JaDE: A JXTA Support for Distributed Virtual Environments

Laura Ricci
Dipartimento di Informatica
Universitá degli Studi di Pisa
Largo Bruno Pontecorvo, 3 Pisa, Italy
Email: ricci@di.unipi.it

Luca Genovali
IMT
Institutions, Market, Technologies
Piazza S.Ponziano, 6 - Lucca, Italy
Email: l.genovali@imtlucca.it

***Abstract*—This paper introduces *JaDE*, a P2P support for the development of *Distributed Virtual Environments* that improves *DVE* scalability through the notion of *Area of Interest*. *JaDE* defines a set of protocols to support both the active entities and passive objects of the *DVE*. The state of passive objects is replicated on a set of peers to increase the reliability and the responsiveness of the application. Since passive objects may be concurrently updated by the active entities of the *DVE*, a novel consistency protocol is defined together with a set of mechanisms to guarantee the persistence of *passive objects* in a *DVE* environment. The paper presents an implementation of *JaDE* which exploits the *JXTA* distributed platform and shows how the *JaDE* functionalities may be supported by *JXTA* protocols. A set of preliminary experimental results are discussed.**

I. Introduction

Distributed Virtual Environments (DVE) such as military or civil protection distributed simulations and massively multi-player online games (MMORG), for instance *World of Warcraft* or *Second Life*, currently represent a class of challenging distributed application, because they integrate graphics, network and AI programming. In each of these applications, a set of *active entities*, represented by *avatars*, interact with each other and with a set of *passive objects* like weapons, potions, etc. Each avatar is generally aware of both other avatars and passive objects in its *Area of Interest, AOI,* in general a circular or rectangular area surronding the avatar.

Even if these applications may fully exploit the high scalability of *P2P* model, most of them still follows the client server model because of the complexity of defining a fully distributed *DVE* platform. On the other hand, the definition of a *P2P* support for *DVE* is currently an active research area. To improve the scalability of the *DVE*, most of the approaches recently proposed [7], [8], [9], [6] exploit the concept of *AOI* [11], [10] so that an avatar receives only notifications of events occurring in its *AOI*, for instance movements of other players or passive objects updates. This results in a lower number of messages as well as in a lower number of objects managed by each peer. On the other way, poses novel problems because, as an example, each avatar must be able to dynamically acquire the objects as it moves within the *DVE*. Furthermore, a set of mechanisms to guarantee the persistence of the objects of the inhabited regions of the *DVE* have to be defined.

Another challenging issue is the management of passive objects in a fully distributed environment. A solution based on the replication of the state of the objects onto a set of peer improves the scalability of the *DVE*, but requires the definition of proper mechanisms to guarantee the consistency of the objects in spite of the concurrently updates of the avatars. This paper will show that the *JXTA* platform [2] is a suitable support to implement all these mechanisms.

JXTA is a distributed platform for the development of *P2P* applications which defines a stack of protocols offering different services and functionalities. *JXTA* peers may publish resources/services and retrieve them within a *group*. The group abstraction makes it possible to define peers characterized by "common interests" in order to bound the search of resources/services.

This paper presents *JaDE, JXTa Distributed Environment*, a support for the development of *DVE* which exploits *JXTA* to define a fully distributed and scalable support. *JaDE* defines the *AOI* of peers through *JXTA groups*. Furthermore, *JXTA* discovery mechanisms may be exploited to enable avatars to dynamically discover new avatars and passive objects as they move within the *DVE*. Alternative *JXTA* protocols have been evaluated in order to test their efficiency for interactive applications requiring an high responsiveness.

The paper is organized as follows. Sect. II reviews the current literature on $P2P$ based *DVE* environments. Sect. III describes the general architecture of *JaDE* protocols, while *JXTA protocols* are described in Sect. IV. The implementation of *JaDE protocols* through *JXTA* is discussed in Sect.V. Sect. VI presents some experimental results, while Sect.VII present some conclusion and describes the future work.

II. Related Works

This section reviews some projects of *DVE* based upon $P2P$ overlay topology.

Solipsis, [7], is a massively shared virtual reality system based on a network of peers. It does not rely on any server nor on IP multicast, and its goal is to scale to an unbounded number of participants and accessibility by any computer connected to the Internet. Each peer implements the entities of the virtual world and "perceives" its surroundings. Connected

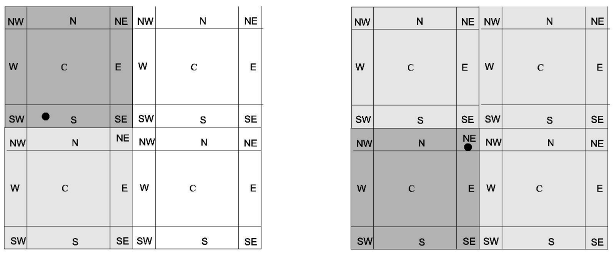

Fig. 1. Definition of Area of Interests

peers may exchange data such as video, audio, avatars movements or any kind of events affecting the representation of the virtual world.

[3] proposes a *Voronoi-based Overlay Network (VON)* where each peer sets up dynamic connections with its *Voronoi* neighbours whenever it moves within the *DVE*. This approach exploits *Dynamic Area of Interests* to define dynamic connections. Since the *AOI* of a peer may be crowded, a set of optimizations are applied to reduce the number of connections with neighbouring peers. The approach defined in [4], [5], [6] exploits a publish/subscribe computational model for the interactions among peers.

Mopar[8] is a fully distributed $P2P$ infrastructure supporting *DVE* applications. It defines an overlay network using both a Pastry based DHT and an hybrid P2P architecture and it decomposes the virtual environment into hexagons(the cells), which assures a continuous view to all the participants in spite of their discrete nature.

III. JaDE: Protocol Specification

This section presents a high level specification of the *JaDE* protocols. Their *JXTA* implementation will be discussed in Sect.V. In the following we assume that each peer controls a single avatar. Hence, the terms "avatar" and "peer" will be used in a interchangeable way.

A basic choice for the definition of *JaDE* protocols is the adoption of the concept of *Area of Interest* to improve the *DVE* scalability.

Furthermore *JaDE* statically partitions the *DVE* into a set of *regions* whose shape and extension depend upon the characteristics of the *DVE*. We assume that a peer P located in a region R of the DVE renders, at any instant of time, the events occurring in R only. For this reason, at any instant of time, the *AOI* of P includes at least R. Each peer periodically

- sends its current positions to the any other one in its *AOI* through an *heartbeat message*
- receives the position of other avatars in its *AOI*
- updates its local view of the *DVE* through the received messages.

For the sake of simplicity, this paper assumes a decomposition of the *DVE* into a set of equal, square regions as those shown in Fig. 1. Furthermore, the *main region* of a peer is the *DVE* region where a peer is located.

Since a peer dynamically moves within the *DVE*, its *AOI* may change and anytime it enters a new region R, it must be initialized with the state of each passive and active entity located in R. Then, as far as the peer stays in R, it is interested in any event occurring in its *AOI*, such as the update to the position of any other peer or to the state of a passive object in R. It is worth noticing that, since in a *WAN* the latency of any notification mechanism cannot be neglected, a *prefetching mechanism* is required to avoid that a peer perceives a delay in the acquisition of the state of R. For this reason, each region of the *DVE* is further partitioned into a *central zone C* and eight *peripheral zones*, as shown in Fig. 1. When the peer P stays in C, its *AOI* overlaps its main region. When the peer approaches the border of its main region and enters a peripheral zone, it starts prefetching the state of the entities of the region R it is going to enter. In this way, it initializes its *AOI* before entering R. Note that any peer in R must promptly detect the presence of P as well.

A straightforward implementation of this prefetching mechanism just requires the extension of the *AOI* of a peer when it enters a peripheral zone. For instance, the *AOI* of the peer displayed by the black circle in the left part of Fig. 1 includes its dark grey main region R and the southern light grey neighbour region of R. For the same reason, the *AOI* of the peer located in the north-eastern peripheral region in the right part of the figure 1 overlaps the whole *DVE*. To avoid that peers belonging to the new region perceive a delay in detecting a new peer entering their main region, the entering peer should notify its presence to any peer in its extended *AOI* as well.

To reduce the number of events that are prefetched from a neighbouring region before entering it, the size of the enlarged *AOI* may be reduced. For instance, a peer entering a new region may be interested in initially perceiving the state of the entities close to the border of the new region while acquiring the knowledge of the state of the whole region later. As shown in Sect.V *JXTA* can efficiently support both solutions.

A challenging issue in the definition of *JaDE* is the *consistency model* to implement the *DVE* passive objects and the *consistency protocol* supporting this model.

JaDE adopts the *Sequential Consistency* model which guarantees that, while any interleaving of the updates to passive objects updates may be accepted, all the peers observe the same interleaving of the updates of the objects in their *AOI*. *Sequential Consistency* may be easily implemented in client server architectures where a central server manages the state of any passive object forwarding them to any interested client, while its implementation in a $P2P$ environment is more complex.

A similar solution may be adopted in a $P2P$ environment as well by the *dynamic election* of one of the peers of a region R to manage the state of the passive objects. Even if this solution is more scalable, because it distributes the load among a set of servers, the election of a single server for each region still introduces a bottleneck resulting in both a lower *DVE* responsiveness and a lower reliability.

On the other hand, a fully distributed solution which *repli-*

cates the state of any passive object to each peer of a region may be adopted. Here, each peer holds a *local copy* of the objects of its *AOI* and updates their state by accessing its copy. This approach increases the reliability of the *DVE* because the crash or the voluntary departure of a peer does not imply the lost of the objects of a region. Also the *DVE* responsiveness improves because concurrent updates are possible. However, a mechanism to preserve the consistency of replicated copies in spite of concurrent updates has to be adopted. Note that this situation often occurs in a *DVE* because an object may "attract" peers so that a typical *DVE* scenario is a crowd of peers that try to modify the same object.

JaDE exploits object replication to improve responsiveness and reliability. Each peer stores in a local cache, the *Object Cache*, the state of any object in its *AOI*. The cache is initialized when the peer enters a region and flushed when it leaves the region in order to avoid to overwhelm the cache with a large amount of useless information.

As far as concerns object consistency, several approaches have been proposed [12] to guarantee the consistency of multiple replicated copies in the presence of concurrent updates. It is worth noticing that while a process of a concurrent application is not generally aware of the other processes which may concurrently update a copy of data, in a *DVE* each peer is always aware, because of the heartbeats notifications, of the positions of the other peers of its main region and it may detect when a crow of peers gathers around a shared object. *Jade* exploits this property to optimize the consistency protocol. The full protocol will be described in Sect. III-A.

Another important issue in the definition a $P2P$ support is the definition of a set of mechanisms to guarantee the *persistence* of the objects belonging to regions which are not inhabited by any peer. A region may be inhabited because either no peer has still visited it or every peer has left it.

JaDE assumes that each peer holds a map of the whole *DVE* that includes static objects, like landscapes, trees and other graphical elements. Some of these objects may be modified by the player, while others are immutable. We are interested in objects which may be modified dynamically. If the peer modifies some object, the object is *activated*, i.e. a data structure is allocated by *JaDE* to store its state. This state is replicated to the peers of that region and, to avoid that the state is lost if all them exit the region, the last peer leaving the region stores the state of any object in a *Backup Cache*. A more complex situation is that where the last peer of the region leaves the *DVE*. If its departure is voluntary, this peer may choose another one in the *DVE* and send to this peer the content of its local cache. The choosen peer stores these objects in its *Backup Cache*. In both cases, the peer holding the objects of the region becomes the *Backup Peer* of the objects. Note that a set of *Backup Peers* may be defined to take into account abrupt peer crashes.

It is worth noticing that the identity of the *Backup Peer* must be notified to all the peers in the *DVE* because they need to find out the objects when they enters the region. As a matter of fact, when a peer P enters a region it first checks if it is inhabited by any peer. In this case, it chooses at random a peer in the region and asks it for the region objects, otherwise P must contact the *Backup Peer*. Sect. V will show that this global notification may be easily implemented through *JXTA groups* and *pipes*.

A. JaDE: Passive Object Consistency

JaDE consistency protocols exploit the relative positions of the peers and the knowledge of the maximum latency of the underlying notification mechanism to detect scenarios where replicated objects may be concurrently updated.

It is worth noticing that in a *DVE* each peer may update an object only if it is close to it. For instance, a peer should be close to a magic potion to drink it. If the peer is far from the potion, it may throw a stone to break the bottle containing the potion. In any case, for each object O, we can define an *Update Area*, that is the portion of the *DVE* region where a peer must be located in order to modify O. This area is different for different kind of objects. In *JaDE*, it is a circle centered at the object location.

The updates performed by a set S of peer in the *Update Area* of an object O should be considered as concurrent ones because it is likely that a peer in S modifies O before receiving a previous update of O by another peer in S. On the other hand, we cannot neglect that, if the latency of the underlying notification mechanism is high, even a peer located outside the *Update Area* of O may enter it and update O before receiving the updated value of O.

For this reason, *JaDE* defines the *Conflict Area* of each object as a larger, circular area centered at the object. The radius of the *Conflict Area* of an object O should be defined so that the time interval required to reach the *Update Area* of O from any point in its *Conflict Area* is smaller than the interval of time required to notify an update to O to any peer of the region. The radius of the *Conflict Area* depends upon both the *larger peer speed* and the *maximum latency* of the mechanism to notify updates. *JaDE* assumes that the *Update Area* of an object is always included in its *Conflict Area*.

When a peer P in the *Update Area* of an object O modifies it, P checks if the *Conflict Area* does not include any other peer and, in this case

- reads the current state of O from the local cache, because its copy of O is up to date
- notifies the update to any peer in its main region, since they will receive the update before entering the *Update Area*.

On the other hand, if P finds out at least another peer in the *Conflict Area*, it should exploit a mechanism to guarantee the consistency of O in presence of potential concurrent updates. Different approaches have been proposed [12].

Totally-ordered Multicast based on Lamport's timestamps [13], may be exploited to guarantee that each peer orders concurrent updates in the same way since the update messages are delivered in the same order to each peer. However, the implementation of this mechanism requires a high amount of messages, that results in a low scalability and prevents its adoption in a large distributed systems.

JaDE solution is based upon the *distributed definition* of a *coordinator* for each object of a region R. In *JaDE* each peer may *create* a new object, which is not present on the static map of the *DVE*, or it may *activate* an object, if the object is initially present on the static map, but it has not been modified yet by any peer. The peer which creates or activates an object O becomes the *coordinator* of O. As soon as it is elected, the coordinator informs all the peer in its main region of the election and detains the coordination of the object as long as it remains within the region. The notification of the coordination acquisition may be easily supported by *JXTA* publish service. When the coordinator leaves R it passes the coordination to another peer in R.

The coordinator is the unique *owner* of the object O, it holds the up to date state of O and it serializes the updates when concurrent updates to O occurs. When a peer updates an object whose *Conflict Area* R is not empty, it sends the update to the *coordinator* which resolves the update conflicts among the peers in the *Conflict Area*. Then the coordinator sends the state of the updated object to any peer of its main region. *JaDE* exploits a timestamp based mechanism [1] to resolve the conflict among peers trying to activate the same object concurrently so competing for the acquisition of the coordination of the object.

IV. JXTA: the Protocols

This section briefly introduces the *JXTA* [2] protocols which have been exploited to implement the *JaDE* protocols. *JXTA* is a distributed platform for the development of $P2P$ applications based on the definition of a stack of *protocols*. A *JXTA* application includes a set of peers which can publish and discover *resources* and *services* and communicate through *logical communication channels*. *JXTA* peers may be classified as *edge*, *rendez vous* or *relay* peers. *Edge peers* have transient, low bandwidth network connectivity and usually reside on the border of the Internet, hidden behind firewalls or accessing the network through non-dedicated connections. In general, a *Rendezvous peer* is characterized by enough computational and storage capacity to *index* and *discover* resources and services published in the *JXTA network*. A *Relay Peer* enables the peers behind *firewalls* or *NAT* systems to take part in the *JXTA* network. This is implemented through a protocol such as *HTTP* which can traverse the firewall. Peers in different subnets of the *JXTA network* should be connected to at least one *Rendez-vous Peer* in order to interact with each other. Peers of a *JXTA* application may dynamically cluster by joining a *JXTA* group. A group is an application level concept which allows to bound the extent of resource and service discovering and message broadcasting. A default group, i.e. the *NetPeerGroup*, includes any peer belonging to a *JXTA* application.

The *JXTA Discovery Protocol* supports the publication and discovery of different kind of resources in the *JXTA* network. As an example, a peer may publish a service, a communication channel or any information it want to share with other peers. Each resource published by a peer is described by an *advertisement*, i.e. an *XML document* describing the resource. The *Discovery Protocol* defines a publish service supporting the notification of the advertisement indexes to the *rendez-vous peers*. The *Shared Resource Distributed Index, SRDI* [2] is an indexing and discovery service for advertisements based on a *Loosely Consistent Distributed Hash Table*. When a peer P sends a discovery requests to a rendez-vous peer R, it exploits the *SRDI* service to look for advertisements matching the request. The advertisements which has been discovered are sent to P which stores them in its *Local Cache*.

The *Pipe Binding Protocol* supports the definition of *virtual communication channels* which support a direct interaction among peers. A *JXTA Pipe* is a virtual communication channel which defines a connection between a sending end-point and one or more receiving endpoint. The default *JXTA* service pipe supports *unidirectional, asynchronous, unreliable communication*, but other types of pipes are currently offered by the *JXTA* distributions. Two modes of communication are supported, i.e. *point to point* communication, connecting exactly two pipe endpoints and *propagate pipes* connecting one output endpoint to multiple input pipes. *JXTA Propagate pipes* connect an output pipe with several input pipes. Messages are propagated from the output pipe to the input pipes within the same peergroup. Many-to-many communication paradigm can be exploited by *JXTA* as well.

A peer p_1 sends messages to p_2 through a shared pipe p if it connects to the input endpoint of p, while p_2 connects to one output point of p. The pipe may have been published by one of the two peers and then discovered by the other one through advertisement discovery. Note that pipes are uniquely identified by pipe advertisements as any other shared resource.

The *Resolver Protocol* is a basic protocol supporting higher level protocols such as the *Discovery* one. It allows the propagation of *generic queries* within a group of the *JXTA* network. The main characteristics of the *Resolver Protocol* are:

- queries are defined as XML messages.
- it supports applications based on *request-response* computational paradigm and *broadcast-and-listen* communication pattern

V. JaDE: the JXTA support

This section shows that *JXTA* is a proper support to implement the *JaDE* protocols defined in section III.

A basic choice of *JaDE* is to model the *AOI* of the peers in terms of *JXTA groups*. To automatically restrict the notification of the events to the *AOI* of each peer, each region of the *DVE* is statically paired with a distinct *JXTA group*. In this way, the notification of the events is automatically restricted to the *AOI* of each peer. Different kind of *JXTA* protocols are then chosen to support the notification of different kind of events.

Since the most frequent event of a DVE is the peer movement, an efficient protocol has to be chosen to notify the heartbeats. We have experimented two alternative solutions exploiting, respectively, *JXTA propagate pipes* and *JXTA resolver protocol*. The first one pairs a *propagate pipe* with each region R of the DVE by creating the pipe within the group associated with R. It is worth noticing that the many-to-many paradigm of propagate pipes is suitable for heartbeats exchange and that the *Propagate Pipe Service* can be exploited even if it is an *unreliable* service because an heartbeat is a transient event whose loss can be tolerated since it is repetitively notified.

A pipe advertisement for a region R is published within the group associated to R by the first peer entering into R. From this moment, any peer entering R can discover the pipe advertisement through the *JXTA Discovery Protocol* and it binds the pipe both in input and output. As long as the peer moves within R, it exchanges with other peers through the same pipe an heartbeat every 200 milliseconds. The heartbeat describes the position of the peer. The heartbeats sent by other peers are received through the same pipe.

According to the *prefetching mechanism* discussed in Sect.III a peer entering a peripheral region joins the *JXTA* group associated with the neighbours regions and discovers the propagate pipe associated with that group. Then, it exchanges the heartbeats through the discovered pipe, thus implementing the prefetching of the peers in the neighbour regions.

The implementation of more sophisticated prefetching strategies implies the definition of a larger set of pipes for each group. For instance distinct pipes may be associated with the central and the peripheral zones of each region. In this way, the peer can prefetch the positions of peers at the border of the neighbour region only.

To compare the efficiency of different *JXTA protocols* we have exploited the *Resolver Protocol* to implement heartbeat notification. In this case the notification is sent as *XML query*. We evaluate the implementation of these services in Sect. VI.

JaDE exploits *JXTA caches* to store the state of the replicated objects. When a peer creates or activates a new object O, it *publishes* an advertisement describing O. The advertisement is published again when a peer modifies O. Each advertisement is published within the *JXTA* group associated to the region where O is created. *Object prefetching* is analogous to *heartbeat prefetching* and is based upon the anticipated join to the group associated with the neighbour region. Each peer periodically looks for new advertisements through the *JXTA getRemoteAdvertisement()* method. Each *JXTA* advertisement is characterized by an *expiration time* after which the advertisement index is flushed from any rendez-vous peer. *JaDE* assigns a low expiration time of the advertisement in order to avoid to fetch advertisement of objects which have not been modified. As discussed in Sect. III, *JaDE* persistence mechanisms require the choice of a *BackUp Peer* to store the passive objects of inhabited regions. *JXTA* offers a simple mechanism based on the *Pipe Binding Protocol* to notify the identity of the *Backup Peer* to all the peers of the DVE. The last peer exiting a region R copies all the objects of R into a *backup cache*. Then, it publishes a *backup pipe* advertisement identified by the name of R and binds this pipe in input. From now on, each peer P entering R checks if R is inhabited and, in this case, it exploits the *Discovery Protocol* to find out the advertisement of the *backup pipe*. Then P binds the *backup pipe* and contacts the *backup peer* to obtain the objects in R.

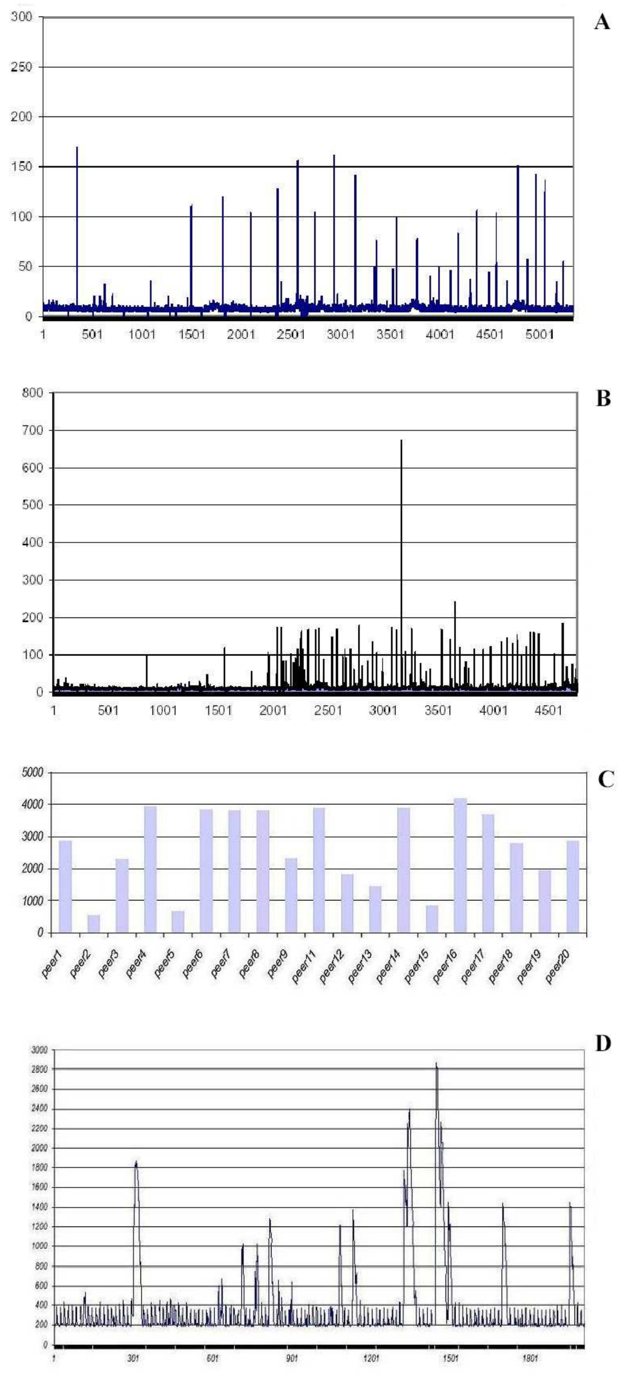

Fig. 2. Experimental Results

If the last peer P that leaves a region R leaves the *DVE* as well, it choses at random a peer of the application within the *NetPeerGroup* group which includes all the peers of the

DVE. This peer becomes the *backup peer* of R, receives all the objects of the region from P and stores them in its *backup cache*. Then, it publishes a backup pipe advertisement identified by the name of the region within the *NetPeerGroup*, then it behaves as in the previous case.

Finally, the *Discovery Service* is exploited by the *coordinator* of an object to publish the state of an updated object.

VI. Experimental Results

JaDE has been implemented in *JAVA* and tested on a *LAN*. A simple experiment has been performed on the *Internet* as well. We have developed a simple game where each peer connects to the virtual world, moves at random in one of 8 possible directions, and sends an heartbeat to its neighbors every $200ms$.

The goal of the first set of tests is to compare the *Propagate Pipe Service* against the *Resolver Service* to support of *heartbeat notifications*. Each peer is executed on a different Athlon 2600+ processor with 512 MB, and a Ethernet Realtek 100 *LAN* is exploited.

In the first set of tests, each peer sends to a propagate pipe a heartbeat timestamped with the value of its clock. Since synchronization of physical clocks could not be guaranteed in our environmental test, it is not possible to evaluate the propagation time of an heartbeat by considering the associated timestamp. For this reason, a *ping mechanism* has been defined through a *Replay peer* that replies to each heartbeat it receives from the propagate pipe. The reply enables the sender to compute the heartbeat latency.

Fig. 2 A shows the results of an experiment where 20 peers have been activated. The goal is to evaluate the average latency of heartbeat notification in a region of the *DVE*. Any peer sends an heartbeat on the propagate pipe every 200 ms and one of the peers acts as a *replay peer*. The Figure shows the latency (misured in ms.) computed by one of the peers at each instant of time of the experiment. The value of the average latency over all the peers is $8,57$ ms, while the maximum value is $255ms$. The results show that, in spite of some peak values, the average latency is low and compatible with the real time requirements of a *DVE*. It is worth noticing that any peak regards a pair of messages, while a total of 5000 messages are sent.

Fig. 2 B shows the same results when the *Resolver Protocol* is exploited. The value of the average latency over all the peers is $17,43$ ms, while the maximum value is 802 ms, The results show that the *Pipe Binding Protocol* is more suitable to support heartbeat notification.

A third test evaluates the time to discover the set of passive objects in particular region. Now, 20 peers are activated in a region R and one of them creates and publishes 20 objects inside R. The other ones look for the objects by executing a *Discovery Request* and compute the time to receive all the objects in the region.

Fig. 2 C shows the average time each peer requires to discover the 20 objects. The average time is $2,7$ sec. Even if this time is rather high, we think it may be hidden by applying the prefetching mechanism. Furthermore, object updates are more unfrequent than players movements and each peer can tolerate some delay in the notification of events located far from it.

The last test is a first evaluation of *JaDE* heartbeat notification mechanism on the Internet. This test involves 3 peers. $Peer_1$ is both a *Seed rendezvouz* and a *replay peer* while $Peer_2$ and $Peer_3$ are *edge peers*. The test exploits the same ping mechanism previously described. Fig. 2 shows the latency $Peer_2$ observes. Obviously, this latency is larger than the one of the LAN test, but it may be acceptable for a *DVE*.

VII. Conclusion and Future Work

This paper has presented *JaDE*, a *JXTA* based support for the development of *DVR*. At the best of our knowledge, this is one of the first proposals of a *DVE* support based on the *JXTA* technology. *JaDE* protocols guarantee both the *persistence* and the *consistency* of the *DVE* passive objects. A prototype has been implemented and a first set of preliminary results has been presented. We plan to refine the prototype in several directions. First of all, the definition of dynamic *AOI* will be investigated. We plan also to investigate an hybrid solution, where the *DVE* is statically partitioned into a set of regions, but *dynamic* areas of interest are exploited within each region. Furthermore, we are going to develop a larger set of evaluations of *JaDE* on a *WAN*.

VIII. Acknoledgments

We wish to thank Silvia Ciotti for her valuable support.

References

[1] S. Ciotti *JaDE: un Supporto JXTA per ambienti Virtuali Distribuiti*, Grad. Thesis, Univ. of Pisa, June 2007

[2] JXTA official web site *https://jxta.dev.java.net/*

[3] L.Ricci, A. Salvadori *Nomad: Virtual Environments on P2P Voronoi Overlays*, First Int. Workshop on Peer to Peer Networks (PPN 2007), Lecture Notes in Computer Science Vilamoura, Portugal, November 2007

[4] F.Baiardi and A. Bonotti and L.Genovali and L.Ricci. *A publish subscribe support for networked multiplayer games* IASTED Internet and Multimedia Systems and Applications (EuroIMSA 2007), Chamonix, France, March 2007.

[5] F.Baiardi and L.Genovali and L.Ricci *Improving Responsiveness by Locality in Distributed Virtual Environments* 21 ECMS, Prague, June 2007

[6] A.Bonotti and L.Genovali and L.Ricci *DIVES: A Distributed Support for Networked Virtual Environments* 20th IEEE AINA 2006, Wien, Austria, April 2006

[7] J. Keller and G. Simon *Toward a Peer-to-Peer Shared Virtual Reality* 22nd International Conference on Distributed Computing, July 2002

[8] A. Yu and S. T. Vuong *MOPAR: a mobile peer-to-peer overlay architecture for interest management of massively multiplayer online games*, NOSSDAV 05 Washington, June 2005, pp. 99-104 .

[9] B.Knutsson, H.Lu, W.Xu, B.Hopkins *Peer-to-Peer Support for Massively Multiplayer Games* IEEE INFOCOM 2004, Hong Kong, March 2004.

[10] Li Zou and M.Ammar and C.Diot, *An Evaluation of Grouping Techniques for State Dissemination in Networked Multi-User Games*, 9th Int.Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS) Cincinnati, Ohio, 2001.

[11] G.Morgan and F.Lu., *Predictive interest management an approach to managing message dissemination for distributed virtual environments* Richmedia2003, October 16th - 17th, 2003, Lausanne, Switzerland.

[12] A.Tanenbaum and M.VanSteen. *Distributed Systems: Principles and Paradigms* Prentice Hall, 2002.

[13] X.Defago and A.Schiper and P.Urban, *Total Order Broadcast and Multicast Algorithms: Taxonomy and Survey* ACM Comput. Surv., Vol. 36, No. 4. (December 2004), pp. 372-421.