Proxync : A Framework for Proxy-based Mobile Database with SyncML

Dong M. Park[a] **and Eenjun Hwang**[b]

[a] *Graduate School of Information and Communication*
Ajou University, wonchon, paldal, Suwon 442-749, Kyounggi-do, Korea
Tel: +82-31-219-2535, Fax: +82-31-219-1614, E-mail: minerva@madang.ajou.ac.kr

[b] *Graduate School of Information and Communication*
Ajou University, wonchon, paldal, Suwon 442-749, Kyounggi-do, Korea
Tel: +82-31-219-2654, Fax: +82-31-219-1614, E-mail: ehwang@madang.ajou.ac.kr

Abstract

Mobile agent technologies are getting popular as means for accessing network resources efficiently. In order for mobile agents to be accepted as a reliable technology for applications such as e-commerce, a proper framework for mobile database should be established. In this paper, we first discuss weak points of current mobile computing systems that mostly result from the limitations of current mobile computing technology including frequent disconnection, limited battery capacity, low-bandwidth communication and reduced storage capacity. These weak points also have become the cause of transaction problem where mobile devices issue transactions. In order to eliminate this transaction problem in the mobile environment, we propose a mobile database framework, Proxync, which is based on the proxy and SyncML

Keyword:

Mobile Database; Wireless Network; Proxy; SyncML; Transaction Management

1. Introduction

Recently, advances in wireless network and portable devices have opened a new world of computing, so called *mobile computing*. Mobile computing technologies enable mobile users to access information through wireless network. Moreover, the access is possible anywhere and anytime.

Mobile computing environment is quite different from that of traditional distributed computing. Some of the characteristic features of mobile computing environment can be summarized as fallows [2, 6, 7, 10]:

- *High communication cost*: Mobile clients do not stay connected to the network continuously due to the communication cost.
- *Poor quality of communication*: Mobile clients may have their channel disconnected frequently during communication.
- *Limitation of power and memory*: Most of mobile users turn off their device while they do not use them. The batteries of mobile devices cannot last long. Besides, since mobile devices have limited memory capacity, applications running on the device should be small enough.

Mobile devices have already been used in many applications such as e-commerce, transportation, maintenance teams and telemedicine, to name a few. Currently, mobile service system support mixed platform since most information system support both legacy system (wire network) and mobile system (wireless network).

Information servers usually provide programming interfaces based on relational database. Unfortunately, no framework is

widely available to help programmers to easily deal with mobility including reduced, varying and even unavailable connectivity in mobile environments [5]. Considering new types of mobile services are introduced at a rapid rate, a framework for mobile computing environment is necessary. Motivated by the lack of generic framework, in this paper, we propose a mobile database framework called ***Proxync***, which is based on proxy concept to cache the contents of database and **SyncML** (Synchronization Markup Language) [12] to synchronize transactions within mobile database. The *Proxync* has been designed to consider characteristics of mobile environment such as frequent disconnection and high communication cost that we mentioned before. Especially, in this paper, we focus on how to manage transactions while maintaining database consistency.

The paper is organized as follows. In Section 2, we review ongoing researches on the mobile system, agent technology and existing model for mobile transaction processing. In Section 3, we describe the architecture of *Proxync* and describe the model for transaction processing. In Section 4, we present a scenario to explain how the system works. Finally, in Section 5, we give concluding remarks.

2. Related Work

Many researches have been done so far on the mobile environment. [1, 2] report the development of mobile system and its environment. Especially, [11] offers general architecture in the mobile environment that supports consistency and transaction schema. To access database, it incorporates concepts from concurrency control and advanced transaction models. Particularly, [6] offers a framework for providing consistent and recoverable access to heterogeneous mobile databases. In [10], a mobile agent middleware is proposed. Mobile agent middleware wraps each user profile with an agent.

Transaction management in mobile computing environments is addressed in [3, 4, 7]. They suggest that their own transaction model can be fully adapted to mobile environment. [2] describes how to maintain consistency of replicated data and provide transaction schemes for the frequent but predictable disconnections, the mobility, and the vulnerability of the wireless environment. Finally, [8] presents MobiSnap, a research project that aims to develop SQL based applications for mobile environments, providing configurable support for data divergence control and connectivity abstractions.

SyncML [12] is a common language for synchronizing all devices and applications over any network. SyncML leverages **XML** (eXtensible Markup Language), making SyncML a truly future-proof platform. With SyncML, networked information can be synchronized with any mobile device, and mobile information can be synchronized with any networked applications. *Proxync* is based on this feature of SyncML.

3. System Architecture

When designing *Proxync*, we have considered not only mobile host but also information database system and emphasized following functionalities.

- *Flexibility*: Any system can be integrated with *Proxync*
- *Scalability*: When a new database is added, whole system should not be stopped
- *Concurrency*: Multiple mobile users should be supported
- *Persistency*: Contents of database and cached data should be persistent

Taking these functionalities into consideration, an *object-based architecture* is best fit to the mobile information system. An Object Distribute Management System (ODMS) consists of a number of distributed hosts and clients [5]. Each host supports one or more information systems. Besides, each host has its own information resources. Also, each component has its own agent. The agent is represented as an object, which makes it possible to reference it on the network using RPC (Remote Procedure Call) or Java RMI (Remote Method Invocation).

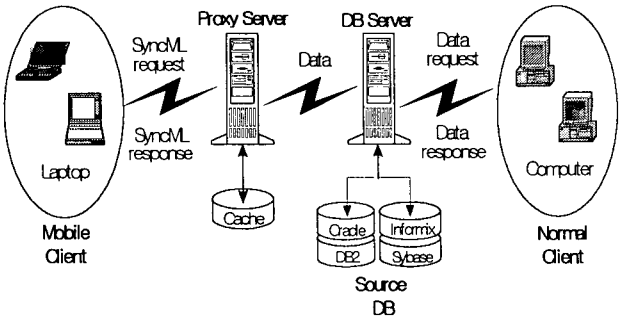

Figure 1 - Mainframe of ***Proxync*** *System*

The overall architecture of *Proxync* is shown in Figure 1 and is composed of three main components:

- ***Database system***: Target database is legacy system that can be connected by normal clients or mobile clients. Target databases could be multiple and need not be homogeneous.
- ***Proxy server***: In *Proxync*, proxy servers have caching data that is a partial replication of source database. It

can be configured by mobile client's profile. Mobile client may access the caching data maintained in the proxy server rather than source database. It has a SyncML server module that deals with connection and transaction control.

- ***Mobile client***: To access the caching data in the proxy server, mobile clients need modules for connecting to proxy server and synchronizing transactions.

Proxy Server The proxy server listens for requests from mobile clients and forwards the requests to external remote Internet servers. The proxy server reads responses from the external servers and then sends them back to the clients. Mobile client does not need to connect to the remote database if a proxy server exists. When mobile client finishes a session with his(her) device, client module issues a transaction to the proxy server. Transactions on the mobile network are error-prone so we used SyncML to handle transaction synchronization. SyncML considers a transaction as an atomic unit, therefore if an unacceptable event occurs, SyncML module rollbacks its operation and redo the transaction after a predetermined time period.

After sending the query, mobile client could turn off his(her) mobile device to save the battery. Proxy server will issue a transaction to the database on behalf of mobile client. This means that mobile client need not maintain the connection for updating the information.

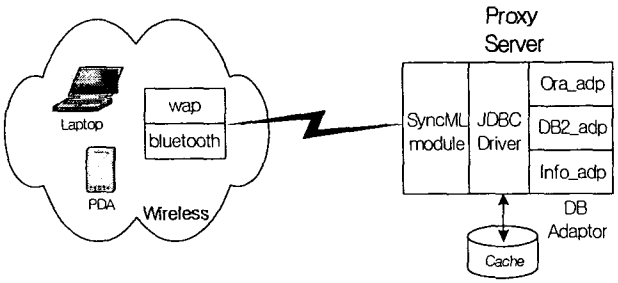

Figure 2 - Modules in proxy server

Proxy server module is implemented in Java. Since Java application is platform independent, it is suitable for mobile devices. Also, every connection module is implemented using JDBC for the following advantages:

- JDBC driver is highly optimized for remote access to the database and supports caching of data (result sets) from queries, which reduces the load on both the database and the network.
- JDBC is easy to integrate with java applications on mobile device and legacy java applications.
- JDBC provides a connection pool on the database server to enhance scalability. Connection pools are especially beneficial for client applications that remain active for a long duration but require only occasional database access.

Proxy Client Communication through the wireless network is very expensive both in cost and in network bandwidth. As a result, it is much better to perform the operation locally on the mobile client. It leads to performance gain and increase availability during disconnection.

Client module is a flexible, effective and light agent that will be loaded on the mobile device with client programs to interact with proxy server. Client module is implemented in Java so that developers can create smart, synchronized applications for a broad range of devices. Since Java is a platform independent language, it relieves a lot of issues like remote execution and migration. Also, it provides rich APIs *(J2ME, JDBC, RMI, CORBA)*, that facilitate development of agent-based applications. Most mobile devices have limited memory capacity, so light Java class file is suitable to make mobile application lighter.

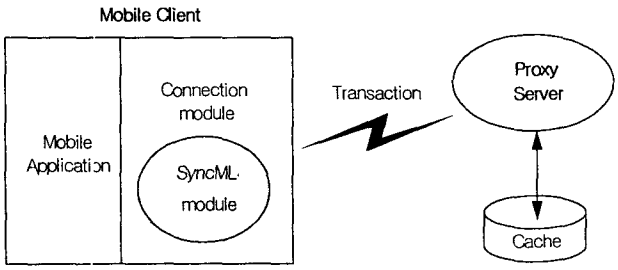

Figure 3 - Modules of mobile client

Most of mobile devices have memory to store the replicated data and a Java application agent to control the replication with proxy server. This application sends a message encapsulated in SyncML to overcome transaction problem within error-prone wireless network.

As mentioned before, both mobile client and proxy server have SyncML agent module. The module on the client side is *SyncML Client* and on the proxy server side is *SyncML Server*. These two modules are connected to each other and send messages for synchronization.

For example, suppose a mobile client turns on his mobile device to update his customer information *SyncML Client* module in mobile application initializes a synchronization request message and sends it to the *SyncML Server* in proxy server. In return, *SyncML Server* responds with an encapsulated message to the *SyncML Client*. After all, client

device becomes to have a replication of proxy data and performs necessary operations with the replication. When the client finishes his operations, changed information is sent back to the proxy.

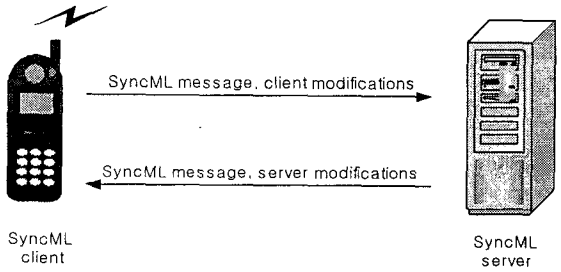

Figure 4 - Sync example with mobile phone and server

Figure 4 shows a synchronization example between a mobile phone and a proxy server. They send a SyncML message to each other if there has been any modification.

Table. 1 SyncML command list

Command	Description
Sync	Allows the originator to indicate that a set of commands should be performed in the specified sequence.
Add	Allows the originator to ask that a data element or data elements supplied by the originator be added to a synchronization data accessible to the recipient. This command must only be specified within a *Sync* command
Atomic	Allows the originator to indicate that a set of commands should be performed with all or nothing semantics.
Copy	Allows the originator to ask that a data element or data elements accessible to the recipient be copied.
Delete	Allows the originator to ask that a data element or data element accessible to the recipient be deleted. This command can include a request for the archiving of the data. The deletion can either be soft-or hard-delete.
Search	Allows the originator to ask that the supplied query be executed against a data element or data elements accessible to the recipient.
Status	Indicates the completion stats of an operation or that an error occurred while processing a previous request.

Table 1 shows some of the SyncML commands that can be used in order to synchronize transactions within the mobile database. *SyncML Client* module encapsulates the replicated data to make it an atomic unit and then sends it to the *SyncML Server*. While two modules are communicating with each other, encapsulated message would be missed or damaged. In such case, transaction processing will be stopped and rollback operation will start. Since the whole transaction is considered as an atomic operation, every message between *SyncML Client* and *SyncML Server* is stored in the log repository that will be used for tracing synchronization information.

4. Transaction Management

Characteristics of mobile transactions [2] can be summarized as follows :

- Mobile transactions are transactions that involve the wireless network and may be executed in both mobile and non-mobile hosts.
- Mobile transactions tend to be:
  - Monetarily expensive;
  - Long lived, because of long network delays;
  - Quite error-prone, because of frequent disconnections but also because mobile hosts are more prone to accidents than fixed hosts; and
- Mobility results in transactions with the following characteristics:
  - Transactions may access to heterogeneous information system.
  - Transactions may access (possibly imprecise) local data.

In [2, 9], an axiomatic definition for a transaction model of mobile transaction is presented. To deal with the mobility, we adopted some of the model that includes *transaction migration*. To deal with consistency, we supplemented it with *snap shot* technique and to deal with the concurrency we used *time-stamp* technique.

Transaction migration is similar to the dual of data relocation. When a mobile client is moving from cell i to cell j, the transaction partially executed on the old base station must be transferred to the new base station. Let T_i be a transaction that was initially submitted at site i and then was migrated to site j. We use the notation $T_{i \to j}$ to indicate the portion of the transaction executed at site i before moving to cell j.

Proxync's caching and update propagation algorithms will be based on the notion of database snap shot. Each database snapshot is composed by several table snapshots [13]. For instance, a client would need a snapshot related with his interests. So, *Proxync* should be responsible for assuring that data divergence is kept below a threshold that can be changed by mobile client.

When several transactions occur simultaneously, *Proxync* uses *time stamping* technique. A unique identifier that is created by the *Proxync* indicates relative starting time of a transaction [13]. For instance, for time units i and j, if $i < j$ then, $ts(i) > ts(j)$. Time stamp with smaller value gets priority when conflict occurs. Every client module sends queries with GMT (Greenwich Mean Time) provided by the base station. Therefore, it is possible to arrange transactions by time order.

We intended to support *Proxync* systems can redress the transaction problems with *transaction migration, snap shot, time stamp* methods.

So far, we have described the architecture of *Proxync* system and how it works in the mobile computing environment. Tts advantages can be summarized as follows:

- *Lower connection cost*: Mobile clients may not need to access distant database. And they can turn off their device while their proxy server connects to the database on behalf of them.
- *Support multiple heterogeneous databases*: Proxy server implemented in JDBC to make it possible to connect to diverse databases with connection pool.
- *Reduce bottleneck of main database*: Centralized database has bottleneck problem, and to avoid this problem, we use a proxy server to distribute requests from clients. *Proxync* has a decentralized architecture. Of course, there is a trade-off in efficiency, consistency and cost.
- *Remove transaction problem*: Most database systems have transaction problem and many efforts have been made to solve it. In this paper, we used SyncML in order to get rid of the transaction problem by synchronizing the replicated data in mobile device with the data in proxy server.

5. Example Scenarios

In this section, we will present two application scenarios with our *Proxync*-based system. The first scenario will show the data synchronization between the proxy server and mobile client. In the second scenario, we will consider the case where the mobile client is switching to neighboring cell while executing a transaction.

Transaction synchronization scenario: John is on his way to repair a customer's computer. John always carries his laptop to access the corporate database in order to download customer's personal information into his laptop. When repair is done at the customer's site, he would want to update customer's record while he is back to his company. First, he updates the customer's information in his laptop. And then, the SyncML client module in his laptop encapsulates the updated information and sends it to the proxy server. It may be possible for the mobile client to be disconnected while sending the update information to the proxy server due to some reason.

In order to handle such case, SyncML provides a <*sync*> tag, which specifies a sequence of commands. The agent on the proxy server checks the sequence of commands. If the sequence is not complete, then SyncML agent waits for the remaining data. If John's computer becomes accessible to the SyncML agent server within a predefined period, client module will send the remaining data to the proxy server where the update will be reflected to the corporate database permanently.

Transaction migration scenario: Another important thing that has to be considered is transaction problem. Actually, this problem has been one of important research issues in the distributed database community.

Let us continue the scenario with John. He is interested in stock investment, and he frequently connects to the stock market sites on the Internet with his laptop. He searches for the issue price of shares and deals in stocks. Sometimes, he might move from cell A to cell B while he is transferring his shares. In this case, the transaction that has been executed at cell A has to be migrated to the cell B.

SyncML has <*atomic*> tag that makes it possible to consider a sequence of commands as an atomic operation. In other words, a set of commands is performed in the all-or- nothing style. In this case, transferring shares is considered as an atomic operation, so any incomplete transaction will not be reflected to the database. Later, the proxy server will try to revisit the stock market to complete the transfer without client's intervention.

6. Conclusion

It is expected that in the near future, a lot of users will access distributed information systems through wireless connections. As mobile computing is getting popular, a generic framework for mobile information processing is required. In this paper, we have designed a flexible platform called *Proxync* for mobile computing. It mainly consists of proxy-based caching

server module and mobile client module and they both are implemented in Java. *Proxync* supports multiple heterogeneous databases and uses SyncML to synchronize the transaction issued by mobile clients. Our strategy was to design a synchronizable and flexible mobile database platform. The contribution of this paper is twofold. First, we have identified existing problems of current mobile computing system especially in the database aspect:

- Current status of mobile computing
- Weak points of current mobile environment
- Ongoing research on this area

Second, we have designed *Proxync* framework appropriate to the mobile environment and presented general principle under which we implemented the system including:

- Transaction migration
- Database snapshot
- Time stamp technique

Finally, we have shown two example scenarios that are involved with synchronization and transaction control. And we plan to expand our system to support home network by implanting the SyncML module into the home appliances.

Reference

[1] B. R. Badrinath, S. H. Phatak, "A Database Architecture for Handling Mobile Clients".

[2] E. Pitoura and B. Bhargava. "Building information system for mobile environments", *Proc. of the 3rd International Conference on Information and Knowledge Management*, pp.371-378, Guithesburg, MD, November 1994.

[3] Gary D. Walborn and P. K. Chrysanthis, "Pro-Motion: Management Of Mobile Transactions". *Proc. of the 11th ACM Annual Symposium on Applied Computing*, pp.101-108, 1997.

[4] Herman Chung-Hwa Rao, Yih Farn Robin Chen, Di-Fa Chang and Ming-Feng Chen, "iMobile: An Agent-Based Platform for Mobile Services", WWW Posters 2001.

[5] John R. Nicole, C. T. Wilkes, and F. A. Manola. "Object Orientation in Heterogeneous Distributed Computing Systems", *IEEE Computers*, 26(6): pp.57-67, June 1993.

[6] K. Karlapalem, Qing Li, Chung-Dak Shum, "An Architectural Framework for Homogenizing Heterogeneous Legacy Database", *SIGMOD Record 24(3)*: pp.44-49, 1995.

[7] M. H. Dunham and A. Helal, "Mobile Computing and Database: Anything New?", *ACM SIGMOD Record*, pp.5-9, December 1995

[8] N. M. Preguiça, C. Baquero, F. Moura, J. Legatheaux Martins, R. Oliveira, H. João, L. Domingos, J. O. Pereira, and S. Duarte, "Mobile Transaction Management in Mobisnap", *ADBIS-DASFAA*, pp.379-386, *2000*.

[9] P. K. Chrysanthis, "Transaction processing in mobile computing environment", *In IEEE Workshop on Advances in Parallel and Distributed System*, pp.77-82, October 1993.

[10] Paolo Bellavista, Antonio Corradi, and Cesare Stefanelli, "Mobile Agent Middleware for Mobile computing", *Computer*, Vol. 34, No. 3, pp.73-81, March 2001.

[11] Synchronization Markup Language (SyncML), Available at http://www.syncml.org.

[12] Tomasz Imielinski and B. Badrinath, "Mobile wireless computing: Challenges in data management", *Communications of the ACM*, 37(10): pp.18-28, October 1994.

[13] T. Connolly, C. Begg, and Anne Straghan, "Database Systems", ADDISON-WESLEY, pp.597-598, 1996.