Multifeatured and Integrated P2P Platform for Distributed Networking Application Using Juxtapose

H.Vignesh Ramamoorthy¹, B.Sabarigiri², Dr.D.Suganya Devi³

¹Research Scholar of Computer Science – Director of MCA,
Sree Sarwath Thyagaraja College, Pollachi-642107
Coimbatore, India

²Research Scholar of Computer Science
PSG College of Arts and Science, Coimbatore-641014, India

Abstract — There is an increasing need to design and deploy multifeatured networking applications due to the fast growth of the Internet infrastructure and the use of large-scale complex applications in industries, transport, logistics, government, health, and businesses. Important features of such applications include the capability to be self-organized, be decentralized, integrate different types of resources (personal computers, laptops, and 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 Juxtapose (JXTA)-Overlay, which is a JXTA-based peer-to-peer (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 a user interface as well as security issues are also tackled.

Keywords — End-device control, Java applications, Juxtapose (JXTA) library, JXTA-Overlay, peer-to-peer (P2P) systems.

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 multifeatured 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 are emerging as a paradigm for supporting multidisciplinary and multi paradigmatic applications that are 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 the integration of different types of resources [personal computers (PCs), laptops, and mobile and sensor devices] providing global, transparent, and secure access to resources. Supporting various forms of collaborative activities, such as business, online 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. 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. Moreover, in large-scale networks such as the 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. Indeed, P2P systems 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. 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 personal digital assistants (PDAs) to PCs and servers.

In this paper, we present the JXTA-Overlay, which is a Juxtapose (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 and thus overcoming intrinsic difficulties of current Internet architecture and protocols. Moreover, the design of an advanced user interface, as well as the enhancement security requirements of JXTA library, is also tackled.

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 massive processing computations is presented in Section IV. We introduce the issue of integration of end devices into P2P systems in Section V. The paper ends with some conclusions in Section VI.

II. JXTA LIBRARY FOR P2P COMPUTING

JXTA technology is a generalized group of six Extensible Markup Language (XML)-based protocols that allow different types of peers to communicate and collaborate among them. Peers can be organized into peer groups 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 a 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 network address translations (NATs) or when different communication protocols are used. The performance of JXTA has been evaluated in several research works, and it has been shown that the library is efficient and highly scalable. Peers are uniquely identified by allowing them to change their address but still conserving their unique peer identification (ID).

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 these protocols (core and standard protocols) as follows.

1) 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.

2) 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.

3) 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 peer group, a pipe, or any other resource that has associated an advertisement.

4) 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.

5) 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.

6) 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, input and output pipes 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 Hypertext Transfer Protocol (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.

1) Peer: Any interconnected node is called a 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, complete edge, rendezvous, and relay peers) according to their role in the P2P network.

a) Rendezvous: The rendezvous peers are in charge of coordinating other peers in the JXTA network. In addition, they provide the necessary services for the propagation of messages. Each subnetwork of JXTA must have at least a rendezvous peer.

b) Relay: The relay peers allow peers behind firewalls, NATs, or special peers having limited computational power, such as mobile devices, PDAs, etc., to be part of a JXTA network. The relay peers achieve this by using protocols which allow one to cross the limitations imposed by these systems, e.g., the HTTP protocol.

c) 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.

2) 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.

3) 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.

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

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

III. JXTA OVERLAY PLATFORM

JXTA-Overlay 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 the following: 1) peer discovery; 2) peer resource discovery; 3) resource allocation; 4) task submission and execution; 5) file/data sharing, discovery, and transmission; 6) instant communication; 7) peer group functionalities (groups, rooms, etc.); and 8) 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 communication of peers in JXTA and the structure of JXTA overlay is shown in figure 1 and 2 respectively.

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. The architecture of the P2P distributed platform that we have proposed 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.

A. Primitives

The set of primitives includes functionalities that allow peer discovery, peer's resource 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. The main functionalities of these interfaces are given below.

1) 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 verifyAuthentication 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), and login, and disconnect.

2) 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.

3) 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. 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.

- `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 the list of pending tasks.
- `executableTaskAccepted` indicates acceptance of a task execution in a specified resource.
- `executableTaskDenied` indicates denial 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 the destination resource.
- `executableTaskCanceledBySender` indicates that the executable task has been cancelled by the task's sender.
- `addExecutableType` adds a new type of executable tasks that a resource supports. Similarly, there is the `delExecutableType` method.

4) 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`, and `localInfoTransfer`.

5) Instant Communication: This interface supports instant communication between peers and includes methods `sendMessagePeer` (for sending a message to a specified peer) and `sendMessageGroup` (for sending a message to a peer group).

6) 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.

B. Peer Types

For the definition and implementation of the JXTAOverlay primitives, the Broker peer and two types of Client peers have been defined and implemented.

1) 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 do not 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 log-in control of peers, management of peer groups 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.

2) Graphical User Interface (GUI) Client Peer: This is an edge peer that offers a GUI, 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, and share, send, and receive files. The accomplishment of these functionalities is done through the generation of events propagated to other peers in the network.

3) 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, particularly 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.
C. Transmission Control and Management in JXTA-Overlay

JXTA protocol uses Universally Unique Identifier 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, Line Print Terminal, and Universal Serial Bus ports. 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.

D. GUI

JXTA-Overlay has been developed to support multifeatures applications and different needs of users. It comes with a GUI (through its GUI Client 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.

E. 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 can enhance 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.

IV. JXTA – OVERVIEW 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. The 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; thus, it is the upper 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 the overlay to be 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 Java ARchive file of program and/or data), and the result type. Tasks can be composed of other small tasks (subtasks). 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.

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, and laptops to handheld 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 diverse computational devices, and, not less importantly, the security issues. A current research work is addressing the integration of handheld and end devices in Web, Grid, and P2P applications.

A. Use of End Devices in Online Learning:

Virtual campuses are nowadays a common approach that is widely used for online distance teaching and learning. 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. This new paradigm of distance teaching and learning is being possible due to the everyday increase of handheld devices, such as
PDAs and 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 handheld 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.

VI. CONCLUSION

P2P systems have evolved from simple systems of file sharing among Internet users to a disruptive technology for collaborative and social activities. Such systems are capable to deliver content, profiling, grouping, and control to the ordinary users in intelligent and interactive environments. In this paper, we have presented the JXTA-Overlay, which is 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.

REFERENCES