Joint Virtual Reality Conference of EuroVR - EGVE - VEC (2010)
T. Kuhlen, S. Coquillart, and V. Interrante (Editors)

A New Adaptive Data Distribution Model for Consistency Maintenance in Collaborative Virtual Environments

Cédric Fleury[1,2,4], Thierry Duval[1,3,4], Valérie Gouranton[1,2,4], and Bruno Arnaldi[1,2,4]

[1]Université Europenne de Bretagne, France
[2]INSA de Rennes, F-35708 Rennes
[3]Université de Rennes 1, F-35042 Rennes
[4]INRIA, IRISA, UMR CNRS 6074, Rennes, France
{cedric.fleury, thierry.duval, valerie.gouranton, bruno.arnaldi}@irisa.fr

Abstract

Ensuring that all the users see the same state of a Collaborative Virtual Environment (CVE) at the same time is very important to provide effective collaboration between these users. Absolute consistency is nearly impossible to achieve because it would prejudice the system responsiveness during user interactions. Consequently, existing solutions make a trade-off between consistency and system responsiveness according to their own requirements. We propose a new adaptive data distribution model that is able to dynamically change data distribution according to application requirements, user's actions and functions that virtual objects fulfill in the virtual environment. Our solutions can deal with several kinds of requirements imposed by various applications and network constraints. The choice of the data distribution can be made at the object level because all the objects of a virtual environment do not necessarily have the same need for consistency. Finally, we evaluate this model for collaborative scientific data visualization using a client/server architecture and HTTP/HTTPS connections. The results show that our model can minimize both interaction latency and gap in consistency between users, so it enables users to always perform real-time interactions in a consistent CVE.

Categories and Subject Descriptors (according to ACM CCS): Information Interfaces and Presentation [H.5.3]: Group and Organization Interfaces—Computer-supported cooperative work; Computer-Communication Networks [C.2.4]: Distributed Systems—Distributed applications; Computer Graphics [I.3.7]: 3-Dimensional Graphics and Realism—Virtual reality

1. Introduction

Collaborative Virtual Environments (CVE) enable several users to work together in a natural way and provide them with a truly interactive experience. When the users are not located in the same geographical place, each user uses his own computer to have individual interaction capability and to meet the other users in the same virtual environment. So collaborative work must be achieved over a local area network (LAN) or a wide area network (WAN) that connect users' computers, which we will call "nodes". For example, remote experts can analyze scientific data together using an Internet connection. Such network connections have a strong impact on the consistency of the shared virtual environment because they delay transmissions of state modifications. Moreover, some users can interact in a CVE through immersive devices, powerful computers, and high-bandwidth network connections, while some other users share the same CVE through simple workstations and low-bandwidth network connections. Even if inadequately powered computers or low-bandwidth network connections put some users at a disadvantage, these users have the right to share the same state of the virtual environment as the other users. Ensuring CVE consistency is the best way to make an effective collaboration possible between users because it prevents misunderstandings when users perform collaborative tasks.

To define the consistency of a CVE, Delaney et al. [DWM06] explain that a CVE must be considered as a distributed database with users modifying it in real-time. Consistency maintenance consists in ensuring that this database is the same for all the users, i.e. users observe or interact

with the same data. Consistency is directly linked to system responsiveness. This responsiveness during interactions can be quantified by the system latency, i.e. the time between user's action and system response. This latency is due to transmission time over the network and to processing delays of the events. According to Delaney et al. [DWM06], maximum values for latency fluctuate between 40 and 300 ms to ensure real-time interactions, and a maximum latency of 100 ms seems sufficient for most of applications. Improving the consistency of a CVE can increase latency during interactions and vice versa. Users can get annoyed with this lack of responsiveness, so CVE systems must reach a trade-off between consistency and responsiveness.

We propose an adaptive data distribution model to deal with several kinds of requirements imposed by various applications and different network constraints. This model makes it possible to dynamically change the data distribution during a collaborative session to adapt it to the tasks that users perform in the virtual environment. The choice of the data distribution can be made at the object level rather than at the application level, because all the objects of a virtual environment do not necessarily have the same need for consistency.

In this paper, Section 2 examines the data distribution used in existing CVE systems and its impact on consistency and system responsiveness. Section 3 presents our adaptive data distribution model. Then section 4 describes how this model has been used in a CVE system dedicated to scientific data visualization. Finally, section 5 compares measurements of gap in consistency and responsiveness with the different data distribution modes proposed in our model.

2. Related Work

As stated by Macedonia et al. [MZ97], the location of the virtual environment data (i.e. geometric data, textures, etc.) is a critical decision when designing a CVE system. It determines which computers store this data and which computers execute the processing related to each virtual object. We distinguish three data distribution modes: centralized, homogeneously replicated, and partially replicated. Further details about the impact of system architectures on CVE consistency can be founded in [FDGA10].

2.1. Shared centralized world

Some systems such as Vistel [YTAK95] use a database shared by all the nodes. All the CVE data is stored on a central server (client/server network architecture). Similarly, virtual object behaviors are executed on this server. When a user wants to modify an object, he sends a request to the server. It processes the modification request, then transmits the up-to-date state of the object to all the nodes, including the one that has asked for modification. This method implicitly ensures consistency between all the nodes and avoids data replication, but it has two main drawbacks:

- Each modification request has to pass through the server before returning to the user's node. So the transmission delays can introduce latency during user interactions.
- With many users, a performance "bottleneck" can appear on the server because it has to send updates to all the nodes at the same time (especially when using *unicast*).

2.2. Homogeneous replicated world

In many CVE systems such as SIMNET [CDG*93] or MR Toolkit [SG93], all nodes are initialized with a same database that contains all information about the virtual environment (geometric models, textures, object behaviors, etc.). Similarly, bcDSG [NLSG03] replicate a shared scene graph on all the nodes. The data can already be present on the node when the user logs in (as in most video games). Otherwise, data must be replicated from a server or from the other nodes already connected. During a session, the database evolves independently on each node and all object behaviors are executed locally. A synchronization mechanism can be used to control the executions of behaviors on each node. Object modifications are performed locally before being sent to the other nodes by using update messages. So latency is very low during user interactions. Only update messages are transmitted over the network in order to enable all nodes to update their database. So few messages are transmitted over the network. However, data replication also has some drawbacks:

- Data replication can introduce inconsistencies between users' states of the virtual environment because delays or losses during transmission of update messages can appear.
- Users can locally perform object modification, and some conflicts can only occurred when the modifications are transmitted to the other nodes. So additional mechanisms must be used to manage concurrent access to the objects.
- This solution is not really appropriate for very large data sets such as scientific data or complex CAD models, because each node may store all the data.

2.3. Partially replicated world

Many CVE systems choose hybrid solutions between totally centralized and totally replicated data distributions to avoid some of their drawbacks. These solutions distribute the data and their processing among the nodes. For example, RAVE [GAW09] uses a central server to process object behaviors, and uses asynchronous transmission to perform "best effort" collaboration. Nodes maintain a local copy of the virtual environment, only receiving update messages from the server.

DIVE [FS98] distributes the virtual object data among the nodes: when a new user joins a collaborative session, only the necessary objects are replicated on his node instead of downloading the whole virtual environment data. The database can be seen as a shared and distributed memory. DIVE uses peer-to-peer connections to manage communications between nodes (transmissions of updates or object data

when necessary), but it also uses a server to backup the distributed data. It is useful to save the whole state of the virtual environment when users log out, to restore a session later, or to transmit the current state to a new user. With this kind of data distribution, many users can share a very large amount of data without necessarily replicating all this data on each node. However, if a user needs additional objects during the session, they must be dynamically downloaded. Moreover, consistency is hard to maintain and induces a high communication cost between nodes. Indeed, update messages must be sent to all the nodes, because a node cannot know which other nodes shared the same objects.

To reduce this communication cost, BrickNet [SSP*95] uses a server to keep information about the shared objects: access rights, ownership, list of nodes that share the object. When a user wants to modify an object, his node must first ask the server for the modification rights on this object (only one user can modify an object at the same time). When its request is granted, it can modify this object locally and transmit the new object state to the server. Then the server transmits this new state only to the nodes that require an update. With this approach, the server eases the consistency maintenance and the concurrent access management.

In OpenMASK [MAC*02], each object behavior is executed only on one node. To achieve this, a referent/proxy paradigm is used for each object. The referent is assigned to a particular node and defines the object behavior. On the other nodes, proxies are created to represent the object. A proxy provides the same interface as the remote referent (same inputs, same outputs, etc.). However, no processing is done locally and the proxy is driven by its referent. The OpenMASK kernel maintains the consistency between the referent and its proxies. When a user modifies an object whose referent is on his node, the object is first locally modified, then an update is sent to all the other nodes. But, when a user modifies an object whose referent is on a remote node, the modification is first computed on the node that owns the referent. Then the remote node transmits updates to all the nodes, including the node that asked for modification. In this second scenario, transmission delays on the network can introduce latency during interactions, but migration mechanisms can be used to move the referent to the interacting user node [DZ06]. This referent/proxy paradigm makes it possible: (1) to combine the processing power of all the nodes, (2) to ensure a better consistency of the virtual environment, and (3) to manage concurrent access to the objects implicitly (only the referent can be directly or indirectly modified).

Similarly, Schmalstieg et al. [SRH03] proposes to use referent/proxy paradigm to distribute a shared scene graph. Each node only replicated a part of the scene graph according to its location in the virtual world. Nongraphical application data are also embedded in the scene graph. For each object, a particular node is responsible for processing its data. A migration mechanisms can be used to change this node.

All these hybrid solutions mix features of the centralized world and features of the totally replicated world to meet particular requirements of consistency and responsiveness. The partially replicated world enables CVE systems to make a trade-off between the advantages and drawbacks of the two other data distributions. For example, replicating only the necessary objects on each node avoids replicating all the data as in the centralized world, and it makes the system as responsive as in the replicated world. However, these hybrid solutions also have some drawbacks: in the previous example, the consistency is still hard to maintain.

2.4. Synthesis

A universal solution, which would meet the requirements of all CVE systems, does not yet exist. Existing CVE systems have chosen the most adapted data distribution mode to meet their requirements, and they have done a trade-off between CVE consistency and system responsiveness. So each solution has some advantages, but also some drawbacks. It would be interesting to combine the advantages of each solution to deal with several kinds of application and several network capabilities. Contrary to the network architecture that is often imposed by the technical requirements, the data distribution can be adapted according to the application requirements (collaboration, interactivity, etc.), the tasks that users are performing during a session, and the functions that the objects fulfill in the virtual environment.

3. A New Adaptive Data Distribution

Some applications require a good responsiveness to user's actions, while some others require a strong consistency of the virtual environment. Moreover, during the same collaborative session, some manipulation tasks require a good responsiveness, while some collaboration tasks require a strong consistency. Finally, in a same virtual environment, some objects shared by several users can require a strong consistency, while some other objects do not. Consequently, it would be interesting to adapt the data distribution to each particular case, especially when the network capabilities are limited (low bandwidth, high latency). So we propose an adaptive data distribution model to reach the best trade-off between consistency and responsiveness according to the requirements of each case and to the network constraints.

This adaptive data distribution model is based on a referent/proxy paradigm that enables our CVE system to:

- implement the three modes of data distribution,
- define a particular data distribution mode for each object,
- dynamically change the data distribution mode.

On each node, an object is represented by a referent or a proxy. A referent stores the application data of the object, executes the object behavior, and processes the modification requests for this object. According to the data distribution, a referent can also send update messages to its proxies on the

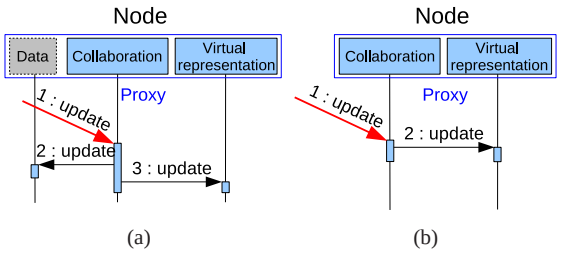

Figure 1: *Proxy (a) stores up-to-date object data, contrary to proxy (b) which only updates the visual representation*

other nodes. A proxy only stores the application data of the object and updates this data when it receives update messages from a remote referent (see Figure 1(a)). So a proxy performs no processing locally, but it keeps an up-to-date state of the object and transmits the modification requests from the user of its node to a remote referent. Even if the proxy has the same inputs and outputs as a referent, it is necessarily controlled by a remote referent.

This first referent/proxy distribution (see Figure 1(a)) makes an easy migration of the referent possible (see part 3.3), because all the application data is already on the proxy. When a proxy must become a referent, it has just to start to process the object behavior and the modification requests. No additional data transfers are required, but the application data are replicated on all the nodes. If we do not want to replicate this data, it is possible to use a second referent/proxy distribution: a proxy can directly update the object outputs (visual representation, sound, etc.) when it receives an update message from a referent, but it does not store any data in the application part (see Figure 1(b)). In this case, application data is not replicated, so it requires transferring all this data to achieve a migration of the referent.

3.1. Three Modes of Data Distribution

According to the location of the object referent, three modes of data distribution can be implemented without major system modifications. Each data distribution mode meets particular requirements of consistency and responsiveness. In this part, gap in consistency (*GC*) and interaction latency (*IL*) are quantified in an abstract way according to the value *L* of network latency. This network latency *L* corresponds to the time needed to transmit a message between two nodes or between a node and a server. To simplify the explanations, we assume that processing delays of events (updates, modification requests) are negligible in comparison to network latency. Actual measurements of interaction latency and gap in consistency can be found in section 5.

3.1.1. Centralized Mode

To achieve a centralized data distribution for an object, only one referent is put on a server. All the other nodes have a

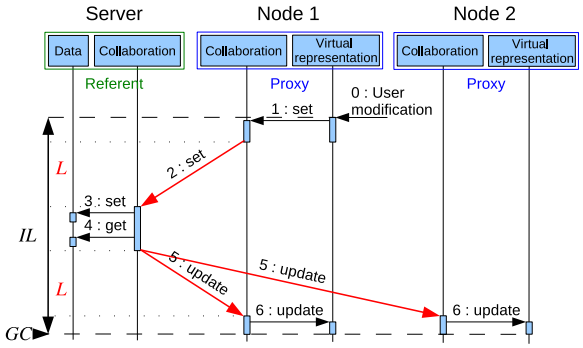

Figure 2: *Modification of an object in centralized mode*

proxy that is controlled by the referent on the server. Behavior and modification requests are processed on the server, and then update messages are sent to all the nodes. So *GC* is quasi-null if all the nodes have similar network connections, but *IL* is about 2*L* during user interactions (see Figure 2).

3.1.2. Hybrid Mode

To achieve a hybrid data distribution for an object, only one referent is put on a particular node. All the other nodes have an object proxy that is controlled by this remote referent. For behavior execution and modification processing, *GC* is quasi-null for all nodes, except for the node which owns the referent: it perceives the object state with an advance of *L* in comparison with the other nodes. *IL* can be quasi-null if the user interacts with the referent (see Figure 3(a)), but it can be about 2*L* if the user interacts with a proxy (see Figure 3(b)). However, migration mechanism can be used to move the referent to the interacting user node (see part 3.3)

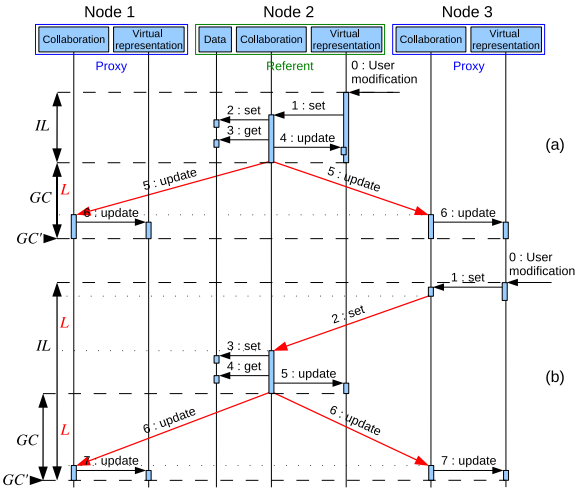

Figure 3: *Modification of an object in hybrid mode when user is interacting with the referent (a) or with a proxy (b)*

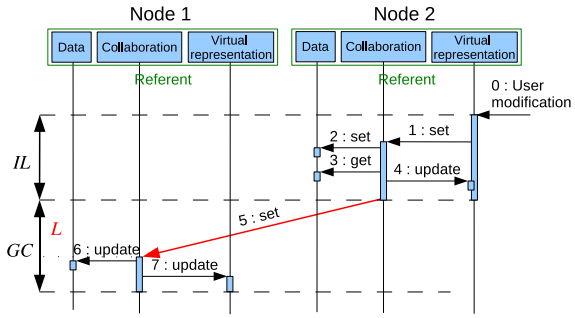

Figure 4: *Modification of an object in replicated mode*

3.1.3. Replicated Mode

To achieve a replicated data distribution for an object, the referent/proxy paradigm must be adapted: one referent for this object is put on each node. The object behavior is executed on each node and no update messages are sent between nodes. So the gap in consistency can be very large if no synchronization mechanism is used between nodes. A synchronization mechanism enables object behavior to be executed at the same time on each node, and reduces the gap in consistency. For modification, requests are processed locally on the interacting user node, then update messages are sent to the other nodes (see Figure 4). So *IL* is quasi-null, but *GC* is about *L* for modifications.

This data distribution mode offers at least the same or better responsiveness during user interactions than the hybrid mode. However, to guarantee the same CVE consistency between all the nodes, a strong synchronization must be achieved, especially with objects whose behavior independently evolves in time. Processing all the object behaviors on each node and achieving the synchronization cost a lot of processing power on each node, so it can reduce the user interaction capability. Moreover, simultaneous interactions with a same object by remote users are quite impossible because each node locally computes its own modifications.

3.1.4. Quantitative comparison of the three modes

Table 1 summarizes values of interaction latency and gap in consistency of the three data distribution modes. This table takes into account the time between a user's action and the system response on his node (user *IL*), on the other users node (others *IL*), and the gap in consistency between these users (*GC*). We can see that only the centralized mode guarantees a perfect consistency, whereas the replicated mode makes low latency during interactions possible. Finally, the hybrid mode can ensure at the same time a good consistency and low latency interactions, but it requires to move the object referent to the interacting user node.

Data distribution mode	user IL	others IL	GC
Centralized	2L	2L	0
Hybrid *Referent on user node*	0 *(referent)*	L *(proxy)*	L *(ref. ↔ proxy)* 0 *(proxy ↔ proxy)*
Hybrid *Proxy on user node*	2L *(proxy)*	L *(referent)* 2L *(proxy)*	L *(ref. ↔ proxy)* 0 *(proxy ↔ proxy)*
Replicated	0	L	L

Table 1: *Interaction latency and gap in consistency for user interaction according to the data distribution mode*

3.2. A Particular Data Distribution for each Object

Using a referent/proxy paradigm enables us to choose a particular data distribution for each object, contrary to the existing CVE systems, which are designed with only one data distribution mode for all the object. In fact, for each object and on each node, a simple boolean indicates if the object version is a referent or a proxy, and consequently if this object version has to process the object behavior and modification requests or not. So the referent locations can be different for each object, and an object can also have several referents on different nodes. When several referents are used for a same object, a concurrency control must be achieved to avoid that concurrent modifications are done on each referent.

The choice of the data distribution at the object level is motivated by the fact that each object does not have the same need for consistency according to the function that this object fulfills in the virtual environment. For example an object, such as a pointer, used to show something to another user, or temporally animated objects which have to progress at the same time on each node will require a strong consistency. However, a static object decorating the virtual environment will probably not require a strong consistency. So it can be interesting that this kind of objects requires just few network transmissions in order to reduce the global communication cost. Moreover, some tools used for object manipulation will require a good responsiveness during interactions instead of a strong consistency.

3.3. Dynamic Changes of Data Distribution Mode

With the referent/proxy paradigm, it is quite simple to dynamically change an object data distribution mode during a collaborative session. If the data is already on proxies (see Figure 1(a)), only the status boolean has to be changed to indicate that a proxy is now a referent, and vice versa. If the data is not on proxies (see Figure 1(b)), object data has to be transferred from a referent, before a proxy can become a referent. So data distribution can be easily changed from one mode to another to adapt this data distribution to user's actions or network troubles during a session. In the same way, the referent can be moved to the interacting user node when the hybrid mode is used (migration mechanism). The following examples illustrate dynamic changes for each data distribution mode:

- centralized mode → hybrid mode: if a user needs to use a 3D pointer to precisely manipulate a virtual object, the

3D pointer referent can be moved to the user node to offer good responsiveness during the manipulation.
- hybrid mode → centralized mode: if now the user wants to use this 3D pointer to show a point of interest in the virtual environment to another user, the 3D pointer referent can be moved to a server to ensure that the two users see this pointer at the same location at the same time.
- centralized or hybrid mode → replicated mode: in case of network troubles, the replicated mode can be used to reduce network communications and enables users to interact in spite of network issues.
- replicated mode → centralized or hybrid mode: if an object behavior suddenly requires high processing power for specific processing as scientific data computation, the object referent can be moved on a server or on a node with high processing power to not overload all the nodes.

Rules for data distribution changes can be determined by the object behaviors or by the application controller. So this rules could be defined in the application configuration files or in the object description files (see section 6).

4. Instantiation of the Data Distribution Model for Collaborative Scientific Visualization

This adaptive data distribution model has been used to design a new CVE system dedicated to collaborative scientific data visualization. This project about collaborative scientific data visualization aims to enable remote experts to visualize complex scientific data together by sharing a common virtual environment in an industrial context. So it imposes several requirements on the system design:

- The amount of scientific data is very large and users need to perform post-processing on this data. So the data must be computed on a cluster.
- This data can be confidential data from industry. So users can only access to the data through a secured portal and secured connections such as HTTPS connections.
- The CVE system must be easy to deploy in an industrial context. So it has to be able to deal with mainstream hardware, standard internet access, and restricted firewalls that only authorize HTTP/HTTPS connections.
- The project has many different fields of applications from fluid mechanics simulation to CAD model assembling. So the CVE system has to enable users to perform various tasks in the virtual environment from simple visualization of animated objects to collaborative manipulations.

To deal with these requirements, our system uses a client/server network architecture. It makes it possible to have a “single point of truth”: the server enables users to access to data on the cluster and to log on the secured portal. HTTP/HTTPS connections through this server are used to deal with security restrictions, standard internet access, and easy deployments in industrial contexts. Using a client/server network architecture can have a strong impact

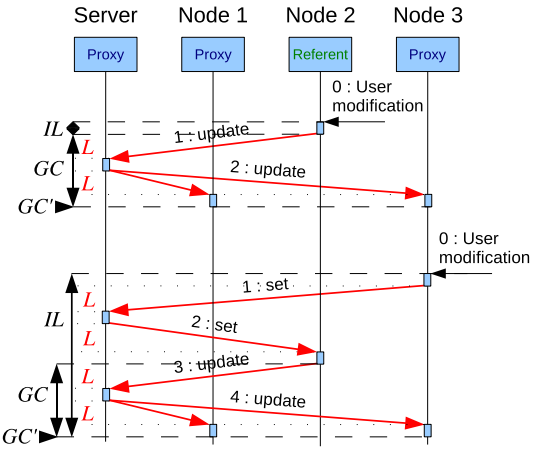

Figure 5: *Modification of an object in hybrid mode when a client/server architecture is used*

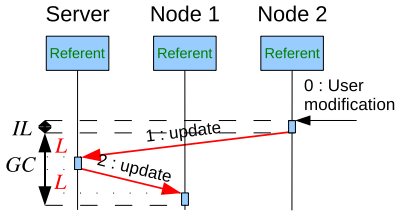

Figure 6: *Modification of an object in replicated mode when a client/server architecture is used*

on interaction latency and gap in consistency according to which data distribution mode is used. For the centralized mode, these values are the same because all communications already pass through a server in this mode. However, for the hybrid mode, the interaction latency when a user interacts with a proxy and the gap in consistency are doubled (see Figure 5). And, for the replicated mode, only the gap in consistency is doubled (see Figure 6). Despite these drawbacks, the hybrid and replicated mode are useful with a client/server network architecture. Even if the scientific data are computed on the server, these two modes enable users to have good responsiveness when they manipulate the scientific data (modify pre-computed parameters, etc.). They also make it possible to execute simple behaviors on the client such as temporal animation.

Table 2 summarizes the new values of interaction latency and gap in consistency of the three data distribution modes when a client/server network architecture is used. The features of each mode are still the same than when communications do not have to pass through a server, but differences between modes are increased.

To achieve HTTP/HTTPS connections, a “long polling" technique, as in ShareX3D [JFM*08], is used to overcome the fact that it is not possible for HTTP to make requests

Data distribution mode	user IL	others IL	GC
Centralized	2L	2L	0
Hybrid *Referent on user node*	0 *(referent)*	2L *(proxy)*	2L *(ref. ↔ proxy)* 0 *(proxy ↔ proxy)*
Hybrid *Proxy on user node*	4L *(proxy)*	2L *(referent)* 4L *(proxy)*	2L *(ref. ↔ proxy)* 0 *(proxy ↔ proxy)*
Replicated	0	2L	2L

Table 2: *IL and GC for user interaction according to the data distribution mode with a client/server architecture*

from server to nodes. Establishing a new connection after each message sending slows down the communication, especially when the events are sent with high frequency rates. So this technique increases the value L of the network latency, and consequently differences of interaction latency (*IL*) and gap in consistency (*GC*) between each data distribution mode are also increased.

A "lockstep synchronization", as in OpenMASK [MAC*02], can be achieved for the three modes of data distribution. For replicated data distribution, it ensures that the object behavior is executed at the same time on all the nodes. It is the only way to guarantee consistency, otherwise each node executes the object behavior as fast as it can and inconsistencies quickly appear. For centralized or hybrid data distribution, the consistency is already guaranteed by the data distribution model, but a synchronization can be used to improve the consistency by ensuring that the update messages are processed at the same time on all the nodes.

5. Experimental Results

We performed measurements on our CVE system. We ran an experiment on an Intel® Core™2 laptop and on two Intel® Xeon® workstations: the laptop ran a server, while each workstation ran a user's node. We created an object in a simplified virtual environment and we measured the latency (user *IL*) during interactions with this object and the gap in consistency (*GC*) between the two nodes when the object was modified. We perform simple translations and rotations on this object. The measurements were done:

- for the three mode of data distribution,
- with a strong synchronization (*Sc*: Yes) between server and nodes or with no synchronization (*Sc*: No),
- at several scheduler frequencies (30, 60 and 120 Hz),
- on a LAN (Table 3) or a WAN (Table 4).

The scheduler processes simulation steps at a given frequency, so the frequency value impacts message sending and receiving rates. The strong synchronization is performed with a "lockstep" managed by the server. For LAN, machines were connected by 100Mb Ethernet. For WAN, they were connected over the Internet by about 450 km of professional network and 350 km of standard Internet network between server and nodes. Tables 3 and 4 show mean values (in ms) of 2000 object modifications for each case.

Sc	Data distribution	user IL Frequency (Hz) (Sim. step time (ms)) 30 (33)	60 (17)	120 (8)	GC Frequency (Hz) (Sim. step time (ms)) 30 (33)	60 (17)	120 (8)
No	**Centralized**	64.7	29.4	20.1	4.5	4.2	2.8
	Hybrid - *Referent*	0.005	0.005	0.005	39.0	22.0	12.6
	Hybrid - *Proxy*	104.4	55.4	35.5	42.9	25.0	17.2
	Replicated	0.005	0.005	0.005	37.2	21.8	12.4
Yes	**Centralized**	65.1	36.2	27.6	0.0	0.2	0.2
	Hybrid - *Referent*	0.005	0.005	0.005	45.5	24.3	14.4
	Hybrid - *Proxy*	120.0	56.1	42.7	46.1	24.3	18.0
	Replicated	0.005	0.005	0.005	42.3	25.2	14.0

Table 3: *IL and GC (in ms) for user interaction on a LAN*

Sc	Data distrib.	user IL Frequency (Hz) (Sim. step time (ms)) 30 (33)	60 (17)	120 (8)	GC Frequency (Hz) (Sim. step time (ms)) 30 (33)	60 (17)	120 (8)
No	**Centralized**	200.6	180.4	160.4	18.6	23.7	22.1
	Hybrid - *Ref.*	0.004	0.004	0.004	209.3	207.3	207.4
	Hybrid - *Pro.*	405.9	387.2	373.0	195.1	202.3	195.5
	Replicated	0.003	0.003	0.003	215.4	216.4	214.2
Yes	**Centralized**	207.7	203.2	195.5	1.4	1.8	1.9
	Hybrid - *Ref.*	0.004	0.004	0.004	166.0	164.2	161.7
	Hybrid - *Pro.*	410.4	400.1	396.6	166.6	163.6	164.1
	Replicated	0.003	0.003	0.003	166.0	162.9	162.5

Table 4: *IL and GC (in ms) for user interaction on a WAN*

These measurements correspond to the abstract values presented in Table 2. However, when network communications are really fast (LAN), we can see that the frequency impacts the user *IL* and the *GC*. For example, in Table 3, user *IL* in centralized mode is about twice as long as the simulation step time (it requires one simulation step time on the node and one on the server). But, the simulation step time becomes almost negligible in comparison to network latency with slow connections (WAN - Table 4). In the same way, a strong synchronization increases the user *IL* and does not improve the consistency on the LAN because it slows down the simulation step processing. But, on a WAN, a strong synchronization reduces the *GC* and the increase of the user *IL* is almost negligible. Finally, we can see in Table 4 that even the worst interaction case on a WAN (interaction with a remote referent through a proxy in the hybrid mode) seems acceptable according to Delaney et al. [DWM06] (user *IL* between 30 and 400 ms). Nevertheless, our adaptive model makes it possible to always avoid this worst case by either moving the referent on the interacting user node, or changing the data distribution mode (hybrid to centralized).

We also performed measurements of processing load and network communications. This experiment was done on the WAN with a scheduler at 60 Hz and a strong synchronization. So we created 12 animated objects (with a behavior evolving in time) in the virtual environment. First, the 12 objects used the centralized mode (*12 Centr.*). Then, 4 used the centralized mode and 8 used the hybrid mode with 4 referents on the user1's node and the 4 other referents on the user2's node (*3x4 Hybr.*). Finally, the 12 objects used the replicated mode (*12 Replic.*). We measured the execution

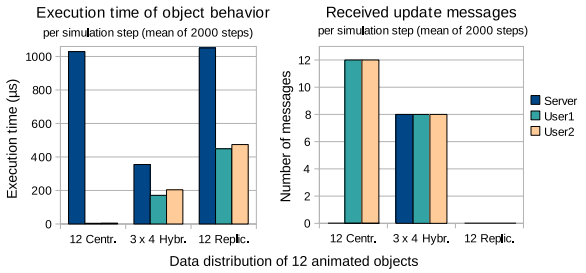

Figure 7: *Execution time of behaviors and update messages received on each node for processing 12 animated objects*

time (in μs) of the object behaviors and the number of update messages received on each node (see Figure 7).

In the centralized mode (*12 Centr*), all processing is done on the server and the other nodes receive an update messages for each object. The hybrid mode *3x4 Hybr* distributes processing among the nodes, but requires a lot of network communications. On the contrary, the replicated mode (*12 Replic*) increases processing on each node, but does not send update messages. For *3x4 Hybr* and *12 Replic*, the difference between execution time on server and on nodes is due to a high processing power of the nodes (workstation).

6. Conclusion and Future Work

To enable users to perform an effective collaboration in a virtual environment, a good consistency has to be maintained between the users' nodes. However, maintaining CVE consistency must not reduce user interaction capability by decreasing the system responsiveness. The stronger the network constraints are (low bandwidth, secured protocols), the more CVE systems must deal with the application's particular requirements to reach a good trade-off between consistency and responsiveness. So we propose an adaptive data distribution model. This model enables our CVE system to achieve three data distribution modes to deal with several applications requirements and various kinds of network connection. The data distribution mode can be individually chosen for each object according to the function that it fulfills in the virtual environment. Moreover, this data distribution can be dynamically changed during a session to adapt itself to the tasks that users need to perform in the CVE. Finally, measurements of interaction latency and gap in consistency between remote computers show that each data distribution mode has its own performance characteristics. So, adapting the data distribution according to the application requirements improves collaboration and interaction capabilities of CVE users, especially with a WAN.

Future work will focus on developing a syntax to describe rules of data distribution changes for each object. According to the user actions or to the network status, these rules could determine when the data distribution mode of an object must be changed. We would propose extensions to an XML language (such as X3D or Collada) in order to include such rules in the description of the shared virtual world.

Acknowledgements

This work was supported by the French Research National Agency project named Collaviz (ANR-08-COSI-003-01).

References

[CDG*93] CALVIN J., DICKENS A., GAINES B., METZGER P., MILLER D., OWEN D.: "The SIMNET virtual world architecture". In *Proc. of Virtual Reality Annual Int. Symp. - VRAIS* (1993), pp. 450–455.

[DWM06] DELANEY D., WARD T., MCLOONE S.: "On consistency and network latency in distributed interactive applications: A survey – part I". *Presence: Teleoperators and Virtual Environments 15*, 2 (2006), 218–234.

[DZ06] DUVAL T., ZAMMAR C. E.: "A migration mechanism to manage network troubles while interacting within collaborative virtual environments". In *Proc. of Virtual reality continuum and its applications - VRCIA* (2006), pp. 417–420.

[FDGA10] FLEURY C., DUVAL T., GOURANTON V., ARNALDI B.: "Architectures and Mechanisms to efficiently Maintain Consistency in Collaborative Virtual Environments". In *Proc. of Software Engineering and Architectures for Realtime Interactive Systems - SEARIS* (2010), pp. 87–94.

[FS98] FRÉCON E., STENIUS M.: "DIVE : A scaleable network architecture for distributed virtual environments". *Distributed Systems Engineering 5* (1998), 91–100.

[GAW09] GRIMSTEAD I. J., AVIS N. J., WALKER D. W.: "RAVE: the resource-aware visualization environment". *Concurrency and Computation : Pract. & Exper. 21*, 4 (2009), 415–448.

[JFM*08] JOURDAIN S., FOREST J., MOUTON C., NOUAILHAS B., MONIOT G., KOLB F., CHABRIDON S., SIMATIC M., ABID Z., MALLET L.: "ShareX3D, a scientific collaborative 3D viewer over HTTP". In *Proc. of 3D Web Technology - Web3D* (2008), pp. 35–41.

[MAC*02] MARGERY D., ARNALDI B., CHAUFFAUT A., DONIKIAN S., DUVAL T.: "OpenMASK: Multi-Threaded or Modular Animation and Simulation Kernel or Kit : a General Introduction". In *Proc. of Virtual Reality Int. Conf. - VRIC* (2002), pp. 101–110.

[MZ97] MACEDONIA M. R., ZYDA M. J.: "A taxonomy for networked Virtual Env.". *IEEE Multimedia 4*, 1 (1997), 48–56.

[NLSG03] NAEF M., LAMBORAY E., STAADT O., GROSS M.: The blue-c distributed scene graph. In *Proc. of Eurographics Workshop on Virtual environments - EGVE* (2003), pp. 125–133.

[SG93] SHAW C., GREEN M.: "The MR Toolkit Peers Package and Experiment". In *Proc. of the Virtual Reality Annual Int. Symp. - VRAIS* (1993), pp. 463–469.

[SRH03] SCHMALSTIEG D., REITMAYR G., HESINA G.: Distributed applications for collaborative three-dimensional workspaces. *Presence: Teleoperators and Virtual Environments 12*, 1 (2003), 53–68.

[SSP*95] SINGH G., SERRA L., PNG W., WONG A., NG H.: "BrickNet: sharing object behaviors on the Net". In *Proc. of Virtual Reality Annual Int. Symp. - VRAIS* (1995), pp. 19–25.

[YTAK95] YOSHIDA M., TIJERINO Y. A., ABE S., KISHINO F.: "A virtual space teleconferencing system that supports intuitive interaction for creative and cooperative work". In *Proc. of symp. on Interactive 3D graphics - SI3D* (1995), pp. 115–122.