Data Dissemination over Wireless Internet

Wireless data dissemination

- **Application distribution service**
  - Information feeds:
    - Stock quotes and sport tickets, electronic newsletters, mailing lists, traffic and weather information systems, cable TV on the Internet.
  - Commercial products:
    - AirMedia's Live Internet broadcast network,
    - Hughes Network Systems' DirectPC

Wireless data dissemination

- **Abstraction model**
  - Server
  - Base station
  - Clients
  - Wireless channels
    - Uplink channels for sending data query requests
    - Downlink channel to receive data from BS

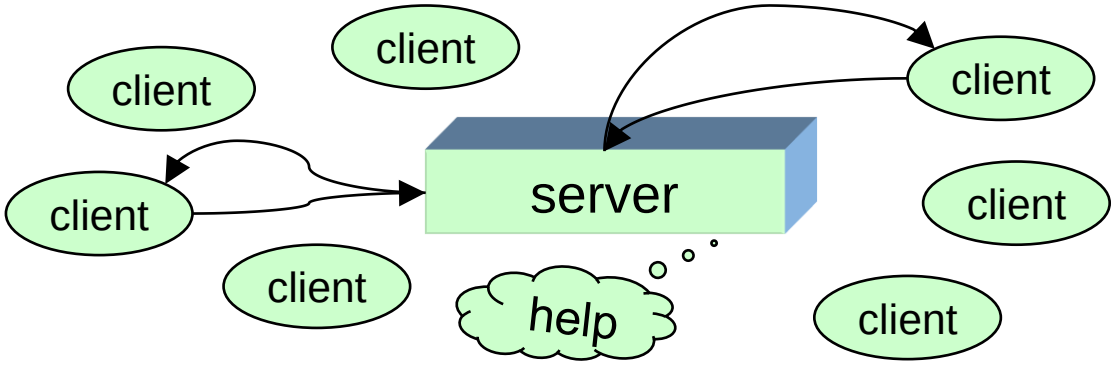
Wireless data dissemination

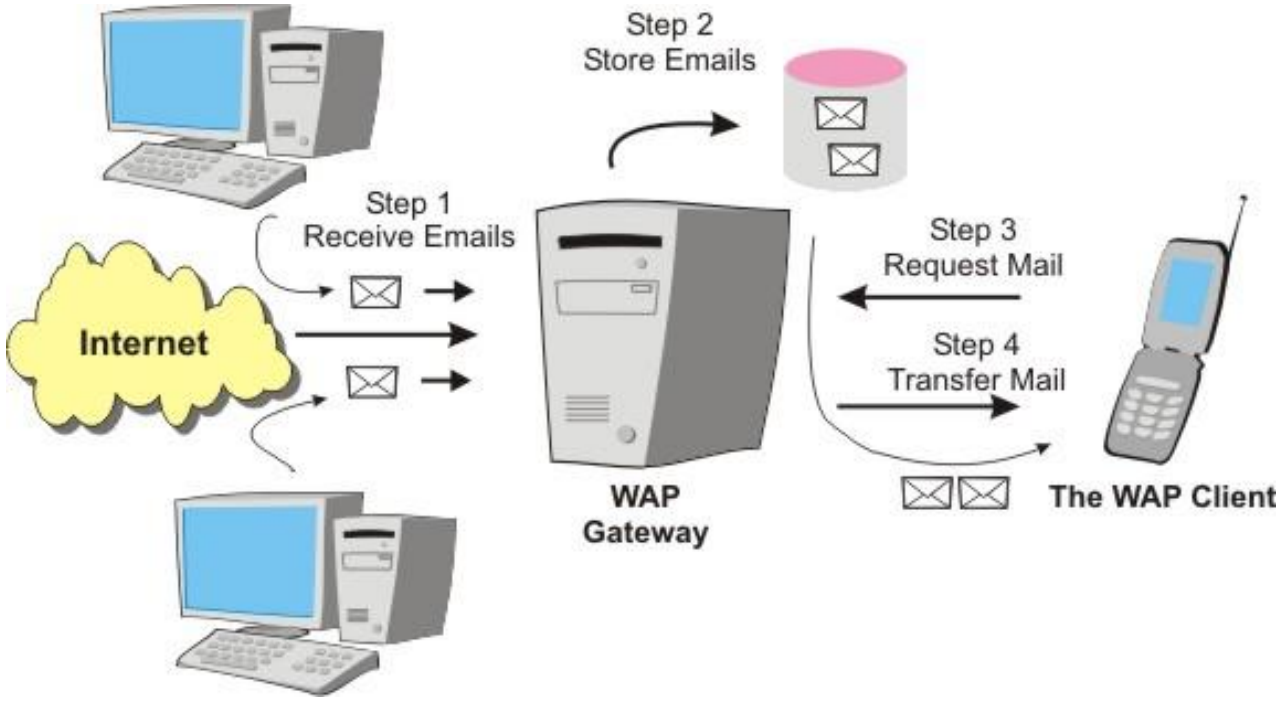
- **Goals:**
  - Reduce access delay
    - Wireless networks have less bandwidth and long delay
  - Minimizing energy consumption at client
    - MUs have limited energy power
  - Maximize capacity of servers

Wireless data dissemination

- **Two basic approaches for data dissemination**
  - Pull-based mechanisms (on-demand mode) - Client actively pulls data from the server
    - music album server, ring tones server, video clips server, or bank account activity server
  - Push-based mechanisms (publish-subscribe mode) - Server actively pushes data to client - as per the subscription for a push service by a client
    - advertisers or generators of traffic congestion, weather reports, stock quotes, and news reports
- **Hybrid mechanisms (hybrid mode)**

Pull-based mechanisms





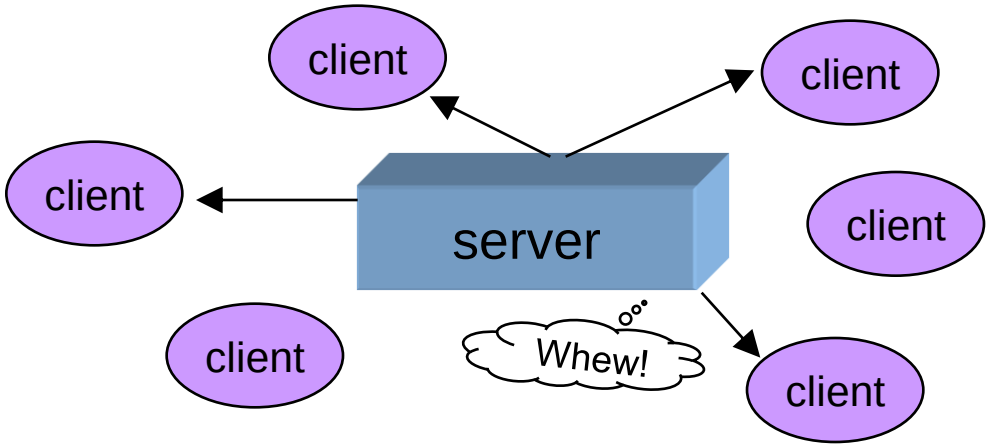
Pull-based mechanisms

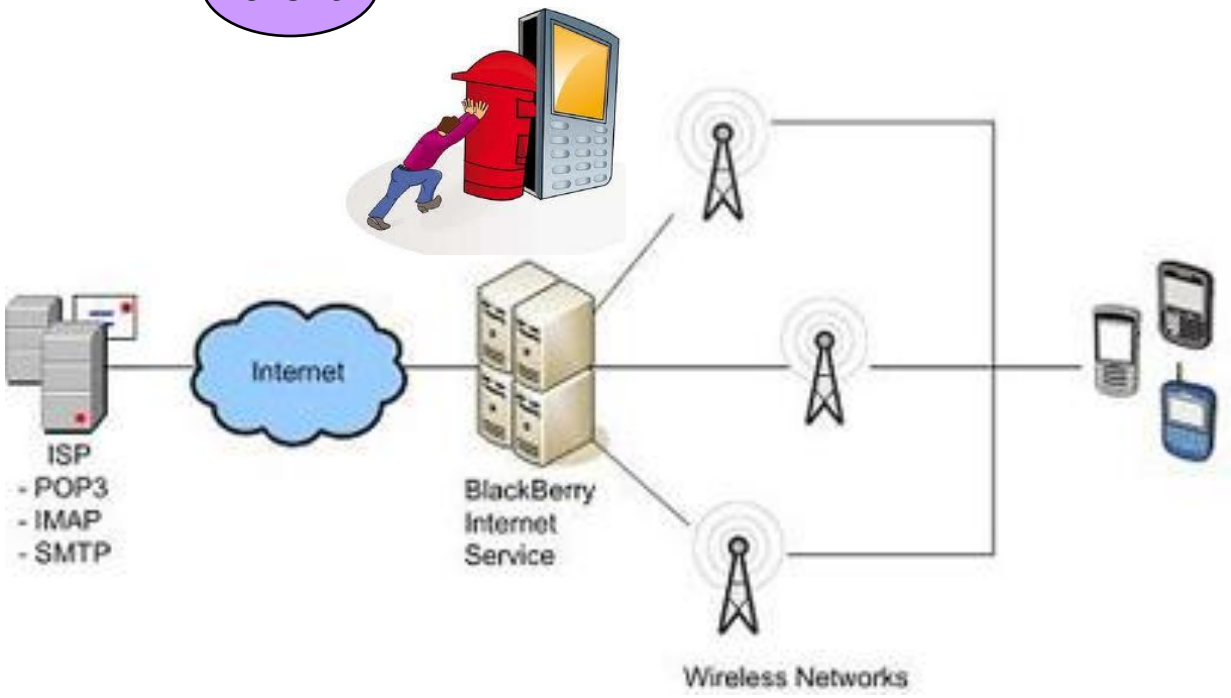
- **Pull is the best option when the server has very little contention and is able to respond to many device requests within expected time intervals**
- **No unsolicited or irrelevant data arrives at the device**
  - Relevant data disseminated only when user asks for it

Pulling bandwidth

- **Bandwidth used for the uplink channel depends upon the number of pull requests**
  - Assume uplink bandwidth of 19.2 kbps and server accepts 384 kbps, only 20 pull requests can be used at 19.2 kbps
  - If number of pull requests is larger, uplink channel bandwidth lowered to 9.8 kbps or 4.8 kbps
  - Similarly, server adapts to the bandwidth required for serving requests (downlink) in case the server is unable to deliver response in a reasonable period
- **Pull threshold limits the number of pull requests in a given period of time**
  - Controls the number of server interruptions

Push-based mechanisms





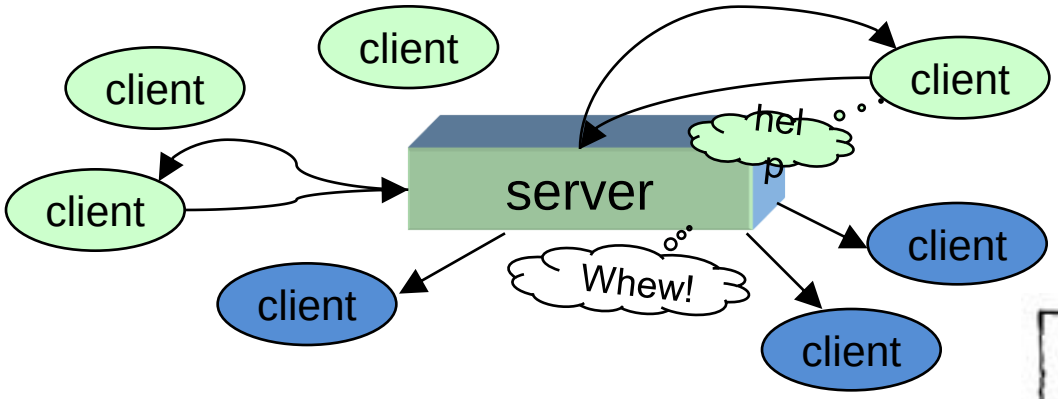
Push-based mechanisms

- **Push is suitable when information is transmitted to a large number of clients with overlapping interests.**
  - The server saves several bandwidth and is prevented from being overwhelmed by client requests.
  - Push is scalable: performance does not depend on the number of clients. Pull cannot scale beyond the capacity of the server or the network.
- **In push, access is *sequential*, so access latency degrades with the volume of data.**

Push algorithms

- **Select a structure of data records to be pushed**
  - an adaptable mechanism that permits data items to be pushed uniformly or non-uniformly after structuring them according to their relative importance
- **Push the data at selected time intervals**
  - pushing at periodic intervals provides the devices that were disconnected at the time of previous push with a chance to get the data when it is pushed again
- **Allocate bandwidths for downlink (for pushes)**
  - Usually higher bandwidth is allocated to data records with higher number of subscribers or higher access probabilities
- **Stop pushes when device is handed over to another BS**

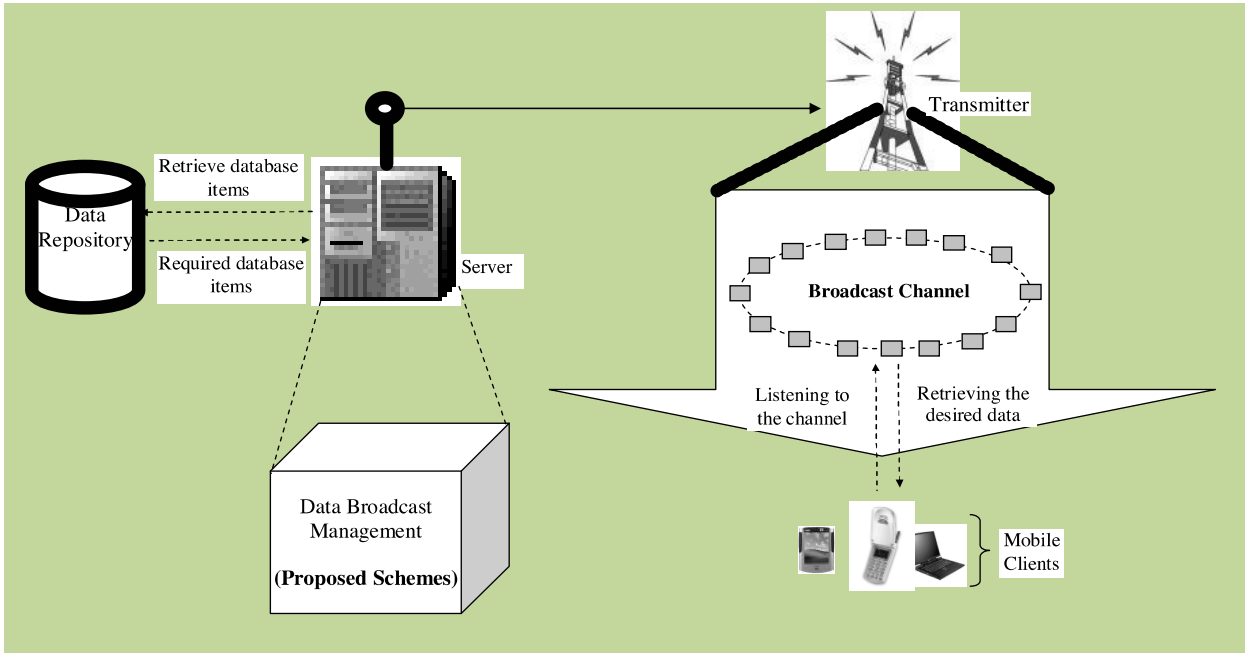
Hybrid mechanisms



Hybrid mechanisms

- **Two channels between devices and server**
  - *Front channel* for push and *Back channel* for pull
  - Bandwidth is shared and adapted between the two channels
    - Bandwidth adapted in downlink and uplink channels depend upon the number of active devices receiving data from the server and the number of devices requesting data pulls from the server

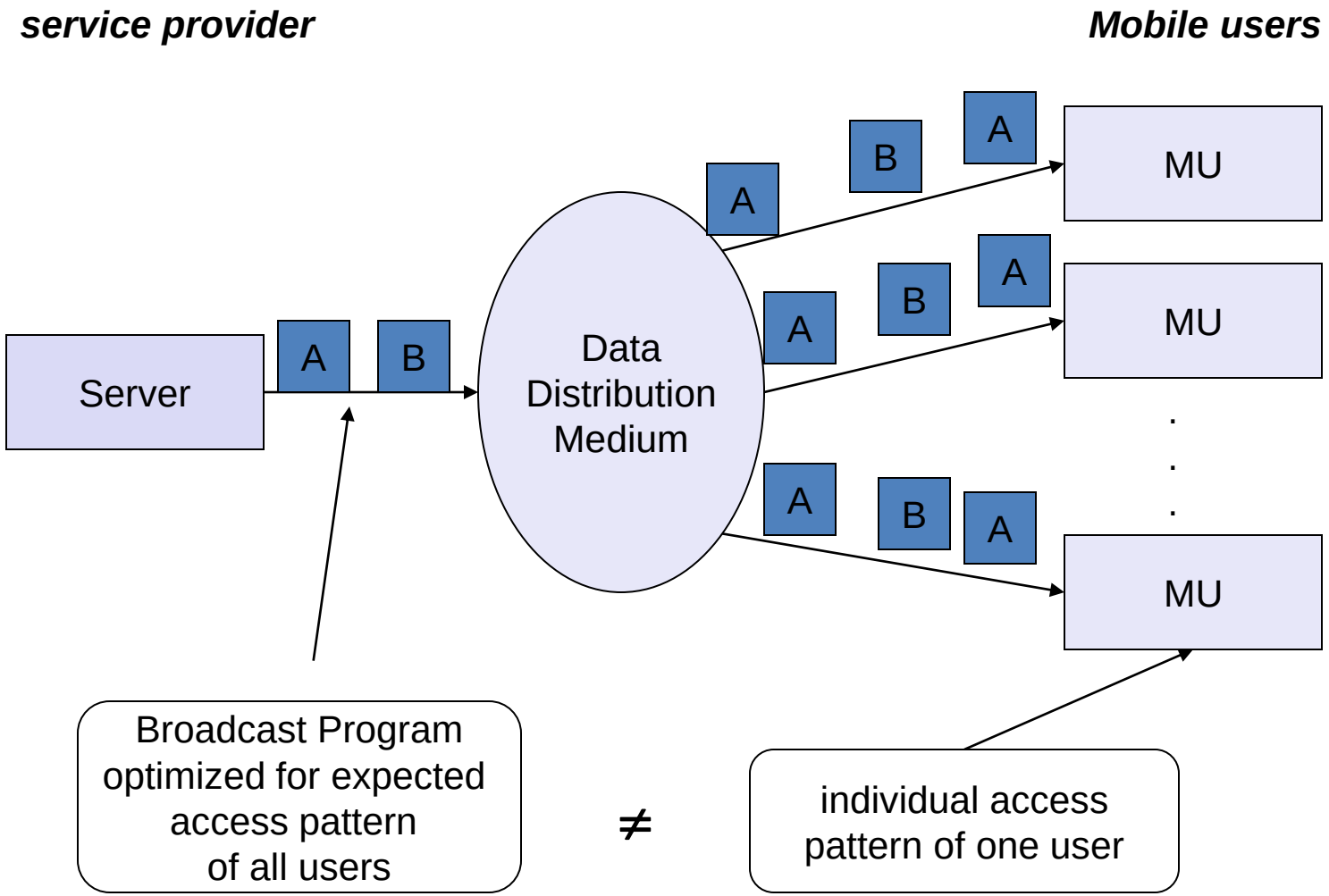
Wireless data broadcast



Wireless data broadcast

- **Server continuously broadcast a set of most frequently accessed data on some fixed radio frequency channel.**
  - Mobile users receive broadcast data packets from the server by *tuning* their mobile units to this broadcast frequency, listen and download the desired data to local cache.
  - Server determines, according to users' demands or their interests, what data should be broadcast and its schedule on the channel.
- **A *broadcast program* consists of a set of data items, ordered in time of broadcast**
  - The program can also use *index* or some other method to inform users when their interested data are available, so as to save battery consumption on mobile units.

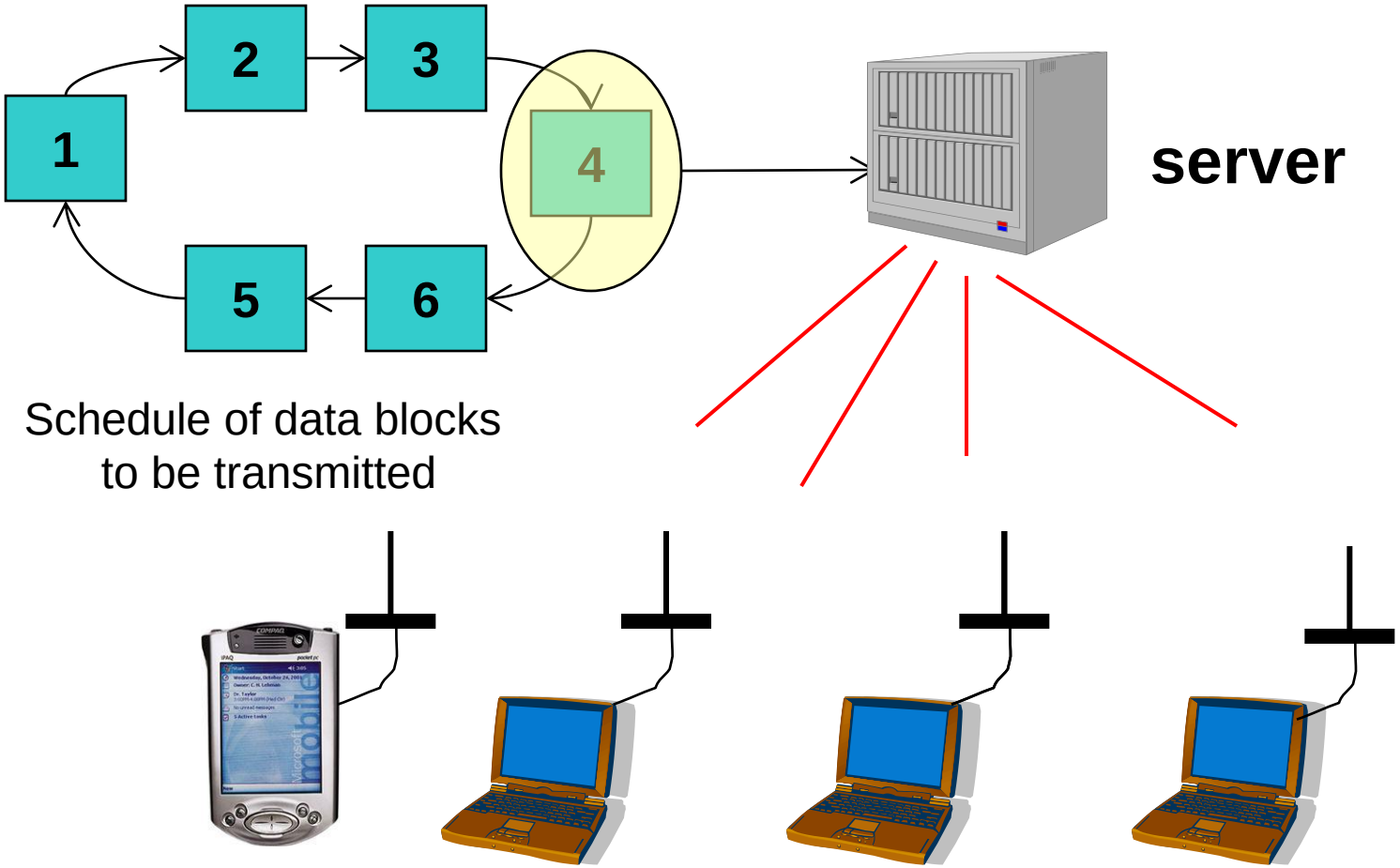
Wireless data broadcast



Pure vs. On-demand broadcast

- **Pure broadcast**
  - MUs do not send any query but listen passively to the channel for interesting data items – e.g., stock quote
- **On-demand broadcast**
  - MUs send queries, and the result to a query is broadcast together with results from other queries on the same channel.
  - MU must filter out the designated result.

Models for wireless broadcast



Models for wireless broadcast

- Assume that records to be broadcast are stored on a hypothetical circular disk - the disk revolves and the angle changes from 0° to 360°
- The entire *N* bits in *n* records get pushed through a hypothetical reading-head over the disk.
- The head continuously reads each bit of a record just beneath it and broadcasts it instantaneously on the wireless network
- During next revolution each bit in the records positioned from 0° to 360° broadcast once again in the same sequence as in earlier revolution
- In case a device misses a record in first revolution, it can cache the same in next or any of the successive revolutions

Models for wireless broadcast

- **T_s = time taken for one revolution of the disk**
  - Time interval between successive broadcasts of each bit in each record
  - $Ts = N \times ts$
- **t_s = time interval between successive bits transmitted**
  - Time interval between transmission of successive bits
  - $ts = (Ts/N)$ [= reciprocal of bandwidth]
- **Due to sequential access of broadcast data items, increasing number of broadcast items causes mobile users to wait for longer time before receiving desired data items.**

Access Time

- **The main purpose of broadcast is to push records of greater interest with greater frequency in order to reduce access time or average access latency**
- **t_{access} = the time interval between request from device and reception of interested data item from broadcasting or data pushing or responding system**
- **t_{access} is dependent on the number (n) and size (N) of the records to be broadcast**
  - the greater the n and N, the greater will be t_{access}

Tuning Time

- **MU should be activated for listening and caching only when it is going to receive the elected data records or buckets of interest**
  - MU does not have sufficient energy to continuously cache the broadcast records and hoard them in memory
  - MU dissipates more power if it gets each pushed item and caches it
- **Tuning means that MU gets ready for caching at those instants and intervals when a desired record of interest is broadcast**
  - t_{tune} = time MU spent on listening actively to the channel
  - Since MU must scan the channel for interested data item, its power consumption can be considered as directly proportional to t_{tune}

Access Time & Tuning Time

- Do they have any correlation?

Broadcast disks

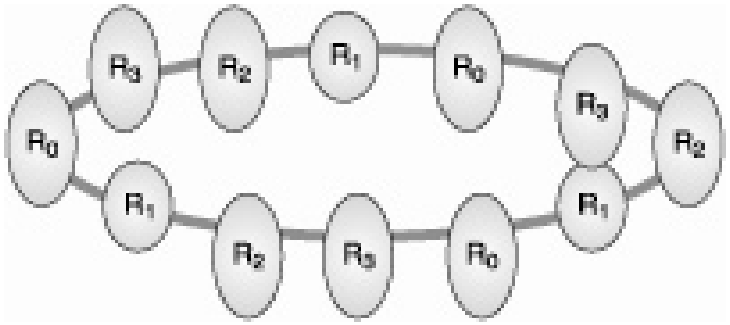
- **Recall that broadcast-disk programs consists of a set of data items on the disk, ordered in time of broadcast.**
- **Item-based broadcast program is expensive to generate, especially when there are many items**
- **For better performance we can collect items of similar access probability into partitions.**
  - Each partition is viewed as an individual disk.
- **Broadcast disks - Put items into different disks with different spinning speed.**
  - Broadcasting hot partitions more often translates into a disk spinning with faster speed.
  - Multiplex multiple disks onto the same broadcast channel.

Broadcast disk models

- **A broadcast file is similar to a disk file but located on the air – so it is also called *broadcast disk - file on the air***
- **Several disk models**
  - *Flat Disk Model*
  - *Circular Multi-disk Model*
  - *Multi-disk Model with repetition rate proportional to priority*
  - *Skewed disk model*

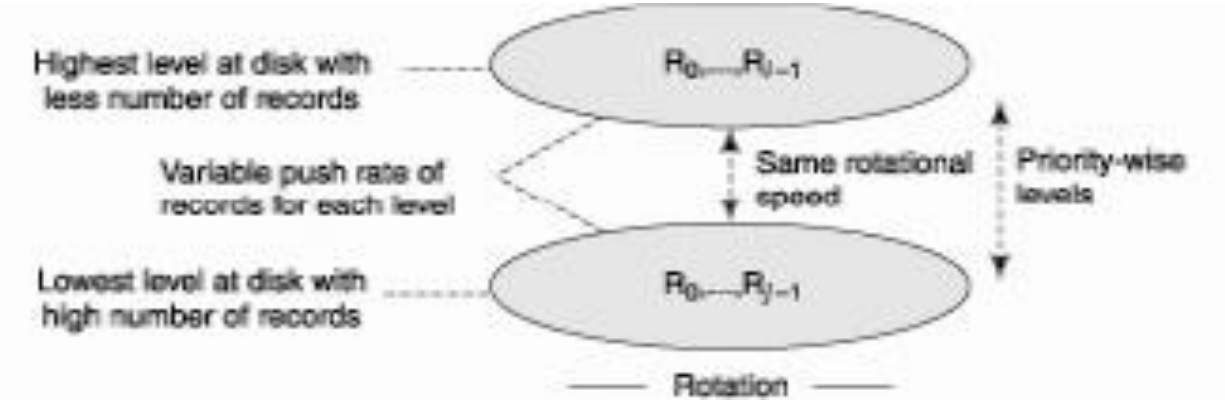
Broadcast disk models

- **Flat Disk Model**
  - Data records, *R0, R1, R2, R3, R0, R1, R2, R3,* are transmitted in a single broadcast cycle
  - All record blocks have an equal priority level (round-robin)
  - Server broadcasts the data as per cyclic requests (subscriptions) without taking into account the number of devices that subscribe to a particular record



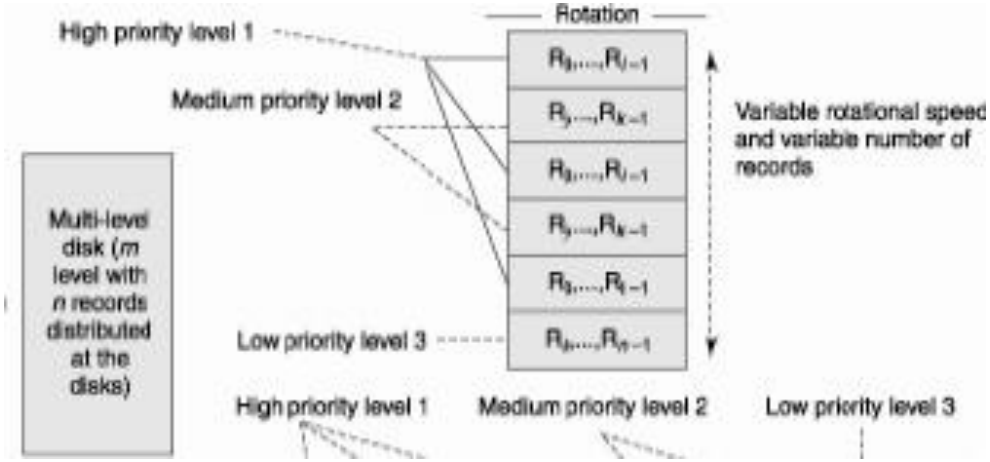
Broadcast disk models

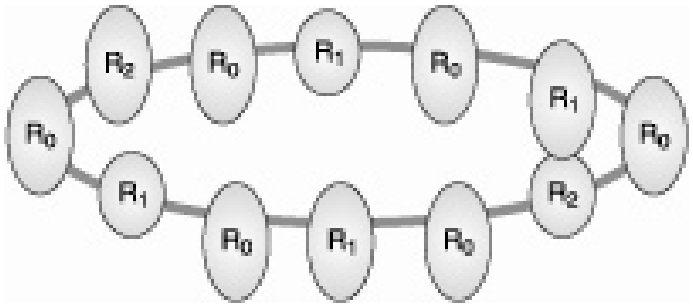
- **Circular Multi-disk Model**
  - Each block of records is pushed with a repetition rate proportional to its hierarchical level



Broadcast disk models

- **Multi-disk Model with repetition rate proportional to priority**
  - The transmission rate at each level is same but the repetition rate of a record is proportional to the record's priority level
    - Popular records are assigned to multiple levels.
    - Priority of a record can be as per the number of users subscribing to it





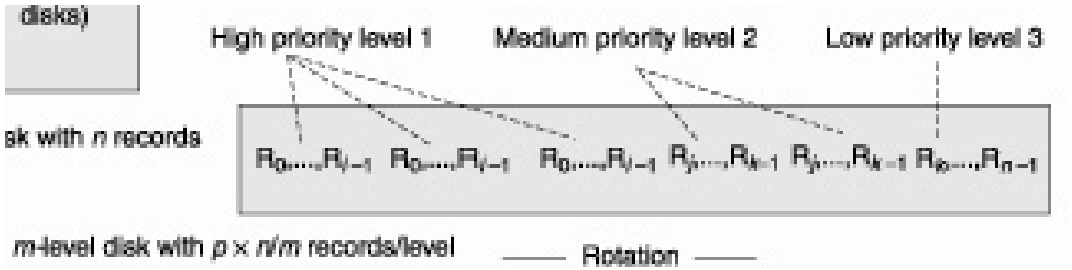
3 priority levels

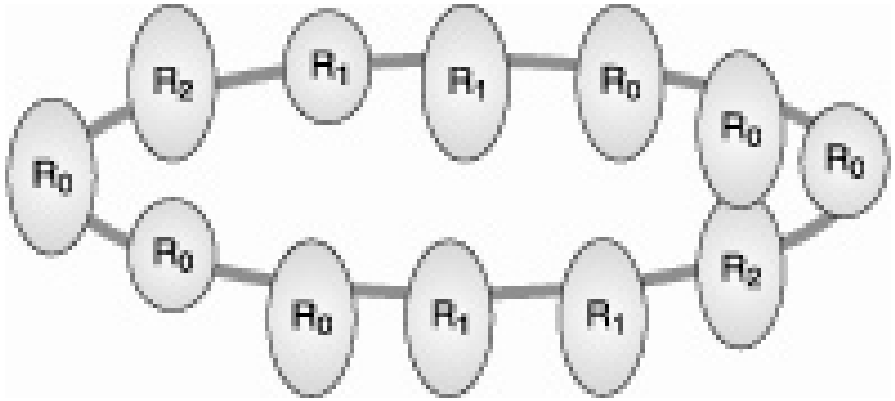
Broadcast disk models

- **Skewed disk model**
  - Block of records are repeated as per their priorities for pushing
    - Priority of a record can be set as per number of its subscribers
  - Unlike the multi-disk model, the skewed-disk model entails consecutive repeated transmissions of a record block, followed by consecutive repeated transmissions of another record block, and so on

Broadcast disk models

- Skewed disk model





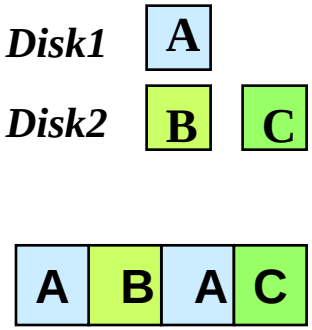
High priority record blocks pushed more often than the low priority ones because these are repeated one after one more often

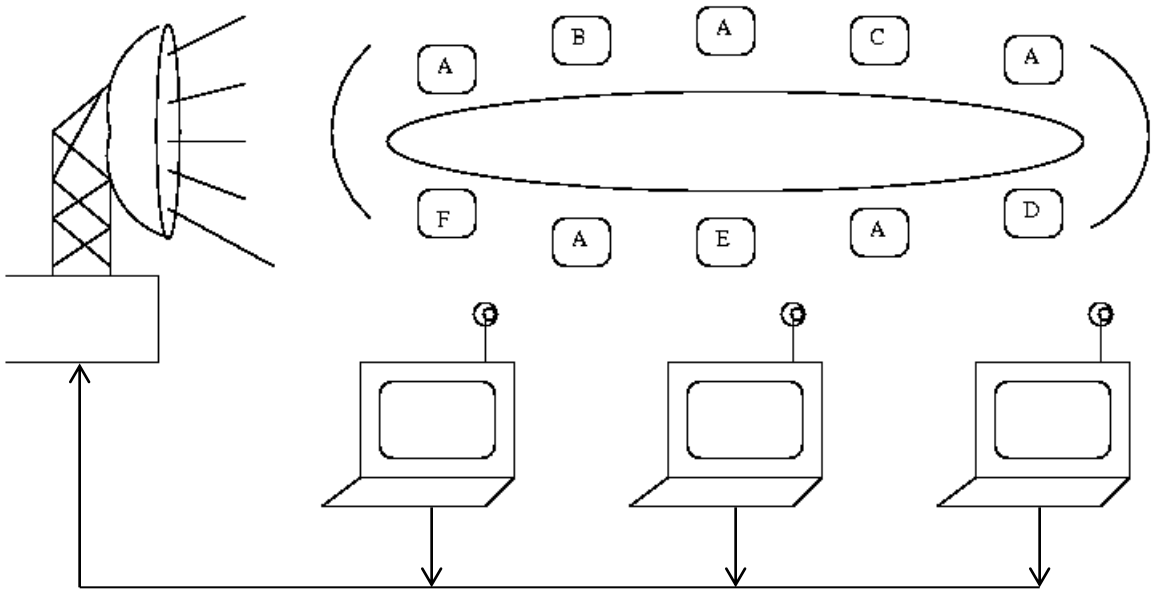
Objectives of broadcast disk program

- Allow mobile users to access broadcast data with optimal access time and power consumption.
- Determine how to organize the data items (e.g., in priority order) and schedule the time of broadcast (e.g., how often).
- Determine how MUs should retrieve and process the broadcast data items on air.

Broadcast disk program

- Schedule of periodic broadcast of one or more disks
  - Disks are of different sizes and can be broadcast at different speed
  - Broadcast frequency of each item depends on its access pattern





Broadcast disk program

- **Disk parameters:**
  - first, the number of disks (i.e., different frequencies);
  - then, for each disk, the number of data items to be stored and the relative frequency of broadcast
- **Scheduling Problem:**
  - Given a list of data items ordered by their expected access probabilities and a specification of the disk parameters, design an algorithm that assigns data items to disks and determines the interleaving of disks.
  - The algorithm produces a periodic broadcast program with fixed inter-arrival times per data item.

Broadcast disk program

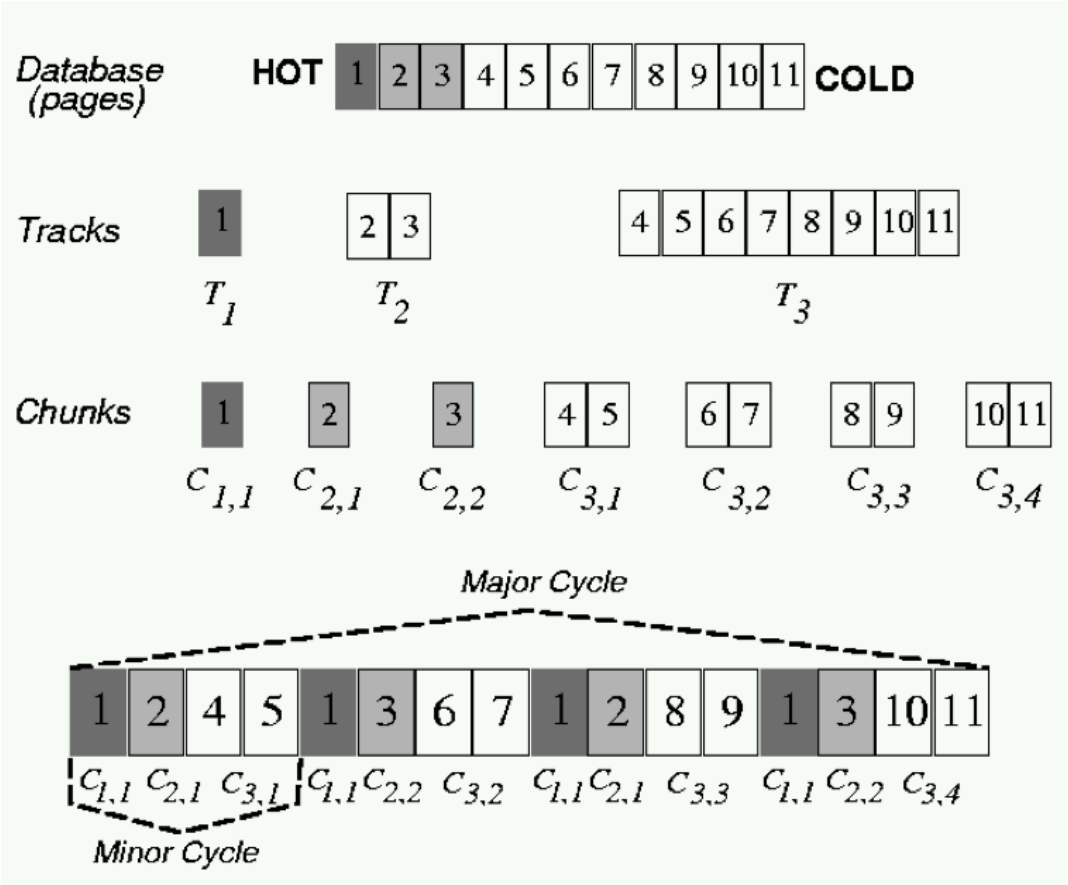
- **An example:**

  - Order pages from "hottest" to coldest (page is record of data with same format and length)
  - Partition pages into tracks ("disks") — pages in a track have similar access probabilities
  - Determine relative frequencies of the disks (e.g., rel_freq (1) = 3, rel_freq (2) = 2)
  - Split each disk into "chunks"
    - maxchunks = LCM(relative frequencies)
    - numchunks(J) = maxchunks / relativefreq(J)
  - Broadcast program is then:

```
for I = 0 to maxchunks -  1
      for J = 1 to numdisks
        Broadcast( C(J, I mod numchunks(J) )
```

Broadcast disk program

- An example:



rel_freq (1) = 4

rel_freq (2) = 2

rel_freq (2) = 1

Selective tuning

- **Clients may actively listen to thousands of data records on the broadcast just to pick up one of them to use.**
- **Selective tuning is the process by which MU selects only the required data records, tunes to them, and caches them**
  - MU activates only when the data arrives, saving power consumption
- **How to enable selective tuning?**
  - Place a structure and overhead over the broadcast data
  - In addition to data, each broadcast cycle broadcasts a *directory* or *index* which is the overhead prefixed by server before the data
  - A disadvantage of selective tuning is that it extends the broadcast cycle and hence increases t_{access}

Selective tuning

An example:

1. n records R0 to Rn–1 interleaved and broadcast as in a multi-disk model

2.Only the records *Ri' and Rj' are of* interest and required by applications on a device

3.The broadcast disk broadcasts *Ri ' and Rj ' thrice and once, respectively, as the* subscription probability of *Ri ' is three* times that of *Rj* '

4.The record *Ri' is partitioned into k* buckets, *bi0 to bik–1*

5.The record *Rj' is partitioned into k*' buckets, *bj0 to bj k'–1. Each bucket has* equal length *lb, which means equal* number of bits and the devices takes identical time *lb × ts to cache each bucket* data. [*ts the time interval between* successive bits]

6.In addition to data, each broadcast cycle broadcasts a directory, or index which is the overhead prefixed by server before the data

7.Device selects only the buckets of *Ri'* and *Rj ' which are of interest and* receives the signals only during first, second, or third instances of *Ri ' or* during instances of *Rj' , that is, during the* intervals Ti0,… ,Ti k–1, Tj0, … ,Tjk*'–1 of* broadcasting of *bi0 … bi k–1, bj0 … bj k'–1,* respectively

8.In the remaining intervals, where either the other records which are not of interest are being broadcast or when record of interest is already cached in an earlier broadcast cycle, the device remains idle.

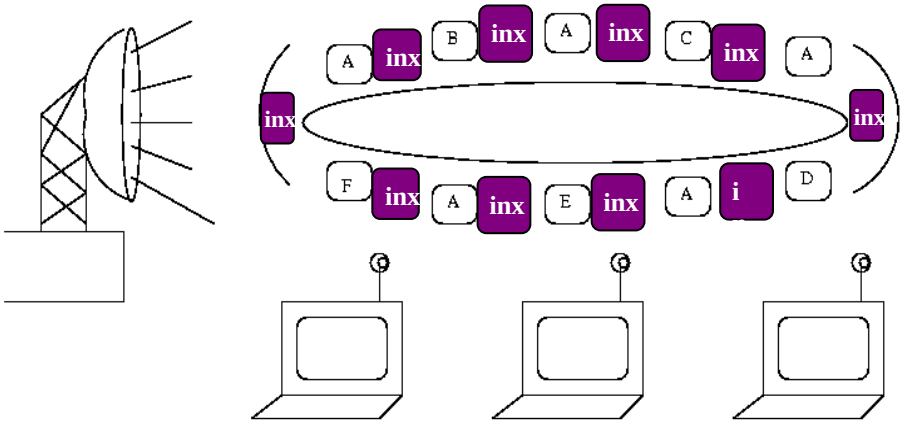
9.During this period it does not dissipate power and hence saves energy

Directory method

- **Broadcasting a directory as overhead at the beginning of each broadcast cycle**
  - A directory specifies when a specific record or data item appears in data being broadcasted, consisting of ***start sign*****,** ***pointers*** **to records,** and ***end sign***
  - If interval between the start of the broadcast cycles is T_B, then directory broadcasts at each successive intervals of T_B
- **A device has to wait for directory before it can get tuned to its interested data and then start caching it**
  - Recall that tuning time t_{tune} is the time taken by the device for selection of desirable records.

Index-based method

- **Indexing is another method for selective tuning, used to guide the MU in the listening process**
  - Index can be broadcast interleavingly with data records.
  - Index only contains key names so is relatively small.
  - A traditional database index maps a key to where the key value is stored in memory; Here, an index for broadcast maps a key to the time when the key value is broadcast "on air".
  - When client is looking for information, it tunes to the channel to locate the index for the broadcast cycle – broadcast time of the data and sleep until it arrive.



Index and offset

- **At each location (data record), an offset value may also be specified**
  - An index maps to the absolute location of from the beginning of a broadcast cycle, while an *offset* maps to the beginning of the next index
  - Offset is used by the MU along with the present location to calculate the wait period for tuning to the next record

(1, m) indexing

- **Broadcast the index every fraction 1/m of the broadcast data items.**
  - An index transmits *m* times during each period of the broadcast.
- **All data items have an offset to the beginning of the next indexed data item.**
  - To access a record, a client tunes into the current data item on the channel, uses the offset to tune in again when the index appears.
  - From the index, it determines the required data item.

(1, m) indexing

- **An algorithm is used to adapt a value of *m* such that it minimizes the access latency in a given wireless environment**
  - Index format is adapted to with a suitable *m* chosen as per the wireless environment, so as to decrease the probability of missing the index
  - If *m* is chosen small then the power dissipated by device is less but the data access latency increases
  - The value of *m* therefore needs to be optimized by employing an algorithm

Distributed indexing

- **Improves over the (1, m) method**
  - instead of replicating the whole index (large overhead), each index segment describes only the offset of data items which immediately follow
- **Each index I is partitioned into two parts I' and I″**
  - I″ consists of unrepeated k levels (subindexes), which do not repeat and I′ consists of top j repeated levels (subindexes)