ANNALS OF THE FACULTY OF ENGINEERING HUNEDOARA
2006, Tome IV, Fascicole 1, (ISSN 1584 – 2665)
FACULTY OF ENGINEERING HUNEDOARA,
5, REVOLUTIEI, 331128, HUNEDOARA

A FRAMEWORK FOR A MOBILE VIRTUAL TERMINAL

[1]BALAN Mihai, [2]MUSCALAGIU Ionel, [1]BALAN Mirela-Ramona

[1]DENMARK TECHNICAL UNIVERSITY,
INFORMATICS AND MATHEMATICAL MODELLING,
Lyngby, Denmark

[2]UNIVERSITY "POLITEHNICA" OF TIMISOARA,
FACULTY OF ENGINEERING HUNEDOARA,
HUNEDOARA, ROMANIA

ABSTRACT
A Framework for a Virtual Terminal is developed for CLDC 1.1/MIDP 2.0 enabled mobile devices. Different commands are issued from the mobile virtual terminal and executed on a remote. A response is sent back to the virtual terminal and displayed to the user. All operations are performed in a secure environment based on secure authentication mechanisms and data encryption techniques. A high quality–of-service is ensured by using the iBUS JMS system.

Keywords: virtual terminal, GPRS, GSM, Java 2ME, CLDC 1.1/MIDP 2.0, mobile device, servlet, server, JMS, iBUS.

1. INTRODUCTION

This paper describes the solution of a **Framework for a Mobile Virtual Terminal** for CLDC 1.1/MIDP 2.0 enabled mobile devices. Different commands can be issued from the mobile virtual terminal and executed on a server (service provider) running on the Internet. A response is sent back to the virtual terminal and displayed to the user. Security issues are taken into consideration such as: authentication mechanisms and data encryption techniques. Other related applications can be built on top of this proposed framework with minimum of effort.

2. ANALYSIS OF THE SYSTEM

This section discusses different choices taken during the analysis of the Mobile Virtual Terminal Framework. It depicts identified issues and proposes different solutions to the problem and argues the chosen ones in order to ensure a high-security and reliability of the framework.

2.1. IDENTIFIED ISSUES IN CASE OF THE FRAMEWORK FOR A MOBILE VIRTUAL TERMINAL

Several issues are identified in case of the mobile virtual terminal framework, such as:

- **Mobile Device Limitations**: limited device hardware i.e. small memory, low CPU power, short battery life, etc.
- **Security Concerns**: What kinds of threats one can encounter in such a system? How to secure the communication between the virtual terminal and the service provider? How to authenticate a user? How to send data over the air in a secure way? How to protect the MIDlet against piracy?
- **On-device data storing**: How to store user preferences and service provider configuration parameters in the mobile device? How to protect the on-device data?
- **Mobile Virtual Terminal – Service Provider Communication**: How to realize the communication between the virtual terminal (VT) and the service provider (SP)? How to implement error handling, and quality-of-service? How to guarantee message delivery to the SP and notify the sender? How to achieve a secure communication?
- **Service Provider - Mobile Virtual Terminal Communication**: How to notify the user if an unexpected error occurs at the SP side? How to send the result of a command execution back to VT? How to achieve a secure communication?
- **Mobile Virtual Terminal Configuration**: How to configure the VT from the SP side? How to synchronize the configuration parameters of the VT with the ones stored on the SP side? How to perform the configuration operations in a secure way? How to ensure the commands sent to the SP are error-free before sending them to the network?
- **Slow and Unreliable Networks**: How to overcome the issue of slow and unreliable networks?
- **Virtual terminal GUI**: How to design the VT UI for a CLDC 1.1/MIDP 2.0 based mobile device? How to embed rich UI in the application? How to prevent UI lock-up during the network operations? How to make the information easy to read on such small screens?

2.2. SOLUTIONS TO THE IDENTIFIED ISSUES IN CASE OF THE FRAMEWORK FOR A MOBILE VIRTUAL TERMINAL

Mobile Device Limitations - The solution of the Mobile Virtual Terminal needs to overcome all previous mentioned hardware limitations. Due to the application requirement i.e. "CLDC 1.1/MIDP 2.0 devices" these limitations can be overcome by using lightweight libraries, obfuscation, minimizing object creation and disposing any unused objects, using design patterns, reusing objects rather then creating new ones, closing network connections and record management system after use, etc.

On-device data storing - CLDC 1.1/MIDP 2.0 enabled devices do not have one of the advantages regular PCs have i.e. conventional file system. This can be overcome by using the **Record Management System (RMS)** that gives MIDP applications local, on-device data persistence. [1] The data to be stored is placed in RMS as a Record. A record can contain anything that a sequence of bytes can represent. [1] A record store is a collection of records, each of them with a unique ID. Each RMS is uniquely identified for an application and can be configured to be accessed only by a particular application

From the security point of view, a stolen device containing sensitive data, keys or user credentials can pose a security risk to the whole system. Use of strong encryption with RMS is a must to protect the data.

Mobile Virtual Terminal – Service Provider Communication - The communication between the VT and the SP can be done by using several approaches e.g. Socket connection, or HTTP connection.

The first solution involves mobile devices that can establish socket connections over carriers e.g. GPRS. This implies devices supporting socket connections and an agreement with a GSM network provider in order to allow socket connections for the device - very difficult to achieve. Moreover, the solution should work on most of the devices and not only on a the ones that allow socket connection

The second approach i.e. HTTP connections can be used with all GPRS enabled phones and it does not need any special agreement with the network provider. This is also the chosen solution for the VT.

In order to deal with error handling and ensure a high quality-of-service, the HTTP based solution need to be extended with a middleware service between the VT and the SP. This middleware service can be a server that receives the request from the VT, processes the request, takes care of any security requirements, and sends the message further on to the SP. Due to the HTTP based solution, the chosen middleware is a **Servlet** running on a remote machine.

In order to achieve a secure communication between the **VT**, **Servlet** and **SP**, the middleware is split into two servlets i.e **Authentication Servlet (AS)** used for security purposes, and **Control Service Server (CSS)** that acts as a service provider servlet for sending messages further on to the SP. These two servlets communicates with each other for an enhanced security solution.

How to guarantee message delivery from the CSS to the SP and notify the sender? What happens if the service provider is not available for a period of time but the job need to be performed as soon as the service provider is online? These are just a few quality-of-service issues that require an extension of the VT - SP communication solution.

A possible extension of the solution consists in using some kind of application that manages message queues. The CSS can forward messages received from the VT to a queue. The manager of the queue will be responsible for sending the message further on to the SP. The SP can also post messages to the queue and they will be delivered to the CSS and further on to the VT. This solution can be implemented either by using the EJB technology or the **Java Messaging Service (JMS)**. JMS can manage queues, topics, send or receive messages, etc. A JMS solution is more reliable, scalable, and flexible. Therefore, a JMS solution is chosen - in particular the **iBUS//Mobile JMS** solution from softwired inc.. [2, 3]

The chosen solution to VT - SP communication consists in using a GPRS carrier to send HTTP requests to two HTTP Servlets i.e. AS and CSS. In case user is authenticated, the CSS sends the command received from the VT to the iBUS JMS system. The JMS systems forwards the command to the SP when the SP is online. The command is executed by the SP and a response is sent back to the VT on the same path.

Service Provider - Mobile Virtual Terminal Communication - The VT - SP communication is based on the HTTP protocol. This is a stateless protocol and it needs a client to initiate a connection to a server - not the other way around.

One solution consists in using the VT to pull the SP at different periods of time and get an answer from the service provider. This involves to pay for all requests to the SP even the SP has not changed its state. This is not an economical solution.

A better approach implies the SP to send messages to the VT when a change of its state occurs - **SP initiate the communication** by using either the JMS technology or the **SMS based Push Technology** from the SP to the VT. [5]

Mobile Virtual Terminal Configuration - The VT configuration need to be realized from the SP side. When a new job is assigned to the SP, the manager application of the SP must send a configuration message to the VT. The information embedded in the message can be used to configure the VT for sending error-free commands to the SP before sending to the network. Accordingly to the SP - VT communication

described above, the **Push Technology** can be used for this purpose. Two approaches can be used: the SMS based solution, or the JMS.

Slow and Unreliable Networks - The solution to this issue consists in using the on-device data techniques by means of RMS to avoid slow connections for obtaining the configuration parameters all the time a command is to be sent, or for user credentials and session keys. Reading/Writing network data need to be done using a buffer mechanism because reading/writing data byte by byte is very slow. [4] To ensure a high quality-of-service level - the command is actually send to the SP – the iBUS JMS system can be used.

Virtual terminal GUI - Rich UI can be developed by taking advantage of the advanced UI components in CLDC 1.1/MIDP 2.0. One must prevent the UI lock-up during network operations by using background threads. An animated gauge can be displayed to keep user informed on the status of the operation. A cancel button should be present in case user wants to cancel the operation at any time. Due to the limited size of the mobile device display, a "One screen at a Time" approach needs to be considered. Long operations need to be split into small pieces. [4]

2.3. SECURITY CONCERNS IN CASE OF THE MOBILE VIRTUAL TERMINAL FRAMEWORK

Security may be divided into the security of the communication and physical security of the device. In today's world one additional thing is added to the classic requirements: protection against piracy. The following security concerns are considered: **data integrity, confidentiality, availability, and non-repudiation**.

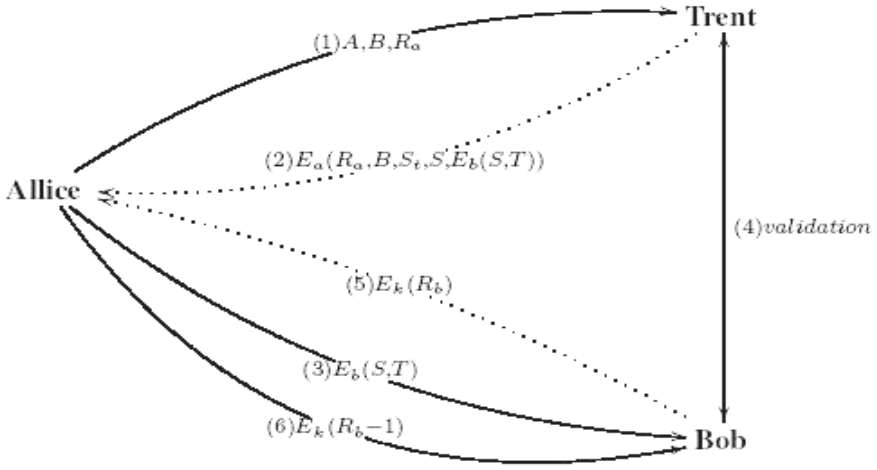

Figure 1. AUTHENTICATION WITH THE USE OF SINGLE SIGN ON SERVER AND NEEDHAM - SCHROEDER PROTOCOL

In order to make the system secure, it is necessary to design a security protocol. This is built on the concept of **Single Sign On Server** and the **Needham-Schroeder Protocol**. [6] There are three parties involved in the protocols: **Trent** (authentication server, trusted arbitrator in the protocol), **Alice** (user connecting from the mobile terminal), and **Bob** (service provider). The diagram of the protocol is displayed in figure 1, where **A** is the user credentials, **B** - the address of the targeted system, $\mathbf{R_a}$ - a random number, **S** - a random salt value, **T** - a token, $\mathbf{E_b}$ - a key preshared with Bob, $\mathbf{S_t}$ - the serial number of Allice, and $\mathbf{R_b}$ - a random number generated by Bob.

The protocol is not vulnerable to the replay attacks during the communication with Trent. Moreover the inclusion of the random number in the message makes it practically impossible to decrypt, even if the preshared key used at this stage is weak. All three factors of the authentication must be compromised in order to compromise the system. The session key is used only for one communication session.

This is enforced by checking the random password from the token with Trent. The session key is 128 bit key. It is generated from **securely random salt value** and a serial number of Allice by using the **AES** algorithm. All messages involved in the communication between Allice and Bob contain a timestamp. Sufficient approximation of this timestamp should be a serial number. It prevents the replay attacks. Authentication of each message by Bob is no longer necessary. Possession of the session key by Allice is sufficient proof of her authenticity.

2.4. FINAL PROPOSED SOLUTION FOR THE MOBILE VIRTUAL TERMINAL FRAMEWORK

Considering the previous formulated requirements, the final solution to the Mobile Virtual Terminal can be stated. The architecture of the chosen solution is depicted in figure 2.The proposed solution has five components:

- **The Virtual Terminal (VT)** - running on a CLDC 1.1/MIDP 2.0 enabled device. It is used for authentication purposes and sending commands from the mobile device to the remote SP.
- **The Authentication Servlet (AS)** - used for authentication and key exchange purposes.
- **The Control Service Server (CSS)** - acts as a service provider. It receives requests sent from the VT, checks if the user is authenticated, processes the requests and sends the processed request to a JMS queue.
- **The iBUS JMS system (server and gateway)** - is the manger of a queue where the CSS sends messages to, and the SP receives from. It ensures a high quality-of-service level and it guarantees message delivery to the SP and notification to the sender.
- **The Service Provider (SP)** - actually the manager application that controls the SP delegates a component that listens to the JMS queue where messages are sent by the CSS. The SP registers with the JMS system. Once a message is received in the queue, it is automatically send to the SP. Once received, the command is analyzed, and executed. A notification is sent back to the user. Configuration parameters can also be sent from the SP side to the VT.

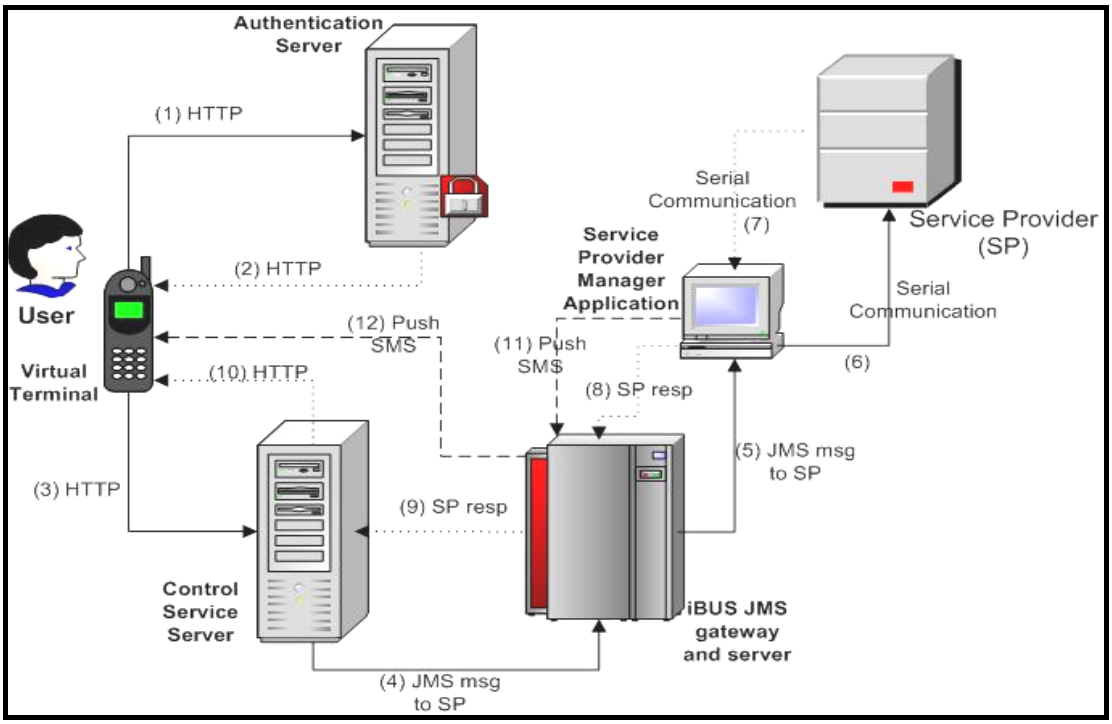

Figure 2. THE DIAGRAM OF THE MOBILE VIRTUAL TERMINAL FRAMEWORK

The VT is built for CLDC 1.1/MIDP 2.0 enabled devices. Therefore, the code has to conform to all previous mentioned constraints. Information e.g. user credentials, keys, configuration parameters, etc are stored in the RMS of the mobile device. The communication between the VT and the SP is performed by using two middleware servlets i.e. **Authentication Servlet** for authentication purposes, and **Control Service Server** as a service provider server that sends the commands received from the VT further on. The CSS is connected to the iBUS JMS server and gateway. Each message received from the VT and sent by an authenticated user, is processed and sent forward to a JMS queue where the SP listens. Once a message is in the queue, the SP receives the message by registering to the JMS system as a consumer for that queue. In case the SP is offline, the message is sent when the connection is reestablished. The **Push Technology** is used to realize the communication between the SP and the VT - the SP initiates the communication. The VT configuration is done by using the **SMS based Push Technology** or the **iBUS JMS** system. Slow and unreliable networks can be overcome by using RMS for the on-device data storage, buffered read/write operation to the network data, and the iBUS JMS system to ensure a high quality-of-service level of communication. The VT GUI takes advantage of the rich capabilities provided by CLDC 1.1/MIDP 2.0. It uses background threads for network operations to prevent UI lock-up. Long UI operations are split in small pieces.

3. CONCLUSIONS

This paper described the architecture of a framework for a secure mobile virtual terminal. The architecture proposed in this paper is a step in this direction. Security issues in case of mobile applications are taken into consideration and dealt with them. All data states are cryptographically protected. Keys are exchanged between the VT and the servers. Authentication based on the Single Sign On server with the use of Needham-Schroeder protocol is used.

Future improvements include more complex control abilities and more robust feedback solutions. The iBUS JMS system can be replaced by the EJB technology. An open-source gateway can be used and an agreement with a GSM network provider can be realized instead of the paid service currently used. Thus, the ESMS based Push Technology can be easy implemented.

REFERENCES

[1.] Giguere, E.: Databases and MIDP, Part 1: Understanding the Record Management System. Sun Microsystems - Technical Article and Tips, February 2004.
[2.] Grant, S., Kovacs, M., Maffeis, M., Morrison, S., Raj, G., Giotta P.: Professional JMS, Wrox Press, March 2001.
[3.] www.softwired.com
[4.] Yuan J., M.: Enterprise J2ME: Developing Mobile Java Applications. Prentice Hall, October 2003.
[5.] Muchow, J.: Core J2ME Technology, Prentice Hall, December 2001
[6.] Schneier, B.: Applied Cryptography, Second Edition: Protocols, Algorthms, and Source Code in C, Wiley Computer Publishing, John Wiley & Sons, Inc., January 1996