UNIT-III: Broadcasting in Mobile Environment

1 Introduction to Broadcasting

- Broadcasting services are by their nature downlink transmissions with programs in a time sequence determined by the broadcaster (so called linear broadcasting). Enhanced broadcasting services complement the traditional broadcasting services and offer non-linear services (in an order and at time determined by the viewer) by means of interactivity, time-shifted viewing and continued reception at any location.
- Enhanced broadcasting services could be offered by terrestrial broadcasting network, cable, IPTV and satellite networks in combination with broadband networks; so called hybrid broadcast-broadband (HBB) services. In addition, some delivery means can offer enhanced broadcasting to a certain extent. For instance DTTB can offer mobile and portable reception, MTV handheld reception and cable TV and IPTV may offer video-on-demand services.

❖ Enhanced television broadcasting

- Enhanced broadcasting services are developed around three concepts:
  - TV anytime, aiming at watching a specific program at the time by choice of the viewer. Time shifted viewing is in particular of interest for shows, documentaries, movies etc., but a relative short time shift for sports and news programs is also popular.
  - TV anywhere, aiming at watching the broadcast service not only in the living room, but also in other rooms, on the move, etc. Mobile devices like smart phones and tablet computers are used for this application.
  - Interactivity, aiming at contributing or reacting by the viewer to a specific program, demanding for additional information regarding the program or receiving programs or information of particular interest.

❖ Enhanced audio broadcasting

- The service concepts in audio broadcasting follow similar patterns as in television broadcasting. However, the "anywhere" concept is much more developed in audio broadcasting. Reception of analogue audio broadcasting in FM and AM and DTAB takes place almost everywhere: with portable receivers or high-end audio-sets in every room in the house, with car radios while driving, outside and in public places with small pocket radios and mobile phones and in waiting rooms and shopping centers by means of central audio installations.
- Streaming via the Internet is becoming a very important means of delivery. Thousands of radio stations from all over the world can be received in good quality with radio receivers equipped with Internet access, or with mobile phones and computers.
- Interactivity and hybrid broadcast-broadband (HBB) is also developing in audio broadcasting. HBB radio receivers with a screen for displaying additional personalized information appear on the market.
- Radio DNS (Domain Name Service) is an initiative to help broadcasters to offer HBB services with the aim that the listener is unaware that the linear broadcasting services and the personalized broadband services are combined. This is achieved by making use of existing identifiers of the radio station used with, e.g. FM-RDS, DAB, DRM or IBOC and to locate the IP delivered services of that station.

❖ Broadcast and broadband delivery

- It is expected that linear broadcasting services aimed at reception by the general public in a country or region will be enhanced with individualized services delivered by fixed (including domestic distribution by WLAN) and mobile networks. When broadband connections are available to a large part of the population, broadband will

not only be the main means of delivery for individual non-linear broadcasting, but could also deliver linear broadcasting to the general public.

- The relative importance of broadcasting and broadband delivery will be different from country to country depending on the market conditions and the regulatory situation. It may also be different for audio broadcasting and television services.
- Figure shows in a matrix the position of broadcast (BC) and broadband (BB) delivery with regard to the linear and non-linear broadcasting services.

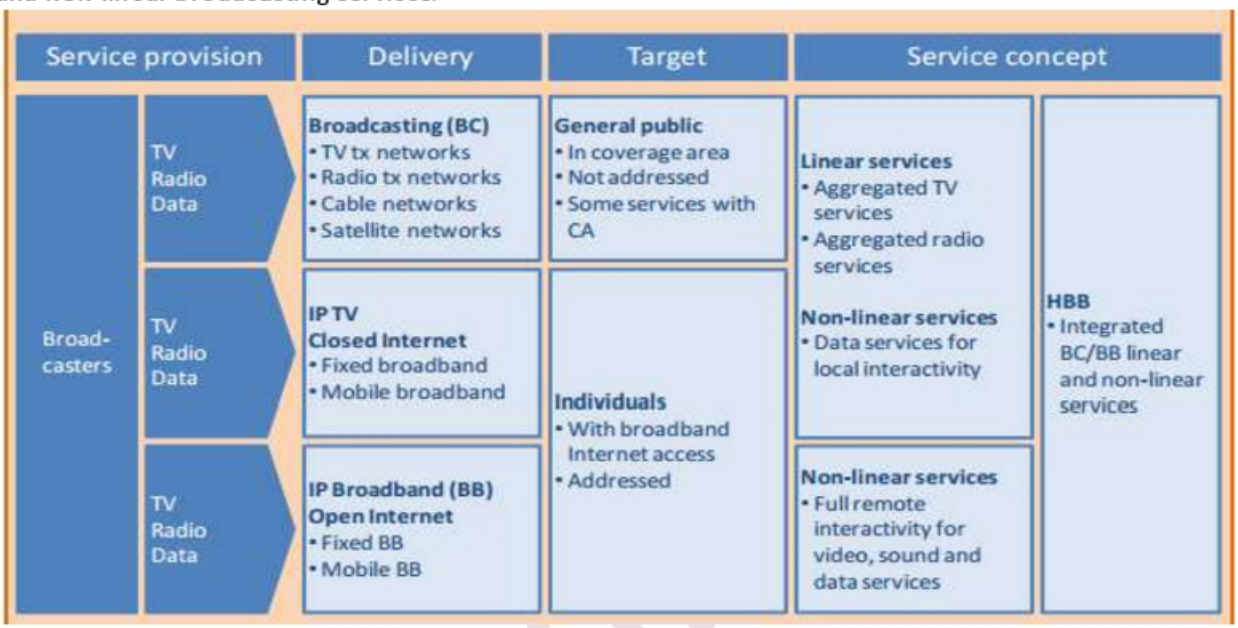


Figure 1: Position of broadcast (BC) and broadband (BB) delivery

- The importance of broadband delivery is expected to increase and will enable integrated hybrid broadcast-broadband (HBB) services. It is not expected that broadband will replace broadcast as the main means of delivery for linear broadcasting to the general public, but it cannot be excluded on the long term. It will depend on national market conditions and regulatory situation.

❖ **Issues in broadcasting**

- Local market requirements.
- Existing transmission networks and receivers.
- Alternative means of content delivery, including IP broadband, via mobile, fixed and satellite networks.
- Regional and international regulatory requirements regarding the use of the frequency spectrum and in particular the impact of decisions adopted at the World Radiocommunication Conference (WRC-12).
- Existing broadcasting transmission standards and future developments.
- Demands of spectrum from other than broadcasting services.

2 Data Management Issues in Mobile Environment

- The mobile computing environment is observed as a distributed computing. The complete database may be distributed among wired components as in mobile switching stations, this is one approach. But in next approach the entire database is being distributed in wired and wireless components of the computer systems. Some of the parameters that influence and complicate database management are,
  - Design of database
  - Replication of data

- **Design of database**

➢ The mobility of clients (hosts-MH) and disconnection between hosts and servers is very difficult to predict. In addition to this the dynamic nature of constantly changing location has to be updated carefully and it also adds more complexity to system design.

- **Replication of data**

➢ In mobile computing scenario the data is partially replicated in different places and the availability of these duplicates is updated periodically. There is also consistency management and version control available.

3 Data Delivery Models

➢ With the Explosion of internet techniques and popularity of mobile terminals such as laptops, personal digital assistants, people with battery powered MTs wish to access various kinds of web services over wireless networks at any time any place. However, existing wireless internet services are limited by the constraints of mobile environments such as narrow bandwidth, asymmetric communication channels, unstable connectivity and limitations of battery technologies. The request-response type protocol can't scale up these problems. Data Dissemination is a type of protocol where server initiates and manages the transfer of data as well as updates and manages the transfer of data as well as updates. Data Dissemination also helps in maintaining data consistency and cache management. It entails distributing and pushing data generated by a set of computing systems and broadcasting data from audio, video and data services. The output data is sent to mobile devices. Then, mobile device can select, cache and tune required data items.

➢ **Communication Type**

- Symmetric Communication:
  - It is a type of communication in which transmission rates are same for both uplink & downlink.
  - E.g.-GSM network: Transmission rate of GSM network is 14.4 kbps for both uplink & downlink.
- Asymmetric Communication:
  - It is a type of communication in which downlink capacity is much greater then uplink capacity.
  - E.g. -I-mode: downlink = 384 kbps Uplink = 64 kbps.

➢ Asymmetrical communication is best suited for mobile environment. In this architecture, there will be a stationary server continuously broadcasting different data items. Mobile Clients continuously listen to the channel and access the data of their interest. In mobile computing environments, large number of devices are attached or access the network. Bandwidth in the downstream from the server to the device is much greater than the upstream from device to server. This is because mobile devices have limited power resources.

4 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. In GSM 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 GSM is only used for voice communication.

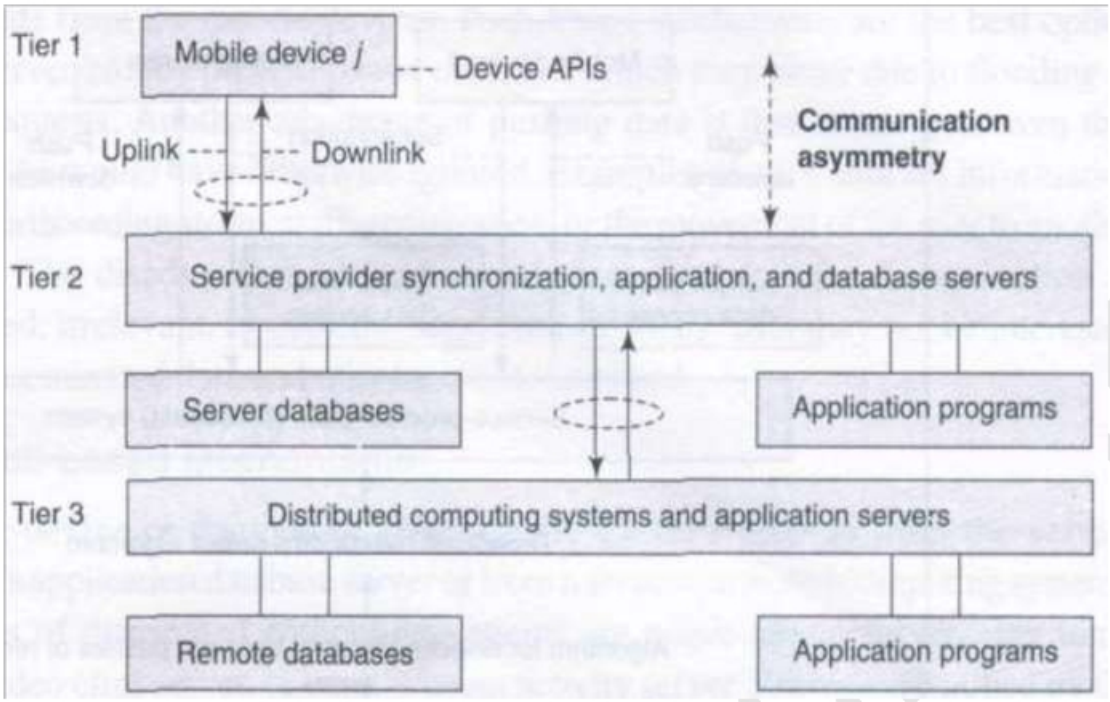


Figure 2: Communication asymmetry in uplink and downlink and participation of device APIs and distributed computing systems when an application runs

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.

5 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).

6 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.

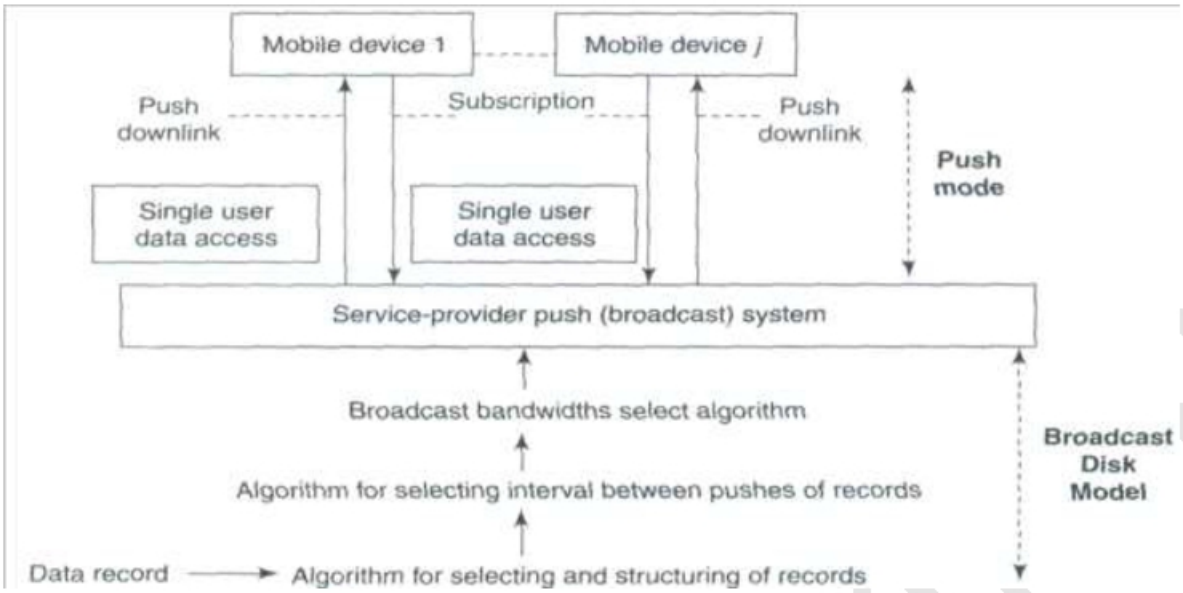


Figure 3: Push-based data-delivery mechanism

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 time of previous push with a chance to cache the data when it is pushed again.
3. 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.

7 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 music album 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.

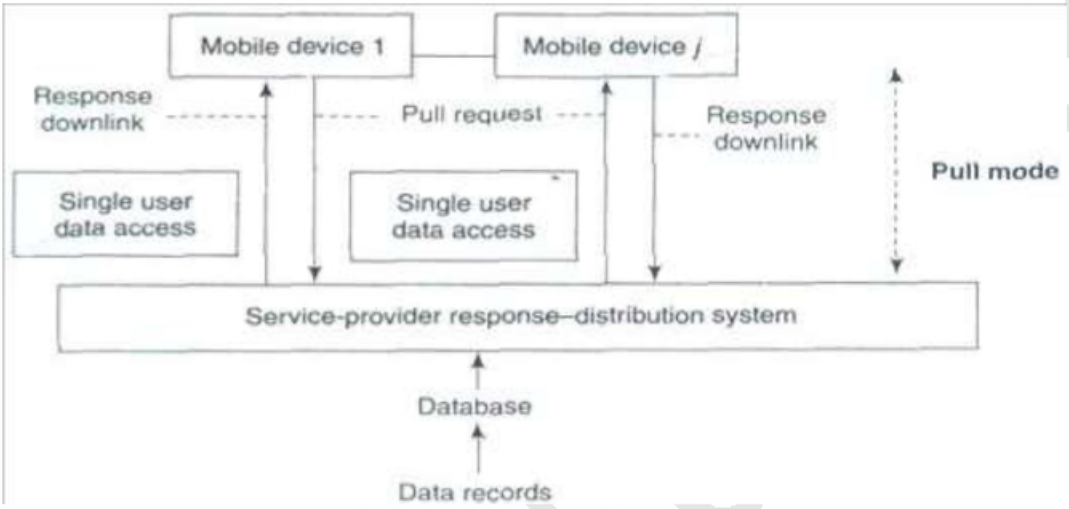


Figure 4: Pull based Delivery Mechanism

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, the subscription 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 for sending the requests for hot items and temporal records

8 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 distributed computing 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.

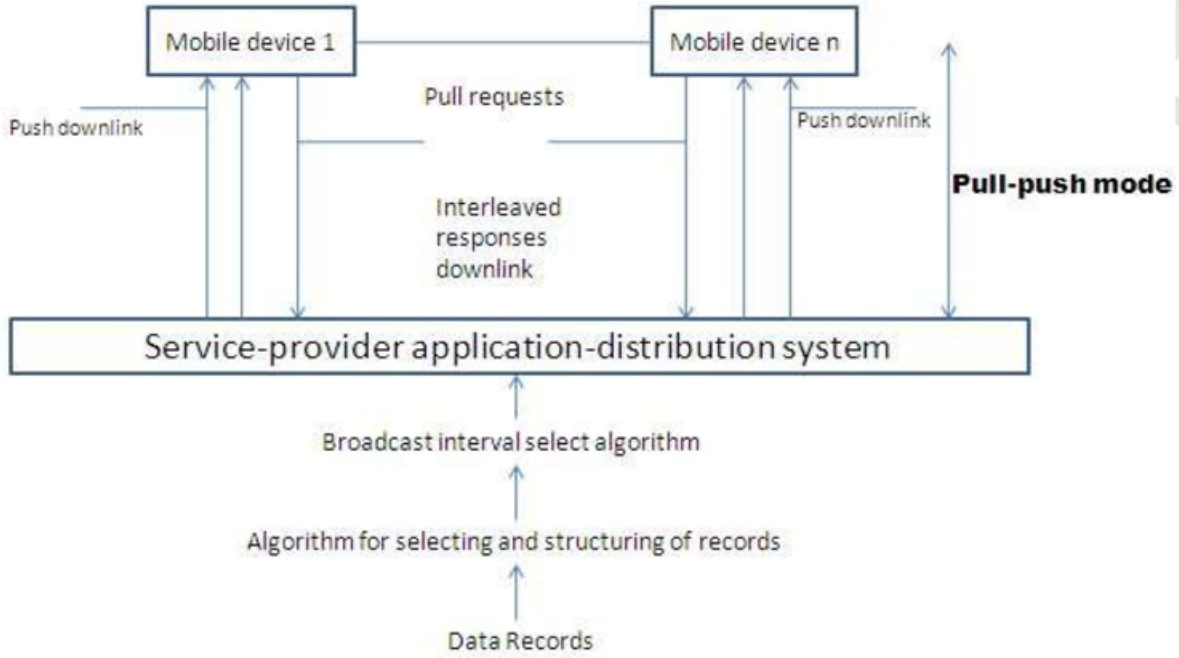


Figure 5: Hybrid interleaved push-pull-based data-delivery mechanism

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.

9 Explanation of Push Taxonomy

- Both Pull and Push can be periodic or aperiodic.
- Periodic push can be for set of data updated or created in regular interval. Such as Stock prices.
  - Useful in situation where clients not available always.
  - Client request processing load is high
  - There are several clients for what is being pushed.
- Aperiodic push is a kind publish and subscribe. There is no periodic interval for sending out the data.
  - It assumes that the clients are always listening.
  - What is sent out is not crucial.
  - Uses downstream channel more effectively.
- Aperiodic pull is found in traditional system or request/response data delivery.
- Periodic pull arises when client uses polling to obtain data. It uses a regular schedule to request for data. Useful I application such as remote sensing.

10 Point-to-point and 1-to-N

- In a P-to-P communication data sent from source to a single client.
- P-to-P is not suitable for scaling up.
- In 1-to-N communication data is sent to a number of clients.
- 1-to-N can be either multicast or broadcast.
- Usually, multicasting is implemented by sending data to a router which maintains the list of recipients and forwards the data to those.
- So clients interests are known a priori as opposed to broadcast where clients are unidentified.
- In 1-to-N broadcast clients receive data for which they may not be interested.

11 Broadcast Disk

- It is a periodic push. Clients wait until the item appears. So in a sense it is like accessing of a storage device who average latency is half the interval at the searched item repeats. Periodic push can use either P-to-P or 1-to-N link. But 1-to-N more likely.
- The periodic push is essentially akin to a secondary storage device access as far as the client is concerned.
- If every item in a broadcast schedule appears only once then the worst case wait for the client is same as one broadcast period.
- It rarely is the case that all items are accessed equally frequently.
- It tends to be skewed to a few hot spots. So it makes sense to capture the pattern of access in a broadcast program.
- Broadcast disk is a paradigm for organizing the structure of a periodic broadcast program.
- Proposes a mechanism called Broadcast Disks to provide database access to mobile clients.
- Server continuously and repeatedly broadcasts data to a mobile client as it goes by.
- Multiple disks of different sizes are superimposed on the broadcast medium.
- Exploits the client storage resources for caching data.

In traditional client-server information systems, clients initiate data transfers by sending requests to a server. Such systems are pull-based; the clients "pull" data from the server in order to provide data to locally running applications. Pull based systems are a poor match for asymmetric communications environments, as they require substantial upstream communications capabilities. To address this incompatibility, a new information system architecture that exploits the relative abundance of downstream communication capacity in asymmetric environments. This new architecture is called Broadcast Disks. The central idea is that the servers exploit their

advantage in bandwidth by broadcasting data to multiple clients. We refer to this arrangement as a push based architecture; data is pushed from the server out to the clients.

In this approach, a server continuously and repeatedly broadcasts data to the clients. In effect, the broadcast channel becomes a "disk" from which clients can retrieve data as it goes by.

The broadcast is created by assigning data items to different "disks" of varying sizes and speeds, and then multiplexing the disks on the broadcast channel. Items stored on faster disks are broadcast more often than items on slower disks. This approach creates a memory hierarchy in which data on the fast disks are "closer" to the clients than data on slower disks. The number of disks, their sizes, and relative speeds can be adjusted, in order to more closely match the broadcast with the desired access probabilities at the clients. If the server has an indication of the client access patterns (e.g., by watching their previous activity or from a description of intended future use from each client), then hot pages (i.e., those that are more likely to be of interest to a larger part of the client community) can be brought closer while cold pages can be pushed further away.

Data items are assigned to different logical disks so that data items in the same range of access probabilities are grouped on the same disk. Data items are then selected from the disks for broadcast according to the relative broadcast frequencies assigned to the disks. This is achieved by further dividing each disk into smaller, equal size units called chunks, broadcasting a chunk from each disk each time, and cycling through all the chunks sequentially over all the disks. A minor cycle is defined as a sub cycle consisting of one chunk from each disk. Consequently, data items in a minor cycle are repeated only once. The number of minor cycles in a broadcast cycle equals the Least Common Multiple (LCM) of the relative broadcast frequencies of the disks. Conceptually, the disks can be conceived as real physical disks spinning at different speeds, with the faster disks placing more instances of their data items on the broadcast channel. The algorithm that generates broadcast disks is given below.

Broadcast Disks Generation Algorithm {
  Order the items in decreasing order of access popularities;
  Allocate items in the same range of access probabilities on a different *disk*;
  Choose the relative broadcast frequency $rel\_freq(i)$ (in integer) for each disk i;
  Split each disk into a number of smaller, equal-size chunks:
    Calculate max_chunks as the LCM of the relative frequencies;
    Split each disk i into $num\_chunk(i) = max\_chunks/rel\_freq(i)$ chunks;
    let C_{ij} be the j^{th} chunk in disk i;
  Create the broadcast program by interleaving the chunks of each disk:
    for $i := 0$ to $max\_chunks - 1$
    {
      for $j := 0$ to num_disks
        broadcast chunk $C_{j,(i \bmod num\_chunks(j))}$;
    }
}

Figure 6 illustrates an example in which seven data items are divided into three groups of similar access probabilities and assigned to three separate disks in the broadcast. These three disks are interleaved in a single broadcast cycle. The first disk rotates at a speed twice as fast as the second one and four times as fast as the slowest disk (the third disk). The resulting broadcast cycle consists of four minor cycles.

We can observe that the Bdisk method can be used to construct a finegrained memory hierarchy such that items of higher popularities are broadcastmore frequently by varying the number of the disks, the size, relative spinning speed, and the assigned data items of each disk.

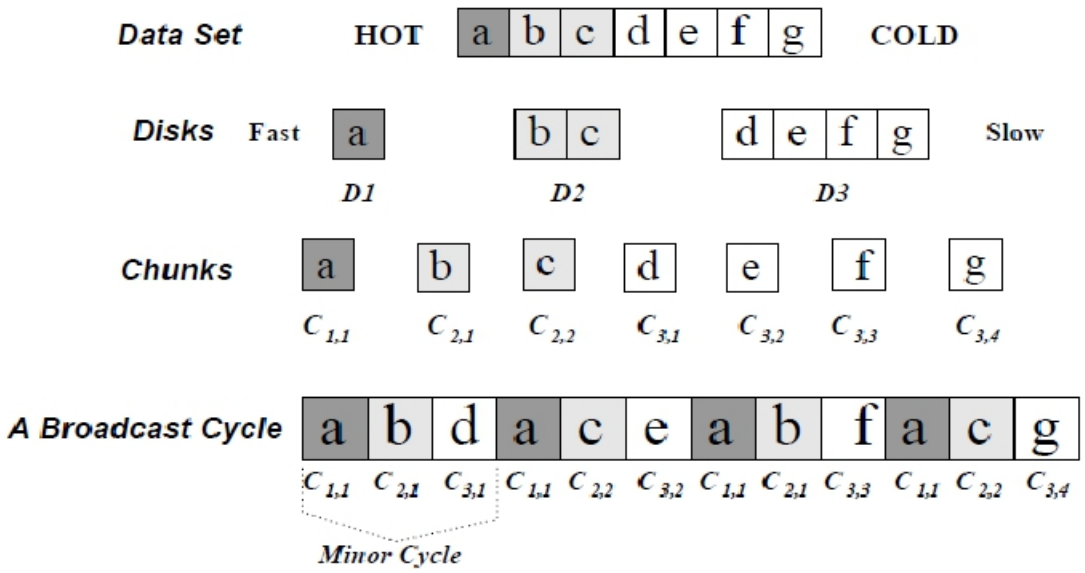
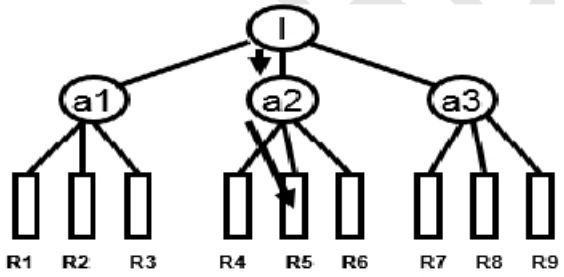


Figure 6: An Example of a seven-item, Three-disk Broadcast Program

12 Indexing on air

- Air indexing is used to save battery power of mobile devices
- The server builds an index tree based on access probabilities of broadcasting data items such that a data item with a high access prob. is at a shallow level
- The server interleaves index information with data items on the broadcast channel
- By searching the index, a mobile client can predict the desired data's arrival time, stay in doze mode and tune in when the requested data arrives
- Tuning time can be limited to the number of index probes + data



An index tree with fanout of 3 holding nodes with equal access probabilities

Index probing scenario to data item R5 for which the number of index probes = 2

13 Data Caching

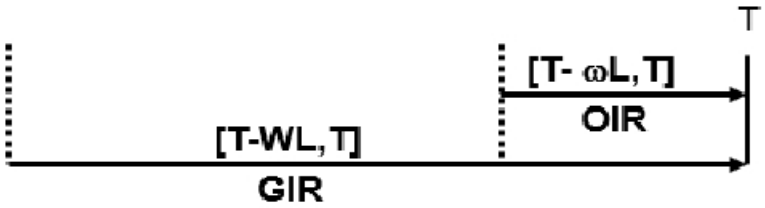
- Data is cached at the mobile client
- When the client receives a query, the mobile first searches its cache
  - If there is a "valid" copy, the mobile client returns an answer
  - If not, the mobile client obtains a copy through on-demand or from broadcast
- Caching is particularly important in mobile environments
  - Save power
  - Improve access latency
  - Improve data availability in case of disconnection
- Cache consistency: the server sends invalidation reports to mobile clients

14 Cache Invalidation Design in Mobile Database Applications

- *Stateful-based*: The server knows the data items cached by the mobile clients and will send invalidation messages to the affected clients when there is any update to the database
  - Not scalable
- *Stateless-based*: the server broadcasts information on objects that are most recently updated and the clients listen and use the reports to invalidate their caches
  - Scalable
  - Asynchronous: once a data item is updated, the server broadcasts immediately – not useful for clients who may be disconnected for a period of time
  - Synchronous: periodic broadcasting of invalidation reports

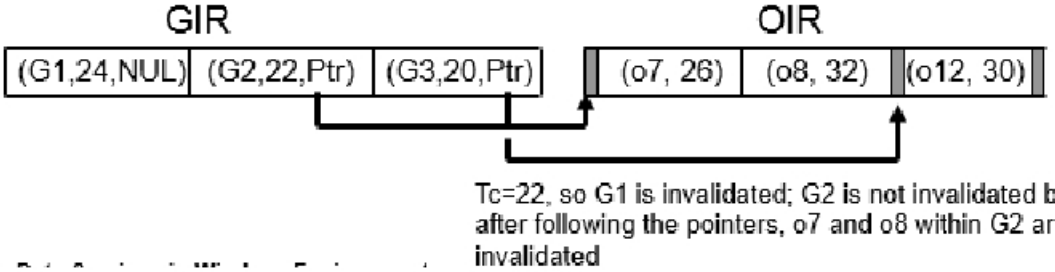
❖ **Dual-Report Strategies for Periodic Broadcasting of Invalidation Reports**

- The server broadcasts in every L time units a pair of invalidation reports, i.e., an *object invalidation report* (OIR) and a *group invalidation report* (GIR):
  - OIR is used to invalidate individual objects
  - The content of OIR is a list (o_{id}, t_{id}), where o_{id} is the object id and t_{id} is the timestamp at which the object was updated during [T- wL,T]
  - GIR is used to invalidate the remaining objects whose group has been updated during [T-WL,T], W > w
  - The content of GIR is a list (G_{id}, T_{id}), where G_{id} is the group identifier and T_{id} is the most recent timestamp at which the group is valid
  - GIR excludes objects already reported in OIR



❖ **"Selective" Dual-Report Strategies for Power Saving**

- Selective Dual-Report is designed to minimize the tuning time
- GIR is broadcast before OIR in the form of (group-id, time-stamp, ptr), where "ptr" is a pointer to point to the starting position of the objects within this group in OIR
- For each group queried, the client first selectively tunes to the GIR
- The timestamp of the last valid report Tc is compared against the timestamp of the group Tid, if Tc < Tid, all objects in that group are invalidated
- For the remaining objects in the query, the mobile client selectively tunes to the respective position in the OIR
- The pair (oid,tid) of the queried object is retrieved and the timestamp Tc is compared against tid, if Tc < tid, the object is invalidated; otherwise the cached object is valid



15 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 and caches 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 a structure 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 (t_{access}) 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 t_{access} – (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).

15.1 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, a directory (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 or records, 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 t_{tune} is the time taken by the device for selection of records. 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 t_{tune} and t_{access}. 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, $t_{tune} = t_{access}$ and the device is not idle during any time interval.

15.2 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 by a 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.
3. Assume that a broadcast cycle pushes the hashing parameters $H(R_i)$ [H and S] and record R_i. The functions H and S help in tuning to the $H(R_i)$ and hence to R_i as follows—H gives a key which in turn gives the location of $H(R_i)$ in the broadcast data. In case H generates a key that does not provide the location of $H(R_i)$ by itself, then the device computes the location from S after the location of $H(R_i)$. That location has the sequential records R_i and the devices tunes to the records from these locations.
4. In case the device misses the record in first cycle, it tunes and caches that in next or some other cycle.

15.3 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. All buckets 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 t_{access}.

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

1. $I_{offset}(1)$ which defines the offset to first bucket of nearest index.
2. Additional information about T_b, 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. I_{offset} (next) which is the index offset of next bucket record of interest.
4. I_{offset}(end) which is the index offset for the end of broadcast cycle and the start of next cycle. This enables the device to look for next index I after the time interval as per I_{offset}(end). This also permits a broadcast cycle to consist of variable number of buckets.
5. I_{type}, which provides the specification of the type of contents of next bucket to be tuned, that is, whether it has an index value or data.
6. A flag called dirty flag which contains the information whether the indexed buckets defined by $I_{offset}(1)$ and I_{offset}(next) are dirty or not. An indexed bucket being dirty means that it has been rewritten at the server with

new values. Therefore, the device should invalidate the previous caches of these buckets and update them by tuning to and caching them.

The advantage of having an index is that a device just reads it and selectively tunes to the data buckets or records of interest instead of reading all the data records and then discarding those which are not required by it. During the time intervals in which data which is not of interest is being broadcast, the device remains in idle or power down mode.

Transmission of an index I only once with every broadcast cycle increases access latency of a record as follows: This is so because if an index is lost during a push due to transmission loss, then the device must wait for the next push of the same index-record pair. The data tuning time now increases by an interval equal to the time required for one broadcast cycle. An index assignment strategy ***(I, m)*** is now described. *(I, m)* indexing means an index I is transmitted *m* times during each push of a record. An algorithm is used to adapt a value of **m** such that it minimizes access (caching) latency in a given wireless environment which may involve frequent or less frequent loss of index or data. Index format is adapted to (I, *m)* with a suitable value of m chosen as per the wireless environment. This decreases the probability of missing I and hence the caching of the record of interest

Indexing reduces the time taken for tuning by the client devices and thus conserves their power resources. Indexing increases access latency because the number of items pushed is more (equals m times index plus *n* records).

15.4 Distributed Index Based Method

Distributed index-based method is an improvement on the (I, *m)* method. In this method, there is no need to repeat the complete index again and again. Instead of replicating the whole index *m* times, each index segment in a bucket describes only the offset I' of data items which immediately follow. Each index I is partitioned into two parts—I' and I''. I'' consists of unrepeated *k* levels (sub-indexes), which do not repeat and I' consists of top I repeated levels (sub-indexes).

Assume that a device misses I(includes I' and I' once) transmitted at the beginning of the broadcast cycle. As I' is repeated *m* - I times after this, it tunes to the pushes by using I', The access latency is reduced as I' has lesser levels.

15.5 Flexible Indexing Method

Assume that a broadcast cycle has number of data segments with each of the segments having a variable set of records. For example, let *n* records, R_o to R_{n-1}, be present in four data segments, $R_{()}$ to R_{i-1}, R_i to R_{j-1} , R_j to R_{j-1} and R_k to R_{n-1}. Some possible index parameters are (i) I_{seg}, having just 2 bits for the offset, to specify the location of a segment in a broadcast cycle, (ii) I_{rec}, having just 6 bits for the offset, to specify the location of a record of interest within a segment of the broadcast cycle, (iii) I_b, having just 4 bits for the offset, to specify the location of a bucket of interest within a record present in one of the segments of the broadcast cycle. Flexible indexing method provides dual use of the parameters (e.g., use of I_{seg} or I_{rec} in an index segment to tune to the record or buckets of interest) or multi-parameter indexing (e.g., use of I_{seg}, I_{rec}, or I_b in an index segment to tune to the bucket of interest).

Assume that broadcast cycle has *m* sets of records (called segments). A set of binary bits defines the index parameter I_{seg}. A local index is then assigned to the specific record (or bucket). Only local index (I_{rec} or I_b) is used in (I_{loc}, m) based data tuning which corresponds to the case of flexible indexing method being discussed. The number of bits in a local index is much smaller than that required when each record is assigned an index. Therefore, the flexible indexing method proves to be beneficial.

15.6 Alternative Methods

Temporal Addressing Temporal addressing is a technique used for pushing in which instead of repeating I several times, a temporal value is repeated before a data record is transmitted. When temporal information contained in this value is used instead of address, there can be effective synchronization of tuning and caching of the record of interest in case of nonuniform time intervals between the successive bits. The device remains idle and starts tuning by synchronizing as per the temporal (time)-information for the pushed record. Temporal information gives the time at which cache is scheduled. Assume that temporal address is 25675 and each address corresponds to wait of 1 ms, the device waits and starts synchronizing the record after 25675 ms.

Broadcast Addressing: Broadcast addressing uses a broadcast address similar to IP or multicast address. Each device or group of devices can be assigned an address. The devices cache the records which have this address as the broadcasting address in a broadcast cycle. This address can be used along with the pushed record. A device uses broadcast address in place of the index I to select the data records or sets. Only the addressed device(s) caches the pushed record and other devices do not select and tune to the record. In place of repeating I several times, the broadcast address can be repeated before a data record is transmitted. The advantage of using this type of addressing is that the server addresses to specific device or specific group of devices.

Use of Headers: A server can broadcast a data in multiple versions or ways. An index or address only specifies where the data is located for the purpose of tuning. It does not specify the details of data at the buckets. An alternative is to place a header or a header with an extension with a data object before broadcasting. Header is used along with the pushed record. The device uses header part in place of the index / and in case device finds from the header that the record is of interest, it selects the object and caches it. The header can be useful, for example it can give information about the type, version, and content modification data or application for which it is targeted.

❖ *Indexing Techniques:*

15.7 (1, m) Index

The (1, m) indexing scheme is an index allocation method where a complete index is broadcast m times during a broadcast. All buckets have an offset to the beginning of the next index segment. The first bucket of each index segment has a tuple containing two fields. The first field contains the key value of the object that was broadcast last and the second field is an offset pointing to the beginning of the next broadcast. This tuple guides clients who missed the required object in the current broadcast so that they can tune to the next broadcast.

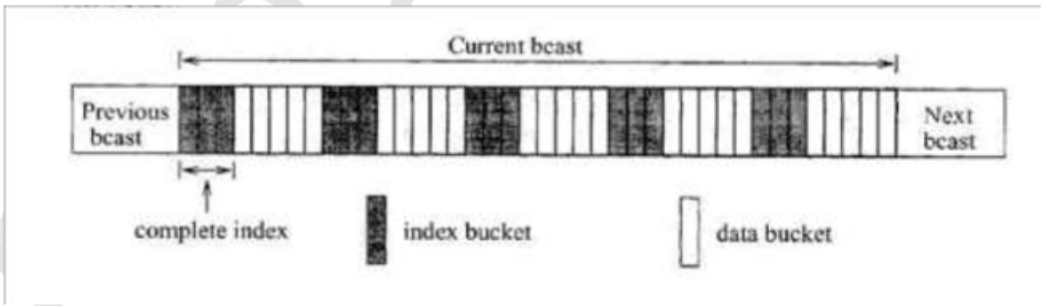


Figure 7: Bcast organization in the (1, m) indexing method

The client's access protocol for retrieving objects with key value k is as follows:

1. Tune into the current bucket on the broadcast channel. Get the offset to the next index segment.
2. Go to the doze mode and tune in at the broadcast of the next index segment.
3. Examine the tuple in the first bucket of the index segment. If the target object has been missed, obtain the offset to the beginning of the next bcast and goto 2; otherwise goto 4.

4. Traverse the index and determine the offset to the target data bucket. This may be accomplished by successive probes, by following the pointers in the multi-level index. The client may doze off between two probes.
5. Tune in when the desired bucket is broadcast, and download it(and subsequent ones as long as their key is k).

Advantage:

1. This scheme has good tuning time.

Disadvantage:

1. The index is entirely replicated m times; this increases the length of the broadcast cycle and hence the average access time.

The optimal m value that gives minimal average access time is (data file size/index size)$^{1/2}$.

There is actually no need to replicate the complete index between successive data blocks. It is sufficient to make available only the portion of index related to the data buckets which follow it. This is the approach adopted in all the subsequent indexing schemes.

15.8 Tree-based Index/Distributed indexing scheme

In this scheme a data file is associated with a B$^+$-tree index structure. Since the broadcast medium is a sequential medium, the data file and index must be flattened so that the data are index are broadcast following a preorder traversal of the tree. The index comprises two portions: the first k levels of the index will be partially replicated in the broadcast, and the remaining levels will not be replicated. The index nodes at the (k+1)th level are called the nonreplicated roots.

Essentially, each index subtree whose root is a non-replicated root will appear once in the whole bcast just in front of the set of data segments it indexes. On the other hand, the nodes at the replicated levels are replicated at the beginning of the first broadcast of each of its children nodes.

To facilitate selective tuning, each node contains meta-data that help in the traversal of the trees. All non-replicated buckets contain pointers that will direct the search to the next copy of its replicated ancestors. On the other hand, all replicated index buckets contain two tuples that can direct the search to continue in the appropriate segments. The first tuple is a pair(x, ptr_{begin}) that indicates that key values less than x have been missed and so search must continue from the beginning of the next bcast(which is ptr_{begin} buckets away). The second pair (y, ptr) indicates that key values greater than or equal to y can be found ptr offset away. Clearly, if the desired object has key value between x and y, the search can continue as in conventional search operation.

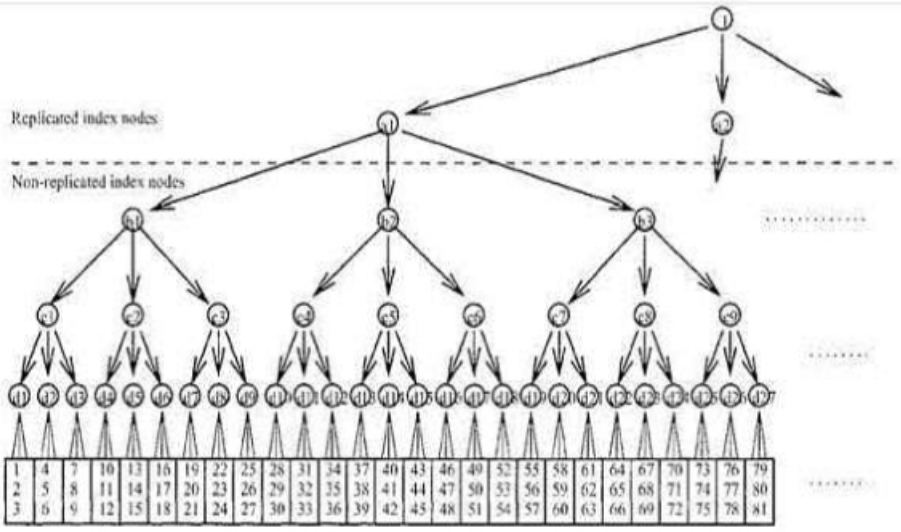


Figure 8: A partial data file and its index tree

The client's access protocol for retrieving objects with key value k is as follows:

1. Tune to the current bucket of the bcast. Get the offset to the next index bucket, and doze off.
2. Tune to the beginning of the designated bucket and examine the meta-data.
   - If the desired object has been missed, doze off till the beginning of the next bcast. Goto 2.
   - If the desired object is not within the data segment covered by the index bucket, doze off to the next higher level index bucket. Goto 3.
   - If the desired object is within the data segment covered by the index bucket, goto 3.

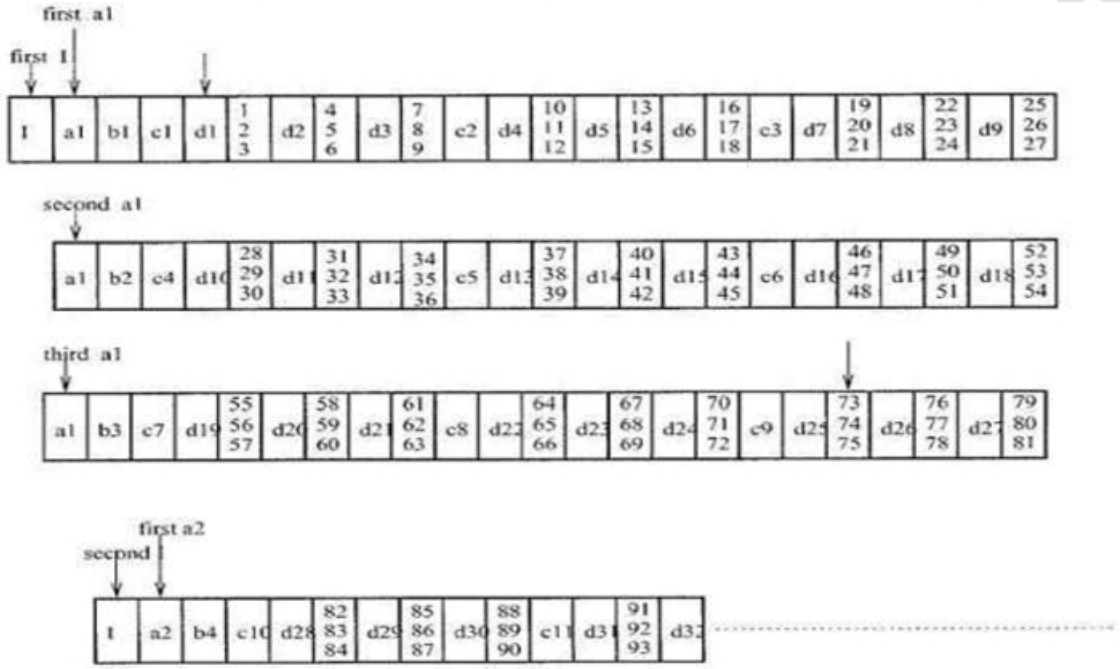


Figure 9: The data broadcast for the distributed indexing scheme with path replication

3. Probe the designated index bucket and follow a sequence of pointers to determine when the data bucket containing the target object will be broadcast. The client may doze off in between two probes.
4. Tune in again when the bucket containing objects with key k is broadcast, and download the bucket (and all subsequent buckets as long as they contain objects with key k).

Advantage:

1. Compared to (1, m) index scheme this scheme has lower access time and its tuning time is also comparable to that of (1, m) index scheme.

15.9 Flexible Indexing Scheme

This scheme splits a sorted list of objects into equal-sized segments, and provides indexes to navigate through the segments. At the beginning of each segment, there is a control index which comprises of two components: a global index and a local index. The global index is used to determine the segment which object may be found, while the local index provides the offset to the portion within the segment where the object may be found.

Suppose the file is organized into p segments. Then the global index at a segment, says, has $[\log_2 i]$ (key, ptr) pairs, where i the number of segments in front of and including segment s, key is an object key, and ptr is an offset. For the first entry, key is the key value of the first data item in segment s and ptr is the offset to the beginning of the next version. Bold examining this pair, the client will know if it has missed the data and if so wait till the next bcast. For the

j^{th} entry (j>1), key is the key value of the first data item in the $([\log_2 i/2^{j-1}] +1)^{th}$ segment following segment s and ptr is the offset to the first data bucket of that segment.

The local index consists of m(key, ptr) pairs that essentially partition each segment further into m+1 sections. For the first entry, key is the key value of the first data item of section m+1 and ptr is the offset to that section. For the jth pair, key is the key value of the first data item of section (m+1-j) and ptr is the offset to the first bucket of that section.

Hence, it is clear that the number of segments and the number of sections per segment can affect the performance of the scheme. Increasing the number of segments or sections will increase the length of the broadcast cycle and reduce the tuning time, and vice versa. Thus, the scheme is flexible in the sense it can be tuned to fit an application's needs.

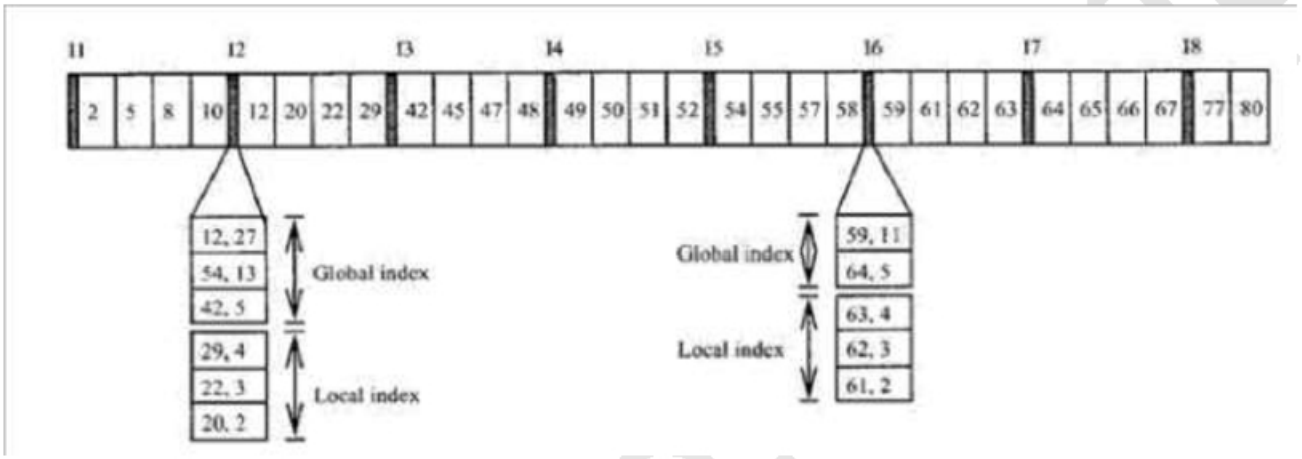


Figure 10: Flexible index scheme.

The client's access protocol for retrieving objects with key value k is as follows:

1. Tune into the channel for a bucket, obtain the offset to the next index segment. Doze off until the next index segment is broadcast.
2. Examine the global index entries. If the target object belongs to another segment, get the offset; doze off for appropriate amount of time and goto 2.
3. Examine the local index entries. Obtain the offset to the section where the target data is stored. Switch to doze mode for appropriate amount of time.
4. Examine objects in the data bucket for the desired object, and download the object.

16 On-demand Data Scheduling

Pushbased wireless data broadcasts are not tailored to a particular user's needs but rather satisfy the needs of the majority. Further, pushbased broadcasts are not scalable to a large database size and react slowly to workload changes. To alleviate these problems, many recent research studies on wireless data dissemination have proposed using on-demand data broadcast. A wireless on-demand broadcast system supports both broadcast and on-demand services through a broadcast channel and a lowbandwidth uplink channel. The uplink channel can be a wired or a wireless link. When a client needs a data item, it sends to the server an on-demand request for the item through the uplink. Client requests are queued up (if necessary) at the server upon arrival. The server repeatedly chooses an item from among the outstanding requests, broadcasts it over the broadcast channel, and removes the associated request(s) from the queue. The clients monitor the broadcast channel and retrieve the item(s) they require. The data scheduling algorithm in on-demand broadcast determines which request to service from its queue of waiting requests at every broadcast instance.

16.1 On-demand Scheduling for Equal-size Items:

Early studies on on-demand scheduling considered only equal-size data items. The average access time performance was used as the optimization objective.

- **First-Come-First-Served (FCFS):** Data items are broadcast in the order of their requests. This scheme is simple, but it has a poor average access performance for skewed data requests.
- **Most Requests First (MRF):** The data item with the largest number of pending requests is broadcast first; ties are broken in an arbitrary manner.
- **MRF Low (MRFL):** MRFL is essentially the same as MRF, but it breaks ties in favor of the item with the lowest request probability.
- **Longest Wait First (LWF):** The data item with the largest total waiting time, i.e., the sum of the time that all pending requests for the item have beenwaiting, is chosen for broadcast.

When the load is light, the average access time is insensitive to the scheduling algorithm used. This is expected because few scheduling decisions are required in this case. As the load increases, MRF yields the best access time performance when request probabilities on the items are equal. When request probabilities follow the Zipf distribution, LWF has the best performance and MRFL is close to LWF. However, LWF is not a practical algorithm for a large system. This is because at each scheduling decision, it needs to recalculate the total accumulated waiting time for every item with pending requests in order to decide which one to broadcast. Thus, MRFL was suggested as a lowoverhead replacement of LWF. However, it was observed that MRFL has a performance as poor as MRF for a large database system. This is because, for large databases, the opportunity for tiebreaking diminishes and thus MRFL degenerates to MRF. Consequently, a lowoverhead and scalable approach called RxW is proposed. The RxW algorithm schedules for the next broadcast the item with the maximal $R \times W$ value, where R is the number of outstanding requests for that item and W is the amount of time that the oldest of those requests has been waiting for. Thus, RxW broadcasts an item either because it is very popular or because there is at least one request that has waited for a long time. The method could be implemented inexpensively by maintaining the outstanding requests in two sorted orders, one ordered by R values and the other ordered by W values. In order to avoid exhaustive search of the service queue, a pruning technique was proposed to find the maximal $R \times W$ value. The performance of the RxW is close to LWF, meaning that it is a good alternative for LWF when scheduling complexity is a major concern. To further improve scheduling overheads, a parameterized algorithm was developed based on RxW. The parameterized RxW algorithm selects the first item it encounters in the searching process whose $R \times W$ value is greater than or equal to $\alpha \times threshold$, where α is is a system parameter and $threshold$ is the running average of the $R \times W$ values of the requests that have been serviced. Varying the α parameter can adjust the performance tradeoff between access time and scheduling overhead. For example, in the extreme case where $\alpha = 0$, this scheme selects the top item either in the R list or in the W list;

this has the least scheduling complexity, but its access time performance may not be very good. With larger α values, the access time performance can be improved, but the scheduling complexity is increased as well.

16.2 On-demand Scheduling for Variable-size Item:

To evaluate the performance for items of different sizes, a new performance metric called stretch was used:

- **Stretch:** the ratio of the access time of a request to its service time, where the service time is the time needed to complete the request if it were the only job in the system.

Compared with access time, stretch is believed to be a more reasonable metric for items of variable sizes since it takes into consideration the size (i.e., service time) of a requested data item. Based on the stretch metric, four different algorithms have been investigated. All of the four algorithms considered are preemptive in the sense that the scheduling decision is reevaluated after broadcasting any page of a data item (it is assumed that a data item consists of one or more pages that have a fixed size and are broadcast together in a single data transmission).

- **Preemptive Longest Wait First (PLWF):** This is the preemptive version of the LWF algorithm. The LWF criterion is applied to select the subsequent data item to be broadcast.
- **Shortest Remaining Time First (SRTF):** The data item with the shortest remaining time is selected.
- **Longest Total Stretch First (LTSF):** The data item which has the largest total current stretch is chosen for broadcast. Here, the current stretch of a pending request is the ratio of the time the request has been in the system thus far to its service time.
- **MAX algorithm:** A deadline is assigned to each arriving request, and it schedules for the next broadcast the item with the earliest deadline. In computing the deadline for a request, the following formula is used:

$$deadline = arrival\_time + service\_time \times S_{MAX},$$

Where S_{MAX} is the maximum stretch value of the individual requests for the last satisfied requests in a history window. To reduce computational complexity, once a deadline is set for a request, this value does not change even if S_{MAX} is updated before the request is serviced.

The MAX scheme, with a simple implementation, performs quite well in both the worst and average cases in access time and stretch measures.

16.3 Energy-efficient Scheduling:

This algorithms broadcast the requested data items in batches, using an existing indexing technique to index the data items in the current broadcast cycle. This way, a mobile client may tune into a small portion of the broadcast instead of monitoring the broadcast channel until the desired data arrives. Thus, this method is energy efficient. The data scheduling is based on a priority formula:

$$Priority = IF^{ASP} \times PF,$$

Where IF (Ignore Factor) denotes the number of times that the particular item has not been included in a broadcast cycle, PF (Popularity Factor) is the number of requests for this item, and ASP (Adaptive Scaling Factor) is a factor that weights the significance of IF and PF. Two sets of broadcast protocols, namely Constant Broadcast Size (CBS) and Variable Broadcast Size (VBS). The CBS strategy broadcasts data items in decreasing order of the priority values until the fixed broadcast size is exhausted. The VBS strategy broadcasts all data items with positive priority values. Simulation results show that the VBS protocol outperforms the CBS protocol at light loads, while at heavy loads the CBS protocol predominates