BUILDING A DISTRIBUTED TRUST MODEL OF RESTFUL WEB SERVICES FOR MOBILE DEVICES

A Thesis Submitted to the College of

Graduate Studies and Research

In Partial Fulfillment of the Requirements

For the Degree of Master of Science

In the Department of Computer Science

University of Saskatchewan

Saskatoon

By

Min Luo

PERMISSION TO USE

In presenting this thesis in partial fulfilment of the requirements for a Postgraduate degree from the University of Saskatchewan, I agree that the Libraries of this University may make it freely available for inspection. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor or professors who supervised my thesis work or, in their absence, by the Head of the Department or the Dean of the College in which my thesis work was done. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of Saskatchewan in any scholarly use which may be made of any material in my thesis.

Requests for permission to copy or to make other use of material in this thesis in whole or part should be addressed to:

Head of the Department of Computer Science

176 Thorvaldson Building

110 Science Place

University of Saskatchewan

Saskatoon, Saskatchewan (S7N 5C9)

ACKNOWLEDGMENTS

First of all I would like to thank my supervisor, Dr. Ralph Deters for his support, guidance, patience throughout my graduate study. I would like to thank Dr. Julita Vassileva, Dr. Gord McCalla and Dr. Chris Zhang. I also would like to thank other faculty &staff at the department of Computer Science, who has been very helpful throughout my study at University of Saskatchewan. Finally, I would like to thank friends and classmates at MADMUC lab for their selfless support.

ABSTRACT

As of 2011, there were about 5,981 million mobile devices in the world [1] and there are 113.9 million mobile web users in 2012 [2]. With the popularity of web services for mobile devices, the concern of security for mobile devices has been brought up. Furthermore, with more and more cooperation of organizations, web services are now normally involved with more than one organization. How to trust coming requests from other organizations is an issue.

This research focuses on building a trust model for the web services of mobile devices. It resolves the issues caused by mobile devices being stolen, lost, users abusing privileges, and cross-domain's access control. The trust model is distributed in each node of the web servers. The trust value is calculated for every incoming request to decide whether the request should be served or not.

The goals of the trust model are 1) flexible; 2) scalable; 3) lightweight. The implementation is designed and accomplished with the goals in mind. The experiments evaluate the overhead for the trust module and maximum capacity of the system.

TABLE OF CONTENTS

page

LIST OF TABLES

LIST OF FIGURES

List of Acronyms

ABAC .....Attribute-based access control

AES .....Advanced Encryption Standard

CDACM ..... Context based dynamic access control model for web service

DAC .....Discretionary access control

DES ..... Data Encryption Standard

ESB ..... Enterprise Server Bus

GPS .....Global Positioning System

IAAS .....Infrastructure as Services

LBS .....Location Based Service

MAC .....Mandatory access control

MD ..... Message-Digest

OC .....Opinion Credibility

OW ..... Opinion Weight

P2P .....Peer-to-Peer

PAAS ..... Platform as Services

PCA ..... Proof-Carrying Authorization

PDA ..... Personal Digital Assistant

RBAC ..... Role-based access control

REST .....Representational State Transfer

RPC ..... Remote Procedure Call

SAAS ..... Software as Services

SMS ..... Short Message Services

SOA .....Service Oriented Architecture

SOAP ..... Simple Object Access Protocol

UDDI ..... Universal Description Discovery and Integration

WACUAB ..... Web Access Control using User Access Behavior

WSCL ..... Web Services Conversation Language

WSDL ..... Web Services Description Language

WTV ..... Weighted Trustworthiness Value

XML ..... Extensible Markup Language

CHAPTER 1

INTRODUCTION

Mobile devices, such as BlackBerry phones, iPhones and Personal Digital Assistants (PDA) are widely used. As of 2011, there were about 5,981 million mobile devices in the world [14]. Mobile devices have evolved over the years with wireless connection, increased storage and memory, computing abilities and built-in sensors. As a result, mobile devices are no longer just for phone calls or Short Message Services (SMS), they are used for social activities, commercial, entertainment, personal care and personal health and business. There are 113.9 million mobile web users in 2012 [9], and mobile networks have developed from primary 2G (GSM, iDen), 2.5G (GPRS), 3G (EDGE, CDMA2000) to 4G (Mobile WiMAX).

With the improved transfer speed, variety of mobile network channels and unlimited mobile applications there is imperative need to address critical security issues such as: 1) how to protect data that is transferred by wirless connection or stored in a portable device; 2) how to properly use corporations' applications through mobile devices; and 3) how different corporations' mobile applications trust each other.

Many mobile applications are involved with sensitive information. Corporations, for example, store financial business information on their servers that their accountants access. When an accountant requests financial reports through his BlackBerry phone on a business trip, this mobile device may be lost or stolen, and the sensitive financial data can be leaked which can cause enormous profit loss for this corporation.

Mobile computing has more security issues than traditional computing systems. From the mobile network's perspective, the connected network is constantly changing. Mobile devices

connect to the internet using different infrastructures. For example, Wi-Fi and Bluetooth are used for short distance communication; while the cellular telephone network is used for long distance communication. Mobile Networking has these vulnerabilities: 1) The insecurity of the wireless links; 2) No centralized authorization; 3) Energy constraints; 4) Relatively poor physical protection of nodes in a hostile environment; and 5) Malicious nodes in ad-hoc network [15]. Since mobile devices constantly move in and move out of the mobile networks, there is no traditional administration and authentication, and the mobile network is susceptible to link attacks such as eavesdropping, message distortion and message replay.

Mobile devices are relatively small and portable computers, which can not only be the targets of malicious attacks, but also the tools for attacking. Mobile devices have more exposure than traditional computers, and they can be maliciously used by someone to attack host servers. The sensitive information stored on mobile devices can be leaked due to being lost or stolen and transferring such data through wireless network can cause data leaking.

Recently, there has been a growing interest in opening the web service of corporate systems to access through smart phones and tablets as a means to increase employee productivity and to simplify access to IT services. Furthermore, there are interactions between mobile applications for each organization. Because of these reasons, mobile devices can cause either corporate data or personal data leakage and even corporate network attacks. How to protect business and personal sensitive information is a great challenge for mobile computing. In this research, I proposed a distributed trust model built on RESTful web services for mobile devices. This trust model evaluates each incoming request's trust value, thus reduces the security issues of web access for mobile devices. For instance, in a hostile environment, a mobile device cannot access data through the corporate web applications due to the being at high risk location.

The rest of the proposal is organized as follows: chapter 2 identifies the problems current corporate web services have; chapter 3 reviews previous research about web services, mobile devices' context information and building trust models; chapter4 describes the architecture of the proposed system; chapter 5 illustrates every component of the system and implementation details; chapter 6 presents the results of the experiments; and chapter 7 summarize the research and list the tasks for future work.

CHAPTER 2

PROBLEM IDENTIFICATION

2.1 Web Services for Mobile Devices

With the increasing wireless bandwidth, CPU speed, memory capacity and disk storage, mobile devices are now beginning to be used increasingly often as platforms for accessing IT resources. Corporations allow their employees using mobile devices to access enterprise web applications to improve efficiency and reduce cost. Figure 2.1 presents the structure of currently used web services access control.

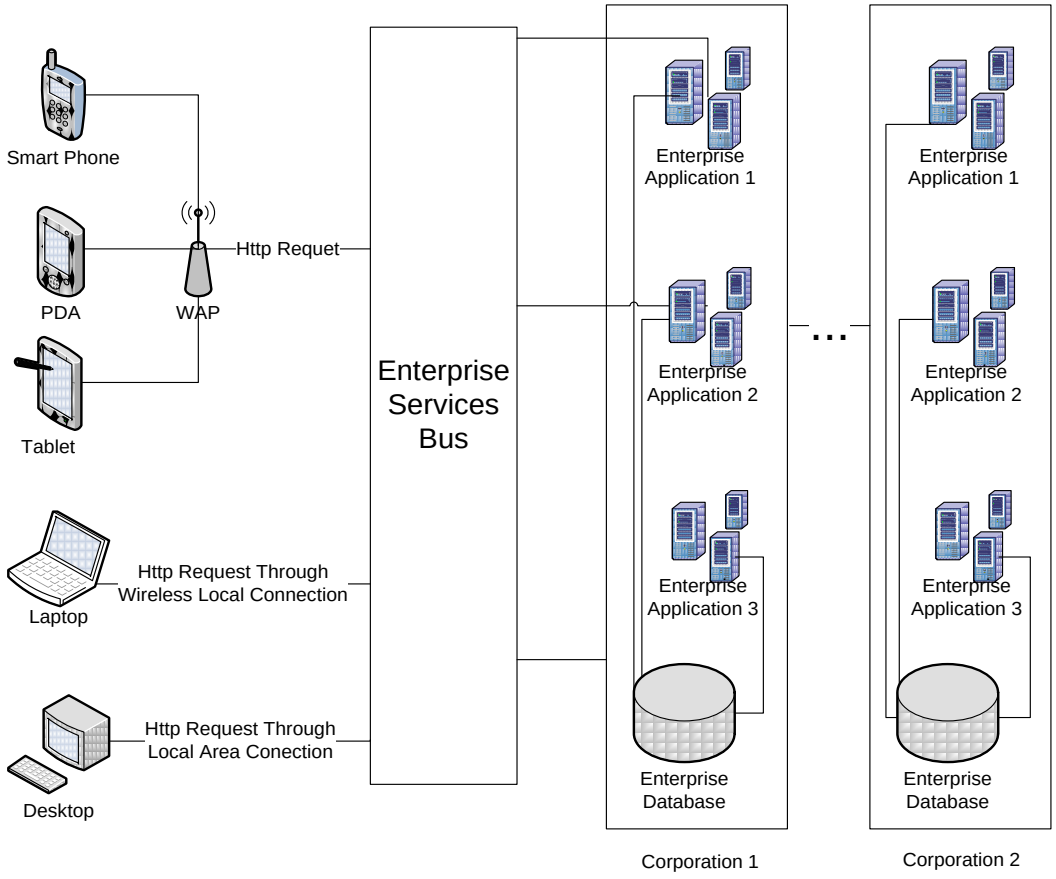

Figure 2-1 enterprise web services architecture

Figure 2-1 shows the standard enterprise web services architecture. It includes clients, Enterprise Service Bus (ESB), applications and databases. The clients can be mobile devices, laptops or desktops. Normally mobile devices like smart phones, tablets, PDAs connect ESB via mobile networks or Wi-Fi; Laptops connect ESB through Wi-Fi or fixed cables; Desktops connect ESBs through fixed cables. ESB then forwards these HTTP requests to different web applications based on the request types. Web applications respond to these requests, retrieve information from databases and send them back to the clients. The requests can be transferred to other ESBs of different corporations if needed.

When it comes to mobile devices' interactions with the web services, there are challenges for security and trust requirements. Mobile devices are typically small and portable devices which use no-fixed infrastructure, no central administration network to access enterprise web services. The convenient of access and vulnerability of network introduces more attacks. The potential risks for mobile devices in composite web services system are:

P1) Lost or Stolen Devices: Mobile devices tend to be less (or not at all) secured, since their users want fast access and prefer to avoid tedious login procedures due to the text interface constraints of mobile devices; and the mobile devices are usually small and carried to all kinds of place, so the chances for being used by other people or stolen are much higher than for a traditional desktop/laptop. Not only may some resource be leaked out to unauthorized users, but it also can be used as an attack tool for some fragile web services.

P2) Abuse of Privileges: Even legitimate users can cause security issues due to their misbehavior. Since the mobile devices are carried everywhere all the time, the chance for abusing privileges is much higher than for desktops/laptops.

P3) Traditional Access Control Issues: Composite web services, especially some cross-domain web services, contain multiple tasks. This leads to the question of how to aggregate multiple services into one logic unit safely. By using authentication access control, we assign users privileges for each domain and service. While assigning a user privilege for a certain task is relatively easy, it is complicated to assign privileges to a user for multiple tasks. Assigning a user privileges cross-domain can cause potential risks. Composite web services must keep changing to adopt to dynamic and increasing business requirements, but current authorized web access may not be sufficient for future or have more than enough access privilege. Under cross-domain web services, more than one organization are involved which makes the management of web access more difficult.

To resolve these risks mentioned above, I propose a trust/reputation mechanism to enhance security for composite web services. As shown in figure 2-2, there are trust modules and trust policies for each web services. When mobile devices invoke web services, they send context information along with http requests. The trust modules handle the http requests before they are sent to web services and calculate trust and reputation of coming requests based on trust policies set by the corporation. If the trust value of the request is not higher than a certain value, it is rejected by the system. Thus, in addition to the traditional authentication access control, we enable trust/reputation mechanisms for enterprise web services.

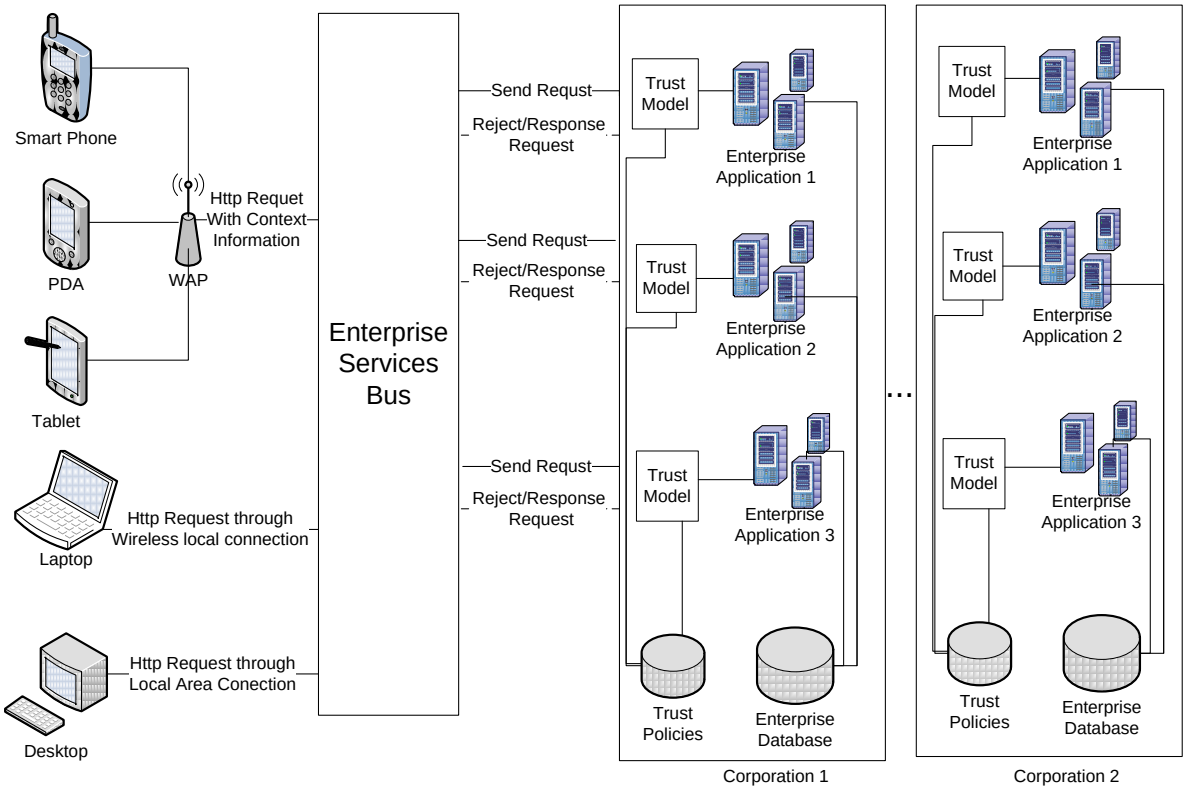

Figure 2-2 proposed enterprise web services architecture

2.2 Research Goals

Since there are distributed trust models and trust policies data for each web service, minimal overhead and robustness are required for the system. This leads to 3 goals; the system should be

G1) Flexible: To adopt the dynamic changed business processes and policies, this trust model should be flexible. Trust policies can be adjusted smoothly.

G2) Scalable: With the rapid development of mobile devices and the enormous variety of mobile services, scalability becomes an important factor. The system should be able to handle a large amount of concurrent requests without significant performance degradation. It should be easy to set up more nodes with hardware and software.

G3) Lightweight: Even though the performance of hardware for mobile devices has significantly improved, there is still a limit for wireless connection, computing capability and storage. Reducing the complicity of user interface and minimize the data flow are needed.

CHAPTER 3

LITERATURE REVIEW

This research is creating a trust model for mobile web services under the enterprises environment based on mobile context information. The system should reinforce the enterprise's trust policies; the system's features should meet our goals: flexible, scalable and lightweight. Previous research has been conducted related to web services, web access, enterprise service bus, cloud computing, mobile context, mobile comfort concept and trust models, which are reviewed in this chapter. Based on the literature review, the most suitable architecture structure and approaches are chosen to implement this distributed trust model system.

3.1 Web Service

According to the World Wide Web consortium (W3C) [31], web service is "a software system designed to support interoperable machine-to-machine interaction over a network" [31]. There are three types of web services, from Remote Procedure Call (RPC) that goes back to 1970 to the current Service-Oriented Architecture (SOA) and Representational State Transfer (REST).

RPC is initialized by a client machine that calls a server machine according to a stub, a contract between a client and a server. It is criticized for its tight coupling compared with SOA.

SOA was first introduced in 1998 in a project for Microsoft, and is a collection of services using SOAP (Simple Object Access Protocol) as a way to transfer messages. SOAP relies on an XML formatted message with an envelope head which contains metadata and an envelope body which contains the actual data. Web services are needed to be published and discovered after being created, so potential consumers can access them. [24] "SOA model is usually composed of three actors and three operations. The three participants are service provider, service registry and service consumer; the three basic operations are publishing, find and bind." [24]. Service

provider is an addressable entity which provides, publishes the service and interfaces the service contract to the service register. The Web Services Description Language (WSDL) is an interface of the service contract. It explains web services' functionalities, parameters and return data structures. Service providers register their services at Universal Description Discovery and Integration (UDDI). UDDI registry, the services registry, offers a standard mechanism to classify, catalog and manage web services, so they can be discovered and utilized. It also provides inquiries of the services for the service requestors. The service requestor finds the services through the service registry and binds the services through the transport mechanism. According to the contract interface, the service requestor invokes the services [24].

A REST web service is a resource-oriented service which is based on web standard and HTTP protocols. Roy Fielding [11] first introduced the concept of REST in his 2000 dissertation. Fielding derived REST by adding a set of constraints. The first constraint uses a client-server architecture, which separates a user's interface and data storage. The second constraint is stateless communication, which improves visibility, reliability and scalability. The third constraint is caching that improves efficiency. The fourth constraint is uniform interface. REST web services supports HTTP methods such as POST, GET, PUT, and Delete. These methods enable developers to manipulate resources through the Create, Read, Update and Delete (CRUD) operations. The generality of the component interface makes the system simpler and more visible. The fifth constraint is a layered system that simplifies components and improves system scalability. The sixth constraint is Code-On-Demand which improves system extensibility but trades off system visibility. [11] "The key abstraction of information is a resource." [11]. A resource is identified by URI (Universal Resource Identification). Overall, the features of REST enable the system's generality and scalability.

Compared with SOA and REST, RPC is not supported by certain vendors due to scalability issues and tight coupling. Nowadays, SOAP services and RESTful services are pervasive. The debate between the two services is on-going. According to the Gartner Survey in 2008 [13], there has been an increase in the number of organizations implementing web services using REST. When asked to indicate their past, present, and estimated future use of SOAP-based web services vs. REST-based web services, respondents showed a marked drop-off in use of SOAP; from 54% in 2008 to a projected 42% in 2009 and 2010. The number of organizations primarily using or considering REST-based web services is predicted to grow by a proportional amount, from 14% to 24% over the same time frame.

Architecture, technology and practice for the RESTful web service and the SOAP web service are compared in table 3-1.

Table 3-1 SOAP and REST

	SOAP	**REST**
Invoke from	Endpoint	URL
Transfer Message	SOAP (Simple Object Access Protocol)	Plain xml & JSON (Usually lightweight)
Operation	Methods	Uniform interface GET, PUT, DELETE, POST, PATCH, HEAD
HTTP	Transport layer	Application layer
Idempotency	Normally Not (Depend on implementation)	GET, DELETE and POST are idempotent, PUT is not
Use Scenario	Usually Enterprise Applications	Ad- hoc Scalable network
Caching	The semantics is not clear so requires some efforts	Clear semantics which enhances caching
Interface	Multiform	Uniform
Browser	More compatible	Some methods such as PUT, DELETE, and PATCH are not fully supported by some browsers.

The debate over using SOA or REST has been on-going for several years. It is an important architectural decision for new and existing projects. Alshahwan and Moessner compared these two web services: REST-based mobile web services and SOAP-based mobile web services on three aspects: 1) Message size and its corresponding response time; 2) Effect of concurrent requests on process time; and 3) Message size and its corresponding consumed memory size. From the result of these experiments, REST-based web services perform better for large size request messages or a large number of concurrent requests.

In summary, REST web services have superior scalability than SOAP web services. Since REST web service uses HTTP and HTTPS as its foundation, it is easier to modify a pre-existing web system. The lightweight of REST web service makes performance more efficient. SOA supports more transport and it is generally believed that SOA provides more reliable, and more secure services compared with REST. Choosing between SOA and REST web services should be dependent on business requirements.

3.2 Access Control

Web services are now becoming the dominant paradigm for e-business; thus, web service access control is getting more interest. There are four models for access control [2]:

1) Attribute-based access control (ABAC); access is granted based on attributes of the users;

2) Discretionary access control (DAC); object owners decide access policies;

3) Mandatory access control (MAC); access policies are determined by the system; and

4) Role-based access control (RBAC); users are assigned different roles and each role has its operation permissions.

Standard web access control uses RBAC. RBAC is performed through authorization and authentication. Authentication is the process that verifies that someone is who they claim to be.

Normally, users are prompted to enter usernames and passwords; or other technologies like scanning a fingerprint, face recognition etc. are involved. Authentication is used to find out what level of access the identified person has. For instance: in which group the user is in, or if the user has permission to access certain resources. An issue for authentication and authorization is that they lack the capability to provide access control for cross-domain web services and cannot provide dynamic control to adapt to business changes. Due to these reasons, some additional "access control" mechanisms are brought up.

To address the issue of access cross-domain services, Context based Dynamic Access Control Model for web service (CDACM), an RBAC extending based dynamic access control model for web service is proposed [28], where Shang et al use "global services" and "global users" concepts to refer to cross-domain composite web services and roles to call these services. "A service is composed of several operation on objects, and services are provided by various providers" [28]. Service is distinguished into global services and local services. "A global service consists of local services or global services from other providers" [28]. "Roles in this model are also distinguished into global and local roles. A global role consists of local roles and global roles from other providers." [10]. A global role is granted to call a global service which includes more local web services from different domains. When a user requests a global service, the global role is active and granted to the user. The "global role" is also suggested by Jeffrey Fischer, Rupak Majumdar [12]. They developed an algorithm that computes a global RBAC policy from RBAC policies of different applications. Web service interfaces in each domain are defined as well as each associated role. A global role is granted permission to one or more global services. For each web portal, local roles are mapped to global roles to satisfy the web interoperation.

Coetzee et al. [6] discuss a conversation process based on context awareness to control web services access. The conversation is a list of tasks to fulfill certain business processes. The specification for conversation is Web Services Conversation Language (WSCL). Conversation is essentially trust negotiation which goes back and forth between services requests and service providers.

Abdrahman [1] proposes Web Access Control using User Access Behavior (WACUAB). The system first prepares mineable data using users' logging history like: log date, log time, access URL and logging frequency; then analyzes this data, generates an access pattern and uses this pattern to enforce access control.

Bauer [4] designed a Proof-Carrying Authorization (PCA) web access control system to solve the interoperability issue [4]. In order not to touch any pre-existing infrastructures, a fact server and a proxy server are added in a standard web browser and web server architecture. The proxy server is an intermediary between a normal web browser and a PCA-enabled web server. The fact server holds the fact gathered from clients. The process of accessing a required URL is: 1) A User's requests to access a URL are generated as challenges and are sent to the PCA enabled web server by the proxy server; 2) The PCA enabled web server returns an unproven proposition to the clients; and 3) The client contacts the fact server to get a certificate asserting for the unproven proposition. The process of step 2 and 3 is called "iterative authorization". "Iterative authorization" continues until all the propositions are successfully proven or the process gives up due to not being able to providing the asserting. If the process succeeds, the user gets access to the requested URL. The whole process is transparent to users and there is some overhead which affects performance. To make the system efficient enough for feasible use, some approaches like caching and pre-guessing tactics are applied.

Trust comes naturally in terms of access control. Trust has played an important role in human interaction and cooperation for a long time. It is one entity's belief of whether another entity can provide certain services or not. Different web services have corresponding trust thresholds. Depth-Analysis authorization, usually trust policy; access control decision-making; and trust computing algorithms compose the trust-based access control of Web Service. More detail about trust model systems will be discussed in later sections.

Overall, current web access is controlled by using RBAC, extended RBAC, historical logging data, users' context information, negotiation between clients and servers and trust calculating.

3.3 Enterprise Service Bus (EBS)

Enterprise Service Bus (ESB) is middleware software that provides requests routing, message queuing and transforming, service orchestration, process monitoring, UDDI registry and security services functionality. The intention for ESB is to integrate and interoperate different applications in a complex enterprise computing environment. Currently, most enterprises run multiple applications in their software ecosystem; how to make these applications interoperate with each other and synchronize data becomes an issue. For example: At a university there are varieties of services providing by different organization. The services for each organization are developed with different computing languages, platforms, and databases. Exchanging data between these services is an issue. ESB can be used to exchange messages and route these messages from service providers to the service consumers.

The main functionalities for ESB are outlined below [19]:

1) *Routing:* services and messages need to be routed from consumers to providers and the vice versa;

2) *Transformation/queuing:* a message from one service in a particular format needs to be converted into another format so that it can be understood by other services. ESBs can also synchronize messages between different services and act as temporary storage repositories for messages;

3) *Service orchestration:* ESBs combine independent services together and expose the service as a logic unit;

4) *UDDI registry:* services are registered using WSDL in an ESB framework. Service requesters can find these services at design time and find the endpoint at the running time;

5) *Security:* ESBs provide identification and authentication services for entities;

6) *Management:* ESBs monitor business process, logging and auditing information.

Numerous companies offer ESB business integration solutions like IBM, Microsoft. There are also many open sources for EBS, such as Apache Camel, and JBoss Enterprise SOA platform. Each ESB has its own strength and pattern; therefore, customers need to understand both the vendors' technologies and their business requirements in order to choose the right ESB.

3.4 Mobile Device Security and Comfort

By 2011, there were about 5.9 billion mobile devices in the world [14]. The increasing capability and wireless connection is becoming a new threat to mobile security. Many solutions for mobile computing security and trust issues have been proposed and used. Some of the approaches from mobile devices perspective are:

1) Adding trust hardware for distributed mobile devices and components for mobile devices' OSs. The Root Trust Model was proposed to improve the trust between users and devices through a set of hardware (HW) and software (SW) mechanisms for

authenticated booting, platform integrity attestation and data access/operation controls [16]; and

2) Customizing IT security policies on mobile devices. Security policies can be mandatorily loaded when the mobile devices' OSs start up. The security policies are usually formulated by the management of organizations. As an example, security policies enforce mobile devices' users to use password authentication.

Another approach enhancing the transmission security is explained below:

1) The data transferred between servers and mobile devices is encrypted. For example, BlackBerry provides Advanced Encryption Standard (AES) and Triple Data Encryption Standard (Triple DES) encryption method; and

2) Support for HTTPS connection. For example BlackBerry supports two types of HTTPS connections: the first one is proxy mode where the security connection is between BlackBerry enterprise servers and BlackBerry application servers. The other connection is end-to-end mode where the data is transferred through Secure Sockets Layer (SSL) or Transport Layer Security (TLS) connection from mobile devices to application servers.

Due to mobile device features such as mobility and accessibility, mobile device usage becomes a consideration. Stephen Marsh [21] proposed the "device comfort" concept to describe the relationship between human beings and devices. Comfort is "a feeling of relief or encouragement …contented well-being … a satisfying or enjoyable experience" [21]. Marsh believes this definition explains device-owner relationships. Marsh also proposes a mechanism for mobile devices to make a value judgment based on reasoning about the following three specific components: user, location and task.

From the perspective of the "user", there are several trust level states between their mobile devices and themselves. These trust level phases are listed below as explained by Marsh [21].

1) *Imprinting:* an initial state; during this period, devices build a strong model of trust and behavior using users' identifiers;
2) *Nurturing:* in this phase, the trust between users and devices are reinforced;
3) *Growth:* when users use the devices properly, the trust grows;
4) *Repair:* if something goes wrong, the relationship between users and device needs to be fixed. For example, the trust level is getting lower when the user misbehaviors; and
5) *Use:* when the user needs to access sensitive information not only his/her credentials need to be provided but also the device comfort level.

The process of building device-owner relationships can go back and forth through these five phases described above.

From the "location" perspective, Marsh categorizes locations as comfort, discomfort, Tahrir, and social zones. The location affects the overall devices' comfort, when in a comfortable zone, such as, home or office, the devices' comfort level increases; on the other hand, when in a discomfort zone the comfort level reduces. There is a zone called Tahrir which is defined as discomfort but it is vital to open certain services; for example, despite uncomfortable events that have occurred in the Middle East, communication is important in this area and mobile device services remain even when the comfort level is low. Social zones are special cases too. Mobile devices detect other devices around and recognize they belong to social friends such as co-workers or some club members. Based on different social zones, different data or applications are accessible.

From the “task” perspective, if a task is a routine job and it is executed in a normal context, the device’s comfort level increases; if a task has never been done before, or the context value is not normal, the device’s comfort level decreases. Some tasks under certain context are proscribed or flagged, for instance driving & calling simultaneously, it is hard to know who is actually calling, the owner might be just a passenger but it certainly affect devices’ comfort level negatively.

To enhance mobile devices’ security, building a proper device-owner trust relationship is a key factor. Some factors like users’ behaviors and users’ context are discussed in the upcoming sections.

3.5 Mobile Context

“Context” is defined as any information that can be used to characterize the situation of an entity [8]. A situation can be a time, a location, a heading direction or a social context. These situations are observed in our daily lives. From a mobile device perspective, a mobile device‘s location, time and current network speed are examples of contexts.

Schilit [27] divided context into three categories: computing context, user context and physical context. Computing context can be current network type, bandwidth, mobile devices’ memory and storage. User context can be current location, a user profile, etc. Physical context can be current temperature, wind direction, etc.

One trend in mobile devices is that a number of built-in sensors have become standard. Cameras, GPS receivers, acceleration sensors and level sensors are now commonly built into smart phones and tablets. Thus, the mobile devices’ hardware meets the requirement for implementation of context aware applications.

Mobile context aware applications are now developing fast with the trend towards a highly mobile workforce. Context awareness was first envisioned by Mark Weiser [33] who uses storytelling style to convey his idea on how context aware computing makes our lives efficient and smoothly.

Dey et al. [8] categorize context-aware applications into three types:

"1) Presentation of information and services to a user

2) Automatic execution of a service

3) Tagging of context to information for later retrieval" [8].

The "presentation of information and services to a user" ensures that the application has the ability to detect the context information and present it to users. "Automatic execution of a service" refers to the ability to execute services based on the context information. "Tagging of context" is the ability to associate digital data with a certain context. For example, a virtual post introducing of a person pops up when this person's face shows.

With the growth of context-enabled applications, some researchers propose context provisioning systems which make building of context-aware applications more efficient. Some examples are: context toolkit [26] which is a context GUI widget; a context provisioning architecture which provides a platform for discovery and provisioning context information to the different entities [25]; middleware which allows for providing context information between consumers and providers [7].

Among all the context information, one of the most important factors is location. Location Based Service (LBS) technology is developing fast for mobile computing due to the mobile devices' nature. Mobile device users sometimes are not contented with just using static applications; they often prefer to get services and information based on their current need and the

surrounding situation. For example: mobile device users may like to know what is going on in the cities where they live. When traveling, mobile device users may want to get the latest traffic status and check the nearest restaurant. Location also provides information for mobile networking security concerns: how safe the current mobile device's network is; what type of the network is in use; what type of location they are at, such as the office or a coffee shop. Trust policies are formulated based on these security concerns. There are several technical ways to get current geographical location. These methods are used in different scenarios according to the accuracy and acquiring speed requirements, mobile devices, indoor or outdoor and available networks. A common method to get location is to use Global Positioning System (GPS) tracking. Most modern mobile devices have built in GPS receivers. GPS provides accurate location service and works well in an open wide area but not so well indoors or around skyscrapers. For indoors and some areas where the satellite signal is blocked, Wi-Fi positioning systems are used. This technical method detects the entire wireless routes around the mobile device and based on the gathered information calculates where the mobile device is located. In San Francisco, for example, mobile device users can get quite accurate results. The third method is using telecommunication networks. The mobile carriers can triangulate the position of the mobile device. The accuracy of this method depends on the density of network cell towers since mobile carriers use cell towers to detect the devices' locations. Usually this technique provides less accuracy than GPS and Wi-Fi positioning.

3.6 Cloud Computing

Mobile web services are usually deployed in a cloud computing environment. Cloud computing is a new emerging computer paradigm, it provides on-demand storage, application and computing service over network—usually the internet. Consumers use the cloud hosted

services without having any knowledge about the physical location of the service provider and technology infrastructure. Cloud computing can be private or public: public cloud computing sells services to everybody while private cloud computing only provide services to certain people.

There are three types of cloud computing based on their services known as [5]:

1. *Infrastructure as Services (IAAS):* This is hardware related services such as providing storage or virtual machine services, for instance, Amazon EC2.
2. *Platform as Service (PAAS).* This is defined as "delivery of a computing platform and solution stack as a service.", for instance: Google App Engine.
3. *Software as Services (SAAS).* This service is the most commonly use services. We use such services in our daily life, like: hotmail and Google mail are examples of SAAS.

Cloud computing provides a flexible business model for organizations, especially medium or small size businesses. Traditionally, businesses purchase hardware and software at one-time payment and hire people for hardware and software maintenance. Now they can purchase these services on demand; similar to purchasing utilities. Small and medium size corporations need scalability as their businesses grow. The hardware and software they invested before might not suitable for current situation. Cloud computing offers these businesses flexibility by simply allowing for upgrading the cloud computing services in order to satisfy growing business needs. Cloud computing is also suitable for massively distributed systems such as ad hoc mobile networks.

A main challenge for cloud computing is scalability: A program can continue running smoothly even when concurrent requests significantly increase.

Cloud computing is evaluated by the standards of security, availability, scalability and performance. A good cloud computing service should deliver consistent, efficient and reliable services to its clients with zero or low maintenance effort. With the growing cloud computing providers, it is difficult to choose the right vendor. In the state-of-the-art survey, Habib et al. [16] summarized some parameters that customers need to measure when choosing cloud computing vendors. These parameters are: "i) Service Level Agreement (SLA), ii) Compliance or accreditation or certification, iii) Portability feature, iv) Interoperability feature, v) Geographical location of the data center (Cloud), vi) Customer support facilities, vii) Performance test, viii) Deployment models(e.g., private, public, and hybrid clouds) ix) Federated identity management solution, x) Security measures, and xi) User recommendation, feedback and publicly available reviews." .

A more accurate evaluated mechanism is needed and a third party should be involved in the evaluation process.

3.7 Trust Model

With the exceptional growth of e-business, e-commerce and emerging enterprise technologies, building trust models has become one of the hottest research areas. It certainly leads the trustworthy computing system. General trust is regarded as one entity's belief about another entity in certain aspects under certain conditions. The essential of building a trust model is defining trust relationships and the mechanisms for calculating the trust values. Normally a trust value is established based on the interaction with other entities directly or indirectly, recommendation and context information, etc. A well-built trust system allows users to share information, do business and store sensitive data without worrying about security issues.

Some existing trust models are characterized in this section. Wang et al. [32] propose Bayesian Network trust Model in a peer-to-peer (P2P) network [32]. The foundation of this approach is the Bayesian rule. Basic Bayesian formula is:

P(C=T|A=T) =P(C=T, A=T)/P (A=T) …….. (3.1)

Where,

P (C=T|A=T) represents the probability of "C=T" given "A=T"

P (C=T, A=T) represents the probability of "C=T" and "A=T"

P (A=T) represents the probability of "A=T"

Wang et al. [32] use file-sharing systems in P2P environment as an example, and they discuss calculating the trust value from the transfer speed, file quality aspects for each peer. The overall satisfaction for a peer can be calculated as:

$\mathbf{S = W_{ds} * S_{ds} + W_{fq} * S_{fq}}$ **…… (3.2)**

Where S is overall satisfaction

W_{ds} is the weight of file download speed

S_{ds} is the satisfaction of file download speed

W_{fq} is the weight of file quality

S_{fq} is the satisfaction of file quality

Wang et al. [32] also propose a formula which calculates the recommendation values from other peers, as shows in equation 3.3:

$$R_{ij} = W_t * \frac{\sum_{l=1}^{k} t_{ij} * tr_{il}}{\sum_{l=1}^{k} tr_{il}} + w_s * \frac{\sum_{z=1}^{g} t_{zj}}{g} \quad \textbf{…….. (3.3)}$$

R_{ij} is the total recommendation value for the jth file provider that the ith agent gets.

k and g are the number of trustworthy references and the number of unknown references respectively.

tr_{il} is the trust that the ith user has in the lth trustworthy reference.

t_{lj} is the trust that the lth trustworthy reference has in j^{th} file provider.

t_{zj} is the trust that the zth unknown reference has in j^{th} file provider.

W_t and W_s are the weights to indicate how the user values the importance of the recommendation from trustworthy references and from unknown references.

Agents also update their trust values of other agents who provide recommendation, as equation 3.4 shows.

$$tr_{ij}^{t+1} = \alpha * tr_{ij}^{t} + (1-\alpha) * e_{\alpha} \quad \textbf{(3.4)}$$

where

tr_{ij}^{t+1} denotes the trust value that the ith agent has for J services for t+1 transaction.

α is learning rate.

e_{α} is new transaction evidence value which can be 1 or -1. If it is positive evidence then it is 1; otherwise -1.

Agents can also exchange their information with each other and update the trust values as equation 3.5 shows.

$$Tr_{ij}^{t+1} = \beta * tr_{ij}^{t} + (1-\beta) * e_{\beta} \quad \textbf{(3.5)}$$

where β is a learning rate and e_{β} is the new transaction evidence. β is less than α because this approach reflects agents' preferences based on their own interactions with the file providers more than the influence of any other agents' recommendations.

Lee et al. introduce a fuzzy trust model [20]. Lee et al use three trusts: situational trust, dispositional trust, general trust to calculate total trust value, "Situational trust (a.k.a.,

interpersonal trust) is the trust that an entity has for another entity in a specific situation. Dispositional trust (a.k.a., basic trust) is the dispositional tendency of an entity to trust other entities. General trust is the trust of an entity in another entity regardless of situation. The reputation is valuable information for estimating the trust of an entity. " [20]. When an entity starts to work, it initializes a dispositional trust value which is use as general trust value. It obtains the situational trust and the reputation from other entities as it interacts with them. General trust becomes situational trust when a sufficient number of interactions have been made for a given situation.

V. Varadharajan [29] defines a trust relationship as a tuple as shown below.

$\{P, Q, C, T, D, \tau, \nu, p, n\}$**…….. (3.6)**

where

P and *Q* are domains belonging to an entity set D

C is a class

T is trust type (direction trust, indirection trust, recommendation)

τ is time duration in which the trust relationship is considered valid

v is the trust value.

p is positive experience in term of trust relationship

n is negative experience in term of trust relationship

Varadharajan et al. [29] explain equation 6 as following: "…entity P trusts entity Q with regard to trust class C, trust type T , time duration τ, that security domains of P and Q are contained in D, and that ν holds the trust valuation" [29]. Varadharajan et al. [29] use subject logic to represent trust evaluation and trust evidence. There is a key component, trust management: trust management uses a combination of trust evaluation, trust evidence mapping

and trust comparison to make trust decisions and send them to a security management in a mobile open network.

Laurent Eschenauer et al. [10] compare trust establishment through the internet and mobile ad-hoc networks. Due to the nature of ad-hoc network, there is no long-term, stable evidence, "therefore, trust relations can be short-lived and the collection and evaluation of trust evidence becomes a recurrent and relatively frequent process". In summary, the protocols in ad-hoc mobile network should be

1) "Peer-to-peer, independent of a pre-established trust infrastructure";

2) "Short, fast and on-line" and;

3)"Flexible and support uncertain and incomplete trust evidence".

Wu [34] introduces three procedures in his trust model for a mobile device environment. The three phases are:

1) collection of observations;

2) filtering of observations

The procedure is designed based on Kalman filter theory. The basic Kalman filter employs the formula:

$S_i = S_{i-1} + \Omega_{i-1}$ …….. **(3.7)**

where

S_i is quality service at time i

S_{i-1} is quality service at time i-1

Ω_{i-1} is a random noise

The procedure uses recursive mathematical equations to update its current trust state; and

3) Predication of trust.

Entity A predicts entity B's trust value based on the last observation. The more frequently A contacts B, the more quickly the filter stabilizes the trust value and reduces the distance between the actual trust value and the predicated trust value. If the entity cannot endure high risk, the value Ω should be set at a high number. This value can also be an initial trust value if there is no previous history. A higher value of Ω means higher importance of freshly available information.

Jiang et al. [17] propose a trust system using interaction experiences, and recommendations from other peers. Their research focuses on mobile devices in a ubiquitous environment. Due to the features of mobile devices, a "hard to gain, easy to lose" trust policy is applied. The trust value is calculated by the following formula:

$$\mathrm{Va_j} = \max\{\frac{a_j * 2^{cn}}{Total_a} * \frac{SL_j}{SL_n}, -1\} \quad \textbf{(3.8)}$$

where

a_j is the security level for j^{th} action , it can be either positive number or negative number.

$Total_a$ is the total action number of a period.

SL_n is the highest security level in an applying domain

SL_j is a security level of target service which j^{th} action performed

cn is a counter number of continuous negative actions

When there is a continuous negative action, the trust value declines dramatically. The trust value increases slowly when the action turns into positive. It is more suitable for calculating the mobile devices' trust value since mobile devices have a high possibility of being used by other malicious people, so if any abnormal situations are found for a mobile device, it takes effort for this device to go back to normal status.

In an open network environment, especially in an ad-hoc mobile environment where there is no central administration, building a trust model is the key to let entities rely on the system where they can perform critical functions securely; process, store and communicate sensitive information safely. Generally speaking, building a trust model comprises setting an initial trust value, updating the trust value based on interactions, context and recommendation and risk considerations. A successful trust model should effectively prevent any attacks or malicious behaviors and adapt to dynamic environments.

3.8 Summary

The RESTful web service constraints derive by Roy Fielding [11] including: stateless communication, cacheable features and a uniform interface, make systems perform better for scalability and lightweight. Hence, the RESTful web service methodology suits our implementation goals. The "Device comfort" concept was proposed by Stephen Marsh [21] addresses the "lost or stolen devices" and "abuse of privileges" issues we raise in chapter 2. Trust relationships should be built and maintained between users and devices through user-location-tasks perspectives. Standard web services access control use authentication and authorization but cannot resolve the "cross-domain web services access" issue highlighted in chapter 2. For cross-domain interactions, many approaches have been proposed and implemented, and myriads factors are used such as user log history, users' context and so on. Among other factors, context information gets more attention for web access controls [28][6]. To build a proper relationship between users and mobile devices, context information like location and time also plays an important role [21]. Context information can be any physical, social or device's information [27]. Location is one of the most important factors for evaluating a trust value. Building a trust model for mobile devices, the previous trust value should be considered.

The previous trust value refers to the accumulated trust and reputation derived from the history transactions. The weight of the previous trust value depends on how risky the system can tolerate [34]. An initial value is set for each mobile device when it starts sending requests [20]. The evidence for establish trust relationship in mobile ad-hoc networks should be independent, fast to retrieve and flexible [29]. The formula to calculating mobile devices should be "hard to gain, easy to lose" to improve the mobile security [17]. Trust policy should be flexible and easy to maintain to adopt the dynamically changed business.

The list of papers reviewed is presented in Table 3-2 below.

Table 3-2 Issues/Goals and Solutions Found from Literature Review

System scalability & lightweight	Create RESTful web services on cloud servers Reference [11]
Lost or stolen devices/ abuse of privileges	Building trust relationship between users and devices based on interaction history and context information; adding a trust module for each web service. Reference [28] [21].
Cross-domain web access	Context information is used for dynamic access control. Reference [28] [6]. User behavior and historical history are used to control web services Reference [1].
Calculating trust value for mobile devices	The trust value is calculated by previous trust value and recently observation of mobile devices' transaction. Reference [32]. A trust threshold is used when make trust decision. Reference [10]. An initial value is set as a general trust value for an entity when the entity starts work. Reference [29]. The protocol for establish trust in MANET should be fast, independent, and flexible. Reference [10]. The formula should be "hard to gain, easy to lose". The continuous negative interaction number should be used in trust formula. Reference [17].

However, since this implementation handles web services from multiple domains, there are still some open issues:

- How to distribute the trust models?
- How much overhead for each transaction is introduced due to the additional trust component?

Not much has been done in the area of building a trust module for mobile devices' requests and cross-domain interactions. In this proposal, I propose and implement a distributed trust module which calculates requests trust value based on context value and trust credit while exploring approaches to achieve efficient performance and lightweight transactions.

CHAPTER 4
DESIGN AND ARCHITECTURE

4.1 Overview

The goal of this research is to create a trust model and integrate it with mobile devices web services.

4.1.1 The Trust Model Analysis

In Chapter 2, three scenarios were used as examples for the issues of using mobile devices and web services.

- Scenario 1: Lost or stolen mobile devices can be used by malicious people to hack web services, including legacy applications, thus causing profit loss and data leakage.
- Scenario 2: Some mobile device users abuse the privileges which are granted to them to access the web services. For example, they retrieve sensitive information in public or update data when they are drunk.
- Scenario 3: There are unforeseen interactions from the requests sent by other domains. Even a well known domain can be hacked and become untrustworthy, so interacting with these domains can be risky.

Figure 4-1 shows a basic structure of the implemented system. There can be many other kinds of system structures. An example of alternative structure is a proxy server connecting two web servers as figure 4-2 shows.

In figure 4-1, the mobile client sends a request to server 1 or its replication node server 1', depending on the trust setting, this request can be sent to server 2 (or replication node server 2') and server 3 (or replication node server 3') and so on. (For the

sake of simplicity, replication server nodes are not mentioned in the following sections). Server 1 questions the request sent by the mobile device; Server 2 questions the request sent by server 1 and so on. The trust modules under each server answer the questions by calculating the trust value of each request.

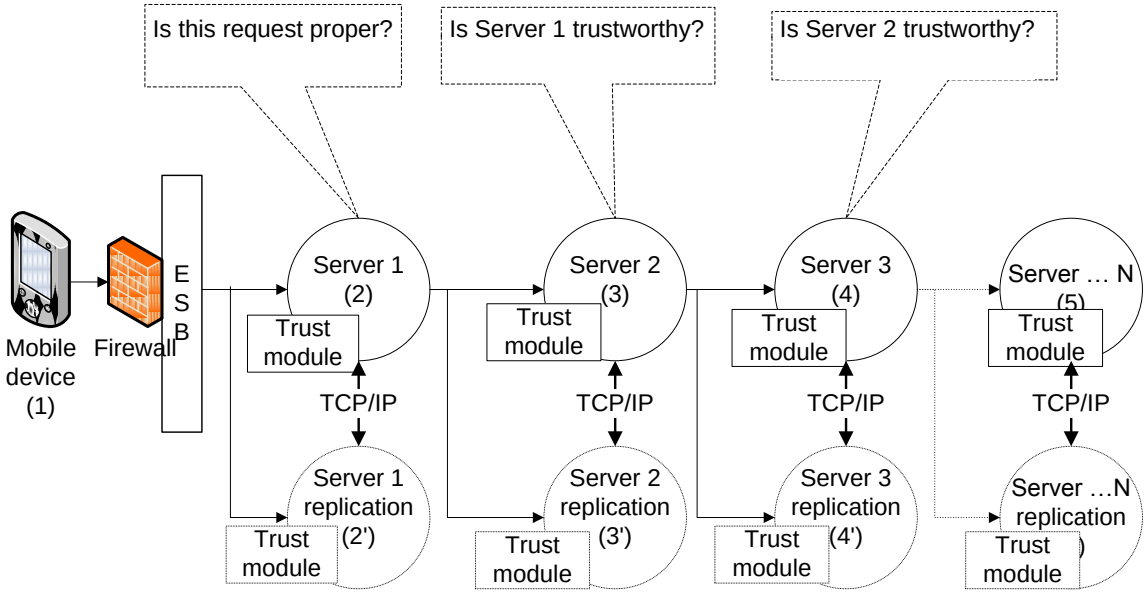

Figure 4-1 the proposed system structure

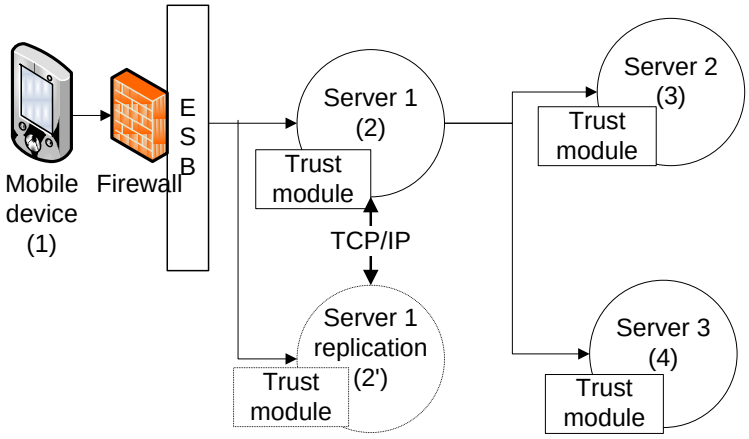

Figure 4-2 an alternative system structure

Server 1 checks if the context value, such as location and time, meets the trust policies, the operation pattern matches the mobile user's pattern, and the mobile device's

previous transactions. If the request is sent by a malicious person who pretends to be the mobile user, the possibility of operation pattern matching is low and the other context factors might not be allowed by the trust policies, thus the request is most likely rejected. On the other hand, if a legitimate user sends a request at an improper time and location, the trust module catches it and either rejects the request or sends a warning to the mobile user. If the request needs information from other server(s), or in another words, it is a cross-domain request, then it is forwarded to other sever(s), in this case, it is server 2. Server 2 calculates the trust value of requests coming from server 1; server 3 calculates the trust value of requests coming from server 2 and so on. If any domain is hacked, the trust value of this domain is adjusted accordingly on other domains; thus, the whole risk of unforeseen interactions from the requests sent by other domains is mitigated.

In figure 4-2, besides check the trust value of coming requests same as figure 4-1, server 1 sends requests to both server 2 and server 3 and combine the responses from the two servers together, and send the merged response to the mobile device.

The trust module is built as a decentralized attachable module for several reasons. First, each organization has its own business rules and security requirements, so each organization's web site correspondingly has their own trust policies. Second, since mobile devices' services are regarded as rapidly changing, a decentralized system is more flexible to adapt these changes. Except for the three goals mentioned in chapter 2, there are several other considerations about the system's features. The trust module is an independent module with its own database and it should be easily called by other web services; adding the trust module with an existing web site should be smoothly. In order to exchange information about the reputation of other domains, the trust module's data

should be exchangeable. To make the system more robust, one or more replication nodes are needed and database backup should be simple and easy to carry on as a routine work.

Figure 4-3 illustrates how the trust module works in server 1. It analyses the context value and the mobile device's history transactions, and then calculates the trust value based on the analyzed result. The formula for calculating trust value will be discussed in later section.

Figure 4-4 illustrates how the trust module works in server 2. After the process of authorize and authenticate, the trust module parses the HTTP header and gets the trust value, it then recalculates the trust value of the incoming request based on how much the current server trusts the requesting server.

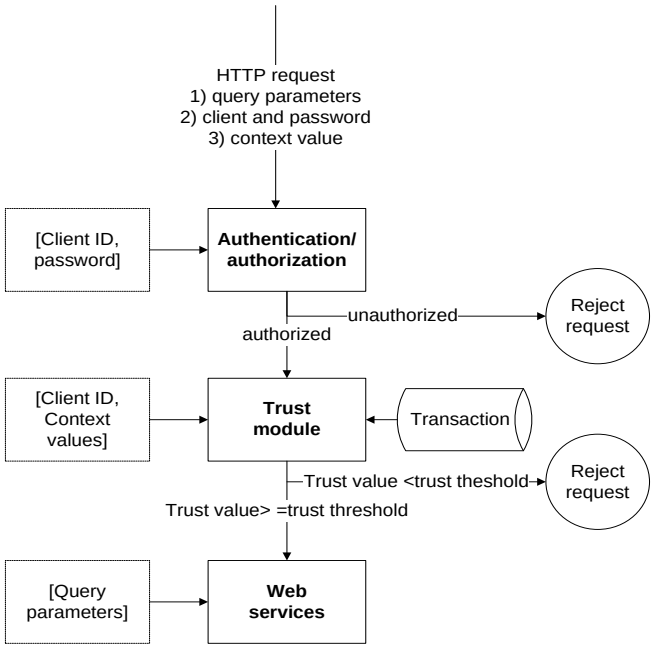

Figure 4-3 calculating trust value for mobile devices

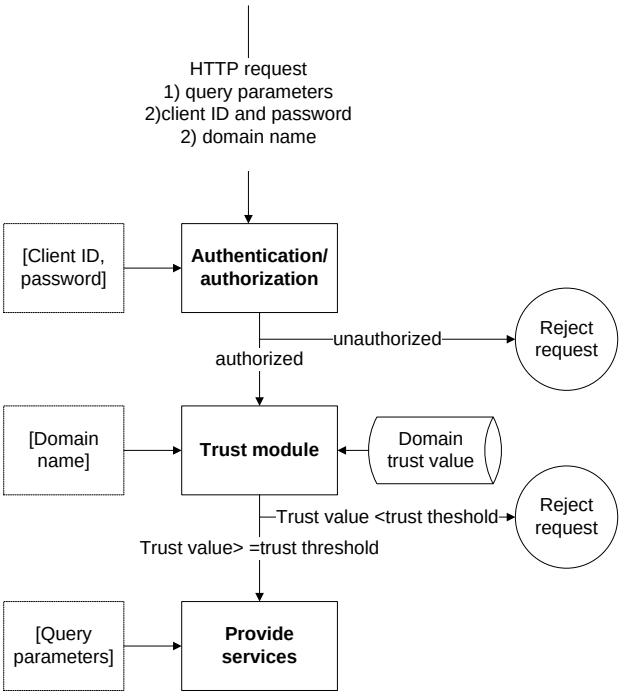

Figure 4-4 calculating trust value for other servers

There can be three results for handling HTTP requests depending on the trust value and the trust setting. The request can be rejected if it is not trustworthy. For the trustworthy requests, the requests are severed by the current server if the current server can provide the services. Otherwise the requests are forwarded to the proper server(s) if the current server finds the server(s) that provide the services.

The trust module's functionalities and features are summarized in table 4-1.

Table 4-1 The Trust Model's functionalities

1. Set constrains for web requests based on business needs
2. Keep HTTP transactions, the trust values of history transactions are considered when calculated trust value
3. Generate and update trust value for each domain
4. Attach/unattached by other web services effortless
5. Replicate/migrate trust model smoothly
6. Exchange data, such as transactions, trust policies and reputation for other domains

4.2 Architecture

The system architecture is explained through a physical perspective and a logical perspective.

4.2.1 Physical Architecture

The proposed system is divided into two components from the physical architecture perspective: the mobile clients and the servers as figure 4-5 shows.

- **Mobile Clients**

The mobile clients initialize requests and send them to the server(s), and represent the response from the server(s) to users.

- **Servers**

Servers analyze requests and calculate requests' trust values. They provide direct services to the mobile requesters as well as forwarding some requests to other web servers. After the requests going through the firewall and the enterprise server bus (ESB) which normally has authorization and authentication process for users to identify themselves, they are calculated by the trust module. The requests are rejected if the trust values are lower than the trust threshold which is initially set in the server. Depending on whether the servers provide direct services or not they are divided into proxy servers and web servers. The proxy server and the web server are illustrated into more detail in the logic perspective below.

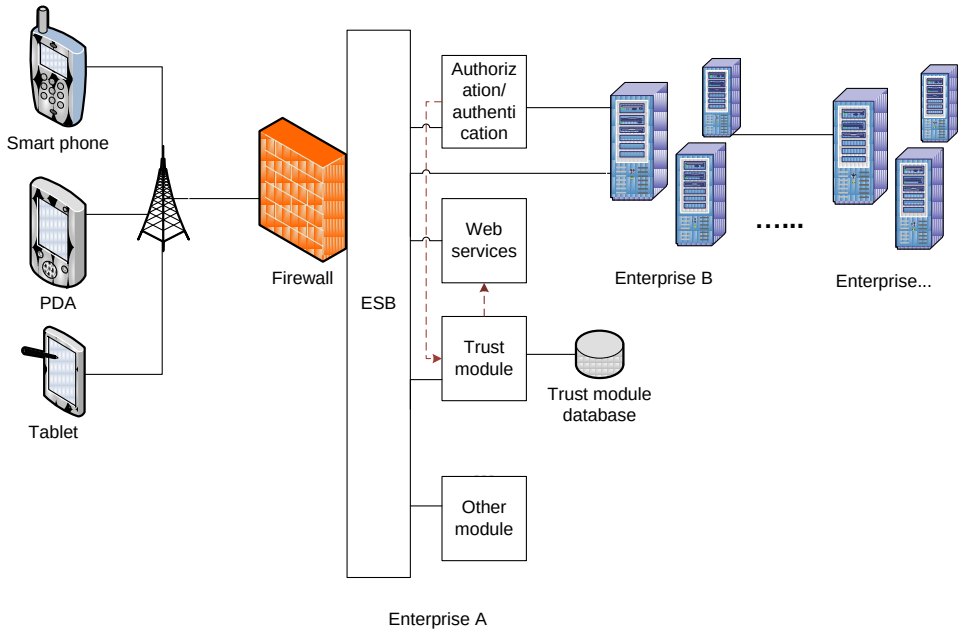

Figure 4-5 system physical architecture

4.2.2 Logical Architecture

From the logical perspective, the architectural structure is divided into three components based on functionalities: the mobile devices, the proxy servers, and the web servers.

- **Mobile Devices**

This component which can be smart phones, tablets, and PDAs as shown in Figure 4.5, is the same as the mobile devices component in physical aspect.

- **Proxy Servers**

The proxy servers do not provide direct services. They analyze the requests' context information according to the trust policies and request types, and calculate the requests' trust value. Based on the trust value and the trust threshold for this type of request, the proxy servers either forward the request to other servers, either web servers or other proxy servers, or reject the request.

There can be more than one proxy server in a request route: a proxy server can forward a request to another proxy server according to the request type and route settings. There is a table "route" in each proxy server which indicates the next forward server. After retrieving the next server, the proxy server acts as an HTTP client and sends this request to the next server.

- **Web Servers**

The web servers provide direct services for the mobile clients. When the web servers receive requests either from proxy servers or directly from mobile devices, they calculate the requests' trust values and decide to provide the requested service(s) or reject them based on the result.

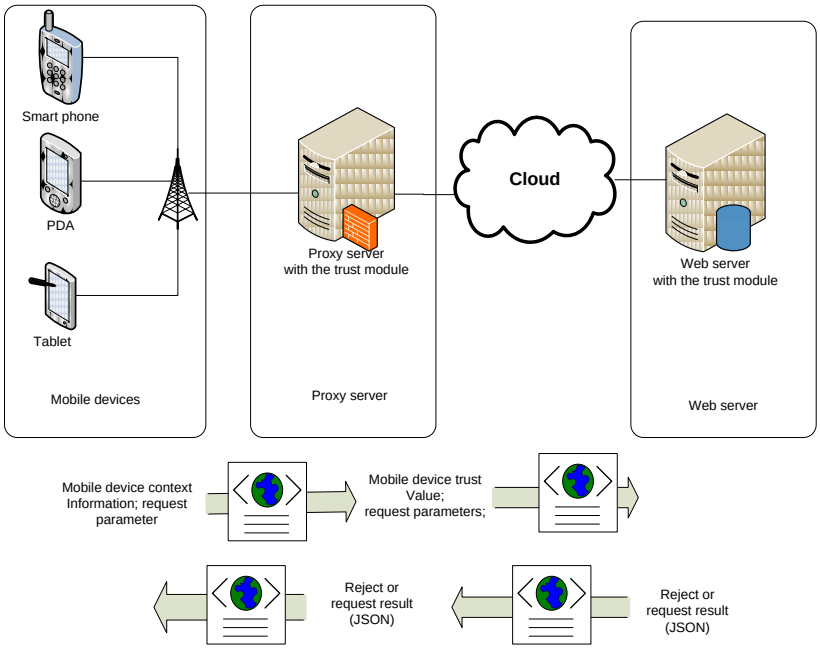

Figure 4-6 logical architecture

4.2.3. Replication Trust Module Nodes

In order to build a robust and efficient trust model, a replication node for the trust module along with the Mensia database is added in the system. Mnesia is chosen as a

NOSQL DBMS in the system since it is fault-tolerant and distributed. It naturally provides mechanisms for building a replication node. A replication node can improve the throughput and act as a backup server if the other nodes fail.

As Figure 4-7 shows, two trust modules are connected with other modules under the enterprise web services environment. ESB routes the requests to different trust modules and balance the workload between different trust modules. A database scheme and tables are created identically in the two modules and transaction data are also written in the two nodes simultaneously using the Mnesia mechanism to support distribution.

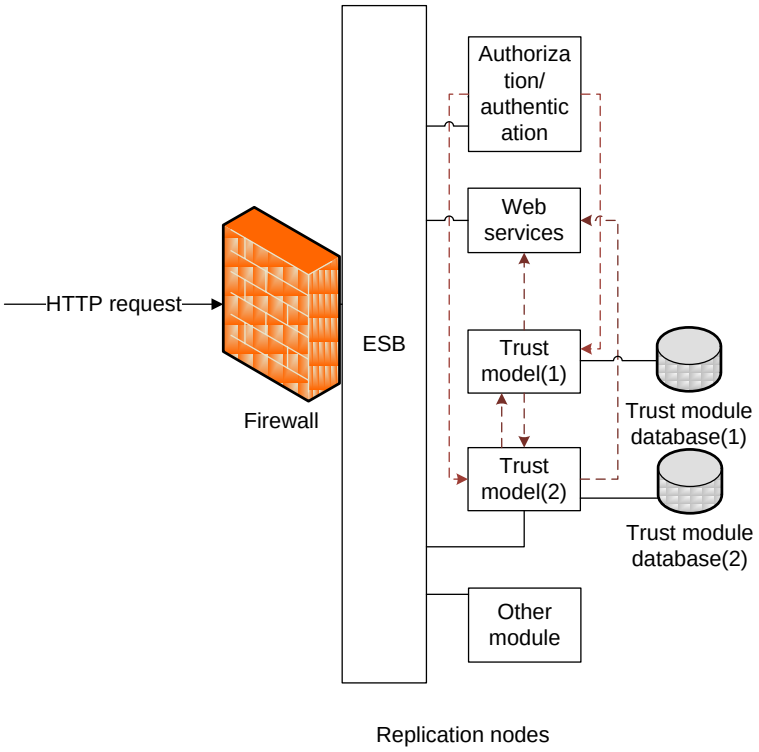

Figure 4-7 replication trust module nodes

4.2 Data Format & Flow

The mobile devices (HTTP clients) initiate the task by sending requests to the proxy servers or the web servers. The content of the HTTP requests contains query parameters and context information. The servers calculate the trust values of the incoming requests and either pass them to the next server(s) or provide the services. If the servers forward these requests to other servers, they act as the proxy servers; if the servers directly provide the services, they act as the web servers.

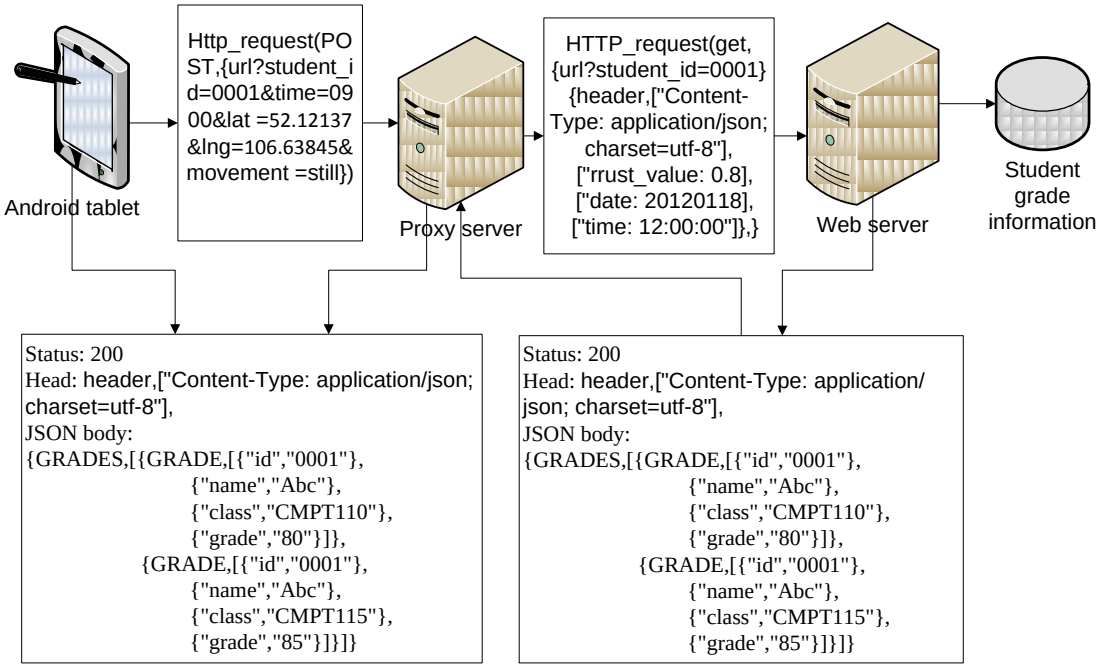

Figure 4-8 data format and flow

After receiving requests, the proxy servers recalculate the trust value, and check their route settings and send the request to the next server along with the new trust value and query parameters.

The web servers retrieve the records from the database and generate JSON (JavaScript Object Notation) objects after receiving the requests, and send these JSON objects back to the last requesting HTTP clients. If the last HTTP client is one of the proxy servers, it

passes the JSON objects to its HTTP client and so on until the JSON objects are delivered to the mobile devices. If the trust value of a request is less than a trust threshold in any servers, the request is rejected and error information is sent back to the mobile device.

Proxy servers also can integrate multiple responses together if a request needs more information which is distributed in different domains.

Figure 4-9 illustrates data exchange between the mobile device, the proxy server 1, the proxy server 2, and the web server.

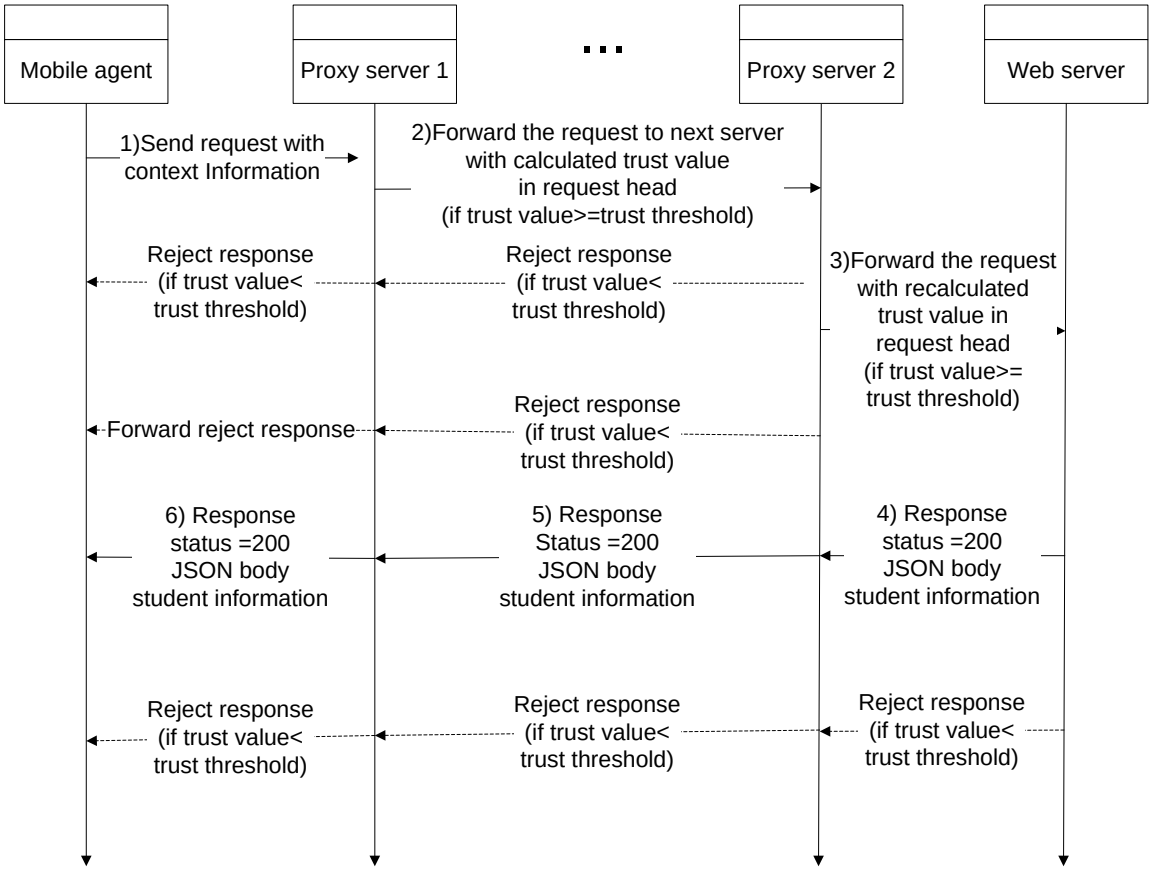

Figure 4-9 entities interaction sequence

4.3 System Functionalities

The system has the following functionalities:

- **Invoking web service from mobile devices:** An Android tablet is used as a mobile client in the implementation. When the mobile clients send requests, the context information, such as location, time, device position and device moving speed are also sent to the proxy servers or web servers along with other query parameters. This context information is used to calculate the trust value.
- **Analyzing mobile devices' context information and calculating trust value:** Prior to calculating requests' trust value for mobile devices, a set of trust policies needs to be defined. The trust policies are stored in the "trust_policy" table. For the "location" context as an example, the records in table "trust_ policy" are shown in table 4-1:

Table 4-1 trust policy for location

	Keyword	Policy Name	Criteria	Trust Value
1	Location	Trust value for location in U of S campus	1;lat:>=52.12137 and <=52.14271;lng:>=-106.63845 and <=-106.62197	0.3
2	Location	Trust value for location in Saskatoon	2;lat:>=52.08657 and <=52.18593;lng:>=-106.72565 and <=-106.55142	0.1
3	Location	Trust value for location in Canada	3;lat:>=46.52863 and <=69.83962;lng:>=-141.15234 and <=-52.82226	0.0

Mobile devices' context values reveal the devices' situation in many aspects; for example, the temperature can indicate outdoor or indoor; a moving speed can shows whether the users is driving, or waling or being still; Bluetooth connects can suggest whom the users are with. With more context values, a more accurate situation can be determined. More context values will be considered in the future work.

Different approaches to calculate mobile devices' trust values depend on the business requirements. Two formulas are proposed for calculating mobile devices' trust values in this research.

When the proxy server and the web servers receive the HTTP requests along with the mobile devices' context information, they look up each corresponding trust policy. For instance, if the location is not at the University of Saskatchewan campus but within the city of Saskatoon, the trust value is 0.1; if the location is not in Saskatoon but within Canada, the trust value is 0.0. Trust values for each context category are summed up as the request's current trust value.

$$T_{current} = \sum_{i=1}^{n} V_i \quad \textbf{(4.1)}$$

Where

$T_{current}$ is current request's trust value

V_i is the context value of i category

To apply the rule specified for mobile devices in Chapter 3 which is "hard to gain, easy to lost", the total trust value of mobile devices is calculated as follows:

$$T_{total} = \alpha T_i + \beta * min((T_{i-1} - T_h) ,0) + \lambda * min((T_{i-2} - T_h) ,0) \quad \textbf{(4.2)}$$

where

T_{total} is the total trust value

T_i is the current transaction trust value

T_{i-1} is the last transaction trust value

T_{i-2} is the second last transaction trust value

T_h is the trust threshold

α, β and λ are weight factors range from 0-1

The equation 4.2 is used as formula 1 in the experiments.

If trust values of the last two previous transactions are less than the trust threshold, they are involved when calculating the total trust value. The previous trust values are recorded in mobile devices registry table.

If previous transactions are not important for business requirement, then context values is enough for calculating the mobile devices' trust values as follows:

$T_{total} = \alpha T_i$ **…….. (4.3)**

where

T_{total} is the total trust value

T_i is the current transaction trust value

α is weight factors range from 0-1

The equation 4.3 is used as formula 2 in the experiments.

The request's current trust value is the summary of each context category as the equation 4.1 shows. The formula 2 is much simpler than the formula 1, and it shows better scalability and less overhead which are discussed in chapter 6.

- **Calculate web requests' trust value:** A proxy server or a web server calculates the trust value sent by other proxy servers. In each server node, there is a table called "domain_trust_mapping" which specifies how the current server trusts other servers. When the proxy servers forward the requests from the mobile devices to other servers, either proxy servers or web servers, they also forward the recalculated trust values. The formula listed in equation 4.4 is employed to calculate the current trust value:

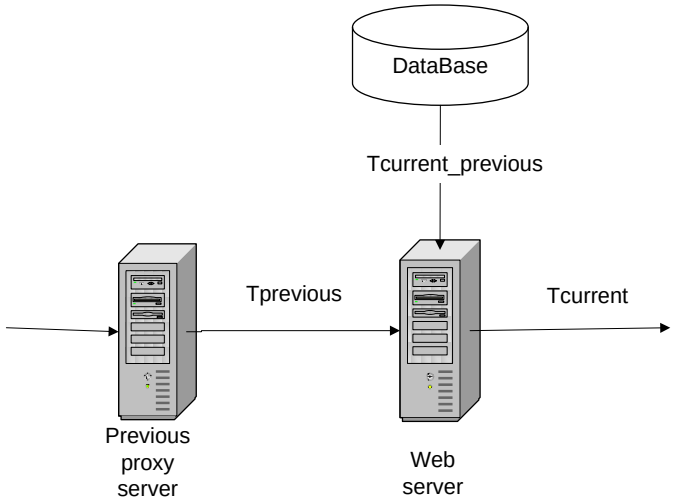

Figure 4-10 calculating trust

$T_{current} = T_{previous} * T_{current\text{-}previous}$…….. **(4.4)**

where

$T_{current}$: the trust value for the coming request calculated by the current server

$T_{previous:}$ the trust value for the coming request calculated by the the last server

$T_{current\text{-}previous}$: how much the web server trusts the proxy server. The value ranges from 0-1

Hence, the new trust value is the incoming trust value multiplied by how the current server trusts the client.

- **Forward web requests to the proper web servers; combine multiple web services:** The HTTP responses from different domains can be combined together to complete a business process. A "route" table is defined in all servers for each type of requests. For example: when request type *A* needs to forward a composite request to web server *B* and web server *C*, we merge the responses from server *B* and server *C* together and send the combined information to the mobile device.
- **Replicate data between two or more trust modules nodes:** To improve throughput and enhance the system's fault tolerance, multiple trust module nodes

are set in the system. The transaction data of the requests is replicated to all other nodes. When one node fails, other nodes can continue work without affecting business.

- **Maintain trust policies interface:** Trust policies need changing according to business requirements. In order to build a more flexible and automatic system, a web form is created to maintain trust policies.

4.4 Design

According to the physical components, system design is divided into mobile device design and servers design.

4.4.1 Mobile Device Design

The mobile devices act as HTTP clients in the system. In additional to send query parameters, they also provide context information. The functionalities of the mobile devices are detecting context information and sending requests to proxy servers and web servers.

Motorola MZ604 is used as the mobile device in this implementation. It runs on android API 3.2. Android is a software stack which includes:

- Applications like phone, web browsers and so on;
- Application framework;
- Libraries. Developers can access these libraries though the Android application framework; and
- Operating system. Android relies on Linux operating system.

Android 3.2 supports mobile tablets also. It provides zoom capabilities and supports extended screens.

There are normally two approaches for mobile devices' development. The first approach is the pure native application most likely using Java or C# and the second approach is the embedded browser design which mostly involves the use of HTML, JavaScript and CSS. Andy Wang [30] compares the two approaches as summarized in table 4-2.

Table 4-2 mobile native application vs. pure embedded browser application

	Native application	Pure embedded browser application
Pro	Performance (compiled code) Full access to native API Easy to test and debug Rich GUI features	Platform independent Less specialty required Easy to maintain and upgrade
Con	Platform dependent Maintenance and upgrade cost	Browser compatibility Performance (interpreter) Browser limitations No access to native API

The hybrid implementation is adopted in our system which involves the combination of embedded browser and the native application. The embedded browser approach is used for sending web requests and the native approach is used for detecting the device's context information via built-in sensors. The mix of two approaches makes development process and maintenance work relatively easy, also it takes advantage of the rich features of the Android SDK.

Table 4-3 shows the tools we use for this implementation.

Table 4-3 mobile device implementation tools

Mobile Devices	Devices Operation System	Programmer Language
Android tablet	Android	Java; Android SDK; HTML,

e.g. Motorola MZ604		JavaScript; Google map JavaScript API

As we recall the issues identified in Chapter 2, the system is going to build a trust model by using the mobile devices' context information. The tasks of mobile devices are initializing requests along with context information and representing the responses from the server. According to the tasks, there are three main Java classes and three activities.

The three major Java classes are "ThesisActivity", "GetContext", and "UseSimulation". There are some other Java classes but they only provide facility functions.

"ThesisActivity": This is the main Java class in this implementation. It provides the entry interface. Users can choose how they get the context value: through the mobile device's hardware or through simulation. Depending on the user's choice, different Java classes are called to get context values. After getting context values, a request is initialized and sent to the server, and then it waits for the server's response and represents the result to users. A widget called "webView" is used in this class which is an extended view class that allows developers to display HTML pages on the mobile embedded browser. The HTML page provides an interface for end user to enter query parameters and send requests to web servers along with context information.

"GetContext": The function of this class is getting context values such as current location and speed from the devices' hardware. It calls the Android *"Android.location"* SDK package. There are three location providers which are GPS, Wi-Fi and telecommunication network. The *"Android.location"* package provides the function to choose the best location provider according to requested criteria. *"accuracy_high"* is

chose as the criteria to get location provider. The speedy can also be retrieved from *"Location"* class from location provider.

"UseSimulation": The function of this Java class provides interface for users to enter simulated context information if use choose simulated context.

The three activities are "CreateMapActivity","UseSimulation" and "ThesisActivity". An activity is a user interaction to complete certain task in android development.

"ThesisActivity": The tasks for "ThesisActivity" are 1) collecting context information and request parameters; 2) Sending the HTTP requests to the web servers and the proxy servers and 3) presenting the result from the web servers or proxy servers.

"SimulationActivity": "SimulationActivity" is created for experiment purposes. This activity feeds the context values users enter into the "webView" in "ThesisActivity" in order to simulate different scenarios to test the trust model. After starting the "UseSimulation" activity, users can enter simulated location, speed and time. These values will be returned to the "TrustActivity" activity.

"CreateMapActivity": "CreateMapActivity" feeds location information to "SimulationActivity". It shows the map when activated and lets users choose the location they like to use for the simulation. This location value is sent to "SimulationActivity".

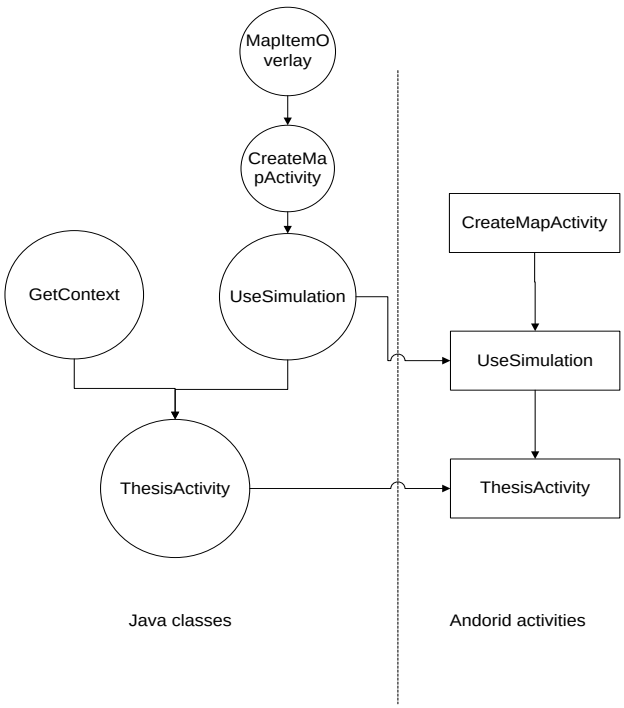

Figure 4-11 mobile devices modules

4.4.2 Proxy Server and Web Server Design

In Chapter 2, the requirements and features of the system were brought up: the system should be flexibility, scalability, and lightweight. To meet the requirements, Yaws 1.9 and Mnesia database have been chosen as tools to implement the web server and the proxy server. Table 4-4 shows the tools used in this implementation.

Table 4-4 proxy server/web server implementation tools

Operating System	Windows 7
Web Server	Yaws 1.9
Program Language	Erlang OTP
Database	Mnesia
Present Page	HML; Yaws
Interchange Data	JSON
Style Sheet	Cascading Style Sheet
HTML Script Language	JavaScript

The proxy servers and the web servers are different in functionality in respond to users' requests, but they have same data structures and source files.

There are several approaches to implement dynamic web contents for YAWS 1.9, such as EHTML tag, Web Sockets, embedded mode, appmods (application modules) etc. In this implementation, "appmods" is chosen since it lets programmers take control of URL paths which is suitable for dynamic web contents. Table 4-5 is the list of URLs and their corresponding modules. The map between URLs and Erlang modules needs specified in configure file too. The following is a snippet of the configuration file.

```
<server semeru.usask.ca>
        port = 8080
        listen = 0.0.0.0
        docroot = "C:\Program Files (x86)\Yaws-1.91\thesis/www"
        appmods = </mobile_services,mobile_services,
       /proxy_services,porxy_services>
</server>
```

This snippet defines: 1) listening IP address, "0.0.0.0" menas listens all the IP address; 2) listening port, 8080 in this case; and 3) the server's root directory and web services.

Table 4-5 gives an example of URLs and their web services.

Table 4-5 maps of URL and Erlang modules

URL	Erlang Module/Yaws Pages
http://semeru.usask.ca:8080/	Home page
http://semeru.usask.ca:8080/mobile_services	Web services for mobile devices
http://semeru.usask.ca:8080/proxy_services	Web services for proxy servers
http://semeru.usask.ca:8080/trust_policy	Trust policy editing interface

The main modules in YAWS1.9 that we use are discussed below.

"mobile_services:" This is the one of the main services for handling requests from mobile devices. It monitors and handles the mobile devices' requests, parses mobile

devices' parameters and context information, calls the *"calculating_trust"* function to get the requests' trust values, and compares these values with the trust threshold. If the current trust value is higher than trust threshold value, it either retrieves information the mobile devices ask or transfers this request to other server(s) depending on the setting which is defined in table *"route"*; otherwise it rejects the request with "trust is not enough" error message.

"proxy_services": This is the main module for handling requests from the proxy servers. It monitors and handles proxy servers' requests, parses proxy servers' requests to get the trust values and query parameters, recalculates the trust value for this request based on how much the current server trusts the proxy server, and compares this value with the trust threshold value. If the current trust value is higher than the trust threshold value, it either retrieves information the proxy server asks for or transfers this request to other servers depending on the setting which is defined in the table *"route"*.

"calculating_trust": This module analyzes the context information, calculates each context factor's trust value based on the trust policies. The content of trust policies includes context keywords such as location, time, criteria for the context value and corresponding trust values. The process of calculating trust values is: 1) search the trust policies table to get all records with the specific keyword ordered by sequence number; 2) traverse the set of records and check if the criteria matches the given context value. If matched, get the trust values; if no matched record is found, return 0 which means no trust value.

"student_information": This module maintains student grade information. It provides "update" and "query" functions. The "update" function can add or delete student

grade records. The “query” function returns a set of student grade records based on query parameters.

4.5 Data Model

In this system, the proxy server and web servers are required to provide constant and high performance services. Mnesia is a distributed DBMS written in Erlang which It features fast data searching and runtime DBMS configuration, so it is chose as database management system (DBMS) in this system.

Mnesia data can be either disk-based or memory based and it can be replicated to different Erlang nodes.

The trust module’s data includes mobile devices’ registry information, trust policy, request route, and trust value for each domain; other data include the experiment data which is student grade information in this implementation.

Mensia’s data is organized as a schema and tables. The schema and tables are created in multiple Erlang nodes. An Erlang node refers to an executing Erlang runtime system which is assigned a name. Nodes are connected through TCP/IP connections. Authentication between Erlang nodes is processed by comparing Erlang cookies. Erlang cookie is an Erang atom which is assigned a name when the node starts. When one node tries to connect to other nodes, it checks the other nodes’ cookies. If they don’t match then the connection fails. “Since Mnesia is running on top of distributed Erlang the implementation is greatly simplifed. In a distributed application there are separate Erlang nodes running on different machines. Erlang takes care of the communication between processes possibly on separate nodes transparently. Processes and nodes can easily be started, supervised and stopped by processes on other nodes. This makes lots of

communication implementation problems disappear for Mnesia as well as for applications" [23].

To create schemes and tables, a node list of parameters is provided in creating schema and table functions. The "node list" is an array of Erlang server nodes. The schemes and tables are created in each node after calling the functions. The transaction data is written in each node as well. Data can be either in memory or disk. In this implementation, data is recorded in the disk. The updating data can be performed asynchronously and synchronously. Updating synchronously means the transaction function waits all the nodes successfully being updated and then continues. Performing asynchronously means the transaction function waits for only one node successfully being updated but not all other nodes. In the synced approach, there are transaction operations and dirty operations. Transaction operation ensures the data's consistency and isolation while losing some performance. [9] "Dirty operations are short cuts which bypass much of the processing and increase the speed of the transaction." [9]. Table 4-6 lists different features of the approach.

Table 4-6 transaction functions comparison

Function	sync_transaction	async_dirty	sync_dirty	Ets
Synchronous	Yes	No	No	No
Data Consistent	Yes	No	No	No
Replication	Yes	Yes	Yes	No
Performance	Slow	Fast	Fast	Very fast
Usage Scenario	For application need to make sure all nodes are updated	No need for 100% consistence for replication	Be sure the remote update is completed before any process is spawned	Only for local use; no replication

In this system, data consistency is ensured to make sure the system is reliable, so the synced transaction approach is chosen to spread out data.

There are three types of Mnesia tables 1) Set. 2) Ordered Set. 3) Bag. A set table has a unique key, if a new record is inserted with same key as an existing item, the old item is overwritten. Ordered sets store data by the unique key, they perform efficient for searching. Bag type table can holds several records with the same key.

More details such table structure and of transactions are provided in Chapter 5.

CHAPTER 5
IMPLEMENTATION

More details about the design and coding for the mobile clients, servers and data model are discussed in this chapter.

An Android tablet is used as the mobile client since this type of mobile devices is commonly used now due to their capability and compatibility. YAWS 1.9 web server is chosen as the programming platform for the proxy servers and the web servers because of its light weight and scalability.

5.1 Android Tablet Implementation

The development tool, development environment, the source files structure and the mobile client's design, including three activities, are discussed in this section.

5.1.1 Set development environment

5.1.1.1 Mobile device information

Device Model: Motorola, MZ604 Android tablet

OS: Android Version#: 3.2.

Internal Memory: 29475MB

RAM: 719MB.

Total Storage: 30GB.

5.1.1.2 Mobile device setting

Wireless connection setting: Wi-Fi is used to connect to the internet or any available networks, set mobile device's security configuration and access point for the connections.

Since mobile context information is crucial in this research, the location providers should be enabled. In “Setting” application, under the “Location & security” category, “Use wireless network”, “Use GPS satellites” and “Use Location for Google search” items should be enabled.

In order to install the mobile application into the mobile devices, set the “unknown source” under the “Applications” enabled.

5.1.1.3 Mobile devices’ development tools:

Eclipse IDE with Android Development Tools (ADT) plug in is the typical development environment for Android devices. For the particular MZ604 Motorola device, Android SDK 3.2 and Google API3.2 are used.

5.1.1.4 Set Google Map API key

For the convenience of the experiments, “MapView” class that integrates Google map is called. The map shows the simulated locations. Since MapView gives the access to Google Map data, registration with Google Developer Community is required to get the services.

There are two steps required in the Maps API key Registering process.

1) Registering the Message Digest Algorithm (MD5) fingerprint of the certificate.
2) To sign up for the Android Maps API, under *“http://code.google.com/android/maps-api-signup.html”*, enter the MD5 fingerprint and generate the API key.

The API key is used in MapActivity activity layout file, as the following XML file shows:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/mainlayout"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">

<com.google.android.maps.MapView
    android:id="@+id/mapview"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:apiKey="0_YnbkWeTacthupcU7xOaaHdDaGEOrI_7wckyKQ"
    android:clickable="true" />

</RelativeLayout>
```

5.1.2 Android Implementation Files

5.1.2.1 Mainfest.xml

Every Android application must have a mainfest.xml file. This file defines application package name, the Android SDK API version, the application permissions and application components.

Based on the requirements of the system, permissions such as internet and location access are needed. The following snippet shows how to grant these permissions in manifest.xml file.

```
<uses-sdk android:minSdkVersion="13" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission android:name="android.permission.INTERNET"/>
```

There are three activities in the application: “ThesisActivity”,”UseSimlation” and “CreateMapActivity”, as the part of manifest.xml below shows.

```
<activity
android:label="@string/app_name"
android:name=".ThesisActivity" >
    <intent-filter >

       •  •  •
    </intent-filter>
</activity>
    <activity
     android:label="@string/pop_simulation"
     android:name=".UseSimlation" >
</activity>
<activity
       android:label="@string/pop_map"
       android:name=".CreateMapActivity" >
</activity>
```

Figure 5.1 shows the file structure of the implementation on the Android device.

5.1.2.2 Layout files

Corresponding to the three activities, there are three layout xml files which declare every UI components in the interface.

The following snippet is a part of main.xml file which describes the layout for MainActivity activity. It claims Textview, RadioGroup, Radio Button and WebView component and their screen position.

- Main.xml: main frame layout xml file

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical"
    android:padding="10px" >

    <TextView
        android:id="@+id/tvChoose"
       . . .
    <RadioGroup
        android:id="@+id/ChooseTypeQueGroup1"
       . . .
    <WebView
        android:id="@+id/web_simulation"
       . . .
</LinearLayout>
```

- Simulation.xml: The layout interface for simulation activity.
- Mapview.xml: The layout for the Google map activity.

5.1.2.3 HTML & JavaScript Files

In order to migrate and upgrade the application smoothly, the mobile embedded browser is used to facilitate the communication between the servers and the clients. The following HTML and JavaScript files ensure message passing and parsing.

- *Request.html*: An HTML form to send the requests to servers.
- *Utility.js*: A JavaScript file used for message parsing.

The JavaScript library for Erlang JSON AJAX call is provided by YAWS which includes three JavaScript files. They are:

- *Jsonrpc.js*
- *Urllib.js*
- *Jsolait.js*

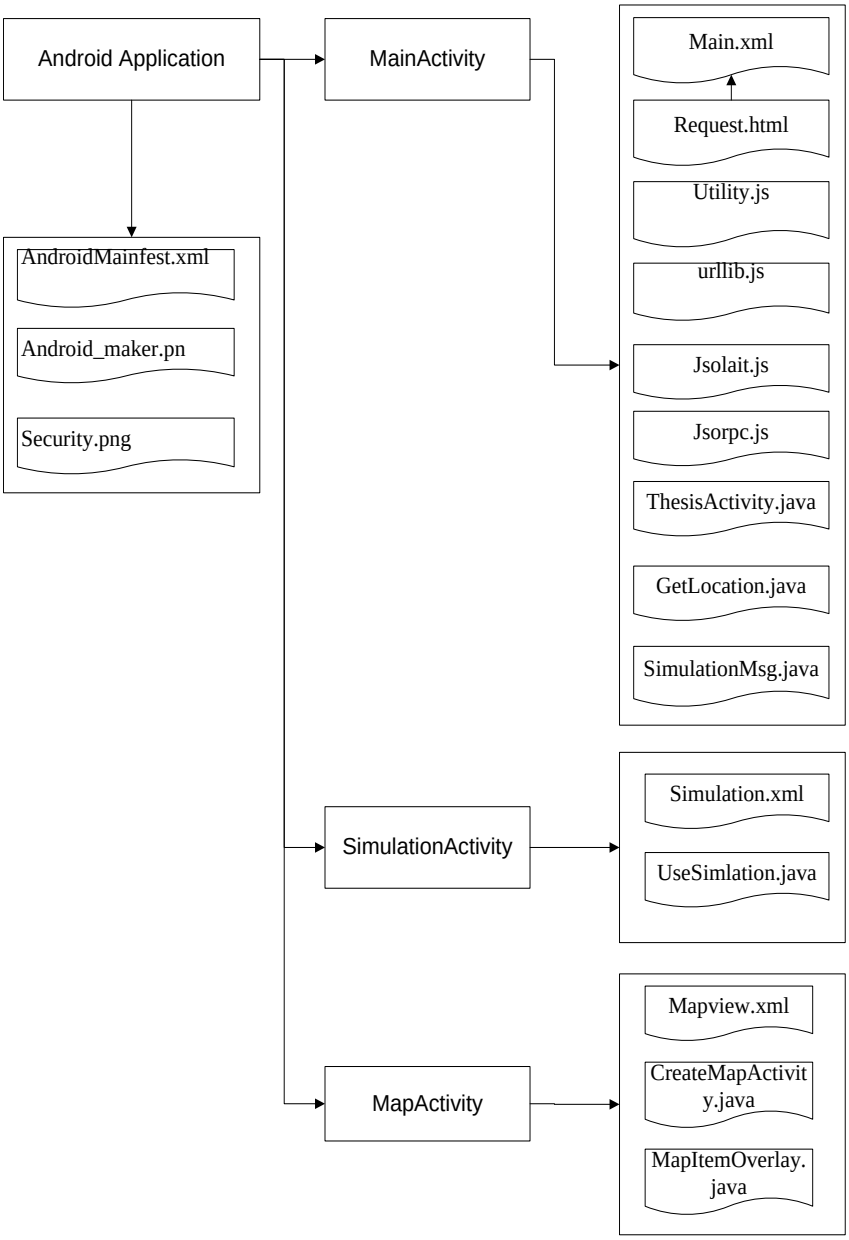

Figure 5-1 files and activities structure

5.1.2.4 Resource File

- *Android_maker.png:* A map marker resource file
- *Security.png:* The application log resource file

5.1.2.5 Java File

- *CreateMapActivity.java:* Defines the map view activity.

- *GetContext.java:* Retrieves the devices' current location information. The location information includes latitude, longitude and speed.
- *MapItemOverlay.java:* The Google map overlay class which shows the markers for simulated location that the users choose.
- *SimulationMsg.java:* Defines the structure of the communication message between the main activity and the simulation activity.
- *ThesisActivity.java:* The main activity. This is the entry activity for users and other activities are triggered by users' actions.
- *UseSimlation.java:* The simulation activity.

5.1.3 Mobile Application Design

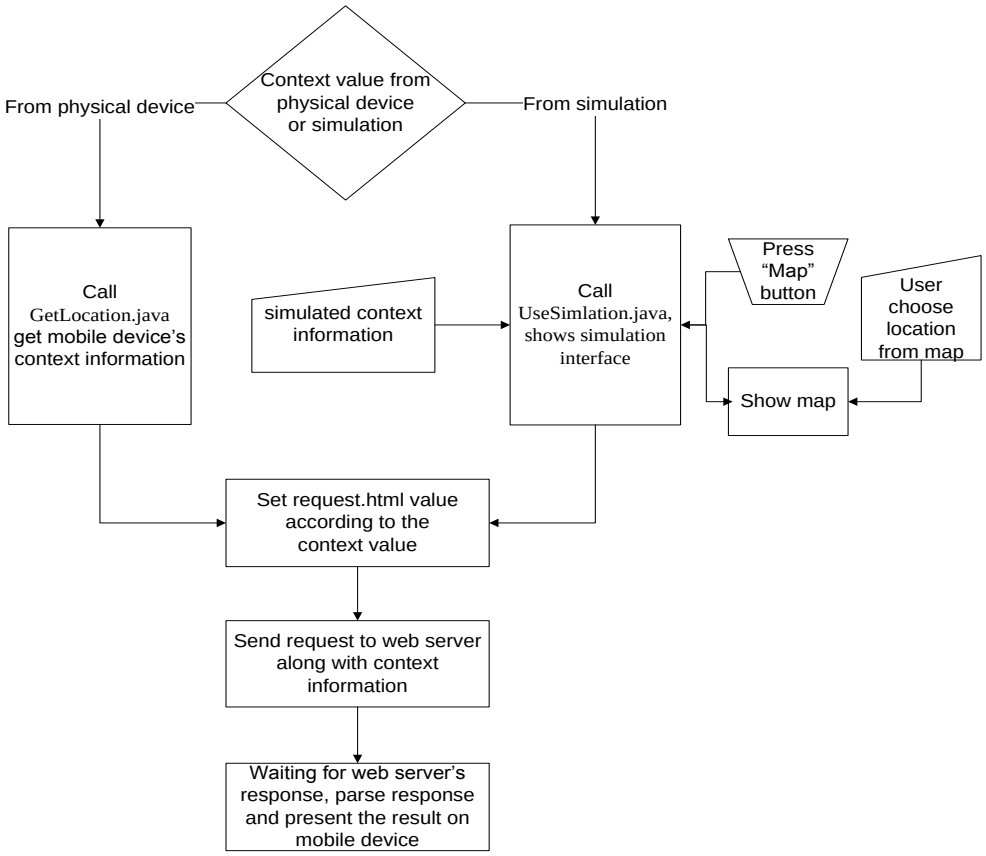

Figure 5-2 Android application process

As illustrated in Figure 5.2, the mobile application detects current context information and sends requests to the proxy server and the web servers. The interface of the main activity, *"ThesisActivity"*, shows as Figure 5-3, first users select the source of context information, the source can be detecting from the mobile device's hardware or users' simulation. If users choose *"context information from the device"*, the application calls class *"GetContext"* to get the current context information from the mobile device.

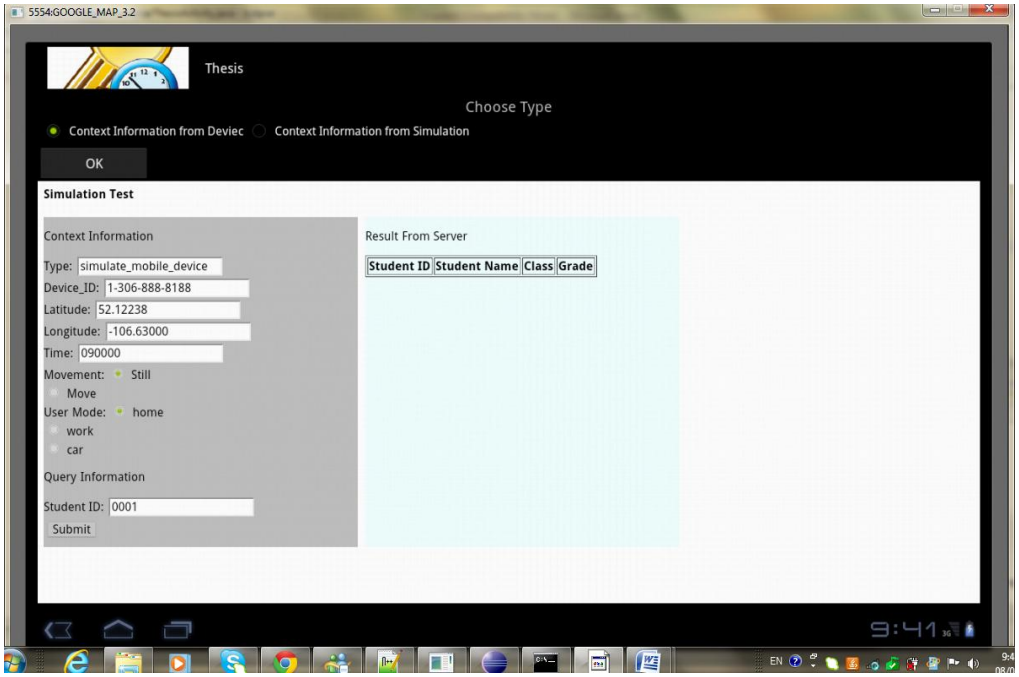

Figure 5-3 the main activity for Android Tablet

"SetWebValue" is a Java function in *"ThesisActivity"* class which calls JavaScript function *"setValue"*. This function sends data from the Java class to the HTML web page. The code snippet of the *"setWebValue"* function show as following:

```
protected void setWebValue(String s_lat,String s_lng,String s_speed,String s_time)
{
web_simulation.loadUrl("javascript:setValue(\""+s_lat+"\",\""+s_lng +"\",\""+s_speed
+"\",\""+s_time +"\")");
}
```

If users choose *"context information from simulation"*, the *"UseSimulation"* activity class is triggered, as figure 5-4 shows. In this activity, users enter the simulated context information.

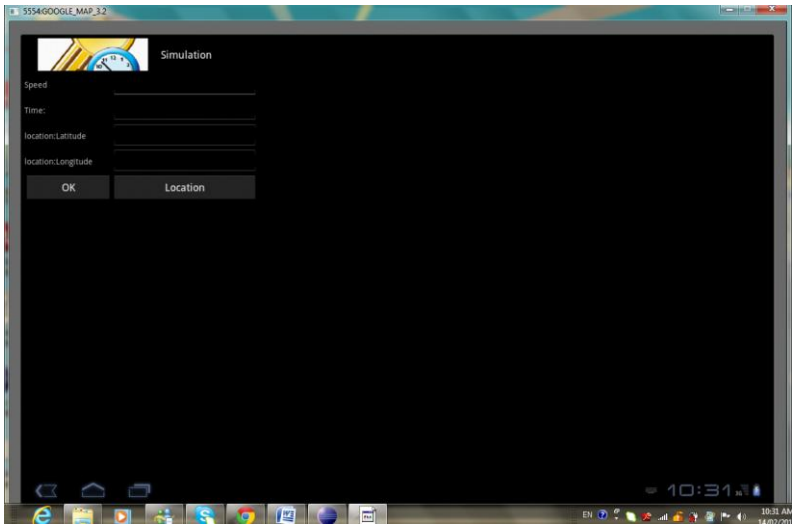

Figure 5-4 the simulated activity for Android tablet

By pressing the *"Location"* button, the *"CreateMapActivity"* activity is triggered. This

activity shows a map and let users choose the location. The Android marker shows the location users choose, as Figure 5-5 shows.

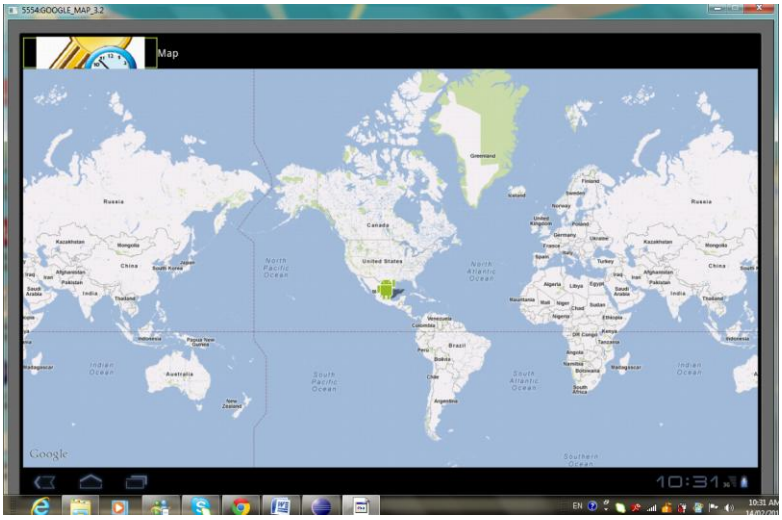

Figure 5-5 MapView activity for Android tablet

The *"startActivityForResult"* callback function is used to get the return value from other activities. The following snippet shows how to call *"UseSimulation" activity from "ThesisActivity"* activity.

```
Intent i = new Intent(ThesisActivity.this,UseSimlation.class);
startActivityForResult(i, SUB_ACTIVITY_REQUEST_CODE);
```

The *"UseSimulation"* activity set the returned value in a bundle, as the following snippet shows: the *"UseSimulation"* activity returns the simulated context information.

```
Bundle bundle = new Bundle();
f_speed=  Float.valueOf(et_speed.getText().toString());
s_time =et_time.getText().toString();
bundle.putInt("LAT",mPoint.getLatitudeE6());
bundle.putInt("LNG",mPoint.getLongitudeE6());
bundle.putFloat("SPEED",f_speed);
bundle.putString("TIME",s_time);
Intent mIntent = new Intent();
mIntent.putExtras(bundle);
setResult(RESULT_OK, mIntent);
finish();
```

5.1.4 JavaScript AJAX call for Erlang Module

The JavaScript remote procedure is used to capture the server's return message. YAWS1.9 provides JavaScript and Erlang library to call a server's function remotely. To call a server side function, the service's URL and a method need be defined to create a proxy for the server.

The following snippet shows the URL is "*http://domain_name:8080/mobile_services"*, and the method is *"get_result"*. They are explained into more detail in the proxy server and the web server implementation section.

```
<script>
var serviceURL = "mobile_services";
var methods = ["get_result"];

var jsonrpc = imprt("jsonrpc");
var service = new jsonrpc.ServiceProxy(serviceURL, methods);
function get_result() {
    try {
     type = document.getElementById("type").value;
      . . .
      document.getElementById('result').innerHTML =
       "<PRE>" + service.get_result(type,lat,lng,time,movement,user_profile,studentid)
+ "</PRE>";
     } catch(e) {
        alert(e);
     }
     return false;
}
</script>
```

If the context values meet the servers' requirements, the mobile client gets response from the server(s) successfully, as Figure 5-6 shows.

Figure 5.6 shows a successful result. In this scenario, grade information for a student with ID *"0001"* is retrieved in Saskatoon at 9:00am. The *"user mode"* is *"at home"*; the *"movement"* is "still". The device ID is *"1-306-881-8188"*. Since the grade information is distributed in different departments, this request is also sent to each department which has the grade information for this student. The trust value for this request is higher than the trust thresholds in each department, so the result, which combines responses from two departments, is successfully retrieved.

5.2 Proxy Server/Data Server Implementation

YAWS 1.9 is chosen as the proxy server and the web server because of its high performance which is suitable for dynamic changed mobile devices' services. YAWS 1.9 web server is written in Erlang which features light weight processes, thus YAWS has better performance for multiple concurrent requests and good scalability.

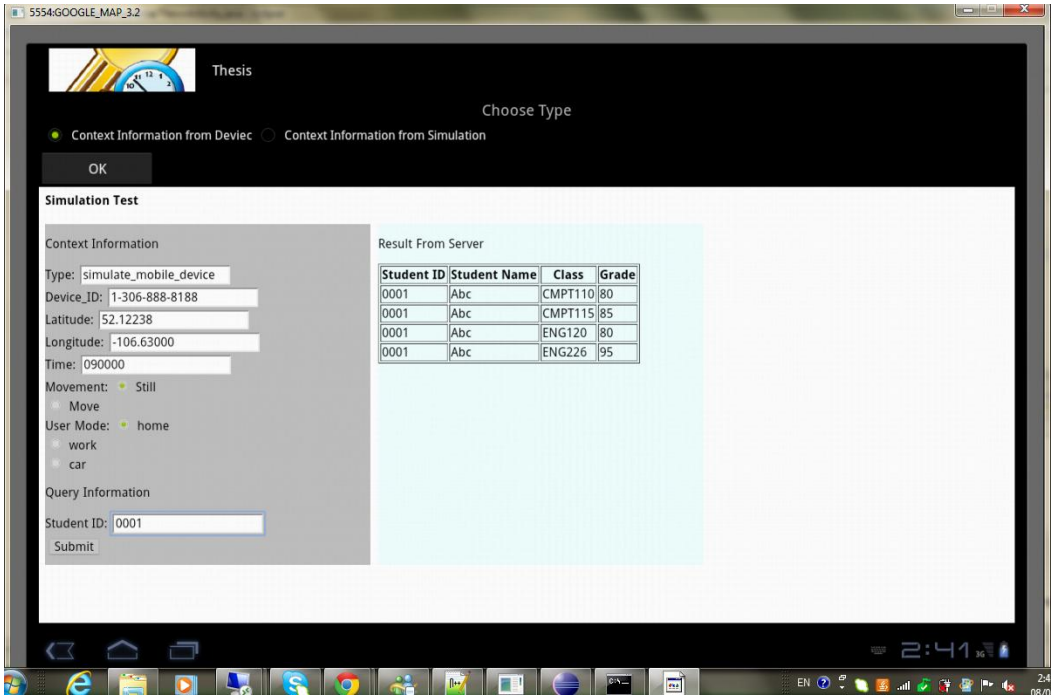

Figure 5-6 get student grade result

Based on different type of requestors, there are two main Erlang modules:

- *"mobile_services":* handles requests coming from mobile devices.
- *"proxy_services"*: handles requests coming from proxy servers.

5.2.1 mobile_services

The module "mobile_services" uses *"http://domain:8080/mobile_services"* as its URL. The function*"out"* is an entry point for handling requests, and it parses mobile devices' context information and calculates the mobile device's trust value. The process flow for the mobile services is shown in Figure 5.6.

As code snippet below shows, the function "out" calls the *get_result"* function as shown in the following snippet.

```
out(Arg) ->
                %% record current time
                utility:track_time_log("Start:"),
                mnesia:start(),
                %% ywas remote call
                yaws_rpc:handler_session(Arg, {?MODULE, get_result}).
```

The mobile device clients call the *"get_result"* function through a JavaScript function. The parameter *"Value"* in function "get_result" contains the mobile device's context information as a tuple, and the function first parses parameters *"Value"* to get mobile devices' context value and query information. It calls function *"calcuate_trust_mobile"* which calculates the trust value based on context value. The code snippet below shows the *"get_result"*function.

```
%% this is the function which is called by javascript
%% Value is the post parameters
get_result(Argument =_State, {call, get_result, Value} = _Request, Session)->
{array,[_,Device_id,Lat,Lng,Time,Movement,User_profile,StudentID]} =Value,
case
%% should we calculate trust, if so, call calculate_trust_mobile, otherwise,
%% set Total_trust_value to full trust value
      ?CALCULATE_TRUST of
       true->
            %% get the trust value of current transaction
            Total_trust_value
            =calcuate_trust_mobile(Time,Lat,Lng,User_profile,Movement);

      . . .

      %% get current domain and foward domain list
      Current_domain = route:get_current_domain("student_info"),
      %% get forward domain list
      Domain_list =route:get_foward_domain("student_info"),
      Context_info =lists:concat([Time,Lat,Lng,User_profile,Movement]),
      case
      %% handle the request according to the domain list
      handle_multiple_request(Domain_list,Device_id,StudentID,Total_trust_val
ue,Current_domain,Context_info) of

      . . .
end.
```

The function *"route:get_foward_domain('student_info')"* in the above code checks whether the current server can provide direct services or not, if not, then the request is forwarded to the proper server(s). Before forwarding this request to the next

server, the server recalculates the trust value and sets this value in the request header. If the other server(s) respond successfully, then the server forwards the responses to mobile devices; otherwise, it returns error messages.

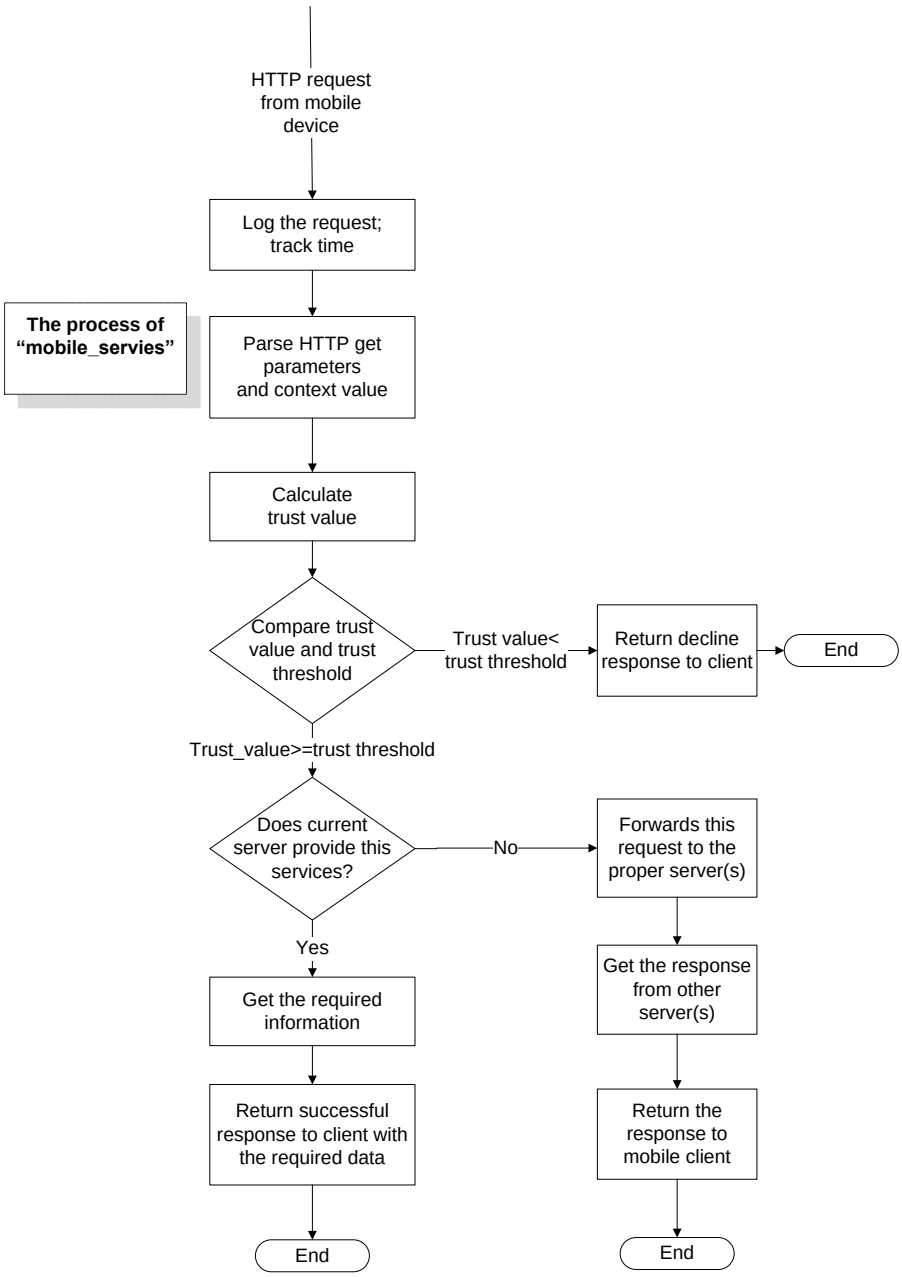

Figure 5-7 Process of module "mobile_services"

5.2.2 proxy_services

The module *"proxy_services"* uses *"http://domain:8080/proxy_services"* as its URL. The process flow is shown as Figure 5-8.

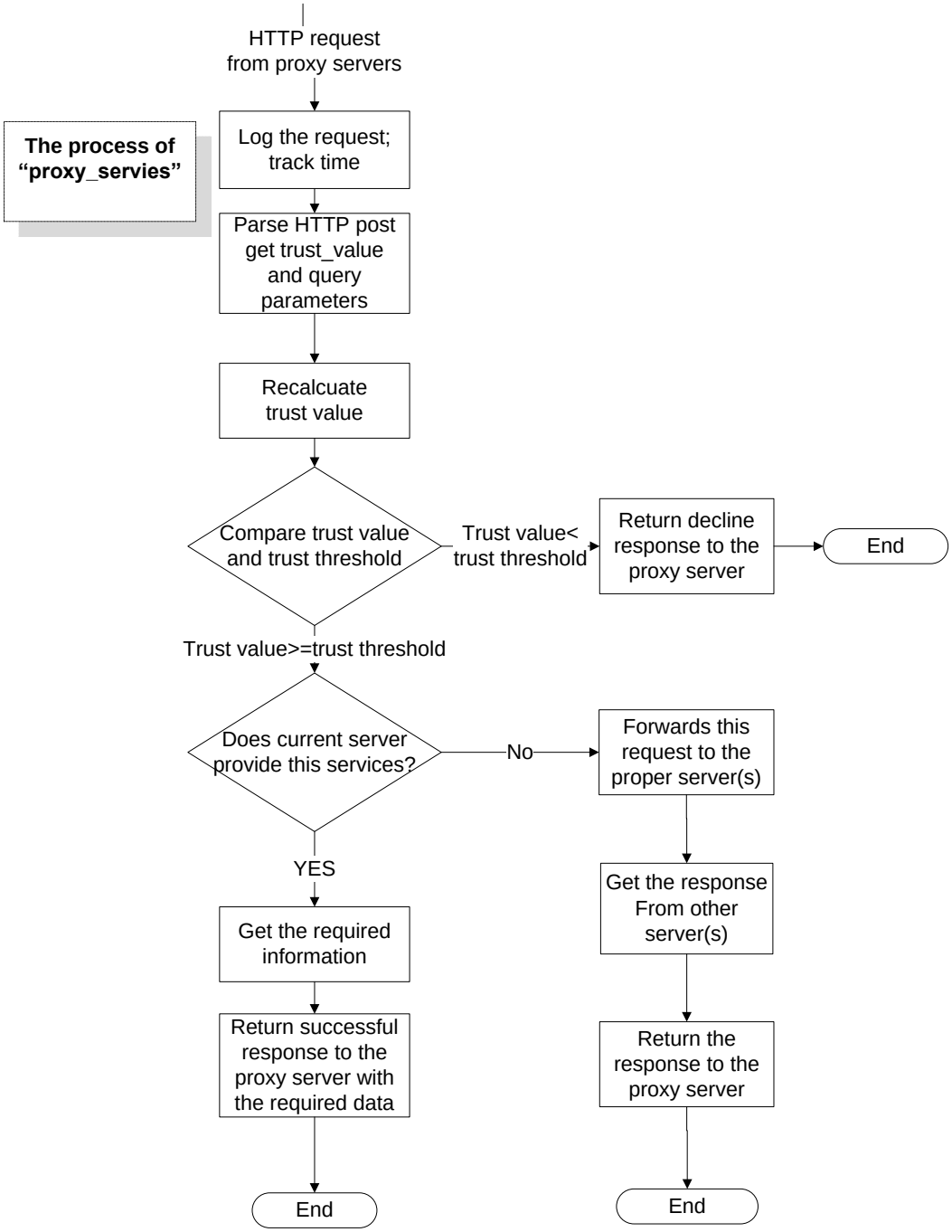

Figure 5-8 process of module "proxy_services"

The “out” function is an entry function for “proxy_services” module as the following snippet shows. The *“out”* function calls the *“forward_request”* function to get queried information, if it successfully gets the result which is JSON format, it returns the JSON data; otherwise it returns an error message.

```
out(Arg) ->
        %% record current time
        utility:track_time_log("start:"),
        mnesia:start(),
        %%call foward_request , get json object
        Response =foward_request(Arg),
        utility:track_time_log("get response from foward server"),
        %% generate http response header
        Header =http_server_utility:generate_header(),
        case
              Response of
              %% if succeeds
              {ok,NewBody} ->
                        [{status,200},
                         {header,{"Vary","Accept"}},
                         {content,"application/json",NewBody}];
              %% if fails
            {error,Ret,NewReasonPhrase} ->
                  [{status,Ret},
                         Header,
                         {content,"text",NewReasonPhrase}]
        end.
```

As the following snippet shows, the *“forward_request”* function first parses the query parameters and gets the trust value from the HTTP header. If the trust value is greater than or equal to the trust threshold, the process continues, otherwise an error message is returned. The rest of the process is same as *“mobile_services”* module: it checks the *“route”* table to decide if it provides the direct services or not, if so, it retrieves the information from the Mnesia database, otherwise it forwards the requests to the next server(s). Before sending the request to the next server(s), it reset the trust value in the request header, and encodes the parameters which include the current domain name and student ID.

```
%% it is called by out module
%% it parse request parameters,recalculate trust value and call
%% handle_multiple_request to handle the request
foward_request(Arg)->
      %% get http request parameters
      Response =http_server_utility:get_query_parameters(Arg),
      %% get student ID from parameters
      StudentID =http_server_utility:search_parameter("studentID",Response),
      %% get trust value from http header
      Request_trust=http_server_utility:get_request_trust(Arg),
      case Request_trust of
      . . .
      end,
      %% get the coming request's domain
      Request_domain
      =http_server_utility:search_parameter("current_domain",Response),
      . . .
      Total_trust_value = Trust_value_float * Request_domain_trust_value,
      Current_domain = route:get_current_domain("student_info"),
      %% get the foward domain list
      Foward_domain =route:get_foward_domain("student_info"),
      %% handle the request through the domain list
      handle_multiple_request(Foward_domain,StudentID,Total_trust_value,Curre
nt_domain).
```

5.2.3 Other Erlang Module

Besides the two main web services, there are other Erlang modules which server as utility functions and data providers.

"grade": This module maintains table "grade". It provides updating and querying functionalities, also provides data formation conversion such as from data list to JSON object, or from data list to string etc.

The record *"grade"* is defined as:

```
-record(grade,{studentid,studentname,class,grade}).
```

The table is created in *"disc_copies"* modes on the local node, using bag type.

```
init() ->
    mnesia:create_schema([node()]),
    mnesia:start(),
    mnesia:create_table(grade,
        [ {disc_copies, [node()] },
             {attributes,record_info(fields,grade)},
                   {type, bag} ]),
    insert_grade_data().
```

The function *"insert_grade()"* inserts a record into this table. It defines a write transaction for table "grade". The four parameters *"id,name,class,grade"* are fields of *"grade"* records.

```
insert_grade( Id,Name,Class,Grade) ->
 Fun = fun() ->
      mnesia:write(
      #grade{studentid =Id,
                     studentname=Name,
                     class=Class,
                     grade=Grade } )
            end,
      mnesia:transaction(Fun).
```

The "*insert_grade_data()"* function calls *"insert_grade()"* function and creates some initial records for *"grade"* table.

```
%% insert grade data
insert_grade_data()->
      insert_grade("0001","Abc","CMPT110","80"),
      …
      insert_grade("0001","Abc","CMPT115","85").
```

The following code searches the record with the given student ID.

```
select_id(Id) ->
    Fun =
        fun() ->
              mnesia:read(grade,Id)
        end,
    {atomic, Results}=mnesia:transaction(Fun),
Results.
```

To convert an Erlang list to a JSON object, the mochijsion2 library [22] is called. In the following example, the Erlang list is first converted to an array variable and then a JSON object through function *"encode()"* from the mochijsion2 library.

```
%% create erlang list to json structure
generate_json_term(List)->
      generate_json_term(List,[]).

generate_json_term([Head|Tail],Prev) ->
      {grade,Studentid,Studentname,Class,Grade} =Head,
      Tmp
={struct,[{"id",Studentid},{"name",Studentname},{"class",Class},{"grade",Grad
e}]},
      generate_json_term(Tail,lists:append(Prev,[Tmp]));

%%generate json object using the return array of student info
generate_json_term([],Result) ->
      {array,Result}.

%% create json object from the generate json structure
generate_json(List)->
      Json_term =generate_json_term(List),
      mochijson2:encode(Json_term).
```

"http_server_utility": It provides HTTP server side functionalities, including getting query parameters, getting clients' post data and checking header information.

"http_client_utility": It provides HTTP client side functionalities, including generating post and get requests data; presenting data on HTML pages.

"mobile_context": It supports looking up for a trust value based on mobile devices' context value. Each category of mobile context has a set of trust values, in the implementation, the trust policies are set based on time, location and other factors. Take location as an example, if the location of the incoming request is within the campus of the University of Saskatchewan, the trust value is 0.3, if not within the campus but within the city of Saskatoon, the trust value is 0.1, otherwise it is 0.0. Given a set of context values, total trust value can be calculated by calling *"mobile_context"* function.

"route": This module maintains table *"route"*, table *"route"* provides route path for each type of requests. Under some scenarios, a request is sent to multiple domains to get a proper result, for example, on the server *"sermua.usask.ca:8080",* a request *"get_student_info"* is set to be forwarded to two web servers. First the request is sent to

"xoxo.usask.ca:8080/proxy_services", then the request is sent to *"yuting.usask.ca:8080/proxy_services"*. The two sets of returned JSON objects are put together and the combined result is sent back to the mobile device.

"domain_trust_value": It maintains table *"domina_trust_value"*. The *"domain_trust_value"* table keeps information about how much the current domain trusts other domains. When the current domain recalculates the trust value for the incoming requests, how much it trusts the request domains is used.

"mochijsion2": This is a JSON library module written by Bob Ippolito [22]. It exports encode () and decode () functions to convert a JavaScript JSON object to a Erlang binary list and the vice versa.

5.3 Data Model

The Mnesia database is used in this implementation. The system's data structure, setting Mnesia database in Erlang node and data transaction are discussed in this section.

5.3.1 Data Structure

Schemas and tables are Mnesia's basic data structure. In this system, some tables serve as setting configuration information and some serve as data providers.

Tables are created by *"init"* function in each module. Also, there are functions to maintain the tables.

Tables are described in the following sections.

5.3.1.1 Trust Policy

This table builds a link between the mobile devices' context value and the trust value.

For example, if a request is sent from the University of Saskatchewan campus and the trust value is 0.3, then this record is: keywords :*"Location"*; policy_name: *"U of S trust*

value"; criteria: *"1;lat:>=52.12137 and <=52.14271;lng:>=-106.63845 and <=-106.62197"*; trust_value: *0.3*.

Table 5-1 trust policy table

trust_policy			
It exists in each proxy server node and web server node bag type table; disc copy			
Fields Name	Type	Key Words	Description
keywords	string		identifies mobile device context information "location","time","user profile","movement"
policy_name	string		The name for each policy
criteria	string		this field gives more detail specification For example: if location is within Saskatoon, the trust value is 0.1, otherwise it is 0.
trust_value	float		Trust Value if the criteria are matched

5.3.1.2 Domain Trust Mapping

This table builds a trust relationship between the current domain and other domains it interacts with.

For example: if the current domain is *serum.usask.ca:8080* and it fully trusts domain *xoxo.usask.ca:8080*, then this record on node *sermua.usask.ca* is: domain_name: *"xoxo.usask.ca:8080"*, trust_value:*1*

Table 5-2 Domain Trust Mapping Table

domain_trust_mapping			
It exists in each proxy server and web server node. set type table; disc copy			
Fields Name	Type	Key Words	Description
domain_name	string	yes	URL for each servers
trust_value	float		trust value, value ranges from 0-1 0—is not trusted; 1-fully trusted

5.3.1.3 Request Routing

This table builds a path for each type of requests on each server node. A request type has one or more route records. When a request comes, it is either served or forwarded to other server(s) according to the route setting.

For example: when domain *serum.usask.ca:8080* receives a request asking for querying a student grade information, it sends the request to other two domains: *"xoxo.usask.ca:8080"* and *"yuting.usask.ca:8080"*, then the two records on "*sermua.usask.ca"* are show as Table 5.3:

Table 5-3 an example of route table

Request Type	**Seq.**	**Current request path**	**Forward request path**	**Response request path**	**Trust threshold**
student_info	1	serum.usask.ca:8080	xoxo.usask.ca:8080/ proxy_server	""	0.5
student_info	2	serum.usask.ca:8080	yuting.usask.ca:8080 /proxy_server	""	0.5

Table 5-4 routing table

request_routing			
It exists in each proxy server node and web server node bag type table; disc copy			
Fields Name	Type	Key Words	Description
request_type	string		URL for each server; different type request is routed to send to different servers.
Sequence	int		Forwarding domain sequence#
current_request_path	string		Refers to current URL for the requests
foward_request_path	string		Refers to forwarding URL.
response_request_path	string		Not used
trust_threshold	float		The trust threshold for the current domain

5.3.1.4 Mobile Device Registry Information

To reduce the risk for mobile devices, the previous transactions' trust values are used to calculate the mobile device's overall trust value.

Table 5-5 Mobile Device Register Table

mobile_device			
It exists in each proxy server node and web server node set type table; disc copy			
Fields Name	Type	Key Words	Description
mobile_id	string	Yes	Mobile device ID
last_trust_value	decimal		The second last transaction's trust value
trust_value	decimal		The last transaction's trust value

5.3.1.5 Student Grade Information

Student grade information is used as an example used in this implementation. Mobile devices invoke student grade information via student ID.

Table 5-6 Student Grade

grade			
It exists in each web server node Bag type table; disc copy			
Fields Name	Type	Key Words	Description
studentid	string	Yes	ID
studentname	string		student name
class	string		class name
grade	integer		Grade

5.3.1.6 Transaction Information

This table is transaction logs for the HTTP requests. When a server receives a request, it logs the related information for future reference.

Table 5-7 Transaction Table

Trans			
It exists in each proxy server node and web server node Bag type table; disc copy			
Fields Name	Type	Key Words	Description
mobile_id	string		Mobile ID
time	string		Transaction happen time
request_type	string		Request type, for example,

			“GET_STUDENT_INFO”
server	string		Current server name
trust_value	string		Calculated trust value
context_info	string		Mobile context information
result	string		Success or fail

5.3.2 Setting Mnesia database in Erlang nodes

Mnesia database is running on top of Erlang code. In order to share the Mnesia database, Erlang nodes need to be connected. Erang node is executing running time Erlang system, and they communicate with each other through TCP/IP connection.

- Set TCP/IP connection

Make sure computers can *“ping”* with each other. If not, check the TCP/IP connect. There could be some other reasons for connection failure, for example: DNS name setting, firewall setting etc.

- Start Erlang nodes

Start Erlang using the following command:

Erl –name computer1

This command starts a Erlang running with name *computer1@computer1.usask.ca*, *“computer1”* is the name used when the Erlang node is started, *“computer1.usask.ca”* is the machine’s host name.

- Set Erlang cookies

Every Erlang node has its own cookie. Under the windows system, the Erlang cookie is stored in *“user\username\erlang.cookie”*. In order to communicate with other nodes, the cookies must be same for the security reason. Run the following command to set cookie to “trust_model”.

erlang:set_cookie(node(),”trust_model”).

- Connect Erlang nodes

Run command "*net_adm:ping"*, to connect to other Eralng nodes. For example, computer1 uses the following command to connect computer2 under domain "usask.ca".

net_adm:ping("computer2@computer2.usask.ca")

- Building distributed Mnesia schema

mnesia:create_schema([node()|nodes()]).

This command creates a Mnesia database schema on all the Erlang nodes to which it connects to.

- Building distributed tables under the schema

First, start Mnesia database on all connected Erlang nodes.

Mnesia:start().

Then create table on all connected nodes, for example, create student grade information tables using the following code:

```
mnesia:create_table(grade,
    [ {disc_copies, [node()|nodes()]},
    {attributes,record_info(fields,grade)},
       {type, bag} ]),
```

- Sharing data in each Erlang node

Each node can share the data now. The updating happens on one node spreads out to other nodes. For example, "Student Grade Information" records are inserted at computer1.

```
insert_grade_data()->
insert_grade("0001","Abc","CMPT110","80"),
insert_grade("0001","Abc","CMPT115","85"),
insert_grade("0002","Def","CMPT110","90"),
insert_grade("0002","Def","CMPT115","95"),
insert_grade("0003","Ghi","CMPT110","70"),
insert_grade("0003","Ghi","CMPT115","75").
```

These records can be seen on other Erlang nodes.

- Data Initialization

To run the system properly, tables include: *"trust_policy"*, *"domain_trust_mapping"* and

"request_routing" are initialized. The table *"trust_policy"* reflects the business requirements which are the basis for calculating trust value. The table *"domain_trust_mapping"* keeps the trust relationship between each server. The table *"request_routing"* lists the path information for each type of requests. For the experiments of the system, the table*"grade"* is also initialized for showing the request result. Each module has a function *"init"* which added initial records at the beginning.

5.3.3 Mnesia Table Operation

There are two tables associated with transaction updates. The table *"mobile_device"* keeps mobile ID and its last two transactions' trust values. The table "trans" keeps transaction information including request mobile ID, transaction time, request context value and transaction's result.

When a requests come, the system checks the previous trust value of this mobile device from the table *"mobile_device"*, and then calculates trust value based on the context information and the previous trust value of the mobile device. If there is no previous trust value, then this mobile device is regarded as a new mobile device and the

system automatically registers it, and a trust value “0” is used as a initial trust value. After finishing the HTTP session, a transaction record is written to the distributed Mnesia databases, and the previous trust values of *“mobile_device”* are updated according to the trust value of the current transaction.

CHAPTER 6
EXPERIMENTS

6.1 Experiment environment setup

An android tablet, a client with Internet browser and four YAWS web servers are used for the experiments.

6.1.1 Android tablet

The Android tablet‘s hardware is listed as following:

Model #: Motorola XOOM MZ604

CPU: Dual-core 1 GHz Cortex-A9

Memory: 32GB

Storage: 30GB

Sensors: GPS, accelerometer, gyro, barometer, compass

6.1.2 Client terminal

In order to test the system’s loading capacity, a batch of requests is sent to the server(s). In this case, using an Android tablet is not feasible, so APACHE JMETER is used. APACHE JMETER is a desktop application written in JAVA which is designed for loading and performance testing.

The terminal needs Java SE 1.5 or later and APACHE JMETER 2.6. APACHE JMETER could be downloaded from *“http://jmeter.apache.org/download_jmeter.cgi”* [35].

In each test plan, a thread group or more are created. *“Number of Threads (users)”* represents the number of concurrent users. *“Ramp-Up Period (in seconds)”* indicates the

total time for APACHE JMETER to create the users. *"Loop Count"* specifies how many times the test is to be repeated.

Under each thread, a sample for HTTP request, three listeners 1) views result, 2) summary report, and 3) aggregate report are created. Requests' average process time, median process time, maximum process time and error rate are recorded. In this HTTP request, suppose a POST request is sent to web server *"xoxo.usask.ca"*, and the path is *"mobile_services"*, then the HTTP request's setting is shown as Figure 6-1.

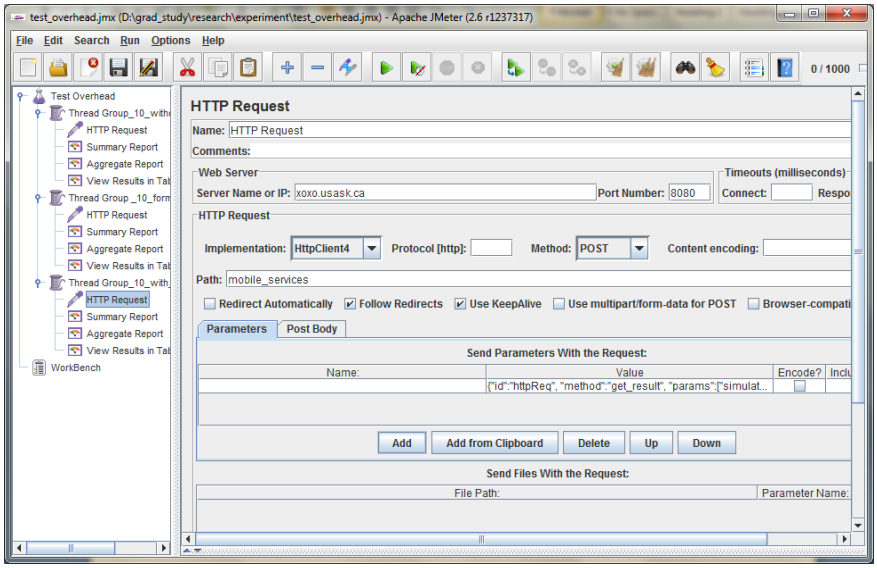

Figure 6-1 APACHE JMETER HTTP -request

The result can be seen in the *"result table"* after this experiment is completed. The summary information is gathered and presented in the *"aggregate report"* and the *"summary report"*.

6.1.3 Proxy/web servers

YAWS servers act as proxy servers and web servers with a Mnesia database. In this experiment, four servers are used as the proxy servers and the web servers. The configuration of the four servers is list in table 6-1.

Table 6-1 servers setting

	CPU	**Memory**	**Storage**	**OS**	**Other Software**
burgas.usask.ca	Intel (R) Xeon(TM) CPU 3.20GHz	5.0GB	69.1GB	Windows 7 Enterprise	Erlang OTP R14B03 YAWS 1.91 Web Servers
yuting.usask.ca	Intel (R) Xeon (R) CPU 2.33GHz	10GB	465GB	Windows 7 Enterprise	Erlang OTP R14B03 YAWS 1.91 Web Servers
xoxo.usask.ca	Intel (R) Xeon(R) CPU 2.33GHz	16GB	148GB	Windows 7 Enterprise	Erlang OTP R14B03 YAWS 1.91 Web Servers
varna.usask.ca	Intel (R) Xeon(TM) CPU 3.20 GHz	4 GB	69.1GB	Windows 7 Enterprise	Erlang OTP R14B03 YAWS 1.91 Web Servers

6.1.4 System setup

To evaluate system's functionalities, an android tablet, and a YAWS web server with Mnesia database are set up as figure 6-2.

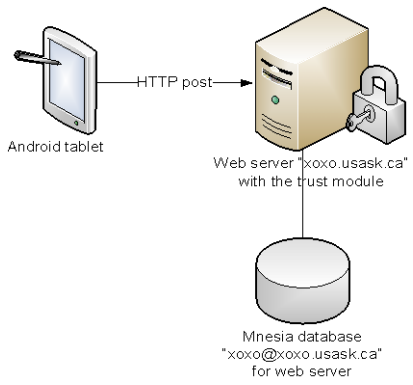

Figure 6-2 Experiment setup for testing functionalities

Different situations are simulated to test diverse trust policies. HTTP post requests with the combinations of different context values are sent to web servers, trust polices and trust thresholds are adjusted according to the situations.

To evaluate the system's scalability and overhead for the trust module, four scenarios are categorized and tested in the experiments.

Scenario A includes 1) a client running APACHE JMETER, 2) A YAWA web server with the Mnesia database as figure 6-3 shows.

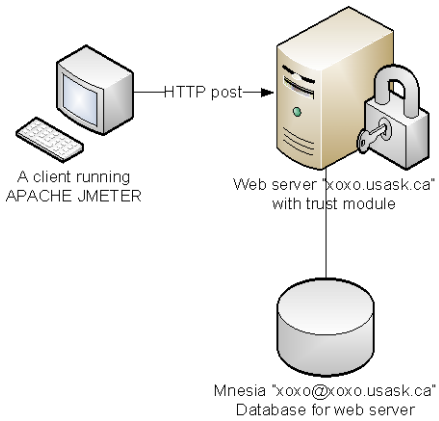

Figure 6-3 Experiment setup for scenario A

Scenario B includes 1) a client running APACHE JMETER, 2) a YAWA web server (“yuting.usask.ca”) with the Mnesia database and 3) a YAWS proxy server (“xoxo.usask.ca”) with the Mnesia database.

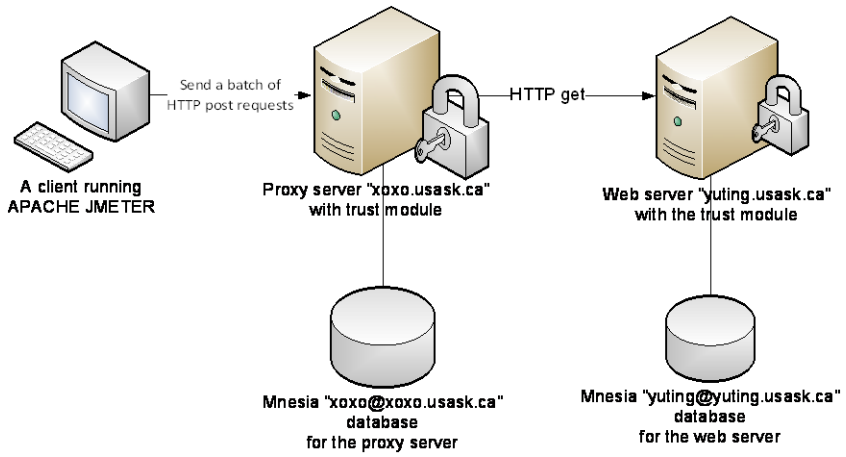

Figure 6-4 Experiment setup for scenario B

Scenario C includes 1) a client running APACHE JMETER; 2) two YAWA web servers (“yuting.usask.ca” and “burgas.usask.ca”) with the Mnesia databases and 3) a YAWS proxy server (“xoxo.usask.ca”) with the Mnesia database.

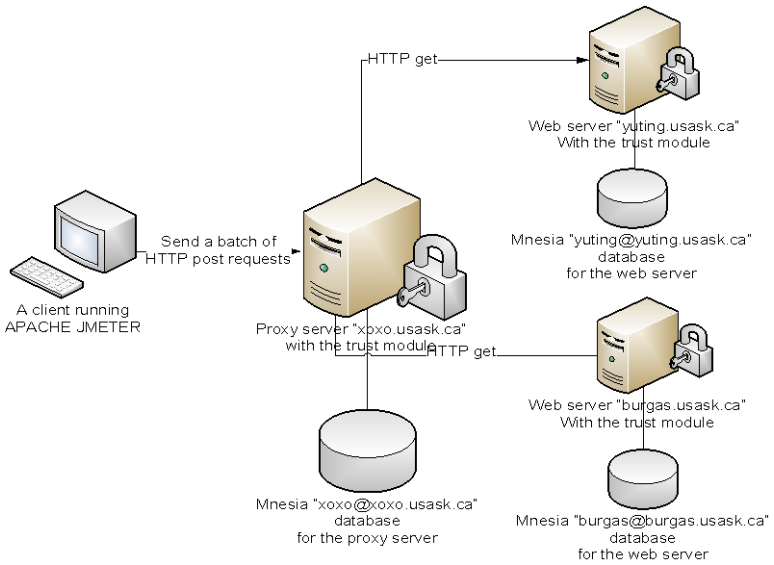

Figure 6-5 Experiment setup for scenario C

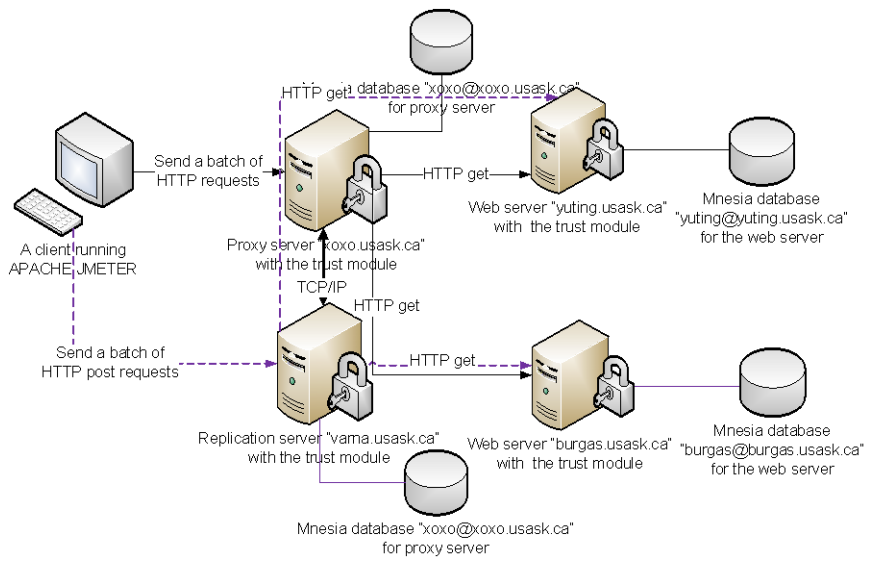

Figure 6-6 Experiment setup for scenario D

Adding a replication server can improve the system's scalability. System with a replication server is tested in scenario D. Scenario D includes 1) a client running APACHE JMETER, 2) two YAWA web servers with the Mnesia databases and 3) a YAWS proxy server and its replication server with the Mnesia database.

6.1.5 Trust formulas

As mentioned in chapter 4, the two formulas that are used for calculating mobile devices' trust value are shown in equation 6.1 and 6.2.

$$T_{total} = \alpha\, T_i + \beta * min((T_{i-1} - T_h) ,0) + \lambda * min((T_{i-2} - T_h) ,0) \quad \textbf{.....(6.1)}$$

$$T_{total} = \alpha\, T_i \quad \textbf{..... (6.2)}$$

Equation 6.1 calculates the mobile devices' trust value based on context values and previous transactions while equation 6.2 calculates the mobile devices' trust value based only on context value.

One formula used for calculating other domain's trust value is shown in equation 6.3.

$T_{current} = T_{previous} * T_{current\text{-}previous}$ ..... **(6.3)**

Three cases are evaluated in each scenarios when evaluate the system's scalability and the trust module's overhead, they are 1) the trust module using formula 1, 2) The trust module using formula 2 and 3) Without the trust module.

Case 1: the trust module using formula 1

The system calculates the trust values of requests from mobile devices using equation 6.1 and calculates requests from other domains using equation 6.3.

Case 2: the trust module using formula 2

The system calculates the trust values of requests from mobile devices using equation 6.2 and calculates requests from other domains using equation 6.3.

Case 3: Without the trust module

The system does not calculate any trust values. The trust module is not used in this case.

6.2 Experiment results

6.2.1 Functionality

Checking functionality is one of the experiments' goals. The trust module is evaluated under different business requirements. Varieties of combination of trust policies and trust thresholds are set according to these requirements.

Scenario 1: the request must be sent at University of Saskatchewan and must be sent between 9am and 5pm.

The trust policies are set as table 6-2:

Table 6-2 Scenario 1 trust rules

Context	Criteria	Trust value
Location	Within University of Saskatchewan	1.0
Time	Invoking time is greater or equal than 9am and less than 5pm	1.0

Trust formula: trust formula 2 which calculates the mobile devices' trust value(s) is only based on current context information.

Trust threshold: 2.0.

The experiment setup is as figure 6-2 shows.

The experiment results are shown in table 6-3.

Table 6-3 Scenario 1 experiment results

Condition	Result
Sending a request on the University of Saskatchewan campus at 10am	Get required information
Sending a request on the University of Saskatchewan campus at 12pm	Get an error message: "The trust value is not enough"
Sending a request in another place rather than the University of Saskatchewan campus at 10am	Get an error message: "The trust value is not enough"

Scenario2: If the request must be sent at University of Saskatchewan and must be sent between 9am to 5pm, and the mobile device's credit must be good.

The trust policies are set as in table 6-4:

Table 6-4 Scenario 2 trust policies

Context	Criteria	Trust value
Location	Within University of Saskatchewan	1.0
Time	Invoking time is greater or equal than 9am and less than 5pm	1.0

Trust formula: trust formula 1 which calculates the mobile devices' trust values based on context information and previous transactions.

Trust threshold: 2.0

The experiment setup is as figure 6-2 shows.

The experiment results are shown in table 6-5.

Table 6-5 Scenario 2 experiment results

Condition	Result
Sending a request on University of Saskatchewan campus at 10am at first time	Get required information
Sending a request on University of Saskatchewan campus at 12pm	Get an error message: "The trust value is not enough"
Sending a request in other place rather than University of Saskatchewan campus at 10am	Get an error message: "The trust value is not enough"
Sending a request on University of Saskatchewan campus at 10am at second time	Get an error message: "The trust value is not enough"

Scenario 3: If request location is not within Canada or the sending time is not between 9am to 5pm, then the request is prohibited.

The trust policies are set as in table 6-6:

Table 6-6 Scenario 3 trust policies

Context	Criteria	Trust value
Location	Within the city of Saskatoon	1.0
Time	Invoking time is greater or equal than 9am and less than 5pm	1.0

Trust formula: trust formula 2 which calculates mobile devices' trust values only based on current context information.

Trust threshold: 1.0

The experiment setup is as figure 6-2 shows.

The experiment results are showed in table 6-7.

Table 6-7 Scenario 3 experiment results

Condition	Result
Sending a request on University of Saskatchewan campus at 10am	Get required information
Using simulation interface, simulate another place rather than Saskatoon and sending a request at 10am	Get required information
Using simulation interface, simulate another place rather than Saskatoon and sending a request at 12pm	Get an error message: "The trust value is not enough"

Scenario 4: the request must be sent using "at office" profile and must have good credit.

The trust policy is set as table 6-8.

Table 6-8 Scenario 4 trust policy

Context	Criteria	Trust value
User profile	Office	2.0

Trust formula: trust formula 1 which calculates the mobile devices' trust values based on context information and previous transactions.

Trust threshold: 2.0.

The experiment setup is as figure 6-2 shows.

The experiment results are shown in table 6-9.

Table 6-9 Scenario 4 experiment results

Condition	Result
Sending a request using "at office" profile.	Get required information
Sending a request using other profile rather than "at office" several times.	Get an error message: "The trust value is not enough"
Sending a request using "at office" profile again.	Get an error message: "The trust value is not enough"

Scenario 5: if the request must not be sent when driving.

The trust policy is set as table 6-10.

Table 6-10 Scenario 5 trust policy

Context	Criteria	Trust value
Speed	=0	2.0

Trust formula: trust formula 1 which calculates the mobile devices' trust values based on context information and previous transactions.

Trust threshold: 2.0.

The experiment setup is as figure 6-2 shows.

The experiment results are shown in table 6-11.

Table 6-11 Scenario 5 experiment results

Condition	Result
Sending a request when the mobile device is in still state.	Get required information
Sending a request when the mobile device is in moving state.	Get an error message: "The trust value is not enough"

6.2.2 Scalability

The system's maximum capacity is tested for the system's scalability. The system's scalability is discussed from three aspects: 1) concurrent users, 2) transfer message size, 3) the number of server hops. Concurrent users are changed by adding concurrent threads in JMETER; transfer message size is changed by changing the records of the students' score information; the number of server hops is changed by changing the architecture of the system. To evaluate system's scalability under different scenarios mentioned above, the maximum capacity and average process time are tested. Under each scenario,

experiments are tested several (3-5) times, a typical set of data are chosen as the experiment results.

- Scenario A

Two different message sizes, of 2K and 3K, and gradually increased concurrent users are tested for this scenario.

Set test plan in JMETER as following:

Concurrent: start from 100 concurrent users, gradually add 100 concurrent users every time till the server(s) crashes, or the error rate is greater than 10%.

Ramp-up period: 10 seconds.

Loop count: 10.

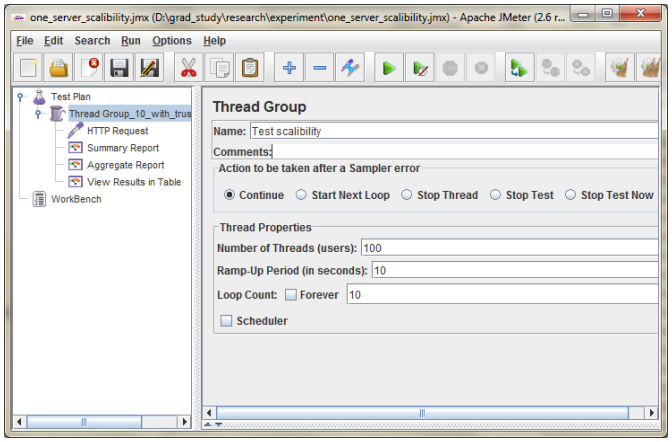

Figure 6-7 JMETER setup for testing scalability (1)

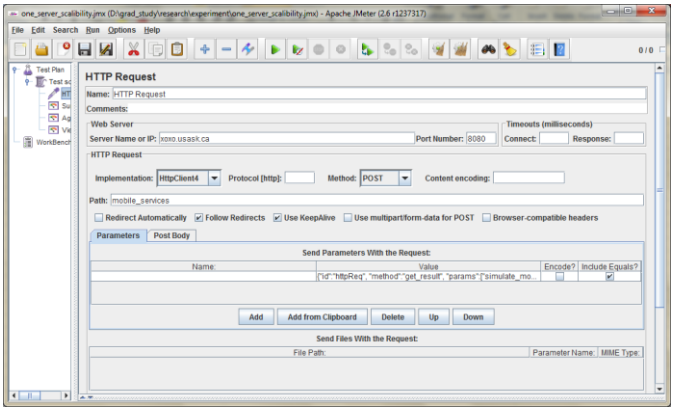

Figure 6-8 JMETER setup for testing scalability (2)

The results for average process time and median process time are shown in figure 6-9 and figure 6-10 for message size 2K. The horizontal number represents concurrent users and the vertical number represent average process time in milliseconds.

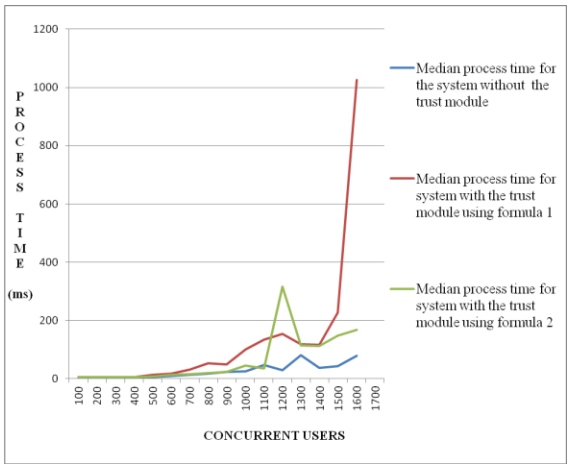

Figure 6-9 Average process time for scenario A (message size 2K)

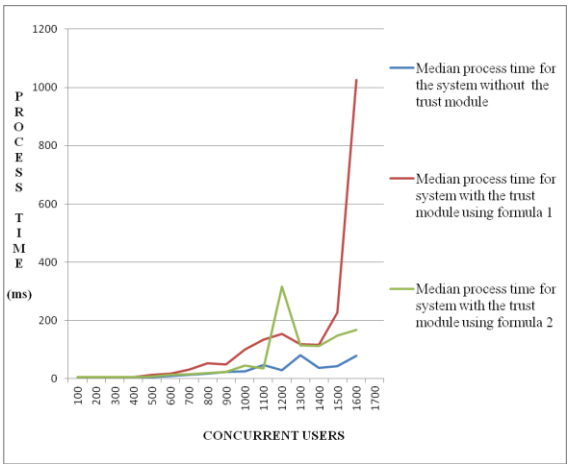

Figure 6-10 Median process time for scenario A (message size 2K)

The results for average process time and median process time for message size 3K are shown in figure 6-11 and figure 6-12.

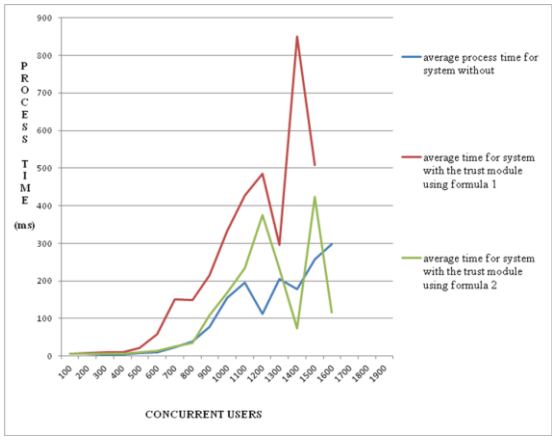

Figure 6-11 Average process time for scenario A (message size 3K)

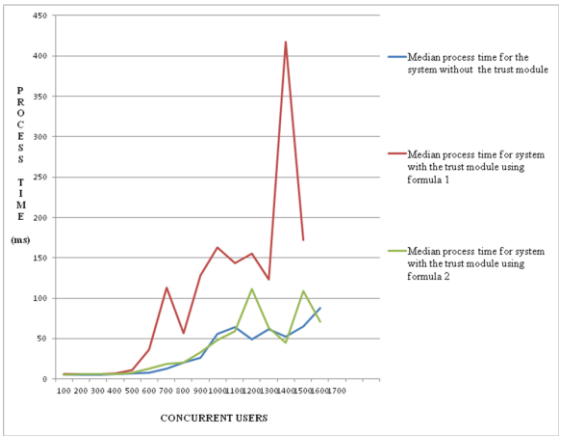

Figure 6-12 Median process time for scenario A (message size 3K)

Maximum capacities for the two different message sizes are summarized as in table 6-12.

Table 6-12 Maximum capacity for scenario A

	System without the trust module	System with the trust module using formula 1	System with the trust module using formula 2
Message size 2K	1600 users/10 sec	1600 users/10 sec	1500 users/10 sec
Message size 3K	1600 users/10 sec	1500 users/10 sec	1600 users/10 sec

The results show the trust module has a very slight impact on the system's scalability under scenario A. However, the trust module using formula 1 takes a noticeably longer time than the no trust module system while the trust module using formula 2 takes a little longer time. The overhead is evaluated and discussed in section 6.2.3.

There are some fluctuations in these figures due to the different network traffic. The duration of these experiments is about 2 hours. During the two hours, there were some fluctuations on how busy the internet was. These factors affect the process speed.

Generally, the process time goes up when there are more concurrent users.

- Scenario B

Two different message sizes of 1K and 3K, and gradually increased concurrent users are tested under JMETER.

JMETER's setting is the same as scenario A.

"xoxo.usask.ca" is set as a proxy server which forwards HTTP requests to "yuting.usask.ca" and sends responses from "yuting.usask.ca" to the clients.

The results for the average process time and the median process time for transfer message size of 1K are shown in figure 6-13 and figure 6-14.

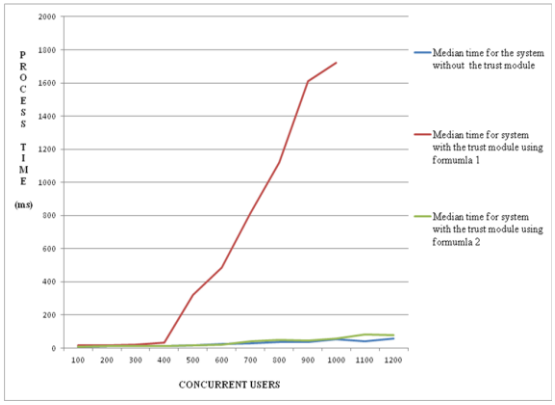

Figure 6-13 Average process time for scenario B (message size 1K)

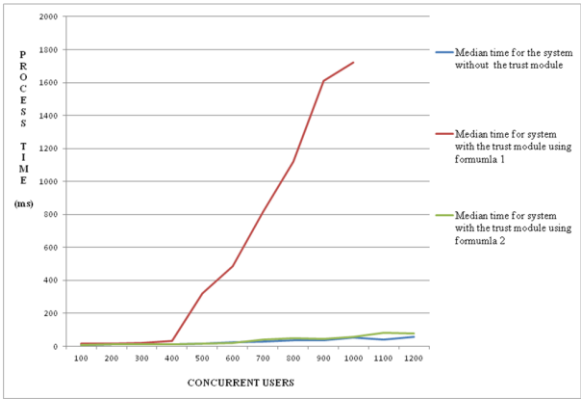

Figure 6-14 Median process time for scenario B (message size 1K)

The results for the average process time and the median process time for message size of 3K are shown in figure 6-15 and figure 6-16.

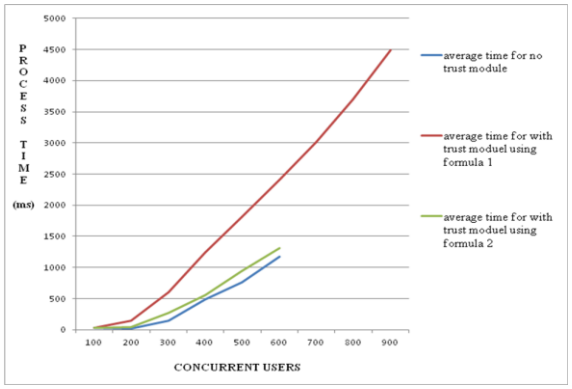

Figure 6-15 Average process time for scenario B (message size 3K)

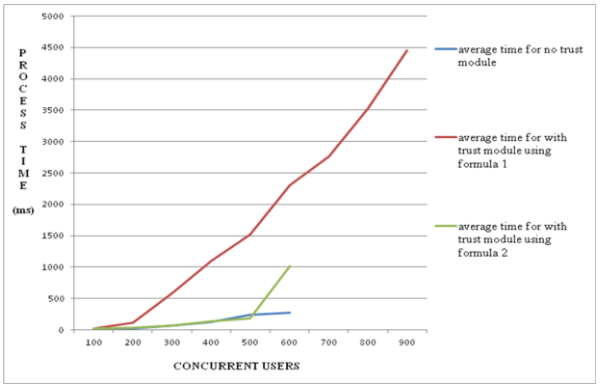

Figure 6-16 Median process time for scenario B (message size 3K)

Table 6-13 Maximum capacity for scenario B

	System without the trust module	System with the trust module using formula 1	System with the trust module using formula 2
Message size 1K	1200 users/10 sec	1000 users/10 sec	1400 users/10 sec
Message size 3K	600 users/10 sec	900 users/10 sec	600 users/10 sec

When the message size is 3K, the scalability for the "system without the trust module" and the "system with the trust module using formula 2" is worse than the "system with the trust module using formula 1", since in this experiment setup, the proxy server "xoxo.usask.ca" which calculates the trust value of the mobile devices is much faster than web server "yuting.usask.ca" and "yuting.usask.ca" cannot handle too many requests sent by "xoxo.usask.ca" while retrieving relatively bigger size data. The trust module using formula 1 takes a longer time to process, so "yuting.usask.ca" can match the speed better. This experiment also demonstrates a balanced system has better scalability.

The process time for scenario B is longer than the process time for scenario A since there are 2 hops in scenario B.

- Scenario C

Two different message sizes, of 1K and 2K, and gradually increased concurrent users are tested under JMETER.

JMETER's setting for scenario C is same as scenario A.

Transfer message size: 1K bytes.

The result for the average process time and the median process time shows in figure 6-17 and figure 6-18.

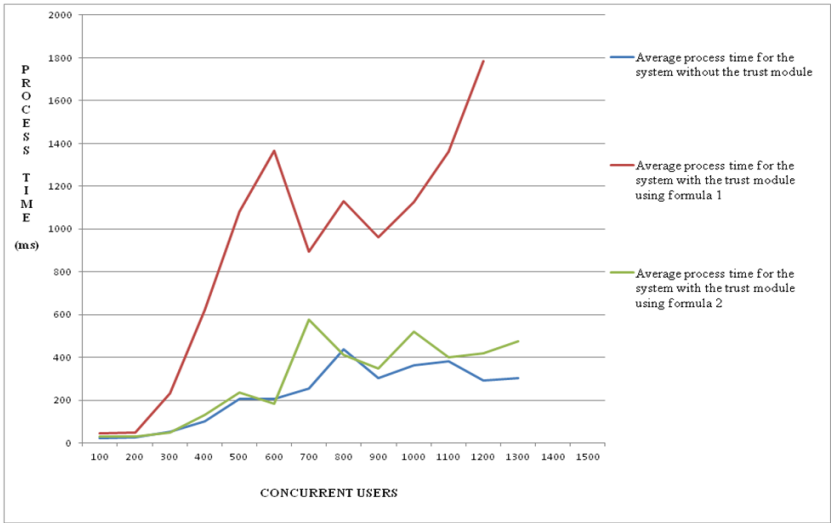

Figure 6-17 Average process time for scenario C (message size 1K)

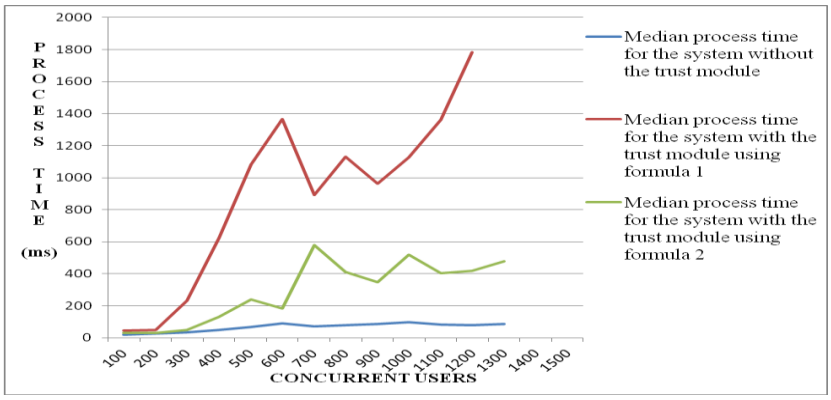

Figure 6-18 Median process time for scenario C (message size 1K)

Transfer size per transaction: 2K bytes.

The result for the average process time and the median process time shows in figure 6-19 and figure 6-20.

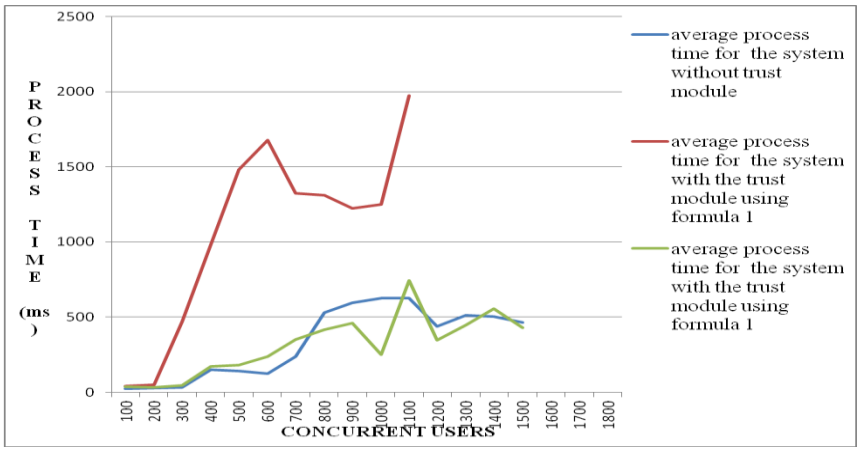

Figure 6-19 Average process time for scenario C (message size 2K)

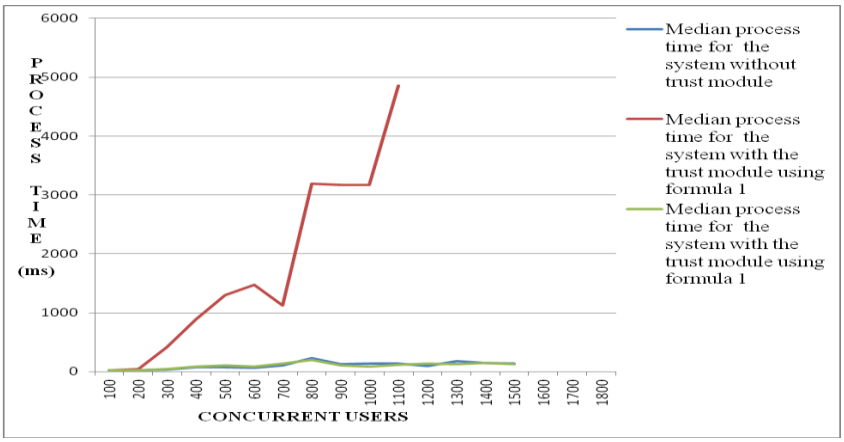

Figure 6-20 Median process time for scenario C (message size 2K)

Table 6-11 Maximum capacity for scenario C

	System without the trust module	**System with the trust module using formula 1**	**System with the trust module using formula 2**
Message size 1K	1300 users/10 sec	1300 users/10 sec	1300 users/10 sec
Message size 2K	1500 users/10 sec	1100 users/10 sec	1500 users/10 sec

Scenario C's scalability is better than that of scenario B because there are two web servers which can handle more requests than one web server.

The scalability for message size 2K is better than message size 1K because in scenario C, the bottleneck is the proxy server "xoxo.usask.ca". It crashes because it fails to connect the other two web servers. The communication frequency is less when the message size is bigger, so "xoxo.usask.ca" can handle more concurrent users when the message size is bigger.

There is a noticeable drop for process time and median time before the system reaches its maximum capacity from the figure 6-17, figure 6-18, and figure 6-19, because when the system is reaching its maximum capacity, some errors occurs (error rate < 10%), so the process time and the median time drops a bit. After that, the error rate gets bigger (>10%) and the system crashes.

- Scenario D

Adding replication web server makes the system more robust and increases scalability. In this experiment, "xoxo.usask.ca" acts as a proxy server and "varna.usask.ca" acts as a replication server for xoxo.usask.ca. "burgas.usask.ca" and "yuting.usask.ca" act as web servers.

The results for message size of 2K are shown in figure 6-21 and figure 6-22.

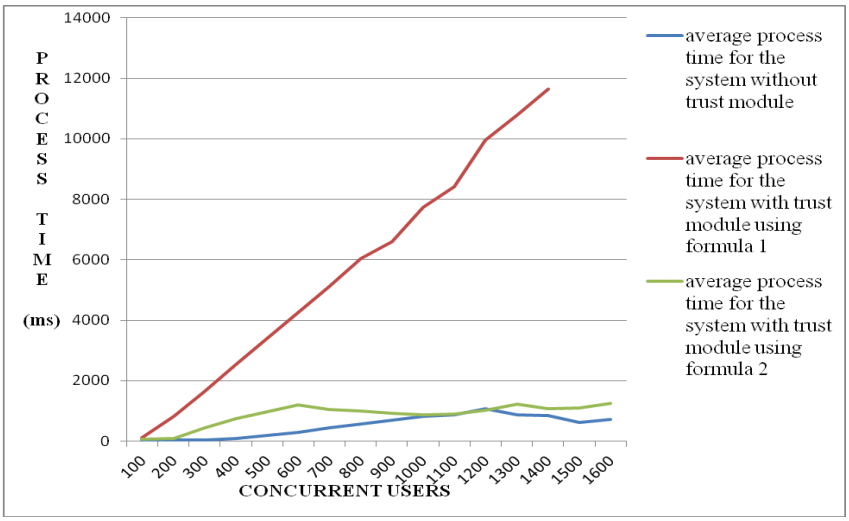

Figure 6-21 Average process time for scenario D (message size 2K)

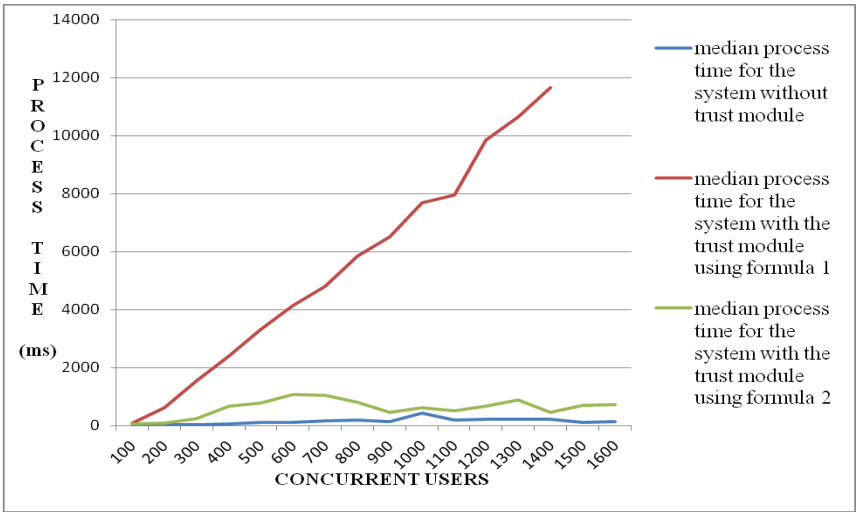

Figure6-22 Median process time for scenario D (message size 2K)

The experiment results for message size of 3K are shown in figure 6-23 and figure 6-24.

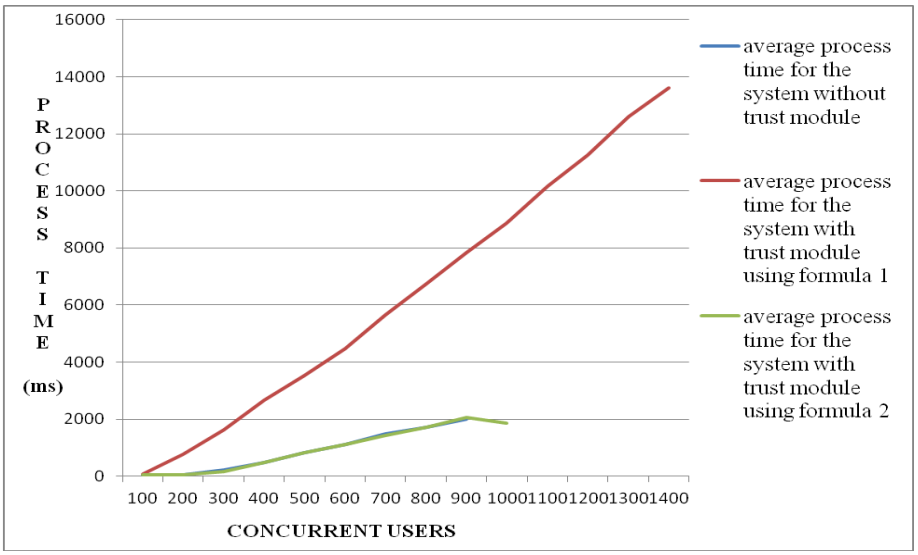

Figure 6-23 Average process time for scenario D (message size 3K)

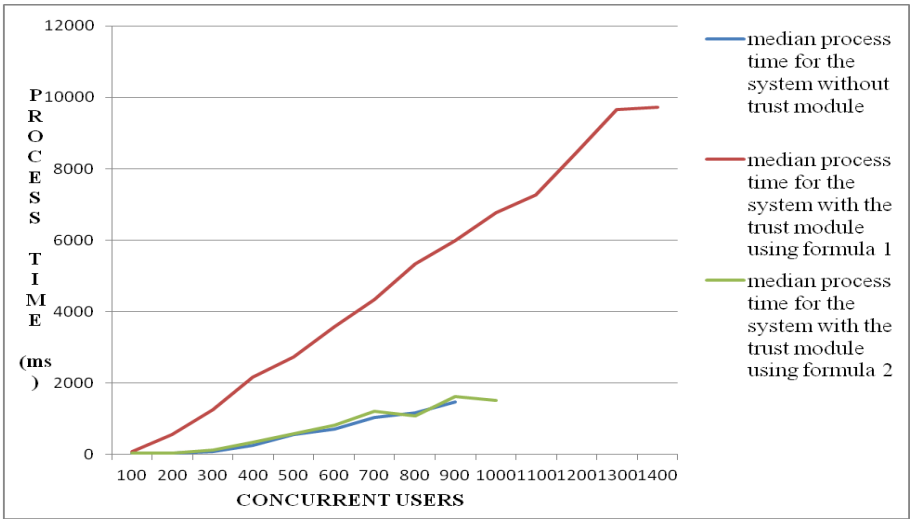

Figure 6-24 Median process time for scenario D (message size 3K)

Table 6-12 Maximum capacity for scenario D

	System without the trust module	System with the trust module using formula 1	System with the trust module using formula 2
Message size 2K	1600 users/10 sec	1400 users/10 sec	1600 users/10 sec
Message size 3K	900 users/10 sec	1400 users/10 sec	900 users/10 sec

This experiments show that adding a replication server improves the system capacity. A load balancer is simulated using JMETER. Since “varna.usask.ca” is not as efficient as “xoxo.usask.ca”. The proportion of number of requests sent to “varna.usask.ca” and “xoxo.usask.ca” is 2:1.

In this experiment for message size 3K, the “system with the trust module using formula 1” shows better scalability. The reason is same as scenario B. With the proxy server “xoxo.usask.ca” and its replication server “varna.usask.ca”, the bottleneck switches to the web server “burgas.usask.ca” and “yuting.usask.ca”. The two web servers cannot handle too many requests while retrieving 3K data.

From the experiments above, the trust module using formula 1 affects system’s scalability to a certain degree, but balancing work load, adding replication server(s) and improving servers’ hardware can help to solve this issue.

6.2.3 Overhead

Overhead for the trust module using formula 1 and formula 2 is tested in this set of experiments.

- Scenario A

JMETER is set as in figure 6-25. In order to minimize the influence of network traffic, the requests to “the system without the trust module”, “the system with the trust module using formula 1” and “the system with the trust module using formula 2” are sent simultaneously.

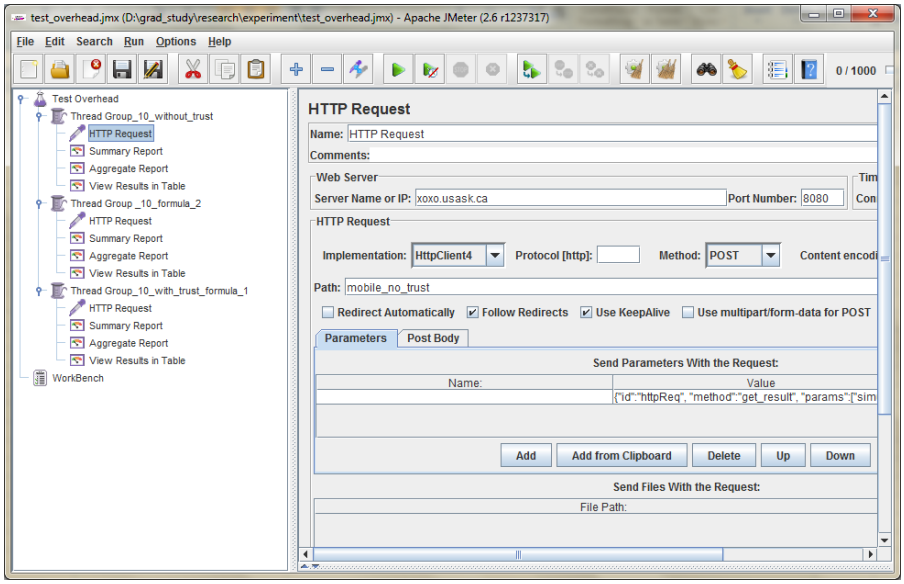

Figure 6-25 JMETER setting for testing overhead

The transfer message size of 300 bytes is tested in this experiment. The result is shown in table 6-13 and figure 6-26.

Table 6-13 Average process time for one web server

Concurrent users	Average process time without the trust module (ms)	Average process time with the trust module using formula 1 (ms)	Average process time with the trust module using formula 2 (ms)	Overhead for the trust module using formula 1	Overhead for the trust module using formula 2
100	5	6	5	20%	0%
200	7	9	7	28.6%	0%
300	18	32	13	77.8%	N/A
400	32	53	44	65.6%	34.4%
500	55	70	53	27.2%	N/A
600	132	283	136	114.4%	3.0%
700	141	184	177	30.5%	25.5%
800	102	320	146	213.7%	43.1%

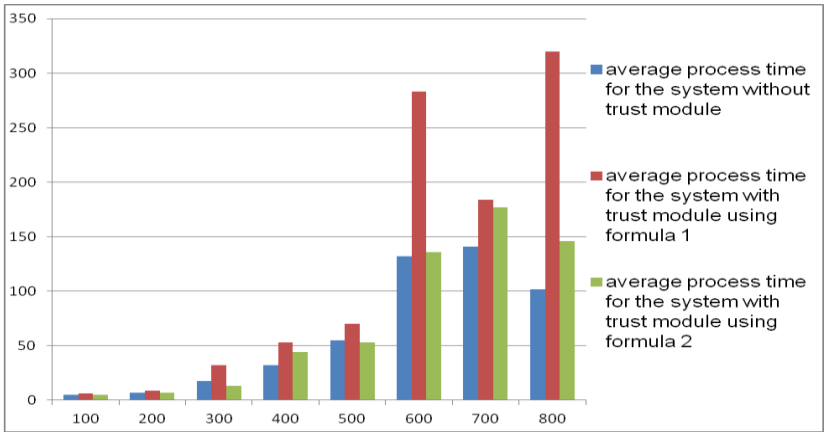

Figure 6-26 Overhead for scenario A

- Scenario B

  The test plan setting is the same as scenario A.

A transfer message size of 1K is tested in this experiment. The result is shown in Table 6-14 and figure 6-27.

Table 6-14 Average process time for one web server and one proxy server

Concurrent users	Average process time without trust module (ms)	Average process time with trust module using formula 1 (ms)	Average process time with trust module using formula 2 (ms)	Overhead for the trust module using formula 1	Overhead for the trust module using formula 2
100	18	25	18	38.9%	0%
200	148	277	167	87.2%	12.8%
300	849	1135	931	33.7%	9.8%
400	675	1118	764	65.6%	13.2%

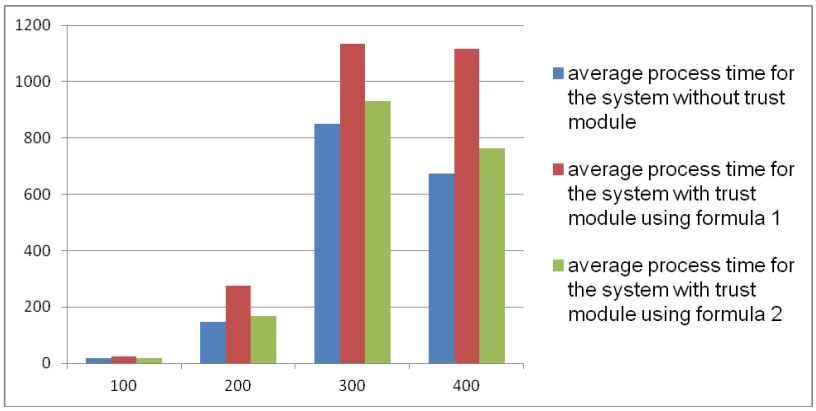

Figure 6-27 Overhead for scenario B

- Scenario C

  The test plan setting is the same as scenario A.

A transfer message size of 1K is tested in this experiment. The result is shown in table 6-15 and figure 6-28.

Table 6-15 Average process time for two web servers and one proxy server

Concurrent users	Average process time without trust module (ms)	Average process time with trust module using formula 1 (ms)	Average process time with trust module using formula 2 (ms)	Overhead for the trust module using formula 1	Overhead for the trust module using formula 2
100	18	25	18	38.9%	0%
200	148	277	167	87.2%	12.8%
300	849	1135	931	33.9%	10.0%

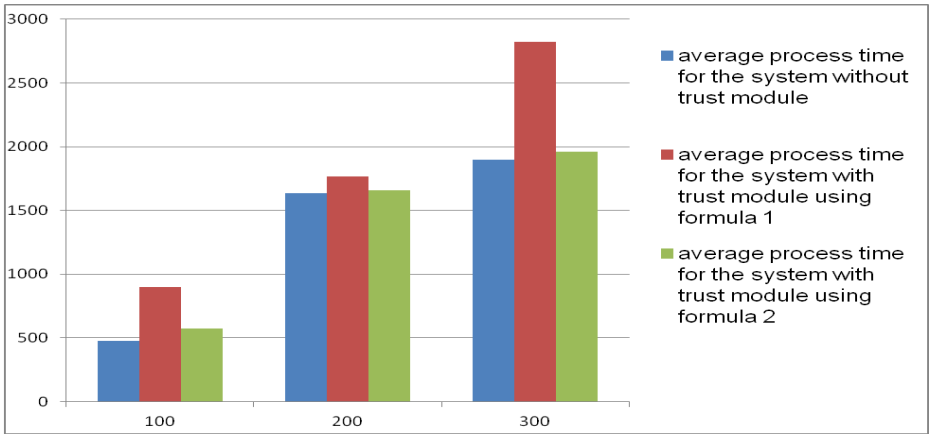

Figure 6-28 Overhead for scenario C

When there are more concurrent users, the overhead for the trust module using either formula 1 or formula 2 is getting bigger. Overhead of the trust module using formula 1 is much more than using the trust module using formula 2 because the trust module using formula 2 calculates trust values only based on current context values while the trust module using formula 1 calculates trust values not only based on current context values but also on history transactions, and the process involves searching history transactions and updating the new transactions.

6.3 Summary

The goals for the experiments are testing the system's functionality, scalability and overhead for the trust module.

Functionalities: the trust module either for using formula 1 or formula 2 works as expected. According to different business requirements, by adjusting trust policies and trust thresholds, the trust module prevents improper requests.

Scalability: Generally, the system can handle hundreds of requests/per second. Normally this is sufficient for small to medium-sized business demands. If the business demand grows, adding replication servers and balancing work load can improve the system's scalability.

Overhead: “The trust module using formula 2” adds little overhead to the system since it only calculates the current trust values based on the context values while “the trust module using formula 1” adds some overhead since it includes searching previous transactions and updating the trust credits for the mobile devices. A suitable trust formula should be chosen based on business requirements. If previous transactions are very important when considering the trust value, then “the trust module using formula 1” should be applied even though it adds more overhead.

Overall, the experiments prove the system can prevent scenarios such as the mobile devices being used by malicious persons or being abused, or the requests sent by hacked systems. The overhead for the trust module is reasonable, especially if history transactions are not considered. The system’s scalability is sufficient for small to medium-sized size business. With the growth in the size of the business, improving hardware and balancing the system, and adding replication servers can improve the system’s scalability.

CHAPTER 7

SUMMARY AND FUTURE WORK

7.1 Summary

As mobile device hardware and mobile applications rapidly grow and enterprises open their web applications for mobile devices, it is getting more and more common to access data either on mobile devices or on web servers. How to trust the requests sent by mobile devices becomes an issue. This research has proposed a distributed trust module which is specific for mobile devices' web services. The trust module calculates mobile devices' requests based on the devices' context values and calculates the requests from other domains based on the trustworthiness of the domains. The system is built in the Erlang language to achieve better performance and scalability.

7.1.1 Problem and solutions

In chapter 2, I describe three scenarios which are 1) lost or stolen devices; 2) abuse of privileges; and 3) traditional access control issues. Normal mobile access control and web access control cannot solve these issues. A distributed trust module is proposed in this research which is built on the top of enterprise web system, adding additional security mainly for the problems mentioned.

1. **Lost or stolen devices**

By calculating the trust value based on mobile owners' operation patterns, mobile owners' normal working routine (schedule, location etc.), most of lost or stolen devices' cannot be used to access protected data.

2. **Abuse of privileges**

Many enterprises open their gates to mobile devices for their employees to improve productivity. Unlike desktops which employees use during office hours, mobile devices can be access at any time and locations; therefore the possibility of abusing the privileges is much higher. For example, an employee can easily make mistakes when they are drunk. With the limitation of sending location type and sending time, or other restrictions, abusing privilege can be reduced.

3. **Tradition access control**

Cross-domain access is always a challenge which involves assigning global roles to users, and considering other domains' security. The trust module which is built on the top of the original security settings, calculates the requests from other domains based on the context values and trustworthiness of the other domains. It protects current domain from other malicious requests.

7.1.2 The system's features

- **Lightweight**

Since the proposed system is also designed to be built on top of the existing system, in order to minimize the impact of the existing system, the system must be lightweight to achieve good performance. The system is written in Erlang which features lightweight process, scalability and distribution, so the system is good at handling multiple concurrent users and easy to expand.

- **Attachable**

As mentioned above, since the trust module can be built for a new system or a legacy system, being attachable is important. It is an independent module which includes a client side component, a server component and a database. Adding a new trust module in an enterprise's legacy system can be seamless.

7.1.3 Novelty

- **Evaluate the requests' trust value**

Most trust systems focus on evaluating servers' or other peers' trust value and reputation, for example, how to choose the best vendor, or exchange files between other peers which have good reputations. The approaches for calculating trust value and reputation are specific for servers. Not much has done on how to evaluate clients' requests. Since the requests from other clients can also be risky, especially for mobile devices due to their nature, there is the need for building a system calculating the trust value of mobile client's requests.

- **Decentralized module**

The trust module is light weight component which can be attached to a distributed system. Every module works independently and also communicates with each other, exchanging trust information of mobile devices and server domains. Different trust policies, trust thresholds and trust history for mobile devices and servers can be set individually for each trust module. Each trust module acts as a small agent to protect sensitive data.

7.2 Future work

The trust module enhances the security for enterprise mobile web services and provides a mechanism to calculate requests' trust value. To make it work better in any new or existing system, additional needs to be done.

7.2.1 More context values should be considered

Context value is essential when calculating the trust value for mobile devices. The more context values, the more accurate to predicate mobile devices' situation. For example: if a mobile device detects any Bluetooth connections around, it most likely can determine its

environment: office or any social club; If it detects certain wireless network, it can determine how risky the network is; A mobile device's sensor can detect the owner's low blood pressure and irregular heart rate, or if any abnormal condition occurs, some tasks which demand certain health condition are prohibited. The mobile devices can also send the owner's health data to a doctor if the device can connect to a health care system. Also, current location only refers to geography information. It is helpful to estimate the risk of the request if more location information like location type can be found and used. If a location type is a "bar" or a "restaurant", then it is less safe than an "office" type for accessing work information.

7.2.2 Explore iPhone, Windows mobile and Blackberry other mobile devices

Current mobile clients' component is implemented for Android tablets. It applies hybrid implementation, a combination of a native application and a pure embedded browser application. The "pure embedded browser" part can be migrated to other mobile clients smoothly, but the "native application" needs some work. In this system, "native application" part is mainly collects context values from sensors of mobile clients. Some open sources provide a platform/framework to develop a variety of mobile device implementations, like iPhone, Blackberry and Android. PHONEGAP is one of these frameworks which uses HTML, CSS and JavaScript to create applications for mobile devices. These tools certainly give some freedom to developers, and it is the trend for mobile application development. For future work, a platform like PHONEGAP can be used to implement more general applications.

7.2.3 Adding exchange data functionality in the trust module

The trust module has mobile device's transactions, so they are considered when calculating trust value if the trust policy involves mobile devices' credit. Each trust module only stores its own transaction history, data can be copied to other trust modules, but it is not timely and

efficient. A function for automatically propagating data to other domains can improve system's robustness and practicality as figure 7-1shows. If a mobile device sends any improper request to a server, this transaction not only is stores in this server, but also is propagated to other servers and other servers' data is propagated into current server. If a malicious person fails to hack a certain sever, the chance he success hack into other servers is low.

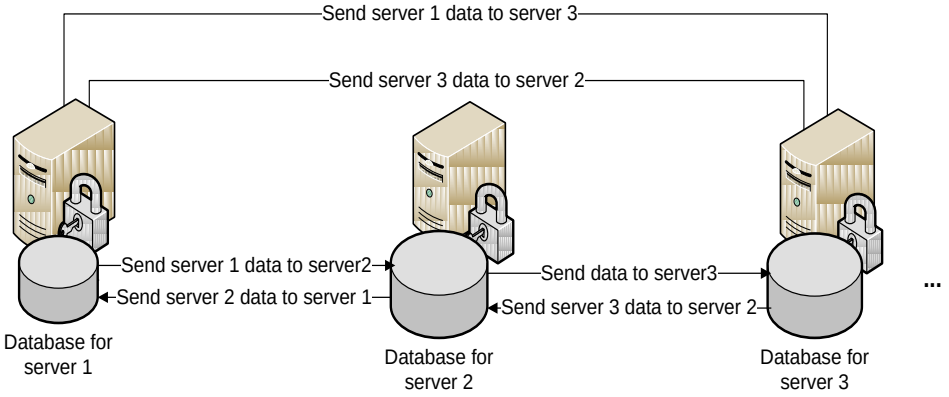

Figure 7-1 Exchange data between the trust modules

7.2.4 Adding more trust formulas suitable for a variety of business requirements

I propose two trust formulas to calculate mobile devices' trust value which are 1) using context values and mobile devices' credit to calculate the trust value, 2) only use context values to calculate trust value, and one formula which is using calculated trust value and trustworthiness of the domain to calculate the trust value of the requests from other domains. Enterprises have different security level for their web access. So these formulas may not fit all security requirements, as some have higher security standards and some have lower security standards. For example, an alternation for calculating the trust value of a request coming from another domain is recalculating trust value based on the context values. This may cause overhead but it

certainly gives better protection. For less secure system, using the calculated trust value may be good enough to satisfy business needs while being more efficient.

7.2.5 Apply the trust module for practical use, such as: health care system, commercial industry

This trust module is proposed and implemented but it has not been put into any real system yet. As mentioned before, the trust module can be used either for new systems or any legacy systems. When the trust module is attached to a legacy system, how to integrate it with the existing system is aan issue. Different operation systems, EBS, web services and databases of the existing systems have different solutions. The trust module can be used in a variety of areas. In the implementation, I use student academic records as an example. These records are stored in different departments and each department has its own rules for accessing these records. This research presents how to use the trust module to control the access. Other areas, like health and personal care system, commercial system can certainly benefit from this module too. Health systems which involve patient's sensitive information are access by only a few people, such as the doctors who are in charge of the patient in certain areas to prevent the data leaks. A trust policy for restricting location, restricting any other wireless connection can be set to ensure data is only access to the doctor in the specific areas. Commercial data could involve commodity prices, purchase orders, sales volume, etc which are only open to trustworthy vendors. Setting rules for selecting vendors so they can view data would be potentially helpful for the business.

LIST OF REFERENCES

[1] S. E. Abdrahman, "Web Access Control Using User Access Behavior (WACUAB)", Proceedings of the 2008 International Conference on Semantic Web & Web Services (SWWS 2008), pages 242-5, 2008.
[2] Access control "http://en.wikipedia.org/wiki/Access_control", last retrieve May 15, 2012.
[3] F. Alshahwan and K. Moessner, "Providing SOAP web services and RESTful web services from Mobile hosts", IEEE fifth international conference on internet and web applications and services, 2010.

[4] L. Bauer, M. A. Schneider, and E. W. Felten, "A Proof-Carrying Authorization System", Proceedings DARPA Information Survivability Conference and Exposition, 117-19 vol.2, 2003; ISBN-10: 0 7695 1897 4; DOI: 10.1109/DISCEX.2003.1194942; Conference: Proceedings DARPA Information Survivability Conference and Exposition, 22-24 April 2003.
[5] Cloud computing "http://en.wikipedia.org/wiki/Cloud_computing", last retrieve May 15, 2012.
[6] M. Coetzee and J. H. P. Eloff, "A Trust and Context Aware Access Control Model for web service conversation", Trust, Privacy and Security in Digital Business. Proceedings 4th International Conference, TrustBus 2007. (Lecture Notes in Computer Science vol. 4657), pages 115-24, 2007.
[7] A. Corradi, R. Montanari, D. Tibaldi, A. Toninelli, and U. Bologna, "A Context-centric Security Middleware for Service Provisioning in Pervasive Computing," Symposium A Quarterly Journal In Modern Foreign Literatures, 2005.
[8] A. K. Dey and G. D. Abowd, "Towards a Better Understanding of Context and Context-Awareness.", Proceeding HUC '99 Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing pages 304 – 307, 1999.
[9] Erlang – Mnesia User's Guide "http://www.erlang.org/doc/apps/mnesia/Mnesia_chap4.html", last retrieve May 15, 2012.
[10] L. Eschenauer, V. D. Gligor, and J. Baras, "On trust establishment in mobile ad-hoc networks", In B. Christianson, B. Crispo, J. A. Malcolm, and M. Roe, editors, 10th International Security Protocols Workshop, Cambridge, UK, April 2002, volume 2845 of Lecture Notes in Computer Science, pages 47-66.Springer-Verlag, 2004.
[11] R. T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures", Doctoral dissertation, University of California, Irvine, 2000.
[12] J. Fischer and R. Majumdar, "A Theory of Role Composition," 2008 IEEE International Conference on Web Services, pages 320-328, Sep. 2008.
[13] Gartner 2008 SOA User Survey. http://www.soabloke.com/2008/11/06/gartner-2008-soa-user-survey/, last retrieve May 15, 2012.
[14] Global mobile statistics 2012 http://mobithinking.com/mobile-marketing-tools/latest-mobile-stats, last retrieve May 15, 2012.
[15] P. Goyal, V. Parmar, R. Rishi, (2011) MANET: Vulnerabilities, Challenges, Attacks, Application, IJCEM International Journal of Computational Engineering & Management, Vol. 11.
[16] S. M. Habib, S. Ries, M. Mühlhäuser, "Cloud Computing Landscape and Research Challenges Regarding Trust and Reputation", uic-atc, pages 410-415,2010 Symposia and Workshops on Ubiquitous, Autonomic and Trusted Computing,2010.

[17] Z. Jiang and S. Kim, “Trust Model for Mobile Devices in Ubiquitous Environment” , Network-Based Information Systems. Proceedings First International Conference, NBiS 2007, pages 426-34, 2007.
[18] Inbound Internet Marketing Blog http://blog.hubspot.com/blog/tabid/6307/bid/30495/25-Eye-Popping-Internet-Marketing-Statistics-for-2012.aspx, last retrieve May 15, 2012.
[19] K. Kanetkar, “A roadmap to building an ESB.”, Integration for everyone, May 22-24, 2006, Boston, MA.
[20] K. M. Lee, K. Hwang, J.-hyong Lee, and H.-joon Kim, “A fuzzy trust model using multiple evaluation criteria”, *Fuzzy Systems and Knowledge Discovery*. Third International Conference, FSKD 2006. Proceedings (Lecture Notes in Artificial Intelligence Vol.4223), pages 961-9, 2006.
[21] S. Marsh, P. Briggs, K. El-khatib, B. Esfandiari, and J. A. Stewart, “Defining and Investigating Device Comfort,” Processing, vol. 19, no. July, pages 231-252, 2011.
[22] mochi / mochiweb https://github.com/mochi/mochiweb/ , last retrieve Jan 31,2012.
[23] H. Nilsson, C. Wikstrom, and E. T. Ab, “Mnesia - An Industrial DBMS with Transactions, Distribution and a Logical Query Language”, International Symposium on Cooperative Database Systems for Advanced Applications. Kyoto Japan, 1996.
[24] X. Pan, “SOA-based enterprise application integration,” 2010 2nd International Conference on Computer Engineering and Technology, pages V7-564-V7-568, 2010.
[25] Dana Pavel, Dirk Trossen, "Context Provisioning for Future Service Environments," iccgi, pages 45, International Multi-Conference on Computing in the Global Information Technology - (ICCGI'06), 2006.
[26] D. Salber, A. K. Dey, and G. D. Abowd, “The Context Toolkit: Aiding the Development of Context-Enabled Applications”, Proc CHI 99Conference on Human Factors in Computer Systems, Pittsburgh, PA, 1999, pages 434-441.

[27] W. N. Schilit, “A System Architecture for Context-Aware Mobile Computing,” Doctoral Dissertation, Columbia University, New York, NY, 1995.
[28] C. Shang, Z. Yang, Q. Liu, and C. Zhao, “A Context Based Dynamic Access Control Model for Web Service,” 2008 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing, pages 339-343, Dec. 2008.
[29] V. Varadharajan and V. Pruthi, “Trust Enhanced Security for Mobile Agents,” Seventh IEEE International Conference on E-Commerce Technology (CEC’05), pages 231-238, 2005.
[30] Q. A. Wang, “Mobile Cloud Computing,” Thesis for the degree of master of computer science, university of Saskatchewan 2010.
[31] Web services “http://en.wikipedia.org/wiki/Web_service”, last retrieve May 15, 2012.
[32] Y. Wang and J. Vassileva, “Bayesian Network Trust Model in Peer-to-Peer Networks”, Agents and Peer-to-Peer Computing. Second International Workshop, AP2PC 2003. Revised and Invited Papers (Lecture Notes in Artificial Intelligence Vol.2872), pages 23-34, 2004.
[33] M. Weiser, “The Computer for the 21st Century” Scientific American, 1991.
[34] X Wu, (2010) A Distributed Trust Model for Mobile Computing Environments Intelligent Computing and Intelligent Systems (ICIS), 2010 IEEE International Conference on pages 248 – 252.
[35] Z. Yan, P. Zhan, “A Trust Management System in Mobile Enterprise Networking”, WSEAS Transactions on Communications, Issue 5, Vol. 5, pages 854-861, 2006.