

TEAM TRANSACTION MODEL

Team transaction model represents the idea of executing transactions in a single cell having mobile hosts. A subset of these hosts will form a team to do a transaction in a collaborative manner. We call the mobile support station (MSS) in which all these mobile hosts lie as bench in the team transaction terminology. Usually a number of mobile hosts (MH) are found under a cell. Thus a number of mobile hosts can connect to a single bench. A bench picks one uncommitted transaction and assigns it to one of the mobile hosts in its cell. This action is performed until no more uncommitted transactions can be found or no free host is available. The MH which gets a transaction from the bench is called a coordinator for that transaction.

The coordinator splits the transaction in subtransactions to form player transactions, and assigns these sub transactions to other willing MHS. Since all nodes are within the same cell, a call setup between them is inexpensive. While decomposing a transaction the coordinator takes into account the dependencies among the sub transactions. The MH running the coordinator for a transaction and the MHS executing the player transactions of that coordinator constitute a team for that transaction, and hence the name team transaction. The player MHS perform the transaction assigned to them by the coordinator and communicate with coordinator through messages. Finally, commit message is sent to the coordinator. The coordinator also exchanges the same type of message with the bench. The main tasks are performed at different levels of control for the completion of a transaction, and involves following main actors:

1. Bench
2. Coordinator
3. Mobile host
4. Recovery system

Bench

The Bench has a list of transactions that are to be executed, and the list of MHs which are active under its cell. The bench picks up a transaction and assigns it to one of free mobile host. This MH is referred to as the coordinator of that transaction. Each transaction is represented by a unique number TTID, and each mobile host by an identifier called MHID. The coordinator id is known as CID. As part of the strong recovery strategy used for this transaction model, the bench maintains a data structure called action buffer. The bench receives messages from the coordinator and performs messages logging in the action buffer. The message logging is useful when recovery is needed. As each TTID is given to a CID, the bench maintains a (TTID - CID) list. When a message comes from the coordinator, it checks TTID, originator ID, etc., in the message for the purpose of identifying, and storing the message at the appropriate position in the action buffer.

Action Buffer :-

The action buffer is organised in form of multilists. For each TTID one entry is maintained. There may be a list of TTIDs and each TTID contains a list of PTIDs, the player transaction IDs assigned for a particular TTID. For each PTID the action buffer by the coordinator. The coordinator sends the message informing which PTIDs were assigned for a particular TTID. For each PTID the action buffer contains a list of messages sent by the MHID having the PTID assigned by the CID of corresponding TTID. So when a message is received at the bench then it checks the TTID, PTID in that TTID list and stores the message the corresponding PTID list. When a commit is

reached at the bench it commits that TTID, i.e. the entries in the corresponding action buffer are made permanent. Until the time commit has not reached, the messages in the log are only temporary. Table 1. describes various type of messages that may be received by bench and the corresponding actions that must be performed at the bench in response.

message type	Action by the Bench
COMMIT	make the entries in the action buffer permanent
SPLIT DELEGATE	choose another coordinator for that TTID
Rollback	Remove entries in the action buffer asked by coordinator
Data	Log message in action buffer

Table 1: Messages received by bench & the corresponding action.

II. Coordinator :-

The main task of coordinator is to decompose a transaction with a given TTID into a number of player transactions, thus generating a number of PTIDs. The coordinator has to maintain the dependency among the PTIDs. Depending upon the dependency order the PTIDs are executed one by one and if no dependency means the PTIDs can be executed parallelly. Each PTID is given to a free MHID. The coordinator can also be a player for the transaction it is executing if no free MHID is available. Similarly, the same node can also be a player for many other TTIDs. After assigning the work to each player (MHID) coordinator expects some messages from its players. It performs necessary actions

depending on the type of message it receives. For purpose of processing the received information CID needs to store some data structures.

1. It should have PTID list for the given TTID
2. PTID-commit list for the purpose of checking which PTIDs have committed

When a player sends a message to the coordinator it checks the type of message. Depending on the message type the PTID-commit list is modified and if all the players have given commit then final commit is passed to the bench for that TTID. This is the scenario when MHIDs resides in the same cell upto the completion of the job assigned to it. Since this transaction model does not restrict the mobility of mobile hosts, there may be situation when MH leaves the cell without committing its job.

Timeout for each player :-

The coordinator maintains a timer for each mobile host for which it assigned a PTID. If this times out then the corresponding PTID is checked for commit and if that is not the case, then the same task is assigned to another MHID in the cell or the coordinator itself does the unfinished job. This case solves the problem of a mobile host crash.

Split-Delegate message by player :-

When the mobile host is leaving the present cell then it sends Split-Delegate message. Then the coordinator assigns the work to another player. No message from the MHID after it had send a Split-Delegate message which is accepted.

Table 2 gives a summary of the actions performed by the coordinator upon receiving various messages from the players.

message Type	Action at CID	message to bench
Delegate	check if all PTIDs are finished	If Yes send Commit message
Data	Log the message ID	send the message
Split Delegate	Choose another MH and give the remaining work	send Rollback message with message IDs logged for uncommitted PTID

MOBILE HOST :-

The work of the mobile host is to execute the work given by the coordinator and sending the information to the coordinator through messages. Due to the message overheads, the host does not send any message until some useful data is generated. The structure of message is provided in Table 3.

Message Id	Id for the message generated, each message will be having distinct Id.
MHID	Identify of MH from which message is originated
CID	Coordinator ID for the MHID
PTID	Assigned PTID for the MH.
Type	Type of message: Data, Commit, Delegate, Split-Delegate
Data	It is the information field

A normal message is marked as **DATA**. When the subtransaction is complete at the MH, it sends a message of the type **DELEGATE**. If the host leaves the current cell without completing the work then it sends a message of type **SPLIT-DELEGATE**.

RECOVERY

Since the hosts are mobile, they may go out of cell any time and also there are the chances of crashing. To handle these situations team transaction model maintains the stable storage log at the bench referred to as action buffer. It reflects the present state of each transaction, this model can guarantee recovery at the mobile host level, the coordinator level and also at the bench.

Recovery at the mobile host:

As the host may be busy with work assigned to it, and it may send wrong data to the coordinator. In such situation, it has to undo what it has done. This is known as local rollback and generally happens with less probability. In this case no message is sent or received. It is done by the mobile host itself without the intervention from the bench or the coordinator.

Recovery at Coordinator:

The mobile host, to which PTIDs are assigned may crash or migrates to a new cell without completing the task. Then the work done by them remain unfinished. The unfinished work should be removed and must be assigned to another players. As the data is stored at the bench, the coordinator should give rollback to the bench for the work done by a crashed player or a player which has left the cell. For this purpose coordinator must store message IDs sent by the PTIDs that are not committed. When the commit message is sent by a player the corresponding storage is freed. So coordinator gives a rollback message to bench with TTID, PTID, start-messId, End-messId. On receiving the rollback message bench removes the

Corresponding entries in order to maintain the consistency.
So this recovery involves the processing at the coordinator
and the bench.

Recovery at Bench:-

As the coordinator itself is a mobile host, but ~~not~~ in
some special ~~properties~~ ^{cases}, it may also crash and also leave the
current cell. This situation is handled by bench by
choosing another MHID as coordinator and removing the
work done by the previous coordinator. This may degrade
the performance. Therefore the coordinator is chosen from
among the MHs which are expected to stay in the
current cell for a duration longer than the expected
completion time for a transaction.