Extension and evaluation of JXTA protocols for supporting reliable P2P distributed computing

Fatos Xhafa

Department of Languages and Informatics Systems
Polytechnic University of Catalonia
Campus Nord, Ed. Omega, C/Jordi Girona 1-3
08034 Barcelona, Spain
E-mail: fatos@lsi.upc.edu

Leonard Barolli

Department of Information and Communication Engineering
Fukuoka Institute of Technology (FIT)
3-30-1 Wajiro-higashi, Higashi-ku, Fukuoka 811-0295, Japan
E-mail: barolli@fit.ac.jp

Raul Fernández, Thanasis Daradoumis, Santi Caballé

Department of Information Sciences
Open University of Catalonia
Av. Tibidabo 39-43, 08035 Barcelona, Spain
E-mail: {rfernandezco,adaradoumis,scaballe}@uoc.edu

Abstract

Purpose: In any distributed application, the communication between the distributed processes/nodes of the distributed systems is essential for both reliability and efficiency matters. In this work we address this issue for distributed applications based on JXTA protocols aiming at extending and evaluating the protocols of the JXTA library for reliable P2P computing.

Approach: After a careful examination of the current version of JXTA protocols, we observed the need for improving the original JXTA protocols such as pipe services to ensure reliable communication between nodes of the grid platform and the discovery and presence service to increase the performance of the applications Our approach is based in using a mixed P2P network based on broker peers and client peers architecture, which served as a basis to extend the JXTA protocols.

Findings: The original JXTA protocols are extented/re-implemented to support the development of reliable P2P distributed applications.

Practical implications: The proposed approach has been validated in practice by deploying a P2P network using nodes of PlanetLab platform and testing each of the re-implemented protocols using this real P2P network. The extended JXTA protocols can be used to develop reliable P2P distributed applications.

Category: Research paper

Keywords: *P2P computing, JXTA protocols, distributed computing, pipe services, discovery services, presence services.*

1 Introduction and motivation

The reliability of distributed applications has been largely investigated by researchers of the distributed computing community. While reliability issues are generally well understood for operating systems and classical distributed systems and LANs (see e.g. Guerraoui and Rodrigues, 2006), there is still few work in addressing these issues for the emergent computational systems. With the development of Internet and other new technologies, distributed systems and applications are becoming the indispensable approach for solving complex problems. Therefore, the reliability issue is nowadays being investigated for Web-based distributed systems and applications (Keidl, Seltzsam and Kemper, 2003; Peterson, Ramasubramanian and Gun Sirer, 2006; Ostrowski and Birman, 2006), for Grid-based computing (Foster and Kesselman, 1998; Krepska, Kielmann, Sirvent and Badia, 2006).

Each time more, P2P are used, beyond file and data sharing applications, for developing large scale distributed applications that benefit from the immense computing power contributed by millions of peers worldwide. Projects such as "Folding@Home on the PS3" by seti@home for studying the protein folding by utilizing the new Cell processor in Sony's Playstation 3 are allowing to achieve performance previously only possible on supercomputers. The main challenge in developing large-scale P2P applications is how to develop efficient, scalable and reliable distributed system from inexpensive unreliable computers contributed to P2P network by millions of individuals. One of the today's technologies used in developing P2P systems is JXTA (Brookshier, Govoni, Krishnan, and Soto, 2002; Li, 2003; Oaks, Traversat and Gong, 2003). This is a recent technology, which has been used in many P2P projects (Jxta Projects, 2007) and is currently attracting the attention of many researchers and developers of the P2P and Grid computing. In particular the reliability and efficiency issue is being studied for JXTA-based applications (Halepovic and Deters, 2003; Riasol and Xhafa, 2006).

This work is motivated by the need to support the development of efficient and reliable P2P applications using JXTA protocols. To this end, we have carefully analyzed the current JXTA protocols and report several limitations of most important protocols of JXTA. After examining and pointing out such limitations, we further propose a solution to them through re-implementation and extensions without damaging the generality of JXTA protocols. More precisely, in this work we have considered the following protocols: *Peer Discovery Protocol*,

Discovery Service, *Peer Information Protocol*, *Peer Information Service*, *Peer Resolver Protocol*, *Resolver Service*, *Pipe Binding Protocol* and *Pipe Service*, *Endpoint Routing Protocol* and *Endpoint Service*. Our approach is validated in practice by deploying a real P2P network using the nodes of the PlanetLab platform (PlanetLab, 2007), a planetary scale distributed infrastructure and have experimentally evaluated the performance and reliability of the re-implemented protocols.

The rest of the paper is organized as follows. We briefly overview JXTA basic entities n Section 2. The evaluation of the JXTA protocols and their limitations are given in Section 3. In Section 4 we present the re-implementation of the JXTA protocols; their evaluation is given in Section 5. Finally, we conclude in Section 6 with some remarks and indicate directions for future work.

2 JXTA overview

JXTA technology is a set of open protocols proposed by Sun Microsystems that allow any connected device on the network ranging from cell phones and wireless PDAs to PCs and servers to communicate and collaborate in a P2P manner (Brookshier, Govoni, Krishnan, and Soto, 2002; Oaks, Traversat and Gong, 2003). JXTA peers create a virtual network in which any peer can interact with other peers and resources directly even when some of the peers and resources are behind fire-walls and NATs or when they are on different network transports.

JXTA library offers a set of basic protocols to implement P2P networks independently of the platform, ubiquitously (any connected digital device can be accessed) and inter-operability (by using identifiers, clients can traverse different networks and can change transport).

The current specification of this architecture determines the management of the clients (peers) of the developed networks:

- Discover peers and resources on the network even across fire-walls.
- Share files with peers across the network.
- Create your own group of peers of devices across different networks.
- Communicate securely with peers across public networks.

One important advantage of JXTA is its platform independence as regards:

- *Programming language*: Implementations of JXTA protocols are found in Java, C, C++ and other languages.
- *Transport protocol*: It is not required to use an specific net structure. P2P applications can be developed over TCP/IP, HTTP, Bluetooth and other transports.

- *Security*: Developers are free to manage the security issues of the developed platform.

The JXTA protocols and services have been structured as a *working stack* (see Fig. 1). It should be noticed that JXTA protocols are independent among them. A definition of a JXTA peer doesn't require implementing the set of all protocols in order to participate in the network.

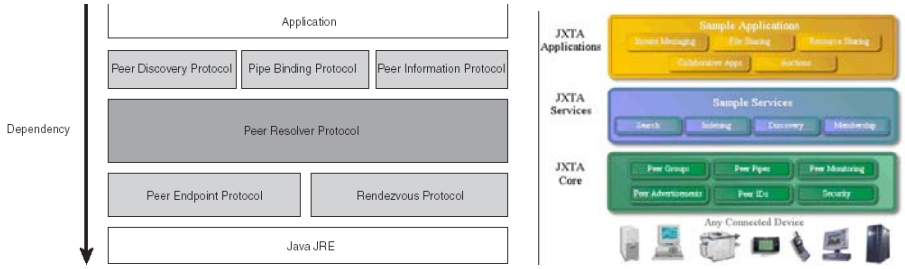

Figure 1: JXTA Architecture and Services.

2.1 Main entities of JXTA networks

The main entities of a JXTA network are as follows.

Peer. Any interconnected node is called peer. Peers work independently and asynchronously with other peers. Peers publish one or more interfaces that are used by other peers to establish peer-to-peer connections.

PeerGroup. A peerGroup is a collection of peers that are joined to provide a secure shared environment for the participating peers. PeerGroups can decide their own policy of peer membership. Note that peers can simultaneously belong to more than one peerGroup. As in the case of peers, peerGroups offer their services such as discovery services, pipe services, peer join services, monitoring services etc.

Pipes. A pipe is a virtual communication channel established between two processes. A computer connected to the network can open, at transport level, as many pipes as its operating system permits. These channels act as data links between the two communication points. Through pipes can be sent any types of objects such as strings, binary code, java objects, etc. JXTA offers both unidirectional, not secure pipes and bidirectional secure pipes.

Messages. Messages are objects used for communicating and interchanging data. A message is essentially an ordered sequence of tags/values, which could also include binary code (appropriately encoded).

Advertisements. The JXTA resources and services are represented using advertisements. An advertisement is a meta-data structured information (XML document), which are published with a certain lifetime that specifies its availability.

2.2 Types of peers

Peers can be classified in different types according to their role in the P2P network as follows.

Limited Edge peer: peers that can send and receive messages but cannot send advertisements neither store advertisements in their cache.

Complete Edge peer: peers that can send and receive messages, store advertisements but cannot manage resource discovery advertisements.

Rendezvous peer: peers that can send and receive messages, store advertisements and moreover can manage resource discovery advertisements. Each group of peers must have at least a rendezvous peer; when a peer joins a group of peers, it automatically looks for a rendezvous in the group. Moreover, rendezvous keep a list of other rendezvous and their respective peers.

Relay peer: this type of peers serve as intermediate nodes. Relay peer keep routing information to other peers and sends messages to other peers that can't access directly to another peer.

3 The JXTA protocols and their limitations

JXTA protocol standardize the way client peers discover other client peers, communicate among them, get organized into groups, announce, advertise and discover services and resources in the network and can monitor other peers.

3.1 The JXTA protocol and services

The services offered by JXTA library are based on its own protocols and serve as the starting point in the development of a P2P network. Certainly, it is upon the developer to extend/implement these services according to his needs. JXTA offers essentially six protocols, each one with its corresponding services. It should be noted that when developing a JXTA-based P2P application, all these protocols and services are not necessarily used. These protocols and services are the following:

- *Peer Discovery Protocol* and *Discovery Service*, which serve to advertise the proper resources (peers, groups, pipes, services...) and to discover the resources of other peers.
- *Peer Information Protocol* and *Peer Information Service*, which are in charge of obtaining state information of local or remote peers (time, state, recent traffic, etc.). In a sense these protocols and services allow to monitor other peers participating in the network.

- *Peer Resolver Protocol* and *Resolver Service*, which are used for sending a general query to one or more peers and receiving one or more responses in order to interchange desired information.
- *Pipe Binding Protocol* and *Pipe Service*, which serve for establishing a virtual communication channel, that is a *pipe*, to enable sending and receiving data from other peers.
- *Endpoint Routing Protocol* and *Endpoint Service* for routing to other peers. The route information contains an ordered sequence of relay peers that must be used to send a message to the specified destination peer.
- *Rendezvous Protocol* and *Rendezvous Service*, which constitute the mechanisms through which peers can be subscribed to a propagated service or a source peer can propagate a message to all peers subscribed to the service.

3.2 Limitations of JXTA protocols

While developing a P2P network, independently of its specific purposes there are several key issues:

(a) the service publishing and reception;

(b) the network connection; and

(c) message sending and reception.

JXTA offers protocols and services to support these needs, however, they show several limitations that we overcame by adding new functionalities to the original services. We analyze next the limitations of JXTA protocols.

Management of the presence mechanism. The presence of a peer node in a peerGroup of the JXTA network is provided by the document of presence notification. This document is published by each peer and reaches the local cache of all peers, where it will "stay" for a certain time. The presence mechanism is very important for the JXTA network in order for a peer's advertisement and therefore its services, to reach all peers that are present. Most importantly, the rendezvous must know all peers connected to the peerGroup because the rendezvous is in charge of synchronizing the local caches of peers and compute the routes to be used by any peer to reach all the connected peers of the group.

Advertisement documents are of *PeerAdvertisement* type and must periodically be published in order for other peers of the group to know the updated presence information of the peer that publishes the advertisement document. However, this publishing is not done automatically in JXTA; hence any application using JXTA must manage it by its own! Essentially, any JXTA-based application has to manage the information of the updated presence (a sort of refreshment of presence) of any peer node in any peerGroup it belongs to.

Management of the connection to groups. In any mixed P2P network, that is, a P2P network in which we have *broker peers* and *client peers* (Riasol and Xhafa, 2006), any client peer can be connected only to one peerGroup by discovering and establishing communication with a rendezvous. Therefore it is necessary to have a broker peer in any peerGroup, which must act as a rendezvous in order to achieve that client peers get connected and the broker could notify the presence of the peer to the rest of the nodes of the peerGroup. It should be noted that the connection to a peerGroup is done differently depending on whether the peerGroup is specific or is a general group.

The general group is the *NetPeerGroup*, the peerGroup to which must get connected any peer to join the JXTA network and to collaborate and work with other peers. On the other hand, the connection to other peerGroups is not indispensable in order for the peer to start working in the JXTA network, but rather it serves to make much more specific the peer's characteristics and functionalities. Moreover, in this way, the number of peers that could be used for a certain purpose can be reduced. In JXTA, there are no such mechanisms for broker peers, as matter of fact, broker peer type is not defined in JXTA.

Managing the communication mechanism. In JXTA, pipes are used as a basic direct communication mechanism between two peers belonging to the same peerGroup. Pipes are of two types: *InputPipe* and *OutputPipe*. Clearly, if a peer A wants to send a message to a destination peer B, it is necessary that an instance of A's outputPipe get connected to an instance of B's inputPipe. In order to establish this connection, peer A must know B's inputPipe and peer B must accept messages from A's outputPipe. This is achieved in JXTA by using descriptions of pipes in a way that they are unique, that is, an inputPipe can only receive messages from an outputPipe having the same description. The description is done in XML documents, called PipeAdverstisement, which is of type advertisement and has defined, besides its unique identifier, several parameters such as name, type and description. Therefore, in order for peer A to send a message to peer B, both inputPipe and outputPipe must be defined in the same PipeAdvertisement and both peers must agree which one to use. Thus, we have to find the appropriate protocol to assure the communication among all peers of the same peerGroup.

4 Re-implementation of JXTA protocols

In order to overcome the limitations of the JXTA protocols shown in the previous section, we re-implemented the JXTA protocols and services by adding new functionalities and control to the original services[1].

[1]Source code, documentation and application examples are available at http://jxta-overlay.dev.java.net

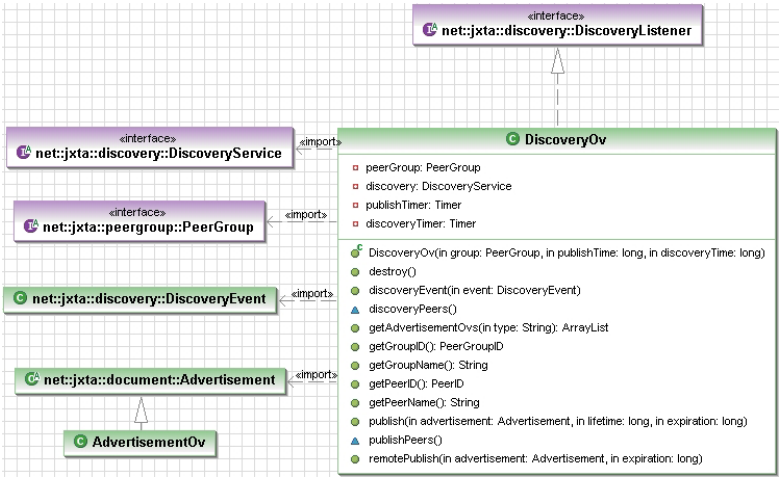

Figure 2: UML diagram of the overlay module

4.1 Management of the presence and service mechanisms

Recall that peer's presence is done through the presence notification document. This document must be periodically published in order for other peers of a peerGroup to know the presence of the peer. This is one of the limitations of JXTA that we have solved by adding to peer's functionalities, the publishing of its own PeerAdvertisement periodically (say, each 5 seconds). In this way, the peers of the peerGroup will know the presence of the peer and hence they will send to the peer their future advertisements. Moreover, in order for the peer to know all services of the P2P network, it has to discover such services, which are found in different advertisements and are stored in the peers' local caches.

Therefore the peer's cache must be frequently updated. One way to do this, is to look up local caches of peers in the peerGroup by sending queries. A query is a request sent to all peers of the peerGroup which upon reaching a peer, examines its local cache, and recollects all the advertisements of the specified type. When such query reaches a rendezvous, it recollects all advertisements of all peers of the peerGroup.

Finally, the results of the query are stored in the local cache of the peer, which can use them later on for its purposes. The main issue here is thus the synchronization of the local cache of a peer with local caches of other peers of the peerGroup. To this end, in our implementation, a query is sent periodically (say each 5 seconds) to all peers of the peerGroup. More precisely this is done in a new module[2], called DiscoveryOverlay, as shown in Fig. 2.

[2]In all figures, the abbreviation ***Ov*** stands for *Overlay*, which refers to the new implementation.

4.2 Management of the connection to groups

Recall that any peerGroup needs a rendezvous peer. The principal peerGroup is that of NetPeerGroup, the connection to which is done according to the peer's configuration:

a) Connection to the default rendezvous of JXTA. This is certainly inefficient since we cannot control the rendezvous;

b) Connection to the rendezvous pre-specified by the concrete JXTA application;

c) Connection to the rendezvous specified in a rendezvous list. In this case, the JXTA application just needs to know the address where to find the rendezvous list.

In our implementation, we have implemented the alternative c). Note that in this case, the rendezvous list can be changed independently of the P2P application. Moreover, once a peer is connected to the P2P network through the principal peerGroup, it can join any peerGroup by knowing the GroupAdvertisement of such groups. However, for this to be done, the peerGroups must exist, there must be at least one rendezvous and the GroupAdvertisements must be propagated. In our P2P network, this is achieved through broker peers and client peers (see Fig. 3 for a general view of our JXTA P2P architecture). Brokers are the governors of the network and control the organization of the groups. Thus brokers are in charge of creating the groups and propagate the GroupAdvertisement accordingly.

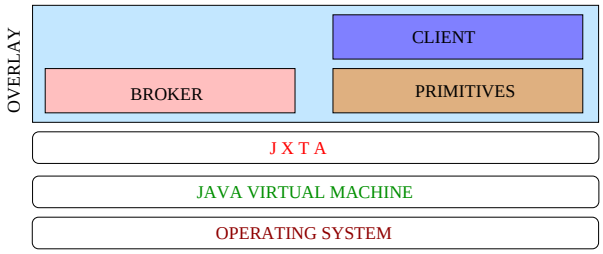

Figure 3: The Architecture of the P2P Overlay.

4.3 Management of the communication mechanism

Recall that in order for a peer to send a message to another peer, there must exist a connection between an outputPipe and inputPipe defined in the same PipeAdvertisement and therefore both peers have to agree on which PipeAdvertisement to use. One possible solution would be to create a PipeAdvertisement for any possible connection between any two peers of a peerGroup according to

a predefined protocol. This is not efficient given the large number of different PipeAdvertisements needed to manage the connections.

We decide then to create a unique PipeAdvertisement by a concrete peer in a peerGroup, having thus operative only one inputPipe through which to receive messages from all the rest of peers of the peerGroup. It is necessary to find a way to create such a unique PipeAdvertisement per peer and how other peers can know the peer's PipeAdvertisement. Since in JXTA, each peer has its unique PeerID, we associate a PipeAdvertisement with a peer by including the PeerID in PipeAdvertisement variables. Once the PipeAdvertisement is created, the source peer has to create an outputPipe and the destination peer has to create an inputPipe, both of them defined with the same PipeAdvertisement. This can be done easily for the destination peer, which just needs to create a unique inputPipe for each peerGroup to which it belongs because the peer will receive messages defined by a unique type of pipeAdvertisement.

It is a bit more complex, however, for the source peer, which has to create an outputPipe defined by the pipeAdvertisement of the destination peer. Notice that the source peer just needs to know the pipeAdvertisement of the destination peer, which is easily since the source peer keeps in its local cache the advertisements of the rest of the peers, and because pipeAdvertisement are identified by PeerID; the source peer can thus find the pipeAdvertisement of the destination peer.

We have thus implemented this approach for the case of receiving and sending messages and pipe services. We show in Fig. 4 the UML diagram of the re-design of the JXTA pipe services mechanisms.

4.4 Management of peer's state information

As already mentioned, keeping updated information of the network is crucial in P2P applications. To this end, we have implemented functionalities that allow to recollect information on the nodes of the network and publish it periodically through advertisements. This is done externally from JXTA. More precisely, each time a node changes its state due to any network event or application event, the node's state information is updated. This information is defined at three levels: (a) for the last k hours (being k a parameter to be specified); (b) for the current session and, (c) for all sessions (historical data).

The peer node publishing this information using advertisements, which are periodically published. Moreover, the peer node computes several statistics, which are sent together with the peer's state information; thus, any peer can use statistical information of other peers in order to take efficient decisions. For the purpose of the implementation, we have implemented a module LocalCache, which has functionalities to control the local cache of the peer (see Fig. 5).

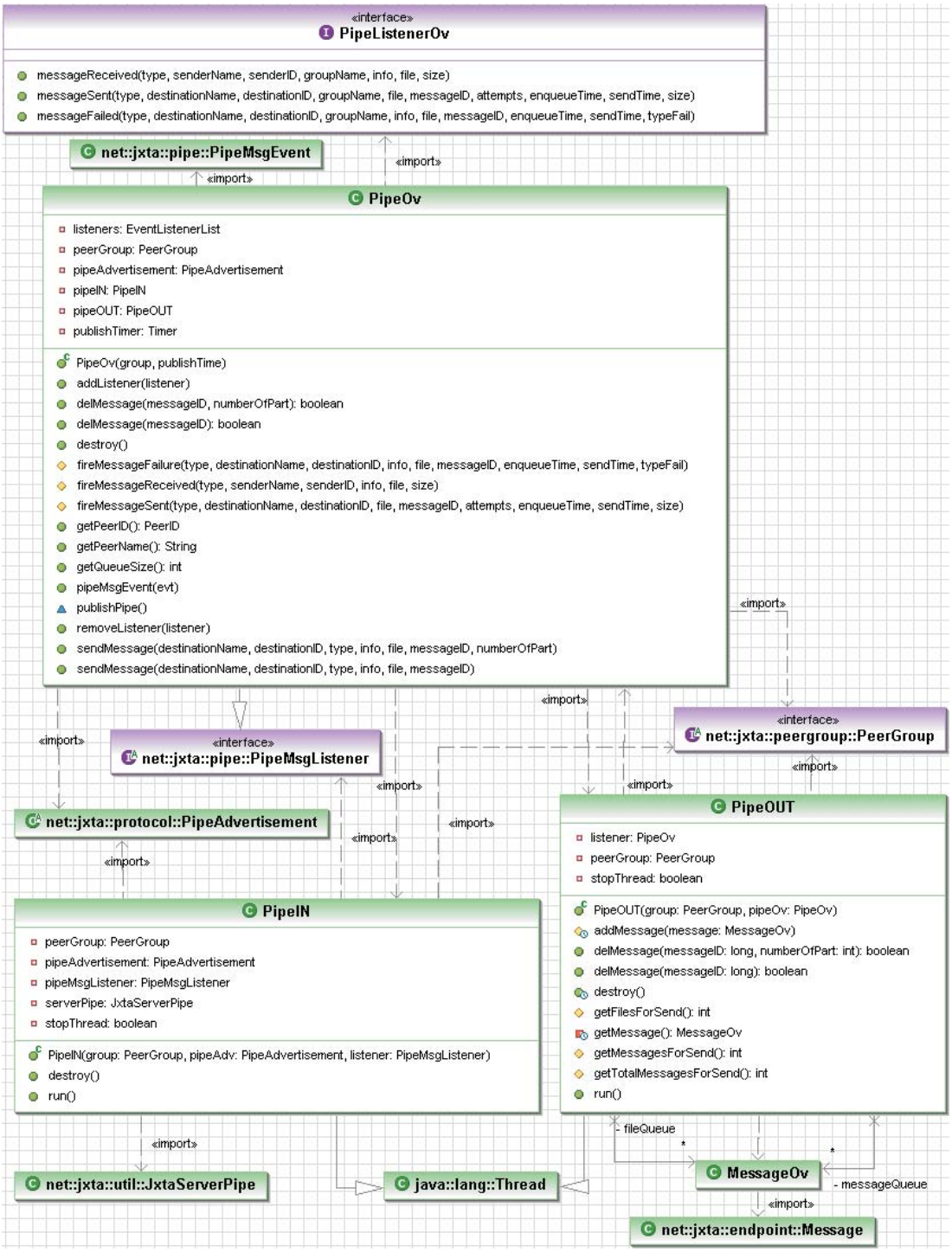

Figure 4: UML diagram of the Pipe Service re-implementation

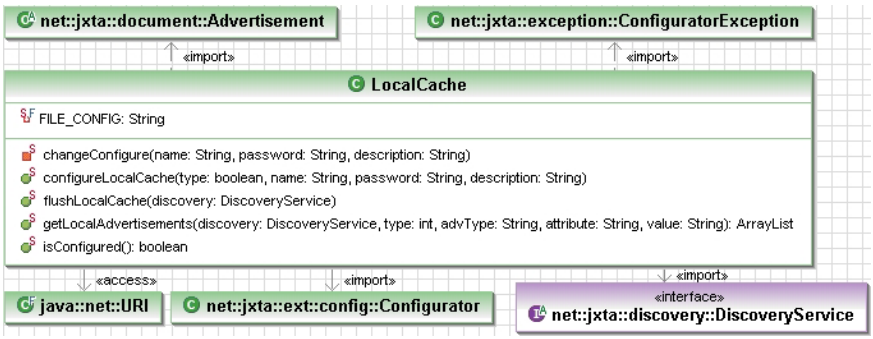

Figure 5: UML diagram of information management re-implementation

5 Testing the re-implementations of the JXTA protocols

5.1 Deployment of the P2P network

In order to evaluate the performance of the extensions of the JXTA protocols, we deployed a P2P network using the extended JXTA library using nodes of the PlanetLab platform. PlanetLab (PlanetLab, 2007) is an open platform for developing, deploying and accessing planetary-scale services. It is, at the time of this writing, composed up of 784 nodes at 382 sites. Each Planetlab node is an IA32 machine that must comply with minimum hardware requirements (i.e. 1GHz PIII + 1Gb RAM) running the same base software, basically a modified Linux operating system offering services to create virtual isolated partitions in the node, called slivers, which look to users as the real machine. Planetlab allows every user to dynamically create up to one sliver in every node, the set of slivers assigned to a user form what is called a slice. It is said that a Planetlab node can run up to 100 concurrent slivers.

The sample set of PlanetLab's machines forming the slice is about 25 nodes shown in Table 1. Moreover we used the cluster nozomi.lsi.upc.edu (a main control node + five computing nodes). The main node was used as one the brokers of the P2P network.

5.2 Reliability tests and computational results

We conducted several experiments to test the reliability and efficiency of the re-implemented protocols and compare them with original JXTA protocols. The experimenting was thus conducted twice under the same P2P network using PlanetLab nodes.

Table 1: Nodes added to the PlanetLab slice.

ait05.us.es	planet01.hhi.fraunhofer.de
planet1.cs.huji.ac.il	planet1.manchester.ac.uk
system18.ncl-ext.net	planetlab1.net-research.org.uk
planetlab01.cs.tcd.ie	planet2.scs.stanford.edu
planetlab01.ethz.ch	planetlab-1.ssvl.kth.se
planetlab1.esi.ucm.es	planetlab1.csg.unizh.ch
planetlab1.poly.edu	planetlab1.cslab.ece.ntua.gr
planetlab2.ls.fi.upm.es	planetlab1.eecs.iu-bremen.de
planetlab2.upc.es	planetlab1.hiit.fi
lsirextpc01.epfl.ch	planetlab5.upc.es
ricepl-1.cs.rice.edu	planetlab1.itwm.fhg.de
planet2.seattle.intel-research.net	planetlab1.informatik.unierlangen.de
edi.tkn.tu-berlin.de	

Testing the presence and service management. When a peer client joins the network for the first time, its local cache is created empty. Next, the local cache must get synchronized, at first place, with the local cache of the rendezvous and then with those of other peers of the same peerGroup. On the other hand, if the peer has already joined the P2P network in previous sessions, the advertisement are stored in its local cache. We observed and experimentally measured that the synchronization for the first connection takes, as expected, much more time than when the peer has already done some prior sessions in the P2P network.

Once the peer has already its local cache synchronized, we experimentally considered and observed the services that a peer can offer. Peer's services are implemented in a way that each service can have its lifetime and its publishing frequency. We briefly mention here three scenarios that we considered:

a) *Statistics services*: these services distinguish for their long lifetime (e.g. order of minutes) and a high publishing frequency (e.g. order of seconds, say, 30 seconds). By monitoring the statistics of the P2P nodes, we observed that these services correctly kept the information on all peers.

b) *Rendezvous services*: these services include criteria for choosing a peer for efficient computation or efficient data transmission, which require that the rendezvous' advertisements must be synchronized because through it are synchronized the local caches of other peers.

c) *Peer common services* (e.g. information on peer node): these services have usually a short lifetime, otherwise in case the peer is disconnected from the P2P network its services would be *alive* even the node is not present in the network. Experimental values showed that 60 seconds was a good value for the lifetime of such services, meaning that an invalid service could be alive during that time but because the publishing time is inferior to that, a new publishing could take place.

The time connection to a peerGroup. Recall that at least one rendezvous of the general group must be permanently operative so that different peers

will try to join the network by trying to connect to the rendezvous. In order to increase the possibilities that a peer will always establish connection to a rendezvous we used a rendezvous list implementation. The experimental data showed that, although it could take more time to establish the connection (now a peer has to examine the list of rendezvous), it considerably increases the possibility of a peer to establish connection with a rendezvous. In fact with our real P2P network, we did not observe failure to establish connection. We show in Fig. 6 the connection time for the case when there is just one broker in the list and when there are 11 brokers (the experiment was repeated several time by changing the order in the broker list and the connection time to each broker is reported).

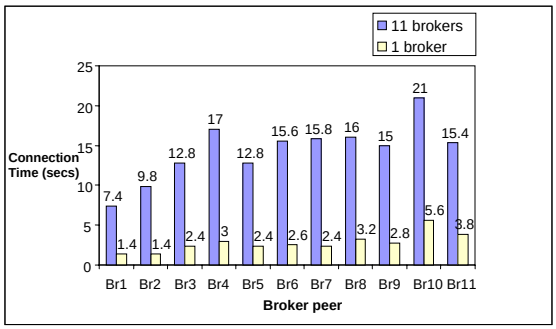

Figure 6: Peer's connection time to a broker of the P2P network.

The communication test. One important parameter to measure is the pipe connection times, which depending in the type of peer and the network could vary considerably. Essentially, in order to establish a pipe, JXTA has to previously do some communications based in a communication protocol and each of these connections has its ping time. Due to this variability, considering a fixed timeout (say 2 seconds) turned out to not be a good choice; it worked very well for cluster-based P2P network but it was not reliable for a geographically distributed P2P network.

This motivated precisely one of the improvements presented in this work, namely, the progressive augmentation of the window timeout, which in our experiments was upper bounded by 40 seconds. Recall that in our re-implementation of pipes, if a message is not delivered to destination peer, it is intended again to send it. The number of intents depends on the time window used. The experimental data showed that by using a time window that progressively increases at most to 40 seconds, almost all messages were sent to destination. By using this approach less than 2% of messages that could not be delivered (e.g. because the peer is already disconnected) were reported as errors as opposed to 32% of undelivered messages that were reported as errors with a fixed timeout. We show in Fig. 7 the progressive timeout used and the corresponding number of sent messages.

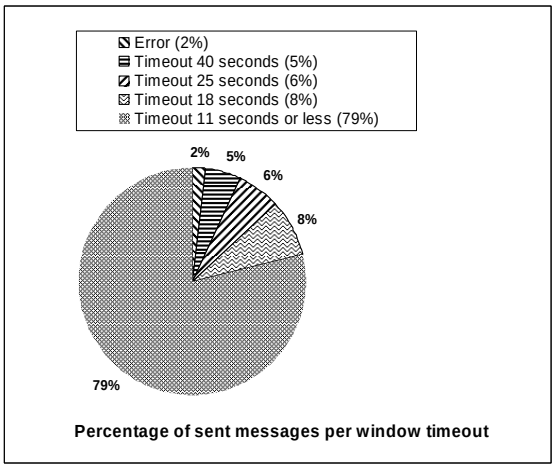

Figure 7: Time out values and the corresponding percentage of sent messages

Information management test. In our approach, each node publishes a large quantity of information and therefore peer nodes of the P2P network have better knowledge of the peerGroup and of the network and can use the information of other peers in decision-taking process. In this context, we considered as a test finding the best peer for efficiently running a task from many candidate peers. In absence of information about the other peer nodes, the task assignment would essentially reduce to a random assignment, which is not efficient. By using the information published by the peer nodes brokers are able to find the best peer among different candidates. Thus, we implemented a data-evaluator model as part of brokerage services to identify the best peer for executing a task.

6 Conclusions and future work

In this work we have analyzed the protocols and services of JXTA library and have shown several limitations of these protocols regarding their efficiency and reliability. The analyzed protocols include discovery, peer information, peer resolver, and pipe binding protocols and services, among others. We then proposed extensions and re-implementation of these protocols in order to overcome the identified limitations. The proposed extensions of the JXTA protocols have been validated in practice using a real P2P network deployed in nodes of PlanetLab platform. The experimental results showed the improvement of both efficiency and reliability of JXTA protocols and services.

In our future work we plan to use the re-implemented protocols in large-scale JXTA-based applications for data intensive computing and measure the performance of the JXTA protocols in highly dynamic environment using a large number of unreliable peer nodes.

Acknowledgements

This work has been partially supported by the Spanish MCYT project TSI2005-08225-C07-05.

References

D. Brookshier, D. Govoni, N. Krishnan, and J.C Soto. *JXTA: Java P2P Programming*. Sams Publishing, 2002.

I. Foster and C. Kesselman. *The Grid - Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, 1998.

R. Guerraoui and L. Rodrigues. *Introduction to Reliable Distributed Programming*. Springer Verlag, 2006.

E. Halepovic and R. Deters. The costs of using jxta. In *Proceedings of the 3rd International Conference on Peer-to-Peer Computing*, page 160. IEEE Computer Society, 2003.

JXTA. Projects:. https://communications.dev.java.net/. (As of October 2007).

M. Keidl, S. Seltzsam and A. Kemper. *Technologies for E-Services*, volume 2819, chapter Reliable Web Service Execution and Deployment in Dynamic Environments. Springer Verlag, 2003.

E. Krepska, Th. Kielmann, R. Sirvent, and R.M. Badia. A service for reliable execution of grid applications. In *2nd CoreGRID Integration Workshop. Integrated Research in Grid Computing. Krakow (Poland), October 2006.*, pages 232–242, 2006.

Planet Lab. http://planet-lab.org/.

S. Li. *Early Adopter JXTA*. Wrox Press Information Inc., 2003.

S. Oaks, B. Traversat, and L. Gong. *JXTA in a Nutshell*. O'Reilly, 2003.

K. Ostrowski and K. Birman. Extensible web services architecture for notification in large-scale systems. In *International Conference on Web Services (ICWS 2006)*. IEEE, September 2006.

J. Esteve Riasol and F. Xhafa. Juxta-cat: a jxta-based platform for distributed computing. In *PPPJ '06: Proceedings of the 4th international symposium on Principles and practice of programming in Java*, pages 72–81, New York, USA, 2006. ACM Press.

R. Peterson, V. Ramasubramanian and E. Gun Sirer. Corona: A high performance publish-subscribe system for the world wide web. In *In Proceedings of Networked System Design and Implementation, San Jose, California*, May 2006.

Bibliographical notes

Fatos Xhafa is an Associate Professor of Computer Science at the Polytechnic University of Catalonia (Barcelona, Spain). He received his Ph.D. in Computer Science from the Polytechnic University of Catalonia in 1998. He joined the Department of Languages and Informatics Systems of the Polytechnic University of Catalonia as an Assistant Professor in 1996 and is currently Associate Professor and member of the ALBCOM Research Group of this department. His current research interest include parallel algorithms, approximation and metaheuristics, distributed programming, Grid and P2P computing. He has served as an Editor and Guest Editor for several Journals and is presently the PC Chair of CISIS-2008 and the Organizing Chair of ARES-2008.

Leonard Barolli is a Professor in Department of Information and Communication Engineering, Fukuoka Institute of Technology, Japan. He received BE and PhD Degrees from Tirana University and Yamagata University in 1989 and 1997, resp. He has published more than 200 papers in refereed Journals and International Conferences. He has served as a Guest Editor for many Journals. He was PC Chair of IEEE AINA-2004 and ICPADS-2005, General Co-Chair of IEEE AINA-2006 and Workshops Chair of iiWAS-2006 and Workshops Co-Chair of IEEE AINA-2007 and ARES-2007. Presently, he is General Co-Chair of IEEE AINA-2008, General Co-Chair of CISIS-2008 and Workshops Co-Chair of ARES-2008. His research interests include high-speed networks, grid computing, P2P, ad-hoc and sensor networks. He is member of IEEE, IPSJ and SOFT.

Raul Fernández holds an Engineering degree in Informatics from the Polytechnic University of Catalonia (Barcelona, Spain). He is interested P2P computing, distributed applications, e-Learning distributed applications. He has participated in Spanish research projects.

Thanasis Daradoumis is an Associate Professor of Computer Science at the Open University of Catalonia (Barcelona, Spain). He has a PhD in Information Sciences from the Technical University of Catalonia (Spain), a Master in Computer Science from the University of Illinois, and a Bachelor's degree in Mathematics from the University of Thessaloniki (Greece). His research focuses on distributed and collaborative learning and e-learning technologies.

Santi Caballé is Professor of Computer Science at the Open University of Catalonia (Barcelona, Spain) since 2006. His research interest include development of distributed applications for e-Learning, Web and Grid Services.