Data Dissemination by Broadcast

Evaggelia Pitoura

Computer Science Department,
University of Ioannina,
Ioannina, Greece
http://www.cs.uoi.gr/~ pitoura

Summer School, Jyvaskyla, August 1998

Introduction

Pull-based data delivery or on demand data delivery: A client explicitly requests data items from the server.

Push-based data delivery: The server repetitively broadcasts data to a client population without a specific request. Clients monitor the broadcast and retrieve the data items they need as they arrive.

Physical support in wireless computing.

Asymmetric

Applications:

Dissemination-based: information feeds such as stock quotes and sport tickets, electronic newsletters, mailing lists, traffic and weather information systems, cable TV

on the Internet (e.g., [15, 30]). Commercial Products for example: the AirMedia's Live Internet broadcast network [6] Hughes Network Systems' DirectPC [26]

Teletext and Videotex systems [11, 28]

The Datacycle project [16] at Bellcore: a database circulates on a high bandwidth network (140 Mbps). Users query the database by filtering information via special massively parallel transceivers.

The Boston Community Information System (BCIS) [18]: broadcast news and information over an FM channel to clients with personal computers equipped with radio receivers.

Hybrid Delivery

Push vs Pull

- Push suitable when information is transmitted to a large number of clients with overlapping interests

  the server saves several messages

  the server 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 only sequential; Thus, access latency degrades with the volume of data In pull, clients play a more active role

Hybrid Delivery

clients are provided with an uplink channel, called *backchannel*, to send messages to the server.

Sharing the channel :

if the same channel is used for both broadcast delivery and for the transmission of the replies to on demand requests

Hybrid Delivery

Use of the backchannel

- to provide feedback and profile information to the server

- to directly request data

Which pages? to avoid overwhelming the server

Page i not in cache and the number of items scheduled to appear before i on the broadcast is greater than a threshold parameter [2]

Selective Broadcast

Broadcast an appropriately selected subset of items and provide the rest on demand

In [25], the broadcast is used as an *air-cache* for storing frequently requested data. The broadcast content continuously adjusts to match the hot-spot of the database. The hot-spot is calculated by observing broadcast misses indicated by explicit requests for data not on the broadcast.

In [19]: the database is partitioned into: a "publication group" that is broadcast and an "on demand" group. The criterion for partitioning is to minimize the backchannel requests while constraining the response time below a predefined upper limit.

Hybrid Delivery

On Demand Broadcast

the server chooses the next item to broadcast on every broadcast tick based on the requests for data it has received

Various strategies [28]: broadcast the pages in the order they are requested (FCFS), or the page with the maximum number of pending requests.

A parameterized algorithm for large-scale data broadcast based only on the current queue of pending requests [7]

Mobility of users?

when users move to areas covered by different servers

differences in the type of communication infrastructure and thus in the capacity to service requests.

the distribution of requests for specific data at each cell changes.

In [17]: two variations of an adaptive algorithm for mobility in a cellular architecture. The algorithm statistically selects data to be broadcast based on user profiles and registration in each cell.

Organization of Broadcast Data

Access time: average time elapsed from the moment a client expresses its interest to an item to the receipt of the item on the broadcast channel

Tuning time: the amount of time spent listening to the broadcast channel

Organize the broadcast to minimize access and tuning time

Flat Organization

Broadcast the union of the requested data cyclicly

Broadcast Disks

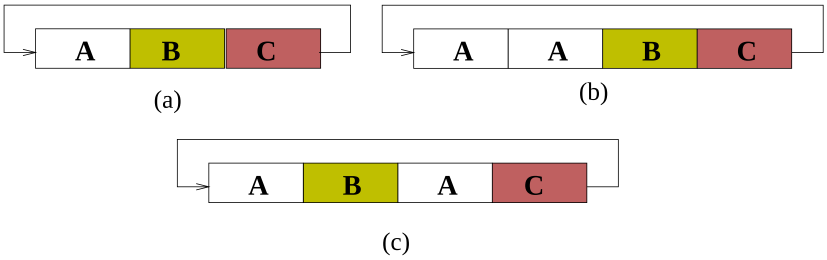
Broadcast data items that are most likely to be of interest to a larger part of the client community more frequently than others.

Consider data broadcast with the same frequency as belonging to the same disk. Thus, multiple disks of different sizes and speeds superimposed on the broadcast medium [3, 1].

Not simply a bandwidth allocation problem: Given all clients' access probabilities, the server determines the optimal percentage of the broadcast bandwidth to be allocated to each item. Then, the broadcast program is generated randomly, such that the average interarrival time between two instances of the same item matches the clients' needs. Not optimal in terms of minimizing expected delay for an item due to variance in the interarrival times.

Broadcast Disks

A simple example [3, 1]: three different organizations of broadcast items of equal length: (a) is a flat broadcast, in (b) and (c) A is broadcast twice as often as B and C.



(b) is a skewed (random) broadcast whereas (c) is regular since there is no variance in the interarrival time of each item. The performance characteristics of (c) are the same as if A was stored on a disk that is spinning twice as fast as the disk containing B and C. Thus, (c) can be seen as a multidisk broadcast. In terms of the expected delay, the multidisk broadcast (c) always performs better that the skewed one (b) [1]

Parameters: first, the number of disks, (i.e., different frequencies); and then, for each disk, the number of items and the relative frequency of broadcast.

Given a list of items ordered by their expected access probabilities and a specification of the disk parameters, *an algorithm* [3] that assigns items to disks and determines the interleaving of disks. The algorithm produces a periodic broadcast program with fixed interarrival times per item.

Indexing

Motivation

Clients interested in fetching from the broadcast individual data items identified by some key.

Provide a directory indicating when a specific data item appears in the broadcast so that each client needs only selectively tune in the channel to download the required data

Broadcast the directory along with data [22, 20, 21] to minimize access and tuning time.

Example: flat broadcast with no index information

For a broadcast of size $Data$ data items, the average access time is $Data/2$. On the other hand, the average tuning time equals $Data/2$

Approaches

1. (1, m) indexing [20]

Broadcast the whole index every a fraction $(1/m)$ of the broadcast data items. All items have an offset to the beginning of the next index item.

To access a record, a client tunes into the current item on the channel, uses the offset to tune in again when the index appears. From the index, it determines the required data item.

Optimum value of m: $\sqrt{Data/Index}$

Indexing

In (1, m) method, the index is broadcast m times during each period of the broadcast.

2. Distributed indexing [20]

Improves over the (1, m) method: instead of replicating the whole index, each index segment describes only data items which immediately follow.

3. Flexible Indexing [21]

The broadcast is divided into p data segments. The items of the broadcast are assumed to be sorted.

Each data segment is preceded by a control index. The control index consists of: a *binary control index* used to determine the data segment where the key is located by performing a binary search and a *local index* then used to locate the specific item inside the segment.

4. Hash-Based Techniques [21]

Instead of broadcasting a separate directory, simply broadcast along with each data item hashing parameters.

If h is perfect, then a client requesting K, tunes in, reads h, computes $h(K)$, and waits for bucket $h(K)$.

In case of collisions, various placements of the overflow buckets with varying tuning and access times.

Updates and Broadcast

Types of changes: [3].

1. the *content* of the broadcast can change in terms of including new items and removing existing ones (e.g., as in hybrid delivery)
2. the organization of the broadcast data (e.g., different index schema or changing the frequency of transmission of a specific item)
3. the values of broadcast data

In the last case, various issues

- when to propagate new values
- do clients perform updates?
- consistency

  A number of consistency models [4], e.g., *latest value:* (clients read the most recent value of a data item), *periodic* etc.
- Cache invalidation

Caching

Clients cache data items to reduce the expected delay

thus, they also lessen their dependency on the server's choice of broadcast priority

Replacement Policies

Traditionally, clients cache their hottest data mostly to improve the cache hit ratio.

In general, the cost of obtaining a page on a cache miss is consider constant.

However, in broadcast systems, the cost of servicing a miss on a page depends on when the requested page will appear on the broadcast. Thus:

Cost-Based Page Replacement [3, 1]:

the cost of obtaining a page on a cache miss is taken into account

The PIX *method:*

replace the cache page having the lowest pix ratio: between 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.

Caching

Prefetching

Client prefetch pages in anticipation of future accesses.

Traditionally, prefetching places additional load on the server and the network. In broadcast, only on the client's local resources.

Prefetching instead of page replacement can reduce the cost of a miss [5].

– *PT prefetching heuristic* [5, 1]

uses the *pt* value to decide whether the current broadcast page is more valuable than some other page in the cache.

The *pt* value is the product of the probability (P) of access for a page and the amount of time (T) before that page appears again

For each page at the broadcast, PT finds the page in cache that has the lowest *pt* value, and replaces it with the currently broadcast page if the latter has a higher *pt* value.

– *Teletext approach* [12]

Control information along with each broadcast page: a linked list of pages most likely to be requested next.

When a request for p is satisfied, the client prefetches the D most likely referenced pages associated with p (D: cache size).

Prefetching terminates either when D pages are prefetched or when the client submits a new request.

Cache Invalidation

The server broadcasts invalidation information to its client.

When?

- *asynchronous*
  broadcast an invalidation report for a given item as soon as its value changes
- *synchronous*
  broadcast invalidation reports periodically.

  Thus, a client has to listen the report first to decide whether its cache is valid or not. That adds some latency to query processing.

To whom?

- *stateful server*
  the server maintains information about its clients, e.g., the contents of their cache and when it was last validated.
- *stateless server*

What?

- invalidation: only which items were updated
- propagation: the values of items that have been updated
- information for individual items or aggregate information for sets of items

Cache Invalidation

False alarms

– Three synchronous strategies for stateless servers [14]:

- The *timestamps* strategy (TS):
  broadcast the timestamps of the latest change for items that have had updates in the last w seconds.
- The *amnestic terminals* strategy (AT):
  broadcast only the identifiers of the items that changed since the last invalidation report.
- The *signatures* strategy:
  broadcast signatures. A signature is a checksum computed over the value of a number of items by applying data compression techniques similar to those used for file comparison.

Clients that are often connected are called workalcoholic, while clients that are often disconnected are the sleepers. Each strategy effective for different type of clients.

Cache Invalidation

– *Bit Sequences* [23]

Asynchronous method: the invalidation report as a set of bit sequences with an associated set of timestamps. Each bit represents a data item.

A bit "1" indicates that the corresponding item has been updated since the time specified by the associated timestamp.

The set of bit sequences is organized in a hierarchical structure.

– ***How can disconnected clients that miss invalidation reports reuse part of their cache?***

Synchronous methods surpass asynchronous in that clients need only periodically tune in. But, if inactive longer than the broadcast period?

Check (validate) the content of the cache

For example, send the identities of all cached objects along with their timestamps to the server for validation.

Alternatively, send group identifiers and timestamps, and check the validity at the *group level*.

Volume checking in the Coda file system.

However, a single object update invalidates the whole group. *GCORE* [29]: the server identifies for each group a hot update set and excludes it from the group when checking the group's validity.

Consistency Control

Certification Reports [13]

In optimistic cc, at commit time, the transaction scheduler checks whether the execution that includes the transaction is serializable or not. If it is, it accepts the transaction.

The server periodically broadcasts to its clients a certification report (CR) that includes the readset and writeset of active transactions that have declared their intention to commit during the previous period and have been certified.

The client uses this information to abort from its transactions those transactions whose readsets and writesets intersect with the current CR. If not aborted, it is send to the server.

The server installs the values in the central database and notifies the clients via broadcast.

New Models

Read-Only Transactions

Other Topics

View Maintenance [27]

- views may contain location and time dependent continuously changing data
- exploit transmission by broadcast, e.g., to update views
- address predictable disconnections

Power-Efficient Query Processing

Selecting energy-efficient query plans during query optimization [8]

Database Interfaces

Constraints such as limited screen size, a need for semantic simplicity, slow and expensive wireless communications, and limited power [9, 10].

A query processing facility, Query by Icons (QBI), that accommodates these constraints [24].

Iconic visual language, use of a pointing device, like a light-pen, semantic data model that encapsulates and hides details.

Metaquery tools during disconnections: a query is formulated in an incremental manner; actual data to materialize intermediate steps are not accessed from remote sites.

References

[1] S. Acharya, R. Alonso, M. J. Franklin, and S. Zdonik. Broadcast Disks: Data Management for Asymmetric Communications Environments. In *Proceedings of the ACM SIGMOD Intl. Conference on Management of Data (SIGMOD 95)*, June 1995. Reprinted in Mobile Computing, Imielinski and Korth, Eds., Kluwer Academic Publishers, 1996.

[2] S. Acharya, M. Franklin, and S. Zdonik. Balancing Push and Pull for Data Broadcast. In *Proceedings of the ACM Sigmod Conference*, 1997.

[3] S. Acharya, M. J. Franklin, and S. Zdonik. Dissemination-based Data Delivery Using Broadcast Disks. *IEEE Personal Communications*, 2(6), December 1995.

[4] S. Acharya, M. J. Franklin, and S. Zdonik. Disseminating Updates on Broadcast Disks. In *Proceedings of the 22nd International Conference on Very Large Data Bases (VLDB 96)*, September 1996.

[5] S. Acharya, M. J. Franklin, and S. Zdonik. Prefetching from a Broadcast Disk. In *Proceedings of the 12th International Conference on Data Engineering (ICDE 96)*, February 1996.

[6] AirMedia. AirMedia Live. www.airmedia.com.

[7] D. Aksoy and M. J. Franklin. Scheduling for Large-Scale On-Demand Data Broadcasting. In *IEEE INFOCOM '98*, 1998.

[8] R. Alonso and S. Ganguly. Query Optimization for Energy Efficiency in Mobile Environments. In *Proceedings of the 1993 Workshop on Optimization in Database Systems*, 1993.

[9] R. Alonso, E. M. Haber, and H. F. Korth. A Database Interface for Mobile Computers. In *Proceedings of the 1992 Globecomm Workshop on Networking for Personal Communications Applications*, 1992.

[10] R. Alonso, E. M. Haber, and H. F. Korth. A Mobile Computer Interface for Heterogeneous Databases. In *Proceedings of the RIDE-IMS Workshop*, April 1993.

[11] A. H. Ammar and J. W. Wong. The Design of Teletext Broadcast Cycles. *Performance Evaluation*, 5(4), 1985.

[12] M. H. Ammar. Response Time in a Teletext System: an Individual User's Perspective. *IEEE Transactions on Communications*, 35(11), November 1987.

[13] D. Barbará. Certification Reports: Supporting Transactions in Wireless Systems. In *Proceedings of the IEEE International Conference on Distributed Computing Systems*, 1997.

[14] D. Barbará and T. Imielinski. Sleepers and Workaholics: Caching Strategies in Mobile Environments. In *Proceedings of the ACM SIGMOD Intl. Conference on Management of Data (SIGMOD 94)*, pages 1–12, 1994.

[15] A. Bestavros and C. Cunha. Server-initiated Document Dissemination for the WWW. *IEEE Data Engineering Bulletin*, 19(3), September 196.

[16] T. Bowen, G. Gopal, G. Herman, T. Hickey, K. Lee, W. Mansfield, J. Raitz, and A. Weinrib. The Datacycle Architecture. *Commuinications of the ACM*, 35(12), December 1992.

[17] A. Datta, A. Celik, J. Kim, D. VanderMeer, and V. Kumar. Adaptive Broadcast Protocols to Support Efficient and Energy Conserving Retrieval from Databases in Mobile Computing Environments. In *Proceedings of the 13th IEEE International Conference on Data Engineering*, April 1997.

[18] D. Gifford. Polychannel Systems for Mass Digital Communication. *Communications of the ACM*, 33(2), 1990.

[19] T. Imielinski and S. Viswanathan. Adaptive Wireless Information Systems. In *Proceedings of the SIGDBS Conference*, October 1994.

[20] T. Imielinski, S. Viswanathan, and B. R. Badrinanth. Energy Efficient Indexing on Air. In *Proceedings of the ACM SIGMOD Intl. Conference on Management of Data (SIGMOD 94)*, pages 25–36, 1994.

[21] T. Imielinski, S. Viswanathan, and B. R. Badrinanth. Power Efficient Filtering of Data on Air. In *Proceedings of the the 4th International Conference on Extending Database Technology*, March 1994.

[22] T. Imielinski, S. Viswanathan, and B. R. Badrinanth. Data on Air: Organization and Access. *IEEE Transactions on Knowledge and Data Engineering*, 9(3):353–372, May/June 1997.

[23] J. Jing, Bukhres O, K. Elmargarmid A, and R. Alonso. Bit-Sequences: A New Cache Invalidation Method in Mobile Environments. Technical Report CSD-TR-94-074, Revised May 95, Department of Computer Sciences, Purdue University, 1995.

[24] A. Massari, S. Weissman, and P. K. Chrysanthis. Supporting Mobile Database Access through Query by Icons. *Distributed and Parallel Databases*, 4:249–269, 1996.

[25] K. Stathatos, N. Roussopoulos, and J. S. Baras. Adaptive Data Broadcast in Hybrid Networks. In *Proceedings of the 23rd VLDB Conference*, 1997.

[26] Hughes Network Systems. DirectPC Homepage. www.direcpc.com, 1997.

[27] O. Wolfson, P. Sistla, S. Dao, K. Narayanan, and R. Raj. View Maintenance in Mobile Computing. *Sigmod Record*, September 1995.

[28] J. Wong. Broadcast Delivery. *Proceedings of the IEEE*, 76(12), December 1988.

[29] K-L. Wu, P. S. Yu, and M-S. Chen. Energy-Efficient Caching for Wireless Mobile Computing. In *Proceedings of the 12th International Conference on Data Engineering (ICDE 96)*, February 1996.

[30] T. Yan and H. Garcia-Molina. SIFT – A Tool for Wide-area Information Dissemination. In *Proceedings of the 1995 USENIX Technical Conference*, 1995.