ISSN:0975-9646

An ARCHITECTURE MODEL for MANAGING TRANSACTIONS in DISCONNECTED MOBILE ENVIRONMENT

J.LWalter Jeyakumar[#1], R.S.Rajesh[#2]

[#]Department of Computer Science and Engineering
Manonmaniam Sundaranar University,
Tirunelveli, Tamilnadu, INDIA

***Abstract*— We present an architecture model for mobile transaction management for supporting disconnected computing. In this environment, Fixed Agents in the wired network cache the frequently accessed data from the server, which can be accessed concurrently by the mobile nodes. Data Access Manager module at the Fixed Agent controls the concurrency using invalidation technique. Using this scheme, a mobile host can assign its transaction execution to Data Access Manager at the Fixed Agent before disconnection. We have proposed two types of transactions namely trusted and non trusted transactions. Non trusted transaction can be allowed to execute without authentication. But a trusted transaction can be executed after running a trust protocol. The proposed transaction management framework has been simulated in J2ME and NS2 and performance of the scheme is analysed.**

***Keywords*— Mobile transaction, Fixed Agent, Concurrency control, Cache invalidation.**

I. INTRODUCTION

In mobile computing environment, the transaction processing models should incorporate the shortcomings of mobile computing such as unreliable communications, frequent disconnections, limited battery power, low bandwidth communications and reduced storage capacity. Frequent aborts due to disconnection should be minimized in mobile transactions. Correctness of transactions executed on both fixed and mobile hosts must be ensured by the operations on shared data. Blocking of mobile transactions due to long disconnection periods should be minimized to reduce communication cost and to increase concurrency. After disconnection, mobile host should be able to process transactions and commit locally. Mobile computing provides the possibility of concurrent access of data by mobile hosts which may result in data inconsistency. Concurrency control methods have been used to control concurrency. Due to limitations and restrictions of wireless communication channels, it is difficult to ensure consistency of data.

In this paper, we present an agent based framework for transaction processing. Frequently accessed data are cached in the Fixed agent situated in the fixed wired network. Whenever an MH enters into a Fixed agent area it can connect and access the data in the cache. But upon update request by a MH, updation is done at the local cache and invalidation report is sent to all the mobile hosts which have already accessed the same data. This will force the mobile hosts to refresh their data values. Data Access Manager at the fixed agent is responsible for concurrency control and data invalidation. This framework also takes into account transaction update during disconnection. Trusted and non trusted transactions are treated separately.

The remaining part of this paper is organized as follows. Section II summarizes the related research. Section III focuses on the agent based frame architecture. Section IV specifies the proposed framework for disconnected mobile computing. Section V gives the performance analysis and in Section VI the conclusion is presented.

II. RELATED WORK

When simultaneous access to data is made at the server, concurrency control techniques are employed to avoid data inconsistency. Conventional locking based concurrency control methods like centralized Two Phase locking and distributed Two Phase locking are not suitable for mobile environment. The system overhead that arises due to concurrency control mechanism can create a serious performance problem because of low capacity and limited resources in mobile environment [1]. More over, it makes mobile hosts to communicate with the server continuously to obtain and manage locks [2].

In Timestamp approach, the execution order of concurrent transactions is defined before they begin their execution. The execution order is established by associating a unique timestamp to every transaction. When two transactions conflict over a data item, their timestamps are used to enforce serialization by rolling back one of the conflicting transactions [3]. In optimistic concurrency control with dynamic time stamp adjustment protocol, client side write operations are required. But it may never be executed due to delay in execution of a transaction[4]. In multi version transaction model [5], data is made available as soon as a transaction commits at a mobile host and another transaction can share

this data. But data may be locked for a longer time at a mobile host before the lock is released at the database server.

In [6], a transaction model for supporting mobile collaborative works was proposed. This model makes use of Export-Import repository which is a mobile sharing work space for sharing data states and data status. But in the Export-Import repository based model, locking is the main technique which has the following disadvantages. (i) More bandwidth is needed for request and reply since the locking and unlocking requests have to be sent to the server. (ii) Disconnection of mobile host or a transaction failure will result in blocking of other transactions for a long period.

In [7] , AVI (Absolute Validity Interval) was introduced for enforcing concurrency control without locking. AVI is the valid life span of a data item. But it calculates AVI only based on previous update interval. In [8], a method based on PLP(Predicted Life Period), which takes care of the dynamicity of the life time of data was proposed. Here, life span of data is predicted based on the probability of updation of data item. This method makes PLP of data item very close to the actual valid life span of a data item. The above mentioned research works have made little or no attempts on the disconnected issues. But our approach takes into account the disconnection issue and also trusted and non trusted transaction cases which are specific to applications such as Banking or Credit Card transactions.

III. AGENT BASED ARCHITECTURE

The proposed Agent based architecture model is illustrated in Fig 1. The model consists of Server, Fixed Agents and Mobile Hosts. The server can be directly connected to a mobile host. Fixed Agents are connected to the server through wired/wireless network. Fixed Agent has a communication range and any mobile host that enters into the agent area can connect to the server through the agent.

In Fixed Agents, cache is used to store the data. Mobile hosts are allowed to access data from the cache. When data request is made for the first time, data is retrieved from the server and stored in the cache. Subsequent requests are handled by the Data Access Manager module itself. When a mobile host requests for data update, after local updation of the data item, invalidation report is sent to all the mobile hosts that have already accessed the same data. This makes all the mobile hosts to refresh their data values. When a mobile host is disconnected from the Fixed agent after updation request, the updation task is transferred to the Data Access Manager in the Fixed Agent. Data Access Manager module is used to coordinate the operations in the cache.

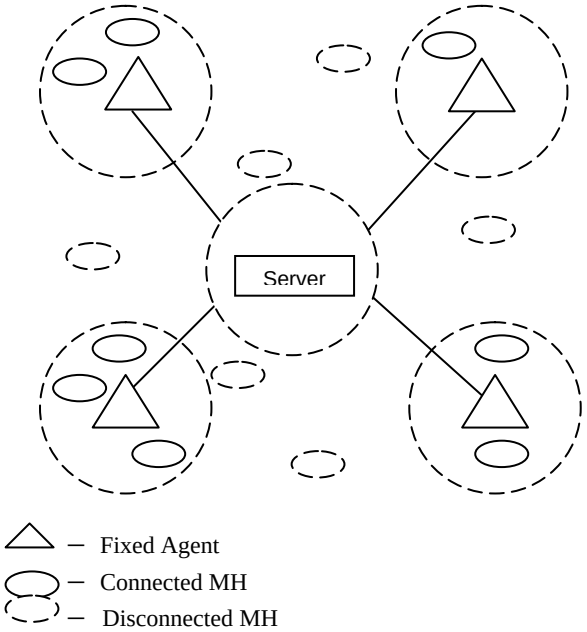

Fig. 1 Agent Based Architecture model

IV. MOBILE TRANSACTION FRAMEWORK

Mobile hosts can directly connect to the server. But simultaneous access to data at the server will increase the system overhead. To overcome this disadvantage, data are cached in the Fixed Agent. The Data Access Manager at the Fixed Agent is responsible for enforcing concurrency and cache invalidation. Fig 2 and 3 illustrate the steps involved in MH, server and DAM algorithms.

A. Concurrency Control mechanism

When more number of mobile hosts are accessing data simultaneously the problem of data inconsistency arises. This problem can be solved if we use an efficient concurrency control mechanism. When data request is made for the first time, data is retrieved from the server and stored in the cache. Future requests for data are managed directly by the Data Access Manager.

Data Access Manager uses a suitable data item format to store data in the cache [8]. It has (id, TLU, PLP, dataval, NT) where id denotes unique Id of the data item, TLU indicates time of Last Update, PLP is Predicted Life Period, dataval is current value of the data item and NT denotes number of transactions that concurrently access the data item.

When Data Access manager fetches data for the first time from the server, it sets TLU to current time, PLP to optimal

time based on the nature of data item and NT to 1. NT is incremented whenever a new data access request is made. Data in the cache becomes invalid, once it is updated in the server. Life span of a data item is predicted using PLP. It makes use of the probability of updation as a basis for setting valid life span of a data item. In PLP interval, data item is valid and all the mobile hosts can access same data item concurrently.

When a MH makes update request or PLP expires, the data item is invalidated. Now PLP is modified and invalidation report is sent. The predicted life period of data item is computed using the formula

PLP=PPLP ± (p*PPLP)

Where PPLP is Previous Predicted Life Period and p is predicted probability of updation of data item. p = Total_updates / NT. It is the ratio of data item update to data

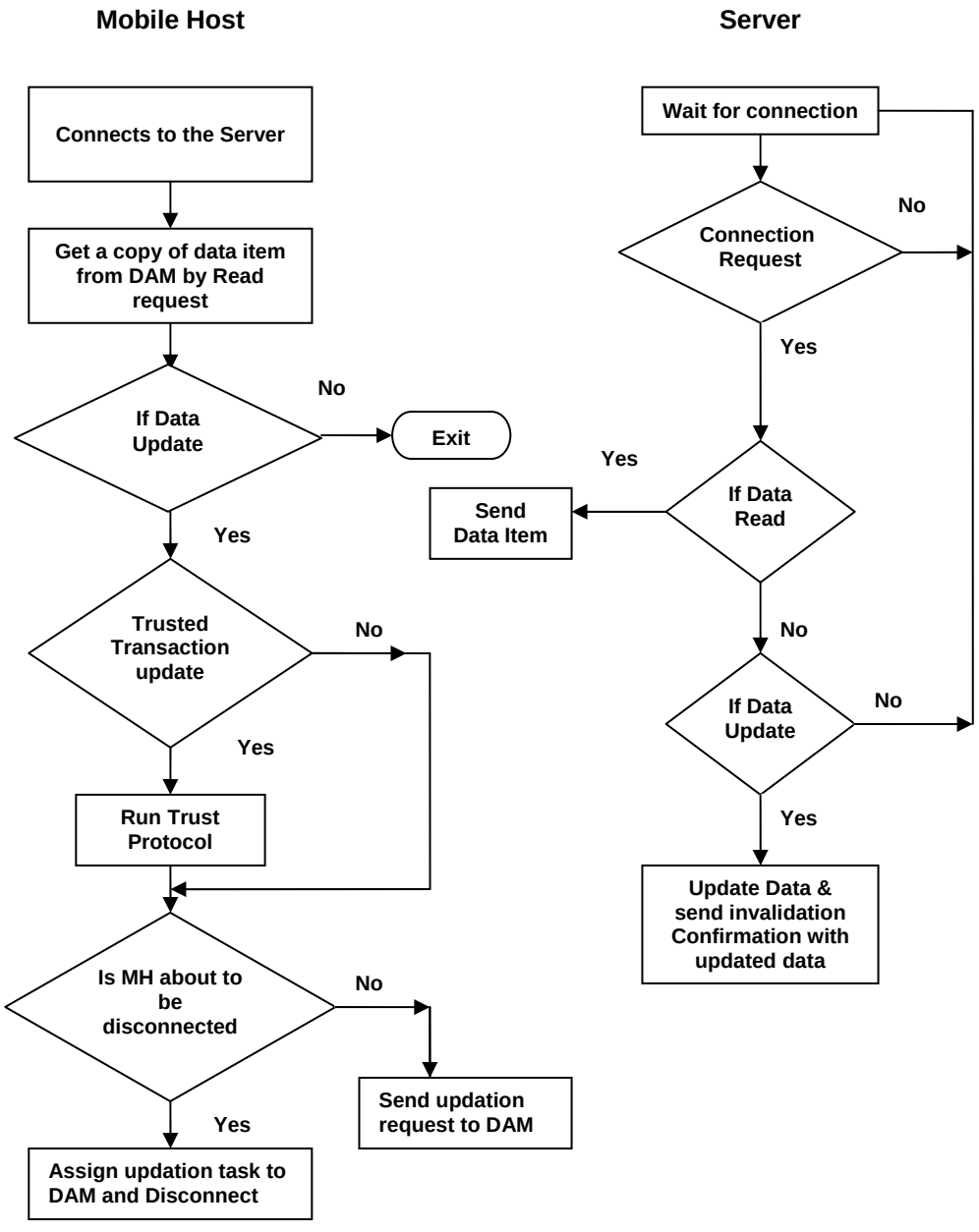

Fig. 2 Flow Diagram for Mobile Host and Server

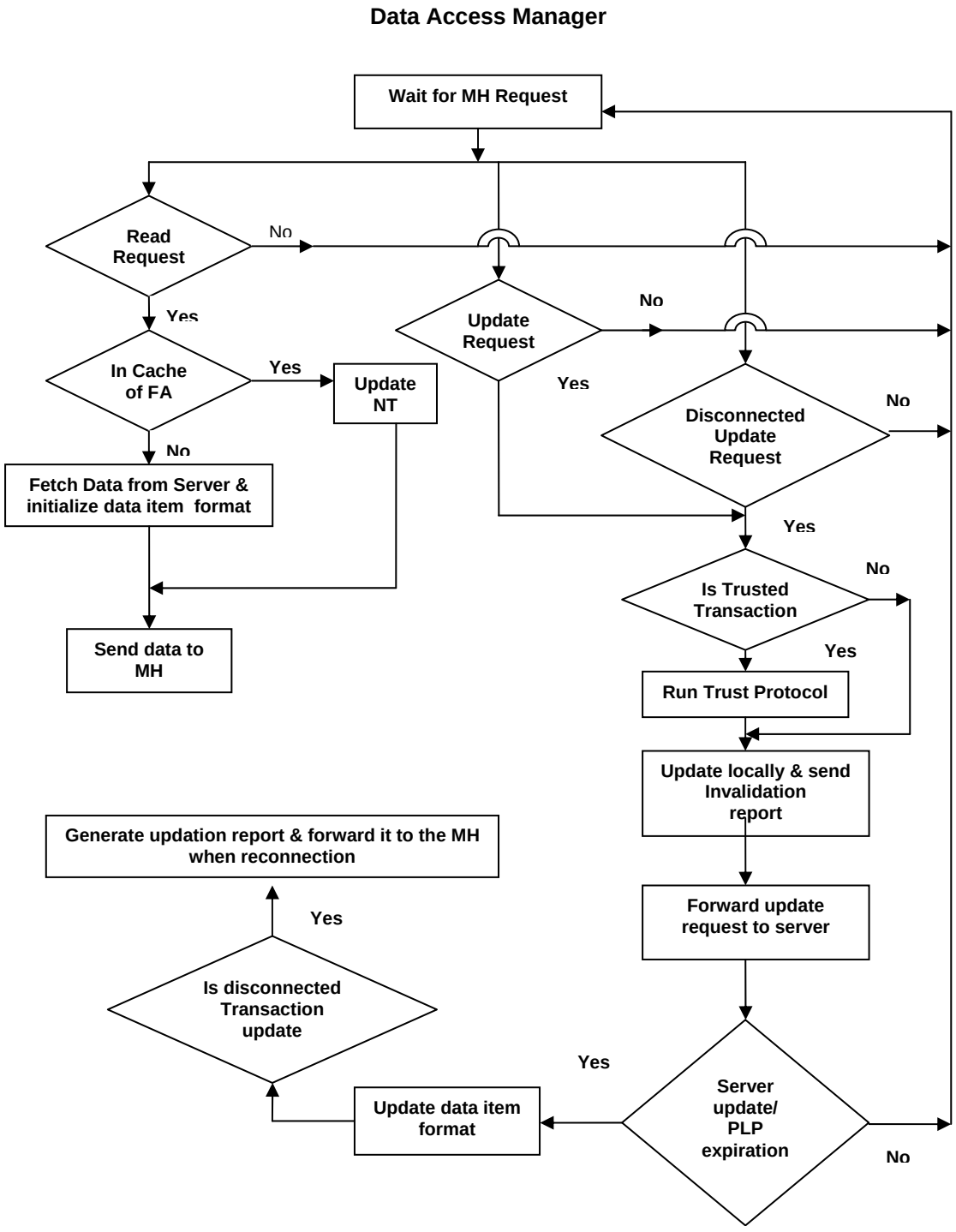

Fig. 3 Flow Diagram for Data Access Manager

item access. Since predicted probability of updation is based on recent past history of updation rate, it is highly probable

that PLP is very close to the actual validity interval of the data item.

B. Transaction execution in the MH

After connecting to the server, MH gets a copy of a data item from the Data Access Manager using read request. If the MH wants to update data, it first checks whether it is a transaction update or not. If trusted transaction update, it runs a trust protocol for authentication. If the MH is about to be disconnected, the updation task is assigned to the DAM before disconnection. Otherwise update request is sent to DAM.

C. Function of Data Access Manager and Server

When MH makes a read request, if it is in the cache of the Fixed Agent, NT is incremented by one. Otherwise, data is fetched from the server and data item format is initialized. Then data is sent to MH.

When MH makes an update request, if it is a trusted transaction update, it runs a trust protocol for authentication. DAM updates data locally and invalidation report is sent to all the mobile hosts that have already accessed the same data item. This forces all the transactions to refresh their data values. This update request is now forwarded to the server. The server updates the data and sends invalidation confirmation along with the updated value. Once Data Access Manager receives the confirmation, it updates the data in the cache. The data in the cache is invalidated if updation is made in the server or PLP expires.

If transaction update is made by the Data Access Manager for the disconnected MH, the above procedure is followed except that at the end, DAM generates updation report and forwards it to the MH when it gets reconnected.

V. PERFORMANCE ANALYSIS

Simulation for the framework is done in Pentium Dual Core System @ 2.4 GHz with 3 GB RAM using J2ME and NS2. The results of the analysis are shown in Fig 4 and 5. Response time is calculated as the time taken to service the request made by the mobile host.

For trusted transactions without disconnection, the response time is more, compared to the non trusted transactions without disconnections. This is due to the extra time required for running the trust protocol. The analysis also shows that the same is true for transactions with disconnections. Also transactions without disconnections take less response time compared to transactions with disconnections for both trusted and non trusted operations.

No. of transactions	Response time in Seconds	
	Non Trusted	Trusted
1	8	9.2
2	5.6	5.9
3	6.1	7.2
4	6.5	7.9
5	7.1	8.4
6	7.4	8.8
7	7.7	9.5
8	8.1	9.9
9	8.4	10.3
10	8.7	11

Table 1 Response time for Transactions without Disconnection

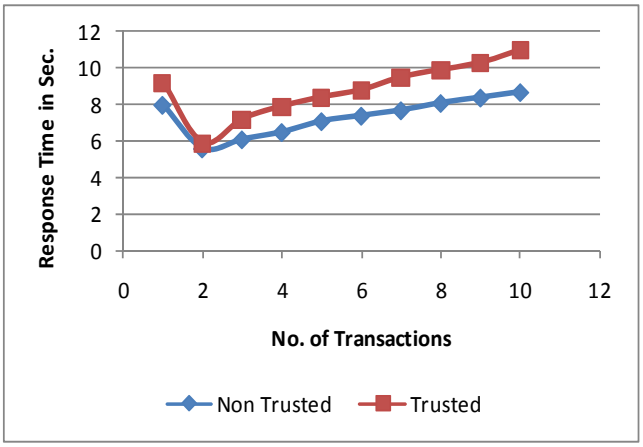

Fig. 4 Analysis of Response time for Transactions without disconnection

No. of transactions	Response time in Seconds	
	Non Trusted	Trusted
1	11.2	14.5
2	9.4	11.6
3	9.6	11.7
4	9.8	12.1
5	10.3	12.5
6	10.7	13.3
7	10.9	13.8
8	11.3	14.4
9	11.7	15.3
10	12	15.9

Table 2 Response time for Transactions with Disconnection

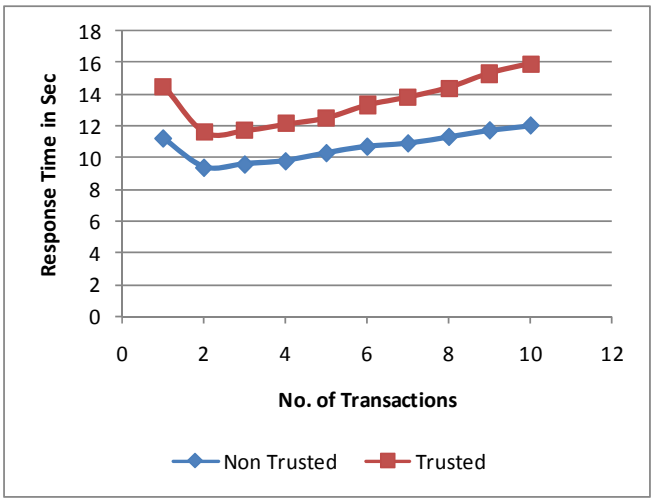

Fig. 5 Analysis of Response time for Transactions with Disconnection

VI. CONCLUSION

In this paper, we have proposed a transaction framework for disconnected mobile computing environment. We make use of Fixed Agents in the wired network to store the cache. This cached data can be accessed by the mobile hosts when they get connected. By using a Fixed Agent and concurrency control with out locking for accessing data, we claim that message communication costs and database update costs are minimized to a larger extent. When mobile hosts are disconnected, transaction update task can be transferred to the Fixed Agent. We also distinguish between non trusted and trusted transactions using a trust protocol.

REFERENCES

[1] Vijay Kumar, *"Mobile Database Systems"*, WILEY INTERSCIENCE, 2006

[2] Victor C.S., Kwok wa Lam and Son, S.H., *"Concurrency Control Using Timestamp Ordering in Broadcast Environments"*, The Computer Journal, Vol.45 No.4, 2002, pp. 410-422.

[3] P.A Bernstein, V. Hadzilacos and N. Goodman, *"concurrency control and Recovery in Database Systems"*, Addison Wesley, 1987.

[4] Ho-Jin Choi, Byeong-Soo Jeong, *"A Timestamp Based Optimistic Concurrency Control for Handling Mobile Transactions"*, ICCSA 2006, LNCS 3981, 2006, pp.796-805.

[5] Madria, S. K., M. Baseer, and S. S. Bhowmick, *"A Multiversion Transaction Model to Improve Data Availability in Mobile Computing,"* CoopIS/DOA/ODBASE, 2002, pp. 322-338.

[6] Le, H. N., and M. Nygård, *"A transaction model for Supporting mobile Collaborative Works,"* IEEE, 2007, pp. 347-355.

[7] Salman Abdul Moiz, Mohammed Khaja Nizamuddin," *Concurrency Control without Locking in Mobile Environments"*, IEEE, 2008, pp. 1336-1339.

[8] Miraclin Joyce Pamila J.C and Thanuskodi K, *"Framework for transaction management in mobile computing environment"*, ICGST-CNIR Journal, 2009, pp. 19-24.