SECURE MOBILE CLOUD COMPUTATION OFFLOADING: NODE PRIVACY AND SECURITY

David I. Fadaraliki[1], S. Rajendran[2]

[1]*M.Tech Student, I.T. Department, SRM University, Tamil Nadu, India*
[2]*Professor, I.T. Department, SRM University, Tamil Nadu, India*

Abstract - Mobile agents provide better computation offloading from the mobile device to the cloud environment. They use https based Message Transfer Protocol (MTP) communication mechanism, from this premise it can be concluded that the communication channel is secure. However security concerns are raised at the processing nodes. Agents need to be protected at the host and from other malicious agents. In this paper, we present the design and implementation of a multi-agent framework on the Android platform, which is built on the Foundation for Intelligent Physical Agents (FIPA) specifications compliant Java Agent Development (JADE). In our design, the framework allows developers to develop applications with high CPU usage and battery consuming running on the framework. The framework then offloads the task to the cloud, bringing back the results to the device. Agents in the system are aware of resource constraints such as battery capacity, CPU usage, and dynamically adjust their behaviours accordingly to achieve a balance between the resources consumption and security expectations.

Keywords: Mobile agents, offloading, communication channel, security

1. INTRODUCTION

Android has gained popularity rapidly as a mobile device operating system in recent years. In this light, it is forming the basis of Mobile Cloud Computing (MCC). The rapid progress of MCC becomes a powerful trend in the development of IT technology as it increases enterprise mobility in commerce and industry fields. However android mobile devices are usually constrained by resource limitations such as: battery life, lower CPU frequency, limited memory and storage.Battery usage can be improved by reducing the CPU usage [1]. A vibrant mechanism to reduce CPU usage in smartphones is to offload computation to remote locations. Previous works have been successfully conducted using various technologies, to offload computation capabilities to the cloud. Concerns were however raised on security measures to protect the data that have been offloaded.

Mobile agents facility a more efficient way of offloading data from the device to the cloud; taking advantage of capabilities to reduce network latency and secure communication channel [2]. The mobile agents bundle the code, data and the instructions to execute to a remote location. In this research, we use java JADE mobile agents. JADE agents are built on FIPAspecification for standardisation and Agent Communication Language (ACL) for communication. The protocol they use is HTTP based MTP [3] which uses the Java Messaging Service (JMS) to deliver inter-platform communication between agent platforms. An analysis of various multi agent platforms is conducted to determine amongst other the advantages of using JADE.

Because the data is processed in the cloud, security and privacy settings depend on the IT (information technology) management of the cloud providers. Loopholes in the cloud might result in a breach of privacy. Authenticity and integrity of each device connected to the cloud needs to be determined. Data, instructions and code is transmitted between nodes (service provider and mobile device) as plain code. Algorithms to encrypt the data may reduce overall performance as they are time consuming and require resources. Agents need to be protected from malicious hosts and/or from other agents.

2. BACKGROUND

2.1 Offloading Overview

With the rise in demand of computational resources by mobile devices, various solutions for computation offloading to more powerful virtual machines on the cloud have been proposed by various scholars. An analysis of some of these techniques has been done. The background of our study is basing on the flaws of these techniques.

A. MAUI Interoperability Issues

MAUIprovides method level code offloading for Microsoft .NET applications [6]. This framework is a real time application, offload decisions are made at runtime. The weakness of MAUI is thatit is more dependent on the hardware architecture of the hosts. Mobile devices typically have different CPU instruction architecture than desktops and servers. Processes cannot be easily run on devices with different architectures.

B. CloneCloud Security Issues

CloneCloud is a system that automatically transforms mobile applications to benefit from a cloud environment [7]. The system has a runtime application partition that works on the mobile applications to offload part of the code and executions from mobile devices onto device clones (application level virtual machines) operating in the cloud. One of the weakness of CloneCloud is security is assumed basing on the trusted virtual machines, all the device data is replicated in the cloud to ensure synchronous virtualization between the device and its clone.

C. ThinkAir Security Issues

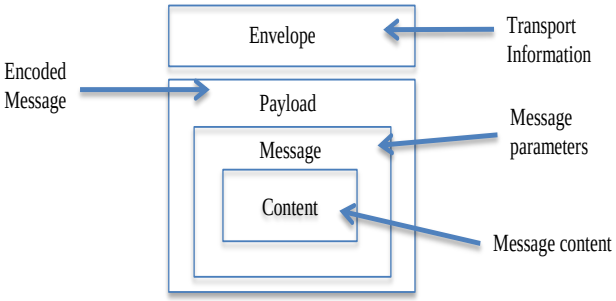
ThinkAir provides on demand resource allocation and parallel execution on the cloud operating at method level [8]. They used 11 Android phones and device clones running on Amazon EC2. Mobile devices are used to store data that are personal data, and also for economic transactions. The weakness with ThinkAir is that, it assumes a trustworthy cloud server execution environment: there is hope that whenever data is offloaded to the cloud, the code and state of the data are not maliciously modified or stolen. The authors assume that the server faithfully loads and executes any code received from the devices.

2.2 Why use Mobile Agents

The use of mobile agents in relation to this research is coupled with a number of security advantages which include the following;

- Mobile agents encapsulate data, instructions and protocols.
- Ability to move transparently across different hosts.
- Capable of adaptive learning and automation.
- Some intelligent agents are equipped with techniques to check self-integrity.

The diagram (fig. 1) portrays the generic structure of an ACL message. The message envelop contains the addressing details (source and destination Agent ID-AID). It have the communicative intention, also called performative (accept-proposal, request, inform, propose etc.)[9]Android code and data is transmitted between nodes bundled as the messages content. Execution is completed at the receiver terminal



2.3 Agent Development Toolkits

Developing as well as usingmobile agents obliges the use of good underlying infrastructure [4]. Over the years it has been widely noted that scarcity of agent development tools has limited the exploitation of this beneficial technology. As technology evolves, nowadays a wide variety of tools are available, for developing robust infrastructure. According Nguyen & Dang [5] there are more than 100 toolkits in this category but at this juncture to the space constraints, we would be focusing on most appealing toolkits among the available choices.
The toolkit of choice in this work is JADE, because it simplifies the implementation of multi-agent systems through a middle-ware that claims to comply with the FIPA specifications and through a set of tools that supports the debugging and deployment phase. [10] Recently, JADE had an android extension (Light Extension Agent Programming - LEAP) which allows agent programming on android environments. This extension limits the memory usage on the device.

2.4 Agent Security

Mobile agents encounter threats which may be realised by different security mechanisms [14]. Below, the applicability of services and mechanisms with respect to mobile agents is evaluated.

A. Authentication

An agent's code can be said to constitute fundamental properties of an agent instantiation. An agent's duties can be authenticated through a signature over the intrinsic properties of that agent. This will also authenticate the agent, if the agent name is included, and one trusts the authority. Alternatively, authentication of the agent can be done through a separate signature by another trusted authority.

B. Access control

Access control in an agent system shall protect the following access categories:

- Agent to agent platform, to ensure that only authorised agents can run on a host;
- Agent to host (via the agent platform), e.g. for access to information in the host;
- Host to agent, which can only be controlled if the agent platform consists of trusted hardware;
- Agent to agent, controlled by the agent platform.

C. Integrity and confidentiality

Integrity and confidentiality of information in the host systems must be preserved by proper access control, and a virtual machine that ensures that agents cannot circumvent the access control procedures.An agent may carry information that needs protection with respect to external.

.

3. DESIGN AND IMPLEMENTATION

The design methodology involves the development of the Java based Framework compatible with Android application (.apk) development. Android application developers have the option to evaluate the CPU usage of their applications. For high CPU consuming applications, they have the options of developing the applications on the framework to allow for computation offloading to the cloud environment. The framework is a generic agent environment which can be customized to meet each application's offloading requirements, essentially providing secure execution of the agents. Energy profiling is the analysis of the energy usage of applications [13]. This can be useful for aiming to measure resource consumption. An energy profiler is an application that can aid in this analysis. The profiler can be used to set the CPU usage threshold of an application to label it as a high energy consumer. Low consuming applications can execute locally, with high end applications migrating to the cloud.

The proposed method for code migration on JADE-LEAP platform is to wrap the state of an agent and its code in the FIPA ACL message and send it to its destination [12]. While analysing this proposal two approaches can be considered:

• Simple mobility: an agent sends a request for migration in the form of a FIPA ACL message to its local container, which is than responsible for locating all of agents code and to transfer it along with agent's state to the destination container.
• Full mobility: an agent communicates directly with the destination container, thus creating a new instance of it and sending its internal state to the newly created instance.

Execution commences at the cloud environment, results brought back to the android device without the knowledge of the user. The communicating agents are equipped with behaviours to detect the environmental changes and anomalies. Integrity checks on the input data to avoid tampering and adaptive procedure learning are some of the security features that the agents are equipped with to avoid malicious hosts and agents. Detection of tampering causes the agents' stateful data migrate to another platform to continue the execution process.

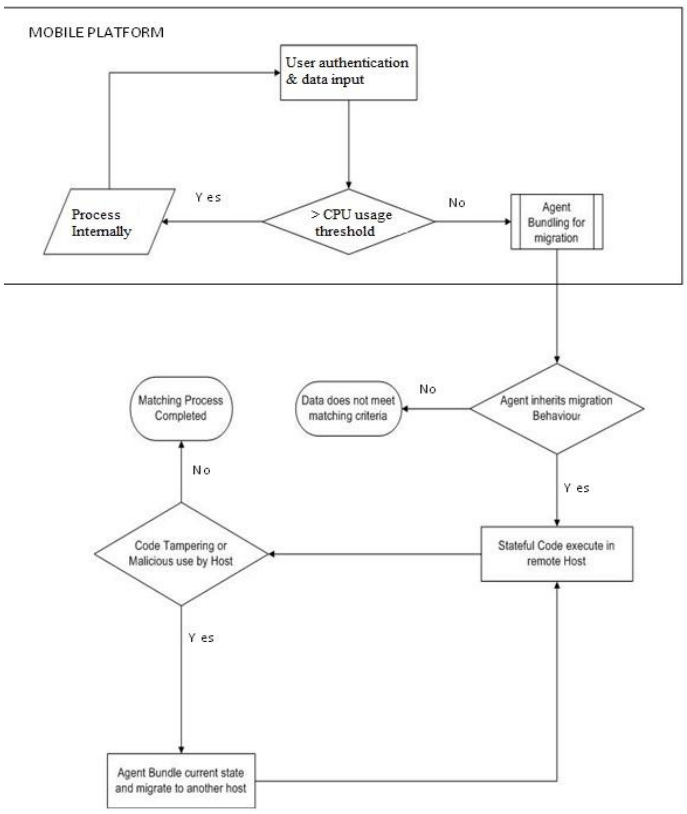


Fig. 2 Data Flow Diagram – Proposed Structure

In this work, a high consuming test-bed application is developed and executed on two different test environments (Locally on the android device and offloaded to the cloud). The application running as the test-bed is the n-queens problem. N- Queensis a backtracking example that asks for the arrangements on n number of queens on an nXn chess board such that no queen attacks the other [11].CPU usage and execution times for the two instances is recorded.

The control backtracking algorithm is tested for 30-Queens on a 30X 30 2-dimensional matrix simultaneously on the offloaded data and computation done locally on the android device.

3.1 Local/Traditional android Application

Execution time for the local application is recorded as 730257milliseconds, which translates to 12minutes and 10 seconds to complete the task. However, it is not the execution time that is of paramount importance, but the CPU usage during the computation time.

Fig. 3 Local Execution Android Interface

3.2 Agent Based Offloading

A. On the cloud, a jdk enabled virtual machine runs the agent platform. The platform via the main listens and accepts incoming connections from other containers and agents. The main container uses the container ID (IP address192.168.11.177) and port number 1099 as the listening connection. This means that the main container should be running at all instances, or should be started first before any other processes can be initiated.

B. Once the main container is running, the android device is configured manually to connect to the platform. The user needs to know the main container ID and the port before initializing a connection request. A successful connection means the android agents can communicate with the agents residing on the platform.

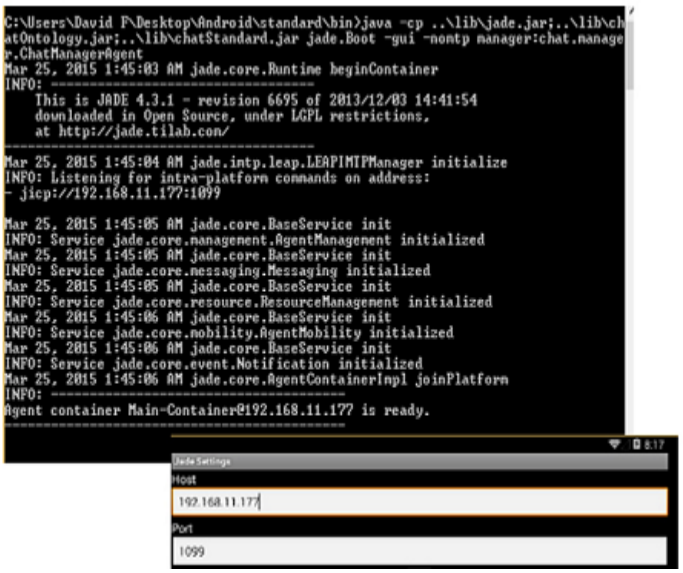


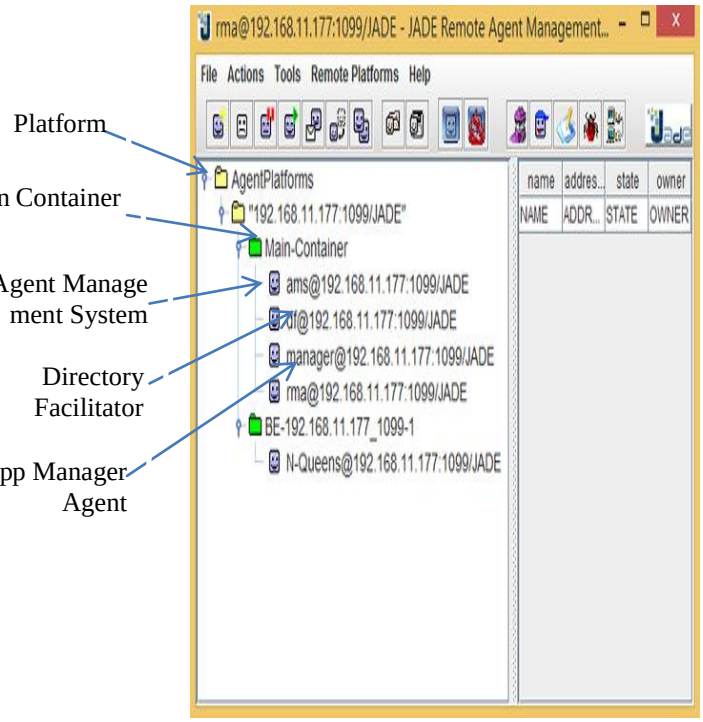
Fig. 4 Fig. 4 Agent to Platform Connection

C. Once the connection has been established data can be transmitted between the two nodes for processing.

Agent Management System - provides the naming service; name, ensure uniqueness, create/destroy agents.

Directory Facilitator - provides a Yellow Pages service of which an agent can find other agents providing the services it.

The manager – tasks the platform to compute the data that has been acquired. This is customizable depending on the application being developed



The process of computing the 30X30 N-Queens on the cloud and rendering data back on the android device takes 630526 milliseconds which translate to 10 minutes 30 seconds. Again the CPU usage time being recorded during the execution process.

4 RESULTS

A. CPU Usage

$ adb shell top -m 10

This is an Android Shell command to record the top CPU using android applications. Percentage usage is manually recorded for the initial 10 seconds of initialising the application. Comparison on the 30X30 matrix indicates that processing the data locally in the mobile device consumes more that activating agents that offload the data to the cloud.

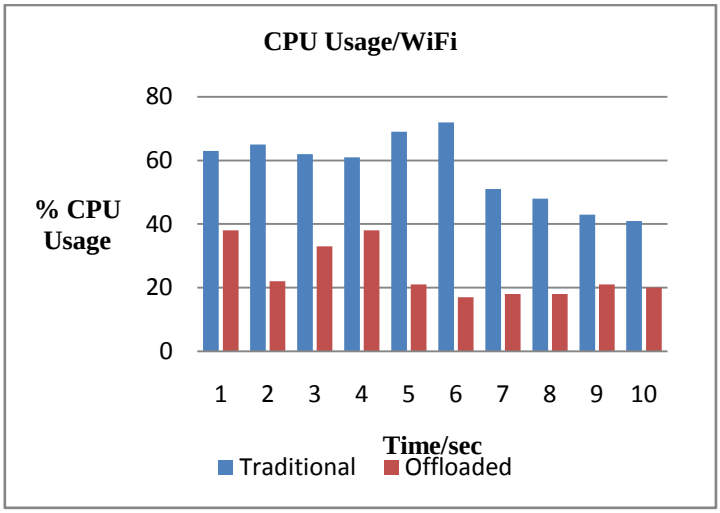


Chart 1 CPU Usage Local vs Offload

B. When and when not to offload

Offloading decisions are made in cases when the overall cost of executing local outweighs the cost of offloading for a single application module:

if (cost_to_offload<cost_to_execute_locally)
then offload
else
execute on device

Where the cost_to_offload = time to send code and data to cloud + time to get response from cloud + time to execute in cloud.

Energy saved by offloading data can be calculated by as follows;

$$Energy\ Saved = P_c \times \frac{C}{M} - P_i \times \frac{C}{S} - P_{tr} \times \frac{D}{B}$$

S: Speed of cloud to compute C instructions
M: Speed of mobile to compute C instructions
D: Data need to transmit
B: Bandwidth of the wireless network
P_c: Energy cost/sec when the mobile phone is doing computing
P_i: Energy cost/sec when the mobile phone is idle.
P_{tr}: Energy cost/sec when the mobile is transmission the data.

Noticeably, an increase in the bandwidth against the data to be processed reduces the overall energy saved, an indication that offloading is ideal for high processing

applications, and avoids communication activities (network traffic).

5 CONCLUSION AND FUTURE WORK

Offloading is a viable solution for mobile cloud computing, unfortunately available solution has limitations in security and interoperability amongst other issues. The use of mobile agent powered offloading addresses the issues mentioned above. The primary objectives of mobile cloud computing is to improve battery life for mobile devices and also the processing capabilities of the devices. Our research has practically proved these features. The limitation of JADE is that it only works on JAVA based operating systems (android for smartphones).

In future, offloading decisions need an algorithmic approach(i.e. when and when not to offload).The framework needs to dynamically set the threshold in relation to the real-time resource availability. CPU usage is allocated to running applications at any given time; the framework should utilize its allocation on offloading decisions

REFERENCES

1. Ari Cotler, Dean Shi, OnurSahin, AyseCoskun"Power Aware Computing Hardware Management Software for Android Mobile Systems"
2. Salah Jowan, Tony Mullins "A Java-based Mobile Agent Framework for Distributed Network Applications"
3. Curry, E.Chambers, D., Lyons, G."A JMS Message Transport Protocol for the JADE platform"
4. Aarti Singh, Dimple Juneja, A.K. Sharma, "Agent Development Toolkits", International Journal of Advancements in Technology Vol 2, No 1 (January 2011)
5. Nguyen G., Dang T.T., Hluchy L., Laclavik M., BaloghZ.andBudinska I., 'Agent Platform Evaluation and Comparison', Published by Institute of informatics, Slovak Academy of Sciences, Pellucid 5FP IST -2001-34519, June 2002.
6. Eduardo Cuervo, ArunaBalasubramanian, Dae-ki Cho, Alec Wolman , Stefan Saroiu, Ranveer Chandra , ParamvirBahl, "MAUI: Making Smartphones Last Longer with Code Offload"
7. Byung-Gon Chun, SunghwanIhm, PetrosManiatis, Naik, Ashwin Patti, "CloneCloud: Elastic Execution between Mobile Device and Cloud"
8. SokolKosta, AndriusAucinas, Pan Hui, Richard Mortier, XinwenZhang,"ThinkAir: Dynamic resource allocation and parallel execution in cloud for mobile code offloading"
9. Anton Naumenko, "Programming Agents with JADE for Multi-Agent Systems", Industrial Ontologies Group
10. http://jade.tilab.com/
11. S. Pothumani, "Solving N Queen Problem Using Various Algorithms – A Survey", International Journal of Advanced Research in Computer Science and Software Engineering, February 2013 ISSN: 2277 128X
12. S. Robles, J. Borell, and J. Ametller, "Agent Migration over FIPA ACL Messages," in *5th InternationalWorkshop, MATA 2003, October 8-10*, Marrakech, Morocco, Nov. 2003, pp. 210-219. [Online].http://jade.tilab.com/papers/EXP/Sergi.pdf
13. Alexander Bakker, "Comparing Energy Profilers for Android"
14. KristianSchelderup, Jon Olnes, "Mobile Agent Security – Issues and Directions"

