Transaction Processing Issues in Mobile Computing

- Extended Transaction Models
- Exploiting Object Semantics
  - Fragmentable and Recoverable Objects [Walborn and Chrysanthis]
  - Reservation Algorithms [Elmagarmid, Bukhres, Jing]
- Lock Management
  - Locks acquired in one place and released in another [Jing, Bukhres, Elmagarmid]

Traditional Definition of Transactions

- Transaction - Unit of Consistency
- The ACID Test for a Transaction
  1. Atomicity
  2. Consistency
  3. Isolation
  4. Durability

Transaction Properties

- **Atomicity**
  - Either all or none of the operations associated with a transaction happen
- **Consistency**
  - A transaction is a correct program segment
  - Integrity constraints of the database satisfied at transaction boundaries

Transaction Properties (Continued ...)

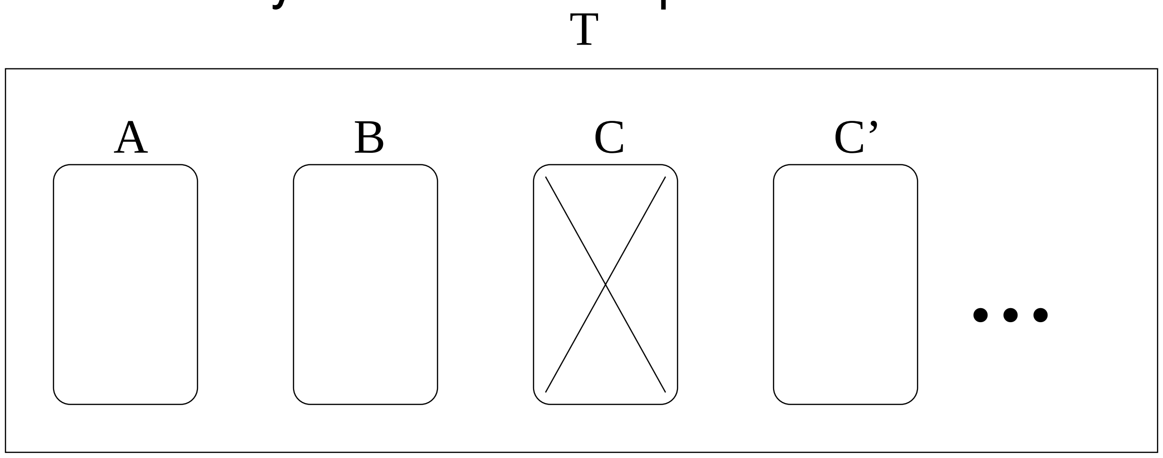
- **Isolation**
  - Transactions are independent
  - Result of execution of concurrent transactions equivalent to execution in some serial order
- **Durability**
  - Effects of committed transactions are permanent in spite of subsequent failures

Atomicity for Mobile Transactions

- Mobile Transactions likely to be long running
  - Interactive Nature
  - Mobility of Data Sources and Data Consumers
- Mobile Transactions are error prone
- If atomicity of transactions was enforced, all the work done by a transaction would be lost as a result of a single failure
- Need ability to handle partial failures
- These argue against strict atomicity

How to weaken Atomicity?

- A transaction may consist of component transactions



- Failure of a component transaction of a transaction T does not mean that all the work done by T has to be lost
- Failure may be restricted to failed component transaction
- An alternative for the failed component may be initiated

Isolation for Mobile Transactions

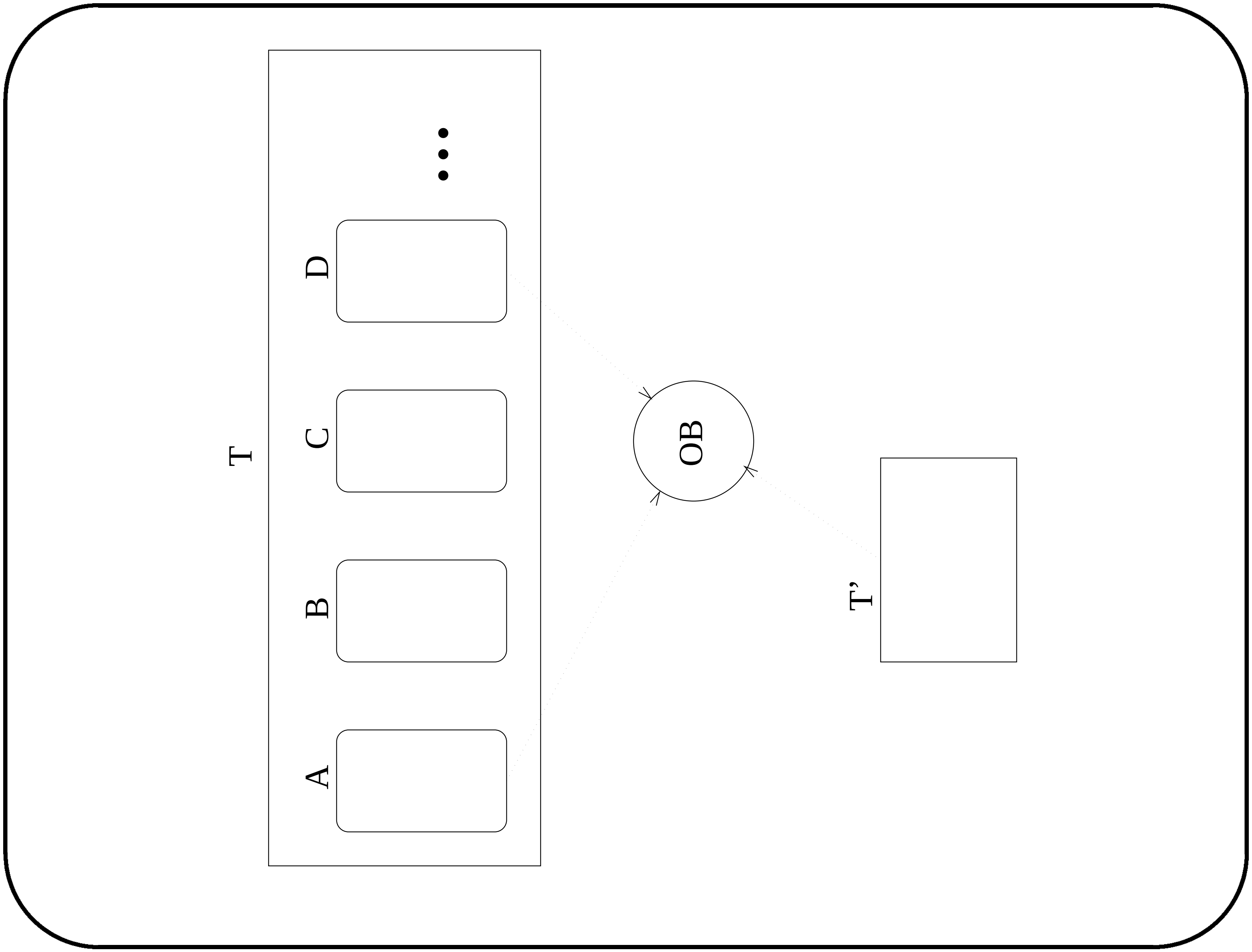
- If we require isolation of top level transactions
  - Updates of component transactions should not be visible to other transactions till top level transaction has committed
  - This requires maintaining the states of Component Transactions till the commit time of top level transactions
- Maintaining this state on the mobile computer is not always feasible because of limited (and unreliable?) disk resources

Isolation for Mobile Transactions (Contd.)

- Maintaining the state on mobile support nodes is not always feasible because
  - Mobile computer may move along several mobile support nodes
  - Makes atomic commitment expensive
- These argue against strict isolation of top level transactions

How to weaken Isolation?

- Allow certain component transactions to unilaterally commit irrespective of the fate of the top level transaction
- This makes changes performed by the component transaction visible to other transactions


T
A
B
C
D
OB
T'

Weakened notion of Isolation

- A modifies Ob and unilaterally commits
- T' reads Ob and commits
- D further modifies Ob
- T' is not serializable with respect to T

Isolation for Mobile Transactions Revisited

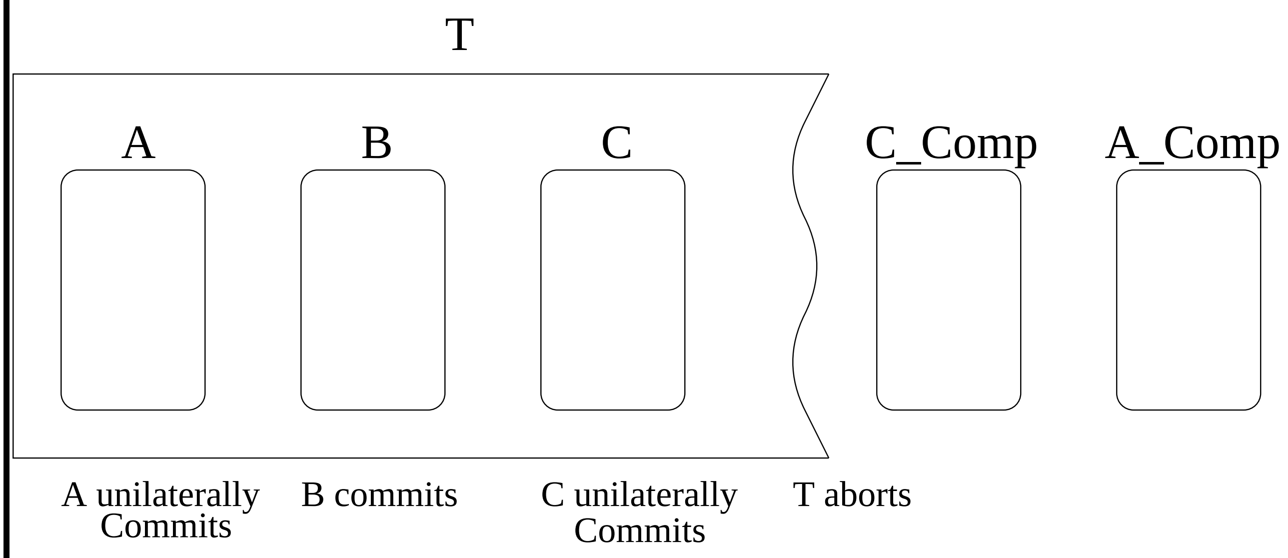
- Interactive applications
  - Better to run transactions on the mobile computer
- Mobile Computer has limited computing resources
  - Better to run transactions on mobile support hosts
- What about interactive, computation-intensive applications?
- Need cooperation between transactions running on mobile computer and mobile support hosts
- Argues against Isolation

How to further weaken Isolation?

- Allow transactions to share partial results
- A reporting transaction can share partial results with the transaction it is reporting to
- Co-transactions are like reporting transactions but do not allow for concurrent execution
- Retain state between executions

Atomicity for Mobile Transactions Revisited

- If a component transaction unilaterally commits and the top level transaction aborts
  - Effects of the component transaction persist
  - These are “semantically removed” by executing Compensating Transactions


T
A
B
C
C_Comp
A_Comp
A unilaterally Commits
B commits
C unilaterally Commits
T aborts

Durability for Mobile Transactions

- Top level transaction not responsible only for operations it performs
- Also responsible for operations performed by component transactions (which do not unilaterally commit)
- All operations for which a transaction is responsible for are made durable after it commits

Consistency for Mobile Transactions

- In traditional transactions, “correctness” of transaction program could be verified in isolation
- A “correct” transaction would transform the database from one consistent state to another
- Weakening the isolation requirement requires considering other transactions during correctness verification
- Extra burden on application programmers

Related Work

- Presented work does not discuss operation during disconnection
  - Working with cached copies
  - Re-integrating results (providing a weak form of consistency)
- These issues addressed using
  - Isolation-Only Transactions [Lu and Satynarayanan]
  - Weak Transactions [Pitoura and Bhargava]