JXTA security in mobile constrained devices

Marc Domingo-Prieto, Joan Arnedo-Moreno
Estudis d'Informàtica, Multimèdia i Telecomunicació
Universitat Oberta de Catalunya
Barcelona, Spain
mdomingopr, jarnedo@uoc.edu

Jordi Herrera-Joancomartí
Escola Tècnica Superior d'Enginyeria
Universitat Autònoma de Barcelona
Campus de Bellaterra, Spain
jherrera@deic.uab.cat

***Abstract*—JXME is the JXTA protocols implementation for mobile devices using J2ME. Two different flavors of JXME have been implemented, each one specific for a particular set of devices, according to their capabilities. The main value of JXME is its simplicity to create peer-to-peer (P2P) applications in limited devices. In addition to assessing JXME functionalities, it is also important to realize the default security level provided. This paper presents a brief analysis of the current state of security in JXME, focusing on the JXME-Proxied version, identifies existing vulnerabilities and proposes further improvements in this field.**[1]

Keywords: peer-to-peer, security,Peer Group, JXTA, distributed systems, Java, JXME, J2ME.

I. Introduction

Peer-to-peer (P2P) networks allow peers to provide and consume services in a collaborative way. Examples of this services are content sharing, processing and messaging. In this kind of network, it is assumed that all peers have equivalent capabilities [1], as well as a high degree of decentralization and autonomy. Such environments have become highly popular in recent times due to its great potential to scale and the lack of a central point of failure. Just like many kinds of applications have evolved from the desktop to mobile environments, it was natural that P2P systems would follow the same steps [2], as both architectures are based on node autonomy and decentralization.

JXTA [3] is a set of open protocols specifications that enables the creation and deployment of P2P networks. Using the JXTA protocols, peers can communicate, publish resources and find and consume remote resources, independently of the actual transport layer and the implementation language. Currently, most efforts to provide a JXTA reference implementation have been made on a fully based desktop environment [4], [5]. However, JXTA has also shifted to the mobile environment in the form of the JXTA Micro Edition (JXME) project [6], which allows mobile devices to participate in a JXTA network. JXME heavily takes into account the idiosyncrasies of mobile devices such as power and storage limitations, and for that reason research has focused on these features [7]. However, security is a very important feature that often has been forgotten in JXME research.

[1]This work was partially supported by the Spanish MCYT and the FEDER funds under grant TSI2007-65406-C03-03 E-AEGIS and CONSOLIDER CSD2007-00004 "ARES", funded by the Spanish Ministry of Science and Education.

The main goal of this paper is to analyze JXME security in order to determine the basic security baseline that this platform provides to current P2P applications. The security analysis performed in this paper follows the idea of [8] where a generic JXTA security survey has been presented. Applying the same methodology, security is not analyzed by reviewing basic peer operations in an isolated manner, but taking into account the whole peer life cycle. With this approach, it is possible to identify the available security mechanisms and how they operate, which may prove useful for application developers.

The paper is organized as follows. Section II provides an overview of the JXME project in order to understand its main characteristics and differences regarding JXTA. Section III presents the security analysis of the most tested JXME version, JXME-Proxied. This analysis includes related research about JXME security and an overview of the standard peer operation cycle. Section IV provides a brief comparison between both versions of JXME. Finally, Section V outlines the conclusions and further work.

II. Overview of JXME

JXME is very similar to the desktop version of JXTA, since they share the same basic specification. In both JXTA and JXME, the basic organizational foundation is the *Peer Group*, a set of peers with common interests which agree on common services. Peer Groups offer a private context to publish and access different services. Peers exchange data using *pipes*, which provide an asynchronous unidirectional communication channel. All resources (Peers, Peer Groups, Pipes, etc.) are described using *Advertisements*, metadata documents exchanged between peers using JXTA protocols. A network resource cannot be accessed without previously recovering its associated Advertisement. A detailed explanation of JXTA's generic protocols and services can be found in [9], [10].

JXME can be viewed as a JXTA compatible platform on resource constrained devices which uses the framework specifications for Java ME *Connected Device Configuration* (CDC) and *Connected Limited Device Configuration* (CLDC). CDC uses the C-Virtual Machine (CVM), an optimized version of the Java Virtual Machine (JVM) adapter for mobile devices, it contains some of the standard Java packages and it is addressed to powerful PDA's and smart phones. On the other hand, CLDC uses the Kilobyte Virtual Machine (KVM) [11], has

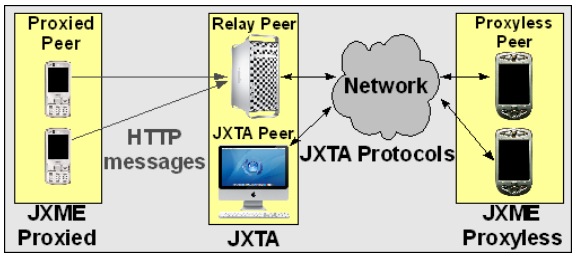

Fig. 1. JXTA and JXME network architecture

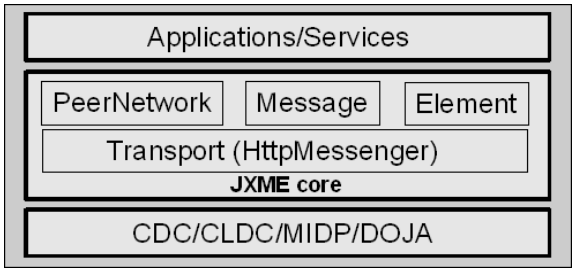

Fig. 2. JXME main components

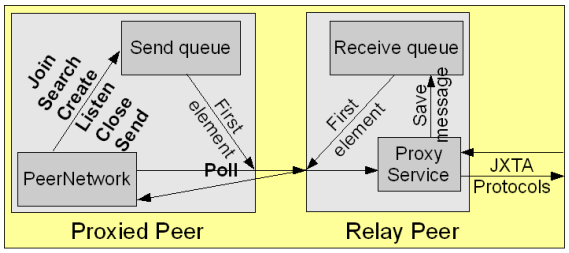

Fig. 3. Proxied communication scheme

few of the standard Java packages and it is addressed to small devices with very slow processors and very reduced memory. CLDC has two profiles which define its operation mode: *Mobile Information Device Profile* (MIDP) and *DOcomo JAva* (DOJA). The former is a specification for the usage of Java on embedded devices and the latter is a Java environment specification for DoCoMo's i-mode mobile phone.

Two distinct versions of JXME have been developed in order to accommodate to a broad set of scenarios, like for instance assuming that the system should provide different methods of peer participation in the JXTA network or having in mind the reduced device capabilities of the peers. On one hand, the *JXME-Proxied* version is a simple implementation for very limited devices, which delegates all work to an external super-peer. On the other hand, the *JXME-Proxyless* version is a more complex one, with CDC devices in mind, where mobile peers may directly interact the JXTA network. In this paper we focus in the JXME-Proxied version since it is the most tested and used of both. However, we will provide some insights on JXME-Proxyless in Section IV.

A. JXME-Proxied operation

The JXME-Proxied version, or Proxy-based, is the simplest and the oldest one. It has been implemented for both CDC and CLDC devices, as well as for the MIDP and DOJA profiles. Devices which use this version are named *Proxied Peers* in the JXTA network and, since they are assumed to have very limited resources, cannot directly communicate with other peers within the JXTA network. All messages are exchanged through a *Relay Peer*, a special kind of super-peer which implements the Relay and Proxy JXTA services. The communication between the Proxied and Relay Peer is performed with a simplified protocol based on HTTP. By default, a single Relay Peer can support up to 150 Proxied Peers. Figure 1 shows the JXME network architecture inside JXTA network.

The main responsibilities of the Relay Peer on regards to its Proxied Peers are:

- Listen to and answer requests from the Proxied Peer.
- Translate messages received from the Proxied Peer to XML and then sending them to the JXTA network.
- Save messages from the JXTA network for the Proxied Peer.
- Summarize and translate XML messages from the JXTA network into a simple format which the Proxied Peer is able to understand.

Figure 2 shows the components of the JXME-Proxied version. The main module is PeerNetwork, that specifies the operations a Proxied Peer can execute in the JXTA network. These operations are **Join** a group, **Search** or **Create** resources (such as Peer Groups or pipes), **Listen** to a pipe to receive data, **Send** data to a specific pipe, **Close** a pipe and **Poll** the Relay Peer for messages from the JXTA network that have the Proxied Peer as the final receiver.

When an operation is executed, a Message, formed by a set of Elements, is created. HttpMessenger provides a messaging service used in the HTTP exchange with the Relay Peer. To reduce the number of messages sent to the Relay Peer, they are stored in a queue at the Proxied Peer and each time a poll operation is performed, the first message in the queue is actually sent to the Relay Peer. Figure 3 shows the communication scheme between a Proxied Peer and its Relay Peer.

Once the main features of JXME-Proxied version are described, it is obvious that the need for Relay Peers is the main design limitation of this approach. Notice that the Relay Peer architecture, that could be ideally considered as an hybrid P2P system, becomes a client-server model when we reduce the scope to a local environment, with one Relay Peer and many Proxied Peers. Furthermore, if a Proxied Peer wants to simultaneously use more than one Relay Peer, it will be assigned a different PeerId for each Relay Peer. As a result, a single Proxied Peer will effectively be considered as several different peers within the network.

B. Related work

Some research exists on JXTA features [12] and security [8], but not many efforts have been made for JXME specifically. For instance, in [7] an analysis about JXME functionality and

performance can be found. Another example is [13], were an authentication group membership has been developed. This authentication is provided by the Relay Peer, which uses an external Sign and LDAP Server, breaking completely the P2P model proposed by JXTA.

JXME has also been analyzed and used to build a broad set of applications. For instance in [14] an analysis to build a Video On Demand (VOD) application over JXME is performed, presenting its main architectural design. The specification and implementation of a simple chat demo can also be found in [15]. In [16] an analysis of requirements for P2P file sharing systems using JXME is performed, concluding that JXME-Proxied does not achieve the desired functionalities, but they are achieved by JXME-Proxyless. Finally, an application to chat and share music is developed using JXME in [17].

III. JXME-Proxied security assessment

Guaranteeing a minimum security level should be one of the main goals in most of the current P2P applications even though this level may differ depending on the particular needs of each application. In this section, a security analysis of JXME-Proxied is made in order to know the security level provided by the platform itself. This analysis follows the methodology used in [8], where a general peer life cycle is examined rather than isolate peer actions. However, in this paper we have focused on the communication between the Proxied and Relay Peer, also taking into account the way that a Relay Peer stores and manages his subscribed Proxied Peers. Security between the Relay Peer and the JXTA network is not analyzed, since it falls into general JXTA security, and it is out of the scope of this paper.

A. JXME-Proxied standard peer operation cycle

The standard JXME-Proxied general operation cycle includes in the same steps as in JXTA: Platform startup, Peer Group joining, Resource discovery and publication, Message exchange and Disconnection. However, internally, some operations work in a different way due to the particular communication scheme between the Proxied and Relay Peer. A brief description of each step follows:

1) **Platform startup**: This is the first action performed by a JXTA Peer and consists in loading the required libraries. In JXME, the Proxied Peer also has to connect to any Relay Peer.
2) **Peer Group joining**: At this step, the peer joins a Peer Group, so interaction with other Peer Group members is possible. Peer Group joining is managed by the Membership Service, one of JXTA's core services, which allows peers to claim identities within a Peer Group.
3) **Resource discovery and publication**: Encompasses the distribution of Advertisements, XML documents which describe a resource in a Peer Group and how to access it. A resource cannot be accessed without previously retrieve its Advertisement.
4) **Message exchange**: This is the most frequently action performed by Proxied Peers. To exchange messages a Proxied Peer has two operations: listen and send.
5) **Disconnection**: It is the last operation a peer performs in order to exit the JXTA network and disconnect.

B. Most common attacks in P2P networks

A security evaluation must be performed once the most common attacks have been identified. In [18], these attacks are described and classified into two large groups: passive and active. Each group can be further classified according to the particular action performed by the attacker.

- **Passive attacks**: The attacker just monitors peer activity.
  - *Eavesdropping* (Evs): Capture traffic searching sensitive information, such as passwords.
  - *Traffic analysis* (TAn): Capture traffic and analyze it searching for patterns.
- **Active attacks**: The attacker performs actions on the messages, such as the deletion of data.
  - *Spoofing* (Spf): Impersonate another peer.
  - *Man-in-the-middle* (MitM): Intercept the communications between two peers, relaying messages transparently to them.
  - *Playback* (Pb) or *Replay* (Rp): Intercept data from one peer so it can be reused at a later time, simulating a real exchange.
  - *Local data alteration* (LDA): Modify local data to corrupt the system behavior.
  - *Software Security Flaws* (SSF): Vulnerabilities due to bugs in the source code.

C. JXME-Proxied Security evaluation

Once the default peer operation cycle is established and the possible attacks are identified, it is possible to proceed with the security evaluation. Basically, JXME-Proxied's main design goal is to minimize the consumption of device resources. Therefore, it is expected that Proxied Peers have a very limited security capabilities. However, since Proxied Peers delegate JXTA communications to the Relay Peer, it is also expected that Relay Peers will manage a basic security model.

1) Platform startup: Before a Proxied Peer may join the JXTA network, a new *peer identifier* (PeerId) must be generated. The standard protocol specifies that it is the Relay Peer who generates the PeerId and sends it back to the Proxied Peer. However, in the actual implementation, Proxied Peers can generate their own PeerId and freely connect to the Relay Peer with this new identity. Because of this initial communication protocol, Proxied Peers are very vulnerable at the startup because his PeerId is transmitted in clear text over the Network, allowing an attacker to learn it (*Eavesdropping*).

The PeerId generation process is very important since the Relay Peer is only able to identify Proxied Peers by their PeerId. However, since no authentication exists between a Proxied and Relay Peer, an attacker can act as an invisible intermediate Relay Peer, redirecting HTTP messages (*Man-in-the-Middle*). In this operation, reusing intercepted data

jxmg	0	01	05	proxy	06	
jxel	2	0	07	request	0004	join
jxel	2	0	02	id	xxxx	peer group id
jxel	2	0	03	arg	xxxx	**password**
jxel	2	0	09	requestId	0001	1
jxel	1	0	26	DestAddr	xxxx	destination address
jxel	1	0	21	SrcAddr	xxxx	source address

TABLE I
JOIN MESSAGE

```
GET /unknown-unknown?0,-1,http://172.16.0.37:2481/
EndpointService:jxta-NetGroup/
uuid-DEADBEEFDEAFBABAFEEDBABE0...0F05/pid HTTP/1.1
Connection: close
Content-Length: 0
User-Agent: UNTRUSTED/1.0
Host: 172.16.0.37:2481
```

TABLE II
PEER IDENTIFIER REQUEST

(*Playback*) has no sense, because each peer start the platform only once.

There is an initial authentication mechanism, outside JXTA's core, described in [13]. This mechanism provides a unidirectional authentication, only the Proxied Peer authenticates the Relay Peer. However, this authentication is provided using a central Sign Server and LDAP database, which break the P2P model followed by JXTA.

Finally, no security model has been considered to identify correct JXME binary releases to another one with malicious code (*local data alteration*).

2) Peer Group joining: In the process to become a Peer Group member, the main restriction is that a JXME Peer can only join to Peer Groups which implements the *None Membership Service*, the default Membership Service in JXTA. It is used in groups without authentication, where any peer can claim any identity, designed for applications with no security concerns. Since the information in transmitted messages is sent, in clear text, an attacker can know the group any Proxied Peer is trying to join (*Eavesdropping*) and identify important peers by its traffic (*Traffic analysis*). Due to the Membership Service used is the None Membership Service, an attacker can impersonate any other peer inside a group by claiming his peer's identity within the Peer Group (*Spoofing*). However, in the join operation, reusing a directly captured message (*Playback*) is pointless because a Proxied Peer can only join once to a Peer Group. Therefore, replay attacks are not a concern.

Currently, there is an initial implementation of a Membership Service based on passwords, the *Passwd Membership Service*. However, as mentioned in the JXTA documentation, it was designed only for testing, since passwords are still transmitted in clear text across the network allowing *Eavesdropping* , *Traffic analysis*, *Spoofing* and *Man-in-the-middle* (see Table I). Again, this allows an attacker to impersonate any other peer or join his groups. Furthermore, since in JXME-Proxied this password is used only in the join operation, this approach is pointless, as far as any other operation in the Peer Group is concerned.

Another implementation of a Peer Group Membership Service which is external to the JXTA specification is developed in [13], where an authentication mechanism is designed between the Proxied Peer and the Peer Group. This is a bidirectional authentication. However, as mentioned before, the authentication mechanism is based in a central Sign Server and LDAP database, breaking the P2P model.

3) Resource discovery and publication: Advertisements are transmitted without encryption and can be easily intercepted (*Eavesdropping*) by an attacker, which can recognize important peers analyzing its traffic (*Traffic analysis*). An attacker can also publish/discover false resources (*Spoofing*) and modify/delete Advertisements (*Man-in-the-middle*) of any Proxied Peer since no authentication is enforced. Furthermore, an attacker can resend captured messages performing several times the original operation (*Replay*) in order to produce multiple resource discovery queries. This attack may saturate the Proxied Peer since all the response messages to these queries will be, finally, received and processed by the Proxied Peer.

4) Message exchange: In JXTA, network messages are exchanged using pipes, briefly introduced in Section II. One important feature in JXTA are secure pipes, which allow peers to send messages using a secure communication channel. Unfortunately, JXME Peers are not be able to use pipes between the Proxied Peers and the Relay Peers (although the Relay Peer will manage the pipes to connect to the JXTA services on behalf of the Proxied Peer). The communication between a Proxied and Relay Peer is performed using HTTP. An example of a HTTP message can be seen in Table II.

Due to this functionalities, different attacks can be performed: *Eavesdropping*, *Traffic analysis*, *Spoofing*, *Man-in-the-middle* and *Replay*. These attacks mainly allow an attacker to send/receive and sniff messages, impersonating any peer. Moreover, as it is described in the next section, an attacker can close legitimate peer pipes at leisure, ending the message exchange.

5) Disconnection: Before a peer may disconnect, all pipes must be closed. However, the main limitation at this step is that a Proxied Peer has no operation designed to the disconnection purpose. It is the Relay Peer who has to decide when to unsubscribe a Peer or a Peer Group. Proxied Peers can only perform an operation to close a pipe by knowing its id.

Since a Proxied Peer has not an specific operation to inform the Relay Peer about to its total disconnection, any Proxied Peer continues being vulnerable even when it is disconnected. Since the pipe disconnection operation follows the same communication scheme as the previous operations, it also inherits their vulnerabilities allowing the following attacks: *Eavesdropping*, *Traffic analysis*, *Spoofing*, *Man-in-the-middle* and *Replay*. These attacks mainly allow an attacker to impersonate any peer, stealing his open pipes, opening new pipes, preventing pipe disconnection, using all his previous

Operation/Threat	Evs	TAn	Spf	MitM	Rp	LDA	SFF
Startup	V(2)	N/A	V(4) - P(TGMS)	V(2, 4) - P(TGMS)	N/A	V(1)	P(OSS)
Join	V(2)	V(3)	V(4) - P(pass) - P(TGMS)	V(2, 4) - P(pass) - P(TGMS)	N/A	N/A	P(OSS)
Publish/Discover	V(2)	V(3)	V(4)	V(2, 4)	V(4)	N/A	P(OSS)
Messaging	V(2)	V(3)	V(4)	V(2, 4)	V(4)	N/A	P(OSS)
Disconnect	V(2)	V(3)	V(4)	V(2, 4)	N/A	N/A	P(OSS)

TABLE III
JXME-PROXIED PEER OPERATION CYCLE SECURITY SUMMARY
(N/A: NON-APPLICABLE. V(TYPE): VULNERABILITY EXISTS. P(MECHANISM): SECURITY MECHANISM USED)

joined groups, joining to new groups and publishing resources on the legitimate peer behalf.

D. Evaluation summary

Since JXME-Proxied is Open Source Software (OSS), supported by a community of developers, and it is also very simple and small built using four classes) it can be considered safe from *Software security flaws*. Furthermore, since Proxied Peers do not store data locally, they are not vulnerable at execution time to *Local data alteration*. The analysis of possible attacks and the existing security mechanisms of JXME-Proxied, classified by peer operations, provides a vulnerability map summarized in Table III. Attacks are those described in Section III-B.

The main vulnerabilities found can be classified as:

- **V(1)**: Malicious executable code can easily be built and cannot be automatically discovered when installed.
- **V(2)**: No encryption mechanism exists.
- **V(3)**: No data flow masquerading mechanism exists. It is easy to identify important peers by its traffic.
- **V(4)**: No real authentication is enforced.

The available security mechanisms are:

- **P(pass)**: Passwd Membership Service.
- **P(OSS)**: Open Source Project.
- **P(TGMS)**: Trusted Group Membership Service [13].

IV. BRIEF COMPARISON BETWEEN JXME-PROXIED AND JXME-PROXYLESS

Even though we have focused on the JXME-Proxied version, in this section we highlight the main differences with the JXME-Proxyless security model. Such differences are mainly based on the fact that JXME-Proxyless is designed for more powerful mobile devices, with CDC in mind. As a result, this version follows the same peer architecture as JXTA, allowing mobile devices to directly participate into the JXTA network by themselves, without the need of an external super-peer. Any peer using JXME-Proxyless is named a *Proxyless Peer* and is able to discover other devices and services, publish Advertisements about its own resources, establish direct connections to any other peer, create/join private virtual domains (Peer Groups) and directly exchange/access content offered by other Peer Group members.

Nevertheless, Proxyless Peers have some limitations on regards to the JXTA base architecture. First of all, even though most JXTA services are implemented, some of they are not, and even when a particular service exists, it must be taken into account that it may not have full capabilities. An additional constraint is the fact that a Proxyless Peer cannot act as a super-peer in a JXTA network. As a result, since super-peers help network management, a JXTA network formed only by Proxyless Peers may have scalability issues.

An important difference between both JXME versions is that a Proxyless Peer does not need a Relay Peer to participate in the JXTA network. Therefore, during the platform startup operation, a Proxyless Peer does not perform any network activity, only loads the JXTA libraries and creates the default network manager, thus being protected from external interference at this level.

In JXME-Proxyless, unlike Proxied, the Membership Service is defined as generic, leaving up to developers to implement their own version, with the security level required by their applications. However, JXME-Proxyless provides no implementation at all, allowing any Proxyless Peer to create and join any Peer Groups. As a result, no security really exists for the join operation, and no authentication process is enforced, allowing any peer to claim any identity within the system.

As far as Advertisement publication is concerned, in contrast of JXME-Proxied, where they are encoded in plain text, JXME-Proxyless has a light XML parser and can directly manage XML-encoded Advertisements, just like in the desktop version. But in terms of security, no security layer over Advertisements is provided either. Therefore, they share the same vulnerabilities.

Another important difference in JXME-Proxyless is that can perform direct input and output TCP connections, using pipes, in contrast with JXME-Proxied where only output HTTP connections are allowed. It allows Proxyless Peers to participate directly in the JXTA network, but paying the cost of higher resource consumption, such as battery power, because they have to forward messages, find routes, save Advertisements, etc. In terms of message exchange, while Proxied Peers send all data in simple text, Proxyless Peers can send it using a secure way. JXME-Proxyless developers are currently working in the implementation of a wire transport security layer that may be applied to pipes. This security layer is based in JXTA's own definition of Transport Layer Security (TLS) [19]. This implementation basically allows private, mutually

authenticated reliable communications, protected against both passive and active attacks.

Finally, the Disconnection operation is different in JXME-Proxyless. It does not require any communication using the network and no security vulnerabilities exist, since pipes cannot be hijacked.

V. Conclusions and further work

Since JXTA and JXME are OSS projects, their expansion mainly depends in the developer community interests. JXME has a relatively short live and until now the main efforts have been focused on achieving functionality features. Unfortunately, the work done in JXME security is very poor, justified by the developers in the assumed reduced capabilities of mobile devices such as limited battery, and storage.

Taking into account the security analysis of a Proxied Peer operation cycle, we can conclude that this JXME version does not have an appropriate security baseline, because messages are exchanged with the Relay Peer in clear text, and no powerful authentication method is provided. This allows any Proxied Peer to impersonate other Proxied Peer which is using the same Relay Peer, using his open pipes to send messages, open or close pipes and basically, perform any defined operations. The JXME-Proxyless version does not look much more advanced on that regard, but at least a confidential and authenticated communication channel is provided, using TLS.

Once the JXME-Proxied security level has been analyzed, further research includes developing new security mechanisms to achieve a minimum security baseline. Security has to be provided using light cryptographic methods, taking into account the limitations of mobile devices. The vulnerabilities that can be addressed are JXME library and peers authenticity, and data flow encryption and mask.

Moreover, further research can be done in the security evaluation of the newest JXME version, JXME-Proxyless.

References

[1] A. Oram, *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*, O'Reilly & Associates, Inc., Sebastopol, CA, USA, 2001.

[2] Skype, "Skype on your mobile", 2004, http://www.skype.com/mobile.

[3] Sun Microsystems, "Project JXTA", 2001, http://www.jxta.org.

[4] Sun Microsystems, "Project JXSE", 2001, https://jxta-jxse.dev.java.net.

[5] Sun Microsystems, "Project JXTA-C", 2001, https://jxta-c.dev.java.net.

[6] Sun Microsystems, "Project JXME", 2003, https://jxta-jxme.dev.java.net.

[7] T. Piedrahita and E. Montoya, "Performance analysis of JXTA/JXME applications in hybrid fixed/mobile environments", *Revista Colombiana De Computación*, vol. 7, no. 1, 2006.

[8] J. Arnedo-Moreno and J. Herrera-Joancomartí, "A survey on security in JXTA applications", *Journal of Systems and Software*, vol. 82, no. 9, pp. 1513 – 1525, 2009.

[9] Sun Microsystems Inc., "JXTA v2.0 protocols specification", 2007, https://jxta-spec.dev.java.net/nonav/JXTAProtocols.html.

[10] "JXTA java standard edition v2.5: Programmers guide", 2007, https://jxta-guide.dev.java.net/.

[11] Sun Microsystems Inc., "J2ME building blocks for mobile devices. white paper on KVM and the connected, limited device configuration (CLDC)", 2000, http://java.sun.com/products/cldc/wp/.

[12] B. Traversat, A. Arora, M. Abdelaziz, M. Duigou, C. Haywood, J.C. Hugly, E. Pouyoul, and B.Yeager, "Project jxta 2.0 super-peer virtual network", Tech. Rep., SunMicrosystems,Inc, May 2003.

[13] L. Kawulok, K. Zielinski, and M. Jaeschke, "Trusted group membership service for JXME (JXTA4J2ME)", in *Wireless And Mobile Computing, Networking And Communications, (WiMob'2005), IEEE International Conference on*, Aug. 2005, vol. 4, pp. 116–121.

[14] L. Yunpeng, "Research on the mobile P2P VOD system of JXME", in *CEA'07: Proceedings of the 2007 annual Conference on International Conference on Computer Engineering and Applications*, Stevens Point, Wisconsin, USA, 2007, pp. 80–83, World Scientific and Engineering Academy and Society (WSEAS).

[15] T. Espiritu and W. Yu, "Universal chat client for J2ME enabled mobile devices using the JXME implementation of JXTA", 2006, PhD Thesis in the Ateneo de Manila University.

[16] J. Biström and V. Partanen, "Mobile P2P - creating a mobile file-sharing environment", *Tik-111.590 Research Seminar on DIgital Media*, 2004.

[17] J. Nutzel and M. Kubek, "A mobile Peer-to-Peer application for distributed recommendation and re-sale of music", in *AXMEDIS '06: Proceedings of the Second International Conference on Automated Production of Cross Media Content for Multi-Channel Distribution*, Washington, DC, USA, 2006, pp. 93–98, IEEE Computer Society.

[18] D. Brookshier, D. Govoni, N. Krishnan, and J.C. Soto, *JXTA: Java P2P Programming - Chapter 8: JXTA and Security*, 2002, http://java.sun.com/developer/Books/networking/jxta.

[19] The Internet Society, "The Transport Layer Security (TLS) Protocol Version 1.1", 2006, http://www.ietf.org/rfc/rfc4346.txt.