A NEW HIERARCHICAL TRANSACTION MODEL FOR MOBILE ADHOC NETWORK ENVIRONMENT

Vikram Bali[1] Rajkumar Singh Rathore[2]
[1]Asstt. Prof, Department of CSE, Galgotias College of Engg. & Tech., Greater Noida
[2]Sr. Lecturer, Department of CSE, Galgotias College of Engg. & Tech, Greater Noida

ABSTRACt*-Mobile Computing Environment poses its unique challenges to the existing Transaction models which are fail to solve. The main challenges of mobile computing environment are its heterogeneous environment, low bandwidth and power resources. The transaction must be able to handle frequent disconnection because mobile user can move anywhere.*

In this paper, I presented a transaction model for mobile adhoc network environment. In this hierarchical model transaction is performed in distributed fashion by the nodes in a MANET. Basically three types of nodes in this transaction model-Captain, Player and Data Manager. Data manager is maintained into hierarchy-Global, Zonal and Local. Players play the role under the supervision of Captain. Mobile Nodes can access data directly with Local data Manager. The data manager is used maintaining the data and log for recovery.

In the previous model for transaction there is no criterion for recovery, in this model I resolve this problem with hierarchical structure of data manager.

Key Word: MANET, DSM-CTM, System Architecture.

INTRODUCTION

Mobile Adhoc Network is future technology; various challenges are superimposed by this technology. MANET inherited the challenges from cell architecture in addition bandwidth and highly dynamic topology and battery back up problem. MANET is used where no infrastructure is available for communication such like disastrous area, military application, sensor network.

A mobile Transaction is structured as a Distributed transaction. In which the transaction is completed by the help of mobile nodes and some fixed nodes. Fixed nodes are used to hold the data and mobile nodes are to initiate the transaction. The mobile environment produces the significant challenges to transaction processing. The wireless networks provide limited bandwidth so network bandwidth is a scarce resource. Battery power drains with data transmission and transaction processing.

RELATED WORK

A few models are proposed for capturing these challenges. Dunham [3] suggests a model that is known as Kangaroo Transaction Model, where mobile nodes are basic unit for transaction initiation. This is further extended to handle data source by data access agent. Data access Agent reside on mobile support station (MSS) and work on the behalf of mobile units which lie in the range of host MSS. The Transaction normally hops from one MSS to another MSS as mobile units move. However model does not discuss about Recovery.

Crysanthis [1] considers the mobile transaction as a multi database transaction and introduces the additional notion of reporting and co-transactions. [1] Introduces a transaction proxy concept. Here a proxy run at MSS corresponding to each transaction and ensures the backup at the mobile hosts. Pitoura and Bhargava [7,8] propose a transaction wherein they consider mobile transaction as an issue of consistency in a global multi database which is divided into clusters. In [9] the model works on semantics based transaction and consider mobile transaction as a cache coherency problem and concurrency in a distributed database.
Yeo and Zaslavksy [10] suggests for disconnected transaction processing and allows the local management of transaction via Compacts. There are many Transaction model exist no single best approach emerged.

PROBLEM FORMULATION

In this paper we proposed a transaction model that will provide high recovery and solve challenges of disconnectivity. This will also support long lived transactions. A cooperative transaction, in which mobile nodes work in collaborative manner with a common objective of processing the query initiated by one node. This Cooperative Transaction (CTM) works like an Army Command, where commander (captain) coordinates the soldiers (Players) and data about each soldier and Commander is stored at own command, this data is send at the Brigade and the data from all Brigade is collected at a Central office. Here in my Transaction Model Data Manager is divided into three levels as discussed in above Army data Management.

The Development of new Transaction model is posed by challenges of Ad-hoc Network. In this paper, solution for the previous problem is formulated.

1. The transaction should support distributive and collaborative processing.
2. Transaction will be long-lived.
3. The Transaction model should support Recovery of data.

WORKING OF PROPOSED DSM-CTM MODEL

The primary entities of Cooperative Transaction Model are:

Captain: A Transaction that supervises and coordinates the work among other nodes in a cluster of the MANET.

Player: A sub-transaction that is allocated by captain to a node in a cluster. The Player works on its allocated sub-transaction.

Data Manager: It is more robust node with low mobility. It also keeps logging information for recovery when Captain crashed. Captain accesses data from data manager. Data manager always keep track of Captain. Basically there are three level of data manager.

1. **Local Data Manager:** Local Data Manager are assigned to Captains. Captains can access the data from Local Data Manager. Local Data Manager also maintains log for recovery purpose when Captain crashes. Mobility is no constraint here.
2. **Zonal Data Manager:** Zonal Data Manager coordinates the Local Data manager, Many Local Data Manager can be assigned to a single Zonal Data Manager. It also maintains the log for recovery when Local Data Manager crashes. So within the zone Local Data Manager can move without any problem.
3. **Global Data Manager:** Global Data Manager coordinates all the zonal data manager. The Zonal Data Manager can move anywhere in the work but with this effect communication increases.

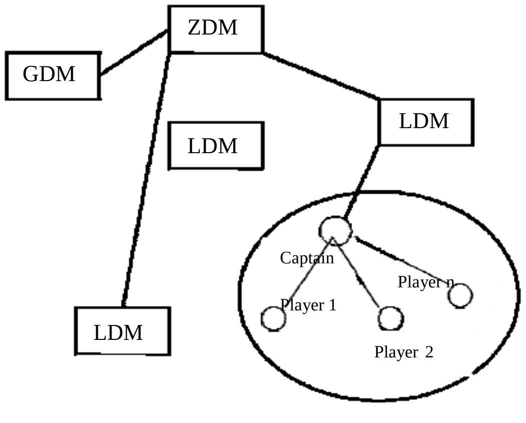


Fig. 1. A Hierarchical data model

A coordinated transaction is initiated by nodes that issue a database query which is big enough for the node to execute handedly, as well as distributive in nature.

A cooperative transaction springs up to handle a query. The node looks for the players that are eager to volunteer for completing the job. Each of the player reports the captain after completing the job the captain being the first. The captain looks for the Local Data manager before allocating the sub-transaction to the players. A node that play vital role in case of crashes for recovery.

A player node reports the update to Captain. The player node can not commit the update in database. But only delegates the commits to Captain. It is up to Captain whether it commit updates to the database. Thus this is the responsibility of Captain to update the Database.

This model can be further subdivided, if A player hold a transaction that big enough to handle. It can also work as a Captain and subdivide this transaction to other players and so on. This is shown in below Figure 2.

CLUSTER, CAPTAIN AND PLAYER TRANSACTION

A coordinated Transaction can be collection of sub-transaction called *Cluster*. Where each Cluster contains only one Captain Transaction and one or more Player Transactions. In fact it is possible that a cluster contains only one transaction that is a Captain transaction for lower level cluster.

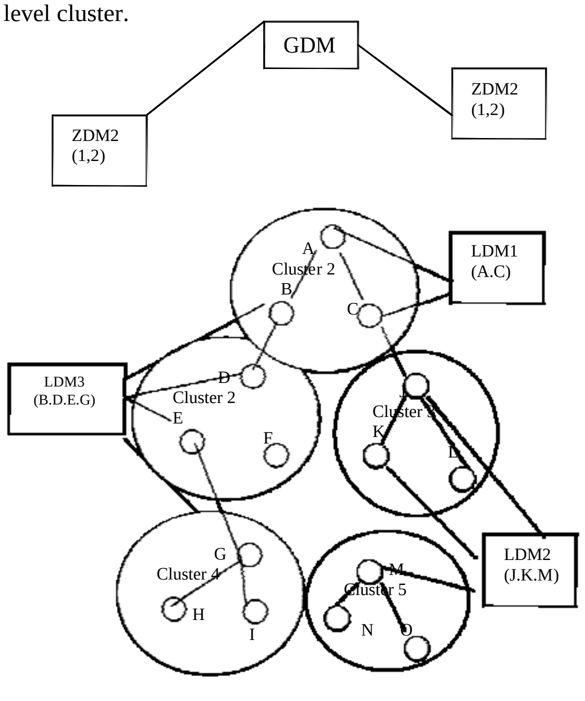


Fig.2 Generalized DSM-CTM

The Captain’s Transaction can only interact with LDM, Player transaction interact with captain only. The Captain transaction can change data in LDM. Captain transaction itself maintain a log for recovery and a log is also maintained at LDM for recovery purpose when captain crashes. The LDMs interact with ZDM, one more than one

LDMs can interact with ZDM. Though a log is also maintained at ZDM for recovery purpose because LDM can also crashes. LDM is fully mobile and ZDM is semi-mobile, They can move within the network. GDM is a type of semi-mobile node, The ZDMs can interact with GDM we can also maintain log at GDM for much recovery.

SIGNIFICANT EVENTS

Many events can occur for transaction management. These events can occur many times at the time of transaction completion. Various such events are accept, begin, commit, abort, spawn, Assign, split, kill.

ASSIGN EVENT AND ASSIGN SET

Assign event is the Captain event that is used to assign the work to players. The set of operations that take part in Assign events are known as Assign Set.

REPORT EVENT AND REPORT SET

Report Event is player event by which player report to its Captain delegating the work done by it. The set of operations performed by player on completing the work is called as Report Set.

SPLIT EVENT

This event occurs when a node delegates its unfinished work to some new node and move out of the cooperative transaction. A Captain as well as player can split at any point of time.

ACCEPT EVENT AND ACCEPTED EVENT SET

This is the event of Captain Transaction; this is performed after the report event by the player transaction for accepting the work done by player. The Captain transaction can accept whole or some part of the report set, it can also reject. This accepted set becomes the Accepted Report set.

SYSTEM ARCHITECTURE

The Architecture proposed for underlying system is described in this section. The system has to generic enough to reflect the generic feature of all the three entities namely, The Captain, Player and Data Manager. As shown in Figure 3, a node meets the system requirement through a number of component modules. Depending upon the role of the node the system requirement can be of two types. A node which

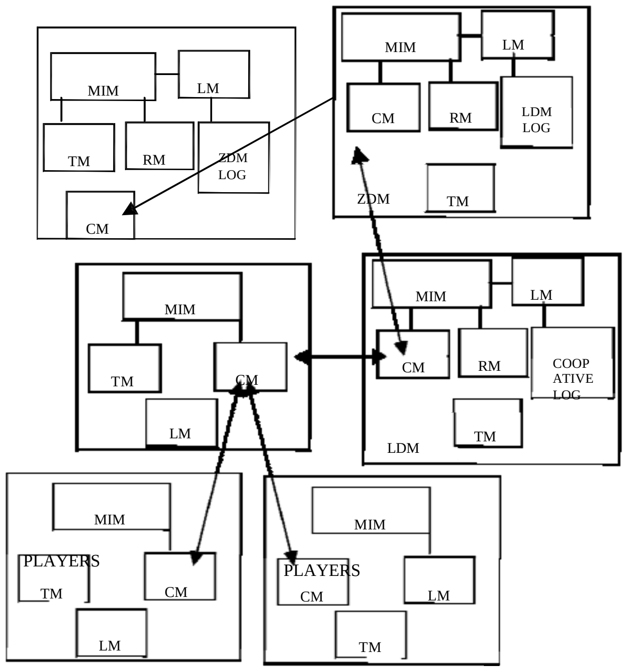


Fig. 3 System Architecture for cooperative transaction model

Data Manager is responsible for recovery of crashes nodes. Any other node can be called as simple mobile node. A simple mobile node can start transaction, help other node in processing small part of transaction as a player node or generate a query.

MIM	- Module Interfacing Manager
TM	- Transaction Manager
CM	- Communication Manager
LM	- Log Manager
RM	- Recovery Manager

A Brief description of all entities inside a node follows.

Module Interfacing Manager (MIM)

The MIM provides an interface among all other components of the node. The MIM is responsible to keep the track of all transactions running on the node with their role in the transaction. It may happen that a coordinator sub-transaction of one node running on the node with player sub-transaction of another node on the same node.

LDM is responsible for maintaining the log of operations performed by Captain. MIM at LDM has to keep the track of all the Captains for whom the LDM is providing service, detect the crash of Captains and assigning the work of crashed Captain to other node which is willing. MIM at ZDM has to keep the track of all the ZDM for whom the GDM is providing service, detect the crash of

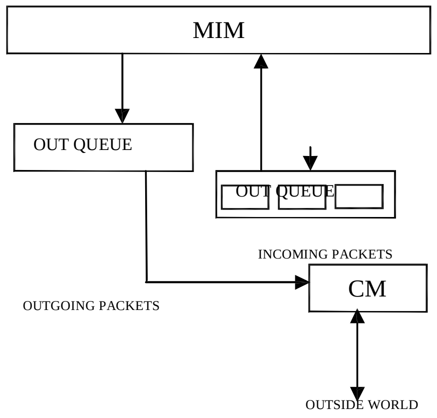
ZDM and assigning the work of crashed ZDM to other ZDM which is willing.

Transaction Manager (TM)

The TM provides a set of operations that are required to performed by a transaction or sub-transaction. It provides basic primitives that are used by transaction such as spawn and report to execute the application. It ensures the logging of operations by using significant events in relation with LM and CM.

Communication Manager

This module facilitates communication among all the nodes. The packets arriving towards the node are inserted into a queue in FIFO manner. It also responsible for pushing the packets onto the net.



Log Manager

The operations that are executed by the player transaction are logged by local at that node. This log information is flushed into the Captain transaction for maintaining the log of coordinated nodes. The log information from the Captains log is flushed into a LDM which manage the log information for those captains which are getting service from that particular LDM. The log information from the LDMs log is flushed into a ZDM which manage the log information for those LDMs which are getting service from that particular ZDM. The log information from ZDMs is sent at GDM which maintains log for ZDMs.

Recovery manager (RM)

The RM work at the DM. It is used fro recovery of the transactions. It is used at DM because the database is maintained at the DMs.

Proposed CTM algorithm:

LDM:

```
Begin (LDM)
        Read packets from IN_Queue;
        Switch (Packet_Type)
        Begin

        Case REQ_LDM;
        Create Log for the requesting Captain;
        Reply back in affirmation;
        Set Alive Timer that detects a node crash;
Case LOG_FLUSH:
        Reset Alive Timer;
        Append the Log;
Case COMMIT:
        Remove the Log entry of the sender node;
        Remove Alive Timer;
Case CANCEL_LDM:
        Remove Log entry of the sender node;
        Remove Alive Timer;
Case REP_REC_CAPTION: Cancel Alive Timer;
        Send Acknowledgment if chosen for
        recovery; End;
End;
```

ZDM:

```
Begin (ZDM)

        Read packets from IN_QUEUE;
        Switch (LDM_Packet_Type)
Begin
Case REQ_ZDM:
        Create Log for the requesting LDM;
        Reply back in affirmation;
        Set Alive Timer that detects a node crash;
Case LOG_FLUSH:
        Reset Alive Timer;
        Append the Log;
Case COMMIT:
        Remove the Log entry of the sender node
        Remove Alive Timer;
Case CANCEL_LDM:
        Remove Log entry of the sender node;
        Remove Alive Timer;
Case REP_REC_LDM:
        Cancel Alive Timer;
        Send Acknowledgment if chosen for recovery;

        End;
End;
```

GDM:

```
Begin (GDM)

        Read packets from IN_QUEUE;
        Switch (ZDM_Packet_Type)
        Begin

Case REQ_GDM:
        Create Log for the requesting ZDM;
        Reply back in affirmation;
        Set Alive Timer that detects a node crash;
```

```
Case LOG_FLUSH:
        Reset Alive Timer;
        Append the Log;
Case COMMIT:
        Remove the Log entry of the sender node;
        Remove Alive Timer;
Case CANCEL_LDM:
        Remove Log entry of the sender node;
        Remove Alive Timer;
Case REP_REC_ZDM:
        Cancel Alive Timer;
        Send Acknowledgement if chosen for recovery;
        End ;
End;
```

CAPTAIN

```
Begin [Captain]

If (new application stared)
Begin

        Read the JOB and DB files;
        Create new application;
        Assign CTID to it;
        Create new Log entry;
        Send request for LDM;
        Send request for Players;
End;
        Read packets from IN_QUEUE;
        Switch (Packet Type)
Begin
Case REP_PLAYER:
        If(distributed environment available)
Begin
        Assign a unique PTID;
        Add it into list of pending JOBS;
        Set Player response Timer;
        Send work;
Case REP_LDM:
        Set application's LDM;
        Initialize Log;
        Set Log Flush Timer;
Flush Log to LDM Periodically;
Case LOG_FLUSH:
        Reset Player response Timer;
        Append Log;
Case ABORT:
        Reset Player response Timer of the sender Player;
        Modify its states in the list of pending jobs;
        Reassign the work to Player Transaction;
Case ROLLBACK:
        Modify its states in list of pending jobs;
        Reply the work to Player nodes;
Case DELEGATE:
        Remove player response time;
        Modify its states in list of pending jobs;
        Accept report;
Case NODE_CRASHED:
        Cancel Player response time of concerned player
Case REQ_REC_CAPTAIN:
If (not engaged in the same transaction)
        Reply in affirmation ;
Case ACK_REC_CAPTAIN:
        Add CTID into list of selected CTIDs;
Case REC_WORK:
        Construct a new application;
        Proceed to execute;
        End;

End;
```

Player:

```
Begin [Player]
        Read packet from IN_QUEUE;

        Switch (packet_type)
Begin
Case REQ_PLAYER:
        If (not engaged with the same CTID)
        Reply back information;
CASE WORK:
        Set log flush timer:
         If (work can be distributed)
        Cal Player;
        Execute operations;
Case KILL:
        Remove the application for the PTID;
        Case REPLY:
        Restart application;
Case REQ_REC_CAPTAIN:
        If (not engaged in the same transaction)
        Reply in affirmation;
Case ACK_REC_CAPTAIN:
                Add CTID into list of selected CTIDs;
Case REC_WORK:
        Construct a new application;
        Proceed to execute;
        End;
End;
```

Recovery Mechanism:

```
 Begin [Recovery mechanism]
        Broadcast node crash information;
        Broadcast a request for new captain;
        Get the log of crashed node;
        Get the database items;
        Get the instruction set of node;
Analyze log to create a table of all involved (Sub)
transaction;
        Redo updates;
Begin
Remove entries of non-committed sub-transaction from
log;
Modify the data base items with the last updated value;
        End;

        Using the state ID's of the database items,
compose the instruction set;
        Get ID of new captain;
        Send the updated database items and modify
instruction set to the new captain;
        Set active Timer for new Captain;
End;
```

CONCLUSION

In this paper we propose a CTM for transaction management in mobile adhoc network environment. Result of this paper improves the recovery of data related to transaction and very good for long lived transaction for distributed transaction processing in comparison to previous works, CTM is very useful where no compromises in data loss. Further work can include the security operations on transaction in MANET.

REFERENCES

1. CHRYSANTHIS, P. Transaction Processing in Mobile Computing Environments in IEEE workshop on Advances in Parallel and Distributed Systems (1993).
2. MOHAN,C., HANDERLE,D., LINDSAT, B., PIRAHESH, H., SCHWARZ, P. Aries: A transaction Recovery Method supporting fine granularity locking and partial rollback using write ahead logging. In ACM Transactions on Database Systems, VOL. 17 NO. 1 (March 1992). Pp 94 162.
3. DUNHAM, M.H., HELAL, A., AND BALAKRISHNAN, S.A Mobile transaction that captures both data and movement behavior. In ACM-Baltzer Journal on Mobile Networks and Application, VOL. 2 (1997). Pp 149 162.
4. Gruenwald, L., Javed, M., and Gu, M. Energy-Efficient Data Broadcasting in Mobile Ad-Hoc Networks. In Proc. International Database Engineering and Applications Symposium (IDEAS'02), July, 2002.
5. G. Pei, M. Gerla, X. Hong, and C. Chiang. A wireless hierarchical routing protocol with group mobility. In Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC), pages 1538-1542, 1999.
6. W. Navidi and T.Camp. Stationary distributions for the random waypoint mobility model. IEEE Transactions on Mobile Computing, pages 99-108, 2004.
7. PITOURA, E., AND BHARGAVA, B., Revising Transaction concept for mobile computing. In first IEEE workshop on mobile computing systems and applications (June 1995). Pp 164 168.
8. Pitoura E., and Bhargava B., maintaining consistency of data in mobile distributed environments. 15th International conference of distributed computing system (1996), pp 404-413.
9. Walborn G., and Chrysanthis P., supporting semantics based transaction processing in mobile data base application 14th IEEE symposium on reliable distributed system (1995), PP 31-40.
10. Yeo L. and Zaslavksy A. submission of transaction from workstation in a cooperative multi database processing environment, 14th ICDCS-1994.
11. Laslie D. Fife, Le Gruenwald, Research Issues for Data Communication in Mobile Ad-hoc Network Database System, SIGMOD Record, Vol.32,NO.2,June 2009