CSE-302: Mobile Transaction Models

Dr. R. B. Patel

Mobile Database

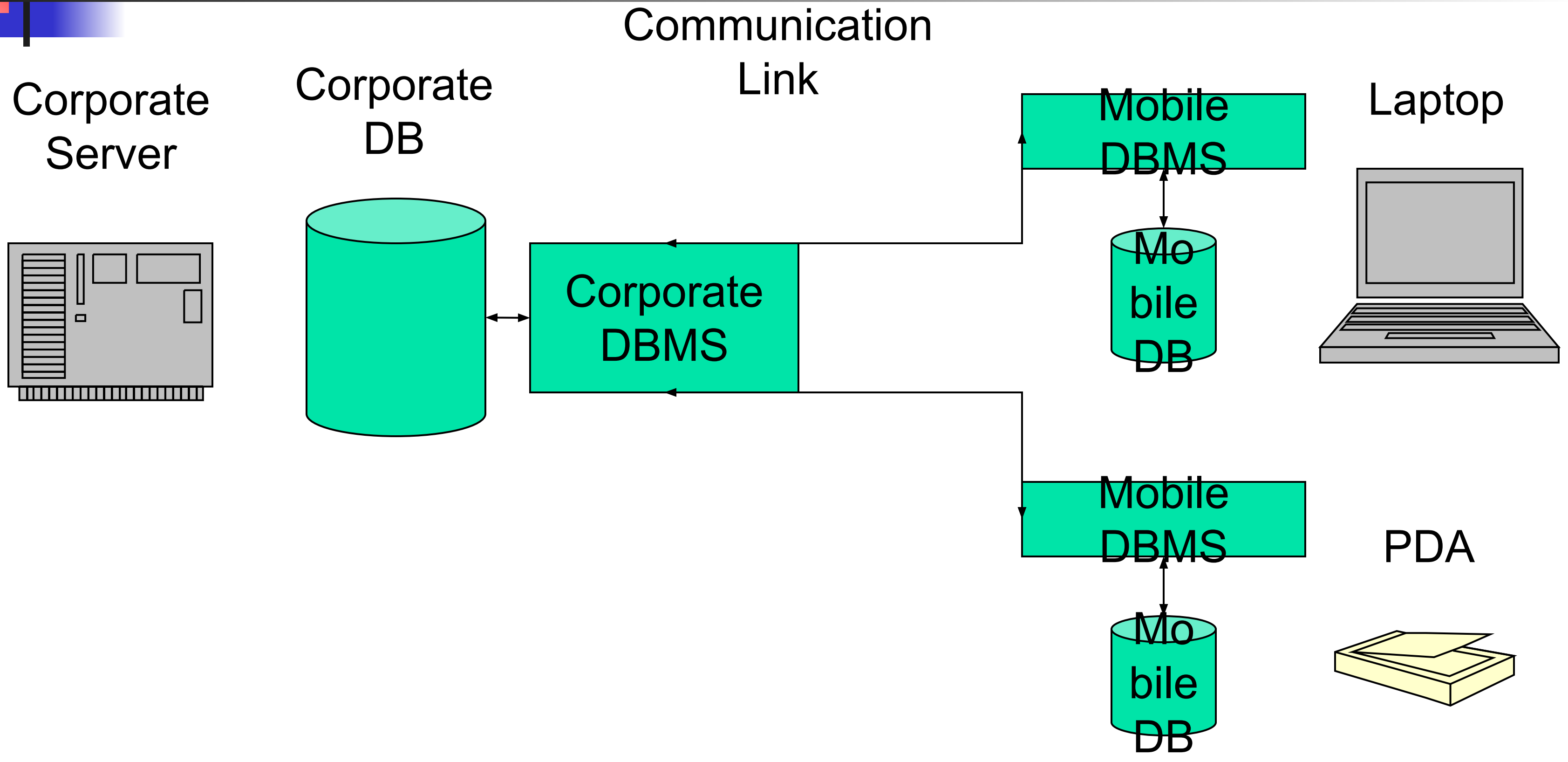
- A database that is portable and physically separate from the corporate database server.
- But **Mobile Database** is capable of communicating with that corporate database server from remote sites allowing the sharing of corporate database.

Continue…

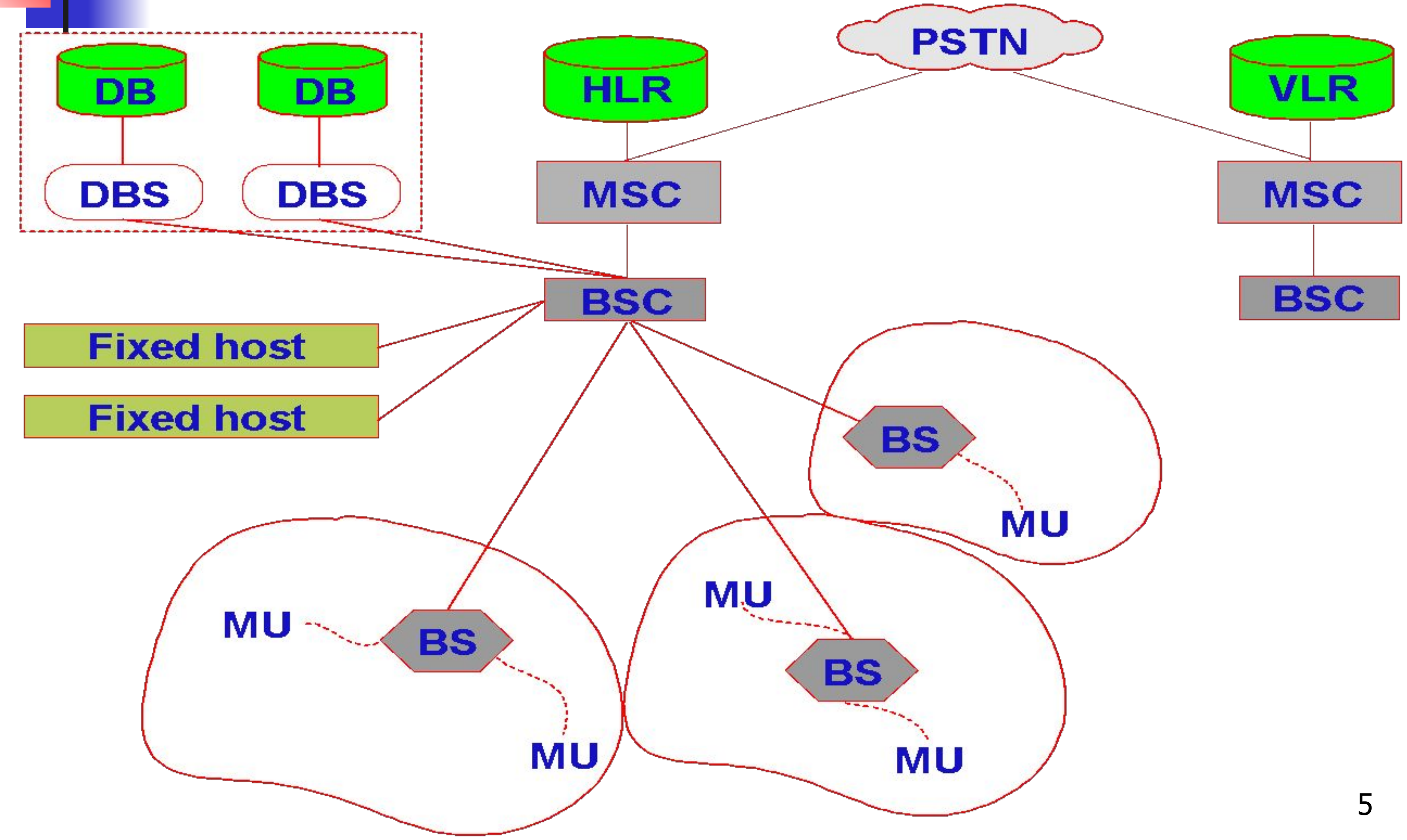
Components of a Mobile Database

- Corporate Database Server
  - Manages and stores corporate data, provides corporate applications
- Remote Database/DBMS
  - Manages and stores mobile data, provides mobile applications
- Mobile Database Platform
  - Laptop, PDA, etc.
- Two-Way Communication Link Between Corporate and Mobile Database Servers
  - Often, but not always, wireless

Continue…


Communication Link
Corporate Server
Corporate DB
Corporate DBMS
Mobile DBMS
Mobile DB
Laptop
Mobile DBMS
Mobile DB
PDA

Mobile database Integrated System



Requirements of a Mobile DBMS

- Communicates with a centralized database server in a mode such as:
  - Wirelessly
  - Using Internet
- Replicates and Synchronizes data on the centralized server and mobile host (MH).
- Can capture data from various sources such as the Internet
- Manages and Analyzes the data on the MH
- Can create custom mobile applications

Current Mobile DBMS

- Current most mobile DBMSs only provide limited prepackaged SQL functions for the mobile application.
- It is expected that in the near-future, mobile DBMSs will provide functionality matching that at the corporate site.
- Some Mobile DBMSs
  - Microsoft SQL Server CE
  - Oracle Lite Edition

Transaction

- A set of operations that translate a database from one consistent state to another consistent state,
- That is a computation processing is considered as a transaction or conventional transaction if it satisfies ACID (Atomicity, Consistency, Isolation, and Durability) properties.

Transaction : Atomicity

- An executable program, assumed that this program will finally terminate, has one initial state and one final state.
- If the program achieves its final state it is said to be committed,
- otherwise if it is at the initial state after some execution steps then it is aborted or rollback.

Transaction : Consistency

- if a program produces consistent result only then it satisfies the consistency property and it will be at the final state or committed.
- If the result is not consistent then a transaction program should be at the initial state, in other word the transaction is aborted.

Transaction : Isolation

- if a program is executing and if it is only single program on the system then it satisfies the isolation property.
- If there are several other processes on the system, then none of the intermediate state of this program is viewable until it reaches its final state.

Transaction : Durability

- If a program reaches to its final state and the result is made available to the outside world then this result is made permanent.
- Even a system failure cannot change this result.
- In other words, when a transaction commits its state is durable.

Transaction (Contd.)

- The programming model of a transaction will be:

```
Begin_transaction ()
 Execution of transaction program
 If (reach_final_state) then
      Commit_Work(final_state)
 Else
        Rollback_Work(initial_state)
```

The simplest form of transaction is flat transaction. A flat transaction can be considered as a sequential correctness computer program. Every execution step is after one another.

Mobile Transaction

- **Flexibility can be introduced using workflow concept. Thus, a part of the transaction can be executed and committed independent to its other parts.**

- A transaction where at least one MH takes part in its execution.
  - Difficult to enforce ACID properties in mobile transactions.
  - Thus, new models are being created to deal with mobile transactions

Continue…

Execution scenario:

- User issues transactions from his/her MH and the final results comes back to the same MH.
- The user transaction may not be completely executed at the MH so it is fragmented and distributed among database servers for execution.
- This creates a Distributed mobile execution.

Continue…

A mobile transaction (MT) can be defined as

T_i is a triple <*F, L, FLM*>; where

$F = \{e_1, e_2, \ldots, e_n\}$ is a set of execution fragments,

$L = \{l_1, l_2, \ldots, l_n\}$ is a set of locations, and

$FLM = \{flm_1, flm_2, \ldots, flm_n\}$ is a set of fragment location mapping where $\forall j,\ flm_i\ (e_{ij}) = l_i$

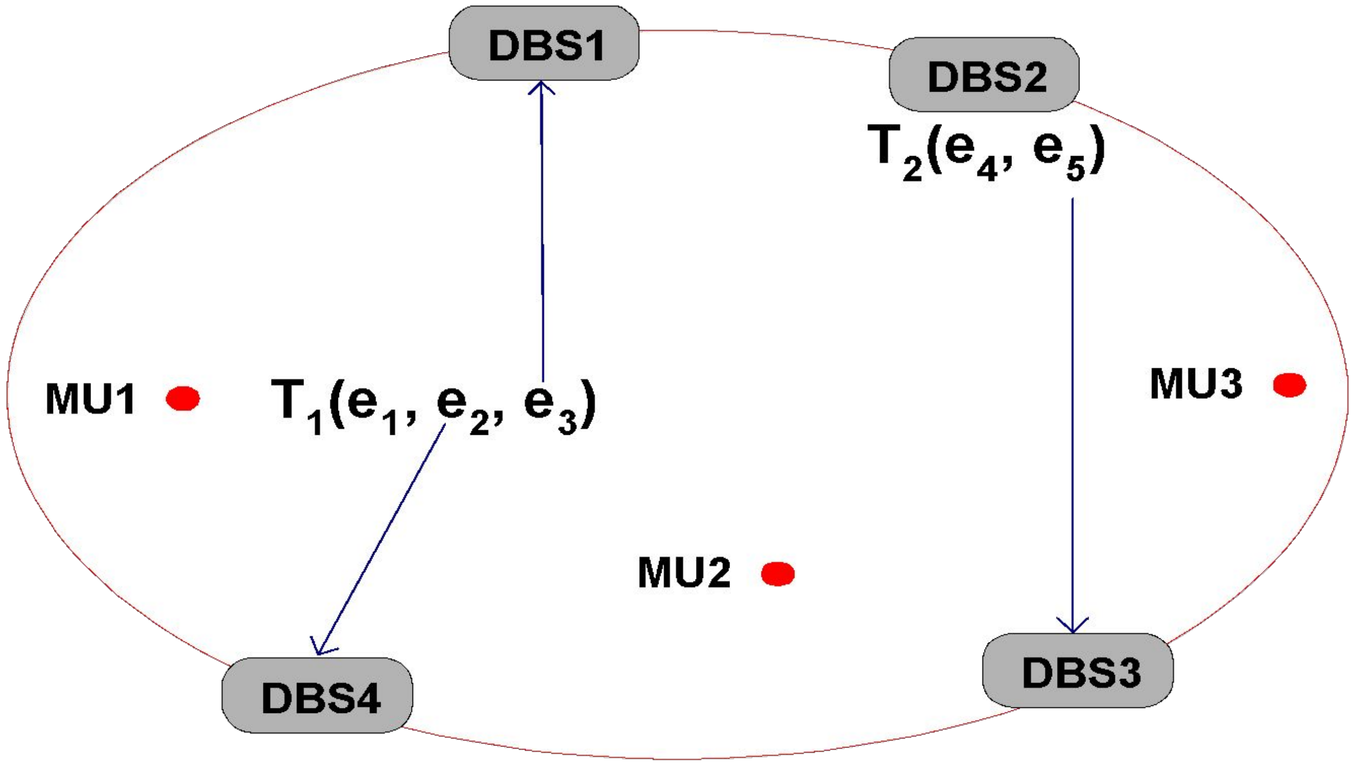
Continue…

An execution fragment e_{ij} is a partial order $e_{ij} = \{\sigma_j, \leq_j\}$ where

- $\sigma_j = OS_j \cup \{N_j\}$ where $OS_j = \cup_k O_{jk}$, $O_{jk} \in \{read, write\}$, and N_j $\{Abort_L, Commit_L\}$.
- For any O_{jk} and O_{jl} where $O_{jk} = R(x)$ and $O_{jl} = W(x)$ for data object x, then either $O_{jk} \leq_j O_{jl}$ or $O_{jl} \leq_j O_{jk}$.

Continue…

Mobile Transaction execution.



Kangaroo Transaction (KT)

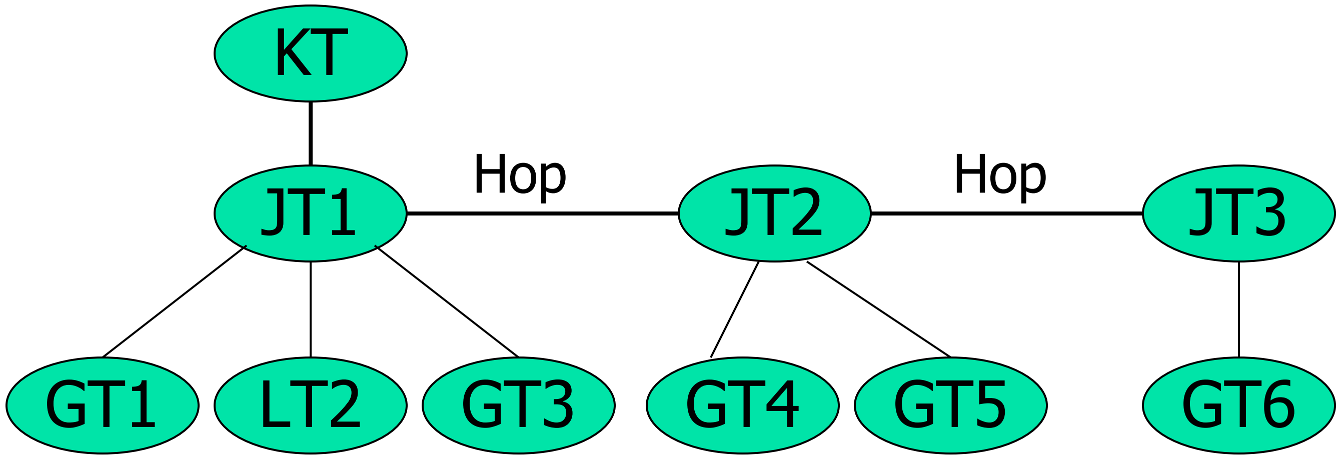
- Mobile transactions (MTs) are generated at the MH and entirely executed at a Multi-Database System (MDBS) on the wired network.
- Built using concepts of open-nested and split transactions.
- **The management of the transaction moves with MH.**
- Addresses the movement of MHs during transactions.
  - Transactions in a mobile environment can hop from one BS to another as the MH moves.
- ACID compliance is responsibility of each DBMS.

Kangaroo Transaction (contd.)

- Uses a Data Access Agent (DAA) at each BS to manage mobile transactions and the movement of the MH.
- Mobile transaction's execution is coordinated by the BS with the MH is currently assigned.
- When MH hops from one cell to another, coordination of the mobile transaction moves to the new BS.
  - Original transaction is split into Joey transactions (JT)
  - One JT at each base station
  - Each BS coordinates the operations that are executed while the MH was in its cell.

Kangaroo Transaction (contd.)

- Three Layers: source system, data access agent, and the mobile transaction
- Handoff between base stations
- Two Processing Modes: compensating and split
- Data access agent maintains transaction information



Kangaroo Transaction (contd.)

- DAA (Data Access Agent) acts as transaction manager at base station
- For each transaction request DAA generates
  - A Kangaroo Transaction (KT) at MH
  - A set of Local Transactions (LTs) & Global Transactions (GTs) at local base station called as Jeo Transaction (JT).
- For each hop a new Joey is created.
- When a Joey fails, all previous Joeys and KT will abort.

Movement and Disconnections

- Kangaroo Transactions
  - Data Access Agent (DAA) tracks MH movement by maintaining a linked list of all BS that have been coordinators of the KT.
  - List will be used in case of cascading aborts.
- KT adds another layer to existing multi-database architecture to manage transactions requested by MH.
  - Coordination is distributed along all BS that the MH visit.
  - Reduces communication cost during execution because always controlled by the BS whose cell the MH is in.
  - In case of cascading aborts, communication increases greatly however
    - Controlling BS must forward aborts to all BS in the list

Team Transaction model

- Designed to deal with mobile transactions in Ad hoc networks

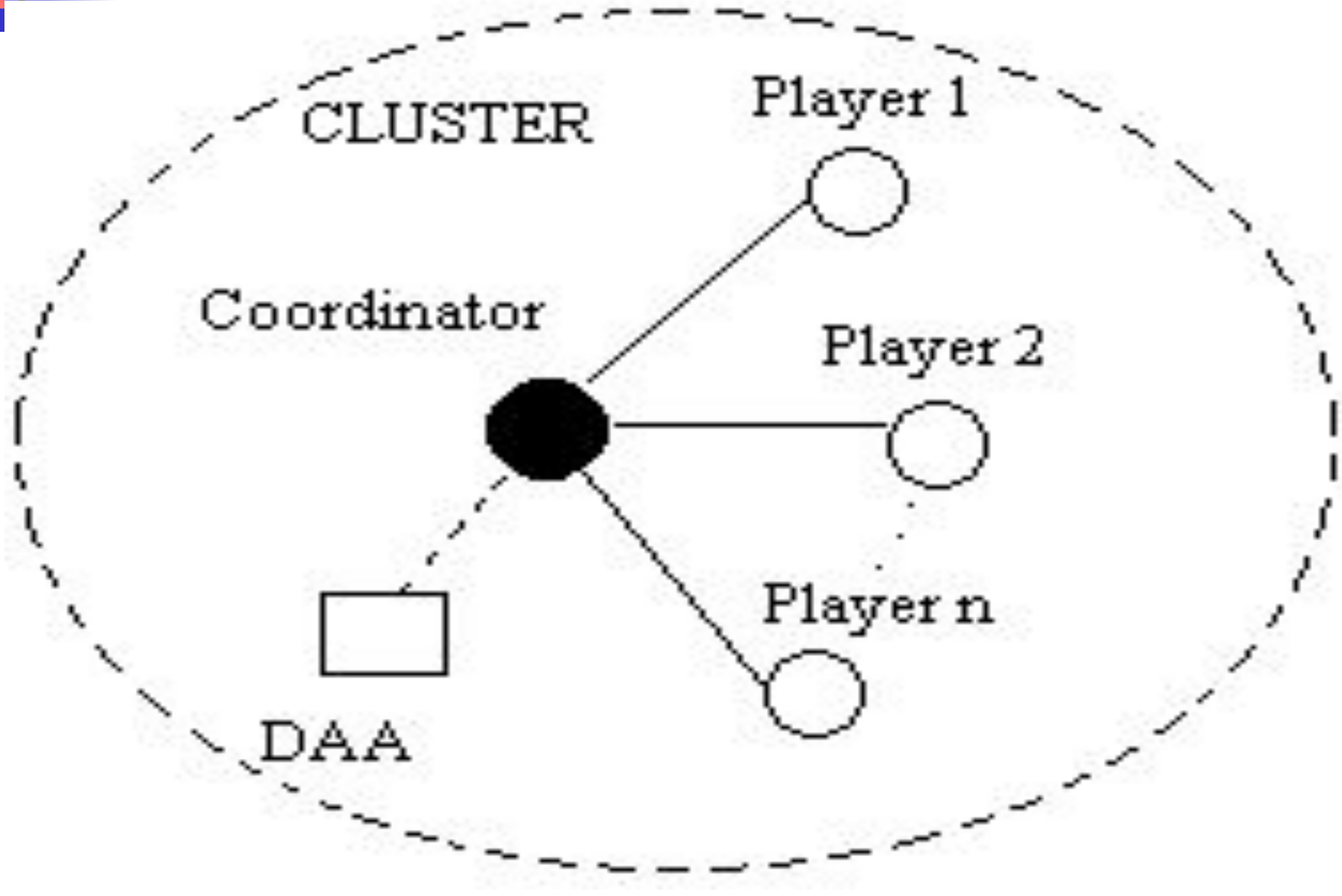
- Team Transaction consists of three entities:
  1) Coordinator 2) Players 3) Data Access Agent (DAA). These entities form a cluster.

- Coordinator is captain of the team and responsible for coordinating the operations of transactions.

- Players carry the operations of sub-transactions assigned by coordinator.

- DAA provides database access, maintains log information for recovery, and keeps track of the coordinator and selects new coordinator incase of crash.

Continue…



Database update to maintain global consistency

- Database update problem arises when MHs are also allowed to modify the database.
- To maintain global consistency an efficient database update scheme is necessary.

Transaction commit

- In **Mobile Database System** (MDS) a transaction may be fragmented and may run at more than one nodes (MH and DBSs).
- An efficient commit protocol is necessary.
- 2-phase commit (2PC) or 3-phase commit (3PC) is no good because of their generous messaging requirement.
- A scheme which uses very few messages, especially wireless, is desirable.

Timeout Based Protocol: Continue…

- **One possible scheme is "timeout" based protocol.**
- **In timeout scheme MH and DBSs guarantee to complete the execution of their fragments of a mobile transaction within their predefined timeouts.**
- **Thus, during processing no communication is required. At the end of timeout, each node commit their fragment independently.**

Continue…: Protocol: TCOT-Transaction Commit On Timeout

Requirements

- **Coordinator:** Coordinates transaction commit
- **Home MH:** Mobile Transaction (MT) originates here
- **Commit set:** Nodes that process MT (MH + DBSs)
- **Timeout:** Time period for executing a fragment

Continue…

- MT arrives at Home MH.
- MH extract its fragment, estimates timeout, and send rest of MT to the coordinator.
- Coordinator further fragments the MT and distributes them to members of commit set.
- MH processes and commits its fragment and sends the updates to the coordinator for DBS.
- DBSs process their fragments and inform the coordinator.
- Coordinators commits or aborts MT.

Transaction recovery

- Complex for the following reasons
  - Some of the processing nodes are mobile
  - Less resilient to physical use/abuse
  - Limited wireless channels
  - Limited power supply
  - Disconnected processing capability

Continue…

- Desirable recovery features
    - Independent recovery capability
    - Efficient logging and checkpointing facility
    - Log duplication facility

Continue…

- When executing transactions in mobile environment, it is necessary to maintain logs to enable recovery after a system crash.
- A MH is highly vulnerable to failures due to the loss or theft of equipment, memory loss, etc.
- In order to recover from such failures, it is necessary to store data objects and their logs at the mobile service stations (MSS) rather than on mobile host (MH).
- A MH transfers a transaction's execution to the MSS by moving all the prewrite values and the log records.
- A separate pre-commit log is maintained for each transaction.

Continue…

- Each log record contains the description of the prewrite values.
- These are appended to the pre-commit record of a transaction.
- Thus, for every pre-committed transaction, the precommit log records for that transaction are stored on the disk as well as in the system log at MSS.
- In case a MH moves to a different cell, it can also append a record indicating its next possible destination.
- When the user arrives at the new cell, he/she can continue the post precommit operations.

Continue…

- Once a transaction's execution is successfully shifted (along with all **prewrite** logs and **pre-commit** log) to the MSS, a MH may delete all the log records and keep only prewrite values in main memory for further processing.
- This way it can gradually discard entries in prewrite logs.
- To build highly reliable systems, these logs can also be replicated in various cells (MSS) by moving MH.
- At MSS, prewrite-logs, pre-commit log record, the write logs and final commit log record are stored.
- If a pre-committed transaction fails at the stationary host, the transaction can be restarted from the point of failure.
- In case of a system crash, the prewrite logs can be used to build the system's state as it existed at the time of pre-commit.

Continue…

- Write logs can help to bring the system to the state as it existed at the time of failure with the help of recovery algorithms.
- Since the locks are managed by MSS, the locks are maintained at MSS.
- The transaction table for active transactions are kept at MSS.
- However, in case the transaction starts its execution at MH then a part of the transaction table is also maintained at MH until the transaction's execution is shifted to the MSS.
- Dirty-data object table is kept at MSS. This table contains information about those objects whose final values are inconsistent with the stable database on the disk.

Continue…

- This also keeps information about those data objects whose prewrite values announced by the pre-committed transactions have not been updated in the database before system crash.
- A log records and the tables stored at MH and MSS.
- Some of these logs are moved to MSS from MH during the shift of a transaction's execution.
- In case of a system crash, only the redo of those prewrite and write values will be performed whose effects are not there on stable storage.
- There is no need for undo pass of the recovery algorithm as data objects are updated only after the transaction is pre-committed.

Log records at MH and MSS.

Log records stored at MH	**Log records stored at MSS**
Prewrite logs	Write log
Pre-commit log	Commit log
'Destination' move log Lock.	dirty object and transaction tables

Recovery Protocols

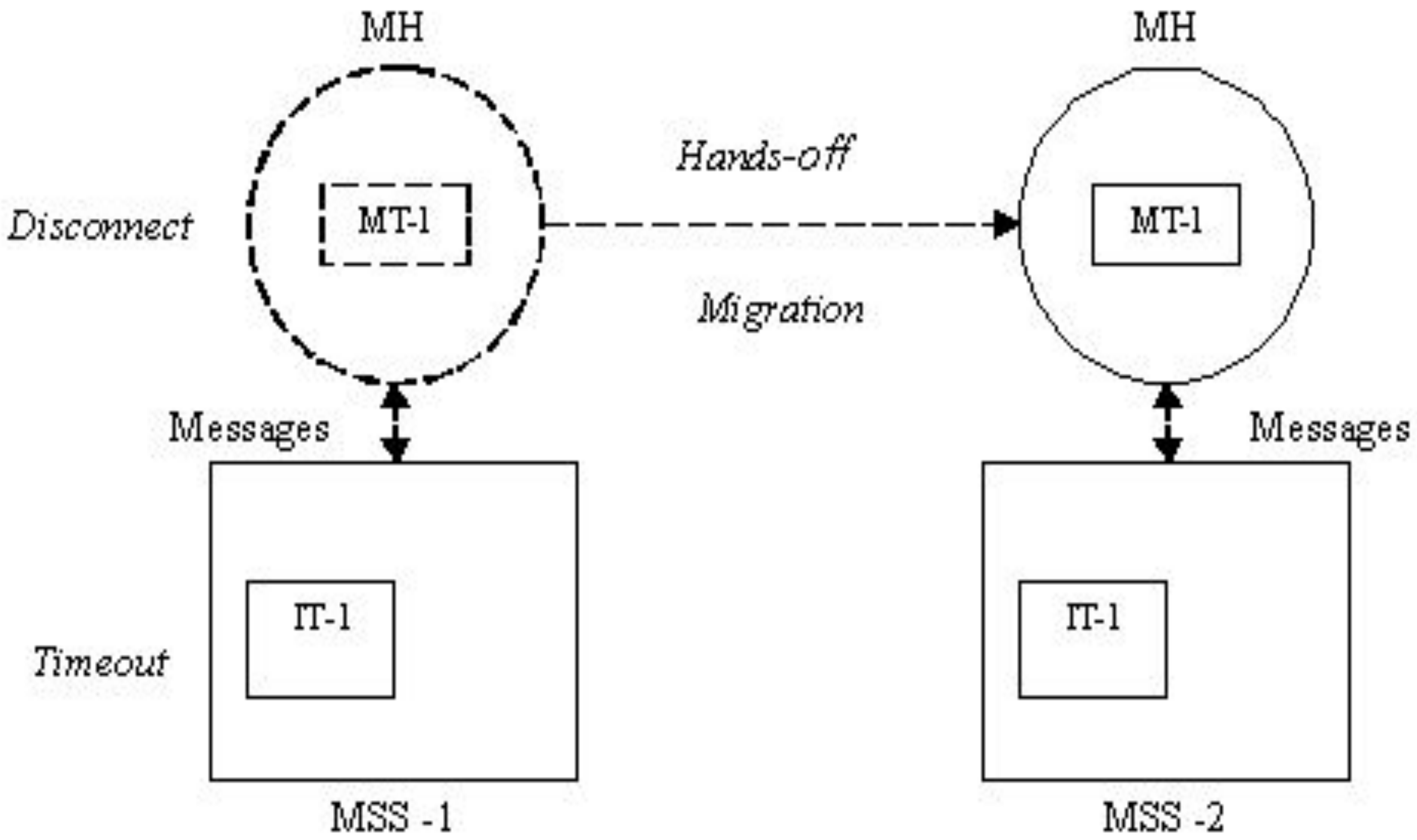
1. ***Timeout protocol:*** executed by MSS. MSS maintains a timer to measure the inactivity period of MH and initiates rollback for the transaction on timeout.

2. ***Disconnect protocol***: executed by MH due to resource problems (like battery discharge, weak signal, etc.).

3. ***Hand-off protocol:*** executed by MH, when it switches from one MSS to another MSS. MH sends it's new MSS address to the old MSS and conveys old MSS information to the new MSS while switching from cell to cell.

4. ***Migration protocol:*** In this protocol migration information and new settings of MH are communicated to the old MSS by the new MSS, before timeout or disconnect protocol execution at old MSS

Mobile Transaction Recovery Protocols



Recovery Guarantees

Recovery guarantee is a recovery assurance of one subsystem (eg.MSS) to another subsystem (e.g. MH) in case of failures.

Protocol:

```
    if p succeeds
    then q  will also succeed, if invoked
```

-The guarantee of assurance by a subsystem is up to its capabilities.

-The guarantee can be given by any subsystem other than the system where operation *p* is executed.

-Recovery protocols are prescriptions based on the recovery guarantees of the system. These protocols satisfy precedence constraints.