P2P Network for Very Large Virtual Environment

Romain Cavagna
PHD Student
France Telecom RD, IRISA
Rennes, France

romain.cavagna@orange-ft.com

Christian Bouville
Research Expert
France Telecom RD
Rennes, France

christian.bouville@orange-ft.com

Jerome Royan
Research Engineer
France Telecom RD
Rennes, France

jerome.royan@orange-ft.com

ABSTRACT

The ever increasing speed of Internet connections has led to a point where it is actually possible for every end user to seamlessly share data on Internet. Peer-To-Peer (P2P) networks are typical of this evolution. The goal of our paper is to show that server-less P2P networks with self-adaptive assignment techniques can efficiently deal with very large environments such as met in the geovisualization domain. Our method allows adaptative view-dependent visualization thanks to a hierarchical and progressive data structure that describes the environment. In order to assess the global efficiency of this P2P technique, we have implemented a dedicated real time simulator. Experimentation results are presented using a hierarchical LOD model of a very large urban environment.

Categories and Subject Descriptors: C.4, C.2.4, C.2.1 : Computer Systems Organization

General Terms: Algorithms, Measurement, Performance, Experimentation

Keywords: Virtual Environment, Simulation, Peer-To-Peer, Self-Organization, Self-Scalability, Self-Adaptation

1. INTRODUCTION

A centralized architecture is obviously not a good framework to build a truly scalable Virtual Environment (VE). Indeed, basic client server-architecture leads to prohibitive deployment and maintenance costs when it comes to very large scale applications with thousands of connected clients. We first demonstrate that, thanks to its dynamic distribution capability, P2P network overlays have clearly a potential to solve this problem. In the following, after a brief overview of the background work, we describe the peer connectivity method that we propose for visualizing very vast and complex environments such as 3D cities as well as sharing these environments among many users. Then, after briefly describing the specific progressive and hierarchical object structure that we use in our simulation environment,

we propose a new and efficient self-adaptive method to exchange data in a spatially-organized P2P network. Before concluding, we present our simulation results showing then the efficiency of our data exchange method. Finally, we present our plan for future research.

2. BACKGROUND AND RELATED WORK

Much research effort have been directed to shared VEs and many of them focus on the crucial problem of efficiently and quickly updating clients visualization data. A straightforward method to reduce client-server data exchange is to replicate as much as possible the scene data in the local storage resource of clients. To avoid full replication, the scene can be partitioned and the resulting subsets can be distributed among clients. Funkhouser [6] uses Potential Visible Set (PVS) information to selectively send updates of avatars location to a relaying backbone of servers. In this architecture, servers are connected to one another in a P2P manner and clients are connected to servers according to their current location in the VE. Similarly, BrickNet [7, 8] overcomes some of limitations of a central server by incorporating multiple communicating servers. In DIVE [2, 3, 5, 9] that was pioneered by SIMNet [12, 15], the Virtual World is divided into a tree the nodes of which are 3D scene partitions. A multicast group is associated to every node so as to reduce the network usage. DIVE uses a distributed architecture to update objects in real time. In NPSNET-IV [14], the Virtual World is divided into static hexagonal cells associated to a specific multicast group. The server which is assigned to the cell or its administrator has the responsibility to send the update data to connected users. Spline [1] has evolved from pure multicast to mixed client-server and multicast approach so as to cope with low-bandwidth networks. The Virtual World is divided into chunks (the locales) and each one communicates only to the users that are interested in it.

A major impediment to the use the above VE architectures lies in the insufficient deployment of multicast routers in the Internet. In any case, such architectures cannot be considered self-scalable since the overall bandwidth capacity of the system remains constrained by the number of servers. Moreover, they are not designed to deal with many connected clients roaming in very vast and complex scenes. Therefore, we have chosen to concentrate our efforts on the creation of a really scalable and adaptive VE. In a self-scalable architecture, each new peer requests and provides services to other peers. The network can thus spontaneously adapt to the demand by leveraging the resources provided by

each connected peer, which increases the global bandwidth capacity. A key feature of P2P overlays lies their ability to self-organize by allowing each peer to dynamically select the peers it will cooperate with.

3. CONCEPT OVERVIEW

Our peer assignment method is based on the spatial **location** of peers in the VE. This notion is very important because neighbor peers in a VE are likely to have a lot of data in common. For example, if two peers have nearly the same viewpoint in a city model, data stored in both peers will be almost the same. Figure 1 present three screenshots of two cities. As we can see, User 2 and the User 3 have nearly the same viewpoint and their visualization data are almost identical. These peers can exchange their data without the help a potentially overburdened central server.
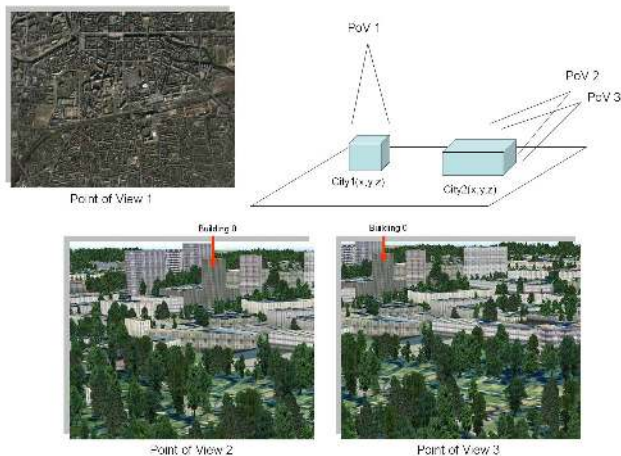

Figure 1: Peers proximity relationship.

However, finding and maintaining the appropriate peer connectivity is a very difficult problem in a changing environment where peers viewpoint are allowed to move freely and peers can disconnect or appear at any time. Solipsis [13] and VON [11] are the first network organization methods that gives a solution for connectivity in 2D environments. In these P2P architectures, peers are connected to each other according to their current 2D location in the VE. Dedicated algorithms are used to achieve the global stability of the P2P network while fulfilling a global connectivity constraint (i.e. there must exist at least one path between each pair of peers). In a 3D VE, 2D connectivity between peers could be a solution if navigation is restricted to walkthrough. However, it clearly does not suit to flyover navigation as visualization data stored in connected peers could be very different. Only 3D connectivity can provide fully-unconstrained peer navigation.

Figure 2 illustrates such a peer connectivity assignment. We can see that there is no relation between locations in the physical network and connections in the P2P overlay. Connections between peers in a spatially-organized P2P overlay can evolve very quickly. Therefore, the network characteristics (i.e. latency and bandwidth) of the neighboring peers could be very different after each connectivity update. Consequently, if we want an optimal exchange of data between
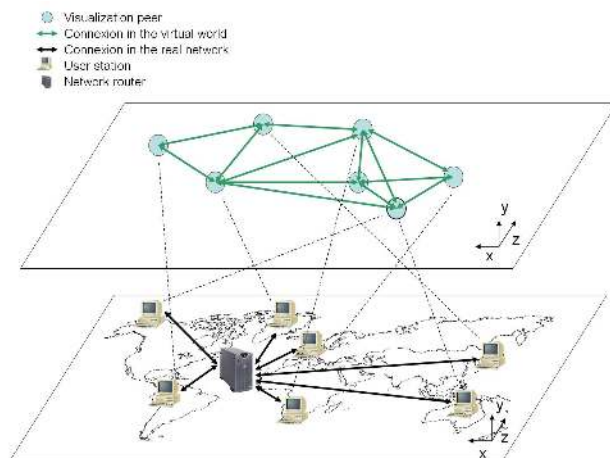

Figure 2: Location of a cluster of peers in the VE and in the physical network. Peers are connected to each other according to their current location in the VE.

peers, the network connection characteristics of peers must be taken into account.

In a recent paper, Shun-Yun Hu [10, 11] presents an intuitive solution for exchanging data between peers in a spatially-organized 2D P2P network. In his P2P network design, the virtual world is described into a specific scene description data structure containing 3D objects location and IDs. The VE is partioned into regular cells and a list of objects is attached to every cell. After receiving an overall scene description (i.e. VE dimension and cells size) from the gateway server, every peer initiates a procedure in order to obtain the required parts of the scene description according to its current Area Of Interest (AOI). After having received the required parts of the scene description, peers can exchange 3D objects.

The exchange procedure described in [10] is quite simple. For every data request, the requesting node prepares a list of potential supplying nodes composed of currently connected neighbors (mostly its AOI neighbors) and the gateway server. The list is sorted according to specific priority criteria (e.g. latency, bandwidth or data availability). The requesting node sends requests to each node of the list in a roundrobin (i.e. sequential) manner. The nodes that receive the requests can respond by either sending back the requested data, or denying service because of data unavailability or lack of bandwidth. If a request is denied, it will be sent again to other supplying nodes. As the gateway server is part of the pool, a data request can always be served.

Our proposal is quite different. First, instead of working at the scene level, we work at the object one as we think that a scene description is too restrictive. When using a scene description, the VE is assumed to be static and it will be obviously difficult to manage real time updates of the VE. We are now working on a specific protocol that enables a dynamic update of the VE. This will be described in a future contribution and this paper focus on the fundamental problems of peers connectivity and data exchange procedure between peers so as to achieve efficient scene data supplying to peers.

By using a P2P spatially-organized network, we address the crucial problem of self-scalability and self-organization. The data exchange method presented below addresses another central problem: the one of **self-adaptation**. Self-adaptation is the ability for a requesting peer to dynamically take into account the serving capacity of neighbor peers. We detail our data exchange method in section 4 and 6.

4. SYSTEM STRUCTURE

4.1 Peer connectivity

In our P2P architecture, there are currently three types of peers. **Memory Peers** (MP) can be likened to central server. They keep in memory all data concerning one or more objects. **Visualization Peers** (VP) are end-user peers. In their local cache, they only store data concerning their current point of view as well as left-over data resulting from previous viewpoint changes. **Connectivity Peers** (CP) are assigned the task of achieving the connectivity between peers. In the future, we think it could be interesting to replace these peers by Solipsis nodes [13] in order to obtain a full P2P network. But so far, they only provide 2D connectivity, which is not sufficient for a 3D VE.
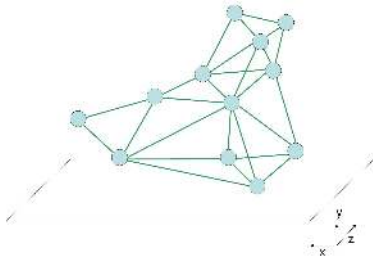

Figure 3: Example of 3D connectivity between peers in a spatially-organized P2P overlay.

Figure 3 represents an example of **3D connectivity** between peers in our spatially-organized P2P overlay. At the present stage of development, the connectivity procedure implemented in CPs does not fulfill the Solipsis constraint mentioned in section 3. Each CP manages all VPs connected to it and, upon a request of a connected VP, it provides a list of the N VP neighbors, N being a parameter of the request. The list of VP neighbors is the **allocated serving pool** (ASP).

Figure 4 represents a VP (VP1) connected to six other neighbor VPs . At time t, VP1 is missing scene data. The goal of our data exchange method is to distribute the VP1 data requests among neighbor VPs. The selection of the serving VPs is made according to the following data that characterizes the serving peer capacity:

- the application **Time To Serve** (TTS),

- the estimated **available data** in the serving peer given its current viewpoint (see section 6),

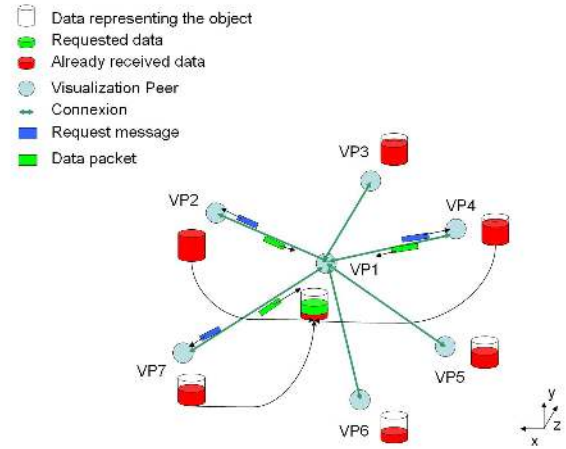- the **upload limit** in bandwidth ($BPUPL$) assigned to this peer for P2P serving tasks,

Figure 4: Requests distribution among serving peers.

- the **number of peers currently served** by the serving peer ($NBPCS$). Used by a requesting peer in conjunction with the upload limit parameter to determine the amount of bandwidth it can expect for its own service (see section 6),

- the **realization ratio** (RR) is the percentage of requests that have been successfully served,

- the **loading ratio** (LR) which is the percentage of data already stored to render the scene under the current viewpoint of the serving peer.

We have not yet implemented the use of the two last features but we think they can improve the self-adaptation behavior. Other interesting characteristics that could be taken into account are the download and processing capacities of the requesting peer. To leverage the upload capacity of the neighbor peers and its own download capacity, each requesting peer must distribute its requests among the selected neighbors. In figure 4, VP1 sends data requests concurrently to three other VPs (i.e. VP2,VP4 and VP7).

4.2 The simulation environment

In order to show the self-adaptation capability of our data exchange method, we have built a dedicated simulator which is able to make real time measurements of the global average accuracy of a cluster of peers. **Accuracy** (ACC) represents the discrepancy at time t between the required data for the current VP viewpoint and the received data that can be used to render this viewpoint. In our case, as we use a hierarchical model for the objects geometry [4], peer accuracy at time t is measured as the proportion of missing nodes ($NMISS$) in the level-of-detail (LOD) trees with respect to the number of required nodes ($NREQ$) to render the current viewpoint:

$$ACC = \frac{NMISS}{NREQ} \tag{1}$$

Figure 5 shows a screenshot of the simulator monitor window. Peers movements and connectivity updates can be visualized in real time during the simulation. For this, the network is modelled by a graph called Simulation Scene-Graph
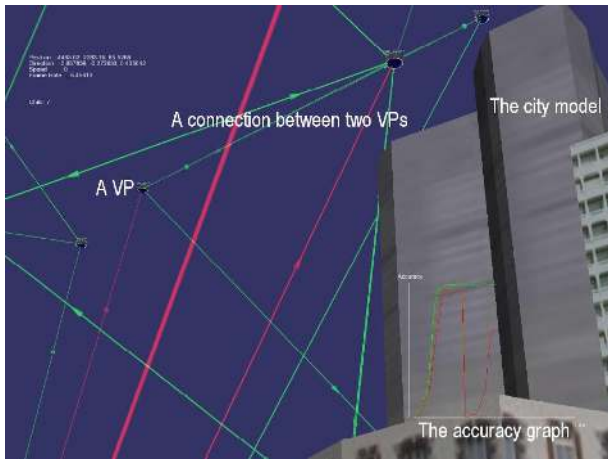
Figure 5: Screenshot of the simulator monitor window during an experimentation.

Figure 6: Example of a progressive hierarchical tree construction of a building.

Figure 7: Example of a LOD description tree construction of a building.

(SSG) in which nodes and branches correspond to peers and connections respectively. Colors are used on peers graphical representation to visualize the accuracy and the status of each peer. In the same way, the color of peers connection changes according to connection activity. The global average accuracy of the peers cluster can be plotted in real time, thanks to a dedicated SSG node that collects accuracy data from all the VPs. The global accuracy is the average of all nodes accuracies. The min and max accuracy among all peers at time t is also plotted so as to assess dispersion.

5. THE HIERARCHICAL LOD OBJECT STRUCTURE

Our experimental platform uses the **progressive and hierarchical representation** for densely built urban areas described in [16]. It is based on a tree data structure (the PBTree or Progressive Building Tree) that holds the recursive applications of geometric merging and simplifications so as to create a hierarchical geometric model of the whole urban environment. Each node of this tree, called a Building Node, describes a 2D1/2 model of a building (i.e. footprint and heights) at a certain level of detail.

Figure 6 shows an example of such a hierarchical construction for a separate building. A complete hierarchical model of a whole city can be built in the same way by allowing close buildings to be merged according to a set of simplification operators. A specific metric error is used to prioritize least error simplifications when building the tree. In order to efficiently manage LOD updates transmission, a LOD description tree (LODDT) is sent at connection startup. Actually, it is a skeleton of the PBTtree as shown in figure 6. It basically replicates the LOD tree structure but each node only holds LOD selection information i.e. parent index, node index, aura and centroid. The spherical aurae geometry is implicitly determined by a scalar giving the metric error introduced by the LOD simplifications.

Figure 7 shows the buildings associated to this tree. The LOD selection information and the dependency relations between nodes are extracted and added into the description tree nodes by traversing the PBTree in breadth-first manner. The LODDT is split into sub-trees to allow progressive
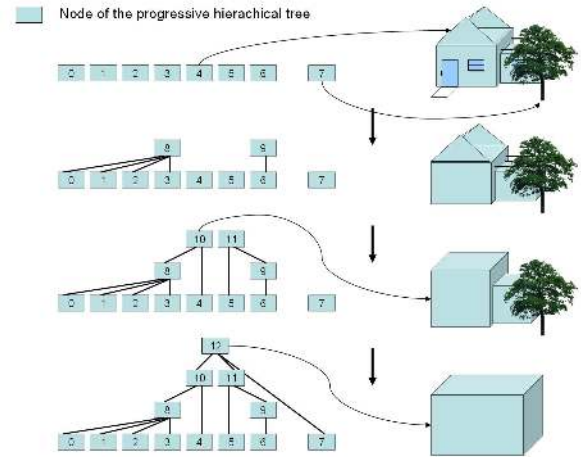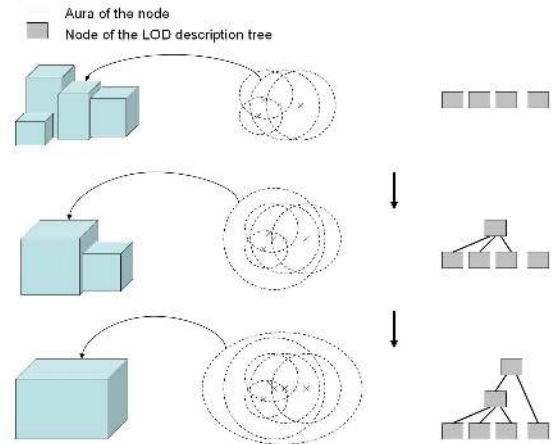
transmission. This functionality is important to deal with very large city models in which the size of the LODDT can be significant. The LODDT splitting is carried out according to size parameters such as the targeted maximum number of building nodes in a packet or the maximum size of a packet. Moreover, all children of a given node must be in the same packet so as to allow immediate use of the node data. The size parameter may be exceeded in the case of large number of children nodes. This is quite rare however with the size parameter we commonly use (roughly 100 nodes per packet).

Figure 8 shows an example of LODDT splitting. The description nodes are ordered in each message so as to allow progressive rebuilding of the LODDT without additional information. Simple aura inclusion tests are used to select the required PBTree nodes. Whenever a new viewpoint is to be rendered, aurae are tested by traversing the LODDT in breadth-first manner in order to build the appropriate visualisation subtree.
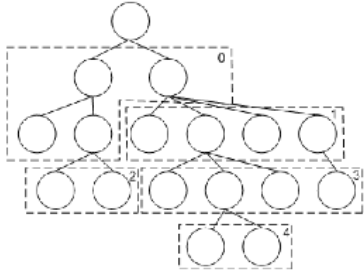
272

Figure 8: Example of LODDT splitting in which the maximum number of nodes per subtree is 4.
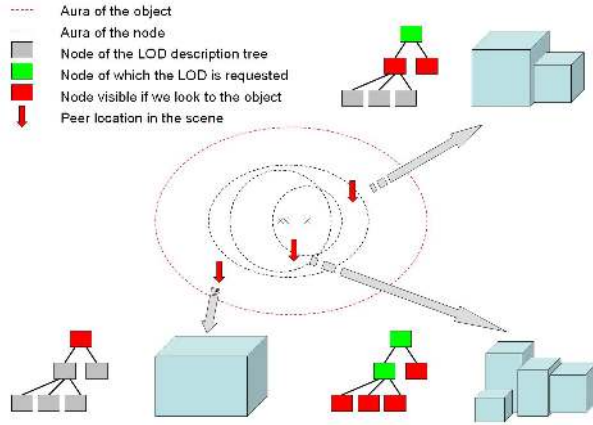

Figure 9: Using LODDT to render different viewpoints.

Figure 9 shows the use of the LODDT to render the scene under different viewpoints. This selection method is a significant improvement compared to the iterative selection method of [16], which forces children nodes to be requested only one level at a time. The LODDT allows direct selection of all necessary geometry nodes in one step as well as additional LODDT subtrees if required. This allows much less network usage and server requests, which is important in the context of P2P overlays.

6. THE DATA EXCHANGE METHOD

In section 4, we have described the criteria that we use for the selection of the serving peers among the neighboring peers. A first important point to take into account is the **data availability** in the caches of neighbor peers. Requesting peers must have a knowledge of the nodes potentially stored in the cache of neighbor peers so as to avoid sending unnecessary request messages. The LODDT presented in section 5 can be used to meet this requirement since knowing the viewpoint of a peer is sufficient to determine its list of visualized objects as well as their current objects visualisation subtrees through LODDT traversal and aurae testing. Therefore, every peer must be informed of the location of their selected neighbor peers in real time. Moreover, the LOD selection parameters of these peers must be known as

well. Indeed the data stored in cache may be very different depending on whether we visualize an object at a high or low level of details.
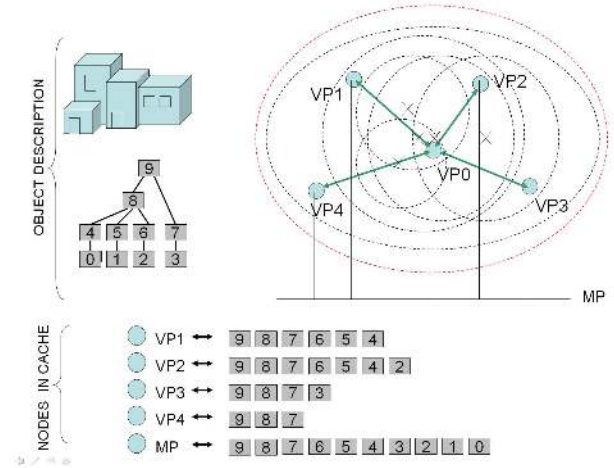

Figure 10: Using LODDT to assess the content of neighbor peers cache.

Figure 10 illustrates the use of the LODDT to assess the content of neighbor peers cache. In this case, VP0 is connected to four other neighbor peers in the VE. Given its current location, VP0 needs all data corresponding to the object. For every required nodes, VP0 tests the corresponding aurae with the position of neighbor peers. The aura inclusion test is made with the current LOD selection parameters of the considered peer. This way, requesting peers can easily select potential serving peers with high probability of obtaining the requested data. Another important challenge for every peer is to efficiently leverage its download capability given the upload capacity of its selected serving peers.

If a peer has a knowledge of the network traffic brought about by its requests, it can limit itself. Moreover, if a peer has some information about the network activity and capability of its neighbor peers, it will be able to prevent them from being over-burdened. In our network architecture, every VP adopts a strategy of self-regulation aiming at a clever usage of the serving capacity of its neighbor peers, i.e. VPs should not request more data than can be expected given the serving peers resources. In the same way, a VP should not request more data than it can receive. In our P2P overlay, the user has to decide on his own the amount of bandwidth that he is ready to grant to P2P overlay serving activity. This bandwidth is the $BPUPL$ parameter mentioned in section 4. Using the current $NBPCS$ value of a serving peer, a requesting peer can assess the usable bandwidth (UPB) that he can expect from this serving peer. Assuming equidistribution of bandwidth among VPs, we have:

$$UBP = \frac{BPUPL}{NBPCS} \qquad (2)$$

If all peers follow the same rule, they can control their generated traffic in order to distribute it adaptively among serving peers according to their upload capacity.

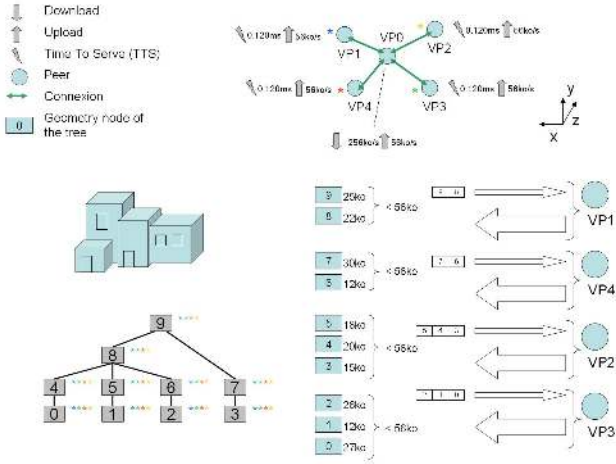Figure 11 shows an example of an ideal distribution of

273

Figure 11: Example of adaptive distribution of data requests from a VP.

Figure 12: Example of P2P messages exchange.

the download traffic generated by a VP. In this very simple example, all potential serving peers of VP0 (VP1-4) are assumed to have in cache a copy of the complete object model. We also assume that VP0 is the only peer that is requesting data. Therefore, it can use the full upload capability of its ASP. All peers in the VE have an upload capacity of 56 Kb/s and a download capacity of 256 Kb/s. If all serving peers have the same TTS value, the full download capacity of VP0 is exploited and the system response can be considered as optimal. In general, the values of TTS, $NBPCS$ and $BPUPL$ of each serving peers are different and time-dependent, and the simple equidistribution solution does not lead to optimal usage of the download capacity of VPs. Besides, the serving capacity of the ASP may not be fully used since some VPs may have a download bandwidth lower than the sum of all UBP values of the ASP. Finding a solution closer to optimality is still a research issue. In practical, it is reasonable to think that fixing a download bandwidth equal to the upload bandwidth for all VPs leads to a fair distribution of the P2P overlay resources among the VPs.

Figure 12 illustrates the protocol that we use to obtain a **self-adaptation** behavior in our P2P overlay. Every VP samples the TTS of its ASP peers at regular time interval. This information is used by VPs to organize their data requests so as obtain PBTree nodes in the right order, i.e. a parent node must be received before its children (see section 5). Consequently, upper level nodes are requested to the serving peer that has the least TTS.

In this example, VP1 applies the above-mentioned equidistribution strategy, i.e. the upload capacity of a serving peer is equally distributed among its client VPs. Note however that VP1 sends its first data request without knowing $NBPCS$, a default value is taken in this case.

The variability of the data stored in the serving peers caches is not yet taken into account in our implementation. The RR and LR parameters described in section 4 can be used to achieve this. For example, a peer can be considered as an interesting source, but the fact that it may be recently connected to the P2P overlay cannot be taken into account if the requesting peer does not analyze the realization ratio
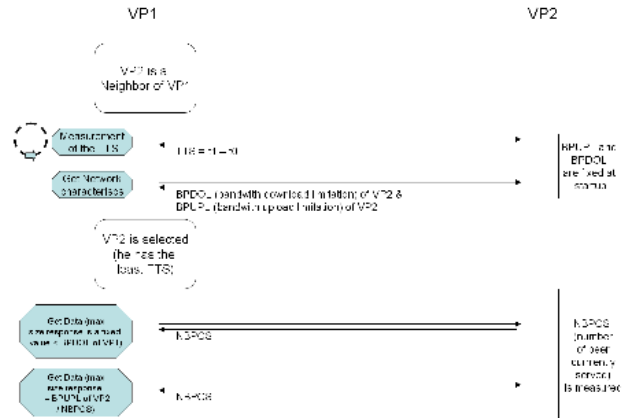
(RR) first. The realization ratio can be taken into account after receiving the first data packet from the serving peer. Then, the LR value of serving peer that is transmitted in every data packet can be taken into account too. With this two parameters, every VP can improve its chance of getting successfully-served requests.

7. EXPERIMENTATION

The experimentation has been carried out with a cluster of twenty peers that cooperate in a given area of the VE. Each VP loops on the path that has been assigned to it. VPs are also assigned a start-up time and a lifetime, i.e. the time at which the peer is connected to P2P overlay and the time during which it stays connected. During its lifetime, a VP is connected to three serving peers and to the MP which is located in the middle of the VE. In order to bring out the behavior of our P2P network overlay, we put stress on it by limiting the bandwidth of every peer (MP included) to 56 Kb/s for both upload and download bandwidth.

7.1 The backlash against flash-crowds

We implement a very simple test scenario with intent to assess the self-scalability and self-organization behavior of our P2P overlay. The test scenario begins with 5 VPs that start to navigate in the VE. During this first stage, these VPs request data from the MP only. It is maintained during 100s so as too give enough time for the VPs to collect data and reach a global accuracy of one. Then, at t=100s, 15 new peers connect to the overlay and start to navigate. Four different methods of serving peers selection have been tested:

- from the MP only (amount to basic client/server),

- from the peer having the least TTS,

- from the closest peer,

- using our self-adaptive distribution method.

The above four methods have been experimented and in each case, we assess the performance of the network by measuring the global accuracy.
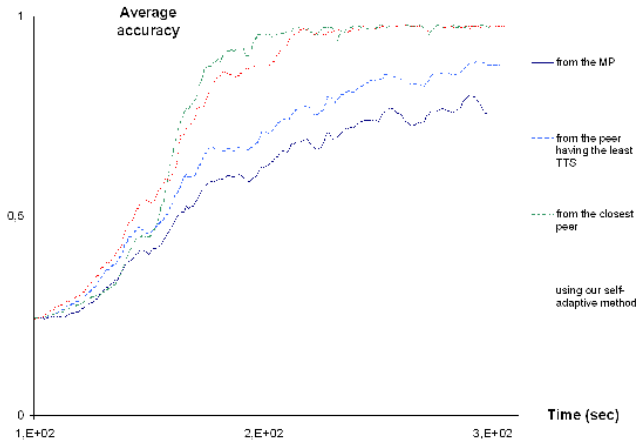
Figure 13: Measured global accuracy with different selection method.

Figure 14: Bandwidth amplification effect.

Figure 13 shows the result of our experimentation. Due to the strong connection bandwidth limitations, the MP is clearly overburdened if the fifteen new peers request data only from the MP. This experiment clearly shows that the closest peer and self-adaptive methods give the best results. The least TTS method provides inferior results than the closest peer method because the selected serving peer may be too far from the requesting peer to have the requested data. With our self-adaptive distribution method, the overlay reacts faster to the flash-crowd and provides the best accuracy during the start phase. At the end of the start phase, we see that our method performs slightly worse than the closest peer one. This is due to excessive requests re-sending because we have not yet implemented the use of the loading ratio and the realization ratio. Some requests may have been transmitted many times to the same serving peer whereas it has not yet loaded the requested data. Indeed we are sure it is possible to correct this flaw.

7.2 The bandwidth amplification effect

In a case of flash-crowds, it is possible to assess the bandwidth gain that we obtain through the use of our self-adaptive distribution method. For this, we increase the bandwidth of the MP in the basic client-server architecture in order to approximately reach the same average accuracy curve as the one obtained with the self-adaptive distribution method. Figure 14 shows the result of our experimentation. The experimental conditions are the same as described in subsection 7.1.

We can see that the gain in bandwidth due to the P2P overlay is 34 Kb/s, i.e. 65% of the central server connection bandwidth in a basic client server architecture. Of course, the overall bandwidth gain in a real system is much higher since there are many such clusters that cooperate concurrently in different areas of the VE.

7.3 The self-adaptation capability

In this part, we focus on the self-adaptation and self-organization behaviors brought about by our serving peers selection method and data exchange procedure. At $t=0$, 5 peers start to navigate requesting data only from the MP.

At $t=100$s, fifteen new peers connect to the VE at different times. The experiment consists into testing the P2P overlay behavior by increasingly varying the arrival rate of the fifteen new peers. For each tested arrival rate, we measure the global average accuracy of all the VPs over a time interval of 300s. Figure 15 summarizes the results that we have obtained.
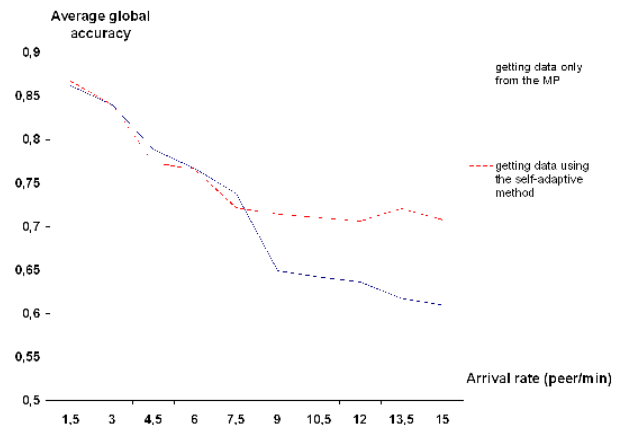

Figure 15: Global average accuracy of a cluster of fifteen peers with various arrival rates.

In figure 15, we see that with low arrival rates, both methods roughly provide the same result. The self-adaptive distribution method begins to provide better results at arrival rates exceeding 6 peers/min. Beyond this rate, the MP alone becomes over-burdened and cannot cope with the demand. Figure 15 shows that the global average accuracy of the MP method is continuously decreasing whereas it stabilizes at a level of 0.7 with the P2P overlay. This clearly demonstrates the self-adaptation capability of our P2P overlay.

8. FUTURE WORKS

As mentioned above, our results have been obtained with

275

a first simulator version that does not implement all the self-adaptation mechanisms that have been described in this paper. In particular, the serving peers selection method does not take into account the realization and loading ratio discussed in section 4. We hope to further improve the performances of our P2P overlay by using these variables. We are also working on other type of experiments with much larger number of peers so as to more completely assess the performances and deeply understand the behavior of the system. So far, we have assumed that only one peer (the MP) holds the complete scene data. Our idea for future works is to introduce special-purpose serving peers that will store some parts of the scene depending on VPs localization in the VE. These special peers would help the system to react faster to local overcrowding of VPs. Dealing with dynamic worlds is also in our plans at longer term.

9. CONCLUSION

Many research works have been focussed on network sharing of large and complex VEs, but little work address the problem of self-scalability when many clients are connected to the VE. Most of the proposed solutions cannot cope with the problem of efficiently supplying scene data to many fast moving clients. With self-scalabity, peers cooperate together to relieve the central server(s) of this burden. Each peer grants some ot is upload bandwidth to serving tasks. In this paper, we propose a new spatially-organized P2P network overlay with intent to obtain a good self-scability behavior even in the case of fast changing environments. Our solution is based firstly on a serving peers selection method that not only take into account the peers proximity in the VE but also some parameters and state variables that allow to assess the current serving capacity of the neighbor peers. Using these data, a requesting peer is able to dynamically select its serving peers with the best chance of being successfully served. Secondly, requesting peers follows a self-limitation strategy aiming at a fair distribution of the serving resources among peers.

Using hierarchical geometric models is vital to cope with very large and complex scenes visualisation. To allow the use of the appropriate adaptive visualisation mechanisms, we have introduced a dedicated data structure that describes the hierarchical LOD model of objects. It allows determining whether a required LOD node is contained in the cache of a neighbor node as well as the required bandwidth to download the node data. Our implementation is based on the PBTree structure which provides a hierarchical LOD model for very large urban environments. Note however that this solution can be extended to any hierarchical object model such as multi-resolution pictures or videos.

A simulator has been implemented in order to assess the behavior of our P2P overlay. Using this simulator, we have shown that a simple serving peer selection method based on proximity already provides good results. Not all self-adaptation mechanisms are yet implemented in our current platform but our first results already show up a graceful self-scalable behavior especially when dealing with high peers arrival rates.

10. ACKNOWLEDGMENTS

11. REFERENCES

[1] J. Barrus, R. Waters, and D. Anderson. Supporting large multiuser virtual environments. *IEEE Computer Graphics and Applications*, pages 50–57, 1996.

[2] C. Carlsson and O. Hagsand. Dive: A multi-user virtual reality system. *Proceedings of the IEEE VIrtual Reality Annual International Symposium*, 17(3):194–400, 1993.

[3] C. Carlsson and O. Hagsand. Dive: A platform for multi-user virtual environments. *Computers and Graphics*, 17(6), 1993.

[4] J. H. Clark. Hierarchical geometric models for visible surface algorithms. *Commun. ACM*, 19(10):547–554, 1976.

[5] E. Frecon and M. Stenius. Dive: A scaleable network architecture for distributed virtual environments. *Distributed Systems Engineering Journal (DSEJ)*, 5:91–100, 1998.

[6] T. A. Funkhouser. Ring: A client-server system for multi-user virtual environments. *Symposium on Interactive 3D Graphics*, 1995.

[7] S. G., S. L., P. W., and N. H. Bricknet: Sharing object behaviors on the net. *Proceedings of the IEEE Virtual Reality Annual International Symposium*.

[8] S. G., S. L., P. W., and N. H. Bricknet: A software toolkit for network-based virtual worlds. *Teleoperators and Virtual Environments*, 3(1):19–34, 1994.

[9] O. Hagsand. Dive: Interactive multiuser ves in the dive system. *Proceedings of the IEEE Multimedia Magazine*, 3(1), 1996.

[10] S.-Y. Hu. A case for 3d streaming on peer-to-peer networks. *Web3D 2006 Symposium proceedings*, 2006.

[11] S.-Y. Hu, J.-F. Chen, and T.-H. Che. Von: A scalable peer-to-peer network for virtual environments. *to appear in IEEE Network*, 2006.

[12] C. J., D. A., G. B., M. P., M. D., and O. D. The simnet virtual world architecture. *Proceedings of the IEEE VIrtual Reality Annual Symposium*, pages 450–455, 1993.

[13] J. Keller and G. Simon. Toward a peer-to-peer shared virtual reality. *IEEE Workshop on Ressource Sharing In Massively Distributed System*, 2002.

[14] M. R. Macedonia, D. P. Brutzman, M. J. Zyda, D. R. Pratt, P. T. Barham, J. Falby, and J. Locke. Npsnet: A multiplayer 3d virtual environment over the internet. *Proceedings of the 1995 Symposium on Interactive 3D Graphics*, 1994.

[15] D. Miller and J. A. Thorpe. Simnet: The advent of simulator networking. *Proceedings of the IEEE*, Aug. 1995.

[16] J. Royan, C. Bouville, and P. Gioia. Pbtree: A new progressive and hierarchical representation for network-based navigation in urban environments. *VMV*, 2003.