Journal of Mobile Multimedia, Vol. 6, No. 3 (2010) 227–242

APPLICATION OF JXTA-OVERLAY PLATFORM FOR SECURE ROBOT CONTROL

EVJOLA SPAHO
Graduate School of Engineering, Fukuoka Institute of Technology (FIT)
3-30-1 Wajiro-Higashi, Higashi-Ku, Fukuoka 811-0295, Japan
evjolaspaho@hotmail.com

KEITA MATSUO
Fukuoka Prefectural Kaho-Sogo High School
1117-1 Haji, Keisen-Machi, Kaho-Gun, Fukuoka 820-0607, Japan
matuo-k7@fku.ed.jp

LEONARD BAROLLI
Department of Communication and Information Engineering
Fukuoka Institute of Technology (FIT)
3-30-1 Wajiro-Higashi, Higashi-Ku, Fukuoka 811-0295, Japan
barolli@fit.ac.jp

FATOS XHAFA
Department of Languages and Informatics Systems
Technical University of Catalonia, C/Jordi Girona 1-3, 08034 Barcelona, Spain
fatos@lsi.upc.edu

JOAN ARNEDO-MORENO
Department of Computer Science, Multimedia and Telecommunication
Open University of Catalonia, Rambla del Poblenou, 156 08018 Barcelona, Spain
jarnedo@uoc.edu

VLADI KOLICI
Department of Electronics and Telecommunication
Polytechnic University of Tirana, Mother Teresa Square, No.4, Tirana, Albania
ladi@istitech.net

Received December 16, 2009
Revised May 19, 2010

In this paper, we present the evaluation and experimental results of secured robot control in a P2P system. The control system is based on JXTA-Overlay platform. We used secure primitives and functions of JXTA-Overlay for the secure control of the robot motors. We investigated the time of robot control for some scenarios with different number of peers connected in JXTA-Overlay network. All experiments are realised in a LAN environment. The experimental results show that with the join of other peers in the network, the average time of robot control is increased, but the difference between the secure and unsecure robot control average time is nearly the same.

Keywords: P2P, JXTA-Overlay, robot control, security.
Communicated by: D. Taniar

1 Introduction

Presently most of the P2P research field has pushed through problems related to the security. P2P systems benefit from high scalability and fault tolerance. Unfortunately in the context of security this is also one of the main disadvantage. Usually there are no central authorities for the verification and enforcement of policies. Security is a fundamental quality criteria in P2P Systems. Fulfilling certain security goals is a vital precondition for the preparation of P2P Technology for larger business applications.

Today research is focused on the design, implementation and deployment of full featured P2P networks that integrate end devices. P2P platforms are a good approach for e-learning and robot control [1].

Robots are rapidly expanding into human environments by interacting, exploring and working with the humans. The new generation of robots will increasingly touch people and their lives [2]. They can be used in situations where it is too dangerous, expensive, tedios and complex for humans to operate. The successful introduction of robots in human environments will relay on competent and practical systems that are safe and easy to use.

In this paper, we present a secure robot control application based on JXTA-Overlay P2P platform. This platform is implemented in JAVA language. Thus, it does not depend on the Operating System (OS). We used secure primitives and functions of JXTA-Overlay for the secure control of the robot motors. We investigated the time of robot control for some scenarios with different number of peers connected in JXTA-Overlay network. All experiments are done in a LAN environment. We evaluated the proposed system by many experiments and have shown that in the proposed system the security has a cost at the application efficiency by adding some overhead in the time of robot control, but JXTA-Overlay can be used successfully to control robots in real time.

The structure of this paper is as follows. In Section 2, we introduce the related work. In section 3, we introduce JXTA Technology. Section 4 provides a description of the JXTA-Overlay. In Section 5, we present JXTA-Overlay security. In section 6, we present application of secure JXTA-Overlay for robot control. In Section 7, we discuss the experimental results. Finally, conclusions and future work are given in Section 8.

2 Related Work

In this section, we discuss the related work for robot control and security in P2P systems.

Developing a multi-robot control architecture has been the subject of several studies. Asama et al. proposed ACTor-based Robot and Equipments Synthetic System (ACTRESS) [3]. The ACTRESS architecture tackles the issues of communication protocols with different abstraction levels, path planning, and task assignment through multi-stage communication protocols. The architecture was tested on heterogeneous mobile robots performing complex object pushing tasks that cannot be accomplished with a single robot.

GOFER is another distributed multi-robot problem solving architecture [4]. It is based on a centralized task planning and scheduling module, which keeps track of the task allocation and the availability of all the robots through direct communication with them. The system also uses conventional artificial intelligence and task allocation algorithms, such as Contract Net Protocol to dispatch the subtasks to be performed by individual robots. Despite its satisfactory performance in applications, like following a wall and pushing objects, most of

the GOFER implementations involved no more than three robots. In addition, GOFER is designed for indoor environments only and its dependence on a centralized task planning and scheduling module represents a major limitation from a flexibility and fault-tolerance perspective.

The ALLIANCE architecture [5] emphasizes some key features in the design of multi-robot coordination architectures with a particular focus on fault-tolerance, reliability, and adaptability. The architecture is designed for small to medium-sized teams of loosely coupled heterogeneous mobile robots operating in dynamic environments to perform tasks with possible ordering dependencies. The ALLIANCE architecture also assumes that the robots are probabilistically capable of assessing the performance of their actions as well as those of others through perception and broadcast communication.

There are some other research works that deal with the autonomous distributed robot systems. In these systems, many autonomous mobile robots cooperate together to carry out complicated tasks [6, 7, 8, 9, 10, 11]. By using many robots in a distributed way, the robot functions can be simplified and the system cost, fault tolerance and flexibility can be improved. In the case where there are many robots and if one robot has a problem, the other robots can cooperate to finish a requested task. Therefore, the distributed robot systems are better than single robots. However, in distributed robot systems, to realise a common task, the robots should cooperate together and consider not only their own task but also the complete task.

Security is one of the key issues when evaluating a P2P system and a security baseline must be kept in any P2P system in order to ensure some degree of correctness even when some system components will not act properly. A survey regarding security in P2P systems can be found in [12].

For message security, the JXTA reference implementation [13] provides two mechanisms: TLS (Transport Layer Security) and CBJX (Crypto-Based JXTA Transfer) [14]. The former provides private, mutually authenticated, reliable streaming communications, whereas the latter provides lightweight secure message source verification (but not privacy).

JXTA provides its own definition of standard TLS as a transport protocol. The JXTA definition of TLS is composed of two subprotocols: the TLS Record Protocol and the TLS Handshake Protocol. The TLS Record Protocol provides connection security using symmetric cryptography for data encryption. The keys for this symmetric encryption are generated uniquely for each connection and are based on a secret negotiated by the TLS Handshake Protocol. In addition, the connection is reliable by including message integrity check using a keyed MAC.

On the other hand, CBJX is a JXTA-specific security layer which pre-processes messages to provide an additional secure encapsulation, creating a new message that is then relayed to an underlying transport protocol. The original message is signed and an additional information block is also added to the secured message. This information block contains the source peer credential, both the source and destination addresses, and the source peer ID.

In order to use both mechanisms, TLS and CBJX, a specific group membership service is required: the Personal Security Environment (PSE). The membership service is one of the JXTA core services, taking care of group membership and identity management by providing each group member with a credential. Peers may include credentials in messages exchanged within a group in order to prove membership and provide a means for implementing access

control in offered services. However, PSE solely supports on X509 certificates as credentials and Java keystores as a cryptographic module [15].

3 JXTA Technology

JXTA technology is an open and innovative collaboration platform. It consist of six protocols that allow different types of peers to communicate and collaborate among them. The main purpose of JXTA is to built P2P systems that offers the basic functions for P2P communication [16]. JXTA realizes this issue working and resolving three main problems of the existing P2P networks:

1. OS independence,
2. Language independence,
3. Providing services and infrastructure for P2P applications.

3.1 JXTA Architecture

JXTA has a structure with three layers, as shown in Fig. 1. The system is designed in a modular way to let the developers to choose the set of services that satisfy their needs [17].

1. **The Core Layer.** This layer implements the essential set of primitives that are common to P2P networking. These primitives includes creation of peers, discovery, transport, peer groups and communication even behind firewalls or NAT. The JXTA core inculdes also basic security services.
2. **The Services Layer.** This layer implements some services that are integrated to JXTA. These services include searching and indexing, file sharing, protocol translation, authentication and Public Key Infrastructure (PKI) services, as well resource search.
3. **Applications Layer.** This layer implements applications that are integrated to JXTA, such as P2P instant messaging, file sharing and content management.

The distinction between services and final applications may not always be clear, since what a client may consider an application may be considered a service by another peer. For that reason, the system is designed in a modular way, letting developers choose the set of services and applications which most satisfy their needs. All JXTA components are within these three layers [18].

3.2 JXTA Protocols

JXTA comprises a set of six open protocols that allow devices with different software to collaborate. A peer can implement one of these protocols and they can ask the other peers for supplement functionalities. Below is a brief description of these protocols.

Peer Resolver Protocol (PRP) - Allows a peer to send a search query to another peer. This protocol is a basic communications protocol that follows a request/response format. The resolver is used to support communications in the JXTA protocols like the router and the discovery protocols. The resolver also allows for the propagation of queries. For example, if a peer receives a query and does not know the answer, the resolver sends the query to other

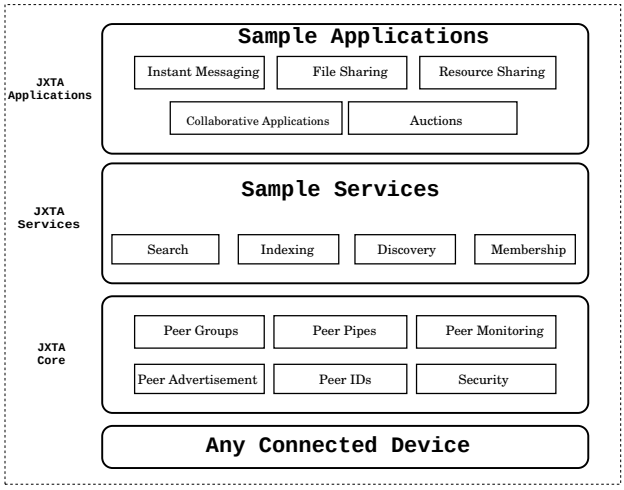

Fig. 1. JXTA architecture.

peers. This is an interesting feature, especially because the originating peer does not need to have any knowledge of a peer that may actually have the result to the query.

Peer Information Protocol (PIP) - Allows a peer to learn about the status of another peer. The information protocol is used partially like ping and partially to obtain basic information about a peer's status.

Pipe Binding Protocol (PBP) - It is used to create a communications path between one or more peers. A pipe is a virtual channel between two peers. The protocol is primarily concerned with connecting peers via the route(s) supplied by the peer endpoint protocol.

Rendezvous Protocol (RVP) - The RVP is responsible for the propagation of the messages in JXTA groups. The RVP defines a base protocol for peers to send and receive messages inside the group of peers. This protocol controls how the messages are propagated.

Endpoint Routing Protocol (ERP) - Is used to find available routes to route messages to destination peers. The protocol uses gateways between peers to create a path that consists of one or more of the pipe protocols suitable for creating a pipe. The PBP uses the list of peers to create routes between peers.

Peer Discovery Protocol (PDP) - Allows a peer to discover other peer advertisements (peer, group, service, or pipe). This protocol uses a searching mechanism to locate information. The protocol can find peers, peer groups, and all other published advertisements. The advertisements are mapped to peers, groups, pipes or other resources.

4 JXTA-Overlay

JXTA-Overlay is a middleware built on top of the JXTA specification, which defines a set of protocols that standardize how different devices may communicate and collaborate among them. It abstracts a new layer on the top of JXTA through a set of primitive operations and services that are commonly used in JXTA-based applications and provides a set of primitives that can be used by other applications, which will be built on top of the overlay, with complete independence. JXTA-Overlay extends JXTA protocols with the goal of overcoming some of

its limitations: the need for the developer to manage the presence mechanism, peer group publication and message exchange. To achieve this, JXTA-Overlay provides a set of basic functionalities, primitives, intended to be as complete as possible to satisfy the needs of most JXTA-based applications.

4.1 JXTA-Overlay network

In a JXTA-Overlay network, the main interacting entities are as follows.

- *End-users*, they connect to the JXTA-Overlay network by authenticating using a username and password. Once the authentication process is successfully completed, they are organized into different overlapping groups, so only members of the same group may interact. It is also important to take into account that JXTA-Overlay end-users are mobile, they may connect at different times using different client peers, as well as may be connected through several client peers at the same time.

- A *client peer* represents an application, which end-users use to communicate and share resources between themselves, effectively acting as end-user proxies within the JXTA-Overlay network. They forward end-user data to client peers that belong to end-users of the same group and authentication data to a broker. A client peer is assumed to belong to the same groups as its current end-user.

- *Brokers* control access to the network, requesting end-user authentication, and help client peers interact between themselves by propagating their related information. Brokers are very important since they exchange information about all client peers, maintaining a global index of available resources, thus allowing all peers to find network services. Brokers also act as beacons which client peers which have recently gone online use to join the network. For that reason, they usually have well-known identifiers, such as a DNS name or a static IP address.

All the information related to user configuration (username, password and group membership) is stored in a special single entity within the JXTA-Overlay network: a central database. Only brokers may access the database data, in order to check end-user authentication attempts and organize them into groups.

4.2 Architecture of JXTA-Overlay

The architecture of the JXTA-Overlay middleware defines three modules, which let the different entities communicate: the Client Module, the Broker Module and the Control Module. Altogether, they form an abstraction layer on top of JXTA. JXTA-Overlay architecture is shown in Fig. 2.

- **The Client Module** defines all necessary primitives for peer clients to join a JXTA-Overlay network and interact with other peers and the broker. Applications developed on top of JXTA-Overlay are always based on the invocation of Client Module primitives defined and the processing of events thrown upon primitive execution or a broker response. Primitives are comprised for: peer discovery, peer resources discovery, resource allocation, task submission and execution, file/data sharing discovery and transmission,

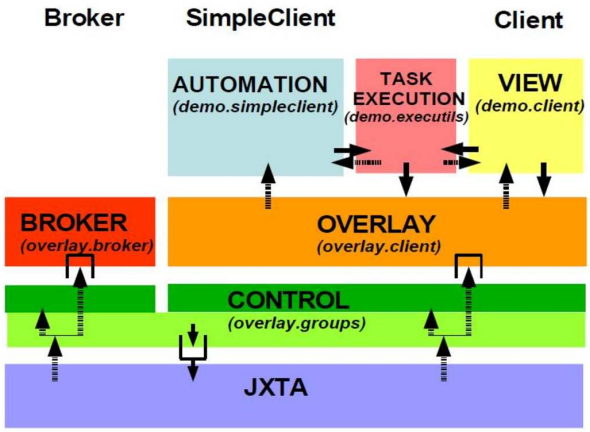

Fig. 2. JXTA-Overlay architecture.

instant communication, peer group functionalities and, monitoring of peers, groups, and tasks.

- **The Broker Module** defines all the functions that client peers may call upon a broker in order to be granted access to the network, create and publish groups or retrieve other client-peers information. Functions produce a reply from the broker to the calling client peer. Broker functions are always called as a result of Client Module primitives.

- **The Control Module** acts as an intermediate layer between the Broker and Client Modules, providing the generic functionalities for group management and messaging. The Control Module provides messaging between JXTA-Overlay entities using JXTA pipes, a virtual communication channel between peers. Client peers have an input pipe for each group it belongs to, so other group members may send messages using the input pipe associated to that group. Brokers have an input pipe which is shared for all incoming messaging.

5 JXTA-Overlay Security

Some of the security concerns in JXTA-Overlay are shown in the following.

- **No Privacy:** Transmitted data may be easily eavesdropped, since no data privacy is provided. Even though it may be argued that data privacy in message exchanges between end-users is just an optional feature, there are some cases where privacy should not be optional, namely the initial authentication username and password. Currently, both fields are sent as plain text. Therefore, any device at broadcast range may read this information with a network protocol analyzer (such as Wireshark).

- **Advertisement Spoofing:** Any legitimate user may forge advertisements with no fear of reprisal. No integrity or source authenticity is maintained. False fields, such as the source client peer identifier or any statistics information, may be added into the

advertisement, which will be automatically distributed by the broker and accepted by all group members, unaware of the false data it contains [19].

- **Broker Impersonation:** Client peers connect to a self-proclaimed broker, but never check if it is a legitimate one. Even in the case that client peers are connecting to the proper broker address, there are no guarantees that the broker is a legitimate one, since it may be that traffic is being redirected to a fake one via methods such as DNS spoofing.

To solve these problems is used a secure architecture.

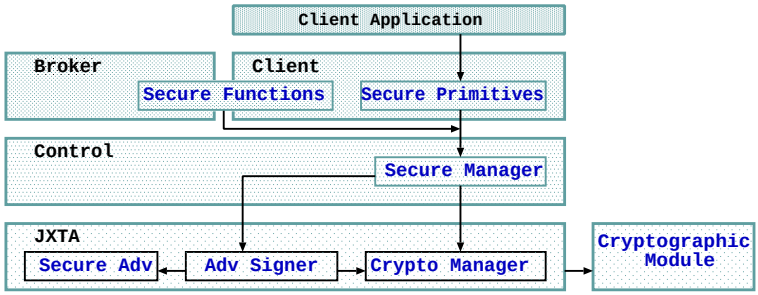

Fig. 3. Security architecture for JXTA-Overlay.

5.1 Secure Architecture Overview

The main concepts of security for JXTA-Overlay are presented in Fig. 3. The following modules are specified as an extension of JXTA core protocols and are not JXTA-Overlay specific.

- *Crypto Manager* provides an abstraction layer for the cryptographic module and key management method. Crypto Manager is completely modular and accepts different implementations, according to the needs of the specific cryptographic module being used for any particular deployment of JXTA-based application.
- *Secure Adv* defines the secure advertisement format and provides a method for key transport and additional fields related to its signature.
- *Adv Signer* manages JXTA advertisement signature and validation by interacting with Crypto Manager.

The JXTA-Overlay specific modules are:

- *Secure Manager* is the common to all interface additional security capabilities within the JXTA-Overlay Control Module, operating as a single entry point for all secure services.
- *Secure Functions and Primitives* modules just extend the base Broker and Client Modules, providing a set of additional primitives and functions which take into account security considerations.

5.2 Secure Messenger Primitives

JXTA-Overlay has 122 primitives and 84 events related to different functions. We will focus on *messenger primitives* which send simple messages between end-users. Messenger primitives define how to directly exchange simple text messages between end-users. For instance, in Robot Control, commands are sent using these primitives. Once the cryptographic set-up has established key and credential information for all participating entities, it is finally possible to secure data exchanges by providing a secure version of each primitive. The two main messenger primitives are:

1. *sendMsgPeer*: Sends a simple message to some other client peers.
2. *sendMsgPeerGroup*: Sends a simple message to all members of a group. It is actually resolved by iteratively calling *sendMsgPeer*.

The secure versions of both primitives provide lightweight privacy, data integrity and message source authentication in a stateless, best effort method. This is in contrast, for example, with JXTA's secure pipes, which rely on TLS and require some previous negotiation between endpoints each time a message exchange is initiated [20]. The necessary steps for some end-user connected to client peer A to send a simple text message to another one connected to B are:

1. A retrieves B's Pipe Advertisement. This step also happens in the original insecure primitive and thus means no additional burden.
2. A validates the advertisement signature in order to ensure that it has not been compromised. If the signature does not validate, the advertisement has been tampered, and is deemed invalid. If the message is sent, no guarantees can be made on regards to its security.
3. A retrieves PK_B from the signed advertisement's enclosed credential, $C^B_{red_{Br}}$.
4. $A \longrightarrow B : \{E_{PK_B}(m, S_{Sk_A}(m))\}$.
5. B decrypts the message using SK_B.
6. B retrieves A Pipe Advertisement and repeats steps 2, 3.
7. B validates the message signature using PK_A obtained via B's signed advertisement.

The *secureMsgPeerGroup* primitive just iteratively uses the *secureMsgPeer* to send the same message to a group of peers. As a standard message exchange between end-user endpoints, these primitives are mainly subject to data eavesdropping and end-user impersonation. The former is avoided by encrypting data using a wrapped key approach, whereas the latter, as well as data integrity, is provided by digitally signing the data. Peer encryption communication system is shown in Fig. 4. In both cases, each other endpoint's public key is necessary in steps 1-2 (encryption) and 6 (signature validation). However, both public keys are contained in the credential provided by the Broker at the secureLogin primitive call, and thus, can be considered that belongs to a legitimate owner, and not an attacker.

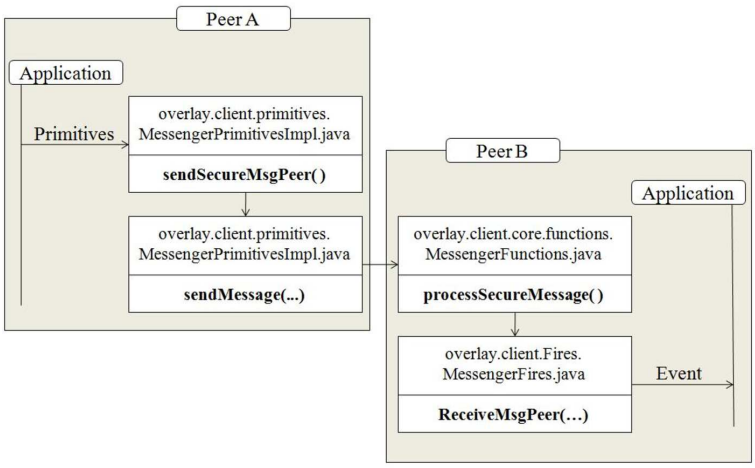

Fig. 4. Peer encryption communication system.

6 Application of JXTA-Overlay for secure Robot Control

The successful introduction of robots in human environments will rely on the development of competent and practical systems that are dependable, safe, and easy to use. The ability to interact with people in the human environment has been a recent motivator of the humanoid robotics community and the service robotics community.

In our application, we deployed a network with three nodes as shown in Fig. 5. The robot is connected with the broker via RS232C connector, which is a legal interface and many devices have implemented it. We considered the robot as end-device. Peer node specifications are shown in Table 1.

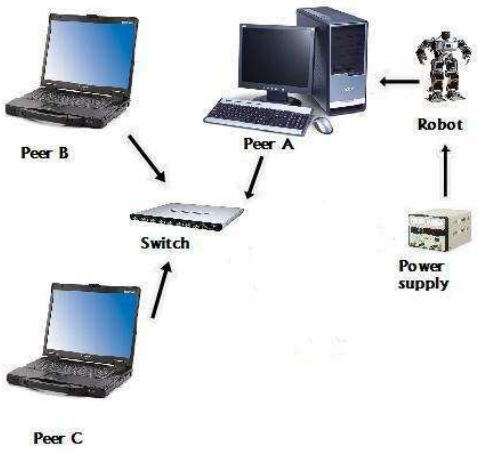

Fig. 5. P2P Robot control system.

For the experiments is used KHR-1 robot with the following specifications: Heigh 340

Table 1. Peer node specifications.

Peer node	Characteristics
Broker	AMD Athlon (tm) 3200+2.01 GHz, 100 GB RAM
Peer 1	Pentium M Centrino 1.2 GHz, 512 MB RAM
Peer 2	Intel Centrino Duo 1.3 GHz, 2GB RAM
Connection	100 Mbps

mm, Width 180 mm, Depth 80 mm and Weight 1.2 kg. The robot has 17 servo-motors as shown in Fig. 6. To create the motions we used Heart-to-Heart (motion editor for RCB-1). A snapshot of this program is shown in Fig. 7.

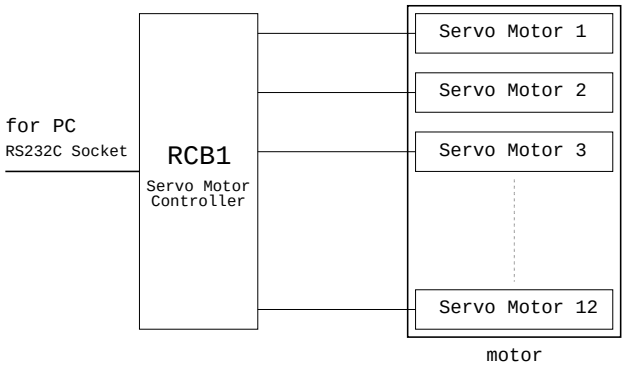

Fig. 6. KHR-1 Robot interface.

During the experiments are observed only the messenger primitives because the login and connect primitives just happens once for the full session. The command of the robot control is sent as simple message using chat service. We experimentally studied the time for Robot Control for two different cases:

- Using *sendMsgPeer* primitive,
- Using *secureMsgPeer* primitive.

These messenger primitives define how to directly exchange simple text messages between end-users, such as a chat service.

After the synchronisation of the time between peer nodes, the time of robot control is measured using both primitives. Three scenarios were realised by joining peers to this network and the times of robot control were measured.

7 Experimental Results

In Fig. 8 is shown the time for Robot Control measured for 50 experiments using sendMsgPeer primitive (non secure primitive). While, in Fig. 9 is shown the time for Robot Control when is used secureMsgPeer primitive (secure primitive). In Table 2 are shown the experimental data for above experiments. For only one peer, the major part of measured time for both cases (using unsecure and secure primitives) is less than 1 s.

In Fig. 10 is shown the average time for robot control using and not using security. As can be observed, the difference between the average time of unsecure robot control and secure

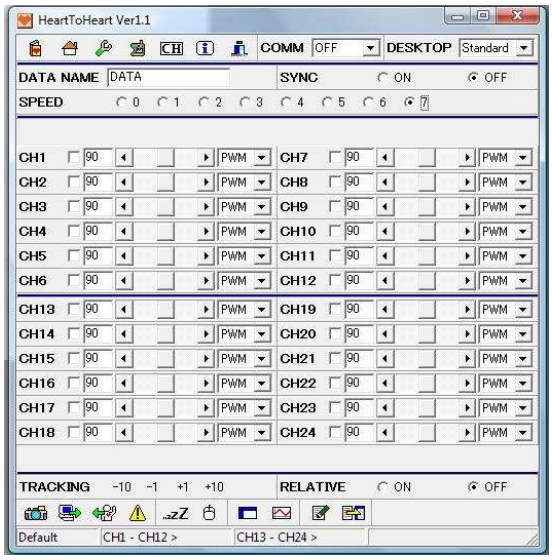

Fig. 7. Snapshot of Heart-to-Heart program for robot control.

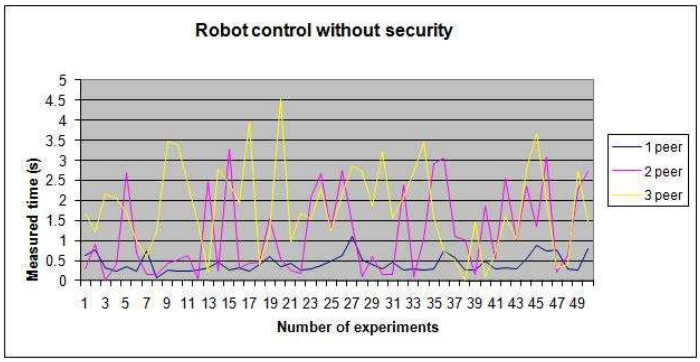

Fig. 8. Experimental results for robot control using sendMsgPeer primitive.

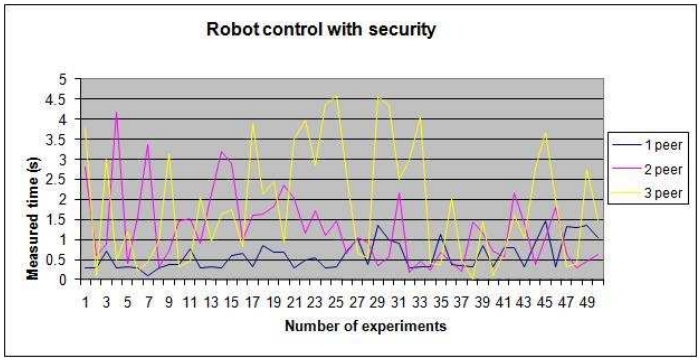

Fig. 9. Experimental results for robot control using secureMsgPeer primitive.

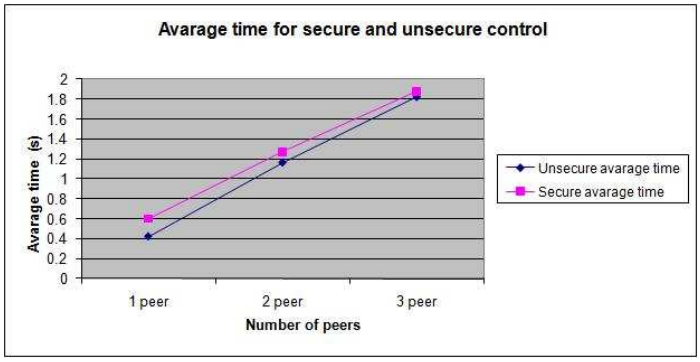

Fig. 10. Average time for secure and unsecure robot control.

Table 2. Experimental results.

No of peers	Primitives	Under 1s	1-3 s	Over 3 s
1 Peer	sendMsgPeer	98%	2%	0%
	secureMsgPeer	82%	18%	0%
2 Peers	sendMsgPeer	56%	38%	6%
	secureMsgPeer	48%	46 %	6 %
3 Peers	sendMsgPeer	62%	22%	16%
	secureMsgPeer	40%	36%	24%

robot control is small and with increase of number of peers the difference decreases.

From the experimental data is observed that some of the values of the time measured are over 3 seconds. The reasons are explained in the following.

- This situation can happen because the broker is very busy. The mission of the broker node is to manage and control all the requests received from the client peers. Broker manages the events produced by such requests and propagates them to superior levels of overlay. Broker performs a big number of tasks because Broker functions are always called as a result of client primitives. Some of the broker functionalities are: event management, controlling the resources connected to the broker, maintaining the organisation of the resources in groups, finding the best resource for file sharing, finding the best resource for executing submitted tasks, maintaining updated statistic information, etc.

- Another reason is that JXTA-Overaly platform used in experiments has no parameter tuning [21]. P2P applications involve many parameters, whose values cannot be arbitrarily fixed and they require a careful tuning to achieve expected performance. These parameters are divided in: parameters related to the Broker peers and parameters related to the Client peers. Broker peer parameters include discovery time, publishing time, advertisement time for broker's statistic. High values of these parameters could lead to an inconsistent state of the network. The small values of the parameters could cause congestion of Brokers peers due to very frequent updates. In the case of the Client peer the parameters include time parameters related to the peer's advertisement, pub-

lishing and expiration. Small values of these parameters could cause saturation of client peers due to very frequent generation of advertisement and high traffic in the network due to the publishing and propagation of the peer's information. These parameters directly relate to the time intervals for checking the state of the network, performing action and sending state information to the network.

- Network latency that is the delay introduced by the network. Network situation such as congestion can be a major contributing factor. Packets are forwarded from source to destination and traverse network link including LAN switches. The combination of various latency sources such as buffering, propagating, switching, queuing, and processing delays produces a complex and variable network latency profile.

The average time for robot control when in the network is only one peer is 0.42 s for unsecure and 0.6 for secure primitive and for two peers 1.17 s and 1.27 s, respectively. When in the network are three peers the time of robot control using unsecure primitive is 1.82 s and using secure primitive is 1.88 s. When another peer joins the network, the time for robot control for both primitives increases, because the join of other peers in the network increase the number of requests that the client primitives send to the broker. However, the time difference between secure and unsecure control is not significant.

From this results, we can conclude that the primitives of JXTA-Overlay *sendMsgPeer* and *secureMsgPeer* can be successfully used to control the robot in a smoothly way. We applied secured JXTA-Overlay architecture for the control of KHR-1 robot. In Fig. 11 from the left to the right, we show the steps of robot during walking. In Fig. 12 from the left to the right, we show the scenario when the robot moves the left hand.

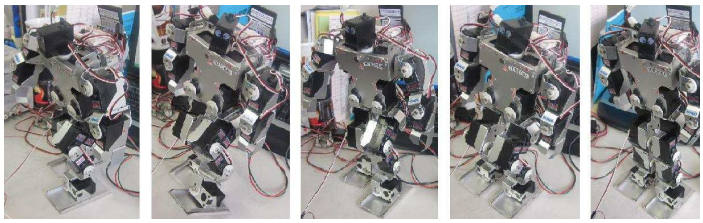

Fig. 11. Robot walking.

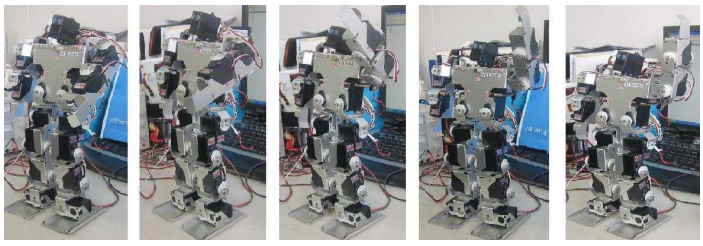

Fig. 12. Robot moving the left hand.

8 Conclusions and Future Work

In this paper is presented the application of JXTA-Overlay P2P system for robot control. Specifically, we have experimentally measured the time for robot control, using two primitives: *sendMsgPeer* and *secureMsgPeer*. The objective was to see the performance as well as limitations of the P2P robot control when it is used as an end-device and the cost of using security in robot control.

The experimental results have shown that the join of other peers in the network increase the number of requests that the client primitives send to the broker. In this situation, the broker should manage all requests of the client primitives, and this cause an overhead in the time of robot control. But, with the increasing of the number of peers in the network, the difference between the secure and unsecure average time for robot control is nearly the same. From the experimental results, we conclude that the primitives of JXTA-Overlay *sendMsgPeer* and *secureMsgPeer* can be successfully used to control the robot in a smoothly way.

In the future, we want to implement new secure functions and primitives for JXTA-Overlay, which can offer several improvements of the existing JXTA-Overlay protocols and services and increase the reliability of JXTA-based distribution applications and support group management of file sharing.

Acknowledgement

The authors would like to thank Japanese Society for the Promotion of Science (JSPS) for supporting this work.

References

1. F. Xhafa, R. Fernandez, T. Daradoumis, L. Barolli, S. Caballe (2007), *Improvement of JXTA Protocols for Supporting Reliable Distributed Applications in P2P Systems*, Proc. of NBiS-2007 (Regensburg, Germany), LNCS, Vol. 4658, pp. 345-354.
2. P. Caloud, W. Choi, J. C. Latombe, C. L. Pape, M. Yim (1990), *Indoor Automation with Many Mobile Robots*, Proc. of the IEEE International Workshop on Intelligent Robots and Systems, pp. 67-72.
3. H. Asama, K. Ozaki, A. Matsumoto, Y. Ishida, I. Endo (1992), *Development of Task Assignment System Using Communication for Multiple Autonomous Robots*, Journal of Robotics and Mechatronics, pp. 122-127.
4. C. Le Pape (1990), *A Combination of Centralized and Distributed Methods for Multi-agent Planning and Scheduling*, Proc. of the IEEE International Conference on Robotics and Automation, pp. 488-493.
5. L. E. Parker (1998), *Alliance: An Architecture for Fault Tolerant Multirobot Cooperation*, IEEE Transactions on Robotics and Automation, pp. 220-240.
6. L. Chaimowicz, T. Sugar, V. Kumar, M. F. M. Campos (2001), *An Architecture for Tightly Coupled Multi-robot Cooperation*, Proc. of IEEE International Conference on Robotics and Automation, pp. 2992-2997.
7. L. Inoue, T. Nakajima (2001), *Cooperative Object Transportation by Multiple Robots with Their Own Objective Tasks*, Journal of the Robotics Society of Japan, pp. 888-896.
8. K. Ozaki, H. Asama, Y. Ishida, A. Matsumoto, I. Endo (1996), *Collision Avoidance Using Communication Between Autonomous Mobile Robot*, Journal of the Robotics Society of Japan, pp. 961-967.
9. M. Parnichkun, S. Ozono (1998), *CDCSMA-CD Communication Method for Cooperative Robot Systems*, Advanced Robotics, pp. 669-694.

10. P. E. Rybski, S. A. Stoeter, M. Gini, D. F. Hougen, N. P. Papanikolopoulos (2002), *Performance of a Distributed Robotic System Using Shared Communications Channels*, IEEE Transactions on Robotics and Automation, pp. 713-727.
11. J. Arai, A. Koyama, and L. Barolli (2008), *AR-TDMA: An Adaptive Reservation Time Division Multiple Access Control Protocol for Robot Inter-communication*, International Journal of Wireless and Mobile Computing (IJWMC), Vol. 3, No. 1/2, pp. 4-11.
12. S. Wallach (2003), *A Survey of Peer-to-Peer Security Issues*, Theories and Systems, Springer Berlin Heidelberg, pp. 253-258.
13. JXTA 2.5 RC1 (June 2007), http://download.java.net/jxta/build.
14. D. Bailly (2002), *CBJX: Crypto-based JXTA (An Internship Report)*.
15. CCITT (1988), *The Directory Authentication Framework: Recommendation*.
16. D. Brookshier and D. Govoni and N. Krishnan and J.C Soto (2002), *JXTA: Java P2P Programming*, Sams Publishing.
17. SUN Microsystem (2003), *Project JXTA v 2.0: Java Programmer's Guide*.
18. SUN Microsystem (2001), *Project JXTA*, Available on line at http://www.jxta.org.
19. D. Sax (2003), *DNS Spoofing (Malicious Cache Poisoning)*, Available on line at http://www.sans.org.
20. T. Dierks and C. Allen (1999), *The TLS Protocol Version 1.0*, Available on line at http://www.ietf.org/rfc/rfc2246.txt.
21. F. Xhafa, Z. Arrizabalaga, J.M. Bertran, L. Barolli (2009), *Parameter Tuning of JXTA-based P2P Platforms*, Proc. of International Conference on Network-Based Information Systems (NBiS-2009), pp. 92-102.