locales*. Figure 2 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.

2.4 MASSIVE-2

MASSIVE-2 (model, architecture and system for spatial interaction in virtual environments) (Greenhalgh, 1997; Greenhalgh & Benford, 1999) is a prototype developed in 1997 at the Computer Science Department at the University of Nottingham.

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 (Greenhalgh, 1996). The idea behind third-party objects is to allow a group of artifacts (called *crowd*) to be represented as a unique object that is seen by others. 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 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.

2.5 SCORE

SCORE (Lety, Turletti, & Bacelli, 1999; Lety, 2000) was developed in 2000 at INRIA, Sophia Antipolis, France. It is based on the division of the world in cells as suggested by Hook, Rak, and Calvin (1994). A user interacts with those cells, which fall, at least partially, within an "area of interest," which is defined as a square region around a user's avatar. Each cell has its own multicast group (MG), and an avatar then subscribes to that set of MGs. SCORE allows for two policies regarding determination of cell size: precalculation 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 predefined parameters, such as number of MGs available, density of participants, and so on. 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 3.

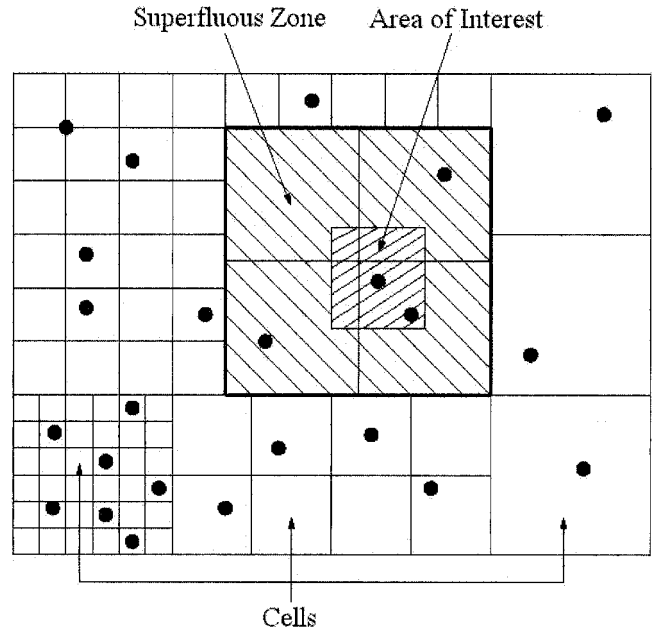

Figure 3. *Cells in SCORE.*

2.6 Howard Abrams' Architecture

Abrams has proposed (Abrams et al., 1998; Abrams, 1999) a three-tiered management scheme for LSVE. In such architecture, the first tier is quite similar

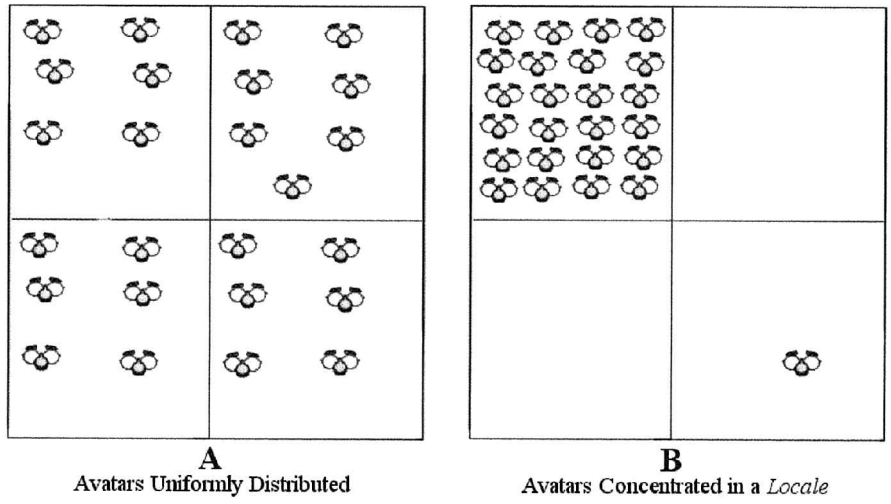

Figure 4. *Space-based solutions and their limitations.*

to that used in SCORE, in the sense that the space is partitioned into regions that can be split into smaller regions in the event of concentration of users in a single region. The regions are organized in an octree so that, when a region is split into smaller pieces, some new leaves are added to the tree. The architecture allows for a user to limit the number of leaf nodes processed based on a "minimum region size." The idea is that, if a given region, once split into smaller pieces, is smaller than a minimum region size threshold, the user's system need not subscribe to the smaller regions, sticking to the larger region, which is located up from the leaves in the tree. The second tier is a per-entity, protocol-independent filtering mechanism, in which case one given user may select a subset of those objects that have passed through the first tier to further discard uninteresting content. The third tier is protocol dependent, which allows filtering of portions of the data broadcasted (multicasted in fact) by a given object (that is, just those protocols that are of interest).

3 The Proposed VELVET Architecture

VELVET is our adaptive hybrid architecture. The approaches previously described, although addressing the issues of a large-scale virtual environment (LSVE), all fail under certain circumstances. In subsection 3.1, we will examine such failure potentials. In subsection 3.2, we will present the VELVET approach in a high-level overview, whereas subsection 3.3 will provide details on how the functionality described in the previous subsection actually works. We will show that VELVET clearly performs better than other approaches, under the circumstances discussed in subsection 3.1.

3.1 Limitation of Existing Models

We may assume that SPLINE represents other models such as NPSNET-IV and others based on geographical partitioning of the virtual world (Singhal & Zyda, 1999; Frécon & Stenius, 1998; Hagsand, 1996; Diot & Gautier, 1999). In both cases, the virtual world is partitioned, and each user is supposed to receive data from objects that are located in a well-defined subset of the partition (or *locales* for SPLINE).

Locale-based models assume that users are somewhat uniformly dispersed in the virtual world (Figure 4a). That is, the idea of reducing the amount of data that each station must deal with is addressed by reducing the area that is "seen" by each participant. This 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 4b). Suppose we have a virtual museum in which some 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 thousands of data flows, as all of them would be in the same *locale* (or in the neighborhood). The *locale*-based approach would hence fail in the task of reducing the amount of data that 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 existing architectures is that of heterogeneity. Let us consider a group of three hundred hosts participating in a CVE. Yet, let us assume that 290 of these 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 that could support such a load gracefully. The remaining ten stations, however, could be 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 neighborhood. 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 performance 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 because the first prevents some users from joining the CVE session, and the second would under-use resources.

With regard to the architecture proposed by Abrams (Abrams et al., 1998; Abrams, 1999), we will discuss it in section 5, once the reader gets to know our proposed VELVET architecture.

3.2 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 that is located within the world
- *avatar:* special kind of object that represents a user
- *bot:* active object that is not an avatar
- *artifact:* an object that is neither avatar nor bot
- *area of interest* (AoI): the radius a user is able to view and directly interact with.
- *check in:* operation that brings an object into a user's AoI
- *check out:* operation that removes an object from a user's AoI

VELVET is a CVE architecture that 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 a feature allows systems with different processing power and networking facilities to still collaborate efficiently because each one may elect to see just as much as possible (or as much as "paid" for).

In subsection 3.2.1, we discuss the AoI in general terms. That is followed by the double layered boundaries, the parallel virtual world (PVW), and finally the support of heterogeneity in VELVET. Using the knowledge of the PVW, we will revisit the AoI management in VELVET, as that can be better defined in terms of the PVW.

3.2.1 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 behavior 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 treatable range. Figure 5 shows a group of avatars in a room with one avatar, indicated by the arrow

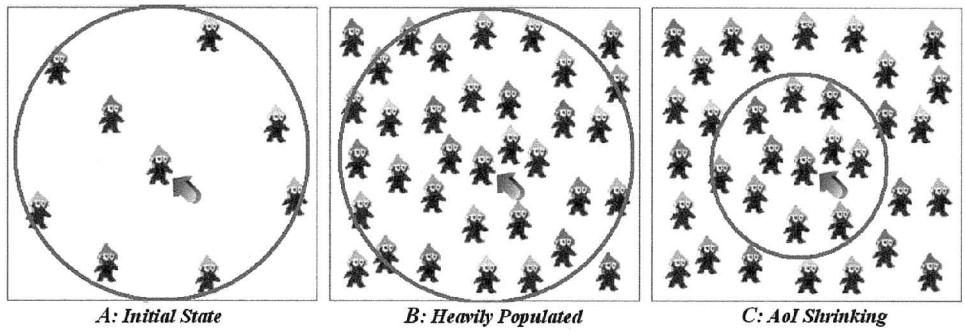

Figure 5. *VELVET's area of interest shrinking*

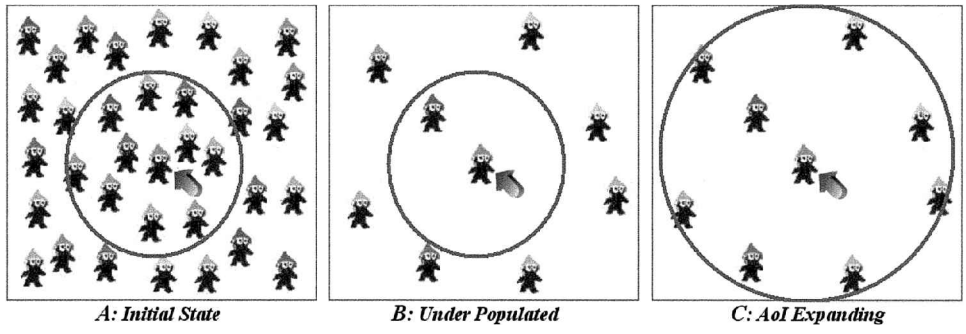

Figure 6. *VELVET's area of interest expanding*

(A). One can then see a larger number of avatars (B), which makes the AoI shrink (C) as needed. Only the objects that are within the AoI are visible, and only information from those objects is received.

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 6. If we let B be the average bandwidth 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 in, we can express incoming bandwidth for space-based solutions as $B \times P_A$ and for VELVET as $B \times P_I$, where $0 \leq P_I \leq P_A$. A more formal description of AoI management is shown in subsection 3.2.5.

3.2.2 Double Layered Boundaries of VELVET's AoI. Additionally, when an object crosses the border of the AoI, it will check in (CI) or check out (CO), depending on the direction being towards the AoI or leaving the AoI, respectively. To avoid multiple CI/CO operations, VELVET in fact defines two borders named *area of 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 7 (top) shows an example of use of AICI/AICO. Figure 7a, b, and c shows an AoI with a single border (or AICI and AICO with distance zero), and two different distances between AICI and AICO, respectively. One can notice the arrows displaying thirteen, seven, and five CI/CO operations for the same path.

If we define the following parameters,

- p_i = the radius of inner boundary of AoI_i;

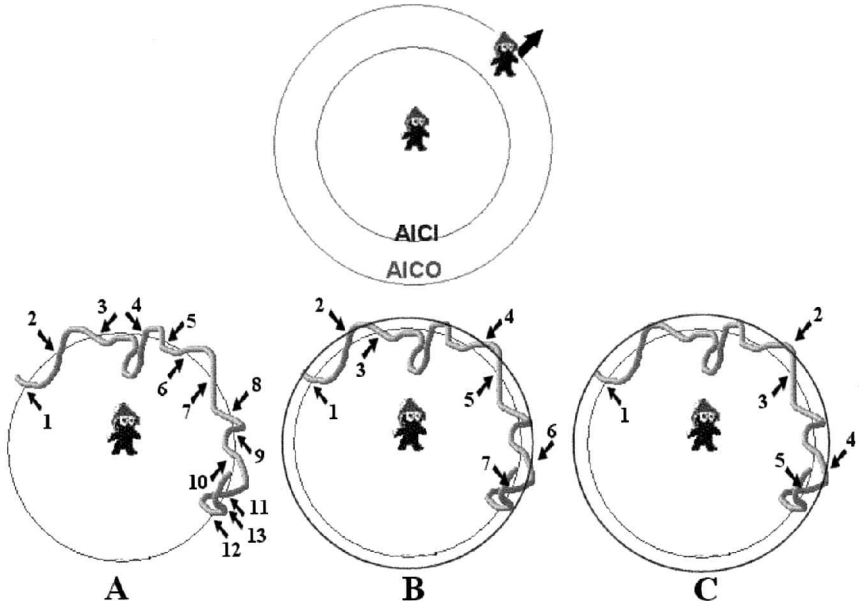

Figure 7. *VELVET's AICI/AICO: (a) single layer (13); (b) double layer (7); (c) wider double layer (5).*

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

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

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

where *timeout* is the minimum time that shall elapse before changes in the AICI/AICO, which prevents too many changes from happening in a short time interval. Such a time interval can be set, for instance, at 1 sec. Please note that this procedure 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 (distance) from the avatar but rather based on a predefined 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 Figures 5 and 6 somewhat indicate. We can assume that the positioning of the avatars in Figures 5 and 6 are based on such a metric rather than on the virtual distance to the avatar. Of course, the metric itself could be that of virtual distance.

3.2.3 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 8 shows the PVW.

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 8 will be subscribed to. Note that, as each avatar has

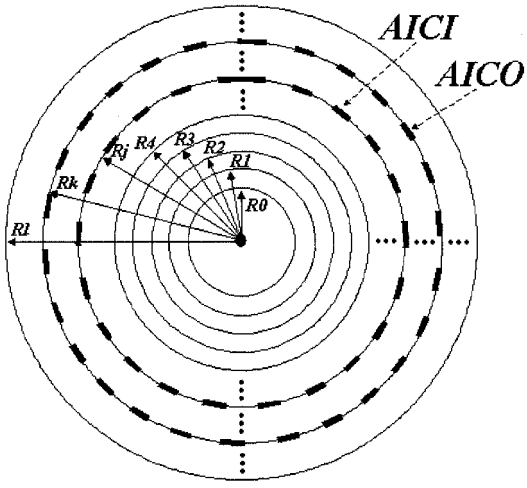

Figure 8. *VELVET's parallel virtual world*

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, such as number of users
M_1 = Metric 2, such as network bandwidth
M_2 = Metric 3, such as distance in the virtual world
M_3 = Metric 4, such as distance in the network (number of hops)
M_4 = Metric 5, such as distance in number of *locale* hops
M_5 = Metric 6, such as average end-to-end delay
M_6 = Metric 7, such as a mix of the above, such as $M_4 * 10000 + M_1$
⋮
M_m = metric $m + 1$, such as 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 $\sum_{n=0}^{j} \xi(R_n) \leq T \leq \sum_{n=0}^{k} \xi(R_n)$, where $\xi(R_n)$ is a function that gives the cost associated with the level R_n, such as the number of participants in level n for a metric considering the number of users. T is a value that will be optimized according to a predefined target value for a given metric. For instance, if one is behind a 56K modem connection and the metric considered is total bandwidth, the target could be something like 48Kbps. T would be maximized considering that it must remain below this target. See subsection 3.3.3 for further details on the implementation of PVW in VELVET.

3.2.4 Degree of Blindness and Support to Heterogeneous Systems in VELVET. Because participants unilaterally decide which objects to subscribe to, based on the PVW, one can notice that such behavior can lead to inconsistencies. Such an inconsistency shows up when a given avatar A "sees" an avatar B even though B cannot "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. In VELVET terminology, this is called *degree of blindness,* which limits one's vision. This is perfectly legal in VELVET. In fact, it is the very reason why VELVET graciously supports collaboration among users in heterogeneous systems in a best-effort approach. Figure 9 illustrates this behavior, which defines degree of blindness in VELVET. The smaller the AoI, the greater the degree of blindness. Figure 9 shows users A and B adopting similar metrics; the area around each avatar is that based on the PVW rather than the Euclidean space.

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 10. We can define that $A_i \subset A_j \mathbf{N}\ \partial_{ji} \leq \rho_j$ that is, if A_i is within A_j's area of interest. Similarly, we can define that $A_i \supset A_j \mathbf{N}\ \partial_{ij} \leq \rho_i$. In other words $A_i \supset A_j \mathbf{N}\ A_j$ is within A_i's AoI, that is, 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, $A_i \supset A_j \mathbf{N}\ A_j \supset A_i$ 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.

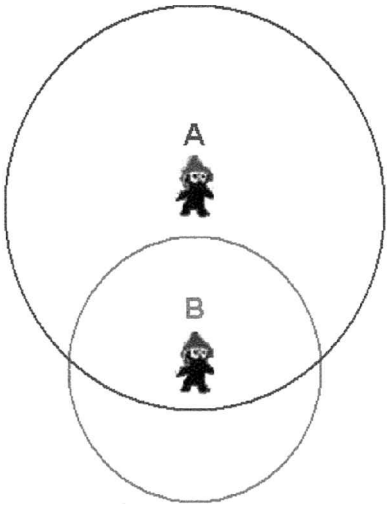

Figure 9. *VELVET and degree of blindness*

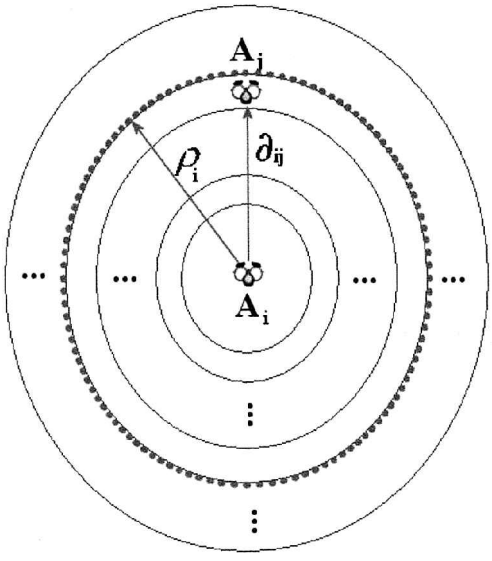

Figure 10. *Parameters ∂ and ρ*

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. This allows heterogeneous systems to collaborate in a best-effort approach that is, 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).

Table 1. *VELVET's $\subset$ and $\supset$ Relations for Objects with Similar Metrics*

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

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.

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 shows several relations supported by VELVET, with the box showing the scope of space-based solutions. This table assumes that both A_j and A_i use the same metric M_k.

3.2.5 Area of Interest Management Revisited.

In this subsection, we detail how the AoI management works, now in terms of the PVW. Subsection 3.2.1 brought a high-level overview of AoI, which could shrink and expand. We 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 to 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 subsection 3.2.1. Figure 8 shows a generic PVW with levels R0 to RL, with RK and RJ the levels of AICO and AICI, respectively, 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 $\partial_{ij} > \rho_K$, A_j will check out (CO) from the PVW$_i$, in the next cycle of

Table 2. *Scope of VELVET and Space-Based Solutions Regarding ⊂ and ⊃ Relations*

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

the AoI management thread (if it was checked in previously). This happens because A_j is by definition outside the AoI of A_i. (See subsection 3.2.3.) On the other hand, if $\partial_{ij} < \rho_j$ at a given time, A_j would check in (CI) into the PVW_i, in the next AoI management cycle (assuming that it was not checked in already).

Additionally, $\forall\alpha$; $J \leq \alpha \leq K$, if $A_\alpha \subset A_i$ at a given time $t + 1$ N $A_\alpha \subset A_i$ at time t. Similarly, $\forall\alpha$; $J \leq \alpha \leq K$, if $A_\alpha \not\subset A_i$ at a given time $t + 1$ N $A_\alpha \not\subset A_i$ at time t; that is, 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 ($J = J - 1$), so that ∂_{JK} will increase and an object will have to move farther towards A_i, in the PVW_i, before it checks in to the AoI of A_i. ∂_{JK} is defined as the distance between the AICI at level J and the AICO at level K.
- If the CO count is high, the AICO will move farther away from A_i ($K = K + 1$), so that ∂_{JK} will increase and an object will have to move farther 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, or any other bottleneck), both levels J and K (namely AICI and AICO) will be reduced, ($K' = K - 1$ and $J' = J - 1$). (See Figure 11a). 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, this is accomplished by increasing both AICI and AICO levels (Figure 11b), leading to CI operations for those objects that were not within the AoI before the expansion and that 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 J' is the new shrunk level of the AICI.

It is important to note that the distances based on the function ∂ are distances in the PVW of a given avatar, and not distances in the Euclidean world. Remember that a PVW is built based on a chosen metric, as defined in subsection 3.2.3.

3.3 Internal Structures and Functionality of VELVET

In VELVET, the world is partitioned into areas, and each area has a multicast address as in SPLINE and NPSNET-IV. VELVET accomplishes the flexible functionality described in the previous subsection by assigning a multicast group for each object that generates a flow of data (such as avatars and bots). This defines an object transmission channel (OTC). Each client has three threads running in parallel: the first thread is responsible for sending data through the network, the second receives and acts upon arriving packets, and the third thread performs management of AoI, joining and leaving areas and OTCs as appropriate. Figure 12 shows the way the receiver thread works. Each *locale* similarly has its own *locale* transmission channel (LTC).

3.3.1. Awareness Control. In VELVET, a given avatar A_i may enter or join *locales* $L_1, L_2, \ldots, L_n$.

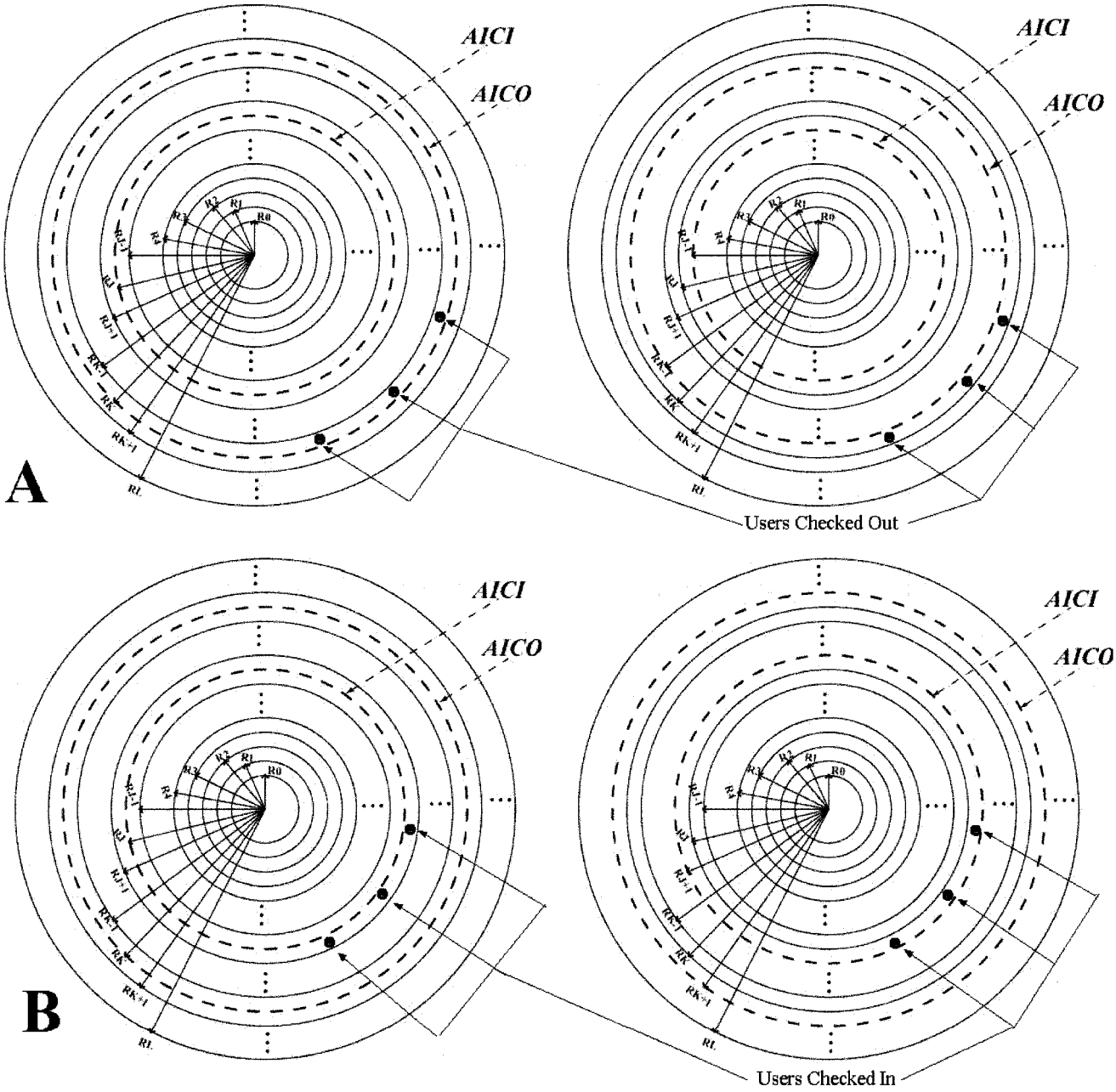

Figure 11. *AoI shrinking (a) and expanding (b) based on the PVW.*

A_i enters a *locale* if it physically moves 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 appropriate message to the communication channel of such *locale*. Every other active object that is located within this *locale* sends an INFORMLOCALE message back to A_i, which then learns about such objects. A_i can then consider such objects in the next cycle of the AoI management thread. On the other hand, the other avatars receiving an

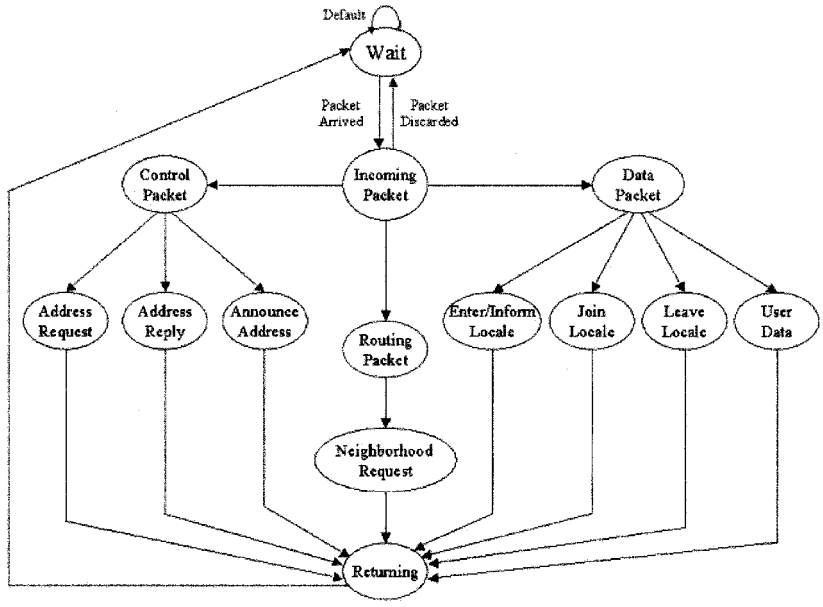

Figure 12. *Receiver's procedure.*

ENTERLOCALE message from A_i will become aware of it and will consider it in the next cycle of the local AoI management thread as well. An avatar sends a LEAVELOCALE message once it is leaving a *locale*. Such a message allows the others to remove it from the list of participants of such *locale*.

Figure 13 depicts the procedure by which an avatar A_1 becomes aware of other active objects. In the figure, the shaded area represents the areas that the local user is aware of. Initially, A_1 enters a *locale*, such as L1 (Figure 13a). At this stage, A_1 sends an ENTERLOCALE message in L1's communication channel. The message is received by active objects that are located in L1, and those then send an INFORMLOCALE message back to A_1, which then learns about such active objects. Similarly, A_2 has learned that A_1 has just entered L1. Let us suppose that the metric M_1 that A_1 is following is to subscribe to the communication channels of no more than 10 avatars. At stage B in Figure 13, only one avatar is known to A_1 (that being A_2). As A_1's metric target is larger than 1, it joins the next level (the first in this case) of neighboring *locales*, namely L3 in Figure 13b/c. A_1 then sends JOINLOCALE messages to L3 and receives INFORMLOCALE replies from A_3 and A_4. Only three objects—A_2, A_3, and A_4—are known, which is still under A_1's metric of M_1. A_1 goes on and subscribes to the next level of neighboring *locales*, namely L4 and L5 (Figure 13c/d). This procedure goes on one step further until when the metric M_1 has been met or a sufficient large number of *locales* has been subscribed to. At stage E in Figure 13, A_1 has awareness of 11 active users, which are then filtered through the AoI management thread, so that the parameter T described in subsection 3.2.3 is kept below the target selected (10 users in the preceding example).

In 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.

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

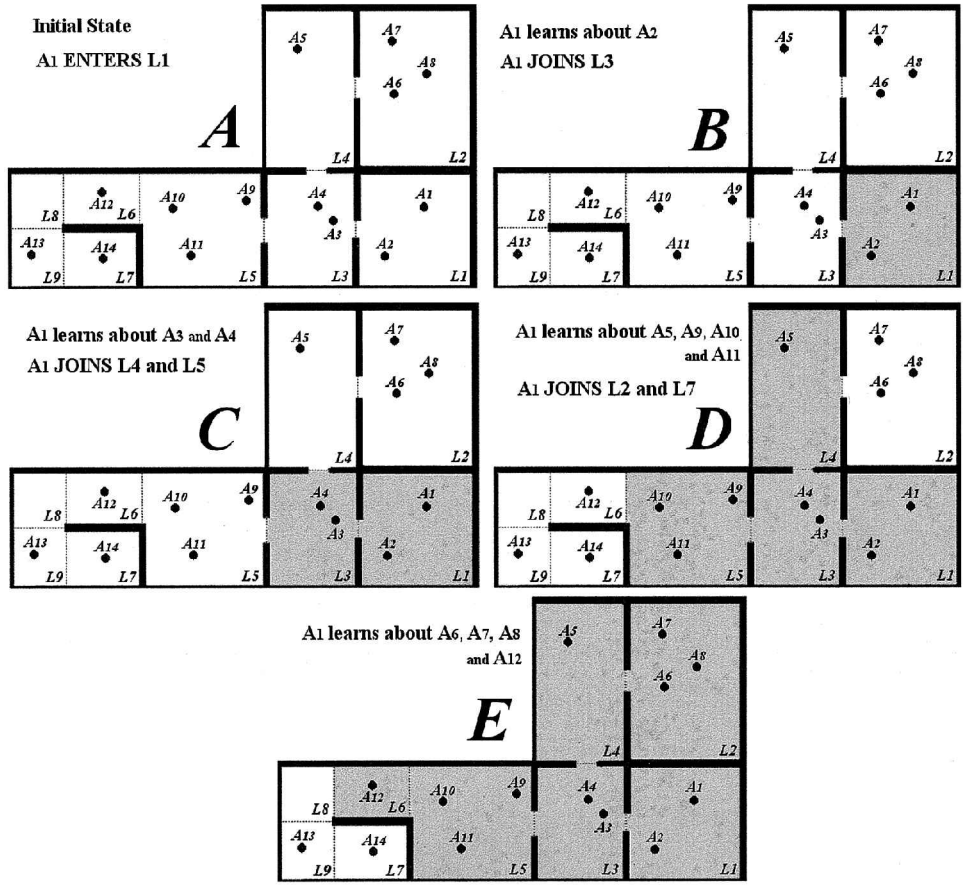

Figure 13. *Awareness control in VELVET.*

Table 3. *Superfluous Data*

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

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 T parameter), which immediately reduces the flow of data arriving at that end. Additionally, VELVET allows a heterogeneous set of hosts to successfully collaborate

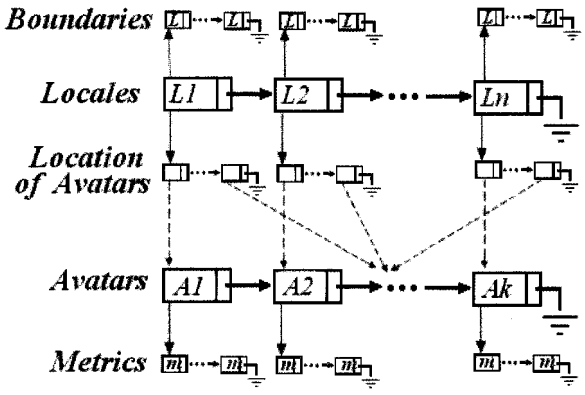

Figure 14. *Internal data structures.*

because each user can have his/her AoI reduced as much as necessary to make it treatable while increasing its degree of blindness, which would allow people in a supercomputer as well as very limited systems to collaborate in a CVE session.

3.3.3. Data Structures. Figure 14 shows how data are organized in a VELVET client. Everything starts with a linked list of *locales*, each of which contains a list of boundaries—that is, pointers to other *locales* that are neighboring each *locale*—as well as a list of active objects (such as avatars), which are located within that *locale*. There is also a list of known avatars, (those that the local user is aware of through the procedure described previously (Figure 13)). Each known avatar has a list of metrics that are initialized with values provided by each active object and updated with corrected statistical values (such as average delays, jitter, and so forth).

The list of avatars shown in Figure 14 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.

3.3.4. Parallel Virtual World and Filtering of Messages. Figure 15 shows how the PVW actually works in light of the structures shown in Figure 14. (The list shown in Figure 15 is just the avatars list from Figure 14.) The idea is that such a list is ordered based on the metric chosen. The levels $R_0, R_1, \ldots, R_l$, are defined based on the metric values known at the moment. For instance, if we consider that the metric chosen is the bandwidth required by a given avatar, the levels in the PVW could be set as 1 Kbps, 2 Kbps, . . . , α Kbps, in which α would represent the maximum transmission rate known at the moment. The active objects that 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 subsection 3.2.1.

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, JOINLOCALE, and LEAVELOCALE (that is, packets that are related with awareness management). The VELVET session management entity will frequently analyze the structures shown in Figures 14 and 15 based on their profile and will choose which ones to sign up for or drop.

4 Modeling and Simulation

VELVET has been modeled using the OPNET Modeler 6.0 PL12 (Mil3, Inc.). In this modeling, we created a multicast-enabled router that allowed VELVET to perform exactly as described in subsection 3.3. Figure 16a shows the node model for a VELVET station. The three main threads of the VELVET architecture (transmitter, receiver, and management) are represented as three independent nodes. Figure 16b shows the process model for the management node. This model consists of a startup procedure, required due to the DHCP server mentioned previously, followed by an infinite loop in which the thread wakes up every once in a while (TIMEOUT) to analyze the current content of the structures presented in Figure 14 and perform changes that may be required. Such changes include

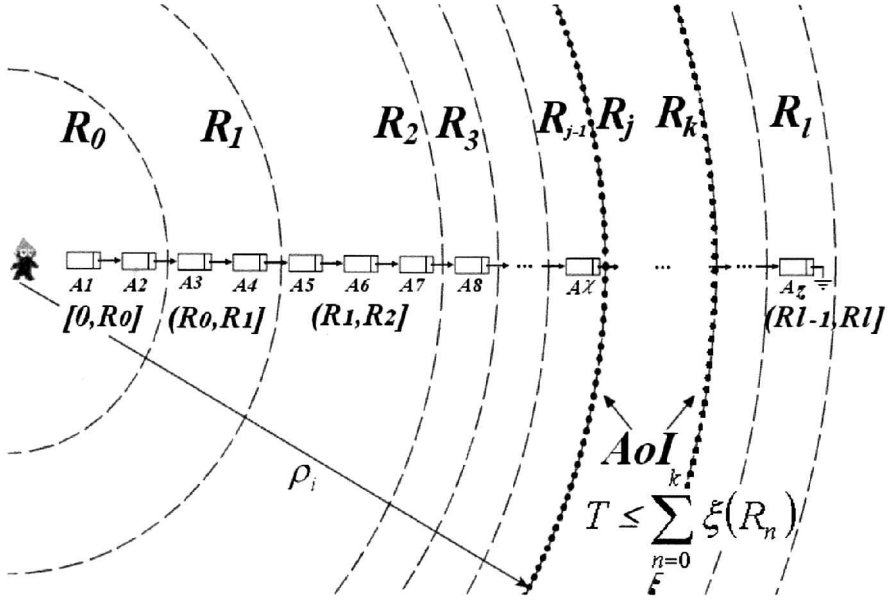

Figure 15. *PVW in the scope of internal structures of VELVET.*

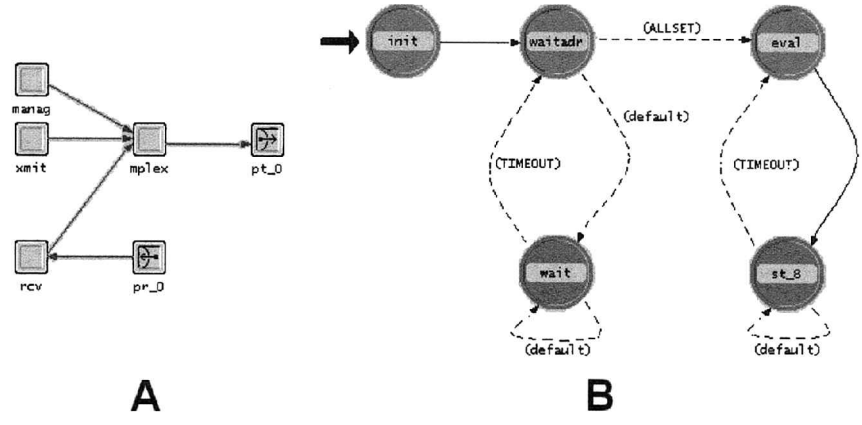

Figure 16. *VELVET's host node/process models.*

subscribing/dropping OTCs, joining/leaving *locales*, and so forth.

The modeling of VELVET uses external configuration files that allow one to select the statistical behavior of each station along with the geography of the VELVET world. For most simulations, we used the world shown in Figure 17 in which nine *locales* exist with well-defined neighborhoods. Several stations were placed and different profiles were chosen.

To make the setup of the simulation even more convenient, we introduced a DHCP server in the simulation. Each station requests this server to assign an IP address for it at startup. This allows easy reconfiguration because 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 ex-

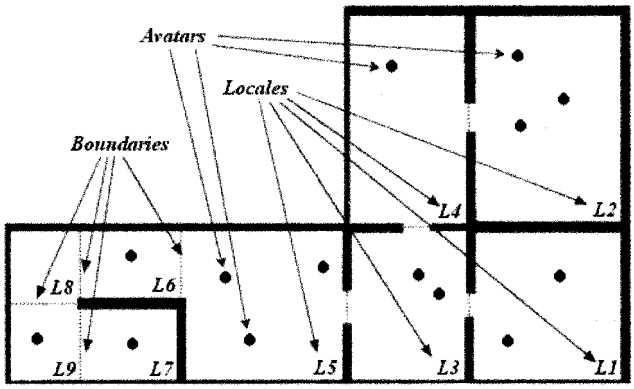

Figure 17. *Virtual world.*

changes after this time interval are those of VELVET management along with data packets sent by the several stations.

Figure 18 shows the average number of packets received by four stations selected in the network. Those are four stations with profiles to view one, five, nine, and twelve 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, the AoI manager is led to drop four 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 19 shows the results from thirteen stations. One can see twelve somewhat well-defined data flow levels, which reflect the fact that the hosts were set to "see" from one to twelve stations. For this simulation, all stations were sending packets according to an exponential distribution with a mean outcome of 0.125 sec. (average of eight packets per second). The probability of an avatar changing a *locale* was set to 15%.

The graphs of Figures 18 and 19 show that VELVET supports heterogeneity well because each station may select a different level of service. This can be extended for quality-of-service purposes wherein 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 18.

The simulation has also been executed in a two-area world (Figure 20). In such a world, SPLINE would lead to rendering all users populating the world because all objects in the current *locale*, as well as the immediate neighbor *locales*, are subscribed for. A given user would, hence, receive packets from every active object in this specific world. Figure 21 shows the performance of such a simulation.

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 21. In our two-*locale* world, SPLINE would receive packets from absolutely all objects within the world. VELVET allows for filtering even within a single *locale* because 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 $\partial_{JK} = 0$ f $\partial_{i\alpha} \leq \rho_J, \forall\alpha$, 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 21.

It is worth mentioning that the high traffic experienced at the first 500 ms is initially due to the use of a DHCP server, which had been introduced into the simulation 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 as that of subscribing to all objects in the avatar's *locale* as well as the immediate neighbors, VELVET would behave just like SPLINE; hence, *locale*-based architectures can be thought of as a subset of VELVET.

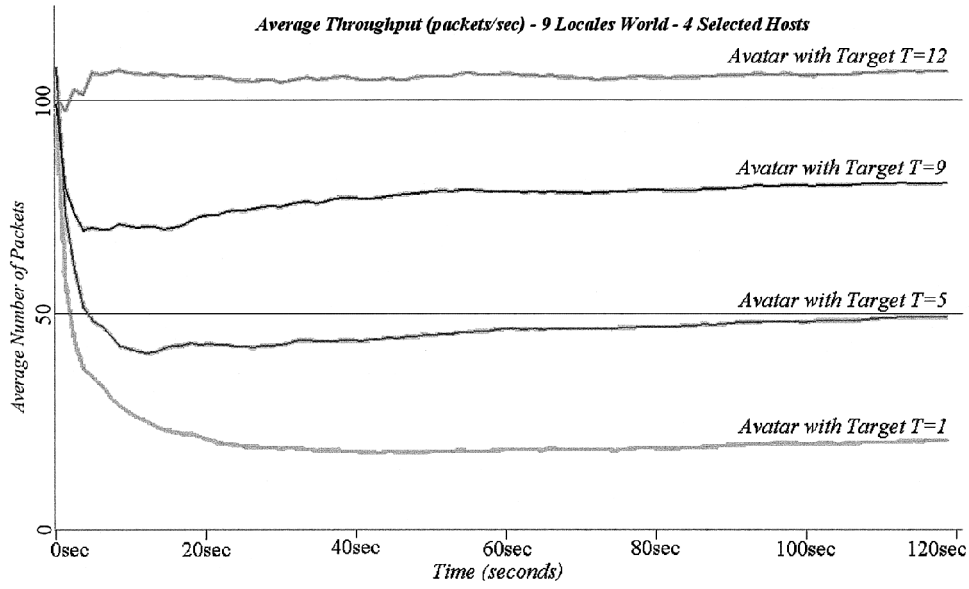

Figure 18. *Incoming traffic: average throughput in packets/second.*

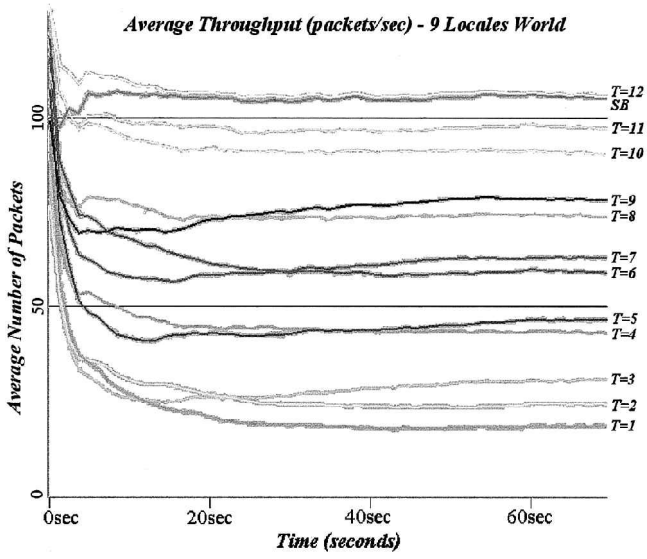

Figure 19. *Incoming traffic: average throughput in packets/second.*

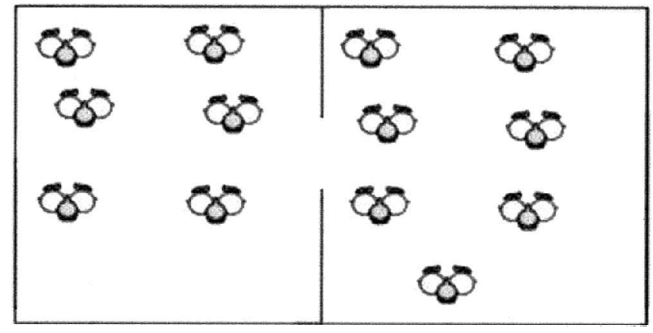

Figure 20. *A two-*locale *world.*

Figure 22 shows the 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) to approximately 1,600 incoming packets per second (when 209 users are within the virtual world), shows how space-based solutions (such as SPLINE) behave when the user population grows. VELVET, on the other hand, stays stable between 70 and 100 incoming packets per second.

With 34 users, VELVET stations had incoming traffic between 40 and 70 packets per second, whereas space-based solutions (SBS) had more than 275 incoming packets per second (Figure 22a). When 59 users were in the world, the results stayed almost identical for VELVET at 45 to 75 packets per second, but SBS achieved more than 475 packets per second (Figure 22b). SBS

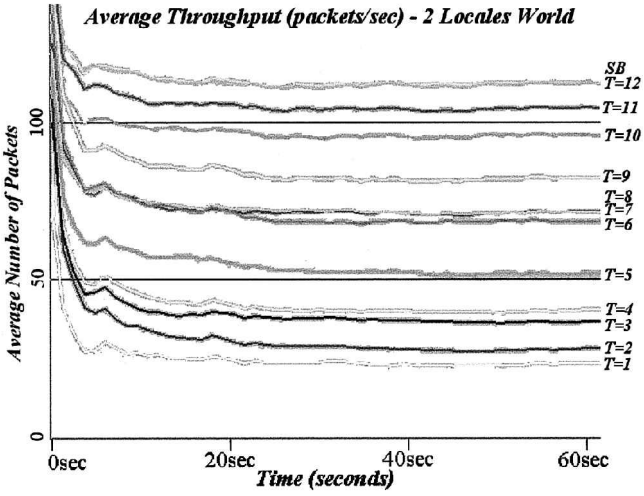

Figure 21. *Incoming traffic: average throughput in packets/second.*

went further to approximately 690 packets per second with 84 users (Figure 22c), 890 packets per second with approximately 109 users (Figure 22d), and so on, achieving more than 1,600 packets per second with 209 users (Figure 22h). In contrast, VELVET stations were stable at around 100 packets per second (Figure 22a–h).

Figure 23 shows a better comparison of VELVET versus SBS, using data gathered from simulation results shown in Figure 22. In this graph, we selected the VELVET station with the highest incoming throughput (that is, that with the AoI more expanded among the various VELVET stations in Figure 22). We can see that this VELVET station keeps a relatively stable average incoming packet count while SBS grow steadily.

These simulation results show how VELVET behaves when compared with *locale*-based architectures. Such results are somewhat obvious because VELVET, being a superset, can behave exactly like SPLINE as well as allow a more aggressive filtering of incoming data and hence reducing incoming bandwidth. It is important to mention that a VELVET station could have its AoI expanded enough so that it could achieve the same packet counts shown previously for SBS. For example, assume user A_i has a metric that allows subscription to up to 250 other users. One advantage of VELVET is exactly this flexibility, allowing users to unilaterally choose a metric that is appropriate to their own needs.

5 Comparison With Other Architectures

In this section, we are comparing VELVET with other architectures, namely MASSIVE-2, SCORE, and the architecture by Abrams (Abrams et al., 1998; Abrams, 1999).

5.1 MASSIVE-2

Greenhalgh presents studies (Greenhalgh, 1997) of total bandwidth of MASSIVE-2 as a means of comparison with MASSIVE-1. In these 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
- M number of times that an avatar changes its scope of interest
- T_S: average amount of time spent by a user in the virtual environment
- B_P: average bandwidth generated by a participant

Bandwidth is given as the number of packets sent by each object in a given time, and $\frac{SI_PM}{T_S}$ accounts for multicast announcement/state transfer of mobile users (Greenhalgh, 1997). When an avatar changes its scope of interest (such as when entering in a different room with new content), all objects will need to be loaded; hence, we should expect $\frac{S(I_A+I_P)M}{T_S}$ average bandwidth due to the change of scope of interest. In addition to that, we also expect I_PB_P, which accounts for data transferred by the various participants.

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

$$(I_A+2I_P)\frac{SM}{T_S}+I_PB_P=\frac{I_ASM}{TS}+\left(\frac{2SM}{TS}+B_P\right)I_P. \quad (1)$$

VELVET's incoming traffic is similarly given by

$$(I_A+2I'_P)\frac{SM}{T_S}+I'_PB_P=\frac{I_ASM}{TS}+\left(\frac{2SM}{TS}+B_P\right)I'_P, \quad (2)$$

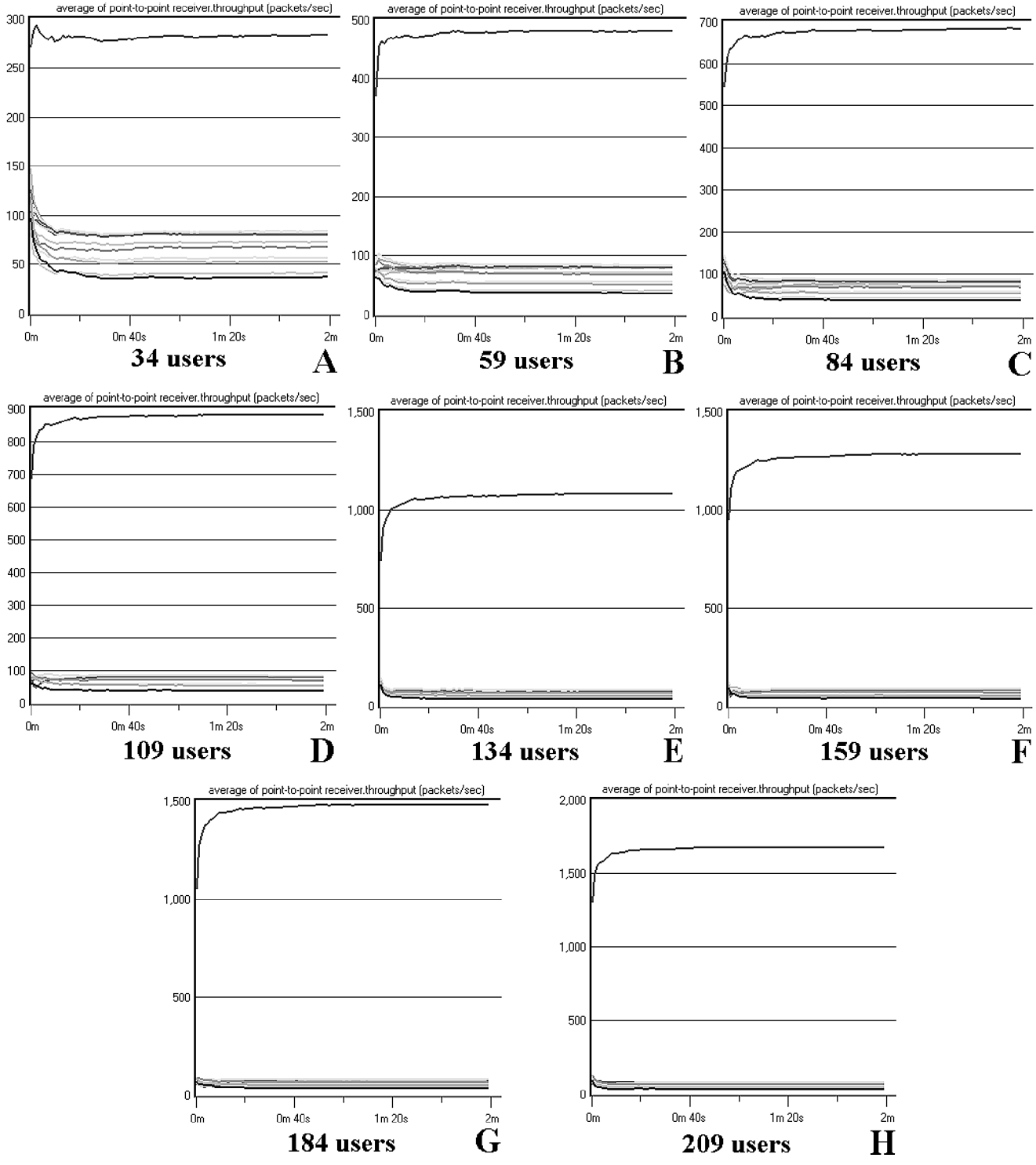

Figure 22. *Incoming traffic: average throughput in packets/second—comparison of VELVET (12 lower lines) vs. space-based solutions (upper line).*

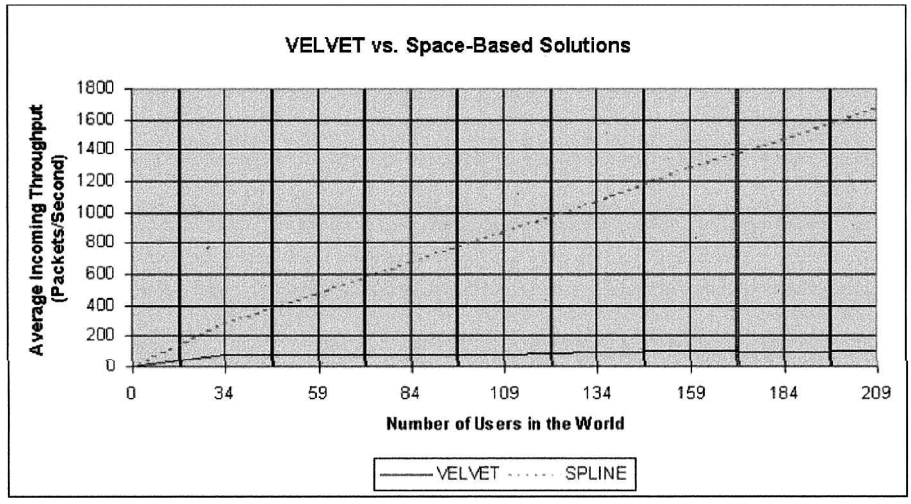

Figure 23. *Comparison of VELVET vs. space-based solutions.*

where $0 \leq I'_P \leq I_P$; that is, VELVET has the second term of the first half of eq. (2), which is responsible for incoming data such as audio, video, motion, and so on in a range from 0 all the way up to the same I_P that MASSIVE-2 receives. If we assume that each participant is on average interested in approximately 50% of objects that he/she is aware of, it would lead VELVET to receive approximately 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, which is not disclosed in the preceding expressions, that has to do with the management of third-party objects. (See subsection 2.4.) It should be remembered that some entity will be responsible for the nontrivial 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 at which the third-party object is located.

MASSIVE-1 (Greenhalgh & Benford, 1995) is an architecture that focuses on small-group meetings, which is why it has not been considered extensively. MASSIVE-1 also uses unicast peer-to-peer distribution, and such an architecture limits its usage to no more than twenty users in a LAN at 10 Mbps (Greenhalgh, Purbrick, & Snowdon, 2000). It did, however, make use of the aura, foci, and nimbi concepts (Benford et al., 1995). Aura, foci, and nimbi, which are also present in MASSIVE-2 (Greenhalgh & Benford, 1999), can be implemented through a metric within VELVET. MASSIVE-3 (Greenhalgh et al., 2000) is an architecture that has moved towards the *locale* model of SPLINE. It adds the concept of "aspects," which are subdivisions of a *locale*, allowing the existence of multiple fidelity representation of objects as well as separation of objects that are located in a single *locale*. Such architecture, being space based, is represented by the other space-based architectures analyzed.

5.2 SCORE

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 for 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 mes-

sages would grow to a significant amount of bandwidth, somewhat presenting the bandwidth problem again. In VELVET, it is of no relevance if users are moving around because the multicast groups that one subscribes to are associated with 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 12).

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

Considering the situation in which SCORE has a fix-sized cell grid defined at startup leads to something similar to a *locale*-based world in which 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 bandwidth than standard *locale*-based solutions in which *locales* are usually larger.

5.3 Howard Abrams's Architecture

Howard Abrams's architecture (Abrams et al., 1998; Abrams, 1999) shows greater filtering capabilities through its three-tiered filtering mechanism, described in subsection 2.6. This architecture is, however, still space based. One may notice that the first tier, being space based, is quite similar to SCORE (in fact, SCORE is similar to the first tier from Abrams's architecture because SCORE was proposed after Abrams's work). VELVET, on the other hand, has its filtering mechanism based on a metric selected by the user (or the creator of a given simulation). VELVET can work similarly to what Abrams's architecture does, if the metric chosen is the Euclidean space distance in the synthetic world, but it may work quite differently if some other metric is chosen.

Additionally, Abrams's architecture has the same complex cell-splitting scheme that is present in SCORE, and also with the same problems with regard to small cell sizes, potentially overwhelming a given object that is moving quickly. The architecture does allow one to select a smaller region, which consists of a threshold used to limit the depth in the octree that is processed. Such a concept presents some complications, such as who will in fact process the tree in all its depth to generate a coarse representation of deeper branches for those who decide to stop some levels up in the tree. Some overhead is also involved when it is decided that a given area is to split in four smaller regions, and a number of control messages must be sent to all objects in the original area (which may be many because their count was the reason for the split in the first place) to reallocate them into the smaller areas (and hence new multicast addresses to be subscribed to).

With regard to heterogeneous collaboration, in Abrams's architecture it is not guaranteed that the number of active objects in one's AoI would be small enough for a given station to handle. Let us suppose that two participants, P_i and P_j, are located nearby. Suppose further than both P_i and P_j have similar interest, for instance, that in tier two they both choose only active objects that have characteristic ξ. In the third tier, let us suppose that both participants selected protocol φ. That means that all active objects with characteristic ξ will be subscribed to by both P_i and P_j. Let us assume, without loss of generality, that there are N active objects in P_i and P_j's AoI. Assume that P_i is a participant who has joined in the CVE through a slow station, which is unable to handle N active objects. VELVET, on the other hand, would allow P_i and P_j to individually select, independently, those active objects that they would choose to add to their AoI according to their own metric. One can notice that VELVET has better support for heterogeneous collaboration, which happens mostly due to the fact that Abrams's architecture is still space based, even though it provides an elaborate three-tier filtering scheme.

A simple extension to Abrams's proposal would be to allow one's AoI to shrink and expand, as described in subsection 3.2.1. This would provide better heterogeneous support; however, one participant, P_i, in a slow system would always subscribe to the active object that passes through tiers two and three and that is closest to its avatar in the virtual Euclidean space. That is not necessarily the active object in which such a participant

would have more interest. For instance, suppose that participant P_i is chatting with the only active object it is able to see/hear, P_j. Let us suppose that another active object, P_k, has similar tier-two and tier-three characteristics that P_j has. If P_k approaches P_i, this one may start to see/hear P_k, rather than P_j; after all, at some point both P_j and P_k would be in P_i's AoI and such an AoI would have to shrink to accommodate P_i's processing limitations. This would cause interruption of an on-going conversation, which is undesirable.

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, with A geographically aware of B but B having no geographical awareness of A, as shown in Figure 9. If users A and B were to physically engage in collision, in the Euclidean world, only one of the parties would be aware of this 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 previously is optionally resolved through emergency collision packets (ECPs) sent from the party that is able to detect collision to the party that is blind to it. In our example shown in Figure 9, A would send a 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 subsection 3.3, each VELVET process signs up for some areas (similar to SPLINE's *locales*) and by doing so becomes aware of 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 who have subscribed to the local transmission channel of A_i. This comparison allows 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 (Singhal, 1996) suggest the use of a server that would know about the position and orientation of every object so that it could inform objects about collisions. VELVET, aiming at an environment with a very large number of users, tries to avoid any kind of server, which is why the collision detection scheme discussed previously is advantageous. A disadvantage of this scheme is that the collisions between two objects that are not geographically aware of each other is not detected. Let us assume that object C is aware and has subscribed to 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; that is, if $A \subset C$ and $B \subset C$, but $A \not\subset B$ and $B \not\subset A$ as defined in subsection 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 that also takes position into account so that one object is more likely to "see" another one that is in the neighborhood, if desired.

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, this 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 that had subscribed to such a 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.

8 Conclusion

We have presented VELVET, an adaptive hybrid architecture for VEry Large Virtual EnvironmenTs. VELVET has been shown to be flexible, allowing a broad range of LSVE, which would otherwise fail, to work gracefully.

However, one disadvantage over other architectures is that VELVET makes use of a potentially larger number of multicast addresses that lead 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 nonartifact object; hence, the cost regarding MGs in VELVET has an upper bound $O(M + P)$ with M being the number of areas in the world (*locales*) and P the number of participants. SPLINE's lower bound MG is $O(M)$, because each *locale* has a multicast group. MASSIVE-2 has an upper bound at $O(M + P)$ as well when considering the case in which 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 needs to deal with only 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 neighborhood. This would lead 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 allow such features without much overhead because all changes can be performed unilaterally by each participant.

It is also worth mentioning that, as technology evolves, routers become faster and more powerful, and such limitations tend to be diminished. 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 that 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 in which every user receives updates from all objects (such as SIMNET), P_r would reduce as the number of objects rise, approaching 0 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 0 in the extreme case as well.
- MASSIVE-2's case would be similar to SPLINE's, potentially having P_r approaching 0 as well. Additionally, one could have special interest in an object that is part of a (potentially recursive) third-party object. This would mean that such a 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 because unsolicited data from objects that 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 that one is really interested in. Some objects of interest may be dropped due to a 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.

Future work includes the actual implementation of a VELVET-compliant CVE, with further enhancements included in what shall be called VELVET-2.

Acknowledgments

We acknowledge the financial assistance of the Brazilian Ministry of Education Agency's CAPES scholarship, the Ontario

Research and Development Challenge Fund (Canada), The Brazilian Intelligence Agency and the Brazilian Ministry of Science and Technology's PCI/LNCC fellowship. We also thank Dr. Seok-Jong Yu for all the input and helpful discussions about this paper, and Mojtaba Hosseini for reviewing this text.

References

Abrams, H. A. (1999). *Extensible interest management for scalable persistent distributed virtual environments.* Unpublished doctoral dissertation, Naval Postgraduate School, Monterey, CA.

Abrams, H. A., Watsen, K., & Zyda, M. (1998). Three-tiered interest management for large-scale virtual environments. *ACM Symposium on Virtual Reality Software and Technology (VRST'98),* 125–129.

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.

Benford, S., Bowers, J., Fahlen, L. E., Greenhalgh, C., Mariani, J., & Rodden, T. (1995). Networked virtual reality and cooperative work. *Presence: Teleoperators and Virtual Environments. 4*(4), 364–386.

Daily, M., Howard, M., Jerald, J., Lee, C., Martin, K., McInnes, D., & Tinker, P. (2000). Distributed design review in virtual environments. *ACM Collaborative Virtual Environments,* 57–63.

Diot, C., & Gautier, L. (1999). A distributed architecture for multiplayer interactive applications on the Internet. *IEEE Network, 13*(4), 6–15.

Doppke, J. C., Heimbigner, D., & Wolf, A. L. (1998). Software process modeling and execution within virtual environments. *ACM Trans. on Software Engineering and Methodology, 7*(1), 1–40.

Fernando, T., Murray, N., Tan, K., & Wilmalartne, P. (1999). Software architecture for a constraint-based virtual environments. *ACM VRST'99,* 147–154.

Frécon, E., & Stenius, M. (1998). DIVE: A scalable network architecture for distributed virtual environments. *Distributed Systems Engineering Journal, 5*(3), 91–100.

Greenhalgh, C. (1996). *Dynamic embodied multicast groups in MASSIVE-2.* (Tech. Rep. No. NOTTCS-TT-96-8), *Department of Computer Science,* The University of Nottingham, UK.

———. (1999). *Large scale collaborative virtual environments.* Distinguished Dissertation Series. New York: Springer-Verlag.

Greenhalgh, C., & Benford, S. (1995). MASSIVE: A virtual reality system for tele-conferencing. *ACM Transactions on Computer Human Interfaces (TOCHI), 2*(3), 239–261.

———. (1999). Supporting rich and dynamic communication in large scale collaborative virtual environments. *Presence: Teleoperators and Virtual Environments, 8*(1), 14–35.

Greenhalgh, C., Purbrick, J., & Snowdon, D. (2000). Inside MASSIVE-3: Flexible support for data consistency and world structuring. *ACM Collaborative Virtual Environments 2000 (CVE 2000),* 119–127.

Hagsand, O. (1996). Interactive multi-user virtual environments in the DIVE system. *IEEE Multimedia, 3*(1), 30–39.

Hook, D. J. V., Rak, S. J., & Calvin, J. O. (1994). Approaches to relevance filtering. *Proceedings of the 11th DIS Workshop on Standards for the Interoperability of Distributed Simulation,* 367–369.

Hindmarsh, J., Fraser, M., Heath, C., Benford, S., & Greenhalgh, C. (2000). Object-focused interaction in collaborative virtual environments. *ACM Transactions on Computer-Human Interaction, 7*(4), 477– 509.

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

Kirner, T. G., Kirner, C., Kawamoto, A. L. S., Cantão, J., Pinto, A., & Wazlawick, R. S. (2001). Development of a collaborative virtual environment for educational applications. *ACM Web3D,* 61–68.

Lety, E. (2000). *Une architecture de comunication pour environnements virtuels distribues à grande-echelle sur l'Internet.* Unpublished doctoral dissertation, L'Universite de Nice-Sophia Antipolis, France.

Lety, E., Turletti, T., & Bacelli, F. (1999). Cell-based multicast grouping in large-scale virtual environments (Tech. Rep. No. 3729 INRIA).

Macedonia, M. (1995). *A network software architecture for large-scale virtual environments.* Unpublished doctoral dissertation, Naval Postgraduate School, Monterey, CA.

Macedonia, M., Zyda, M., Pratt, D., Barham, P., & Zeswitz, S. (1994). NPSNET: A network software architecture for large scale virtual environments. *Presence: Teleoperators and Virtual Environments, 3*(4), 265–287.

Mil3, Inc. *OPNET modeler — Simulation kernel manual* (vol. 1 and 2).

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), 294–299.

Oliveira, J. C., Shen, X., & Georganas, N. D. (2000). Collaborative virtual environment for industrial training and e-commerce. *IEEE VRTS 2000*, 288.

Oliveira, J. C., Shirmohammadi, S., & Georganas, N. D. (1999). Collaborative virtual environments standards: A performance evaluation. *IEEE DiS-RT'99*, 14–21.

Singhal, S. (1996). *Effective remote modeling in large-scale distributed simulation and visualization environments*. Unpublished doctoral dissortation. Stanford University, Stanford, CA.

Singhal, S., & Zyda, M. (1999). *Networked virtual environments, design and implementation*. New York: ACM Press/ Addison Wesley.

Wang, L., Wong, B., Shen, W., & Lang, S. (2002). A Java 3D-enabled cyber workspace. Communications of ACM, *45*(11), 45–49.

Waters, R. C., Anderson, D. B., & Schwenke, D. L. (1997). *The interactive sharing transfer protocol version* 1.0 (MERL Tech. Rep. No. TR-97-10).