Overview of Mobile Data Management

Outline

- Drives of mobile data management
- Objectives of mobile data management
- Data management in client/server mobile environments
- Data management in ad hoc mobile environments

Drives of mobile data management

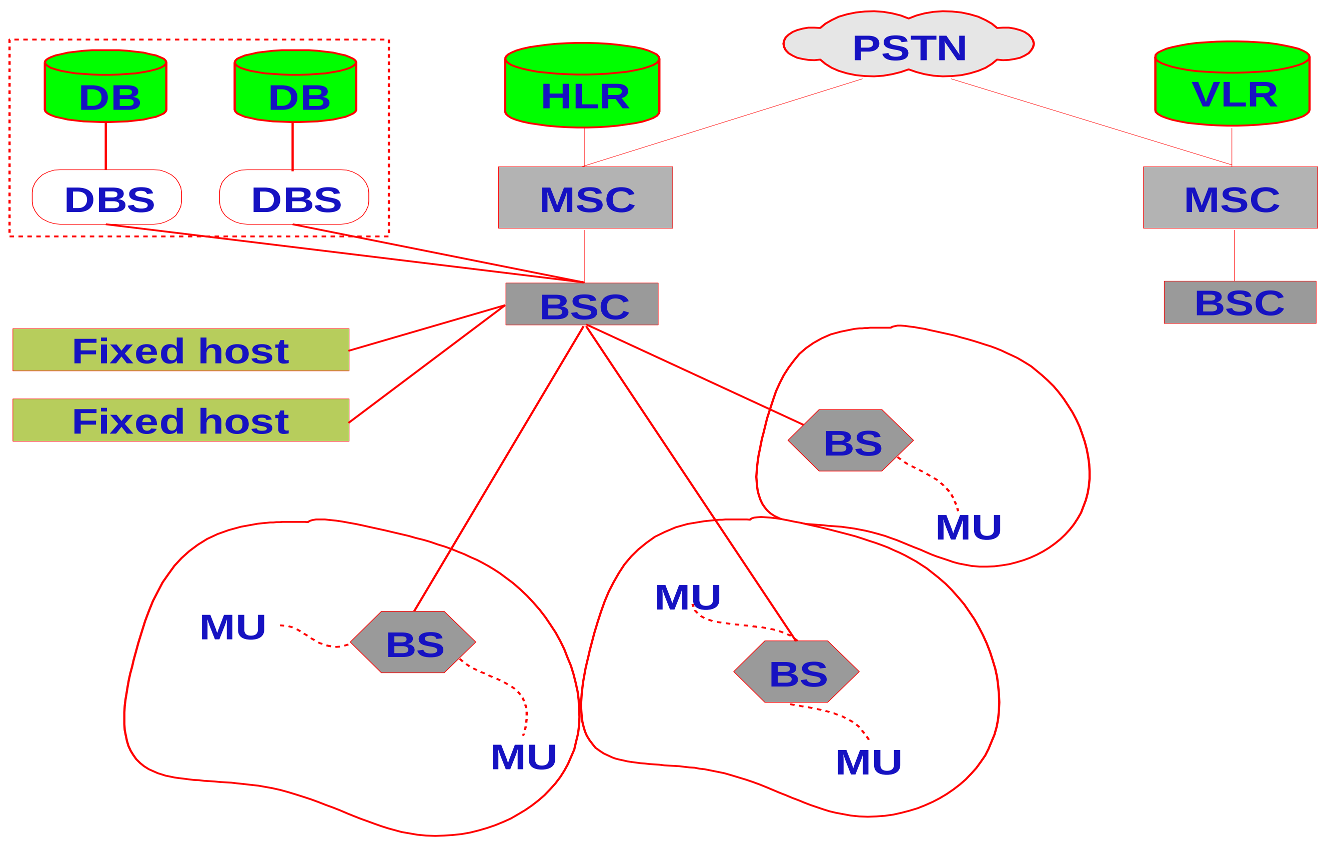
- **With increasing wireless connectivity and popularity of portable devices, mobile users are enabled to access and share on-line data and related services.**
- **Mobile users can also carry extracts of corporate databases with them to have continuous access.**
  - Sales force automation – especially in pharmaceutical industry, consumer goods, parts, etc.
  - Financial consulting and planning
  - Insurance and claim processing
  - Real estate / Property management – maintenance and building contracting
  - Mobile e-commerce

Objectives of mobile data management

- Reducing the number of data transmitted over wireless networks.
- Reducing the response time of accessing data via wireless networks.
- Providing data caching on mobile hosts.
- Maintaining consistency of data and transactions.

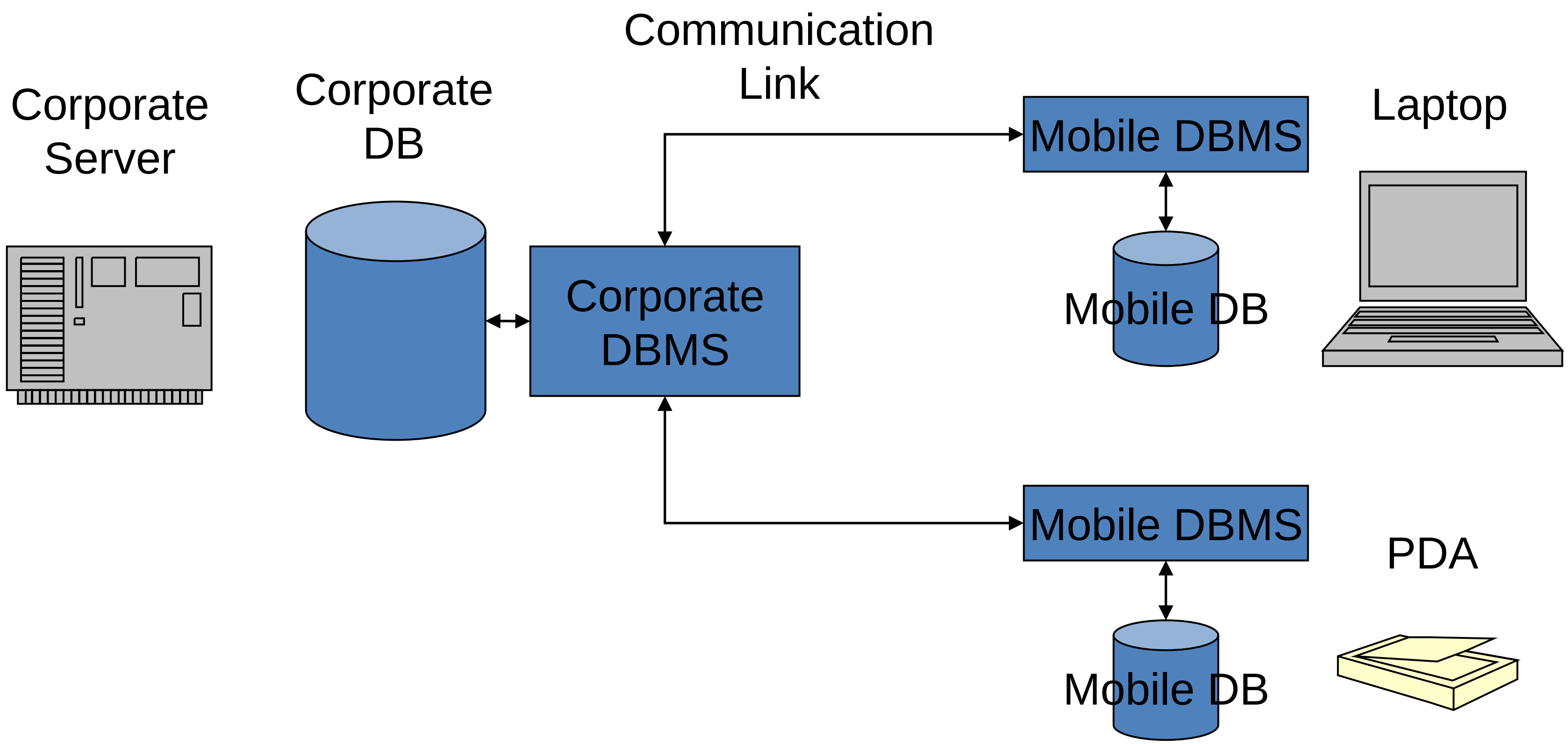
Data management in Client /Server mobile environments

- A reference model of mobie databases



Data management in Client /Server mobile environments

- Components of mobile a database



Mobile databases

- **Mobility and wireless have big impact on deign of DBMS!**
- **Models**
  - One server or many servers
  - Shared data, with full or partial *replication*
  - Some local data on client side, mostly subsets of global data
  - Both client side and server side computing
  - Execution of transactions on multiple nodes – mobile and fixed
- **Requirements: access to accurate & up-to-date information with constraints**
  - Some applications can tolerate bounded inconsistency: e.g., sacrificing strict "ACID" requirements and allowing "weaker" consistent models.
  - Long disconnection should not constraint availability

Mobile database design

- **Data modeling and design**
  - Modeling clients and related data that can change locations
  - Modeling and handling fast changing data.
  - Even distribution of data among servers – design of server databases with partitioning and replication
- **Handling intermittent connectivity**
  - Constraint: only client can, whenever needed, establish communication with server but not vice versa.
  - Replication
  - Synchronization of replicas
  - Update Installation and propagation

Mobile database design

- **Fault tolerance and recovery**
  - Handling various failures, including *site*, *media*, *communication,* and *transaction* failures
    - e.g., transaction failure is common during handoff.
  - *Planned* and *unplanned* failures should be treated differently
  - In most cases, when failures occur, data recovery is needed
    - Shadows
    - Checkpoints
    - Logs

Mobile database: data management

- **Mobile transaction management**
  - Concurrency and Integrity constraint enforcement
  - Recovery of mobile transactions
- **Wireless data dissemination (broadcast)**
- **Mobile data caching / replication management**
- **Disconnected operations**
- **Mobile query processing**
  - Energy efficient query processing, e.g., data shipping vs. query shipping
  - Location dependent query processing – must reflect the constantly-changing location of client
  - Querying moving objects – keep tracking moving objects

Mobile transactions

- **Transaction: a set of operations that translate a database from one consistent state to another consistent state**
  - Transaction operations
    - Serialization of concurrent execution
    - Transaction commit
    - Transaction and database recovery

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

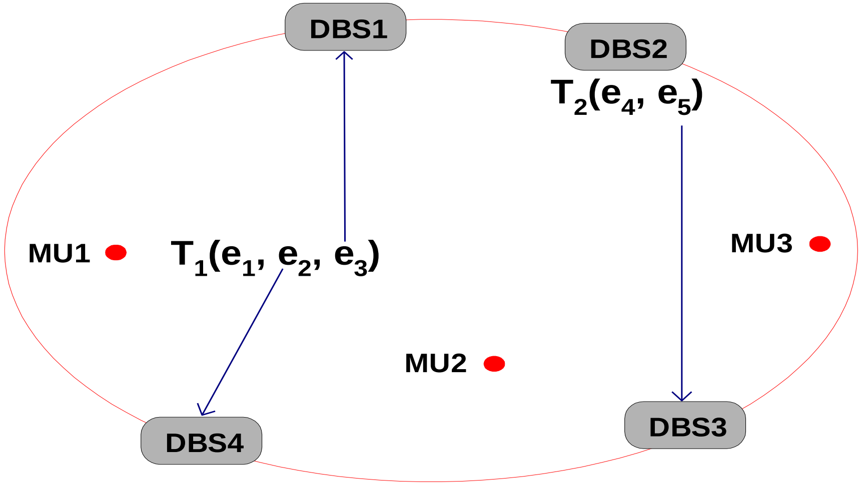
- **Transaction properties:**

  ACID (Atomicity, Consistency, Isolation, and Durability)

- **Too rigid and difficult to enforce for mobile transactions.**

Mobile transactions

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

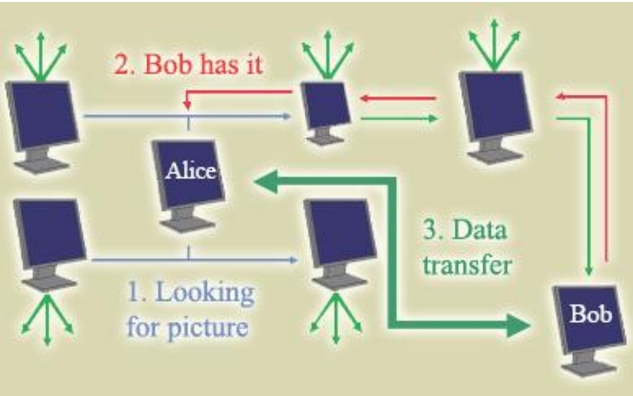


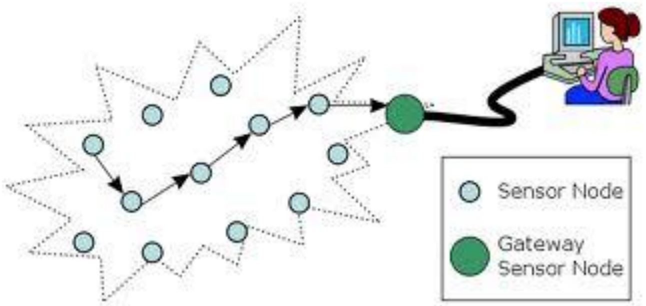
Mobile transactions

- **Flexibility need be introduced.**
  - E.g., using workflow concept, part of the transaction can be executed and committed independent to its other parts
  - New models and solutions are needed.
- **Kangaroo Transactions: the management of transactions move with MU.**
  - A Kangaroo Transaction (KT) is created at MU
  - On each hop to a new BS, A Jump Transaction (JT) is created at the BS - JT consists of a set of Local Transactions (LTs) and Global Transactions (GTs)
  - Each BS manages mobile transactions and the movement of the MU.
    - Mobile transaction's execution is coordinated by the BS the MU is currently associated with.
    - When MU hops from one cell to another, coordination of the mobile transaction moves to the new BS.
    - Maintains a linked list of all BSs that have been coordinators of the KT

Data management in ad hoc mobile environments

- With no dedicated routers, nodes become much more prone to get disconnected from network
- Data availability degrade significantly
- Query & data delivery becomes much more difficult & costly (energy)





Caching & Replication

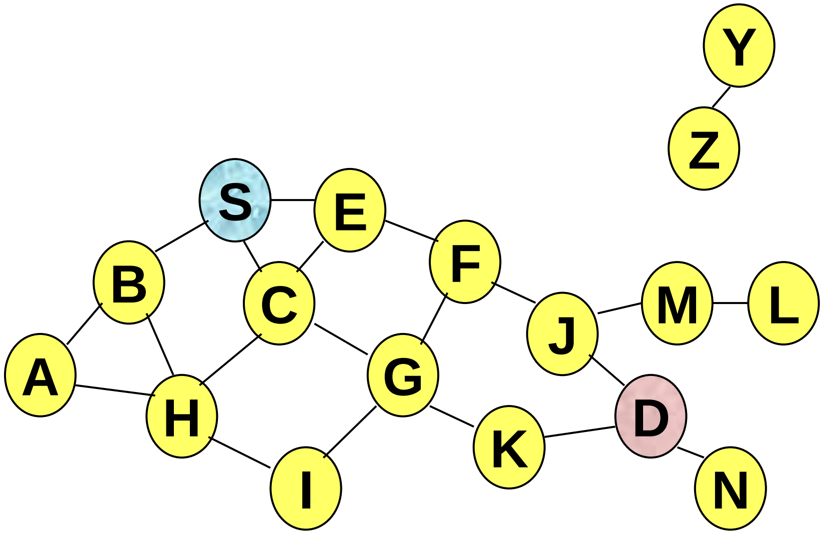
- **Same goal as in C/S model: improve data availability and reduce delay**
- **But, replication is very hard to achieve now**
  - Disconnection makes it difficult to guarantee consistency in a timely and efficient manner, as we rely purely on the mobile nodes, no wired nodes
  - As nodes are disconnected more often, data at each node diverges further and further from others
  - Soon, the database is inconsistent and there is no obvious way to repair it
- **Cooperative caching provides an effective way.**

Caching in ad hoc networks

- **Caching**
  - Cache popular data on the querying node
  - Reduce traffic overhead and query delay
- **Cooperative Caching**
  - Data source asks a collection of caching nodes for help
  - Nodes cooperate to cache popular data
    - Coordination and sharing of cached data
  - Advantages
    - Further explore the potential of caching – nodes collaborate to serve queries without having to frequently send request to data source
    - Shorter delay and less communication overhead

Data dissemination in ad hoc networks

- **Data collection and dissemination**
  - From / to a certain node
  - From / to a certain group of nodes
  - From / to a certain area
- **Data delivery in**
  - Mobile ad hoc networks
  - Sensor networks
  - Vehicular networks
- **Ad hoc routing**
- **Delay-tolerant**



Query processing in WSN

- **Network is abstracted as a database**
  - represents sensors and sensor data in a database
- **Control of sensors and extracting data occurs through special SQL-like queries**

```
SELECT Nodeid, Light
       FROM Sensors
       WHERE Light > 400
       EPOCH DURATION 1s
```

Sensors

Epoch	Nodeid	Light	Temp	Accel	Sound
0	1	455	x	x	x
0	2	389	x	x	x
1	1	422	x	x	x
1	2	405	x	x	x

- **Examples:**
  - TinyDB (UC Berkley),
  - Cougar (Cornell),

Query processing in WSN

- **Network is abstracted as a database**
  - represents sensors and sensor data in a database
- **Control of sensors and extracting data occurs through special SQL-like queries**

```
SELECT Nodeid, Light
       FROM Sensors
       WHERE Light > 400
       EPOCH DURATION 1s
```

Sensors

Epoch	Nodeid	Light	Temp	Accel	Sound
0	1	455	x	x	x
0	2	389	x	x	x
1	1	422	x	x	x
1	2	405	x	x	x

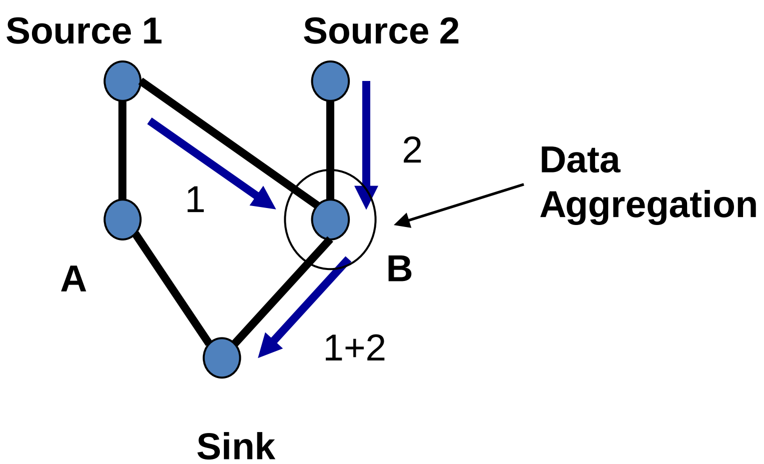
- **Examples:**
  - TinyDB (UC Berkley),
  - Cougar (Cornell),

Query processing in WSN

- **Users specify the data they want**
  - Simple, SQL-like queries
- **Challenge is to how to provide:**
  - expressive & easy-to-use interface
  - high-level operators
    - well-defined interactions
    - “transparent optimizations” that many programmers would miss
      - Sensor-net specific techniques
  - Power-efficient execution of queries

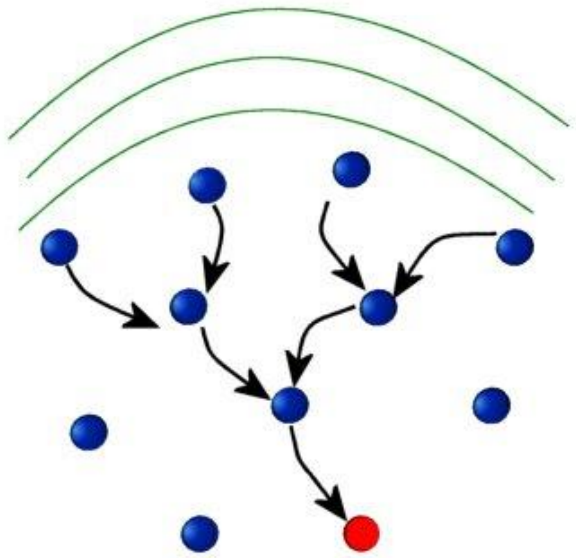
Data aggregation in WSN

- **Exploit query semantics to improve efficiency**
  - Data coming from multiple sensor nodes are aggregated when they reach a common routing or relaying node on their way to the sink, if they have about the same attributes of the phenomenon being sensed
  - Provide energy savings by allowing in-network aggregation of redundant information and reducing transmissions



Data aggregation in WSN

- **In this view, routing structure in a sensor network can be considered as a form of reverse multicast tree**
- **Optimal aggregation is NP-hard in general**
  - Equivalent to forming a minimum Steiner tree.
  - Optimum no. Of transmission = no. of edges in the minimum Steiner tree



*A minimum-weight tree connecting a designated set of vertices, called terminals, in a weighted graph. The tree may include non-terminals, which are called Steiner vertices”

Data aggregation in WSN

- **Aggregation techniques (sub-optimal)**
  - *Center at Nearest Source (CNSDC)*: All sources send the information first to the source nearest to the sink, which acts as the aggregator.
  - *Shortest Path Tree (SPTDC)*: Opportunistically merge the shortest paths from each source wherever they overlap.
  - *Greedy Incremental Tree (GITDC)*: Start with path from sink to nearest source. Successively add next nearest source to the existing tree.