DOI 10.1109/TIE.2010.2050751

JXTA-OVERLAY: A P2P Platform for Distributed, Collaborative and Ubiquitous Computing

Leonard Barolli, *Member, IEEE,* and Fatos Xhafa, *Member, IEEE*

***Abstract*—With the fast growth of Internet infrastructure and the use of large-scale complex applications from industries, transport, logistics, government, health and businesses, there is an increasing need to design and deploy multi-featured networking applications. Important features of such applications include the capability to be self-organized, decentralized, integrate different types of resources (PCs, laptops, mobile and sensor devices), and provide global, transparent and secure access to resources. Moreover, such applications should support not only traditional forms of reliable distributing computing and optimization of resources but also various forms of collaborative activities such as business, online learning and social networks in an intelligent and secure environment. In this paper, we present the JXTA-Overlay, a JXTA-based P2P platform designed with the aim to leverage capabilities of Java, JXTA and P2P technologies to support distributed and collaborative systems. The platform can be used not only for efficient and reliable distributed computing but also for collaborative activities and ubiquitous computing by integrating in the platform end-devices. The design of an user interface as well as security issues are also tackled. We evaluate the proposed system by experimental study and show its usefulness for massive processing computations and e-learning applications.**

***Index Terms*—P2P Systems, JXTA library, JXTA-Overlay, Java Applications, End-device Control, SmartBox.**

I. Introduction

The Internet is growing every day and the performance of computers and networks is significantly increased enabling the development of complex, large-scale applications. We are currently witnessing an increasing need to design and deploy multi-featured networking applications instead of stand alone applications for specific needs. Such applications combine different paradigms and are developed using various technologies with the aim of achieving a multi-disciplinary view. The digital ecosystems [5], [8], [10], [21] are emerging as a paradigm for supporting multi-disciplinary and multi-paradigmatic applications capable of being adaptive and socio-technical, having properties of self-organization inspired by natural ecosystems. Important features of such applications include the capability to be self-organized, decentralized, scalable and sustainable as well as integration of different types of resources (PCs, laptops, mobile and sensor devices) providing global, transparent and secure access to resources. Supporting various forms of collaborative activities such as business, on-line learning and social networks in an intelligent and secure environment is also important in such systems. In fact, digital ecosystems are considered as the next generation of collaborative environments.

L. Barolli is with the 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.

F. Xhafa is with Department of Languages and Informatics Systems Technical University of Catalonia, Jordi Girona 1-3, 08034 Barcelona, Spain, e-mail: fatos@lsi.upc.edu.

Manuscript received November 10, 2009; Manuscript revised November 19, 2009; Manuscript accepted January 11, 2010.

The development of such applications requires the combination of many computing paradigms and technologies such as Web, mobile and sensor technologies to achieve ubiquity features. However, the current Internet architecture based on Client/Server topology shows several limitations to efficiently address the high degree of heterogeneity of computational resources and devices, which are useful for the everyday real life activities. Besides, in large scale networks such as Internet, it is very difficult to control network devices due to security problems. Networks have their own security policies and the information should overcome firewalls, which are used for checking the information between private and public networks.

Peer-to-Peer (P2P) systems are an important paradigm for the development of large scale applications endowed with features of digital ecosystems. Indeed, P2P systems [7] can achieve a good scalability and are decentralized in nature. In P2P systems, the computational burden of the system can be distributed to peer nodes. Therefore, the users become themselves actors by sharing, contributing and controlling the resources of the systems. This characteristic makes P2P systems very interesting for the development of decentralized applications [4], [22].

P2P systems have evolved from simple systems of file sharing among Internet users to a disruptive technology for collaborative and social activities. Indeed, such systems are capable to deliver content, profiling, grouping and control to ordinary users in intelligent and interactive environments. Thus, P2P technologies lay the basis for developing applications to support any group of people having in common technical, scientific, cultural, and political interests.

P2P technologies can also efficiently address the ubiquity features of large scale Internet-based applications by integrating any connected devices on the network, ranging from cell phones and wireless PDAs (Personal Digital Assistants) to Personal Computers (PCs) and servers. Recently, there has been an increasing interest in deploying P2P networks that integrate mobile devices such as PDAs and end-devices.

Finally, using P2P technologies, it is possible to overcome firewalls and other security devices without changing the network policy. The P2P architecture is thus very important for controlling end-devices in Wide Area Networks (WANs).

In this work, we present the JXTA-Overlay, a JXTA (Juxtapose) based P2P platform, designed with the aim to leverage capabilities of Java, JXTA and P2P technologies to support distributed and collaborative systems in a decentralized and self-organized manner, capable to integrate different types of peers. The platform can be used not only for efficient and reliable distributed computing but also for collaborative activities and ubiquitous computing by integrating in the platform also end-devices and overcoming thus intrinsic difficulties of current Internet architecture and protocols. Moreover, the design of an advanced user interface as well as enhancement security requirements of JXTA library are also tackled. We consider as an end-device the SmartBox that is able to stimulate learners during their learning activity and thus increasing their learning efficiency and outcomes. We evaluate the proposed system by experimental study and show the usefulness of using SmartBox end-device in the development of e-learning applications.

The structure of this paper is as follows. In Section II, we give a description of the main protocols of the JXTA library. Section III presents the main features of the JXTA-Overlay platform. The use of the JXTA-Overlay platform for masive processing computations is presented in Section IV. We introduce the issue of integration of end-devices into P2P systems in Section V. In Section VI, we present the integration of the SmartBox end-device. In Section VII, we show the evaluation of JXTA-Overlay. The paper ends with some conclusions in Section VIII.

II. JXTA Library for P2P Computing

JXTA technology [6], [19] is a generalized group of six XML (Extensible Markup Language) based protocols that allow different types of peers to communicate and collaborate among them. Peers can be organized into peergroups in a decentralized way. Peers communicate using pipes, which abstract the way in which two peers communicate, where other peers are allowed as intermediaries if communication would not be able due to network partitioning and restrictions. By using these protocols, peers connected to the JXTA network can exchange messages among them in decentralized way. Moreover, by using JXTA protocols it is possible that a peer in a private network can be connected to a peer in the Internet by overcoming existing firewalls or NATs (Network Address Translations) or when different communication protocols are used. The performance of JXTA has been evaluated in several research work and it has been shown that the library is efficient and highly scalable [1], [11].

Peers are uniquely identified allowing that peers can change their address still conserving their unique peer Id. JXTA layers and services are shown in Fig. 1.

A. JXTA Protocols

JXTA comprises a set of six open protocols that enable 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. We briefly describe below these protocols (core and standard protocols).

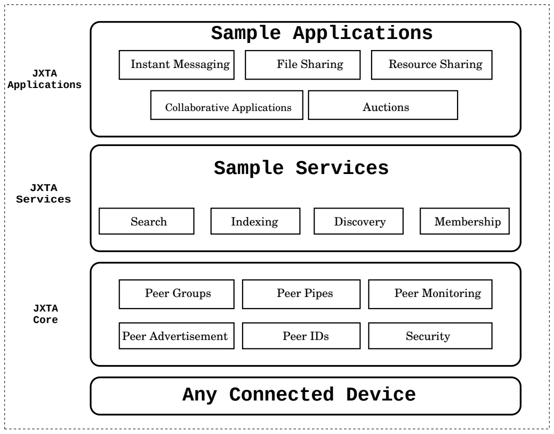


Fig. 1. JXTA layers and services.

Peer Resolver Protocol (PRP): PRP offers a generic interface that allows peers to send generic requests to one or more peers and to receive one or multiple answers. Applications and services can use the protocol for the resolution of services.

Endpoint Routing Protocol (ERP): ERP defines a set of messages which are processed by a routing service to enable a peer's message routing to the destination. Thus, ERP is used to find the available routes to send messages to the destination peers.

Peer Discovery Protocol (PDP): PDP is used to discover the published resources by the peers. The resources are represented through advertisements. A resource can be a peer, a peergroup, a pipe, or any other resource that has associated an advertisement.

Rendezvous Protocol (RVP): RVP is used to propagate messages in a group of peers. The RVP provides mechanisms for controlled propagation of the messages. This protocol comprises the PeerView Protocol, Rendezvous Lease Protocol and Rendezvous Propagation Protocol.

Peer Information Protocol (PIP): PIP provides a set of messages to obtain information about the state of a peer. The PIP protocol is optional for JXTA peers. In fact, a peer does not need to reply to the queries made with PIP protocol.

Pipe Binding Protocol (PBP): PBP is used by the applications and the services in order to communicate with the other peers. A pipe is a virtual channel between two peers and is described as "pipe advertisement". Every time that a pipe is established, an input pipe and an output pipe are established. In fact, PBP can be viewed as a layer over the ERP, and it can use a variety of transport protocols such as "Transport JXTA HTTP", the "Transport JXTA TCP/IP" and the "Secure JXTA TLS Transport" in sending messages.

B. JXTA Entities

The main entities of JXTA networks are as follows.

Peer: Any interconnected node is called peer. Peers work independently and asynchronously with other peers by publishing one or more interfaces that are used by other peers to establish P2P connections. Different types of peers are defined (Limited Edge Peer, Complete Edge Peer, Rendezvous Peer, Relay Peer) according to their role in the P2P network.

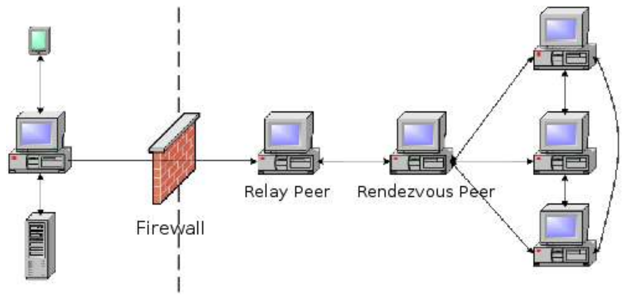


Fig. 2. Communication of peers in a JXTA network.

- *Rendezvous*: The Rendezvous peers are in charge of coordinating other peers in the JXTA network. Additionally, they provide the necessary services for the propagation of messages. Each sub-network of JXTA must have at least a Rendezvous peer.
- *Relay*: The Relay peers allow that peers behind firewalls, NATs or special peers having limited computational power such as mobile devices, PDAs, etc., can be part of a JXTA network. The relay peers achieve this by using protocols which allow to cross the limitations imposed by these systems, e.g., the HTTP protocol (see Fig. 2).
- *Edge*: Edge peers are peers at the edge of the network and usually have limited bandwidth as compared to other types of more powerful peers.

PeerGroup: A PeerGroup is a collection of peers that provide a secure shared environment for participating peers. A PeerGroup can decide its own policy of peer membership. Peers can belong to more than one PeerGroup.

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. JXTA offers both unidirectional, not secure pipes, and bidirectional secure pipes.

Messages: Messages are objects used for communicating and interchanging data. A message is an XML document, which can also include binary code.

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

III. JXTA-Overlay Platform

JXTA-Overlay[1] [23] is a middleware that abstracts a new layer on top of JXTA through a set of primitive operations (services) commonly used in JXTA-based applications.

JXTA-Overlay comprises primitives for: (a) peer discovery; (b) peers resources discovery; (c) resource allocation; (d) task submission and execution; (e) file/data sharing, discovery and transmission; (f) instant communication; (g) peer group functionalities (groups, rooms etc.); and, monitoring of peers, groups, and tasks. This set of basic functionalities is intended to be as complete as possible to satisfy the needs of JXTA-based applications. The overlay is built on top of JXTA layer and provides a set of primitives that can be used by other applications, which on their hand, will be built on top of the overlay, with complete independence. The JXTA-Overlay offers several improvements of the original JXTA protocols/services to increase the reliability of JXTA-based distributed applications and to support group management and file sharing [23]. The architecture of the P2P distributed platform we have developed using JXTA technology has these building blocks: Broker Module, Primitives Module and Client Module. Altogether these three modules form a new overlay on top of JXTA. The JXTA-Overlay structure is shown in Fig. 3.

[1]https://jxta-overlay.dev.java.net/

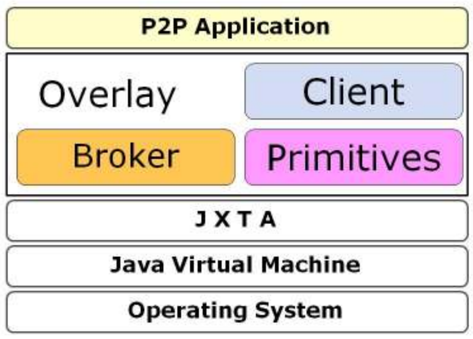


Fig. 3. Structure of JXTA-Overlay.

1) The primitives: The set of primitives includes functionalities that allow peer discovery, peer's resources discovery, resource allocation, file/data sharing, discovery and transmission, instant communication and peer group functionalities, among others. The primitives are organized in interfaces according to an affinity criterion. In what follows we give the main functionalities of these interfaces (we use indistinctly the terms peer and resource).

Authentication: This interface includes typical methods for authentication of the final user/application that will be using the resources managed by the overlay. It should be noted that another authentication could be established at the application level, which would be independent of the overlay. In this interface we have, among others, the methods `connect` (which verifies the authentication by calling the `verifyAuthentification` method of JXTA, connects to a broker, flushes the local cache and fires an event); `configure` (which configures the local cache and is called just once at the beginning); `login` and `disconnect`.

Resource discovery and information: This interface includes functionalities related to the discovery of peers managed by the overlay. The implementation of these functionalities is later done by using JXTA discovery services. It should be noted that, as part of resource discovery, the overlay includes functionalities to discover a resource of certain desired characteristics. Thus we have, among others, the methods `discoveryEvent`, `getPeerName` and `getPeerID`.

Management of executable tasks: An important place in the primitives is given to functionalities related to the management of executable tasks in parallel and distributed applications. The TaskList module is in charge of managing the task of pending list, as shown graphically in Fig. 4.

The task functionalities are intended to give service to users/applications on top of the overlay that submit executable tasks and receive results in turn. Thus we have, among others, the following methods:

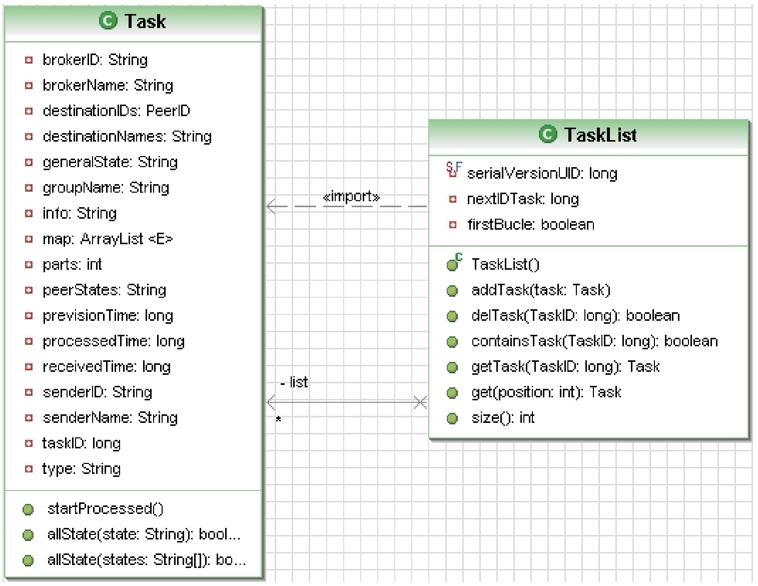


Fig. 4. Task list module.

- `executableTaskRequestRandom`: a new executable task is requested to be executed in another peer selected at random.
- `executableTaskRequestSelectedPeer`: a new executable task is requested to be executed in a (group of) selected peer(s).
- `executableTaskRequestDataEvaluator`: evaluates the execution of a task according to task data/characteristics.
- `executableTaskRequestEconomicEvaluator`: evaluates the execution of a task according to a given economic model based on task data/characteristics.
- `executableTaskDel`: deletes a task from list of pending tasks.
- `executableTaskAccepted`: indicates acceptance of a task execution in a specified resource.
- `executableTaskDeny`: indicates deny of a request for a task execution.
- `executableTaskFinished`: advertises that the execution of task has successfully been finished.
- `executableTaskCanceledRequest`: requests cancellation of task execution.
- `executableTaskCanceledByDestination`: indicates that the executable task has been cancelled at destination resource.
- `executableTaskCanceledBySender`: indicates that the executable task has been cancelled by task's sender.
- `addExecutableType`: adds a new type of executable tasks that a resource supports. Similarly there is the `delExecutableType` method.

File sharing, discovery and transmission: This includes sharing, discovery and transmission of files, which are basic functionalities of the overlay. The objective of these functionalities goes beyond the sharing in P2P systems since file transmission is necessary for submitting tasks to resources. Thus we have, among others, the following methods: `addSharedFile`, `addSharedDirectory`, `delSharedFile`, `getSharedFiles`, `sendFilePeer`, `sendFileAccepted`, `sendFileDeny`, `sendFileGroup`, `findFile`, `fileRequestRandom`, `fileRequestSelectedPeer`, `fileRequestEconomicEvaluator`, `cancelTransfer`, `localInfoTransfer`.

Instant communication: This interface supports instant communication between peers and includes methods `sendMsgPeer` (for sending a message to a specified peer) and `sendMsgGroup` (for sending a message to a peergroup).

Peer's statistics: Statistic information of resources is relevant for applications that will be built on top of the overlay. Thus we have, among others, the following methods:

- `getPeerStatistics`: returns statistic information on a specified resource.
- `getGroupStatistics`: returns statistic information on a group of resources.
- `getBrokerStatistics`: returns statistic information on a specified broker resource.
- `getBrokerStatistics`: returns statistic information on the broker of a specified group of resources.
- `getClientStatistics`: returns statistic information on a specified edge peer.
- `getClientStatistics`: returns statistic information on a specified group of edge peers.

There are also some primitives related to the peer's local cache that we have omitted here.

2) Peer types: For the definition and implementation of the JXTA-Overlay primitives, the *Broker peer* and two types of *Client peers* have been defined and implemented.

Broker peer: Broker peers are extension of rendezvous peers and are in charge to control the JXTA network of peers. As such, broker peers act as *bots* actually they don't interact with users but are permanently waiting for events/advertisements from peers in the networks. Therefore, broker peers are able to keep the state of the network and propagate it to the peers in the network. Thus, broker peer implements both rendezvous and relay peer functions. Among other proper functions of the broker peers we can mention the login control of peers, management of peergroups and rooms, task assignment and allocation, search of the best peer for file transfer, file search by content, and so on. Although not necessary, broker peers should run on fast machines in order to be able to process the amount of information generated in the network in short time intervals and maintain the updated state of the network.

GUI Client peer: This is an edge peer that offers a graphical user interface, hence called GUI Client peer, to facilitate the operability of a user with the JXTA network. The user can thus collaborate with other peers, send requests for task executions, share, send and receive files. The accomplishment of these functionalities is done through the generation of events propagated to other peers in the network.

Simple Client peer: This peer is again an edge peer, but it does not offer interaction with the user. This kind of peers is intended to increase the performance and amount of resources in the JXTA network, especially, for distributed computing applications and data storage. The functioning of such peers is completely transparent to the user. They are used by broker peers for task execution and the users do not need to be aware of such peers, although they can know the simple client peers in the network and those participating in a task resolution.

3) Transmission Control and Management in JXTA-Overlay: JXTA protocol uses Universally Unique Identifier (UUID) in order to identify peers in the private network from the Internet. We implemented a control system that is able to distinguish a peer in a private network from a peer in the Internet. The control targets are considered the network devices such as RS232C port, LPT port and USB port. We have implemented the integration of these devices in our P2P platform. Our platform is able to collect data and control the peers and all devices that are connected to the peers.

4) Graphical User Interface: JXTA-Overlay has been developed to support multi-features applications and different needs of users. It comes with a graphical user interface (through its GUIClient peer type). By using the graphical interface, users interested in parallel and distributed computing can execute their tasks through *Executable Task*, share files through *Files* or use tools for online learning through *Groupware tools* (see Subsection V-B for a snapshot).

5) Security in JXTA-Overlay: JXTA library supports basic security requirements which are desirable in any P2P-based application. Such requirements include confidentiality, integrity and availability, which are achieved through authentication, access control, encryption, secure communication and non-repudiation. In fact, JXTA provides a generic and flexible framework where different security approaches can be adopted. We have enhanced the basic security requirements of JXTA-Overlay with more advanced security mechanisms related to group membership to grant access to group resources and secure resource discovery and message exchange between peer group members [2], [3].

IV. JXTA-Overlay for High Performance Computing

One of the features of JXTA-Overlay is the support to parallel and distributed computing. Users of the JXTA-Overlay can submit the execution of their tasks to the peer nodes of the platform. This is useful to benefit from the large amount of computational resources, which are beyond those of a simple peer. Task execution in JXTA-Overlay is efficient because tasks submitted to the platform by independent peers are efficiently managed and monitored at application level. This is essentially achieved by the independence of overlay primitives from the execution of the tasks, so it is the upper

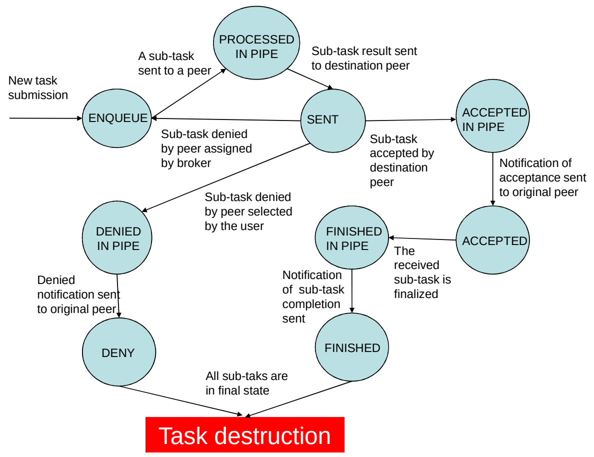

PROCESSED IN PIPE
A sub-task sent to a peer
Sub-task result sent to destination peer
New task submission
ENQUEUE
SENT
ACCEPTED IN PIPE
Sub-task denied by peer assigned by broker
Sub-task accepted by destination peer
Notification of acceptance sent to original peer
Sub-task denied by peer selected by the user
DENIED IN PIPE
FINISHED IN PIPE
ACCEPTED
The received sub-task is finalized
Denied notification sent to original peer
Notification of sub-task completion sent
DENY
FINISHED
All sub-taks are in final state
Task destruction

Fig. 5. Task execution state at Broker peer.

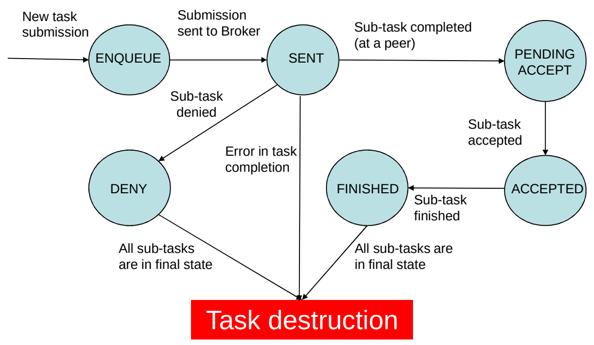


Fig. 6. Task execution state at origin peer.

layer of the applications that carries out the management of task executions. In this way, any different applications running on top of the middleware can use task primitives but manage the task execution on their own. Moreover, this consideration of independence allows that the overlay is transparent to all peers, since different peers with different types of executable tasks use the same overlay. Any executable task consists of its type, the task properly said (e.g. signed JAR –Java ARchive– file of program and/or data) and the result type. Tasks can be composed of other small tasks (sub-tasks). It should be noted that the task itself and the result are treated like objects. That is, they pass through the overlay layer neither being treated nor modified and are handled in superior levels, from where primitives for task executions are invoked (see Subsection III-1).

We show in Figs. 5, 6 and 7 the life cycle process of task executions at broker, origin peer and destination peer, respectively.

V. Integration of End-Devices into P2P Systems

The high degree of heterogeneity of computational resources is a real challenge for the today's Internet applications. The great variety of computational resources ranging from servers, PCs, laptops to hand-held and end-devices makes their integration very complex. Among other issues we could distinguish the difficulties of integration with current Internet architecture, the lack of a standard middleware that would facilitate and make transparent the programming task of the

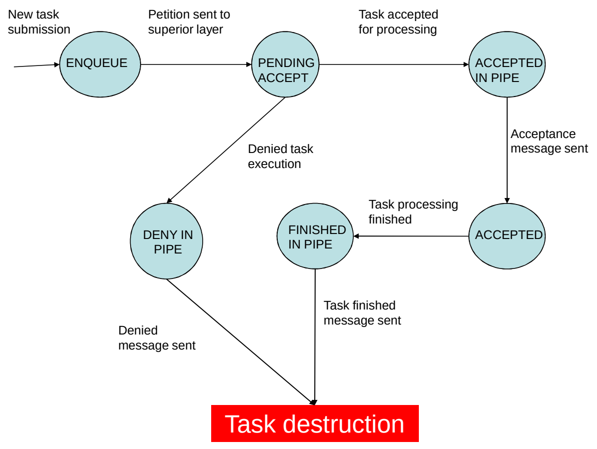


Fig. 7. Task execution state at destination peer.

diverse computational devices and, not less importantly, the security issues.

Current research work is addressing the integration of hand-held and end devices in Web applications, in Grid applications and P2P applications. The objective is to develop pervasive and ubiquitous applications using Web, Grid, P2P, smart environments and sensor technologies.

In this section, we briefly discuss some research work in the literature on the use end-devices in P2P systems. Kumar et al. [14] presented a middleware for digital rights management in P2P networks. The authors considered P2P networks comprising different types of user devices. Charas [9] introduced the concept of local policy enforcement to define terminal centric control with the aim to develop a mobile architecture including ubiquitous end user devices. Hu [12] presented techniques for NAT traversal techniques and P2P applications. Another research work in this direction is reported in [15] where a NAT traversal for Pure P2P e-Learning system is proposed.

Considerable research efforts are currently being devoted to the extension of P2P networks with mobile devices such as PDAs [13], [17], [20].

A. *The Use of End-Devices in Online Learning*

Virtual campuses are nowadays a common approach widely used for online distance teaching and learning. In fact, this approach is used not only for open universities but also in semi-open teaching and learning courses taught by different institutions and organizations world-wide.

Most of the online applications that support distance teaching and learning are Web-based. Due to the very fast development in Web technologies as well as the emergence of new paradigms such as Grid, P2P and mobile computing, the online learning systems are currently undergoing important changes. The rationale behind these changes is to shift from the "*old*" paradigm of offering remotely teaching and learning supported through virtual classrooms, to the *new* paradigm of learning and teaching "anytime, anywhere". This new paradigm of distance teaching and learning is being possible due to the everyday increase of hand-held devices such as PDAs, mobile phones as well as many types of end-devices.

The implementation of this new paradigm has certainly many benefits for online learning as compared to only Web-based online applications. Among the most remarkable features of such new online learning applications using hand-held and end devices, we could distinguish: *permanent connection with the virtual classroom*, *downloading material courses*, *awareness*, *monitoring activity in classrooms* and *alerting about important calendar dates and events*.

It should be noted however that currently the implementation of such advanced online learning systems is a challenging task due to the intrinsic complexities of the hand-held and end devices. Indeed, the programming of computational devices and their integration into distributed applications is very difficult. In such applications the target is to use different technologies: Web, Grid, P2P and computational devices. The heterogeneity of computational resources in such applications is a major research issue still to be addressed and solved for practical purposes. The first steps in this directions are done by combining Web, Grid and P2P computing and nowadays we can find some applications using such technologies. Apart from the heterogeneity and the variety of computational devices, other difficulties arise from the limitations of the computational devices and security issues.

Fortunately, libraries for supporting the programming of computational devices and their integration into distributed applications are proposed in the literature. Among these libraries, there is JXTA, which offers a set of protocols that enable the connection and communication with any type of computational devices able to be connected acting as JXTA edge peers.

B. *Groupware Tools in JXTA-Overlay Platform*

As part of JXTA-Overlay Platform, we have developed also groupware tools to support online learning. The groupware tools in the current version comprise: instant messaging, management of rooms, management of learning scenarios and task coordination among peers of a group (i.e., students of a study group) within a learning scenario (see Fig. 8) for a snapshot of the JXTA-Overlay and groupware tools.

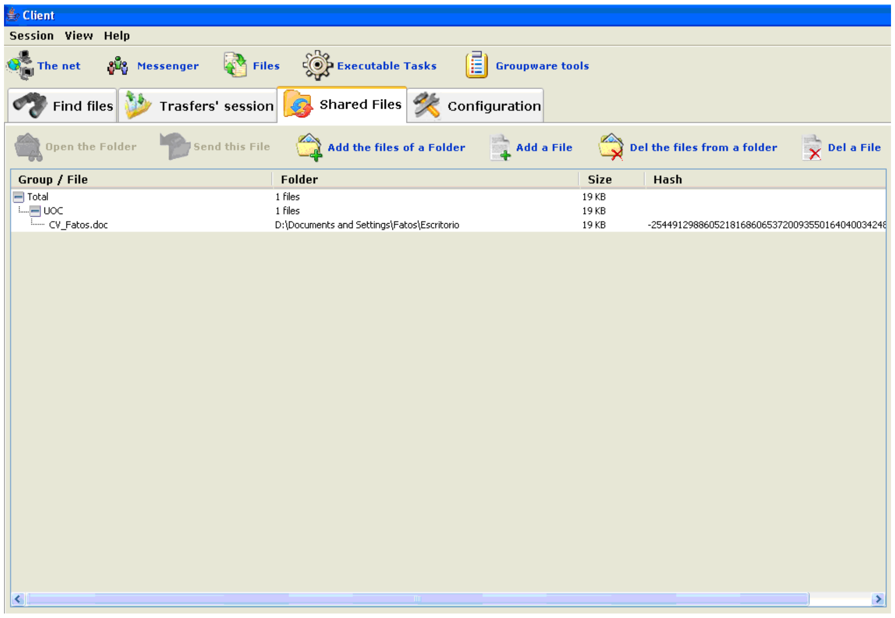
It should be noted that currently, the JXTA-Overlay can be deployed for P2P networks of *standard* peers such as PCs and laptops. In this work, we extend the capabilities of the JXTA-Overlay to support also end-devices and use them for enhancing learners' motivations.

VI. End Devices

Our target implementation is to build and design some end-devices for control in a smart environment. As end devices, we consider mobile car, robot and SmartBox, but for the sake of space we will present in this paper only SmartBox.

The SmartBox uses RFID (Radio Frequency Identification) and Vital Sensors. The size of SmartBox is $37 \times 7 \times 15$ cm. The SmartBox has the following sensors and functions (see Fig. 9).

- RFID Sensor: for identifying user's IC tag card.
- Chair Vibrator Control: for vibrating the user's chair.
- Light Control: for adjusting the room light.
- Aromatic Control: for controlling the room smell.



Group / File	Folder	Size	Hash
Total	1 files	19 KB	
UOC	1 files	19 KB	
CV_Fatos.doc	D:\Documents and Settings\Fatos\Escritorio	19 KB	-254491298860521816860653720093550164040034248

Fig. 8. Snapshot of the peer's Graphical User Interface (GUIClient) in JXTA-Overlay and groupware tools.

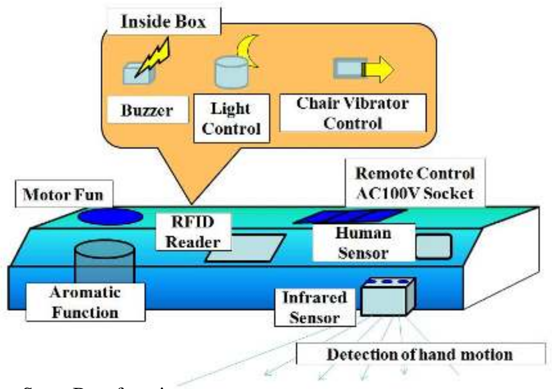


Fig. 9. SmartBox functions.

- Buzzer Control: to emit relaxing sounds.
- Remote Control Socket: for controlling AC 100V socket (on-off control).

A snapshot of the implemented SmartBox is shown in Fig. 10. The SmarBox can detect the computer users' movement by its body sensor and hand sensors. The body sensor is used for controlling the body movement of the user. On the other hand, the hand sensors control the hand motion of the user. The RFID sensor can read IC tag information and record the time a user uses the computer. We used the SmartBox as an end device in a P2P e-learning system and we control its functions by using JXTA-Overlay. We developed a control system for controlling the SmartBox (see interface in Fig. 11).

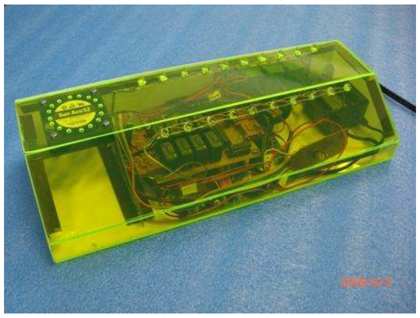
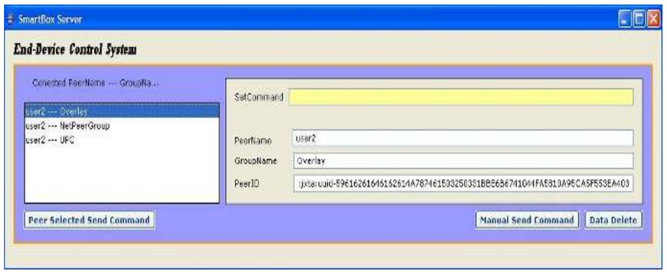


Fig. 10. A snapshot of implemented SmartBox.


SmartBox Server
End-Device Control System
Conected PeerName --- GroupNa...
Peer Selected Send Command
Manual Send Command
Data Delete

user2 --- Overlay
user2 --- NetPeerGroup
user2 --- UPC

SetCommand
PeerName: user2
GroupName: Overlay
PeerID: :jxta:uuid-59616261646162614A787461503250331BBE6B6741044FA5810A95CA5F553EA403

Fig. 11. SmartBox control system interface.

VII. Experimental Results

We have conducted extensive experimental evaluations of the JXTA-Overlay middleware. Here, we present some computational results from the experimental evaluation of two aspects: speed-up gain in massive processing of large size log data files and end-device integration.

A. *Experimental Results with JXTA-Overlay Performance*

In order to evaluate the performance of JXTA-Overlay primitives for massive processing computations, we deployed the P2P network using nodes of the PlanetLab platform [16] and used the cluster nozomi.lsi.upc.edu (a main control node + five computing nodes) where we deployed broker services. The computational results of the experimental study given below were obtained using a group of 8 geographically distributed machines in seven EU countries. The distributed application consisted in processing a log file of 100 Mb from a real virtual campus. In this application, a GUIClient peer (that is a peer with graphical user interface), provides the required data for

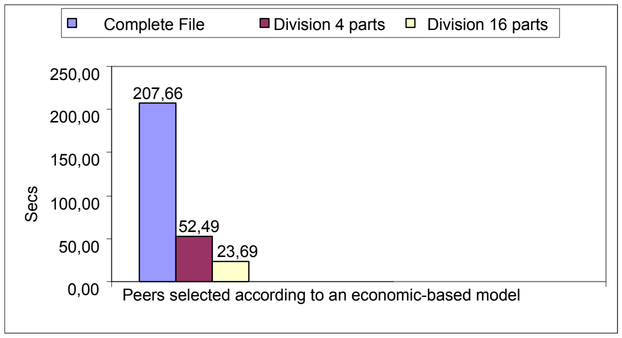


Fig. 12. Total processing time of all chunk files at peer nodes.

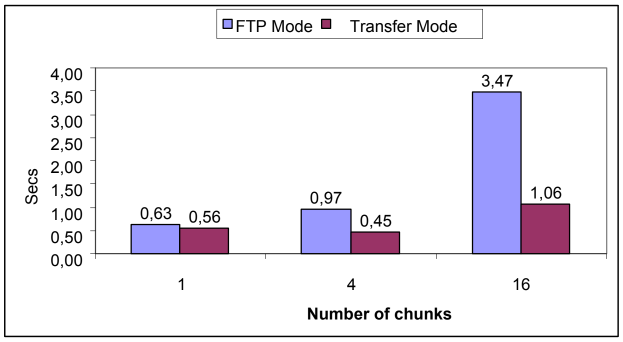


Fig. 13. Final processing time at GUIClient peer.

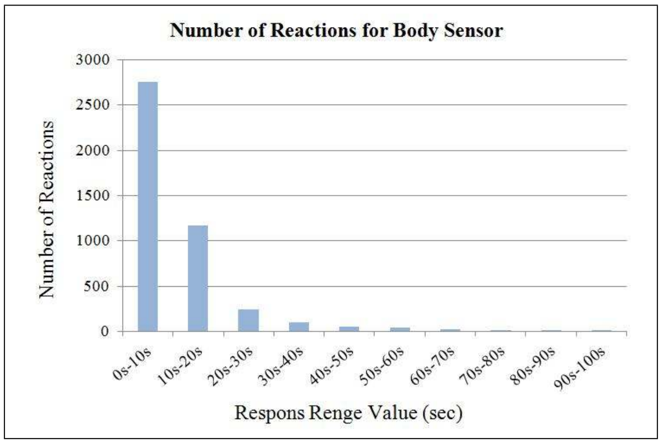


Fig. 14. Reaction number for body sensor.

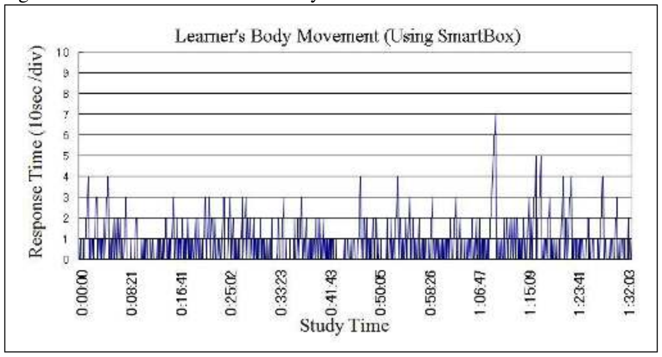


Fig. 15. Measurement of learners concentration using SmartBox.

the application including the file to be processed, number of chunks to split the file and transfer mode (FTP or direct peer transfer). The application starts by splitting the log file into a specified number of parts (chunks) of equal size, the file chunks are sent to an FTP site or directly to peer nodes for processing and results of peer nodes are uploaded to an FTP site or are directly sent to the GUIClient peer, which processes them and produces a final result.

Below we show graphical representation of the two main steps, namely: measuring of the processing time of the assigned chunks by peers (see Fig. 12), and the final processing time at GUIClient peer which merges the results sent by peers (see Fig. 13).

As can be seen in Fig. 12, a considerable speed-up in processing of the log file is gained when the file is split in 4 and 16 parts, respectively, with respect to the sequential processing of the complete file. Thus, the processing time when the file is split in 16 chunks is roughly twice as faster as the processing time when file is split in 4 chunks. It should be noted here that economic-based models were used to select peers in order to achieve a good load balancing.

In Fig. 13, we show the processing time of merging partial results of peers in a final processing result. We measured whether it is more efficient to download the partial results of peers from an FTP address or get them in a direct communication with the peer (P2P mode). As can be seen, using direct communication resulted more efficient. For instance, using direct communication when file is split into 16 chunks is roughly three times faster as compared to FTP mode transfer.

B. Experimental Results with SmartBox

We present here some experimental results about the monitoring and performance of the SmartBox in the JXTA-Overlay P2P system. We carried out experiments in real environment and confirmed the effectiveness of JXTA-Overlay. We verified the effectiveness of SmartBox by stimulating different users with its functions.

The proposed system can detect the learner's movement by using body sensors. The measurement data for learners' body movements are shown in Fig. 14. We obtained these data after observing learners studying for a total of 40 hours distributed along different periods.

In our experiments, we used SmartBox and measured the stimulation effects that the SmartBox has to the learners. We checked the Smell function, Light function (high luminance LED) and Sound function (different kind of music). In order to check the effects of the SmartBox on the learners, we carried out real experiments with learners while they were studying. In the first experiment, we used the SmartBox and in the second experiment we did not use the SmartBox. The learner's body movement for these two cases are shown in Fig. 15 and Fig. 16, respectively. The comparison between these two figures shows that the use of SmartBox is an effective way to improve the learner motivation, because the learner's concentration is higher using the SmartBox.

VIII. Conclusions

P2P systems have evolved from simple systems of file sharing among Internet users to a disruptive technology for

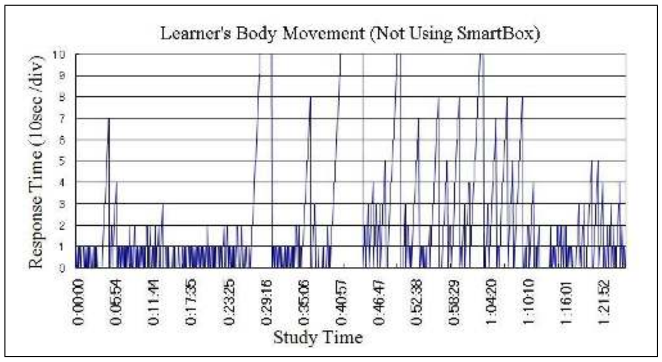


Fig. 16. Measurement of learners concentration without using SmartBox.

collaborative and social activities. Such systems are capable to deliver content, profiling, grouping and control the ordinary users in intelligent and interactive environments.

In this work, we presented the JXTA-Overlay, a JXTA-based P2P platform designed with the aim to leverage capabilities of Java, JXTA and P2P technologies to support distributed and collaborative systems in a decentralized and self-organized manner, capable to integrate different types of peers. The platform can be used not only for efficient and reliable distributed computing but also for collaborative activities and ubiquitous computing by integrating in the platform also end-devices.

Acknowledgements

This research work is supported by Japanese Society for the Promotion of Science (JSPS). Leonard Barolli would like to thank JSPS for the financial support.

Fatos Xhafa's research work completed at Birkbeck, University of London, on Leave from Technical University of Catalonia (Barcelona, Spain). His research is supported by a grant from the General Secretariat of Universities of the Ministry of Education, Spain.

References

[1] G. Antoniu, P. Hatcher, M. Jan, and D.A. Noblet. Performance evaluation of JXTA communication layers. In *Proc. of the 5th IEEE International Symposium on Cluster Computing and the Grid (Ccgrid'05)*, vol. 1. pp. 251-258, 2005.

[2] J. Arnedo-Moreno, K. Matsuo, L. Barolli, F Xhafa. A Security-aware Approach to JXTA-Overlay Primitives. In *ICPP-2009 Proc., ADPNA-2009 Workshop*, pp. 431-436, 2009.

[3] J. Arnedo-Moreno, K. Matsuo, L. Barolli and F. Xhafa. A Security Framework for JXTA-Overlay. In Proc. of NBiS-2009 Conference, pp. 212-219, 2009.

[4] L. Barolli, F. Xhafa, A. Durresi, G. De Marco. M3PS: A JXTA-based Multi-platform P2P System and Its Web Application Tools. *International Journal of Web Information Systems*, vol. 2 (3/4), pp. 187-196, 2006.

[5] G. Briscoe and P. De Wilde. Digital Ecosystems: Evolving service-orientated architectures. In *Proc. of the 1st International Conference on Bio Inspired Models of Network, Information and Computing Systems*, vol. 275, Article No. 17, 2006.

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

[7] J. Buford, H. Yu, E. Lua. *P2P Networking and Applications*. Elsevier, 2008.

[8] E. Chang, M. Quaddus and R. Ramaseshan. The Vision of DEBI Institute: Digital Ecosystems and Business Intelligence. Technical Report of DEBII, 2006.

[9] P. Charas. Peer-to-Peer Mobile Network Architecture. In *Proc. of the 1st International Conference on Peer-to-Peer Computing*, pp. 55-61, 2001.

[10] H. Dong, F. Hussain, E. Chang. A Service Search Engine for the Industrial Digital Ecosystems. to appear in *IEEE Trans. on Industrial Electronics*, vol. 57, 2010.

[11] E. Halepovic, R. Deters, and B. Traversat. Performance Evaluation of JXTA Rendezvous. In *Proc. of International Symposium on Distributed Objects and Applications*, pp. 1125142, 2004.

[12] Z. Hu. NAT Traversal Techniques and Peer-to-Peer Applications. In *Proc. of the New Zealand Comp. Sci. Research Student Conference*, pp. 242-245, 2008.

[13] T. Iwata, S. Miyazaki, M. Takemoto, K. Ueda and H. Sunaga. P2P Platform Implementation on PDAs Organizing Ad Hoc Wireless Network. In *Proc. of Workshops of the Symposium on Applications and the Internet*, pp. 568-573, 2004.

[14] P. Kumar, G. Sridhar, V. Sridhar, and R. Gadh. DMW - A middleware for Digital Rights Management in Peer-to-Peer Networks. In Proc. of the DEXA-2005 Workshops, pp. 246-250, 2005.

[15] K. Kuramochi, T. Kawamura, K. Sugahara. NAT Traversal for Pure P2P e-Learning System. In *Proc. of 3rd International Conference on Internet and Web Applications and Services*, pp. 358-363, 2008.

[16] PlanetLab Platform. http://www.planet-lab.org/.

[17] M.K. Purvis, N. Garside, C. Stephen, M. Nowostawski, M. Oliveira. Multi-agent System Technology for P2P Applications on Small Portable Devices. In *Proc. of 3d International Workshop on Agents and P2P Computing*. Springer Verlag, LNCS vol. 3601, pp. 153-160, 2005.

[18] T. Karagiannis, A. Broido, M. Faloutsos, K. Claffy. Transport Layer Identification of P2P Traffic. In *Proc. of the 4th ACM SIGCOMM Conference on Internet Measurement*, pp. 121-134, 2004.

[19] A. Oram, Peer-to-Peer: Harnessing the Power of Disruptive Technologies. *O'Reilly Publishing*, 2001.

[20] M. Tuisku. Wireless Java-enabled MIDP Devices as Peers in a Grid Infrastructure. Springer Verlag, LNCS vol. 2970, pp. 273-281, 2004.

[21] A. Waluyo, W. Rahayu, D. Taniar, B. Srinivasan. A Novel Structure and Access Mechanism for Mobile Broadcast Data in Digital Ecosystems. to appear in *IEEE Trans. on Industrial Electronics*, vol. 57, 2010.

[22] F. Xhafa, R. Fernandez, T. Daradoumis, L. Barolli, S. Caballé. Improvement of JXTA Protocols for Supporting Reliable Distributed Applications in P2P Systems. Springer Verlag, LNCS, vol. 4658, pp. 345-354, 2007.

[23] F. Xhafa, L. Barolli, T. Daradoumis, R. Fernandez and S. Caballé. JXTA-Overlay: An Interface for Efficient Peer Selection in P2P JXTA-based Systems. *International Journal on Computer Standards & Interfaces*, vol. 31, no. 5, pp. 886-893, 2009.