Direct Mobile data synchronization between servers

Jugeon Pak, SungHyun Im, Keehyun Park
Keimyung University
School of Computer Engineering Daegu
Korea
{corea, burningwing, khp}@kmu.kr

ABSTRACT: *In mobile environment, the Mobile data synchronization (DS) is a crucial activity and operation because of its extended application in multi-server environment. The current Mobile data synchronization protocols fail to work in multi-server environment. The problem of slow synchronization mode thus need to be eliminated to ensure functioning of the data synchronization. Thus we propose to directly synchoronize the data between servers by using multi-agent system which is developed on JADE platform. The experimental results obtained by us prove that the DS operations take less than 9 seconds while slow sync operations take more than 99 seconds tosynchronize 300 data records. We document in our study that the inter-server DS system is effective than the current DS system for multi-server environments.*

Keywords: Mobile data synchronization, Inter-server data synchronization, Multi-servers, JAVA Agent development platform

Received: 13 December 2009, Revised 1 March 2010, Accepted 4 March 2010

1. Introduction

Nowadays, people have more than one computing devices to manage their data such as e-mail, contact information, and other critical data. And they want to access their data, anywhere at home, at work, or even on the road. For this, each device holds one replica of the user's data. In this case, changes of replicated data may be made at anywhere. Even if changes are made on the different devices or in different places, the changes should be reflected on each device. For this reason, *Data Synchronization* (DS) is required. DS propagates the changes on one device to the others. So propagating only the changed data (known as *fast sync* mode) rather than propagating the entire data (known as *slow sync* mode) is one of the key issue to reduce network traffic.

Currently there are several DS protocols such as Palm's HotSync [12], Intellisync [7], and OMA DS [11]. These protocols provide two modes: fast sync and slow sync. Synchronization is usually established in the fast sync mode. Only the case of not meeting the fast sync conditions, a slow sync is performed. In the fast sync mode, devices transfer only the changed data in their database so it is relatively efficient. In the slow sync mode, however, devices transfer the whole data in their database no matter how much data has been changed since the last synchronization. It can be a heavy burden to network and devices, especially mobile devices.

However, theses protocols are often working inefficiently especially in multi-server environments. Consider a DS scenario in Figure 1. When a change is made on data in the mobile device, the user synchronizes it with the computer (A) at home (1st Sync). When he arrives at work later, he synchronizes the mobile device with the computer (B), too (2nd Sync). Some changes also might be made at work. Then he synchronizes the mobile device with computer (A) at home again (3rd Sync). A critical problem occurs in this scenario which slow sync is established during every synchronization process except 1st Sync.

Several solutions have been proposed for this problem. CPISync [14, 13] is a Characteristic Polynomial Interpolation-based synchronization solution. CPISync can reduce the amount of transferred data even in slow sync mode but it is very complex and difficult to implement. In addition, it can be used for improving performance of the slow sync, not substituting for it.

Enkel DS [2] is a solution based on Vector Time Pairs and reuse the current OMA DS protocol. In Enkel Ds, GUID (Global Unique ID) of each record represents the specific value c which contains the replica on with the data is inserted and the timestamp of insertion. CPISync and Enkel DS address the multi-server issue but they require additional efforts for mobile devices.

In this paper, a new synchronization system based on the previous study on Embedded Mobile DS Gateway System [9, 10] is proposed in order to solve the multi-server issue. It doesn't need slow sync operations in multi-server environments. It does not require additional efforts for mobile devices. The main idea of the proposed system is to synchronize data between servers (inter-server synchronization) by a multi-agent system (MAS) application without mobile devices' intervention. If a mobile device synchronizes with one server, then the server synchronizes the other servers automatically. To achieve this, a multi-agent system based on JADE (Java Agent Development platform: the standardized Agent platform) [8] is constructed. And the DS servers are synchronized each other by multi-agents at regular intervals. It is clear that the inter-server DS system proposed in this paper performs much better than the existing DS systems in multi-server environments and it also provides higher scalability in mobile synchronization environments.

The remainder of this paper is organized as follows. Section 2 discusses related works. Section 3 provides the details of the inter-server synchronization and the multi-agent application. Thereafter, in Section 4 experimental results are illustrated. Finally, conclusion of the paper is discussed in Section 5.

2. Related work

2.1. Past synchronization protocols

HotSync [12] is a proprietary DS protocol of Palm Co., and it handles personal data as a record in a database. It has an additional field known as flag. HotSync provides three values of the flag: modified, inserted, and deleted. The flag is set differently according to the changes which are introduced to a record. If a mobile device synchronizes with the same desktop computer, then it uses the fast sync mode. If the mobile device synchronizes with more than one desktop computer, or a new desktop computer as in Figure 1, then it uses the slow sync mode. In Figure 1, the mobile device has four data records and the records have a flag field. During 1st Sync, the mobile device sends records ***B***, ***C*** and ***D*** to the computer (A) at home (the fast sync mode). After the synchronization, the flag of each record is reset. Later, the user tries to synchronize his mobile device with the computer (B) at work. At this time, all of data records are sent to the computer because flags have been reset already [1, 6].

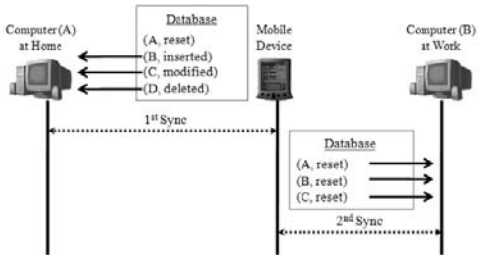

Figure 1. Synchronization with two computers

Intellisync [7] and OMA DS [11, 15] are centralized synchronization protocols. They consist of several devices (clients) and a centralized server called a DS Server. All of the clients should synchronize with the DS Server. If a client tries to synchronize with other clients directly, then a cyclic architecture is generated. In the case of cyclic architecture, Intellisync and OMA DS can't work properly due to anchor mismatch. Neither Intellisync nor OMA DS gives recommendation on how to handle such a case, and they don't support this architecture. And also the problems of scalability and the bottleneck on the DS server arise, as most of the centralized systems suffer.

2.2 Embedded mobile DS gateway system

An embedded mobile data synchronization system which conforms to OMA DS protocol is constructed in our previous studies [9, 10]. Figure 2 depicts the architecture of the system. It consists of mobile clients with embedded gateways and the centralized DS server. It converts personal data on a mobile device into common data specified by OMA DS protocol and vice versa. And it synchronizes data in accordance with the procedures of OMA DS. The DS Server is more complicated than the embedded gateway. Because it has responsibilities to resolve data conflict and determine a sync mode. The Embedded Mobile DS gateway system also has a limitation that can't be adopted in the multi-server.

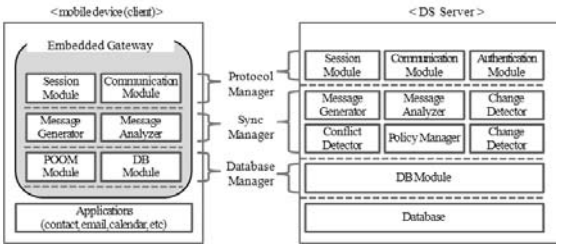

Figure 2. Embedded mobile DS gateway system

2.3 Java agent development environment

Java Agent Development Environment (JADE) is a JAVA framework in compliance with the FIPA (Foundation of Intelligent Physical Agents) specifications [4]. The architecture of a JADE platform is represented in Figure 3.

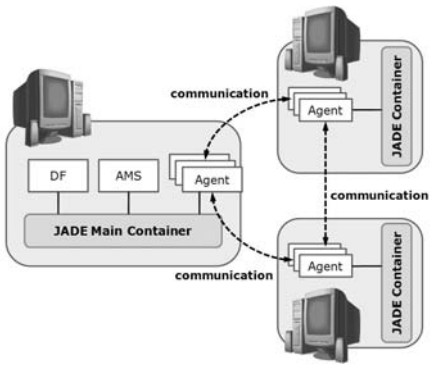

Figure 3. JADE Platform Architecture

- AMS (Agent management System): the agent who exerts supervisory control over access to and use of the Agent Platform. It maintains a directory of agent identifiers (AID) and agent state.
- DF (Directory Facilitator): the agent who provides a yellow page service used when an agent finds other agents.
- Container: a running instance of the JADE runtime environment. It can contain several agents. All other containers register with a main container. Therefore, all containers should know where to find their main container.

Under JADE platform, an agent can find other agents (i.e. a DS server can find other DS servers) and communicate each other (i.e. a DS server is able to send some information to other servers) [3].

3. Inter-server data synchronization

3.1 Peer-to-peer network topology

To achieve inter-server synchronization, the network topology of the servers is changed into a peer-to-peer as shown in Figure 4. There are six computing devices: one mobile device, two computers at home and three computers at work. Five computers (or servers) are fully connected. That is, they can communicate and synchronize each other. Notice that connections are pure logical connections and not permanent network links. Let us consider the scenario as follows:

Step 1) the user synchronizes his mobile device with the computer (C) at work since (C) is the nearest one or the most favorite one (left side).

Step 2) the computer (C) is the one who has the newest data of mobile device, so it synchronizes with the computers (A), (B), (D), and (E) (right side).

Step 3) later, the user comes back his home and synchronizes the mobile device with the computer (A) at home (right side).

The slow sync doesn't occur in this case since all computers are fully synchronized each other. Inter-server synchronization is based on a peer-to-peer network topology and therefore all servers can have the same anchor and data as those of the mobile device, consequently, synchronizations are performed properly even in the multi-server case.

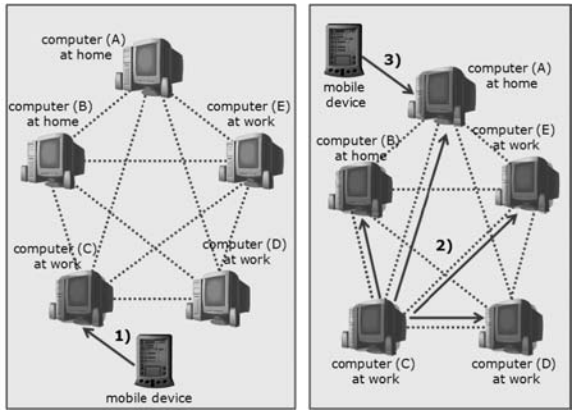

Figure 4. Peer-to-Peer Network Topology for Inter-Server Synchronization

3.2 Multi-agent application

There are four important considerations in order to achieve an inter-server synchronization besides the peer-to-peer network topology.

- Servers (or computers) should be able to communicate and synchronize each other even their location is different.
- If a new server is added in the network, the server should be participated as quickly as possible.
- If a server leaves the network, it should be informed as quickly as possible.
- It should be free from scalability issue: Adding a new server must not lead to additional efforts.

A multi-agent application is designed and implemented to meet these considerations. A multi-agent application has several agents and each agent performs specific tasks (e.g. searching agents, managing agents, communicating and so on). The application in this paper is implemented based on JADE framework, since JADE has some advantages as follows:

- JADE is an open source framework, and written in JAVA language and FIPA-compliant.
- In JADE framework, all agent tasks are modeled as Behaviours such as OneShotBehaviour, TickerBehaviour and CyclicBehaviour [8]. Those Behaviours simplifies complex tasks.

Figure 5 illustrates the inter-serversynchronization system architecture. The inter-server synchronization system consists of three parts: DS servers, mobile devices (clients) and JADE multi- agent application. DS servers and clients are presented in Section 2.2.

The multi-agent applications are placed on each computer. A multi-agent application has a catalog and four agents: AMS, DF, SQL Agent and Comm Agent. AMS and DF are described in Section 2.3

- Anchor Catalog: it is a kind of data store. It maintains device IDs and anchor information as a table. If the ID of a certain mobile device, for example, is ***'MyPDA'*** and the device synchronized at ***4*** lastly, it is simply represented as (MyPDA, 4).

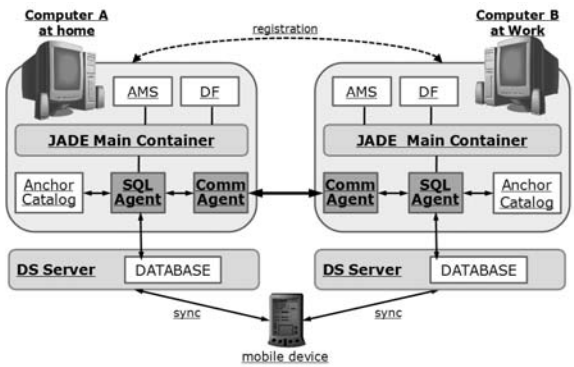

Figure 5. Inter-server Synchronization System Architecture

- SQL Agent: it manipulates the database of a DS server using SQL queries.
- Comm Agent: it takes charge of communication with other servers. Anchor information and data supposed to be synchronized is transferred from a server to a server by this agent.

The mechanism of inter-server synchronization using an example is described in Figure 6. In the example, a user has one mobile device and two computers at home and at work. The ID of the mobile device is ***'MyPDA'***. It has been synchronized with the computer (A) before at ***2***.

Step 1) the mobile device inserts data ***B*** and ***C*** at ***3***.

Step 2) it synchronizes with the computer (A) at ***4***.

Step 3) after the synchronization, the status of data ***B*** and ***C*** is set as ***'synced'***, and the timestamp is changed into ***'4'*** in both devices, and the anchor of mobile device is updated to ***'4'***.

Up to this point, the steps described so far are the same as those of the embedded DS gateway system in Section 2.2. But the next steps are characteristics of the multi-agent application.

Step 4) the SQL Agent running on the computer (A) sends the database a query to get the anchor information, and updates the information on the catalog. As a result, the anchor for ***'MyPDA'*** is updated to ***'4' '*** in computer A's catalog.

Step5) the Comm Agents of both computers request anchor information each other. Generally they do this work at regular intervals through *TickerBehaviour* which is supported by JADE framework. The intervals can be altered manually.

Step 6) the Comm Agents send their anchor information as a response.

Step 7) the received anchor is compared with the one stored in the catalog. In this example, ***'4'*** of computer (A)'s and ***'2'*** of computer (B)'s anchor value are compared. Consequently, the computers can find which computer is the newest one (i.e. a computer which accomplishes synchronization lastly.) Computer (A) is the newest one in this case.

Step 8) the Comm Agent of computer (B) requests data to the computer (A), and computer (A) responses to the request by sending data ***B*** and ***C***.

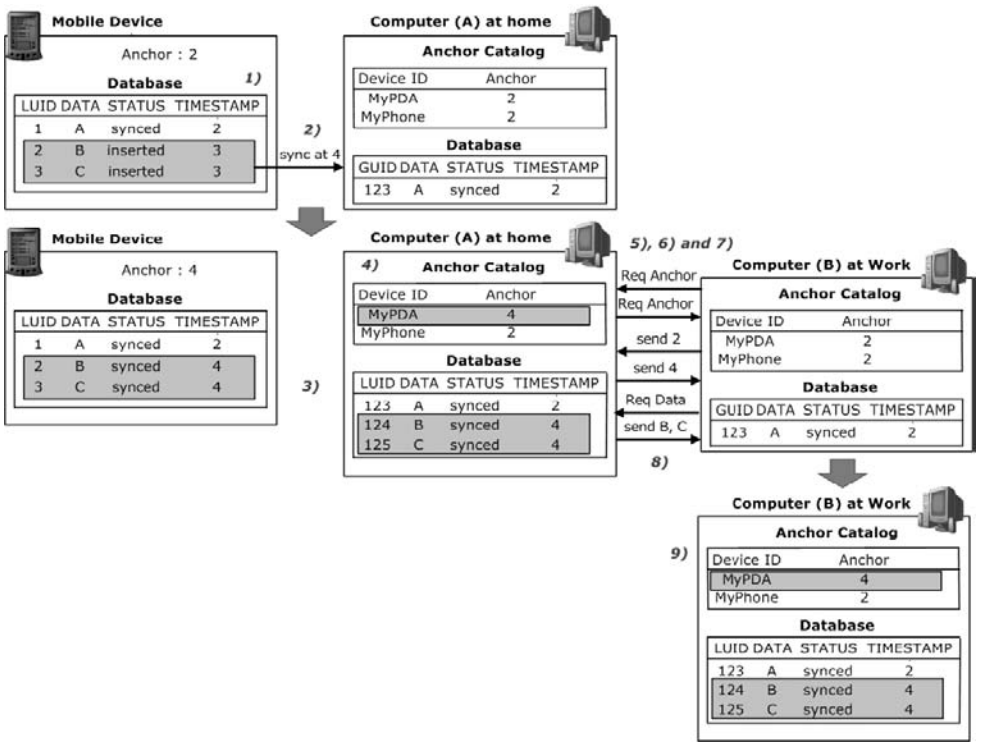

Figure 6. An Example of Inter-server Synchronization

Step 9) the Comm Agent of computer (B) receives data ***B*** and ***C*** and transfers them to the SQL Agent. The SQL Agent makes queries to insert them into the database and also updates its catalog.

After finishing the steps mentioned above, all devices have same data and anchor information. In other words, they are fully synchronized. Even if the mobile device tries to synchronize with one of both computers, a slow sync would not occur.

4. Results

4.1 Implementation results

For the inter-server synchronization, the multi-agent application has been implemented by NetBeans IDE 6. 8.1 and JADE 3.6. It is running on Windows XP. When the application is executed, RMA GUI pops up as shown in Figure 7 and 8. A multi-agent application on each server or computer has its own agents: a CommAgent and a SqlAgent. A SqlAgent manipulates the database of a DS server using SQL queries and a CommAgent takes charge of communication with other servers.

Figure 7. RMA GUI of JADE for WorkPC

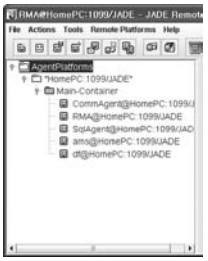

Figure 8. RMA GUI of JADE for HomePC

4.2 Experimental results

In order to evaluate performance of the proposed system, the experimental scenario shown in Figure 9 is designed. HotSync, Intellisync and OMA DS, which are the most representative synchronization protocols, provide two kinds of sync mode: slow sync and fast sync mode. Choosing a mode depends on the condition of anchor or flag states. If anchor of two devices is matched or a flag is not reset, then fast sync mode would be selected. If not, slow sync mode would be selected.

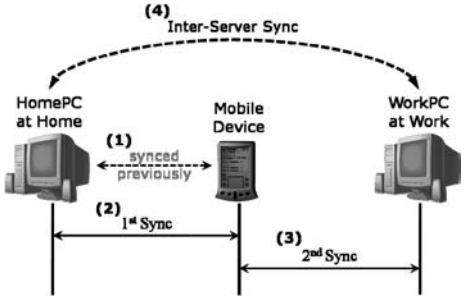

Figure 9. Experimental Scenario

For comparison, several synchronization operations are performed on the embedded DS gateway system.

(1) The mobile device has been synchronized with ***'HomePC'*** previously. It means that they have the same data and anchor information.

(2) Later, the mobile device changes 10 contact records and tries to synchronize with the ***'HomePC'*** again. At this time, only the changed records (10 records) are transferred (i.e. fast sync) because the both devices have the same anchor information.

(3) The user goes to work and tries to synchronize the mobile device with the ***'WorkPC'*** at home. In this case, mobile device sends whole data in its database to the ***'WorkPC'*** since the anchor information of both devices are mismatched called multi-server issue (i.e. slow sync) - It is not only the problem of the embedded gateway but HotSync, Intellisync and OMA DS protocols.

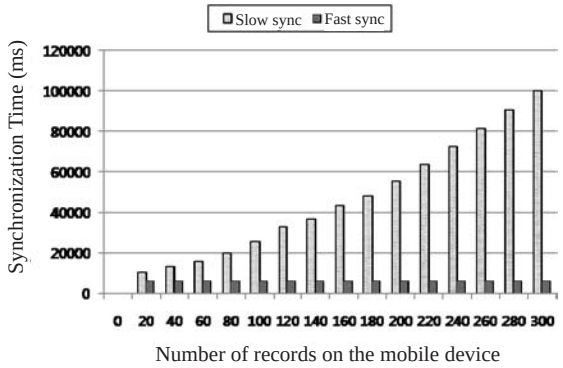

Figure 10. Synchronization Times (number of changes = 10)

The synchronization times are measured in case of (2) and (3) repeatedly with an increasing number of 20 contact records on the mobile device, but a fixed number of changes (i.e. 10). The result depicts in a graphic form in Figure 10. Fast sync time does not grow with the number of records on the device, whereas slow sync time grows linearly. It is obvious that slow sync is extremely time-consuming.

The inter-server synchronization operations are performed between the ***'HomePC'*** and ***'WorkPC'*** as shown as (4) in Figure 9. After that, synchronization operations between the mobile device and the ***'WorkPC'*** are performed as shown as (3) in Figure 9. At this time, slow sync does not occurs since the ***'HomePC'***, ***'WorkPC'*** and the mobile device have the same anchor information. The results depicts in Figure 11. According to the Figure 11, inter-server synchronization operates within an extremely short period. It takes only 2.6 seconds to synchronize 300 records. It takes less than 9 seconds including fast sync time (3). It is clear from the results that inter-server synchronization proposed in this paper provides much better performance than other existing synchronization protocols

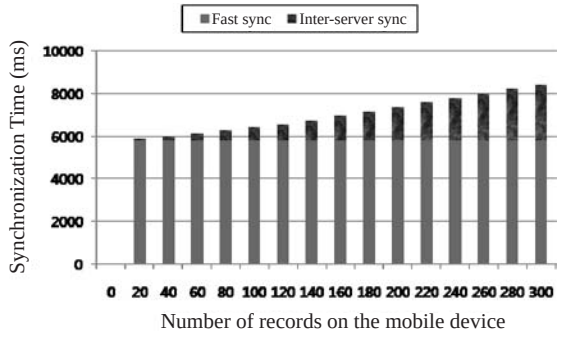

Figure 11. Synchronization Times of Inter-server sync (between the 'HomePC' and the 'WorkPC'), and fast sync (between the mobile device and the 'WorkPC')

5. Conclusion

In multi-server environments, the existing mobile DS protocols have to use slow sync mode. However, the slow sync operation is time-consuming. To solve the problem, several synchronization methods were proposed such as CPISync [13, 14] and Enkel DS [2]. They work well but cause mobile devices' intervention. In this paper, we have introduced the inter-server synchronization system. It synchronizes data between servers without mobile devices' intervention. In the system, data is exchanged directly between servers by multi-agent applications therefore, multi-server issue can be solved. The application described in this paper has several agents and runs on each DS server. We have implemented the application based on JADE (Java Agent Development Platform). According to the experimental results, inter-server synchronization operates within an extremely short period. It takes less than 9 seconds to synchronize 300 records. It is clear from the results that inter-server synchronization provides higher scalability and performance in mobile multi-server DS environments.

6. Acknowledgments

This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology (No. 20100016454).

References

[1] Agarwal, S., Starobinski, D. (2002). On the scalability of data synchronization protocols for PDAs and mobile devices, *IEEE Network*, 16 (4) 22-28.

[2] Alexander, T., Jurgen, N., Frank, K., Michael, W. (2008). Cyclic Data Synchronization through Reusing SyncML, The 9th International Conference on Mobile Data Management, April, p. 165-172.

[3] Fabio, B., Agostino, P., Giovanni, R, (2000). Developing Multi-agent Systems with JADE, *In:* Proceedings of the 7th International Workshop on Intelligent Agents VII. Agent Theories Architectures and Languages, July 2000, p. 89-103.

[4] Foundation for Intelligent Physical Agents, 2002 Specifications, http://www.fipa.org.

[5] Giovanni, C. (2009) JADE TUTORIAL.

[6] Henry, L. (2007). Data representations for mobile devices, *In:* Proceedings of the 13th International Conference on Parallel and Distributed Systems, p. 1-6.

[7] Intellisync Corporation. (2005) Client User's Guide.

[8] JADE, Programmer's Guide, http://jade.tilab.com.

[9] Jugeon, P., KeeHyun, P., Jongjung, W. (2010). Construction of Embedded Data Synchronization Gateway, *The Journal of the Korea Institute of Maritime Information & Communication Sciences*, 14 (2) 335-342.

[10] Jugeon, P., KeeHyun, P., Jongjung, W. (2010). Design of a Data Synchronization Server for Mobile Communication Environments, *The Journal of Korean Institute of Information Technology*, 8 (2) 17-26.

[11] Open Mobile Alliance Data Synchronization. (2009). DS Protocol Specifications Version 1.2. http://www.openmobilealliance.org.

[12] Palm HotSync. Introduction to Conduit Development. http://www.accessdevnet.com/docs/conduits/win.

[13] Starobinski, D., Trachtenberg, A., Agarwal, S. (2003). Efficient PDA Synchronization, *IEEE Transactions on Mobile Computing*, 2 (1) 40-51.

[14] Trachtenberg, A., Starobinski, D., Agarwal, S. (2002). Fast PDA Synchronization Using Characteristic Polynomial Interpolation, *In:* Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and CommunicationsSocieties, p. 1510-1519.

[15] Uwe H., Riku M., Apratim P., Peter, T. (2003). SyncML Synchronizing and Managing Your Mobile Data, PRENTICE HALL PTR, New Jersey.