# Mobile Agent Based Multi-hop Transaction Architecture Model

J.L.Walter Jeyakumar
Associate Prof., Dept. of Computer Science,
St. Xavier's college,
Tirunelveli, Tamilnadu, INDIA
walterjeya@gmail.com

***Abstract*— This paper presents a mobile agent based multi-hop transaction architecture model in which mobile users can share the data stored in the cache of a mobile agent. Mobile agent is a special mobile node for coordinating the sharing process. Any MH can communicate with another MH in multiple hops using intermediate MHs in case they are not within each other's communication range. Data Access Manager module at the mobile agent enforces concurrency control using cache invalidation technique. Four levels of priority are assigned to the requesting mobile nodes based on available energy and connectivity. The proposed mobile agent based transaction frame work has been simulated in J2ME and NS2 and the performances are compared.**

**Key words – Transaction, Concurrency Control, Mobile Host, Mobile Agent, Cache invalidation**

## I. INTRODUCTION

Mobile computing environment gives access to information regardless of the location of the user. With the evolution of PCS and GSM and other technologies, advanced wireless communication services are being offered to the mobile users. Mobile Database System is a distributed client/server system based on mobile communication technology in which clients can move around freely while performing their data processing activities in connected or disconnected mode [4].

The mobile computing environment is characterized by high communication latency, mobility, frequent disconnections, limited battery life, and changing client location. In mobile computing, it is necessary that a computation is not disrupted while an MH is not connected. The part of the computation executing on an MH might continue executing concurrently with the rest of the computations while the MH is moving and not connected to the network [10]. Frequent aborts due to disconnection should be minimized in mobile transactions. Blocking of mobile transactions due to long disconnection periods should be minimized to reduce communication cost and to increase concurrency [11]. After disconnection, mobile host should be able to process transactions and commit locally.

A mobile database is a database that resides on a mobile device such as a mobile phone, or a laptop. Such devices are often limited in resources such as memory, computing power, and battery power. Due to device limitations, a mobile database is often much smaller than the database residing on servers. The mobile database can be is distributed among wired and wireless components. Data management responsibility is shared among fixed hosts and mobile units. While using distributed data, there is a need to decide which data to store on the mobile device and which data to store on the Fixed Host. There is also a need to synchronize the data accessed from the device on a central server. Even though mobile users are not constantly connected to a central server, they need to access data nevertheless. Thus data has to be transferred to the mobile device to be managed locally before being synchronized with the original database.

In this paper, a mobile agent based multi-hop transaction frame work is presented that allows mobile users to share data cached in the Mobile Agent which is a special node for coordinating the sharing process. Whenever an MH enters into a Mobile Agent area it can connect and access the data in the cache. Any MH can communicate with another MH in multiple hops using intermediate MHs in case they are not within each other's communication range. 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. This framework also provides the provision for transaction update during disconnection. Data Access Manager (DAM) at the Mobile Agent will take care of concurrency control while sharing takes place. Concurrency control is enforced using cache invalidation technique. In order to give priority to the mobile nodes running on low power and with low connectivity, four levels of priority are used.

The remaining part of this paper is organized as follows. Section II summarizes the related research. Section III focuses on the Mobile Agent based architecture. Section IV presents the proposed mobile transaction management scheme. Section V gives the performance analysis. Finally, Section VI concludes the paper.

## 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 [4]. Moreover, 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 [1]. 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. Transaction Management solution proposed in [7], is for reducing energy consumption at each MHs by allowing each MH to operate in three modes , Active, Doze, and Sleep thus providing a balance of energy consumption among MHs.

In [8] , 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 [9], 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. But this approach did not take into account the disconnection issue. The proposed approach is better than this framework since it transfers transaction execution to the agent in the Fixed network when a Mobile Host is disconnected. It also includes four levels of priority based on energy availability and connectivity in mobile nodes to avoid disconnection

In [12], an agent based real time mobile transaction was presented. In this scheme, mobile nodes can have simultaneous access to data by using cache stored in the fixed agent. The proposed scheme allows mobile agent to be used instead of fixed agent which results in less number of aborted transactions.

## III. Mobile Agent based Multi-hop Transaction Architecture

The proposed Mobile Agent based Multi-hop Transaction architecture (MAMTA) model is given in Figure 1. This model comprises server, Mobile Hosts and mobile agent. An MH can directly connect and communicate with other MHs which are within its communication range. Any MH can communicate with another MH in multiple hops using intermediate MHs in case they are not within each other's communication range. Mobile agent is a special mobile node which connects to the MSS to cache the frequently accessed data from the server. Disconnected Mobile Hosts can connect to the Mobile Agent using short range wireless communication technologies.

In order to provide a balance of energy consumption among MHs and Mobile Agents, an MH will find out the nearest Mobile Agent and submit firm transactions so as meet their deadlines and soft transactions are submitted to the Mobile Agent with highest available energy. The MH can access data from the cache of the Mobile Agent. When a mobile host is disconnected from the Mobile agent after updation request, the updation task is transferred to the Data Access Manager in the Mobile Agent. Data Access Manager module is used to co-ordinate the operations in the cache. After processing the transaction, the Mobile Agent has to find the route to the requesting MH and submit the results.

After disconnected from the server, Mobile Agent can move along with the connected MHs and the MHs can continue their transaction execution. If data update at the server is requested, mobile agent will wait for reconnection before updation is made.

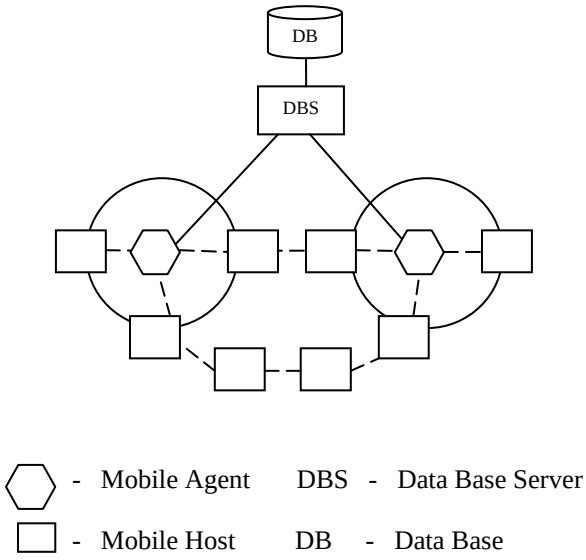


Figure 1 Mobile Agent based Multi-hop Transaction Architecture

In this environment both the user and the Mobile Agent will be moving. So it is necessary to find a route from the MH to the Mobile Agent before submitting a transaction. This environment also requires that transactions of many applications have to be executed within their deadlines. Thus the Transaction Manager at the MH where the database is stored has to consider the mobility of the submitting MHs as well as the deadlines of the transactions. The transaction scheduler has to consider firm and soft transactions with their deadlines in order to minimize abortion of transactions due to deadline violations. The transaction scheduler also considers four levels of priority that are assigned to the requesting mobile transactions based on available energy and connectivity [12].

## IV. MOBILE TRANSACTION MANAGEMENT SCHEME

A mobile agent based multi hop transaction management scheme is illustrated in this section. Frequently accessed data are cached in the Mobile Agent. An MH can directly connect and communicate with the Mobile Agent to access the cached data. The Data Access Manager (DAM) at the Mobile Agent is responsible for enforcing concurrency and cache invalidation. For achieving concurrency control, the concurrency control mechanism which was given in [9] is used. Apart from active, sleep and doze modes, an MH can operate in EC (Energy Conservation) mode [12] in which MH alternates between sleep and doze states during certain time periods.

In this framework, local database in the MHs as well as Mobile Agents consists of id of an MH, position which is used to get coordinates of an MH from GPS and $A_{ij}$ defines status of Energy Availability and Connectivity in an MH. Each Mobile Agent periodically broadcasts its ID and position and MHs will store this information in the MA_list in their local databases.

### A. *Transaction Request by an MH to Mobile Agent*

For submitting a transaction request to the mobile agent, the technique proposed in [7] is used with minor modifications. For firm transactions, time is an important consideration while for soft transactions energy is an important consideration. Therefore, firm transactions are submitted to the nearest MA in order to meet their deadlines while soft transactions are submitted to the MA with highest available energy [7]. Thus if an MH starts a firm transaction, it will find out the nearest MA that is not yet visited by searching its local database. The route to this nearest MA is found out using a LAR [13] route discovery scheme. If MH initiates a soft transaction, it will find out the MA with the highest energy. After finding the route to this MA, it will intimate the status of Energy availability and Connectivity ($A_{ij}$) of the MH to DAM at MA. Then it will submit the transaction request which can be either Data Read or Data Update to this nearest MA.

Now if the nearest MA is in active mode, it will process the transaction. If it is in doze mode and if the transaction is firm, it will wake up and process the transaction in order to

```
Trans_Req_from_MH_to_MA (T,T_type,T_deadline,
MH_pos,MH_ID )
// Transaction T with type T_type and deadline T_deadline is
initiated by an MH whose ID is MH_ID and position is
MH_pos //
Begin
   If T_type is firm
      Search the MA_List to find the nearest MA that is not
                                          yet visited (MA')
   Else
      Search the MA_List to find the MA with highest
                          energy and not yet visited (MA')
   End if
   Find a route to MA'
   Intimate the status of Energy availability and Connectivity
                          (A_ij) of the MH to DAM at MA'
   Start execution of the transaction T locally
   If Data Read
      Submit Read Request to DAM at MA'
   Else If Data Update
         Submit Update Request to DAM at MA'
      End If
   End if
   Set Wait_time = val
   Set timer = Wait_time
   While timer ≠ 0 do
      If the result of T is got
         Find a route to MA'
         Send an Ack to MA'
         Search data item in the latest invalidation report

         If data item exists
            Trans_Req_from_MH_to_MA(T,T_type,
                          T_deadline,MH_pos,MH_ID)
         Else if (A_ij = A_00 or A_01 or A_10)
            Invalidation_Routing()
         End if
         End if
         Commit
         Exit
      End if
      timer--
   End while
   T_deadline = T_deadline – Wait_time
   If T_deadline = 0   // T has missed deadline //
      Abort T
   Else
      Mark MA' as visited
         Trans_Req_from_MH_to_MA(T,T_type,
                          T_deadline,MH_pos, MH_ID)
   End if
End
```

```
Invalidation_Routing()
Begin
    The MH elects the nearest neighbor with A11 (MHp) and
                                                disconnects
    The MHp receives the invalidation report if any and
                        intimates the MH when it reconnects
    Search data item in the invalidation report
    If data item exists
        Trans_Req_from_MH_to_MA(T,T_type,
                        T_deadline,MH_pos, MH_ID)
    End if
End
```

Figure 2 MH execution algorithm for MAMTA scheme

reduce the chance that the transaction will miss its deadline. But if the MA is in EC mode, the requesting MH will wait for some time to get the result of the submitted transaction. If the MH could not receive the result before this waiting time, it will assume that the nearest MA is either in EC mode or disconnected. Hence, the MH will once again search its local database to find the next nearest MA and submit the transaction request after finding the route to this MA. For soft transactions, the MH will wait for some time to receive the results. If the MH does not receive the results within this time period, it will find out from the local database the MA with the next highest available energy.

If the MH moves away after making a transaction request to an MA, it will intimate its current position to the MA. If the result is received by the requesting MH, it will search the data item in the latest invalidation report. If it exists, it will once again make a transaction request. Otherwise, if either energy availability or connectivity of the MH is low, Invalidation_Routing module is called [12]. Here, the MH elects a nearest neighbour with high energy availability and high connectivity ($A_{11}$) to receive invalidation report if any, on behalf of the MH. This nearest neighbour will intimate this invalidation report to the MH, when it reconnects. This algorithm is shown in Figure 2.

### B. *Function of Data Access Manager*

Data Access Manager Algorithm works in the similar manner as given in [12] with minor modifications. When a transaction request is received from an MH, DAM will take the first transaction from the queue. If it is a Read Request and if the data is in the cache of the MA, it will update $T_n$. Otherwise, if the Mobile Agent is connected to the server, it will fetch the data from the server and initialize the quintuple. If Mobile Agent is disconnected, the read request will be put in the queue and it will wait for Mobile Agent to get reconnection with the server. Once reconnection is got, it will fetch data from the server and initialize the quintuple. Now the data is submitted to the MH after finding the route. If update request is made by the MH, DAM will update the data item locally and MA will broadcast the invalidation report to all the MHs that have already accessed the same data item. This forces all the transactions to refresh their data values. In order to forward this update request to the server, it will check whether Mobile Agent is connected to the server. If so, update request is forwarded to the server. Otherwise, the update request will be put in the queue and it will wait until reconnection of Mobile Agent with the server is established. After reconnection with the server, update request is forwarded to the server. The server updates the data and sends invalidation confirmation along with the updated value [12]. Once Data Access Manager receives the confirmation, it updates the quintuple in the cache. The data in the cache is invalidated if updating is made in the server or PLP expires.

## V. PERFORMANCE ANALYSIS

To validate the proposed model, we have developed a simulation test bed using J2ME and NS2 on a Pentium Dual Core System @ 2.4 GHz with 3 GB RAM. The results of the analysis are shown in Figures 3, 4, and 5. Number of transactions missing deadlines, number of completed firm transactions and response time are estimated for the proposed scheme (MAMTA) and compared with and MANET based scheme [7].

### A. *Simulation Conditions*

The MHs are assumed to be located within an area of 1Km × 1Km using random distribution. The total number of Mobile Agents and MHs are taken to be 4 and 100 respectively. The number of MHs per Mobile Agent varies from 5 to 25. Coverage area for Mobile Agent and Mobile Host are 500m and 100m respectively. MHs move with a speed of 5 to 25 meters per second. Mobile agents move with a speed of 1 to 5 meters per second. Response time is calculated as the time taken to service the request made by the mobile host. Mobile nodes move using random walk mobility model [14].

### B. *Results and Discussion*

Figure 3 gives the comparative analysis of a number of transactions missing deadlines for MANET based scheme and MAMTA scheme. As number of transactions increases, compared to the MANET based scheme [7], Mobile Agent based scheme (MAMTA) shows a decline in number of transactions missing deadlines (less number of transaction abortions). This is due to the fact that the Data Access Manager at the Mobile Agent enforces concurrency control using cache invalidation technique which results in less number of aborted transactions. Whereas in MANET based scheme, locking based approach for concurrency control results in more number of transactions abortions. Moreover, the use of EC mode enhances the chances of completion of transactions before deadline without abortion. For example, for 50 transactions, MAMTA scheme shows an improvement of 64% over MANET based scheme.

In Figure 4, the number of completed firm transactions for varying number of firm transactions is shown. It is found that compared to the MANET based scheme, MAMTA scheme allows more number of firm transactions to get completed. This is due to priority based scheduling of transactions which allows more number of firm transactions to get completed before deadlines.

Figure 5 shows the comparative response time analysis for MANET based scheme and MAMTA scheme. It is observed that Mobile Agent based scheme (MAMTA) takes less average response time compared to the MANET based scheme. This is due to the presence of cache in the Mobile Agent which allows transactions to finish quickly.

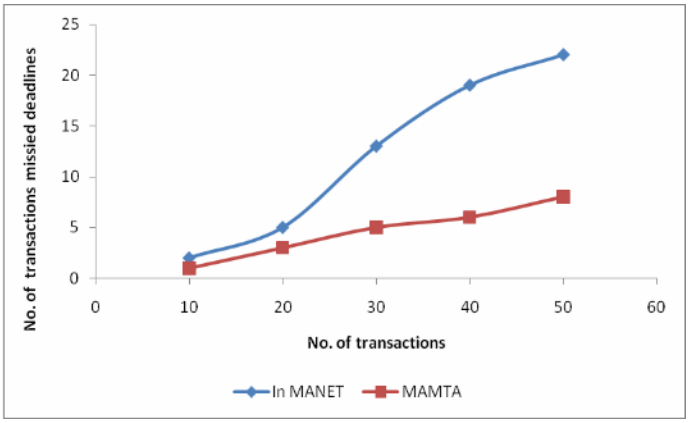


Figure 3. Comparative analysis of number of transactions missing deadlines for MANET based scheme and MAMTA scheme

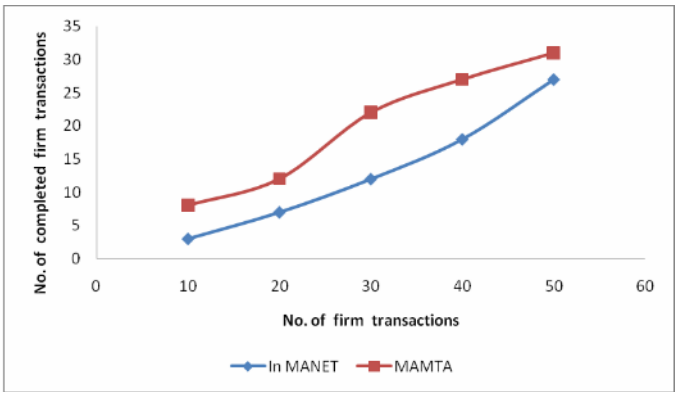


Figure 4. Comparative analysis of number of completed firm transactions for MANET based scheme and MAMTA scheme

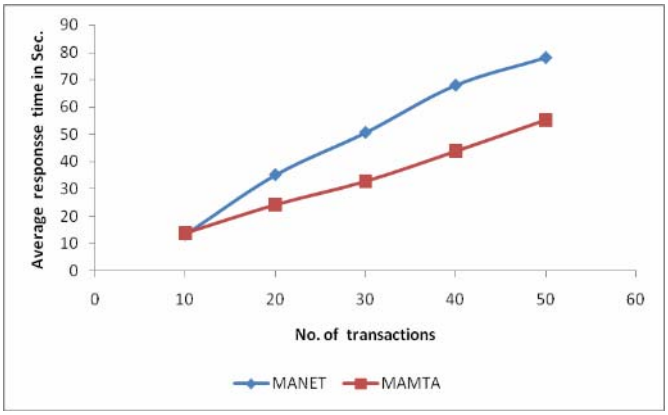


Figure 5. Comparative analysis of average response time for MANET based scheme and MAMTA scheme

## VI. Conclusion

In this paper, Mobile Agent Based Multi-hop Transaction Architecture model has been proposed. Mobile Agent can form a work group with the disconnected Mobile Hosts using short range wireless technology. The frequently accessed data are cached in the Mobile Agent. This cached data can be accessed by the mobile hosts when they get connected. This transaction framework is simulated and their performances are compared.

## References

[1] Ho-Jin Choi, Byeong-Soo Jeong, "A Timestamp Based Optimistic Concurrency Control for Handling Mobile Transactions", *ICCSA 2006, LNCS 3981*, pp. 796-805, 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, pp. 410-422, 2002.

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

[4] Vijay Kumar, "*Mobile Database Systems* ", Wiley Interscience, 2006.

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

[6] Le, H. N., and M. Nygård, "A transaction model for Supporting mobile Collaborative Works", *IEEE*, pp. 347-35, 2007.

[7] Le Gruenwald, Shankar M. Banik, Chuo N. Lau, "Managing real time database transactions in mobile ad-hoc networks", Springer, pp. 27-54, 2007.

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

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

[10] Chrysanthis P.K., "Transaction processing in a mobile computing environment", *in Proc*. IEEE Workshop on Advances in Parallel and Distributed Systems, October 1993, pp. 77–82.

[11] Sanjay Kumar Madria, Mukesh Mohania, Sourav S. Bhowmick, Bharath Bhargava, "*Mobile data and transaction management*", Elsevier Information Sciences [4], pp. 279-309, 2002.

[12] J.L. Walter Jeyakumar and R.S. Rajesh, "Agent Based Energy Aware Real Time Mobile Transaction Management", IRACST – International

Journal of Computer Networks and Wireless Communications (IJCNWC), vol.3. No 3, pp.274-280, 2013.

[13] Y.-B. Ko and N.H. Vaidya, "Location-aided routing (LAR) in mobile Ad-Hoc networks", Proceedings of the IEEE /ACM International Conference, Mobile Computing and Networking MOBICOM '98, pp. 66–75, 1998.

[14] T. Camp, J. Boleng, and V. Davies, "A Survey of Mobility Models for Ad-hoc Network Research", Wireless Communications & Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications, vol. 2, issue: 5, pp. 483-502, 2002.