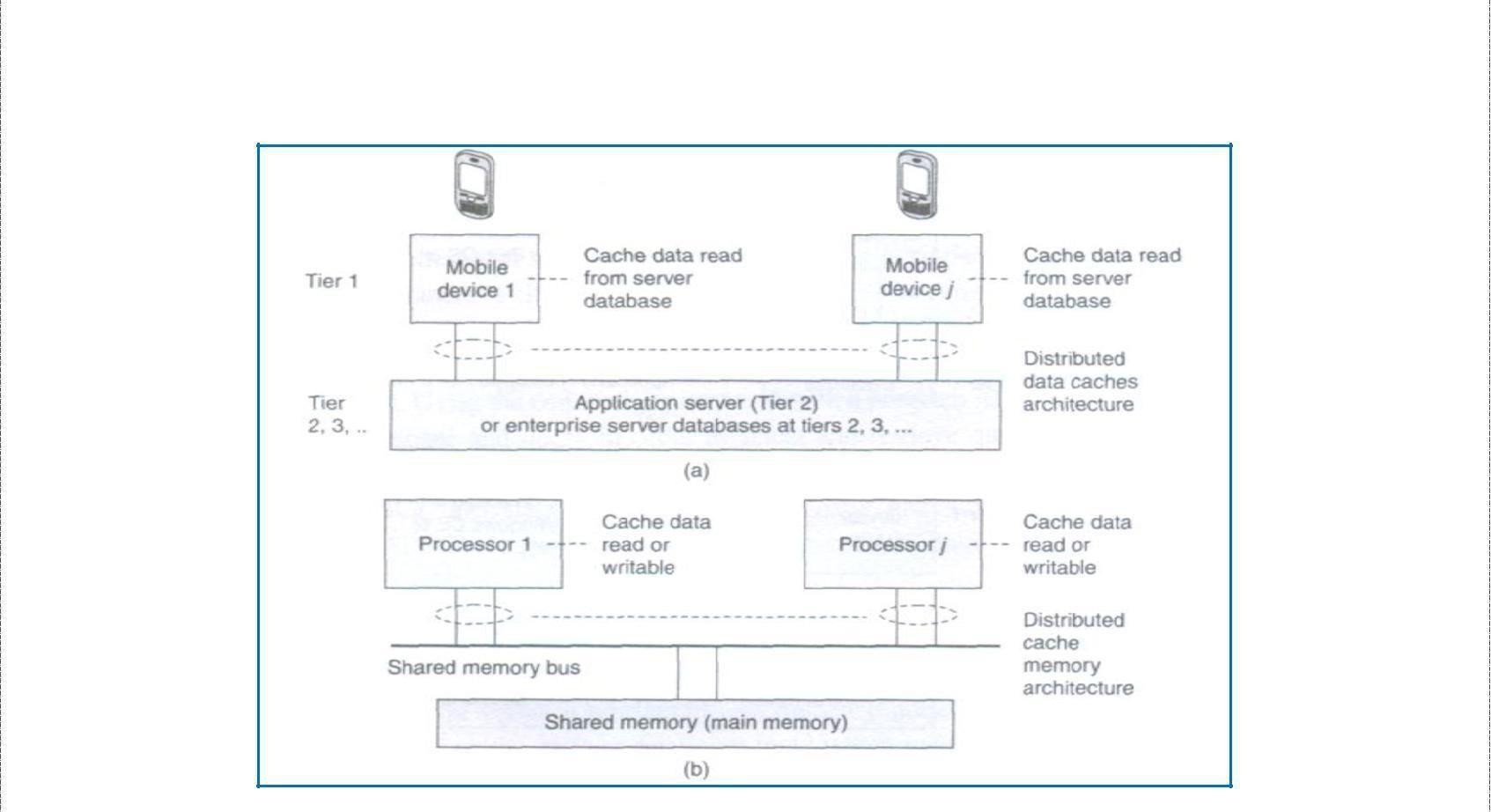
**UNIT 5: DATABASES**

|  |
| --- |
| A database is a collection of systematically stored records or information. Databases store data in a particular logical manner. A mobile device is not always connected to the server or network; neither does the device retrieve data from a server or a network for each computation. Rather, the device caches some specific data, which may be required for future computations, during the interval in which the device is connected to the server or network. Caching entails saving a copy of select data or a part of a database from a connected system with a large database. The cached data is hoarded in the mobile device database. Hoarding of the cached data in the database ensures that even when the device is not connected to the network, the data required from the database is available for computing. |

**Database Hoarding**

Database hoarding may be done at the application tier itself. The following figure shows a simple architecture in which a mobile device API directly retrieves the data from a database. It also shows another simple architecture in which a mobile device API directly retrieves the data from a database through a program, for ex: IBM DB2 Everyplace (DB2e).

1. ***API at mobile device sending queries and retrieving data from local database (Tier 1)***
2. ***API at mobile device retrieving data from database using DB2e (Tier 1)***



Both the two architectures belong to the class of one-tier database architecture because the databases are specific to a mobile device, not meant to be distributed to multiple devices, not synchronized with the new updates, are stored at the device itself. Some examples are downloaded ringtones, music etc. **IBM DB2 Everyplace (DB2e)** is a relational database engine which has been designed to reside at the device. It supports J2ME and most mobile device operating systems. DB2e synchronizes with DB2 databases at the synchronization, application, or enterprise server

The database architecture shown below is for two-tier or multi-tier databases. Here, the databases reside at the remote servers and the copies of these databases are cached at the client tiers. This is known as client-server computing architecture.

* 1. ***Distributed data caches in mobile devices***
  2. ***Similar architecture for a distributed cache memory in multiprocessor systems***

A cache is a list or database of items or records stored at the device. Databases are hoarded at the application or enterprise tier, where the database server uses business logic and connectivity for retrieving the data and then transmitting it to the device. The server provides and updates local copies of the database at each mobile device connected to it. The computing API at the mobile device (first tier) uses the cached local copy. At first tier (tier 1), the API uses the cached data records using the computing architecture as explained above. From tier 2 or tier 3, the server retrieves and transmits the data records to tier 1 using business logic and synchronizes the local copies at the device. These local copies function as device caches.

The advantage of hoarding is that there is no access latency (delay in retrieving the queried record from the server over wireless mobile networks). The client device API has instantaneous data access to hoarded or cached data. After a device caches the data distributed by the server, the data is hoarded at the device. The disadvantage of hoarding is that the consistency of the cached data with the database at the server needs to be maintained.

**Data Caching**

Hoarded copies of the databases at the servers are distributed or transmitted to the mobile devices from the enterprise servers or application databases. The copies cached at the devices are equivalent to the cache memories at the processors in a multiprocessor system with a shared main memory and copies of the main memory data stored at different locations.

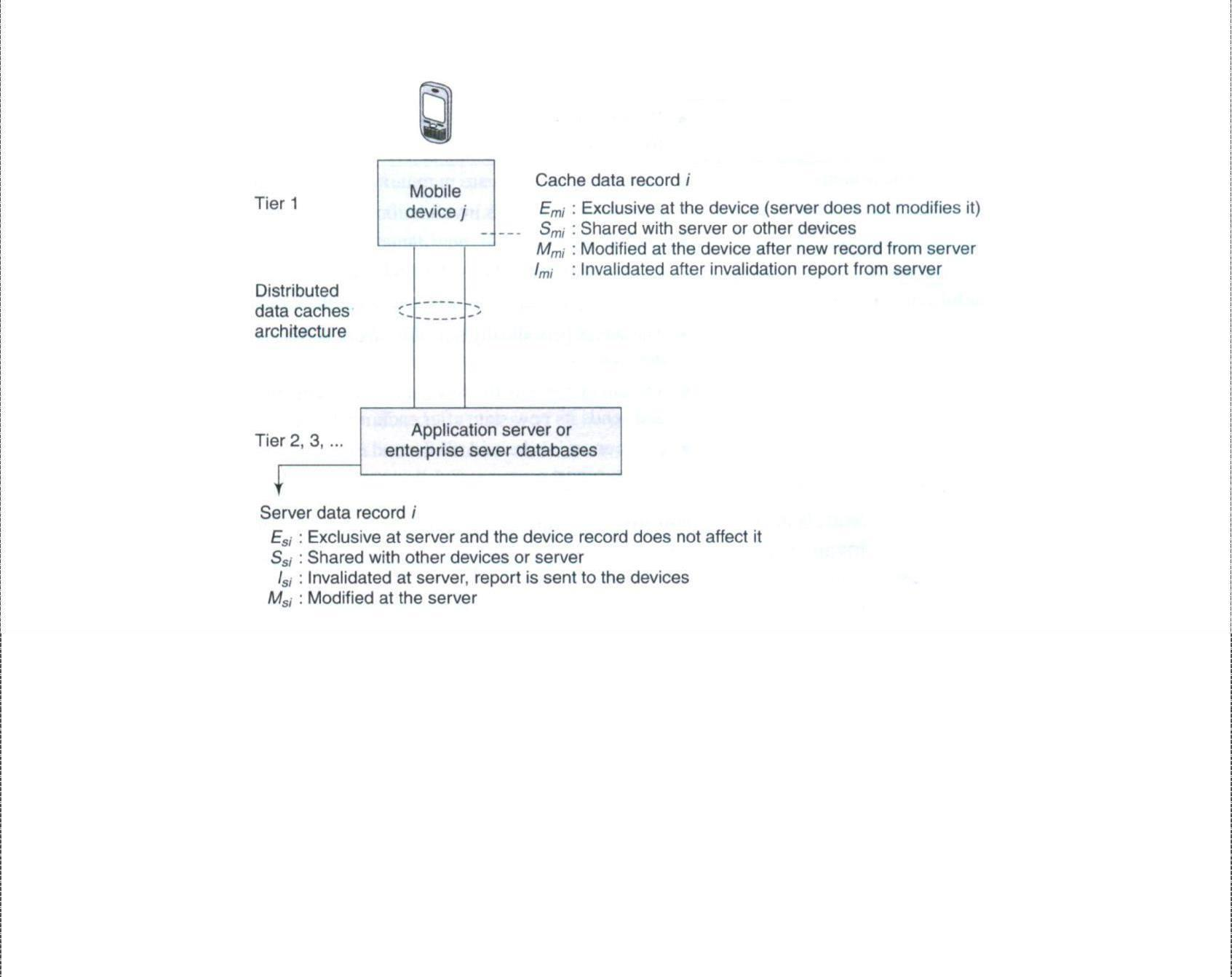
**Cache Access Protocols:** A client device caches the pushed (disseminated) data records from a server. Caching of the pushed data leads to a reduced access interval as compared to the pull (on- demand) mode of data fetching. Caching of data records can be-based on pushed 'hot records' (the most needed database records at the client device). Also, caching can be based on the ratio of two parameters—access probability (at the device) and pushing rates (from the server) for each record. This method is called cost-based data replacement or caching.

**Pre-fetching:** Pre-fetching is another alternative to caching of disseminated data. The process of pre-fetching entails requesting for and pulling records that may be required later. The client device can pre-fetch instead of caching from the pushed records keeping future needs in view. Pre-fetching reduces server load. Further, the cost of cache-misses can thus be reduced. The term 'cost of cache-misses' refers to the time taken in accessing a record at the server in case that record is not found in the device database when required by the device API.

Caching Invalidation Mechanisms

A cached record at the client device may be invalidated. This may be due to expiry or modification of the record at the database server. Cache invalidation is a process by which a cached data item or record becomes invalid and thus unusable because of modification, expiry, or invalidation at another computing system or server. Cache invalidation mechanisms are used to synchronize the data at other processors whenever the cache-data is written (modified) by a processor in a multiprocessor system, cache invalidation mechanisms are also active in the case of mobile devices having distributed copies from the server.

A cache consists of several records. Each record is called a cache-line, copies of which can be stored at other devices or servers. The cache at the mobile devices or server databases



|  |  |  |  |
| --- | --- | --- | --- |
| ***iven time can be assigned one of four possible tags indicating its state—modified (after rewriting), exclusive,*** | | |  |
| ***hared, and invalidated (after expiry or when new data becomes available) at any given instance. These four states*** | | | |
| ***re indicated by the letters M, E, S, and I, respectively (MESI). The states indicated by the various tags are as*** | |  | |
| ***ollows:*** | 1. The ***E*** tag indicates the *exclusive* state which means that the data record is for internal use and cannot be used by any other device or server. 2. The ***S*** tag indicates the *shared* state which indicates that the data record can be used by others. 3. The ***M*** tag indicates the *modified* state which means that the device cache 4. The ***I*** tag indicates the *invalidated state* which means that the server database no longer has a copy of the record which was shared and used for computations earlier.   The following figure shows the four possible states of a data record *i* at any instant in the server database and its copy at the cache of the mobile device *j*.  ***Four possible states (M, E, S, or /) of a data record /at any instance at the server***  ***database and device j cache***  Another important factor for cache maintenance in a mobile environment is *cache consistency* (also called *cache coherence).* This requires a mechanism to ensure that a database record is identical at the server as well as at the device caches and that only the valid cache records are used for computations. |
|  |

***g s a f***

|  |  |
| --- | --- |
|  | Cache invalidation mechanisms in mobile devices are triggered or initiated by the server. There are four possible invalidation mechanisms – Stateless asynchronous, stateless synchronous, stateful asynchronous and stateful synchronous.    **Stateless Asynchronous:** A stateless mechanism entails broadcasting of the invalidation of the cache to all the clients of the server. The server does not keep track of the records stored at the device caches. It just uniformly broadcasts invalidation reports to all clients irrespective of whether the device cache holds that particular record or not. The term 'asynchronous' indicates that the invalidation information for an item is sent as soon as its value changes. The server does not keep the information of the present state (whether *Emi, Mmi, Smi,* or *Imi)* of a data-record in cache for broadcasting later. The server advertises the invalidation information only. The client can either request for a modified copy of the record or cache the relevant record when data is pushed from the server. The server advertises as and when the corresponding data- record at the server is invalidated and modified (deleted or replaced).  The advantage of the asynchronous approach is that there are no frequent, unnecessary transfers of data reports, thus making the mechanism more bandwidth efficient. The disadvantages of this  approach are—(a) every client device gets an invalidation report, whether that client requires that copy or not and (b) client devices presume that as long as there is no invalidation report, the copy is valid for use in computations. Therefore, even when there is link failure, the devices may be using the invalidated data and the server is unaware of state changes at the clients after it sends the invalidation report.  Statele~~ss~~ Synchronous This is also a stateless mode, i.e., the server has no information regarding the present state of data records at the device caches and broadcasts to all client devices. However, unlike the asynchronous mechanism, here the server advertises invalidation information at periodic intervals as well as whenever the corresponding data- record at server is invalidated or modified. This method ensures synchronization because even if the in-between period report is not detected by the device due to a link failure, the device expects the period-end report of invalidation and if that is not received at the end of the period, then the device sends a request for the same (deleted or replaced). In case the client device does not get the periodic report due to link failure, it requests the server to send the report.  The advantage of the synchronous approach is that the client devices receive periodic information regarding invalidity (and thus validity) of the data caches. The periodic invalidation reports lead to greater reliability of cached data as update requests for invalid data can be sent to the server by the device-client. This also helps the server and devices maintain cache consistency through periodical exchanges. The disadvantages of this mode of cache invalidation are—(a) unnecessary transfers of data invalidation reports take place, (b) every client device gets an advertised invalidation report periodically, irrespective of whether that client has a copy of the invalidated data or not, and (c) during the period between two invalidation reports, the client devices assume that, as long as there is no invalidation report, the copy is valid for use in computations. |
|  |



Therefore, when there are link failures, the devices use data which has been invalidated in the in-between period and the server is unaware of state changes at the clients after it sends the invalidation report.

**Stateful Asynchronous** The stateful asynchronous mechanism is also referred to as the AS (asynchronous stateful) scheme. The term 'stateful' indicates that the cache invalidation reports are sent only to the affected client devices and not broadcasted to all. The server stores the information regarding the present state (a record I can have its state as *Emi, Mmi, Smi,* or *Imi)* of each data-record at the client device caches. This state information is stored in the home location cache (HLC) at the server. The HLC is maintained by an HA (home agent) software. This is similar to the HLR at the MSC in a mobile network. The client device informs the HA of the state of each record to enable storage of the same at the HLC. The server transmits the invalidation information as and when the records are invalidated and it transmits only to the device-clients which are affected by the invalidation of data.

Based on the invalidation information, these device-clients then request the server for new or modified data to replace the invalidated data. After the data records transmitted by the server modify the client device cache, the device sends information about the new state to the server so that the record of the cache-states at the server is also modified.

The advantage of the stateful asynchronous approach is that the server keeps track of the state of cached data at the client device. This enables the server to synchronize with the state of records at the device cache and keep the HLC updated. The stateful asynchronous mode is also advantageous in that only the affected clients receive the invalidation reports and other devices are not flooded with irrelevant reports. The disadvantage of the AS scheme is that the client devices presume that, as long as there is no invalidation report, the copy is valid for use in computations. Therefore, when there is a link failure, then the devices use invalidated data.

**Stateful Synchronous:** The server keeps the information of the present state (Emi, Mmi, Smi, or Imi) of data-records at the client-caches. The server stores the cache record state at the home location cache (HLC) using the home agent (HA). The server transmits the invalidation information at periodic intervals to the clients and whenever the data-record relevant to the client is invalidated or modified (deleted or

replaced) at the server. This method ensures synchronization because even if thein- between period report is not detected by the device due to a link failure, the device expects the period-end report of invalidation and if it is not received at the end of the period, then the device requests for the same.

The advantage of the stateful synchronous approach is that there are reports identifying invalidity (and thus, indirectly, of validity) of data caches at periodic intervals and that the server also periodically updates the client-cache states stored in the HLC. This enables to synchronize with the client device when invalid data gets modified and becomes valid. Moreover, since the invalidation report is sent periodically, if a device does not receive an invalidation report after the specified period of time, it can request the server to send the report. Each client can thus be periodically updated of any modifications at the server. When the invalidation report is not received after the designated period and a link failure is found at the device, the device does not use the invalidated data.

Instead it requests the server for an invalidation update. The disadvantage of the stateful synchronous approach is the high bandwidth requirement to enable periodic transmission of invalidation reports to each device and updating requests from each client device.

*Data Cache Maintenance in Mobile Environments*

Assume that a device needs a data-record during an application. A request must be sent to the server for the data record (this mechanism is called pulling). The time taken for the application software to access a particular record is known as *access latency.* Caching and hoarding the record at the device reduces access latency to zero. Therefore, data cache maintenance is necessary in a mobile environment to overcome access latency.

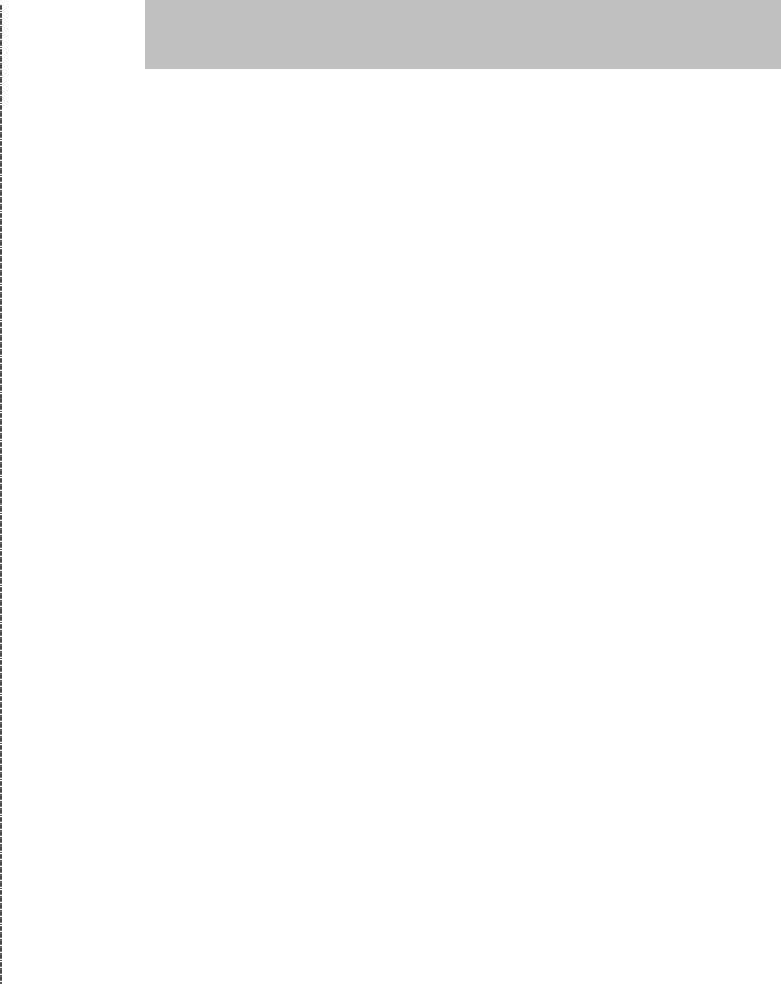
*Data cache inconsistency* means that data records cached for applications are not invalidated at the device when modified at the server but not modified at the device. Data cache consistency can be maintained by the three methods given below:

1. *Cache invalidation mechanism* (server-initiated case): the server sends invalidation reports on invalidation of records (asynchronous) or at regular intervals (synchronous).
2. *Polling mechanism* (client-initiated case): Polling means checking from the server, the state of data record whether the record is in the valid, invalid, modified, or exclusive state. Each cached record copy is polled whenever required by the application software during computation. If the record is found to be modified or invalidated, then the device requests for the modified data and replaces the earlier cached record copy.
3. *Time-to-live mechanism* (client-initiated case): Each cached record is assigned a TTL (time- to-live). The TTL assignment is adaptive (adjustable) previous update intervals of that record. After the end of the TTL, the cached record copy is polled. If it is modified, then the device requests the server to replace the invalid cached record with the modified data. When TTL is set to 0, the TTL mechanism is equivalent to the polling mechanism.

*Web Cache Maintenance in Mobile Environments*

The mobile devices or their servers can be connected to a web server (e.g., traffic information server or train information server). Web cache at the device stores the web server data and maintains it in a manner similar to the cache maintenance for server data described above. If an application running at the device needs a data record from the web which is not at the web cache, then there is access latency. Web cache maintenance is necessary in a mobile environment to overcome access latency in downloading from websites due to disconnections. Web cache consistency can be maintained by two methods. These are:

Time-to-live (TTL) mechanism (client-initiated case): The method is identical to the one discussed for data cache maintenance.



I. Power-aware computing mechanism (client-initiated case): Each web cache maintained at the device can also store the CRC (cyclic redundancy check) bits. Assume that there are N cached bits and *n* CRC bits. N is much greater than *n.* Similarly at the server, *n* CRC bits are stored. As long as there is consistency between the server and device records, the CRC bits at both are identical. Whenever any of the records cached at the server is modified, the corresponding CRC bits at the server are also modified. After the TTL expires or on- demand for the web cache records by the client API, the cached record CRC is polled and obtained from the website server. If the *n* CRC bits at server are found to be modified and the change is found to be much higher than a given threshold (i.e., a significant change), then the modified part of the website hypertext or database is retrieved by the client device for use by the API. However, if the change is minor, then the API uses the previous cache. Since *N* » *n,* the power dissipated in the web cache maintenance method (in which invalidation reports and all invalidated record bits are transmitted) is much greater than that in the present method (in which the device polls for the significant change in the CRC bits at server and the records are transmitted only when there is a significant change in the CRC bits).

Client-Server Computing

Client-server computing is a distributed computing architecture, in which there are two types of nodes, i.e., the clients and the servers. A server is defined as a computing system, which responds to requests from one or more clients. A client is defined as a computing system, which requests the server for a resource or for executing a task. The client can either access the data records at the server or it can cache these records at the client device. The data can be accessed either on client request or through broadcasts or distribution from the server.

The client and the server can be on the same computing system or on different computing systems. Client-server computing can have N-tier architecture *(N=* 1, 2 ...). When the client and the server are on the same computing system then the number of tiers, *N =* 1. When the client and the server are on different computing systems on the network, then *N* = 2. A command interchange protocol (e.g., HTTP) is used for obtaining the client requests at the server or the server responses at the client.

The following subsections describe client-server computing in 2, 3, or N-tier architectures. Each tier connects to the other with a connecting, synchronizing, data, or command interchange protocol.



##### Two-tier Client-Server Architecture

***Multimedia file server in two-tier client-server computing architecture (local copies 1 to j of image and voice hoarding at the mobile devices)***

The following figure shows the application server at the second tier. The data records are retrieved using business logic and a synchronization server in the application server synchronizes with the local copies at the mobile devices. Synchronization means that when copies of records at the server-end are modified, the copies cached at the client devices should also be accordingly modified. The APIs are designed independent of hardware and software platforms as far as possible as different devices may have different platforms.

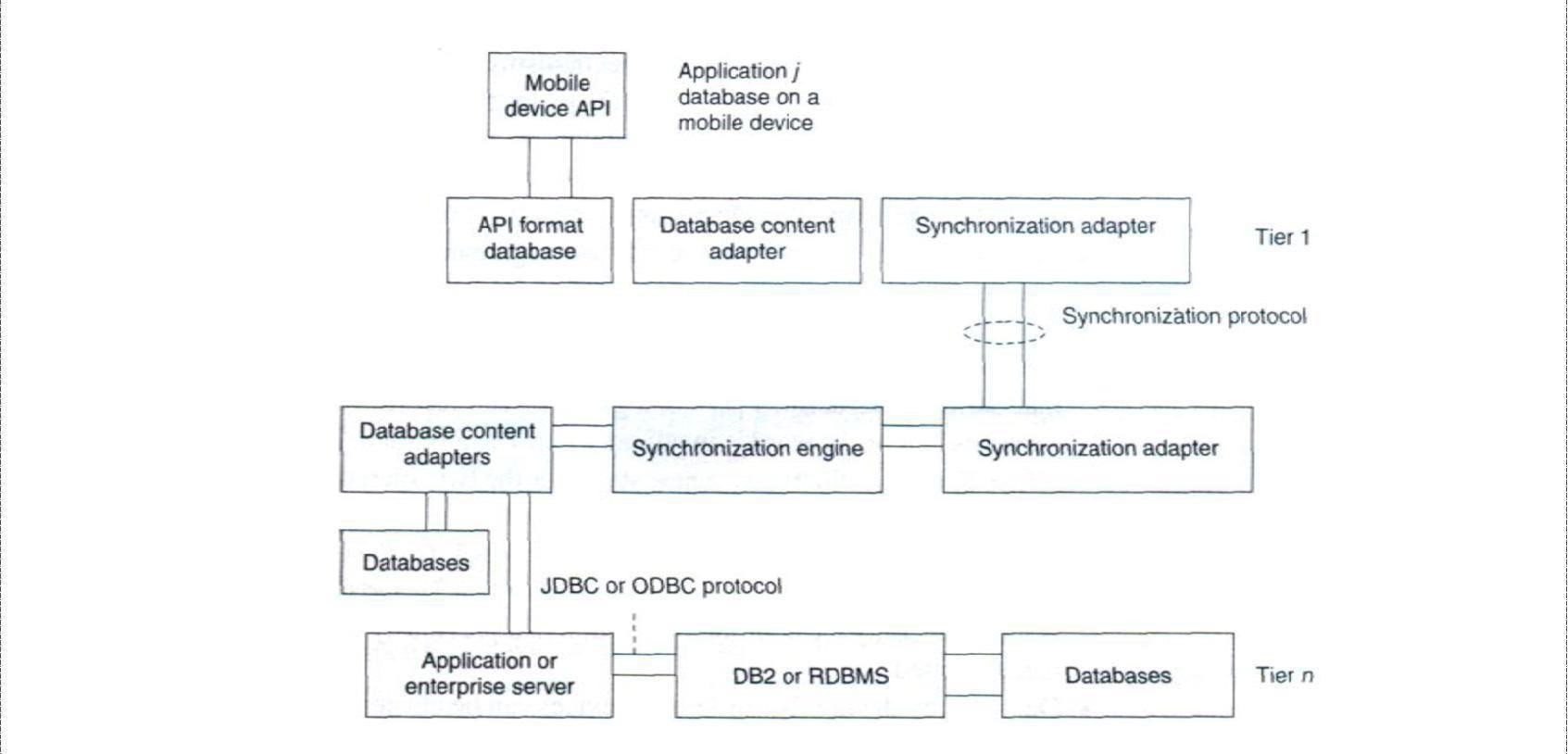
##### Three-tier Client-Server Architecture

In a three-tier computing architecture, the application interface, the functional logic, and the database are maintained at three different layers. The database is associated with the enterprise server tier (tier 3) and only local copies of the database exist at mobile devices. The database connects to the enterprise server through a connecting protocol. The enterprise server connects the complete databases on different platforms, for example, Oracle, XML, and IBM DB2.

|  |  |
| --- | --- |
|  | ***(a) Local copies 1 to j of database hoarded at the mobile devices using an enterprise database connection synchronization server, which synchronizes the required local copies for application with the enterprise database server (b) Mobile device with J2ME or BREW platform, APIs an OS***  ***and database having local copies***  Data records at tier 3 are sent to tier 1 as shown in the figure through a synchronization- cum- application server at tier 2. The synchronization-cum-application server has synchronization and server programs, which retrieves data records from the enterprise tier (tier 3) using business logic. There is an in-between server, called synchronization server, which sends and synchronizes the copies at the multiple mobile devices. The figure shows that local copies 1 to j of databases are hoarded at the mobile devices for the applications 1 to j*.* |
|  |

|  |  |
| --- | --- |
|  | **N-tier Client-Server Architecture**  When *N* is greater than 3, then the database is presented at the client through in-between layers. For example, the following figure shows a four-tier architecture in which a client device connects to a data-presentation server at tier 2.  ***4-tier architecture in whicha client device connects to a data-presentation server***  The presentation server then connects to the application server tier 3. The application server can connect to the database using the connectivity protocol and to the multimedia server using Java or XML API at tier 4. The total number of tiers can be counted by adding 2 to the number of in-between servers between the database and the client device. The presentation, application, and enterprise servers can establish connectivity using RPC, Java RMI, JNDI, or HOP. These servers may also use HTTP or HTTPS in case the server at a *tier j* connects to tier j+1 using the Internet.  Client-Server Computing with Adaptation  The data formats of data transmitted from the synchronization server and those required for the device database and device APIs are different in different cases, there are two adapters at a  mobile device—an adapter for standard data format for synchronization at the mobile device and another adapter for the backend database copy, which is in a different data format for the API at the mobile device. An adapter is software to get data in one format |
|  |

or data governed by one protocol and convert it to another format or to data governed by another protocol.



***Figure shows an API, database, and adapters at a mobile device and the adapters at the synchronization, application, or enterprise servers. Here the adapters are an addition used for interchange between standard data formatsand data formats for the API.***

*Transaction Models*

A transaction is the execution of interrelated instructions in a sequence for a specific operation on a database. Database transaction models must maintain data integrity and must enforce a set of rules called ACID rules. These rules are as follows:



***Atomicity****:* All operations of a transaction must be complete. In case, a transaction cannot be completed; it must be undone (rolled back). Operations in a transaction are assumed to be one indivisible unit (atomic unit).



***Consistency****:* A transaction must be such that it preserves the integrity constraints and follows the declared consistency rules for a given database. Consistency means the data is not in a contradictory state after the transaction.



***Isolation****:* If two transactions are carried out simultaneously, there should not be any interference between the two. Further, any intermediate results in a transaction should be invisible to any other transaction.

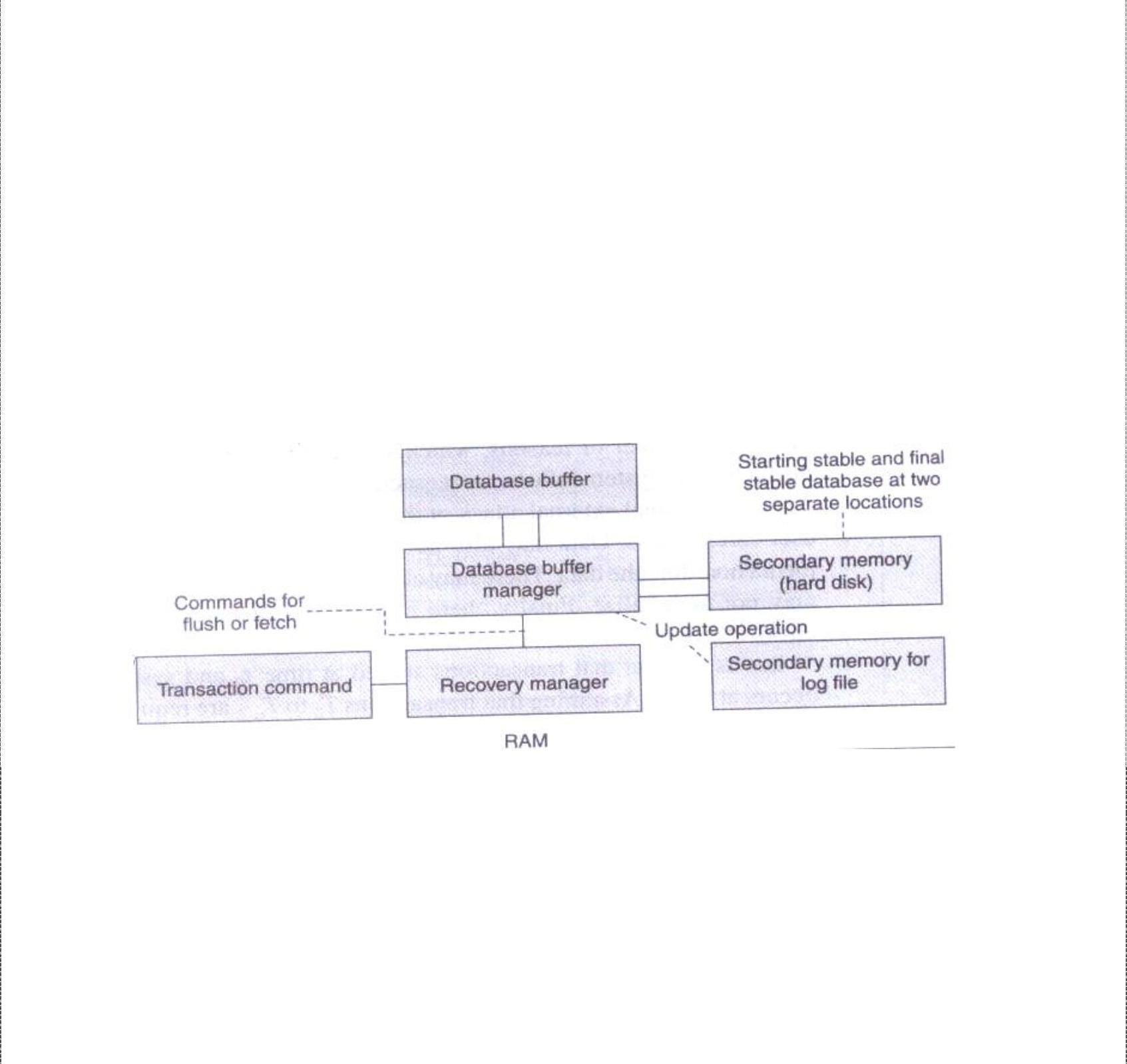
* ***Durability:*** After a transaction is completed, it must persist and cannot be aborted or discarded. For example, in a transaction entailing transfer of a balance from account A to account B, once the transfer is completed and finished there should be no roll back.

### Mobile Computing

|  |  |
| --- | --- |
|  |  |
| Consider a base class library included in Microsoft.NET. It has a set of computer software components called ADO.NET (ActiveX Data Objects in .NET). These can be used to access the data and data services including for access and modifying the data stored in relational database systems. The ADO.NET transaction model permits three transaction commands:   1. BeginTransaction**: It is used to begin a transaction. Any operation after**   BeginTransaction **is assumed to be a part of the transaction till the**  CommitTransaction **command or the** RollbackTransaction  **command. An example of a command is as follows:**  connectionA.open();  transA = connectionA.BeginTransaction();  **Here** connectionA **and** transA **are two distinct objects.**   1. Commit: **It is used to commit the transaction operations that were carried out after the** BeginTransaction **command and up to this command. An example of this is**   transA.Commit();  **All statements between** BeginTransaction **and** commit **must execute automatically.**   1. Rollback: **It is used to rollback the transaction in case an exception is generated after the** BeginTransaction **command is executed.**   A DBMS may provide for auto-commit mode. *Auto-commit mode* means the transaction finished automatically even if an error occurs in between.  **Query Processing**  Query processing means making a correct as well as efficient execution strategy by *query decomposition* and *query-optimization*. A relational-algebraic equation defines a set of operations needed during query processing. Either of the two equivalent relational-algebraic equations given below can be used. | |

******

|  |  |
| --- | --- |
|  | This means first select a column Contacts.cTelNum in a row in Contacts in which Contacts.cTelNum column equals a column DialledNumbers.dTelNum by crosschecking and matching the records of a column in Contacts with all the rows of DialledNumbers. Then in the second step select the row in which Contacts. firstChar = “R” and the selected cTelNum exists. Then in the third step project cName and CTelNum.  This means that in first series of step, crosscheck all rows of Contacts and DialledNumbers and select, after AND operation, the rows in which Contacts.firstchar = “R” and Contacts.cTelNum = DialledNumbers.dTelNum. Then in the next step project cName and cTelNum form the selected records.  ***Query processing architecture***  Π represents the projection operation, σ the *selection* operation, and Λ, the AND operation. It is clear that the second set of operations in query processing is less efficient than the first. Query decomposition of the first set gives efficiency. Decomposition is done by (i) analysis, (ii) conjunctive and disjunctive normalization, and (iii) semantic analysis.  Efficient processing of queries needs optimization of steps for query processing. Optimization can be based on cost (number of micro-operations in processing) by evaluating the costs of sets of equivalent expressions. Optimization can also be based on a heuristic approach consisting of |
|  |



|  |  |
| --- | --- |
|  |  |
| the following steps: perform the selection steps and projection steps as early as possible and eliminate duplicate operations.  The query optimizer employs (a) query processing plan generator and (b) query processing cost estimator to provide an efficient plan for query processing.  ata Recovery Process  Data is non-recoverable in case of media failure, intentional attack on the database and transactions logging data, or physical media destruction. However, data recovery is possible in other cases. Figure below shows recovery management architecture. It uses a recovery manager, which ensures atomicity and durability. Atomicity ensures that an uncommitted but started transaction aborts on failure and aborted transactions are logged in log file. Durability ensures that a committed transaction is not affected by failure and is recovered. Stable state databases at the start and at the end of transactions reside in secondary storage. Transaction commands are sent to the recovery manager, which sends fetch commands to the database manager. The database manager processes the queries during the transaction and uses a database buffer. The recovery manager also sends the flush commands to transfer the committed transactions and database buffer data to the secondary storage. The recovery manager detects the results of operations. It recovers lost operations from the secondary storage. Recovery is by detecting the data lost during the transaction.  ***Recovery Management Architecture***  The recovery manager uses a log file, which logs actions in the following manner:   1. Each instruction for a transaction for update (insertion, deletion, replacement, and addition) must be logged. 2. Database read instructions are not logged 3. Log files are stored at a different storage medium. 4. Log entries are flushed out after the final stable state database is stored. | |

D

|  |  |
| --- | --- |
|  |  |
| Each logged entry contains the following fields.    transaction type (begin, commit, or rollback transaction)   * transaction ID * operation-type * object on which the operation is performed * pre-operation and post-operation values of the object.   A procedure called the Aries algorithm is also used for recovering lost data. The basic steps of the algorithm are:   1. Analyse from last checkpoint and identify all dirty records (written again after operation restarted) in the buffer. 2. Redo all buffered operations logged in the update log to finish and make final page. 3. Undo all write operations and restore pre-transaction values. The recovery models used in data recovery processes are as follows:    1. The *full recovery model* creates back up of the database and incremental backup of the changes. All transactions are logged from the last backup taken for the database.  * 1. The *bulk logged recovery model* entails logging and taking backup of bulk data record operations but not the full logging and backup. Size of bulk logging is kept to the minimum required. This improves performance. We can recover the database to the point of failure by restoring the database with the bulk transaction log file backup. This is unlike the full recovery model in which all operations are logged.  * 1. The *simple recovery model* prepares full backups but the incremental changes are not logged. We can recover the database to the most recent backup of the given database.   **Issues relating to Quality of Service (QoS ) Issues:**  [Quality of service](https://en.wikipedia.org/wiki/Quality_of_service) (QoS) mechanism controls the performance, reliability and usability of a telecommunications service. Mobile cellular service providers may offer **mobile QoS** to customers just as the fixed line [PSTN](https://en.wikipedia.org/wiki/PSTN) services providers and [Internet service providers](https://en.wikipedia.org/wiki/Internet_service_providers) may offer QoS. QoS mechanisms are always provided for [circuit switched](https://en.wikipedia.org/wiki/Circuit_switched) services, and are essential for non-elastic services, for example [streaming multimedia.](https://en.wikipedia.org/wiki/Streaming_multimedia) It is also essential in networks dominated by such services, which is the case in today's mobile communication networks, but not necessarily tomorrow.  Mobility adds complication to the QoS mechanisms, for several reasons: | |

* A phone call or other session may be interrupted after a [handover,](https://en.wikipedia.org/wiki/Handover) if the new [base station](https://en.wikipedia.org/wiki/Base_station) is overloaded. Unpredictable handovers make it impossible to give an absolute QoS guarantee during a session initiation phase.
* The pricing structure is often based on per-minute or per-megabyte fee rather than [flat rate,](https://en.wikipedia.org/wiki/Flat_rate) and may be different for different content services.
* A crucial part of QoS in mobile communications is [grade of service,](https://en.wikipedia.org/wiki/Grade_of_service) involving [outage probability](https://en.wikipedia.org/w/index.php?title=Outage_probability&action=edit&redlink=1) (the probability that the mobile station is outside the service coverage area, or affected by co-channel interference, i.e. crosstalk) [blocking probability](https://en.wikipedia.org/wiki/Blocking_probability) (the probability that the required level of QoS cannot be offered) and [scheduling starvation.](https://en.wikipedia.org/wiki/Scheduling_starvation) These performance measures are affected by mechanisms such as [mobility management,](https://en.wikipedia.org/wiki/Mobility_management) [radio resource management,](https://en.wikipedia.org/wiki/Mobility_management) [admission control,](https://en.wikipedia.org/wiki/Admission_control) [fair scheduling,](https://en.wikipedia.org/wiki/Admission_control) [channel-](https://en.wikipedia.org/wiki/Channel-dependent_scheduling) [dependent](https://en.wikipedia.org/wiki/Channel-dependent_scheduling) [scheduling](https://en.wikipedia.org/wiki/Channel-dependent_scheduling) etc.

**Types**

* [Factors affecting QoS](https://en.wikipedia.org/wiki/Mobile_QoS#Factors_affecting_QoS)
* [Measurement of QoS](https://en.wikipedia.org/wiki/Mobile_QoS#Measurement_of_QoS)
* [Cellular GoS](https://en.wikipedia.org/wiki/Mobile_QoS#Cellular_GoS)
* [Cellular audio quality](https://en.wikipedia.org/wiki/Mobile_QoS#Cellular_audio_quality)

Factors affecting QoS

Many factors affect the quality of service of a mobile network.[ It is correct to look at QoS mainly from the customer's point of view, that is, QoS as judged by the user. There are standard metrics of QoS to the user that can be measured to rate the QoS. These metrics are: the **coverage**, **accessibility** (includes GoS), and the **audio quality**. In **coverage** the strength of the signal is measured using test equipment and this can be used to estimate the size of the cell. **Accessibility** is about determining the ability of the network to handle successful calls from mobile-to-fixed networks and from mobile-to-mobile networks. The **audio quality** considers monitoring a successful call for a period of time for the clarity of the communication channel. All these indicators are used by the telecommunications industry to rate the quality of service of a network.

Measurement of QoS

The QoS in industry is also measured from the perspective of an expert (e.g. teletraffic engineer). This involves

assessing the network to see if it delivers the quality that the network planner has been

required to target. Certain tools and methods (protocol analysers, drive tests and Operation and Maintenance measurements), are used for this QoS measurement:

* Protocol analysers are connected to BTSs, BSCs, and MSCs for a period of time to check for problems in the cellular network. When a problem is discovered the staff can record it and it can be analysed.
* Drive tests allow the mobile network to be tested through the use of a team of people who take the role of users and take the QoS measures discussed above to rate the QoS of the network. This test does not apply to the entire network, so it is always a statistical sample.
* In the Operation and Maintenance Centres (OMCs), counters are used in the system for various events which provide the network operator with information on the state and quality of the network.
* Finally, customer complaints are a vital source of feedback on the QoS, and must not be ignored.

Cellular GoS

In general, [grade of service](https://en.wikipedia.org/wiki/Grade_of_service) (GoS) is measured by looking at traffic carried, traffic offered and calculating the traffic blocked and lost. The proportion of lost calls is the measure of GoS. For cellular circuit groups an acceptable GoS is 0.02. This means that two users of the circuit group out of a hundred will encounter a call refusal during the busy hour at the end of the planning period. The grade of service standard is thus the acceptable level of traffic that the network can lose. GoS is calculated from the [Erlang-](https://en.wikipedia.org/wiki/Erlang_unit#Erlang_B_formula) [B](https://en.wikipedia.org/wiki/Erlang_unit#Erlang_B_formula) [formula,](https://en.wikipedia.org/wiki/Erlang_unit#Erlang_B_formula) as a function of the number of channels required for the offered traffic intensity.

Cellular audio quality

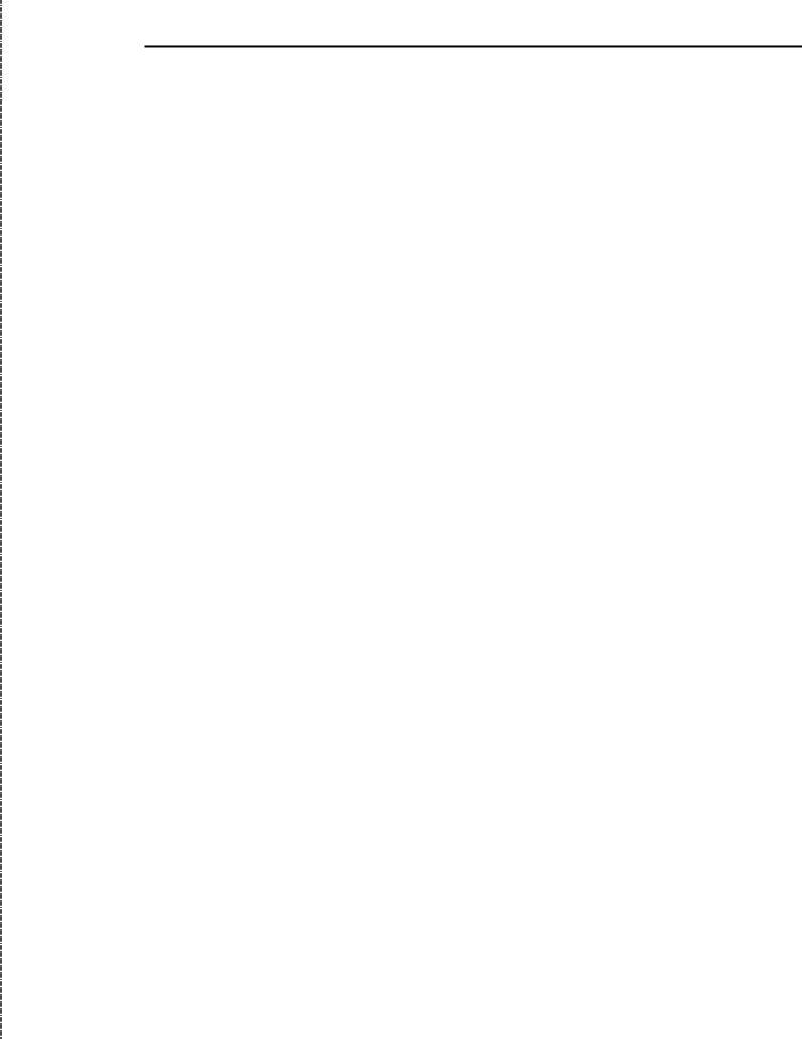
The audio quality of a cellular network depends on, among other factors, the modulation scheme (e.g., [FSK,](https://en.wikipedia.org/wiki/Frequency-shift_keying) [QPSK](https://en.wikipedia.org/wiki/QPSK)) in use, matching to the channel characteristics and the processing of the received signal at the receiver using [DSPs.](https://en.wikipedia.org/wiki/Digital_Signal_Processor)



Data Dissemination

**Data Dissemination and Synchronization :** Communications Asymmetry, Classification of Data Delivery Mechanisms, Data Dissemination, Broadcast Models, Selective Tuning and Indexing Methods, Data Synchronization – Introduction, Software, and Protocols.

Ongoing advances in communications including the proliferation of internet, development of mobile and wireless networks, high bandwidth availability to homes have led to development of a wide range of new-information centered applications. Many of these applications involve data dissemination, i.e. delivery of data from a set of producers to a larger set of consumers.



**Data dissemination** entails distributing and pushing data generated by a set of computing systems or broadcasting data from audio, video, and data services. The output data is sent to the mobile devices. A mobile device can select, tune and cache the required data items, which can be used for application programs.

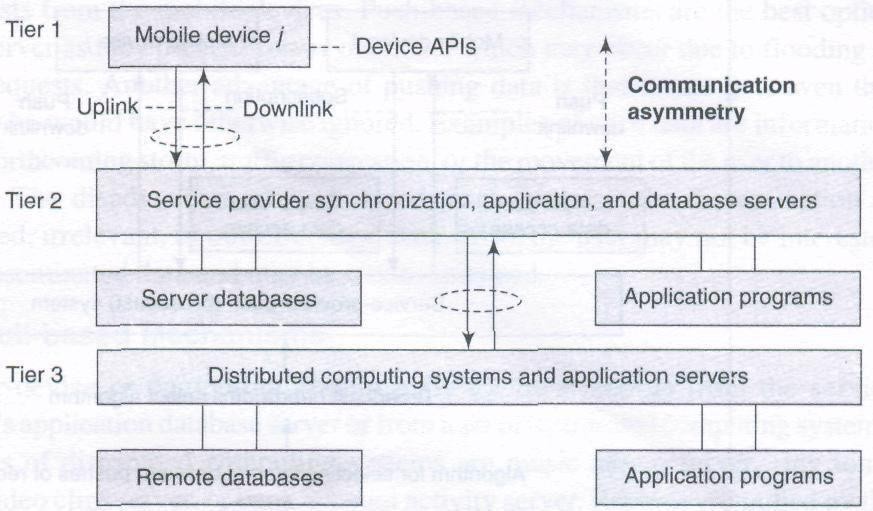
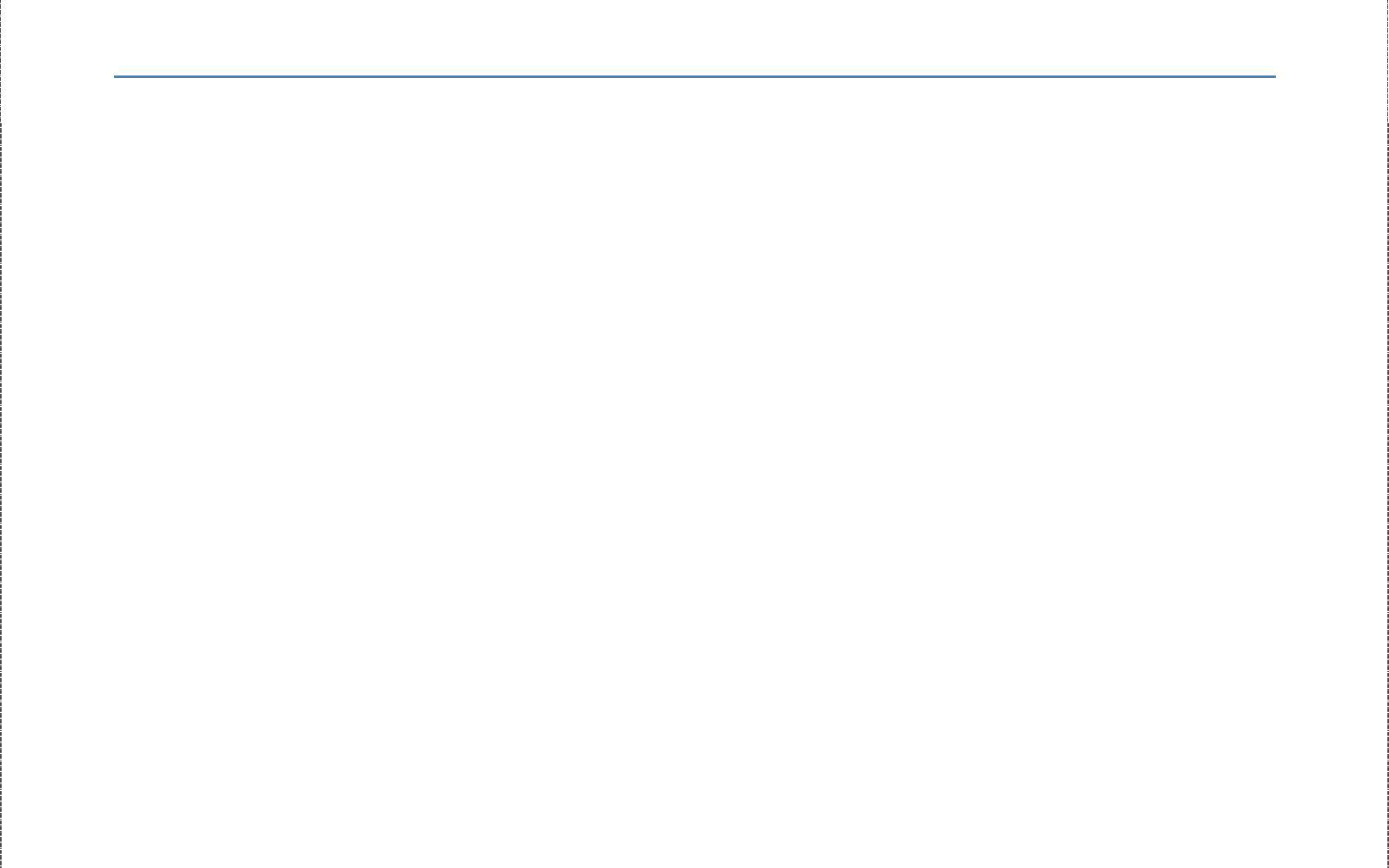
Efficient utilization of wireless bandwidth and battery power are two of the most important problems facing software designed for mobile computing. Broadcast channels are attractive in tackling these two problems in wireless data dissemination. Data disseminated through broadcast channels can be simultaneously accessed by an arbitrary number of mobile users, thus increasing the efficiency of bandwidth usage.

Communications Asymmetry

One key aspect of dissemination-based applications is their inherent communications asymmetry. That is, the communication capacity or data volume in the downstream direction (from servers-to-clients) is much greater than that in the upstream direction (from clients-to- servers). Content delivery is an asymmetric process regardless of whether it is performed over a symmetric channel such as the internet or over an asymmetric one, such as cable television (CATV) network. Techniques and system architectures that can efficiently support asymmetric applications will therefore be a requirement for future use.

Mobile communication between a mobile device and a static computer system is intrinsically asymmetric. A device is allocated a limited bandwidth. This is because a large number of devices access the network. Bandwidth in the downstream from the server to the device is much larger than the one in the upstream from the device to the server. This is because mobile devices have limited power resources and also due to the fact that faster data transmission rates for long intervals of time need greater power dissipation from the devices. InGSM networks data transmission rates go up to a maximum of 14.4 kbps for

both uplink and downlink. The communication is symmetric and this symmetry can be maintained because GSMis only used for voice communication.



**Communication asymmetry in uplink and downlink and participation of device APIs and distributed computing systems when an application runs**

The above figure shows communication asymmetry in uplink and downlink in a mobile network.The participation of device APIs and distributed computing systems in the running of an application is also shown.

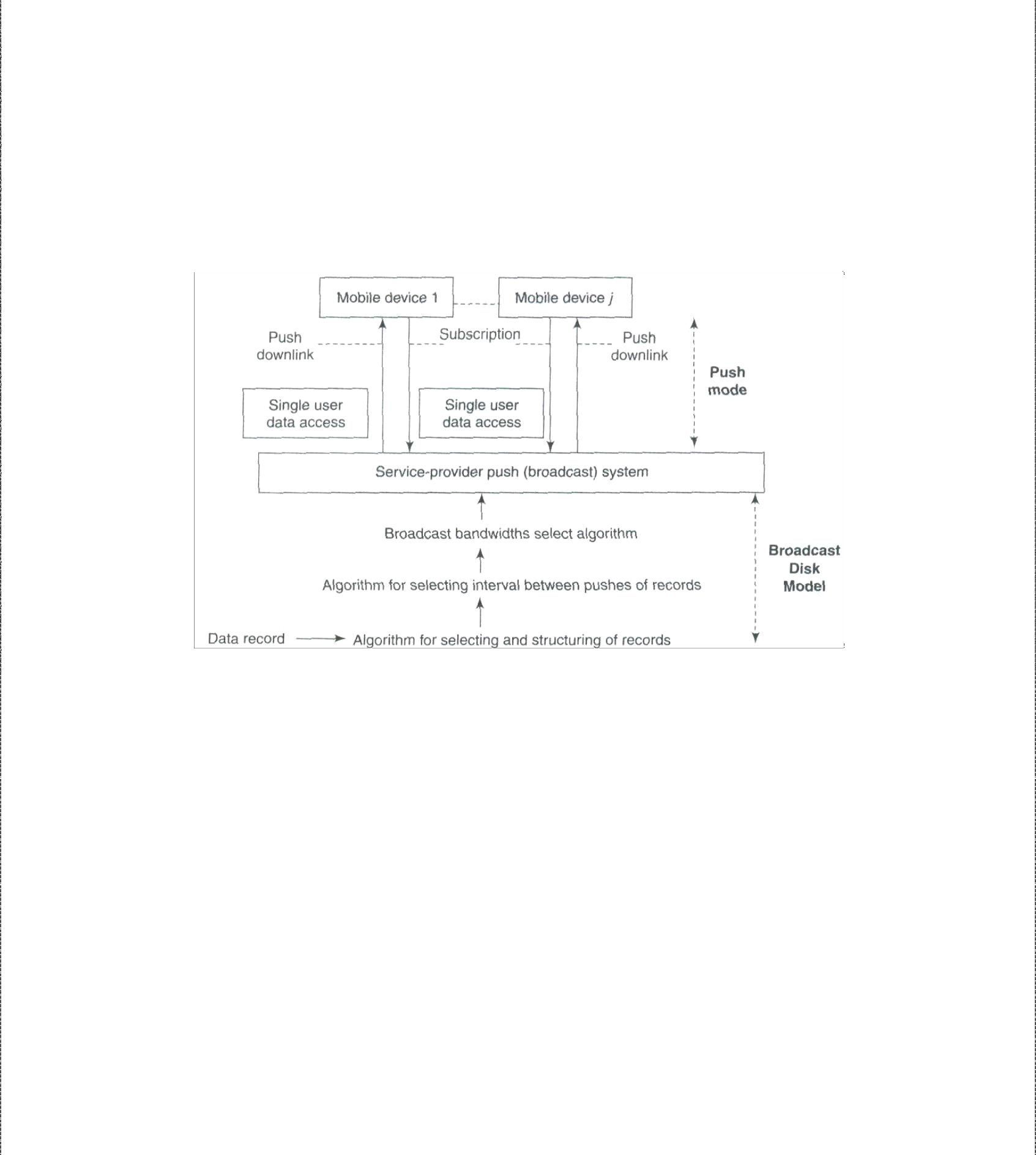
**Classification of Data-Delivery Mechanisms**

There are two fundamental information delivery methods for wireless data applications:Point-to-Point access and Broadcast. Compared with Point-to-Point access, broadcast is a more attractive method. A single broadcast of a data item can satisfy all the outstanding requests for that item simultaneously. As such, broadcast can scale up to an arbitrary number of users.

There are three kinds of **broadcast models**, namely *push-based* broadcast, *On- demand* (or *pull-based*) broadcast, and *hybrid* broadcast. In push based broadcast, the server disseminates information using a periodic/aperiodic broadcast program (generally without any intervention of clients). In on demand broadcast, the server disseminates information based on the outstanding requests submitted by clients; In hybrid broadcast, push based broadcast and on demand data deliveries are combined to complement each other. In addition, mobile computers consume less battery power on monitoring broadcast channels to receive data than accessing data through point-to-point communications.

Data-delivery mechanisms can be classified into three categories, namely, push- based mechanisms (publish-subscribe mode), pull-based mechanisms (on-demand mode), and hybrid mechanisms (hybrid mode).

Push-based Mechanisms



The server pushes data records from a set of distributed computing systems. Examples are advertisers or generators of traffic congestion, weather reports, stock quotes, and news reports.The following figure shows a push-based data-delivery mechanism in which a server or computing system pushes the data records from a set of distributed computing systems. The data records are pushed to mobile devices by broadcasting without any demand. The push mode is also known as ***publish-subscribe*** mode in which the data is pushed as per the subscription for a push service by a user. The subscribed query for a data record is taken as perpetual query till the user unsubscribe to that service. Data can also be pushed without user subscription.

**Push-based data-delivery mechanism**

Push-based mechanisms function in the following manner:

1. A structure of data records to be pushed is selected. An algorithm provides an adaptable multi-level mechanism that permits data items to be pushed uniformly or non-uniformly after structuring them according to their relative importance.
2. Data is pushed at selected time intervals using an adaptive algorithm. Pushing only once saves bandwidth. However, pushing at periodic intervals is

important because it provides the devices that were disconnected at the timeof previous push with a chance to cache the data when it is

pushed again.

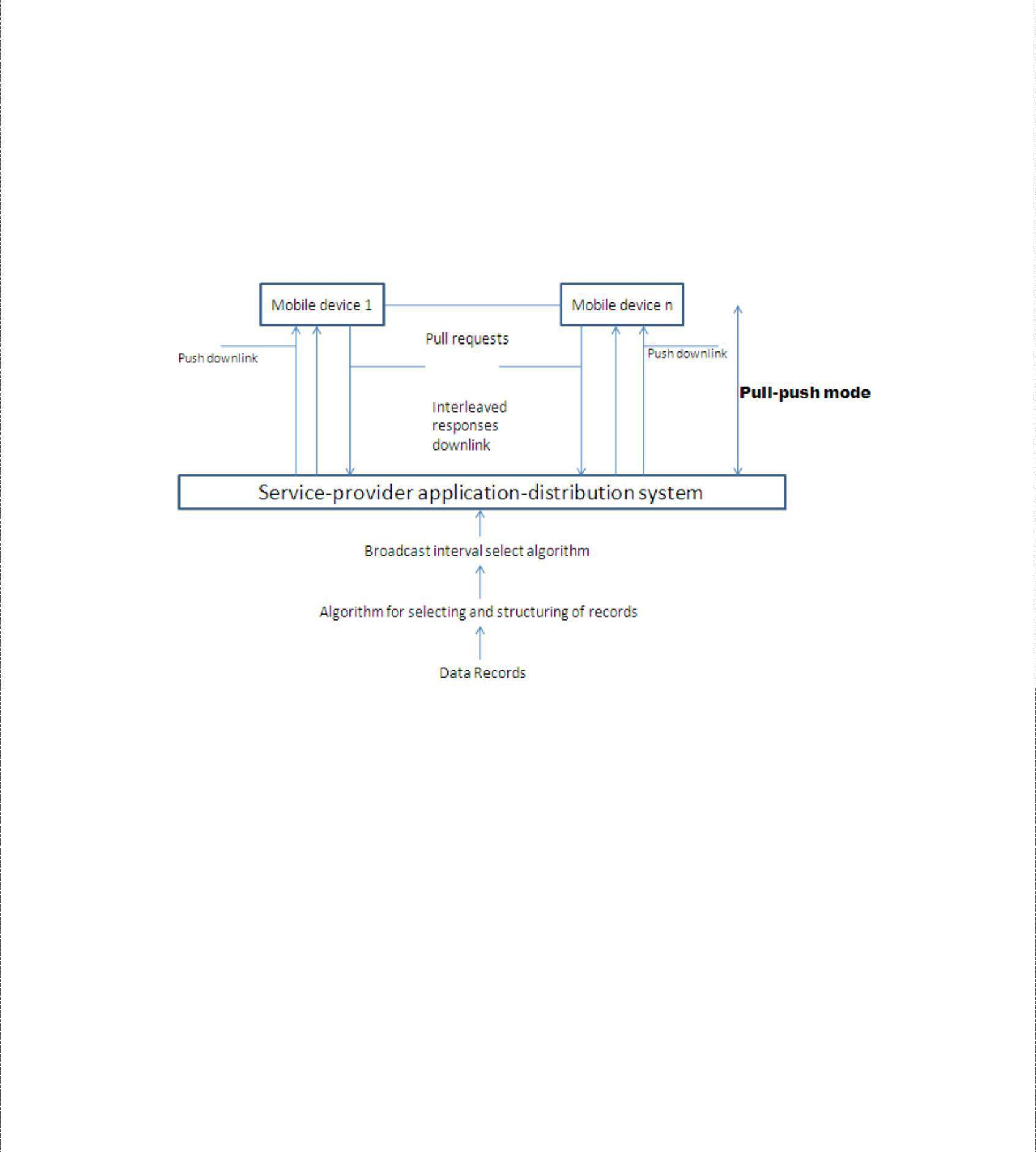
1. Bandwidths are adapted for downlink (for pushes) using an algorithm. Usually higher bandwidth is allocated to records having higher number of subscribers or to those with higher access probabilities.

|  |  |
| --- | --- |
|  | 4. A mechanism is also adopted to stop pushes when a device is handed over to another cell.  The application-distribution system of the service provider uses these algorithms and adopts bandwidths as per the number of subscribers for the published data records. On the device handoff, the subscription cancels or may be passed on to new service provider system.  *Advantages of Push based mechanisms:*  Push-based mechanisms enable broadcast of data services to multiple devices.    The server is not interrupted frequently by requests from mobile devices.  These mechanisms also prevent server overload, which might be caused by flooding of device requests  Also, the user even gets the data he would have otherwise ignored such as traffic congestion, forthcoming weather reports etc  *Disadvantages:*  Push-based mechanisms disseminate of unsolicited, irrelevant, or out-of-context data,which may cause inconvenience to the user.  Pull based Mechanisms  The user-device or computing system pulls the data records from the service provider's application database server or from a set of distributed computing systems. Examples are musicalbum server, ring tones server, video clips server, or bank account activity server. Records are pulled by the mobile devices on demand followed by the selective response from the server. Selective response means that server transmits data packets as response selectively, for example, after client-authentication, verification, or subscription account check. The pull mode is also known as the on-demand mode. The following figure shows a pull-based data-delivery mechanism in which a device pulls (demands) from a server or computing system, the data records generated by a set of distributed computing systems. |
|  |
|  |
|  |

|  |  |
| --- | --- |
|  | **Pull based Delivery Mechanism**  Pull-based mechanisms function in the following manner:   1. The bandwidth used for the uplink channel depends upon the number of pull requests. 2. A pull threshold is selected. This threshold limits the number of pull requests in a given period of time. This controls the number of server interruptions. 3. A mechanism is adopted to prevent the device from pulling from a cell, which has handed over the concerned device to another cell. On device handoff, thesubscription is cancelled or passed on to the new service provider cell   In pull-based mechanisms the user-device receives data records sent by server on demand only.  *Advantages of Pull based mechanisms:*  With pull-based mechanisms, no unsolicited or irrelevant data arrives at the device and the relevant data is disseminated only when the user asks for it.  Pull-based mechanisms are the best option when the server has very little contention and is able to respond to many device requests within expected time intervals.  *Disadvantages:*  The server faces frequent interruptions and queues of requests at the server may cause congestion in cases of sudden rise in demand for certain data record.  In on-demand mode, another disadvantage is the energy and bandwidth required forsending the requests for hot items and temporal records |
|  |
|  |
|  |
|  |



Hybrid Mechanisms



A hybrid data-delivery mechanism integrates pushes and pulls. The hybrid mechanism is also known as interleaved-push-and-pull (IPP) mechanism. The devices use the back channel to send pull requests for records, which are not regularly pushed by the front channel. The front channel uses algorithms modeled as broadcast disks and sends the generated interleaved responses to the pull requests. The user device or computing system pulls as well receives the pushes of the data records from the service provider's application server or database server or from a set of distributedcomputing systems. Best example would be a system for advertising and selling music albums. The advertisements are pushed and the mobile devices pull for buying the album.

**Hybrid interleaved push-pull-based data-delivery mechanism**

The above figure shows a hybrid interleaved, push-pull-based data-delivery mechanism in which a device pulls (demands) from a server and the server interleaves the responses along with the pushes of the data records generated by a set of distributed computing systems. Hybrid mechanisms function in the following manner:

1. There are two channels, one for pushes by front channel and the other for pulls by back channel.
2. Bandwidth is shared and adapted between the two channels depending upon the number of active devices receiving data from the server and the number of devices requesting data pulls from the server.
3. An algorithm can adaptively chop the slowest level of the scheduled pushes successively The data records at lower level where the records are assigned lower priorities can have long push intervals in a broadcasting model.

*Advantages of Hybrid mechanisms*:



The number of server interruptions and queued requests are significantly reduced.

*Disadvantages:*



IPP does not eliminate the typical server problems of too many interruptions and queued requests.



Another disadvantage is that adaptive chopping of the slowest level of scheduled pushes.

**Selective Tuning and Indexing Techniques**

The purpose of pushing and adapting to a broadcast model is to push records of greater interest with greater frequency in order to reduce access time or average access latency. A mobile device does not have sufficient energy to continuously cache the broadcast records and hoard them in its memory. A device has to dissipate more power if it gets each pushed item andcaches it. Therefore, it should be activated for listening and caching only when it is going to receive the selected data records or buckets of interest. During remaining time intervals, that is, when the broadcast data buckets or records are not of its interest, it switches to idle or power down mode.

Selective tuning is a process by which client device selects only the required pushed buckets or records, tunes to them, and caches them. Tuning means getting ready for caching at those instants and intervals when a selected record of interest broadcasts. Broadcast data has astructure and overhead. Data broadcast from server, which is organized into buckets, is interleaved. The server prefixes a directory, hash parameter (from which the device finds the key), or index to the buckets. These prefixes form the basis of different methods of selective

tuning. Access time (**taccess)** is the time interval between pull request from device and reception of response from broadcasting or data pushing or responding server. Two important factors affect taccess – (i) number and size of the records to be broadcast and (ii) directory- or cache- miss factor (if there is a miss then the response from the server can be received only in subsequent broadcast cycle or subsequent repeat broadcast in the cycle).

Directory Method

One of the methods for selective tuning involves broadcasting a directory as overhead at the beginning of each broadcast cycle. If the interval between the start of the broadcast cycles is T, then directory is broadcast at each successive intervals of T.



A directory can be provided which specifies when a specific record or data item appears in data being broadcasted. For example, adirectory (at header of the cycle) consists of directory start sign, 10, 20, 52, directory end sign. It means that after the directory end sign, the 10th, 20th and 52nd buckets contain the data items in response to the device request. The device selectively tunes to these buckets from the broadcast data.

A device has to wait for directory consisting of start sign, pointers for locating buckets orrecords, and end sign. Then it has to wait for the required bucket or record before it can get

tuned to it and, start caching it. Tuning time ttune is the time taken by the device for selection ofrecords. This includes the time lapse before the device starts receiving data from the server. In other words, it is the sum of three periods—time spent in listening to the directory signs and pointers for the record in order to select a bucket or record required by the device, waiting for the buckets of interest while actively listening (getting the incoming record wirelessly), and caching the broadcast data record or bucket.

The device selectively tunes to the broadcast data to download the records of interest.

When a directory is broadcast along with the data records, it minimizes ttune and taccess. The device saves energy by remaining active just for the periods of caching the directory and the data buckets. For rest of the period (between directory end sign and start of the required bucket), it remains idle or performs application tasks. Without the use of directory for tuning,

ttune = taccess and the device is not idle during any time interval.

Hash-Based Method

Hash is a result of operations on a pair of key and record. Advantage of broadcasting a hash is that it contains a fewer bits compared to key and record separately. The operations are done bya hashing function. From the server end the hash is broadcasted and from the device end a key is extracted by computations from the data in the record by operating the data with a function called hash function (algorithm). This key is called hash key.

Hash-based method entails that the hash for the hashing parameter (hash key) is broadcasted. Each device receives it and tunes to the record as per the extracted key. In this method, the records that are of interest to a device or those required by it are cached from the broadcast cycle by first extracting and identifying the hash key which provides the location of the record. This helps in tuning of the device. Hash-based method can be described as follows:

1. A separate directory is not broadcast as overhead with each broadcast cycle.
2. Each broadcast cycle has hash bits for the hash function H, a shift function S,

and the data that it holds. The function S specifies the location of the record or remaining part of the record relative to the location of hash and, thus, the time interval for wait before the record can be tuned and cached.

1. Assume that a broadcast cycle pushes the hashing parameters **H*(Rí)*** [H and S] and record ***Rí***. The functions H and S help in tuning to the H(***Rí***) and hence to ***Rí*** as follows—H gives a key which in turn gives the location of H*(****Rí****)* in the broadcast data. In case H generates a key that does not provide the location of H(***Rí***) by itself, then the device computes the location from S

after the location of H(***Rí***). That location has the sequential records ***Rí***

and the devices tunes to the records from these locations.

1. In case the device misses the record in first cycle, it tunes and caches that in next or some other cycle.

Index-Based Method

Indexing is another method for selective tuning. Indexes temporarily map the location of the buckets. At each location, besides the bits for the bucket in record of interest data, an offset value may also be specified there. While an index maps to the absolute location from the beginning of a broadcast cycle, an offset index is a number which maps to the relative location after the end of present bucket of interest. Offset means a value to be used by the device along with the present location and calculate the wait period for tuning to the next bucket. Allbuckets have an offset to the beginning of the next indexed bucket or item.

Indexing is a technique in which each data bucket, record, or record block of interest is assigned an index at the previous data bucket, record, or record block of interest to enable the device to tune and cache the bucket after the wait as per the offset value. The server transmits this index at the beginning of a broadcast cycle as well as with each bucket corresponding to data of interest to the device. A disadvantage of using index is that it extends the broadcast

cycle and hence increases taccess.

The index I has several offsets and the bucket type and flag information. A typical index may consist of the following:

1. Ioffset(1) which defines the offset to first bucket of nearest index.
2. Additional information about *Tb,* which is the time required for caching the bucket bits in full after the device tunes to and starts caching the bucket. This enables transmission of buckets of variable lengths.
3. Ioffset (next) which is the index offset of next bucket record of interest.