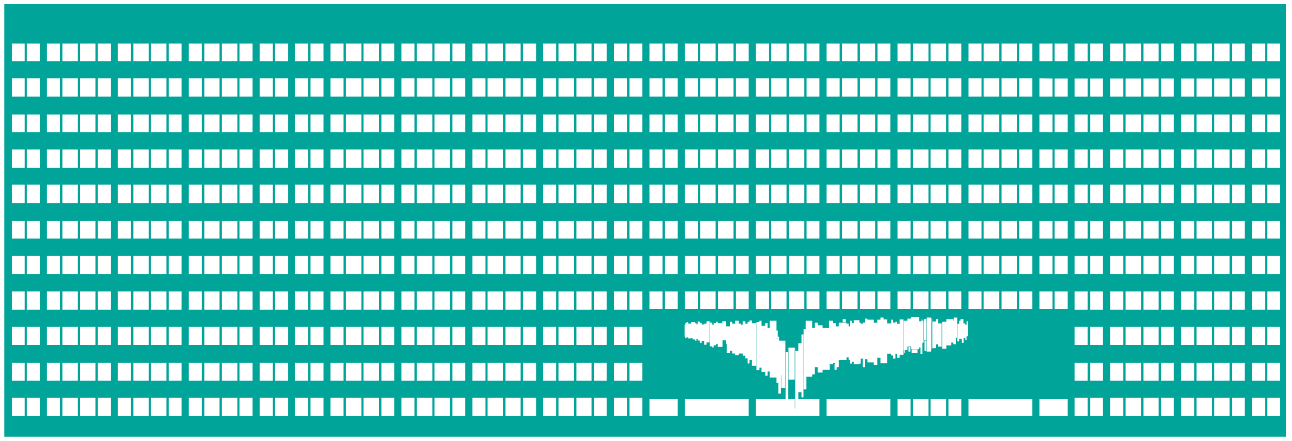
VSB TECHNICAL UNIVERSITY OF OSTRAVA | FACULTY OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE | DEPARTMENT OF COMPUTER SCIENCE

Group data dissemination



MS (Mobile Computing)
Lecture 5

Data delivery

1. Pull-based data delivery (*on demand data delivery*) – client explicitly requests data items from the server.
2. Push-based data delivery – server repetitively broadcasts data to client population without any specific request. Clients monitor the broadcasts and retrieve any data items they need when they arrive.

- Can be physically supported in wireless computing
  - cell broadcast
  - broadcast/multicast traffic in IP networks
- Asymmetric approach – more data flows in one direction

Example: applications

- Dissemination-based: data feeds such as stock quotes, sport tickets, newsletters, mailing lists, weather & traffic information systems
- Cable TV, Teletext and Videotex systems, RDS, EPG
- Special Internet services (commercial):
  - clients at mobile hosts have usually a low bandwidth link while servers may have relatively high bandwidth broadcasting capability (historical solutions).
- Broadcasting example:
  - Boston Community Information System (BCIS) – news and information over an FM channel
- Broadcasting database system over high-speed link, clients filter data based on queries they perform

Pull versus Push delivery

- Push is suitable when information is transmitted to a large number of clients with overlapping requirements:
  - server saves several messages
  - server is not overwhelmed by client requests.
- Push is scalable because performance does not depend on the number of clients; pull cannot scale beyond the capacity of the server or the network.

- Only sequential access is possible in push – access latency degrades with the volume of data.

- In pull, clients play a more active role by requesting data

Hybrid approach

- Clients are provided with an uplink channel – backchannel, to send messages to the server.
- Sharing the channel – the same channel is used for both broadcasting and for the transmission of the replies to on-demand requests
- Use of the backchannel:
  - to provide feedback & profile information to server
  - to directly request data
- To avoid overwhelming of the server, uncached page *p* is requested via backchannel only if the number of pages scheduled for broadcast before $p \geq$ threshold *t*

Selective hybrid broadcasting

- Broadcast an appropriately selected subset of items and provide the rest on-demand, e.g.:
  - air-cache for storing frequently requested data. Broadcasting content continuously adjusts to match the *hot-spot* of the database, calculated by observing broadcast misses (detected from explicit pull requests for data).
  - database partitioned into:
    - broadcast publication group
    - on demand group.
- The criterion for partitioning is to minimize the number of backchannel requests while limiting the response time below a predefined threshold.

On-demand broadcasting

- Server chooses next item to broadcast on every broadcast tick according to the received data requests.
- Various broadcast strategies:
  - pages in the order they are requested (FCFS)
  - page with the maximum number of pending requests.
- A parameterized algorithm exists for large-scale data broadcasting – it is based only on the current queue of pending requests.

Mobility & broadcasting

- Users may move to areas covered by different servers:
  - there are differences in the type of communication infrastructure → the capacity to service requests may drastically differ.
  - the distribution of requests for specific data at each cell changes.
- Example: two variations of an adaptive algorithm for mobility in a cellular architecture. The algorithm statistically selects broadcasted data according to
  - user profiles and
  - registration in each cell.

Broadcast data organization

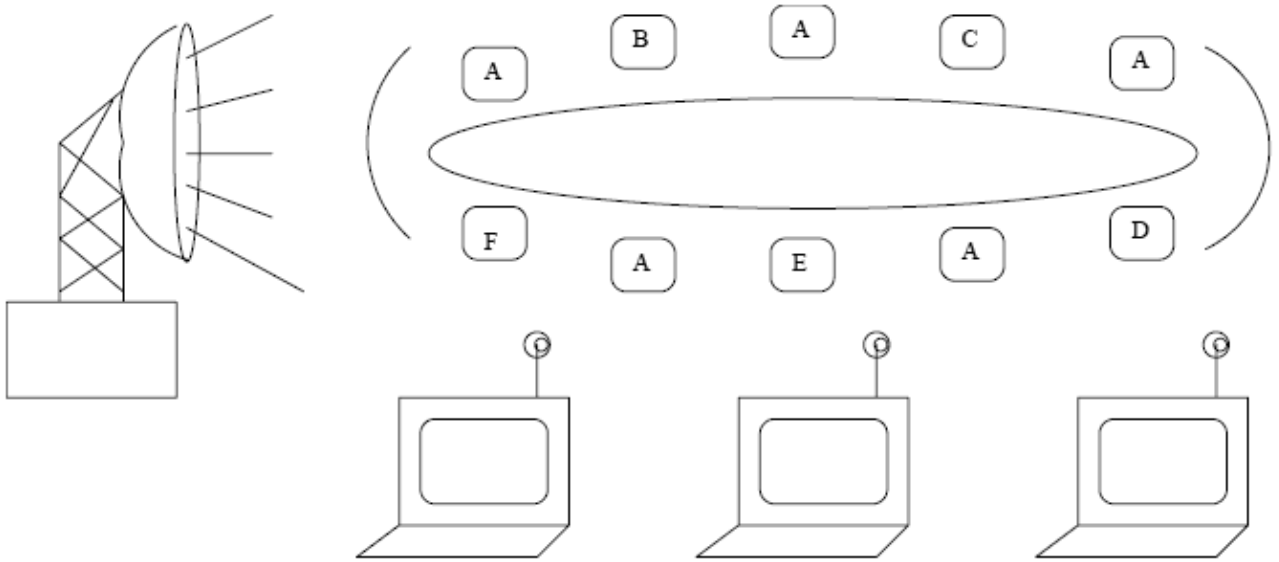
- Organization is necessary for optimal data delivery. Needs to minimize following two time-based criteria:
  - Access/Query time – average time elapsed from the moment client expresses its interest to an item to the reception of said item on the broadcast channel
  - Tuning/Listening time – amount of time spent listening to the broadcast channel
- Common approaches:
  - Flat organization – cyclically, the union of requested data is broadcasted
  - Broadcast disks

Broadcast disks

- Broadcast data items are likely to be interesting to a larger percentage of clients more frequently than other items.
- Data broadcasted with the same frequency are marked as belonging to the same disk → multiple disks of different sizes and speeds superimposed on broadcast medium.
- *Does pose more than the bandwidth allocation problem:*
  - Given access probabilities of all clients, the server allocates the optimal percentage of the broadcast bandwidth to be for each item.
  - Then broadcast program is randomly generated: average interarrival time between two instances of the same item matches clients' needs.
- Not optimal in terms of minimization of expected delay for an item due to interarrival times variance.

Broadcast program example

- Chunks of data from different disks are multiplexed on the same broadcast channel.
- Chunks of each disk are evenly interspersed with each other.
- Chunks of fast disks are repeated more often than the chunks of slow disks



Broadcast disks – remarks

- Parameters:
  - Number of disks (different frequencies)
  - Two parameters for each disk
    - number of items
    - relative frequency of broadcast.
- Algorithm assigning items to disks and determining the interleaving of disks based on list of items ordered by their expected access probabilities and a specification of the disk parameters.
  - Produces a periodic broadcast program with fixed interarrival times per item.

Pyramid broadcasting method

- Similar to the broadcast disk
- Used in Video-On-Demand services for mobile users:
  - the most frequently requested movies are multiplexed on the broadcast network
    → radical improvement of access time and efficient bandwidth utilization.

Indexing on air

- Clients interested in fetching individual data items are identified by some key from the broadcast channel
- A directory indicating when a specific data item appears in the broadcast should be provided → each client needs only selectively tune in the channel to download the required data
  - Broadcast the directory along with data to minimize access and tuning time

Indexing approaches (1)

Broadcast *d* data items, *i* index entries:

0. Flat broadcast (no index) – both access & tuning time: *d/2*
1. *(1, m)* indexing – the whole index is broadcasted at every 1/m of the broadcast data items. Each item has an offset to the beginning of the next index
    - to access a record, client tunes into current item on the channel, retrieves the offset to tune in again for the index. From the index, it determines the required data item. Optimal value of m: $\sqrt{d / i}$. Index *m* times in br. interval
2. Distributed indexing – improves 1. – each index segment describes only data items which immediately follow.

Indexing approaches (2)

3. Flexible indexing – broadcast is divided into *p* data segments. Items of broadcast are assumed to be sorted. Each data segment is preceded by a control index consisting of:
    - binary control index – used to determine the data segment where the key is located by performing bsearch
    - local index – used to locate specific item inside a segment.
4. Hash-Based Techniques – instead of broadcasting a separate directory, broadcast hashing parameters *h* with each data item.
    - If *h* is perfect, then a client requesting K , tunes in, reads *h*, computes h(K) → waits for bucket h(K).
    - In case of collisions, various placements of the overflow buckets with varying tuning and access times are required.

Broadcasting updated data

Changes can appear in

1. Content of broadcast: inclusion of new items and removal of existing ones (e.g. like in hybrid delivery)
2. The organization of the broadcasted data (e.g., different indexing scheme, changing the frequency of a specific item transmission, …)
3. Values of broadcast data, causes many issues:
   - when should be new values propagated
   - will the clients perform updates?
   - consistency – several consistency models, e.g.:
     - latest value (clients read the most recent value of a data item),
     - periodic, …
   - cache invalidation

Caching when broadcasting

- Clients cache data items to reduce the expected delay → they also lessen their dependency on chosen server broadcast priority

- Replacement policies – traditionally, clients cache hottest data mostly to improve the cache hit ratio (constant cost of obtaining a page on cache miss).
  - In broadcast systems, the cost of servicing a miss on a page depends on the time when the requested page will appear on the broadcast.

Cost-based replacement

- Cost-based page replacement – the cost of obtaining a page on a cache miss is taken into account
  - PIX method – replace the cache page having the lowest pix ratio: fraction of its probability of access (P) and its frequency of broadcast (X) → Optimal under certain conditions, but not practical since it requires perfect knowledge of access probabilities and comparison of pix values for all pages.

Prefetching client

- Prefetch pages in anticipation of future accesses. In broadcast, places additional requirements only on the client's local resources. Prefetching instead of page replacement can reduce the cost of a miss.
  - PT prefetching heuristic - uses the PT (**P**robability of access * amount of **T**ime before page appears again) value to decide whether the current broadcast page is more valuable than another one in the cache. Page with lowest PT value is replaced if current PT is greater.
  - *Teletext* approach – control information along with each broadcast page: a linked list of pages most likely to be requested next. After retrieving *p,* client prefetches the *D* most likely referenced pages associated with p (*D* = cache size). Stops when client sends a new request.

Cache invalidation

Server broadcasts invalidation information to its clients

- *asynchronous* broadcast – invalidation report for the item as soon as its value changes × *synchronous* broadcast – invalidation reports periodically → client has to listen for reports first to decide if its cache is valid or not, which adds some latency to query processing.
- *stateful* server – server maintains information about its clients, e.g., the contents of their cache and when it was last validated × *stateless* server
- *invalidation* – which items were updated × *propagation* – values of updated items
- *information* for *individual* items × *aggregate* for item sets

Invalidation strategies (1)

False alarms may happen in some simple approaches

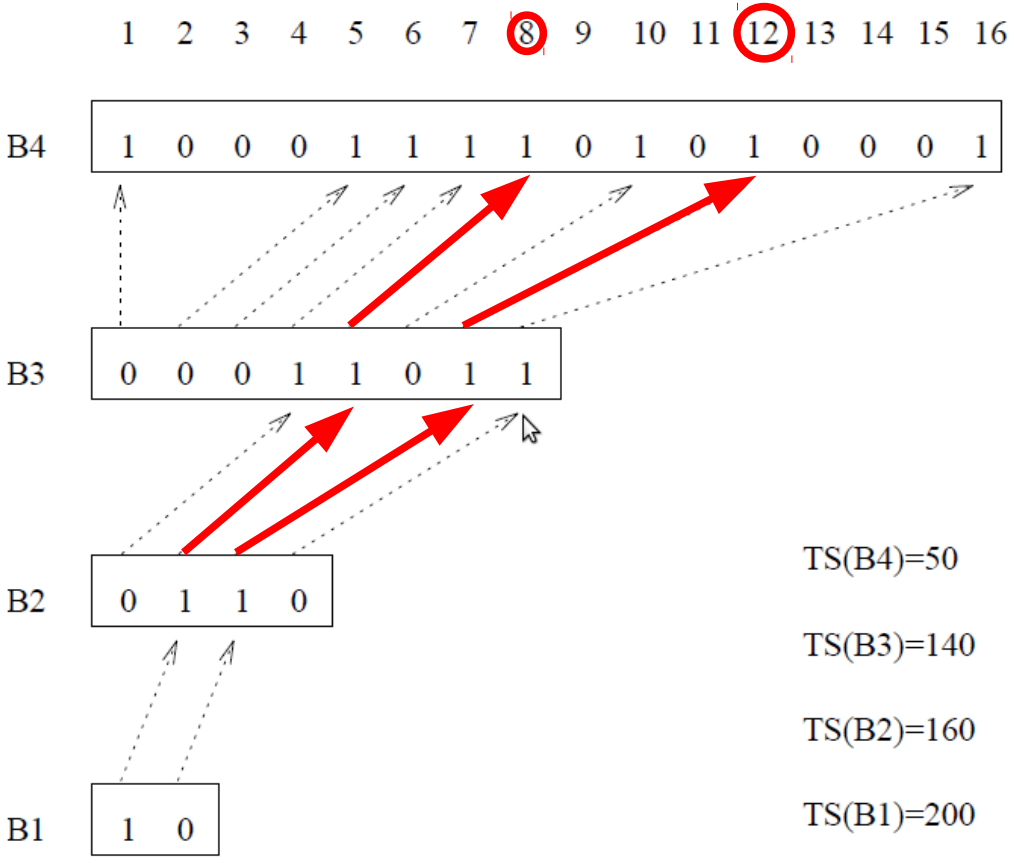
- Synchronous strategies for stateless servers:
  - *Timestamps* strategy (TS) – broadcast timestamps of the latest change for items that have been updated in the last *w* seconds.
  - *Amnesic terminals* strategy (AT) – broadcast only IDs of items changed since the last invalidation report.
  - *Signatures* strategy – broadcast signatures, i.e. checksums computed over the value of several items by applying data compression techniques similar to those used for file comparison.
  - In TS & AT, clients must invalidate entire cache if their disconnection period exceeds a specified length (w–TS)
- Workaholic clients (often connected) × sleepers (disc.)

Invalidation strategies (2)

- Long disconnections: send the identities of all cached objects & their timestamps to the server for validation × send group identifiers and timestamps, and check the validity at the group level
- Asynchronous strategies:
  - *Bit sequences* (BS) – invalidation report: a set of bit sequences with an associated set of timestamps, in a hierarchical structure. Each bit represents a data item. "1" indicates that the corresponding item has been updated since the time specified in associated timestamp.
  - Clients must check the invalidation report before they can use their caches for query processing. They need only to periodically tune in, if the disconnection time precedes the timestamp of the highest sequence, the entire cache in the client will be invalidated

Bit sequences – example

- Client listens to the report at time 250 after having slept for 60 time units. → client disconnected at t =190
  (250-60) ≥ TS(B_2)
  → both the 8th and 12th data items were modified since t=190 and will be invalidated.



Consistency control

- Certification Reports – in optimistic CC, the transaction scheduler checks if the execution that includes transaction is serializable or not at commit time. If it is, transaction is accepted. Server periodically broadcasts to its clients a certification report (CR) that includes the read- and write-set of active transactions that had declared intention to commit during previous period and have been certified.
  - The client uses this information to abort its transactions whose readsets and writesets intersect with the current CR. If a transaction is not aborted, it is sent to the server.
  - The server installs the values in the central database and notifies the clients via broadcast.
- Read-only transactions – without sending uplink requests to servers