A Stateful Database Synchronization Approach for Mobile Devices

B Sri Ramya, Shirin Bhanu Koduri, M.Seetha

Abstract: ***This paper proposes a Stateful Synchronization of data between a server-side database and a mobile database. The Stateful Synchronization stage is an amalgamation of the Stateful and Synchronization modes. This means that invalidation reports are sent only to the affected clients (Stateful) and they are sent periodically (Synchronous).The existing Synchronization Algorithms based on comparison of message digest values of the selected rows of both server-side database and mobile database needed for Synchronization. But in the proposed approach the server keeps the information of the present state of data records at the mobile databases. The Server also transmits the invalidation information at periodic intervals to the clients, whenever the data record relevant to the client is invalidated or modified at the server. Here there is no need to calculate the Message Digest values for Synchronization of databases as the server maintains the state of clients.***

Index Terms: Mobile Device, Server Side Database, Mobile Database, Stateful Synchronization, Invalidation Report.

I. INTRODUCTION

Recent advances in mobile technology and equipment have led to the emergence of a new computing environment and a variety of small sized mobile devices such as PDAs (personal digital assistants), smart mobile phones, HPCs (handheld PCs)and Pocket PCs have been popularized. As various network technologies are increasingly being associated with such mobile devices, the processing of business information can be available using mobile devices. As a result, business models that rely on mobile technologies are appeared [1].

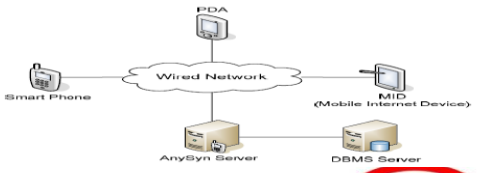
Mobile devices do not have much computing power and rely on batteries. Additionally, constant access to network is difficult due to narrow bandwidth [2] [3]. Therefore, it is not easy to process a large size of stored data and maintain a continuous connection with the server-side database. For these reasons, mobile devices have mobile databases in order to achieve stable data processing. Mobile devices download replications of limited data from a connected server-side database using a synchronization device that has a stable wire communication function. Mobile devices process various tasks using the data downloaded in an off-line state. The work on the network disconnected condition is a crucial point for mobility support [4]. In a disconnected environment, there are inevitable inconsistencies between the server-side database and the mobile database. Synchronization techniques can solve the data inconsistencies and guarantee the integrity of the data. Consequently, synchronization is an essential subject in mobile device computing environments [5]. Commercial DBMS venders offer various solutions to data synchronization in a mobile environment [6], [7], [8]. However, these solutions are not independent of the server-side database because they use database dependent information such as metadata or use specific functions of server-side database such as trigger and time stamp. In other words, the mobile database vender should be equivalent to the server-side database vender.

The solution of operating a separate synchronization server in the middle tier is independent of the server-side database but dedicated to the mobile database. That is, the synchronization solution and the mobile database should be the identical vender product. Additionally, when a client programmer develops mobile applications that are embedded in mobile devices, the developer should uses a particular library that is provided by the vender of mobile database or modify existing mobile applications for synchronization process. Because of these restrictions, the extensibility, adaptability and flexibility of mobile business systems are markedly decrease. This problem must be solved in order to build efficient mobile business systems because upcoming mobile environments will have heterogeneous characteristics in which diverse mobile devices, mobile databases and RDBMS exist.

II. BACKGROUND KNOWLEDGE

A. Synchronization Framework:

Fig. 1 represents a synchronization framework using a synchronization server in a mobile business environment. The whole framework consists of a server-side database, synchronization server (AnySyn) and multiple mobile devices with internal mobile databases. The server-side database maintains all of the data required for business, and the mobile database downloads copies of data the user needs from the server-side database. The synchronization server is located between the two databases to synchronize the data and manage.



Manuscript received June 26, 2012.

B. Sri Ramya, Asst. Professor in IT Department, Sri Vasavi Engineering College, Pedatadepalli, Tadepalligudem, India,

Shirin Bhanu Koduri, Assoc. Professor in IT Department, Sri Vasavi Engineering College, Pedatadepalli, Tadepalligudem, India,

Dr. M. Seetha, Professor in CSE Department, G Narayanamma Inst. Of Technology and Sciences, Hyderabad, India,

Fig 1 SAMD Synchronization Framework

Additional information required for synchronization. The AnySyn synchronization server performs synchronization based on the SAMD algorithm. The synchronization policy is established in AnySyn and the load caused by accessing the server-side database is minimized by operating a connection pool. Every mobile device uses a separate toolkit to access the AnySyn server over a wired network to perform synchronization.

B. Existing System:

The existing System uses Synchronization Algorithm based on comparison of message digest values of the selected rows of both server-side database and mobile database needed for Synchronization.

C. Message Digest:

Message digest consists of a unidirectional hash function that maps a message of a random length to a fixed-length hash value. Message digest **h** is created by the hash function **H**, which can be expressed as follows:

$$h = H(M)$$

M is a message of a random length and H(M) is a fixed-length message digest. Even a single bit changed in the message causes a change of message digest value.

III. SAMD SYNCHRONIZATION ALGORITHM

Objective of SAMD:

In order to guarantee independence of database vender and synchronization solution vender in a mobile business environment that has diverse mobile devices, mobile databases, and RDBMS, the SAMD synchronization algorithm satisfies the following objectives.

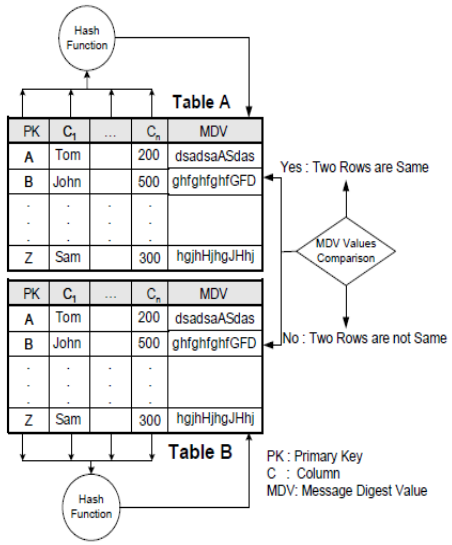


Fig. 2. Message Digest in Data Table

1. Independence of venders.
   Does not use metadata or internal functions dedicated to a particular database.
2. Synchronization using only standard SQL statements.
   Perform synchronization using only standard SQL queries and data manipulation language specified in ISO standard. Therefore, any data processing using trigger is not allowed.
3. Disallows schematic modification of data table of the server-side database.
   The data table schema cannot be modified to add data necessary for synchronization. In other words, synchronization must be performed independent from the existing data table schema. Therefore, additional information such as time stamps cannot be added to the data table.
4. Disallows adding restrictions in implementing applications.
   There can be no restrictions such as performing additional works to an application code or having to use a specific library in order to perform synchronization.

The SAMD synchronization algorithm is applied for the table schema of the server-side database and the mobile database. Both databases have a data table (DSDT: Database Server Data Table, MCDT: Mobile Client Data Table) and a message digest table (DSMDT: Database Server Message Digest Table, MCMDT: Mobile Client Message Digest Table). The data table contains the business data, and the message digest table stores the message digest value from the data table. The message digest table consists of a PK(primary Key) column of data table, message digest value (MDV) column, flag (F) column and mobile device ID (Mid) column. The flag column signals an inconsistency that has occurred in the corresponding column; therefore, the flag column is used to identify a row that requires synchronization. The mobile device ID is a unique number of the mobile device, so this column is used to identify a mobile device that requires synchronization.

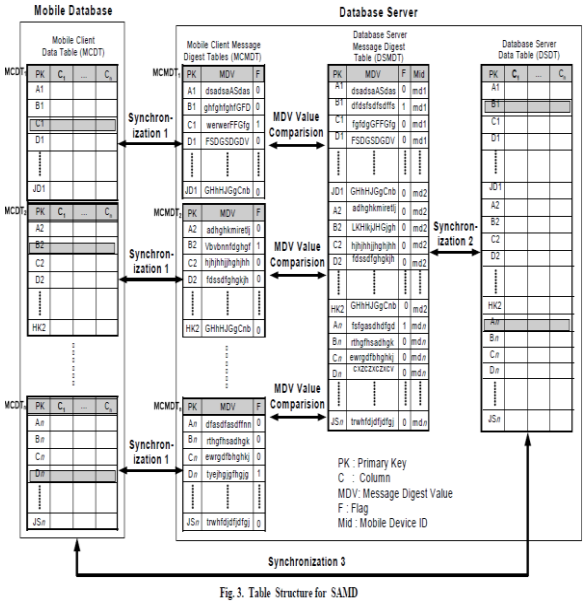


Fig. 3. Table Structure for SAMD

In Fig. 3, if a row's PK value is A1, this value is identical to the two message digest values and there is no need for synchronization. However, if a

row has a PK value of C1, the value of MDV in MCMDT is different from the value of MDV in DSMDT and the MCMDT

flag value is 1. Consequently, synchronization is necessary. The synchronization process is performed for each row to resolve all of the inconsistencies mentioned in Section II/*B*. For example, if there is an inconsistency in row C1, synchronization takes place from the mobile database to the server-side database and DSDT's PK C1 row is replaced with the MCDT's C1 row.

IV. PROPOSED WORK

This paper proposes a stateful synchronization between the server side database and mobile database. The advantage of maintaining states is, whenever a modification is done at the server side of a particular client data then the respective client will be given alerts or invalidation information among all the clients. There is no need of calculating the message digest values as the server maintains the state of clients. This paper performs the stateful database synchronization by taking the application of patient monitoring system.

A patient monitoring system provides monitoring of a patient includes a data acquisition and processing module receiving the physiological data from the client. Various physiological details such as body temperature, blood pressure will be updated by the patient and are monitored as well as updated by the doctor.The patient has given permission to edit the health deatils and view the Doctor updates and the doctor has given permission to view the details given by the patient and update the treatment details.There are 3 status checks performed in this application by setting the flags as mentioned below.

1. When no one is connected to the system then the status flag will become 0.
2. When the Client is viewing the details and doctor is not connected then the status flag will become 1.
3. When the Client wants to view the details and the doctor has been connected and updating the treatment details then the status will become 2. After getting details updated by the doctor an alert will be sent to the client that the treatment details have been updated and the status flag will become 1.

Synchronization takes place in third step i.e when the status has been changed from 2 to 1 and it will be stateful as the alert has been sent to the client about the state after doctor updation

Three modules are used to describe Patient Monitoring System. They are

Client/Patient
Decision Maker/Doctor
System Admin

V CLIENT

1. Everyone needs to have Medical attention at any time. So we allow every user to register freely at any time.
2. The user does registration by specifying their details like His/Her Name, Contact Number, Gender. After validation the user will receive a message regarding his/her membership.
3. The Doctor list is categorized by his/her specialization HEART, SKIN CARE, CHILD CARE respectively. So that the user can easily access the doctor for his/her treatment.
4. System Admin maintains the prescription given by the doctor for future use. Patient can view their prescription any time.

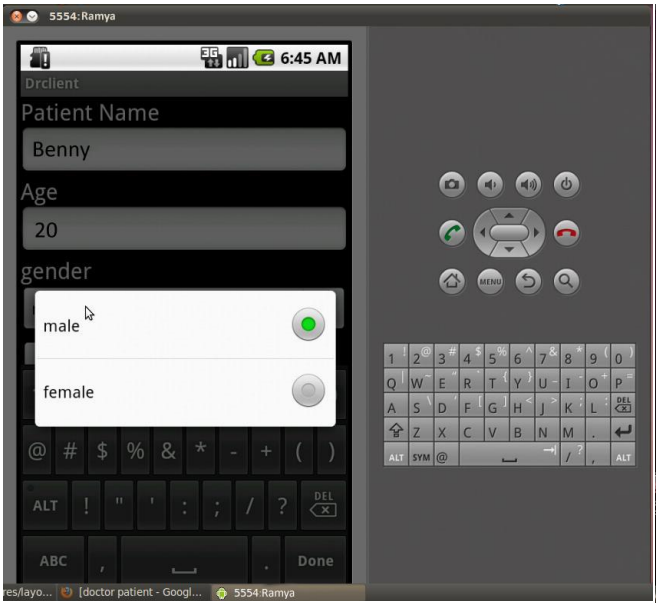
VI. DECISION MAKER

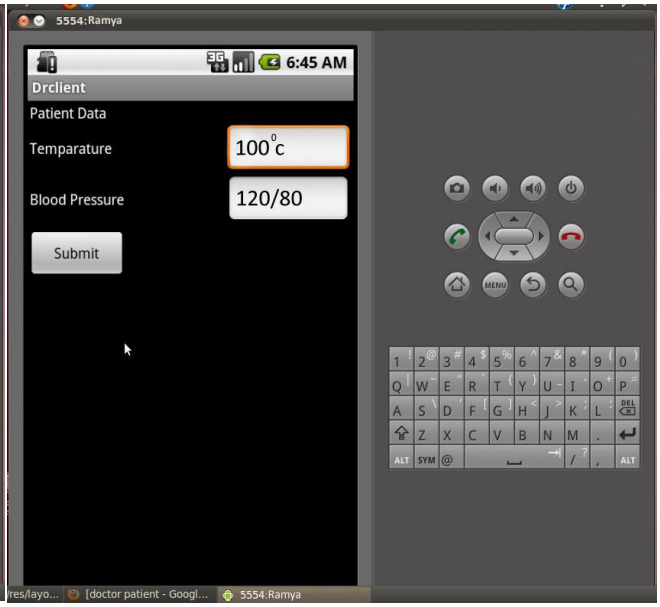
1. Administrator does Doctor registration.
2. Every Doctor will have their own unique Id and Password with which, they will login to this site.
3. After they logged into this site. They will have their main form. From there by choosing the link, they can see their appointments. He/she can see their new appointments and they can also see the previous appointments.
4. After attending the patient, the Doctor selects a particular patient Id for prescription. In the prescription form, the Doctor will enter the detail about the prescription and give to the particular patient like specifying Patient condition, Kind symptom and dosage about the medicine.

VII. SYSTEM ADMINISTRATOR

1. A Genuine person from the Administrator side will collect information about the Doctor like his/her Qualification, specialization, etc for Registration. After filtering the invalid data, the Doctor Details will be uploaded in Patient Monitoring System site.
2. Before uploading their details the Administrator will send mail to the Doctor's E-mail ID about his/her Id and Password.
3. The Administrator can also add new Hospital which is specified by the Doctors while recommending new Hospitals for patient for further or future treatments.

This application uses a cloud server which acts as an access point between client and decision maker. The Client data and the server data is pushed into the database and that database is stored into the cloud server. The reason is that the client and decision maker are having dynamic ip and cloud server is having static ip. A dynamic ip is one that changes every time you connect to the network and a static ip is one that remains the same no matter how many times you connect and disconnect from the network.





Cloud server consists of a static IP and our application will run on a particular port where the phone with GPRS will communicate with the server application for sending and receiving data. The server app is a java socket program which will be listening on particular port as soon as connection from any mobile phone with GPRS starts the server will respond accordingly. The server program uses jdbc drivers to communicate with mysql data base. The programmer will connect to cloud server through any of the following protocol application ssh (putty) and ftp. Ssh is used as command line interface (cli) to communicate with the server. ftp is used for uploading and downloading files from server.

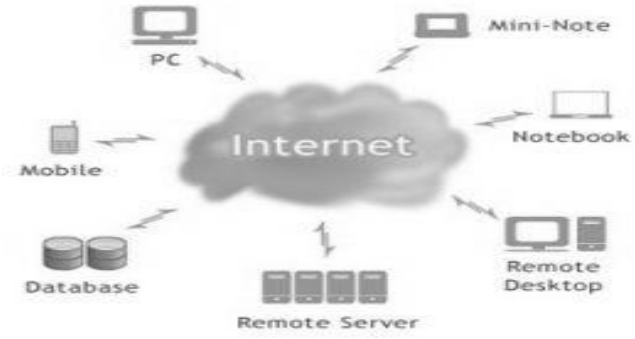


Fig 4 Cloud Server Architecture

VIII. SAMPLE OUTPUT SCREENS

Client Login:

Client Details Updation :

Client *Physiological Details* Updation:

Treatment Details Updation:

Client Viewing the Status:

In the above output screen an alert message has been sent to Patient when the patient has clicked the view status button at the time of doctor treatment details updation.Before sending alert message status flag changes to 2 and after updation of details

IX. CONCLUSION

This paper implements a Stateful Synchronization of Mobile Devices. The existing

system performs synchronization by comparing message digest values. This application maintains the state of the clients without using SAMD Algorithm and without calculating message digest values. Also frequent alerts are sent to the respective clients about the changes made in the database. This application uses a cloud server to achieve stable data processing and maintenance

REFERENCES

1. Mi-Young Choi, Eun-Ae Cho, Dae-Ha Park, Chang-Joo Moon, Doo-Kwon Baik "A Database Synchronization Algorithm for Mobile Device"sIEEE Transactions on Consumer lectronics, Vol. 56, No. 2, May 2010
2. Tomasz Imielinski and B. R. Badrinath, "Mobile wireless computing: challenges in data management", Communications of the ACM, Volume 37 Issue 10, pp. 18-28, 1994.
3. Barbara, D., "Mobile Computing and Databases - A Survey", IEEE Transactions on Knowledge and Data Engineering, Vol. 11 No. 1, pp 108- 117, 1999.
4. EPFL, U. Grenoble, INRIA-Nancy, INT-Evry, U. Montpellier, "Mobile Database: a Selection of Open Issues and Research Direction", SIGMOD Record, Vol.33, No.2, pp.78-83, June, 2004.
5. Joshua Savill, "MobiLink Synchronization Profiles", A Whitepaper from Sybase iAnywhere., October 17th, 2008
6. Thomas Fanghänel, Jonas S Karlsson, Cliff Leung, " DB2 Everyplace Database Release 8.1: Architecture and Key Features", Datenbank-Spektrum, pp. 1~15, 5/2003
7. Gye-Jeong Kim, Seung-Cheon Baek, Hyun-Sook Lee, Han-Deok Lee, Moon Jeung Joe, " LGeDBMS: a small DBMS for embedded system with flash memory", 32nd international conference on very large data bases, pp. 1255~1258, 2006
8. john E. Canavan, "Fundamentals of Network Security", ARTECH HOUSE, INC., 2001, 61~62
9. Jonathan Knudsen, "WIRELESS JAVA : Developing with Java 2, Micro Edition", A press, 2001, pp. 155. [10] "MobiLink Synchronization User's Guide", Sybase, Inc., 2004, pp. 37 ~68, pp. 392~402

AUTHORS PROFILE

Mrs. B. Sri Ramya received her B.Tech Degree in Computer Science From JNTU Hyderabad.At present working as an Asst Professor in IT Department at Sri Vasavi Engineering College, Pedatadepalli., Tadepalligudem, A.P.

Mrs. K. Shirin Bhanu received her M.Tech degree in Computer Science from JNT University Kakinada. At present working as Assoc. Professor in IT department at Sri Vasavi Engineering College, Pedatadepalli, AP, Having 8+ years of experience in teaching. Her research interests includes Mobile Computing, Grid Computing and MANETs. She is working towards her Ph.D. in MANETs at JNTU Hyderabad