

Platform and experimentation of secure service location with P2P/Client-Server over ad hoc networks *

Oscar Botero, Hakima Chaouchi

Telecom Sud Paris
CNRS SAMOVAR Lab, UMR 5157
9 rue Charles Fourier, 91011 Evry, France
{Oscar.Botero, Hakima.chaouchi}@it-sudparis.eu

Ad hoc networking is considered as an inexpensive solution for easy and fast wireless network coverage. However, for real service deployment, security and availability in such networks are still under research even though some experimentation has shown its viability. Additionally, peer-to-peer (P2P) networking has emerged as a key technology to enable resource sharing among users. As a result, the load charge on servers can be reduced by using P2P, but this approach presents a drawback related to the services availability; when a peer is not connected, its resources are not reachable. Alternatively, Client-Server architecture provides a centralized approach; but a high demand on client requests might deteriorate service response and capacity. In the case of ad hoc networks, offering services securely is even more challenging since all nodes are anonymous until being authenticated. In this paper, we propose to offer a secure location of services in an ad hoc environment. We focused on the development of a proxy-based solution to combine a P2P-client/server network overlay in order to provide securely the location information of network services (e.g. AAA service, Kerberos) and to set a platform that can include application services (e.g. multimedia content service). More precisely, we deployed a network overlay P2P platform using JXTA; a set of peer to peer protocols, and we developed simple peer nodes and a proxy server for providing and ensuring service access control. Other security services can also be deployed such as Kerberos, PKI, among others. In our platform, the JXTA was combined; through the proxy server, with a distributed Radius AAA server to perform authentication tasks prior to discovering the location of the required services in the ad hoc network. We added encryption and message authentication code mechanisms to provide a complete secured solution. GPS Bluetooth devices were used to obtain nodes location but other location systems are also possible. Finally, a detailed description of the system architecture and platform is provided describing all the modular components as well as numerical results derived from a simple test bed.

Index Terms— ad hoc networks, client-server, JXTA, peer to peer, network access control, network authentication, service location

I. INTRODUCTION

Providing services over an ad hoc network is a challenging issue from security, performance and feasibility points of view. To address this issue we combined P2P and Client-Server schemes to provide service access control and server location information. The P2P architecture provides a flexible approach when resources are shared between nodes. However, if peers are not available their resources and services are not reachable. A Client-Server architecture provides resources in a centralized manner offering control possibilities and network administration; conversely, server load can be affected when high rates of requests are managed. By combining both approaches we can rely on certain nodes to run some important network functions such as: AAA and main application services. The peers will provide their resources and services once available.

We implemented a network overlay, by using JXTA [1-4], which is a Sun's set of standard protocols. With these protocols, developers can be focused only on the application layer to implement P2P solutions co-existing with Client-Server platforms. We also developed a proxy server to manage service access control to peers on the application layer via an AAA server [5] and simple peers to be authenticated and that will obtain a list of available services with their location. The location information is provided from a GPS Bluetooth device which gives to the nodes the possibility to calculate their

geographical distance from the servicing node and get closer, in order to take advantages of specific applications e.g. social nets, content servers or network gaming.

The security solution uses Radius, EAP-MD5, Bulk Encryption, and Message Authentication Code (MAC). We used for simplicity Radius and EAP-MD5 to provide authentication on the application layer but other strong also authentication methods can be used. Symmetric key encryption and MAC (based on RC4-SHA1) were also used in order to protect the message exchange. The following security requirements are provided: confidentiality: the messages are not disclosed to unauthorized peers, authentication: the sender is the one who claims to be and data integrity: the message was not modified accidentally or deliberately in transit. The approach is a modular design implemented with JAVA SE programming language [6] that provides easy nodes functionalities upgrade.

By using a distributed Authentication Server we can implement different security approaches without modifying the JXTA protocols to grant access control. We are also open to use the authentication method that best fits the network requirements.

We focused particularly on the authentication in the ad hoc network to allow secure service location discovery.

The rest of the paper is organized as follows: In section II the problem is presented followed by section III where the platform architecture is described. In section IV the implementation is described followed by numerical results and finally the conclusions.

* This work is sponsored by ANR SARAH French National research project (2006-2009)

II. PROBLEM STATEMENT

The problem faced was how to provide LBS (location based services) discovery functionalities and to ensure at the same time authentication and secure communications among nodes in a general ad hoc environment. The proposal aims to authentication before service in the application layer by using the JXTA overlay with a proxy server connected to a radius server. The complete proposed solution is described in the following sections.

III. PROPOSED ARCHITECTURE

As stated before, we chose to use a P2P overlay to benefit from its flexibility in resource sharing and service exchange features and also we also considered a Client-Server approach to benefit from its centralized control. The services will be securely located thus allowing the user to estimate its distance from the requested service and to get closer to the servicing node if required.

The proposed solution consists of a JXTA P2P network, with simple peers that will perform authentication requests to our developed proxy server. It is to remark that JXTA itself provides an entry level security solution, where peers have their own certificate authorities and strong cipher functions are used. Despite that, no robust service access control is implemented. In order to implement it, a distributed AAA architecture is integrated into the overlay. We developed a proxy server in the P2P overlay to access the AAA server functionalities. This proxy is the interface between the distributed P2P overlay and the centralized AAA service; it will forward on behalf of the peers the requests to the Radius Server.

Encryption and MAC are used to keep the communications secure. GPS Bluetooth devices provide the current nodes' location; once collected and updated they will be transmitted to the trusted peers in order to communicate the list of available services and allow the calculation of the distance from them if necessary for some specific applications.

All the platform modules were developed by using the JAVA SE programming language. The modularity of the design allows adding new features or upgrading functionalities with hard changes neither on the platform nor on the nodes. Figure 1 shows the simplified building blocks of our platform. The JXTA protocols define the default group to communicate called "Netpeergroup". We defined a new group called "ServiceNetgroup" to provide our services; in our scenario the AAA service.

The basic operation of the platform is as follows (Figure 1): a peer (a node connected to the network, in our scenario it is an ad hoc node) will discover the P2P group advertisement which is published by the proxy server. The peer will send the accorded messages to the proxy in order to be authenticated by using a symmetric key security scheme. The proxy will forward the request to the Radius Server. If the peer is authenticated it will obtain a list of available services indicating the current position by GPS coordinates, allowing the calculation of the distance to them and to further demand the service required. However, if the peer is not successfully authenticated no information will be disclosed.

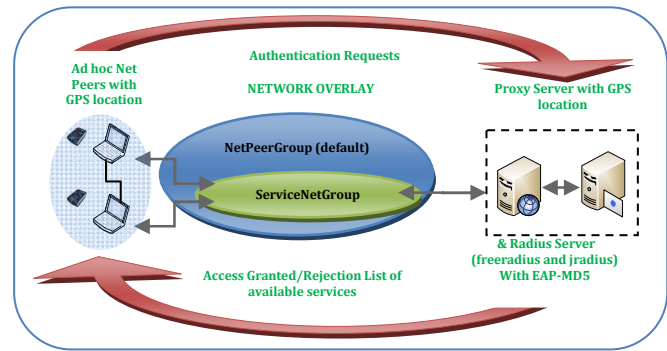


Figure 1. JXTA overlay over Ad hoc

IV. IMPLEMENTATION

As aforementioned, a combined architecture (P2P-Client/Server) solution based on JXTA protocols was implemented. We developed the simple peer application and the proxy server node, interacting with the AAA server by considering and implementing secure communication exchange between nodes as well.

Our contribution lies in providing the authentication prior to the service localization and access for ad hoc networks by using a set of P2P protocols to establish a network overlay and a proxy based solution and also providing the GPS location of the server and nodes to allow distance calculation. We proposed a modular solution that integrates the JXTA protocols, radius, GPS and security schemes.

The technology used and the overlay description is explained in the following parts.

A. Technology Details

In order to develop the proposed solution the following elements were used:

1. JXTA

"JXTA is a set of open, generalized peer-to-peer (P2P) protocols that allow any networked device — sensors, cell phones, PDAs, laptops, workstations, servers and supercomputers — to communicate and collaborate mutually as peers." [4].

JXTA name is derived from the word juxtapose, indicating that P2P solutions can co-exist with client/server solutions rather than replace them. In our case, we used the client/server based services such as AAA service via the P2P overlay. This allows us to benefit from P2P flexibility in resource sharing and service exchange as well as centralized control.

JXTA defines a set of six XLM-based protocols that allow the establishment and use of P2P networks. These protocols are independent of the underlying implementation. Functions include: *organize, discover and publish resources, communicate and monitor peer activity*. With these protocols a virtual network overlay can be established on top of physical networks in order to allow peers to interact no matter which network or connectivity type is used. The standard JXTA protocols are six: Endpoint Routing Protocol (ERP), Rendezvous Protocol (RVP), Peer Resolver Protocol (PRP), Peer Discovery Protocol (PDP), Peer Information Protocol (PIP), and Pipe Binding Protocol (PBP). These protocols are

independent from each other. A peer is not required to implement all of them to be a JXTA peer. It will implement only the protocols needed.

Basically JXTA provides a simple and generic P2P platform independent of the programming language used to implement it, the operating system, the network topology, and the security model used.

2. JAVA

Java is a very popular and functional object oriented platform independent programming language. The application was entirely developed using NetBeans IDE 6.1 with the UML plug-in [7]. The Java version used was J2SE 1.6 beta (Standard Edition), focused on desktops and personal computers with the Swing API for GUI development [6].

3. Security

In order to develop a secured application the following concepts were considered during implementation:

3.1. Encryption

In this work, bulk encryption is used. A bulk encryption algorithm uses a key to scramble data for transmission or storage. It can then only be decrypted using the same key [9]. The Encryption/Decryption process used for the implementation of this project is based on RC4 algorithm with SHA1 and a pre-defined symmetric key that can be modified on the application interface in mutual agreement.

3.2. Integrity

In order to provide integrity and authenticity to the messages that will be exchanged, a MAC (Message Authentication Code) is added. It is based on RC4-SHA1 and a symmetric key. The JxtaCrypto interface is used to access each of the services provided in the Java security package [10]. The MAC is added into the message and it will be verified when it is received, if the validation is successfully achieved the message has not been corrupted.

3.3. Access Control: Radius-EAP

Radius stands for: Remote Authentication Dial In User Service, and it is a networking protocol that provides centralized management of access to networks.

In order to implement the admission control with the Radius protocol the application freeradius V1.17 [5] is used in addition to the jradius module [8], allowing us to link the jradius Java with freeradius on the proxy server.

Jradius is an open source server-client framework oriented to Java implementations. The server module works together with freeradius through module. We used radius for service authentication an rlm_jradius control on the application layer.

EAP or Extensible Authentication Protocol is an authentication framework that provides the freedom of selecting any authentication mechanism. EAP typically runs directly over data link layers without requiring IP; in fact, EAP just provides some common functions and a negotiation of the desired authentication mechanisms, which are commonly called methods. We used EAP-MD5 method just for its simplicity on the implementation. We are focused on service access control on the application layer. For real deployment of the platform more secure methods can be used.

EAP-MD5 is a challenge/response protocol that uses the MD5 hash algorithm of 128-bit to verify the authenticity of user's password. It allows the user authentication to the server but does not allow any server authentication. This method is fast but vulnerable to man-in-the-middle attacks, dictionary attacks and exposes station identity; however, it is an interesting solution when power processing limitations on devices are present.

4) GPS

Global Positioning System is a global navigation satellite system that uses information from at least 24 Medium Earth Orbit satellites. The information obtained determines the location, speed, direction and time of the GPS device.

The GPS device used was the Holux GPS Slim 236 [11] (Figure 2) using Bluetooth and capturing the data in GPGGA format [12] (Figure 3) from a virtual serial port managed through the nodes developed.



Figure 2. GPS Holux Slim 236

\$GPGGA,161721.000,4837.4334,N,00226.5935,E,1,03,2.4,104.7,M,47.4,M,0000*5A.

Figure 3. GPS GPGGA format

To calculate the distance between the Server and the Client the Haversine formula [13] was used but other formulas can be implemented as required to gain in precision (this one provides 3m accuracy)

$$\rho = \text{hav} \sin \left(\frac{d}{R} \right) = \text{hav} \sin(l1 - l2) + \cos(l1) \cos(l2) \text{hav} \sin(\Delta\lambda)$$

$$d = 2R \arcsin(\sqrt{\rho})$$

Where:

$$\text{hav} \sin(\theta) = \sin^2 \left(\frac{\theta}{2} \right)$$

d = distance between the two points

R = radius of the reference sphere

$l1$ = latitude point 1

$l2$ = latitude point 2

$\Delta\lambda$ = longitude point 2 – longitude point 1

B. Application Description

There are two nodes that were developed, one is the proxy server, that will perform the service access control tasks by using a radius server and the other is the client that will be authenticated by using the P2P overlay. The proxy server and the radius one can be located in the same computer or in a distributed way.

In order to develop the applications the following equipments were used:

- Laptop Dell Inspiron 1525, Intel Core2Duo 2.4 GHz, OS Vista (Proxy Server and simple peers).
- Laptop Elonex, Pentium M OS Fedora Core 8 (Radius Server)
- GPS Bluetooth Holux slim236

The Java Libraries required for running the applications are the following:

Jcommport: is an open source library that allows the serial port management and data capture.

Swingx: Provides the graphical objects for the GUI.

JXTA 2.5: Provides the JAVA implementation of the JXTA protocols.

Jxtasecurity: It is used to perform cryptography and MAC functions.

The nodes perform the following functions:

1. Proxy Server:

The proxy server has the following features:

- Create the Peer Group that identifies the Access Control service.
- Capture GPS information from the Bluetooth GPS connected to the virtual serial port.
- Accept and process incoming peers' authentication requests.
- Perform cryptography functions with the data sent and received (Encrypt\Decrypt\MAC).
- Establish communication with the Radius server using EAP-MD5 on the application layer in order to authenticate peers (ad hoc nodes in our scenario).
- Send actual server GPS position to peers (Encrypted).
- Calculate peers' distance to server.
- Generates information related to the peers authentication attempts.
- Send list of peers with location and services provided to authenticated nodes if required.

2. Client:

The client node has the following features:

- Find the Peer Group advertisement published by the proxy server
- Perform cryptography functions with the data sent and received (Encrypt\Decrypt\MAC).
- Send actual GPS position (Encrypted) to the server.
- Calculate distance: peer to server.
- Extract peers available with their location.

The nodes can be deployed by using the JNLP (Java Network Launching Protocol) format that allows us to download the executable file from a server, or it also can be distributed as a jar file compressed with all the libraries required for running the application.

3. Security

The nodes provide the following security features:

3.1. ID and password for the JXTA network Configurator

The first time a JXTA application runs, we observe the JXTA network configurator that asks for the Peer ID: Used to identify the peer and as login for further uses and the Password: Used to identify the peer and to link it with the ID for authentication. We can also specify the type of peer (Relay, Rendezvous, proxy) and establish the HTTP and TCP settings. It is important to remark that when running several peers in the same computer, different TCP ports should be specified in order to perform the connections properly.

3.2. Radius Admission Control (EAP-MD5)

The API used for Radius authentication [5] allows implementing the EAP-MD5 process. This method can be replaced for more secure options.

For the client application the following parameters are required in order to be authenticated through the proxy server: radius login, radius password, radius secret, symmetric key, client ID.

Once the client has obtained the server's advertisement, it will try to be authenticated, so the client application will send the parameters to the proxy server, and then the authentication with the Radius Server is performed.

3.3. Cryptography

The messages exchanged are encrypted in order to secure them. The jxtasecurity library is used to perform this task by using RC4-SHA1 and a symmetric key. The messages are embedded into XML structures that allow us to label the fields and to extract the information easily.

3.4. MAC (Message Authentication Code)

The MAC is generated and inserted into a new field within the messages that will be used later for validation.

4. Application Message exchange:

In the following figure are shown the message exchange between the elements on the platform, (Figure 4) as well as the main screens of the nodes (Figure 5, Figure 6)

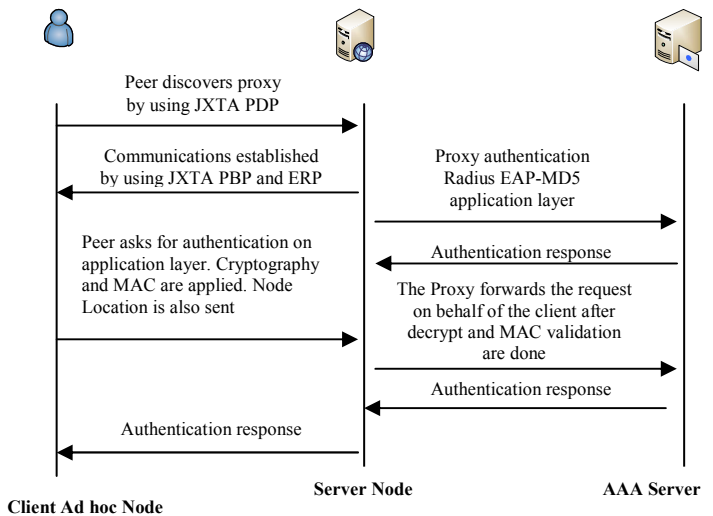


Figure 4. Platform message exchange

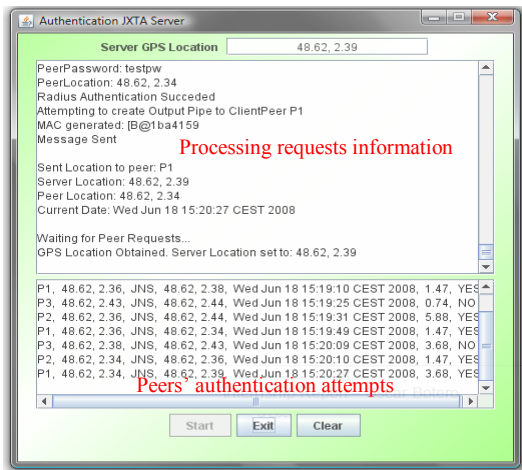


Figure 5. Proxy Server node GUI

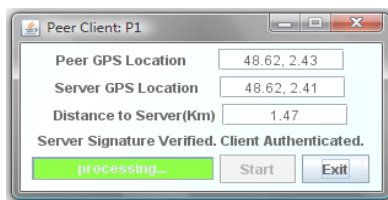


Figure 6. Client node GUI

V. DISCUSSION

We proceed with the measurement of the authentication delay time through our platform and we depicted the results in Figure 8. The results obtained with our testbed (Figure 7) are compared with some obtained for other research experiments for authentication delay by using locally radius on WiFi [14].

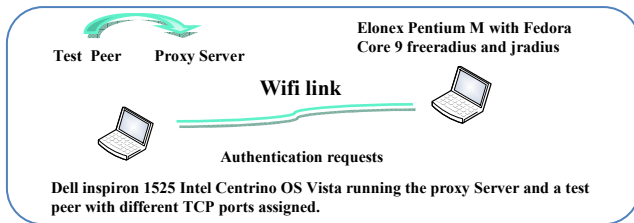


Figure 7. Testbed

Mean	Sample Variance	Confidence Interval 95%
61.6388	11.1925	61.6388 +/- 1.26867

Figure 8. Authentication delay in milliseconds.

The system developed is based on JAVA which is a popular programming platform independent language. New features can be added to extend the basic functionalities of the application in a modular way including the deployment of new services (multimedia content, social nets, gaming among others). Finally, it is easy to distribute and deploy the application for the clients by using *jnp/jar* formats. As mentioned earlier, we used EAP-MD5 method for simplicity but it presents vulnerabilities to man-in-the-middle attacks and dictionary attacks. The symmetric key approach can be used but requires an alternate channel to distribute the

key. The PKI approach can be used but the hardware limitations and processing power should be considered to make it feasible. We perform authentication before accessing the services and all communications between nodes being secure.

VI. CONCLUSIONS AND FUTURE WORK

We presented our solution to handle service access control by using a P2P-Client-Server JXTA platform. This is to allow trusted peer clients to request (in a secure way) location information (GPS) about services available in the same overlay. It is applied specifically for general ad hoc networks to allow only authenticated nodes to obtain the geographical location (and the distance) from servicing nodes.

The practical and original aspect of this application in ad hoc networks is to let the nodes to know securely the geographical distance from the servicing and peer nodes, and to get closer to them as it can be required by different applications such as social networking, content service with QoS among others.

The proposed application was developed in Java keeping the modularity as a strong aspect. Modularity allows to integrate different services running the same set of protocols or to add new features easily (e.g. multimedia content).

On the scope of this project, the application developed provides only communication between the nodes and the server that authenticates them. In order to extend the application functionalities, we can deploy multimedia services on the platform and observe the peers' interaction. We can provide tokens to authenticated peers in order to further request services. Finally, our future plan is also centered on performance measurements and protocol optimization.

VII. REFERENCES

- [1] JXTA: a network programming environment, Li Gong; Internet Computing IEEE, Volume 5, Issue 3, May-June 2001 Page(s):88 – 95.
- [2] Project JXTA 2.0 Super-Peer Virtual Network. B. Traversat, A. Arora, M. Abdelaziz, M. Duigou, C. Haywood, J. Hugly, E. Pouyoul, B. Yeager. May 25, 2003.
- [3] <https://jxta.dev.java.net/> (jxta main web site)
- [4] JXTA Java Standard Edition v2.5: Programmers Guide September 2007.
- [5] Radius server, <http://www.freeradius.org/>
- [6] <http://java.sun.com/>
- [7] Java IDE, <http://www.netbeans.org/>
- [8] <http://coova.org/wiki/index.php/JRadius>
- [9] FIPS 186, "Digital Signature Standard (DSS)". 1994 May. National Institute of Standards and Technology
- [10] Java 2 Network Security, M. Pistoia, F. Reller, June 1999 Prentice Hall PTR
- [11] GPS product details, <http://www.holux.com>
- [12] GPS messages details, <http://aprs.gids.nl/nmea/#gll>
- [13] R. W. Sinnott, "Virtues of the Haversine," Sky and Telescope, vol. 68, no. 2, 1984, p. 159
- [14] An authentication scheme for fast handover between WiFi access points. A. Bohák, L. Buttyán, L. Dóra Laboratory of Cryptography and Systems Security (CrySyS) Budapest University of Technology and Economics, Hungary WICON 2007, October 22-24, 2007, Austin, Texas, USA