Managing Network Troubles while Interacting within Collaborative Virtual Environments

Thierry Duval[1] and Chadi El Zammar[2]

[1] IRISA
Campus universitaire de Beaulieu
F-35042 Rennes Cedex — France

[2] VDL2
700, rue Wellington — bureau 1200 — H3C 1T4
Montréal (Québec) — Canada

Abstract. We are interested in real time collaborative interactions within Collaborative Virtual Environments (CVE). This domain relies on the low latency offered by high speed networks. Participants of networked collaborative virtual environments can suffer from misunderstanding weird behavior of some objects of the virtual universe, especially when low level network troubles occur during a collaborative session. Our aim is to make such a virtual world easier to understand by using some graphic visualizations (dedicated 3D metaphors) in such a way that the users become aware of these problems and can go on working even during these troubles. In this paper we present how two independent mechanisms may be coupled together for a better management and awareness of network troubles while interacting within a networked collaborative virtual environment. The first mechanism is an awareness system that visualizes, through the use of special metaphors, the existence of a network trouble as strong delay or disconnection. The second mechanism is a virtual object migration system that allows the migration of an object from one site to another to ensure a non interrupted manipulation in case of network troubles. We will detail only this awareness system and we will show how it uses the migration system to allow users to go on interacting while network breakdowns occur.

1 Introduction

Within a virtual environment, interacting with virtual objects means that a user can manipulate and control these objects [1]. Furthermore, if this virtual environment is a collaborative one, an interactive object may either be controlled by one user in an exclusive way or by several users in a collaborative way. In both cases the evolutions of the shared environment are more difficult to understand than in a single-user environment because these evolutions can be caused by the other users. And of course, it is more difficult when the evolution is caused by collaborative interactions. We suppose here that these collaborative interactions can occur in real time thanks to the low latency offered by high speed networks. This is the reason why many researchers have focused on making the users aware of the other users' interactions with the objects of the shared

Duval T. and El Zammar C. (2006).
Managing Network Troubles while Interacting within Collaborative Virtual Environments.
In *Proceedings of the 3rd International Workshop on Computer Supported Activity Coordination*, pages 85-94
DOI: 10.5220/0002472700850094
Copyright © SciTePress

virtual worlds [2] [3] [4]. All these studies consider that data transmission over the network is always reliable, so they never matter about the problems that could arise if there were troubles with the network. However, when network troubles happen such as the increase of the network delay or the disconnection of some sites, some problems of consistency can arise between the different instances of the shared universe. We think that we need visual metaphors to make the users aware of these inconsistencies and that we need rescue solutions to allow them to go on working anyway. These users must be aware that there is no possibility to interact with a set of objects, or that the actions of some other users are not updated any longer, so that they can go on interacting with objects of the world that are not disconnected from their site.

We will focus on networked virtual environments that connect a few number of participants who collaborate closely to achieve a specific task, for example to assemble some parts of a car mockup [5]. With such constraints, a network perturbation or disconnection can cause damages to the remote interactions that users are performing. The solutions usually proposed in this domain do not resolve this problem in a global way, despite they try to reduce the effect of network troubles by reducing the amount of circulating updates and messages on the network. These solutions are not sufficient as long as networks are the base structure of distributed virtual environments. This is the reason why we have decided to look at this problem from a different point of view: we admit that these network troubles can arise, so we have to make all the users of a collaborative virtual environment aware of these problems. This way, users may understand and deal with some temporarily network troubles, especially when they are interacting with objects of the virtual universe. Furthermore, we want to provide a prevention method based on virtual object migration that prevents a user from losing the control of a remotely calculated object in case of a network breakdown.

Next section gives a view of related work in this domain, it is a brief glance to different approaches and metaphors that have been used in existing networked virtual environment systems. In section 3 we detail how we manage network troubles at the distributed virtual environment system level, and which visual metaphors we propose to make the users aware of these troubles. In section 4 we explain how we use a mechanism allowing virtual objects to migrate from one site to another. This migration mechanism can be useful to a user, in order to move a distant object from a distant site towards his own site, when he becomes aware of potential network troubles with the site where the object in interaction is initially located. An example of such a scenario is given in section 5. To conclude, we present what more could be done to improve our work.

2 Related Work

2.1 Network Delays and Side Effects

The network affects directly the performance of distributed virtual environments systems. For example when interacting remotely, a user can take the control of a virtual object and manipulate it. Low latency offers a real time interaction by minimizing the delay between user's actions and the object's responses that may be on a very distant site. If a data transmission problem arises on the network, this problem will directly affect the remote interaction. The most frequently types of problems envisaged are the

delay when transmitting data over a network and some little disconnections from time to time due to dynamic rerouting systems. If the disconnection time of a site increases during a distributed virtual simulation, the consequences can be catastrophic.

A network disconnection can drive different kinds of perturbations to a shared distributed virtual reality session. Whatever the kind of the data model the virtual reality system is using (centralized, distributed, ...) we can not avoid a periodic communication between all the sites that are sharing the same virtual world, especially when users interact with objects of the shared universes. So, from the interactivity point of view, when a network disconnection occurs, the user's interactions that exist on the disconnected site are not seen any longer by the other sites. This same user is not able any longer to see the interactions of the other users and it also produces collaboration breakdowns.

When an object is evolving linearly, we can tone down the communication problem between different sites by using a prediction system that achieves local computing to estimate for example a future position of a moving object. NPSNET implements this method using the "Dead Reckoning" algorithm [6]. But when we are facing a complex evolution or totally unpredictable actions like users' interactions with virtual objects (interaction may depend on user skills or mood), such prediction systems are unable to manage the situation in a virtual world, so we find ourselves powerless facing a technical challenge. Some other systems like SPIN [7] duplicate all universes objects and perform parallel calculations locally on each site. In this case network delay is not meaningful to a virtual session from the animation point of view, but this architecture does not resolve the main problem when objects are interactive because the possible interactions of a user can not be "duplicated", so we can not avoid the anomalies provoked by delays and disconnections. OpenMASK [8] implements a predictive architecture based on the concept of referentials and mirrors, which is similar to the distribution concept in NPSNET. A referential is an independent entity that can evolve through internal state calculus. A mirror is some kind of light copy of a referential that performs no internal state evolution. The evolution of a mirror relies completely on updates received from its associated referential. The referential/mirror paradigm of the open-source OpenMASK[3] architecture will be the base of our studies, so we have decided to inform users of the presence of a network problem (if any) referring to referentials and mirrors points of view. First, we signal to users who interact on sites having referentials that other users may not see modifications and interactions performed on the referential side. Second, we signal to users who interact on sites having mirrors that some of these mirrors may not have updated values.

2.2 Providing Awareness of Network Troubles

In [9] Vaghi et Al. have presented an experiment of a collaborative two players ball game where one player was subjected to an increasing amount of delay. They observed that as the network delay increases, the users (being aware) modify their strategies in an attempt to cope with the situation. Fraser et Al. have made a step forward in [3] by giving visibility to what they have called "delay induced phenomena". They have implemented a system that estimates the maximum difference between the "objective"

[3] www.openmask.org

position of a user avatar, and where another might perceive it to be, according to network delay times between the users and speed of motion. For example, in case of network delay, an avatar is shown surrounded by a sphere that represents all uncertain possible positions, and this sphere evolves according to the network delay.

In [10] we have presented two different methods to make network troubles visible and also to make a user aware of possible inconsistencies in the virtual environment. The first method is a marker system that marks all mirror objects existing on a particular site to make a user aware that these objects are neither interactive nor updated anymore. The other method is the creation of an echo object that is associated to each mirror object in the virtual world. This echo is visualized at the same place than the original object's place. In case of delay between the process of the referential and the process of its mirror, we can see a spatial gap between the motion of the referential and the motion of the echo associated with this mirror that shows in this case the position variation between these two processes. This last method presents some limitations when a large number of sites are participating to a simulation due to the very important number of echoes, which will be harmful to visualization.

3 Detection and Awareness of Network Troubles

Whatever the kind of the data model the virtual reality system is using (centralized, distributed, ...), we can not avoid a communication between all the sites that are sharing the same virtual world, especially when users interact with the shared universes. One of the common synchronization methods in distributed systems is the use of periodic synchronization messages. This method ensures a hard real-time synchronization between all sites. Hard real-time applications require a response to events within a predetermined amount of time in order to function properly. Network delays or troubles will deny events and updates from coming in time, which will cause the breaking of the real time concept. In a distributed virtual reality simulation context, the consequence of breaking real time may be one of the two following scenarios: freeze the whole virtual world on all sites until a synchronization message shows up, which is very harmful for the session especially if the delay is important; or let the distributed virtual world goes on even in case of delay or disconnection. This second scenario will split up the virtual world into several parallel worlds, due to different users interactions on the same virtual object, which is a very bad choice. In this paper we present our work, which is a mix of the two scenarios. We choose to let the simulation continue by freezing only the parts of the world whose state is uncertain for consistency considerations. We let the virtual simulation evolve even in case of presence of non updated values due to latency or disconnection. So, the virtual world is not out of use while waiting the reception of updates as it is in hard real time synchronization based distributed systems. Even objects interactivity will be preserved but only for objects that are calculated locally. Remote calculated objects lose their interactivity as long as the disconnection remains.

3.1 Detection of Network Delay or Disconnection

Usually, all participant sites to a virtual simulation using OpenMASK are sending each other synchronization messages. Synchronization between different sites aims to co-

ordinate different processes to evolve similarly in the virtual world. Synchronization messages are used to synchronize an object with its distant mirrors and to carry the updates from a referential to its mirrors. A synchronization message may even contain only a dating element in case of no new changes in the virtual world. Once a participant site is not receiving synchronization messages from one or more sites any longer, these sites will be declared as disconnected. Each site conserves a list of disconnected sites that is not necessarily the same on each site. The synchronization message time-out threshold has to be carefully determined according to the characteristics of the network, otherwise it could downgrade the system performance, either by too frequent creation of unnecessary echoes or by detecting the troubles too late.

3.2 The Awareness Provider System

Our aim is to make the participants aware of any possible network trouble, this is why we have implemented, in the kernel of OpenMASK, some services that provide useful informations to the higher level applications. In [10] we did a first step for realizing an information system provider implemented in the kernel of OpenMASK. Next we will detail the limitations of this first implementation and we will explain the solutions that we have proposed and realized to enhance our system.

The Echoing System. We want to make users aware that sometimes their interactions are not seen by other users as they should be, because of network delay or disconnection. The idea is to use an echo object that represents the state of its associated real distant object. Initially, the echoing system had been implemented in a static way.

The Static Echoes Management. On each process, for each mirror object in the virtual world, we associate an echo object that follows the motion of the mirror. The echoes association is made in a special configuration file where we specify manually which objects in the world will have echoes. The echo is a referential, it will have mirrors on the other sites as all OpenMASK simulated objects and it is visualized at the same place than the original object (e.g. the referential of the mirrors it is echoing). Physically the echo has the shape of its original object but is a little bit smaller and half-transparent so that we can not see it when a simulation is going on normally. We can associate as many echoes as we have mirrors in the virtual world. When the delay between the process of the referential and the process of its mirror is important, we may encounter some inconsistencies between the state of the two processes. For example, in case of spatial motion we can see a gap between the motion of the referential and the motion of the echo associated with its mirror (Fig. 1 (a)). In case of a disconnection, the echo associated with the mirror existing on the disconnected site is frozen on the screen and it does not evolve any longer. This means that the mirror concerned with this echo is not receiving updates any longer because of the disconnection.

The disadvantage of this method is that each referential object of a scene will have as many echoes as there are mirrors (participant sites). This will overcrowd the scene especially when moving objects while interacting, and it will increase the calculus time that the system will need in case of a huge number of users.

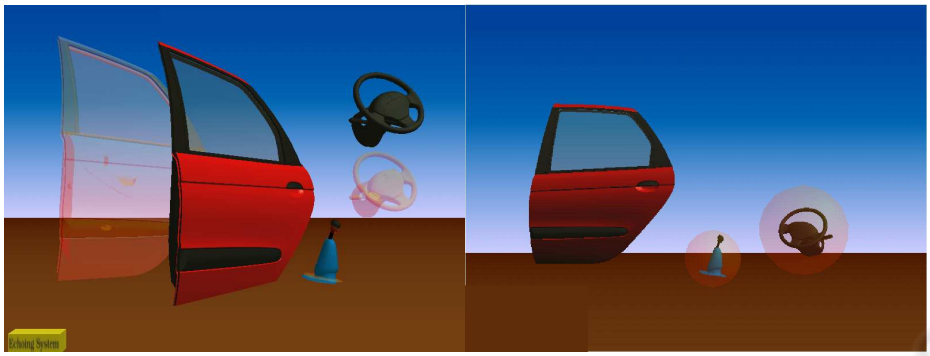

Fig. 1. (a) The echoing system and (b) the Marker System.

For example, consider that N sites are participating to the same virtual simulation. If we have X referential objects within the shared universe, so we will obtain X(N-1) mirror objects distributed on the different participating sites. By applying the static echoes method, we will have as many referential echoes as we have mirrors (X(N-1) referential echoes). These referential echoes will generate a supplementary X(N-1)(N-1) mirror echoes. We can easily imagine the impact in case of an important number of sites beside the hard work of associating manually X(N-1) echo objects.

For the above reason we have decided to change the whole way of creating and managing echoes. Detailed explanation of our new method is given below.

The Dynamic Echoes Management. A better solution to the above cited limitations is to create echoes dynamically in an automatic way, but where and when exactly do these echoes have to be created ? To answer these questions, two approaches will be presented next.

The first approach is to activate dynamically the creation of echoes after the detection of a first network perturbation, and disable the echoes as long as the distributed simulation is going on normally. This method is a clear amelioration relatively to the static creation method, but what if the first perturbation encountered during a simulation was the loss of a participating site ? In this case it is too late to create echoes because we are not able to communicate with this site any longer.

We resolve this problem by providing a new concept that avoids the creation of the referentials echoes on the disconnected site: local objects. Once a site detects the lost of another distant site, it activates locally the creation of local echo objects that appear exactly with the current states (position and orientation for example) of existing referentials. Only one simulation step time separates the physical disconnection of a site from the creation of associated echoes on other sites, so the lost of the last exact value is not really significant (in the order of few milliseconds). Echoes may also appear with the current state of some mirrors in the scene. Actually the dynamic echoes creation system detects not only referentials existing on a site but also mirrors that have “brothers” located on a disconnected site. “Brother” mirrors are mirrors associated to the same

referential. This way, a user becomes aware that his interactions with a mirror are not perceived any longer by some other users.

We have implemented this last approach in the kernel of OpenMASK and we are now in the evaluation phase of this system. We could also present more informations about disconnected or delayed sites by displaying the name of the disconnected site above the associated echo, or may be by using different echo colors in case of a few number of participating sites. Each color could then represent a particular site.

The Marker System. In [10] the marker system is a very limited service that marks a specific kind of objects existing on a particular site (for example the mirror objects): we use a 3D button that launches the marker system once pressed by the user. The marker system surrounds all mirror objects (or may be another kind of object) by a half-transparent sphere that makes it clear to the user that these marked objects are not holding the last updated values (Fig. 1 (b)). The limitation of this version of the marker system is that it was not able to be more specific by designing only mirrors damaged by a delay, so it was marking all the mirrors.

We have extended this system in order to be more specific about objects that we need to mark. For example, the new marker system is able now to mark only the mirrors associated to referentials that exist on a disconnected site without marking all mirror objects. This new realization offers two main advantages. First we can give to the user a very specific idea concerning the disconnected sites. Second we do not charge the scene by undesirable marks unless the user really wants to mark all the objects belonging to a particular type. We can take also a supplementary advantage from the fact of surrounding frozen mirrors, which is the protection from the user attempts for interaction. Actually, even if the user is no longer able to interact with a disconnected mirror, all his attempts are stocked in a special PVM buffer [4]. Once the site holding this mirror is reconnected, all orders stocked in this buffer are transmitted to the referential that may be confused at this time because of some contradictory orders. So, as we surround mirrors with non-interactive objects, it prevents order attempts issued from 3D direct interaction of being transmitted to the PVM buffer.

4 Migration of Virtual Objects

As we have mentioned earlier in the introduction, beside detecting and visualizing network troubles to the users, we want to provide them a rescue technique based on virtual objects migration. Some few virtual reality systems like AVIARY [11,12] and WAVES [13] have implemented object migration, but only to ensure load balancing.

In our case we use object migration to ensure a non-interrupted control of a specific object chosen by the user [14]. Let us remind the reader that, on each site, when network troubles occur, a user can only control referential objects. Mirror objects lose their interactivity as they perform no calculation and receive their updates from distant referentials. If a user is interested in keeping the control of a specific object or a set of

[4] OpenMASK's distributed version rely on PVM: www.csm.ornl.gov

objects, he can claim the need of these objects to the migration system that ensures that the user will own these objects locally on his site.

We have implemented the object migration system at the kernel level of OpenMASK, as detailed in [15]. We choose to migrate an object by changing the state of his referential and mirrors rather than recreating it on the destination site and destroying the original object on the source site. For example, to migrate an object from $site1$ to $site2$ the migration system changes the state of the mirror of the object on $site2$ to make a referential of it and then it changes the state of the referential on $site1$ to make a mirror of it. If no mirror exists on $site2$, the migration system will first create one. This way there is no destruction of existing objects when they migrate. If there where some other mirrors of this object on other sites, they are informed that they have a "new" referential and the OpenMASK kernel ensures that they receive now their updates from the new site of the object.

5 A Scenario Example

To illustrate the interest of coupling the migration system with the awareness system we are going to present a use case of these mechanisms. Three librarians located on distant sites are invited to arrange different sections of a virtual library. Network troubles begin to appear progressively while the librarians are performing their job. Let's see how the awareness system and the migration system will help to allow librarians to continue their job even when such network troubles occur:

1. as soon as network troubles (as delay or disconnections) begin to show up, librarians will be aware of these troubles thanks to the awareness system (marker and echoes systems). These troubles will be seen by users thanks to the appearing and disappearing of echoes and marks.
2. moreover, awareness system will provide users informations about the objects upon which they will lose control (mirror objects) in case of network disconnection: user will lose the control of all marked objects.
3. each librarian may be interested by keeping the control of all objects (books) that he must arrange in his section. This way he will not be interrupted by a strong delay or disconnection on the network. Thus, each librarian can ask the migration toward its site of all the distant books he is interested in.
4. once migration procedure is over, users may disable, if they want, the awareness system and continue working without being disturbed by network troubles.
5. when a user has arranged his section, he may move onto another one hoping that network troubles are over. If it is not the case and if troubles are only strong delays on the network, the user may migrate once again objects he wants to manipulate for a better local manipulation.

This way, our contributions will allow the users to go on working, even in case of network troubles as delays or disconnections, by focusing on tasks that may be realized by one user and does not require a collaborative work.

Of course, if these librarians have to realize some tasks that need simultaneous interactions of several users on the same object at the same time (for example moving

cooperatively a desk or some shelves), these tasks can not be realized efficiently while there are some network troubles. And in case of a temporarily network breakdown, such interactions are impossible to achieve, because it would need to make the user himself migrate from one site to another before the breakdown. Of course it is possible only if the user is a virtual one (for example an autonomous agent), but it is impossible for a real user, unless he physically goes by himself to a site that is not disconnected. So, for such kind of closely coupled interactions, we can only give an advice to the users: within a shared virtual environment that can suffer from some network troubles, they should realize the cooperative tasks that need closely coupled interactions as soon as possible, and delay them when network troubles appear by doing then the work that can be realized alone.

6 Conclusion

We have presented our mechanisms to detect and visualize network problems while interacting within a networked virtual environment. Two kinds of metaphors are used to inform users about the availability of the updates of an object: echoes for users having referentials, and marks for those having mirrors. Those metaphors are coupled with a virtual object migration mechanism implemented within the OpenMASK platform. This migration mechanism is used to ensure a non interrupted control and manipulation of an object by migrating this object onto the site of a user who wants to interact with it, since local interactions are not sensitive to a network problem.

Our contributions, especially virtual objects migration, allow several kinds of extensions such as the dynamic management of processes and users by adding or removing sites to an already running simulation. Another possible extension is the dynamic management of areas of interest by migrating automatically a set of objects to a particular site depending on the interest of users, for example depending on the 3D position of the user, to ensure that most of the objects he can interact with will be located on his site.

There are also interesting research perspectives about how to make the users aware of the differences between referentials and mirrors. Indeed, in this article we have talked mainly about visualizing differences between geometrical parameters of our virtual objects (such as position, orientation, scale factor or color). What about other kind of parameters ? Of course, our simulation kernel is able to notice the differences between any kind of values, and if we are able to visualize any of them, we can show to a user the last known value of any of them. For example a value of pressure or temperature could be visualized with a 3D device or simply thanks to a 2D or 3D text near the virtual object. If the virtual object has also moved, the user will see at two different positions these different values, thanks to our echoing system. But what if the virtual object did not move ? We will have to propose a way to visualize both values “at the same place”: the current one and the last sent to the disconnected mirror.

What would be more difficult would be to show to a user, with our marker system, the possible range of values that could have been taken by the parameter during a network breakdown. For some 3D parameters such as the position of an object, we could visualize a 3D area within which the virtual object could be (according to its last known 3D position, speed and acceleration for example) as proposed in [3]. But how could we

show other kind of areas of possibilities ? We can imagine 3D intervals with bounds that would evolve dynamically for real values like pressure or temperature, thanks to the record of their last known variations, but what could be used to show areas of possibilities for colors ? Or to show the current target of a tracker that would change its target randomly from time to time ? Of course, for this last example we can not propose anything, but there could be interesting fields of investigation with more predictable kinds of parameters.

References

1. Bowman. D., Kruijff E., LaViola J., Poupyrev I.: 3D User Interfaces: Theory and Practice. Addison-Wesley Eds (2004)
2. Fraser, M., Benford, S., Hindmarch, J., Heath, C.: Supporting Awareness and Interaction through Collaborative Virtual Interfaces. UIST'99, Asheville, USA (1999) 27–36
3. Fraser, M., Glover, T., Vaghi, I., Benford, S., Greenhalgh, C., Hindmarch, J., Heath, C.: Revealing the Realities of Collaborative Virtual Reality. CVE'2000, San Francisco, USA (2000) 29–37
4. Gutwin, C., Greenberg, S.: Design for Individuals, Design for Groups: Tradeoffs Between Power and Workspace Awareness. CSCW'98, Seattle, Washington, USA (1998) 207–216
5. Duval, T., Le Tenier, C.: Interactions 3D coopératives sur des objets techniques avec OpenMASK. Mécaniques et Industries **5** (2004) 129–137
6. Macedonia, M., Zyda, M., Pratt, D., Barham, P., Zeswitz, S.: NPSNET: a Network Software Architecture for Large Scale Virtual Environments. Presence, Vol3, No.4 (1994) 265–287
7. Dumas, C., Degrande, S., Saugis, G., Chaillou, C., Viaud, M., Plenacoste, P.: SpIn: a 3D Interface for Cooperative Work. Virtual Reality Society Journal (1999)
8. Margery, D., Arnaldi, B., Chauffaut, A., Donikian, S., Duval, T.: OpenMASK: Multi-Threaded or Modular Animation and Simulation Kernel or Kit : a General Introduction. VRIC 2002, Laval, France (2002)
9. Vaghi, I., Greenhalgh, C., Benford, S.: Coping with Inconsistency due to Network Delays in Collaborative Virtual Environments. ACM symposium on Virtual reality software and technology (1999) 42–49
10. El Zammar, C., Duval, T.: Management and Awareness of Delay Perception when Interacting within Networked Virtual Environments. VRIC'2003, Laval, France (2003)
11. West, A., Howard, T., Hubbold, R., Murta, A., Snowdon, D., Butler, D.: AVIARY - A Generic Virtual Reality Interface for Real Applications. Virtual Reality Systems (sponsored by the BCS), May 1992. Also published in Virtual Reality Systems, eds R.A. Earnshaw, M.A. Gigante and H. Jones, Academic Press, 1993 (1993) 213–236
12. Snowdon, D., West, A.: The AVIARY VR System: A Prototype Implementation. 6th ERCIM Workshop (1994)
13. Kazman, R.: Load Balancing, Latency Management and Separation of Concerns in a Distributed Virtual World. Parallel Computations - Paradigms and Applications (1996)
14. Duval, T., El Zammar, C.: A Migration Mechanism to Manage Network Troubles while Interacting within Collaborative Virtual Environments. to appear in VRCIA'2006, Honk-Kong, China (June 2006)
15. El Zammar, C.: Interactions coopératives distantes en environnements virtuels : gestion des problèmes réseau. PhD thesis, INSA/IRISA (2005)