Toward a Peer-to-Peer Shared Virtual Reality

Joaquín KELLER
France Telecom Research & Development
38 rue du Général-Leclerc
92794 Issy Moulineaux - France
joaquin.keller@rd.francetelecom.com

Gwendal SIMON
IRISA / INRIA-Rennes
Campus Universitaire de Beaulieu
35049 Rennes - France
gwendal.simon@rd.francetelecom.com

Abstract

This paper envisions a shared virtual reality system that could handle millions of users and objets. The SOLIPSIS system does not rely on servers and is based on a network of peers that collaborate to build up a common virtual world. Real-time interactions and virtual co-presence are enabled by the means of emissive and receptive fields *that determine how avatar kinestesia and multimedia streams are established. We describe here the overall architecture and the main algorithms that make SOLIPSIS conceivable.*

1 Introduction

Shared virtual reality is very popular among science fiction fans and it has been pictured out in many books and movies in which is known as *cyberspace* [8], *metaverse* [12] or *"the matrix"* [13].

Of course, shared virtual reality is not only fiction and it is widely used in many areas like networked gaming, collaborative working or military training.

However, unlike their fictional counterparts, these virtual playgrounds, meeting rooms or battlefields can only be shared by a limited number (tens or thousands but not millions) of simultaneous participants.

Essentially, limitation comes from the fact that actual implementations of shared virtual reality rely on servers and/or IP multicast. Servers based solutions reach their limits when servers reach theirs and multicast based shared VR systems are bounded by the limited number of available IP multicast addresses.

The system we are designing and building, SOLIPSIS[1], intends to be scalable to an unlimited number (millions, billions or more) of users and objects. Since we aim to make SOLIPSIS accessible to low end computers at connected at 56Kbs and to mobile wireless devices and not only to full featured broadband connected engines the not so well deployed IP multicast was discarded. And as we mainly focus on solving the scalability issue, the idea arises of getting rid of servers and building up a solution solely based on a network of peers (*à la* Gnutella [2]).

Section 2 explains how users connect together, how they get aware of their neighbors and surroundings, and how they stroll around and move from one virtual place to another. Section 3 shows how data and multimedia streams are established between peers in order to allow real-time communication and enable virtual co-presence.

2 A Dynamic Peer Network

The peers or nodes of the SOLIPSIS network, are called "communicating entities" or, for short, "entities". An entity may be associated to an user and then represents her in the shared virtual reality. In this case, the entity is an avatar of this user in the virtual world. Also, a communicating entity may not be associated to anyone and the entity then represents an (virtual) object or robot of the world.

Each communicating entity e runs on one computer connected to the Internet[2]. An entity is identified by an IP address plus some local data like port numbers and/or a local identifier. So any entity can reach and communicate (exchange data, messages or streams) with any other entity.

An entity e is determined by its position (x_e, y_e) in the virtual world[3], an awareness radius r_e and the set $k(e)$ of its adjacent entities.

When entities e and e_1 are adjacent in the SOLIPSIS network ($e_1 \in k(e) \Leftrightarrow e \in k(e_1)$), they know each other and are able to communicate. So, in this document, when referring to entities of the SOLIPSIS graph or network, "is

[1] The SOLIPSIS name comes from a philosophical doctrine that claims that reality only exists in one's mind.

[2] At first, SOLIPSIS will not take into account firewalls, NATs or ad-hoc wireless networks. Nor it will take in account mobile software based entities.

[3] The geometry of SOLIPSIS is for now a surface but all algorithms could probably be extended to a 3D space.

adjacent", "knows" or "is connected" should be considered as synonyms. Adjacent entities exchange, at least, their respective positions on a regular basis.Since communication between people is the *raison d'être* of the SOLIPSIS system, entities may also exchange data like video, voice or avatars movements (*cf* Section 3).

To "be" somewhere, an entity must know its environment. But, due to evident physical limitations, an entity cannot be omniscient, so it can only be aware of its *local* environment. The awareness radius r_e defines the area in which the entity e should know every entity: $\forall e_1 \in E \quad d(e,e_1) < r_e \quad \Rightarrow \quad e_1 \in k(e)$, where $d(e,e_1)$ refers to the classical Euclidean distance between the two entities e and e_1.

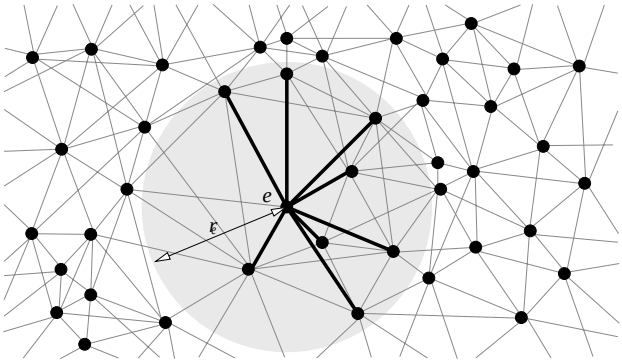

Figure 1. Awareness radius

One important point to understand here is that the SOLIPSIS network is highly dynamic: anytime, users get in and out of the system, entities move (change their virtual positions), physical connections get lost, machines get down, entities make new connections while they drop others, et cætera...

Therefore, when we say the network has some specific characteristic (*e.g.* the local awareness property), we indeed mean that the behaviors or forces that lead to this characteristic prevail over disorder. So the system eventually reaches a sort of stability in which the property is respected almost always and almost everywhere. In that way, inconsistency remains local and never lasts.

2.1 Maintaining the Topology of the Network

To maintain its local awareness, an entity can only rely on its adjacent nodes. Within this collaboration, when an entity e gets aware of a new entity e_1 or a new position of an entity e_0, e informs every concerned entity it knows.

The awareness area of e is the disk of radius r_e and centered on (x_e, y_e) and is written A_e. When $e_0 \in k(e)$ moves, *i.e.* changes its position, e applies the following rules:

R1: for all $e_1 \in k(e)$, if e_0 enters A_{e_1} send e_0's data to e_1

R2: if some $e_1 \in k(e)$ enters A_{e_0} send e_1's data to e_0

Rule R1 ensures that e_1 will get inform of the arrival of e_0 so it will be able to establish a connection with it. And rule R2 ensures that moving entity e_0 will acquire information about its new environment as it goes forward.

But if e_1 does not know any entity in some sector, it will hardly know about an entity e_0 arriving from this sector. Conversely, if e_0 moves forward a sector with no known entity, it will hardly get aware of entities e_1 it should meet on its path. To avoid this situation, an entity e should always ensure that it knows another entity in every 180 degrees sector. We call this property, *global connectivity*, in the sense that this property not only ensures that entity will never "turn its back" to a portion of the world but also ensures the overall connectivity.

First, this impacts on the geometry of the SOLIPSIS universe, that must be unlimited (however not infinite): in a limited world, an entity located at the external "borders" of the world will never be able to address this property. That's why SOLIPSIS space is a torus, probably one of simplest finite unbounded surface. Moreover, torus have, locally, the properties of a plane.

Second, this also impacts the topology of the SOLIPSIS network or graph, that has no *peninsulas*[4]and therefore no *islands*[5]. The topology of the SOLIPSIS network looks like a mesh, in some places loosely weaved, but that covers the whole torus surface. As for the local awareness property the *global connectivity* is dynamically maintained by a local behavior that every entity must implement.

To maintain the global connectivity property, an entity e that notices that it has lost the property, *i.e.* that it has now a $180°$, or more, "empty" sector[6], should find out (quickly) an entity e_α in this sector. The loss of global connectivity may occur when entity e moves or when some entity in $k(e)$ moves or disappear.

When such event occurs, $k(e)$ might reach a situation in which $e_0 \prec_e e_1$ and $e_0 \widehat{e\ e_1} \geq 180°$. Entity e immediately sends out a message to e_0 requesting for an entity e_α that will restore its global connectivity property. As it is assumed that e_0 respects the property, there is some $e_\beta \in k(e_0)$ with $e_\beta \prec_{e_0} e$ and $e_\beta \widehat{e_0\ e} \leq 180°$. One point is sure: e_β is in $e_0 \widehat{e\ e_1}$ sector and $e_0 \widehat{e\ e_\beta} < 180°$. That does not ensure that $e_\beta \widehat{e\ e_1} < 180°$ but since this new sector is strictly narrow than the previous, a message may be pass out to e_β that will furnish an $e_{\beta'}$ that will narrow again the

[4]By geographic analogy, if you draw the edges of the graph as straight lines and consider that the water are the regions with no drawing, the *global connectivity* property forces all stretches of water convex so there are no peninsulas

[5]see previous footnote

[6]If we sort, counterclockwise, the entities of $k(e)$ in a circular list, two consecutive entities of this list define an empty sector. When $e_0, e_1 \in k(e)$ are counterclockwise consecutive we write $e_0 \prec_e e_1$. The angle of the sector originated in e and delimited by e_0 and e_1 is written $e_0 \widehat{e\ e_1}$

too-wide-empty-sector. Eventually, this recursion will end when an entity e_α that finally restores the global connectivity of e will be supplied (see Figure 2).

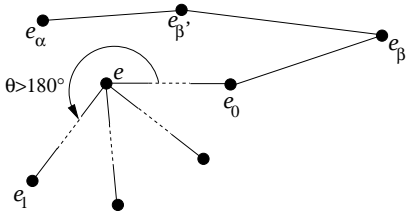

Figure 2. Restoring global connectivity

More refined versions of the algorithm should include: graceful disappearing of an entity that may provide a nearby entity before leaving; starting the search, clockwise and/or counterclockwise, randomly or in parallel, to keep everything symmetric; when several entities candidate to restore the property, a choice should be made upon criteria like nearness, expected aliveness, extension of the awareness area, or verbosity

Previous paragraphs have dealt with how inquire more entities and make more connections. But as entities has limited resources they may also need sometimes to drop out connections. They are allowed to drop out entities located outside their awareness area and that do not break the global connectivity rule. Also, since the whole system is based on collaboration, connections that are needed by others to ensure their local awareness or their global connectivity should not be broken without negotiation. Prior to drop a connection, an entity may choose to lower the cost of the connection by renegotiating the data that flow through (*cf* Section 3).

Finally, within the frame of these constraints, an entity choose to keep or drop connections. Criteria include again nearness, expected aliveness, extension of the awareness area, and verbosity.

If, due to lack of resources, an entity needs to drop connections and cannot do so without breaking local awareness or global connectivity rules, the entity will reduces its awareness radius. This will be the typical behavior of entities in too populated areas. Conversely, in an area entities are sparse, awareness radius should be raised.

2.2 Reverse Localization (a.k.a. teleportation)

Careful readers have certainly noticed, that when an entity e makes an abrupt move to a completely new location, $k(e)$ becomes completely inaccurate and neither local awareness nor global connectivity will be restored using the previously described algorithms. Same problem arise when e has been logged off SOLIPSIS for a long time.

We need a algorithm that allows an entity to restore its local awareness and global connectivity "from scratch", knowing only its expected or desired location. The algorithm is called *reverse localization* in the sense that it takes a location as argument and returns the entities at this location.

When an entity e wants to go to a specific target position $t = (x_t, y_t)$, e first need to know an entity e_0 (it can be e itself before moving) connected to the SOLIPSIS network. We will say e has *reached destination* when its local awareness and global connectivity properties will be ensured for $(x_e, y_e) = (x_t, y_t)$. From e_0 a message will be routed through the SOLIPSIS to the nearest (or near enough) to the target entities. These entities will make contact with e to help it in reaching destination.

To start the search algorithm, a message specifying the target position is sent to e_0 that will forward it to $e_1 \in k(e_0)$ the nearest entity to destination t out of $k(e_0) \cup \{e_0\}$. And so on recursively. Eventually, this heuristic will lead to an entity e_n that is the nearer entity out of $k(e_n) \cup \{e_n\}$. At this point, we will assume that e_n is the nearest entity to destination (that could be false at first hit, since only a local minimum may have been reached). e_n sends then a message to e saying it is the nearest entity to the target and e adds e_n to $k(e)$.

The algorithm enters now a new stage and entities all around the target will be collected. Entity e_n starts the recursive process by sending a message to e_m the nearest known entity after itself.

When an entity e_m receives the message (that carries (x_n, y_n), (x_t, y_t) and contact information for e), it processes it as follows:

If there is an $e_{n'} \in k(e_m)$ that is nearer to the target than e_n, the assumption that e_n was the nearest entity to target does not hold any longer, and the algorithm comes back to the early stage. e_n is forgotten and messages are routed again in the find-the-nearest-entity mode.

e is contacted and e_m is added to $k(e)$. Let e_{m-1} be ($e_{m-1} = e_n$ when starting recursion) the sender of the message. e_m forwards the message to $e_{m+1} \in k(e_m)$, the nearest to target entity in the half-plane delimited by the straight line (t, e_m) and not containing e_{m-1}. Due to the global connectivity property, e_m certainly knows at least an entity in this $180°$ sector, so such an entity (e_{m+1} entity) always exists. Also, since angle $\widehat{e_n\, t\, e_{m+1}}$ is strictly greater than $\widehat{e_n\, t\, e_m}$ we are sure that the algorithm has eventually an end: when angle reaches $360°$.

The algorithm ends when the "trip" around the target ends, *i.e.* when the half-line $[t, e_n)$ is crossed. A message is issued to e telling that it has reach destination.

Entity e is now ready for teleportation, it will take the position (x_t, y_t) and start to act like any other entity: maintaining its local awareness and global connectivity, and helping others to do so. At first, its global connectivity property is ensured by the fact that entities all over the target have been collected, but e local awareness might not be ensured. So e will start with a too wide awareness area to be sure that all nearby entities will be known. e will then reduce it awareness area to its normal size.

At this stage teleportation process has ended, entity e knows all its neighbors and can start multimedia real-time interactions with them.

3 Entities Interactions and Resource Sharing

SOLIPSIS aims to provide real-time communication and data sharing for entities in a virtual world. This implies many simultaneous data flows. Unfortunately, limited resources like CPU and bandwidth have to be shared among these flows. This requires entities to be able to adjust their flow sizes according to physical limitations and still relevant in the virtual world.

The reduction of data flows is a main challenge for all distributed multi-user virtual environment systems that have been designed since the DIS protocol [1]. It exhibits two approaches:

Coarse-grain partition: in 1995, Macedonia [11] proposes to bound the number of interactions by partitioning the world into a grid of disjointed hexagonal cells, each one statically associated with a multicast address. An entity communicates with its neighbors in a cell via the associated multicast channel. Moving implies joining and leaving multicast groups when crossing cell borders. In SPLINE [3], the partition into "*locales*" is mapped on world characteristics (building, rooms,...). Each locale is associated with a separate multicast communication group and every entity receives only messages from entities located in the same locale. To reduce the number of simultaneous allocated multicast addresses and to optimize cell sizes, SCORE [10] introduces dynamic cells. The density of entities per unit of surface is one of the variables taken into account to compute cell extents.

Perception-based approach: the entities perception capabilities may reduce the number of simultaneous interactions [4]. In the space, entities carry their *auras*, sub-spaces which bound the presence of an object and which act as an enabler of potential interaction. Each entity owns different auras for different media. When two auras collide, interaction between the two objects in the medium is triggered and the environment takes the necessary steps to put the objects in contact with one another. The concept of aura is adopted by systems like DIVE [6], Continuum [7] or MASSIVE [9]. Once the interaction of objects is determined, the objects themselves control it by computing, for a given medium, a level of *awareness* between them via *focus* and *nimbus* [4]. The amount of exchanged informations depends on this awareness.

3.1 Adjustable Data Flows

It seems natural to use high quality data flows with close entities, data flows requiring few resources for far entities and to not exchange flows with entities outside the awareness area.This simulates the real world and avoid exchange data with all the entities.

At any moment, each entity has to receive and send several data flows. Among their, we identify:

System flows: Global connectivity and local awareness require to frequently receive and send messages. The awareness radius, the rate the aliveness notifications (*heartbeats*) should be sent, and the movement amplitude that triggers a motion notification, are adjustable parameters that allow to define many different data flow format descriptions ordered by message frequency. However, some unpredictable entity behaviors may lead to an unexpected variation on the volume of system flows, *e.g.* when an entity disappears.

User related flows: Virtual co-presence requires video, audio and avatars kinesthesia flows. Each of them are adjustable: video streams depend on the compression algorithm, the frame size or the frame rate; audio streams on codec types and sampling rate; and avatars kinesthesia on the exhibited[7] number of degrees of freedom. These parameters can be adjusted to define many different flow format more or less resources consuming.

So, each entity has to know, for each medium, several data flow formats, that differ by their resource requirements.

3.2 Emissive and Receptive Fields

Let D_m be the set of data flow format descriptions (codec type and parameters, data exchange rates...) for medium m (audio, video, avatar kinesthesia, position notification flows...). Depending on its capabilities, each entity e is able to use only a subset D_m^e of D_m. We consider that two data flow format descriptions δ_1 and δ_2 can be ordered: we say that $\delta_1 > \delta_2$, when δ_1 is more resource consuming than

[7] *E.g.* from far, the fingers of an avatar are invisible, so the avatar may exhibit less degrees of freedom.

δ_2. We note $\emptyset_m$, the data flow format description corresponding to no data flow for the medium m.

An entity e, for each medium m, in emission and in reception, determines a field F_e, defined at a given position (x, y) by:

$$F_e(m, x, y) \in D^e_m$$

$F_e(m, x, y)$ is maximum for $(x, y) = (x_e, y_e)$ and, in a given direction, decreases to $\emptyset_m$ as distance $d\big((x, y), (x_e, y_e)\big)$ increases. This ensures that e will exchange more data with nearby entities and no data at all with too distant entities.

To avoid that small movements lead to no ending renegotiations, fields do not vary continuously. A given field jumps from one *level* to another, so large portions of the world are at the same level.

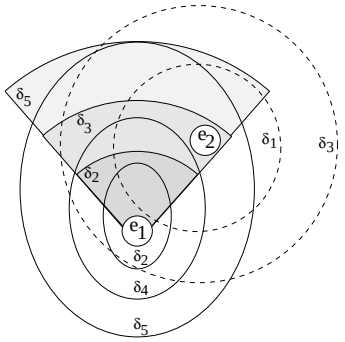

Figure 3. Fields of two entities e_1 and e_2

In Figure 3, entity e_1 is associated to an user. The ovals and the paint in grey zones are world portions at the same level for *resp.* its emissive video field and its receptive video field. Entity e_2 represents an object. Its receptive video field is constant to $\emptyset_v$, while circles bound zones at same emissive video levels.

$\delta_1, \delta_2, \ldots, \delta_5, \ldots$ are video format descriptions, which differ by their codecs (MPEG, H.263, MJPEG...), their frame rates (VGA, CIF, SIF, QCIF...) or their frame rates (30 fps, 25 fps...).

In this example, we have:

$F^{out}_{e_2}(video, x_{e_1}, y_{e_1}) = \delta_3$ and $F^{in}_{e_2}(video, x_{e_1}, y_{e_1}) = \emptyset_v$

$F^{out}_{e_1}(video, x_{e_2}, y_{e_2}) = \delta_5$ and $F^{in}_{e_1}(video, x_{e_2}, y_{e_2}) = \delta_3$

3.3 Entities Interaction

We note $\delta^{e_1 \to e_2}_m$, the format description of the data flow sent by entity e_1 to e_2 for the medium m.

Flow communication requires that both sender and receiver know this format description:

$$\delta^{e_1 \to e_2}_m \in (D^{e_1}_m \cap D^{e_2}_m) \quad (1)$$

To respect heterogeneity, the chosen data flow format must not require more resources than field values:

$$\delta^{e_1 \to e_2}_m \leq min\big(F^{out}_{e_1}(m, e_2), F^{in}_{e_2}(m, e_1)\big) \quad (2)$$

The optimal $\delta^{e_1 \to e_2}_m$ is the higher quality flow format respecting equations (1) and (2).

A negotiation requires only round per entity. The entity, which detects a potential interaction, sends flow format descriptions and field values, while the other determines $\delta^{e_1 \to e_2}_m$ and $\delta^{e_2 \to e_1}_m$ and ends the negotiation by answering its results, before to start flow exchanges.

At any moment, an entity may start a renegotiation if the previously chosen flow format requires more resources than its field value.

In the example of figure 3, negotiation leads to $\delta^{e_2 \to e_1}_v = \delta_3$ and $\delta^{e_1 \to e_2}_v = \emptyset_v$.

3.4 Fields Computation

The goal of field computation is to ensure an optimal resource utilization that respects virtual environment coherency. Each entity computes its fields in order to share out its resources among many simultaneous flows of different medium. Such computation depend on the available CPU and network resources (limited resources implies a faster decay or field shapes), on the entity willings (an entity may prefer to receive video streams than to send them), on the entity abilities (a machine without camera will not send video streams) and on the density of entities in its surroundings. This requires each entity to know the amount of available resources (CPU and bandwidth) and the amount of required resources for each data flow format.

Prior to exchange multimedia flows an entity must ensures its global connectivity and local awareness properties. So an entity computes first its awarness area and virtual position flow fields and the remainder of resources are for entities interactions.

For an entity e, if R_{tot} is the total available resources, R_{sys} the expected resources needed for system flows, p_m the number of flow format in D^e_m, $n_{i,m}$ the number of interactions using the flow format δ_i for medium m and $R_{i,m}$ the expected resources needed by a flow format δ_i for medium m, fields computation should verify:

$$\sum_m \sum_{i=1}^{p_m} (n_{i,m} * R_{i,m}) \leq R_{tot} - R_{sys} \quad (3)$$

An entity may choose to reduce $n_{i,m}$ by decreasing the field extents, while another may prefer reduce $R_{i,m}$ by choosing less costly flow formats for the fields levels. User satisfaction seems to be unpredictable, because she may be pleased with many poor connexions or with few great connexions. But neighbor satisfactions is important

in a way close entities should receive almost similar format description from one entity, far entity should not receive better format description than proximate one, approaching entities should receive new and better format description... Resources should be fairly shared with virtual world coherency.

Since, at any moment, several entities may appear and/or other systems than SOLIPSIS might use the physical resources, the inequation (3) may become invalid and lead to parameter readjustments and/or renegotiations between entities. So an entity should keep enough available resources as a safety margin to delay field recomputations and atenuate the propagation flow renegotiations.

The fields mechanism ensures that the resources are optimally utilized and, also, more resources are used for closed interactions. More refined computation should include: when interactions are low, all the resources work for the "community" of peers; field computation takes into account several resource types; system flows fields are optimized depending on connectivity to SOLIPSIS network...

4 Conclusion

As SOLIPSIS does not rely on servers, the system load is equally distributed among all the nodes of the SOLIPSIS network and no bottleneck will prevent the system to scale up to an unlimited number of communicating entities.

Also, since an entity (user or object) mostly communicate within its virtual neighborhood, at no more than its own capabilities, the amount of data traffic for one entity should remain pretty constant regardless the global number of entities connected to the SOLIPSIS system. And moreover, each entity uses and contributes to the system at the exact level of its capability, so low end computers and light mobile appliances, eventually poorly connected, as well as powerfull well connected workstations may take part in the SOLIPSIS network.

The only SOLIPSIS features that generate messages farther than the local neighborhood are the "teleportation" and the "global connectivity recovery". However, these features, certainly, do not flood the network as Gnutella searches do (a system that somehow works), but do not even generates as much messages as Freenet [5] searches since SOLIPSIS messages, unlike Freenet ones, do not make use of backtrack as they travel through the network.

The only point that might be an issue for SOLIPSIS scalability and usability is that each movement (teleportation as well as local movements) generates flows of messages and induces local (in time and space) inconsistencies. But, since SOLIPSIS intends to be a "place" to meet and communicate, we expect users to stay steadily chatting together and not to move all around all the time (SOLIPSIS is not a car race game;-). This expectation is strengthened by the fact that the global cost of a movement is perceived by the user as a latency, so her movement have also a cost for her.

SOLIPSIS is now at the design stage and we are developing a simulator for thousands of nodes as well as a prototype of a node. This will help us in achieving the fine tuning of the algorithms and might lead to some enhancements. Next step will be the specification of an open protocol that will allow anyone to build up its own SOLIPSIS node.

Acknowledgments

The authors would like to acknowledge Pr. Emmanuelle Anceaume and France Télécom R&D scientific board that have made likely the SOLIPSIS project.

References

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

[2] The Gnutella Protocol Specification v0.4. available on http://www.gnutella.co.uk/, September 2000.

[3] J. W. Barrus, R. C. Waters, and D. B. Anderson. Locales and Beacons : Efficient and Precise Support for Large Multi-User Virtual Environment. Technical report, Mitsubishi Electric Research Laboratory, August 1996.

[4] S. Benford, J. Bowers, L. E. Fahlén, and C. Greenhalgh. Managing Mutual Awareness in Collaborative Virtual Environments. In *VRST'94*, August 1994.

[5] I. Clarke, O. Sandberg, B. Wiley, and T. Hong. Freenet: A distributed anonymous information storage and retrieval system. In *Proceedings of the ICSI Workshop on Design Issues in Anonymity and Unobservability*, Berkeley, CA, 2000.

[6] E. Frécon and M. Stenius. DIVE - A Scaleable network architecture for distributed virtual environments. *Distributed Systems Engineering Journal (special issue on Distributed Virtual Environments)*, Vol. 5, September 1998.

[7] A. Gérodolle, F. Dang Tran, and L. Garcia Bañuelos. A Middleware Approach to Building Large-Scale Open Shared Virtual Worlds. In *TOOLS Europe 2000*, June 2000.

[8] W. Gibson. *Neuromancer*. 1984.

[9] C. Greenhalgh and S. Benford. MASSIVE, a Distributed Virtual Reality System incorporating Spatial Trading. In *Proceedings of the 15th International Conference on Distributed Computing Systems*, June 1995.

[10] E. Léty. *Une Architecture de Communication pour Environnements Virtuels Distribués à Grande-Échelle sur l'Internet*. PhD thesis, Université de Nice-Sophia Antipolis, December 2000.

[11] M. Macedonia. *A Network Software Architecture for Large Scale Virtual Environments*. PhD thesis, Naval Postgraduate School, June 1995.

[12] N. Stephenson. *Snow Crash*. 1992.

[13] L. Wachowski and A. Wachowski. The Matrix. Movie, 1999.