Part II

Related Work

Chapter 4

Broadcast Systems

This and the following chapter provide an extensive presentation of work related to *FCLOS*. *FCLOS* is a complete mOLAP architecture, involving several research fields. Thus, the following paragraphs discuss related works to different components of our system. While issues of distributed data warehousing and mOLAP are presented in Chapter 5, this chapter describes the most important work on data broadcast systems. Section 4.1 deals with general data broadcast systems, i.e., systems which operate regardless of the data content. In Section 4.2, we present systems which exclusively handle database data items, i.e., clients pose database queries instead of data item requests. Finally, Section 4.3 explains why the presented approaches are unsuitable for the mOLAP domain.

4.1 General Data Broadcast Systems

As described in Section 3.2.2, data broadcast can be managed in three modes: on-demand, push and hybrid. This section presents a selection of generic data broadcast systems. State of the art mOLAP architectures, presented in Section 5.3, borrow key ideas from systems presented in this section.

4.1.1 On-demand

Related work in the area of on-demand data broadcast has mainly concentrated on the definition of efficient metrics, while maintaining a moderate complexity. Additional issues to be considered are: data staging, transmission error robustness and the presence of deadlines.

Metrics

[10] constitutes a pioneer approach, introducing not only new scheduling algorithms, but metrics for on-demand broadcast as well. Its authors argue that in heterogeneous settings, access time alone is not a fair measure of individual performance given that the individual requests significantly differ from one another

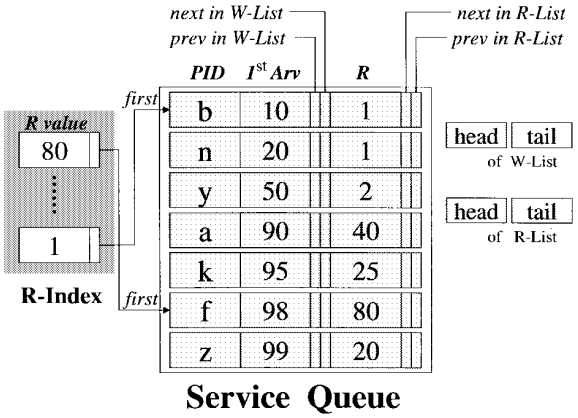

Figure 4.1: The $R \times W$ service-queue data structures (Source: [13])

in their service times. They adopt an alternative performance measure, namely the *stretch* of a request, which for a request i is defined as:

$$stretch_i = \frac{AT_i}{ST_i} \tag{4.1}$$

where AT is the access time for a request and ST its service time, namely the time request i would need to complete if it were the only system job. The rationale for this choice is that the stretch of a job translates more directly to user-perceived performance. Intuitively, clients with larger jobs should be expected to wait in the system longer than those with smaller requests (stretch metrics are also discussed in [26]). Beyond this, it is argued that a fair evaluation must divide the jobs into different classes. The definition of the classes is however dependent on the application domain. Furthermore, schedulers are classified according to *preemptions*. Preemption refers to interrupting a broadcast to serve others requests before resuming the remainder of the original broadcast. In this context, four algorithms are presented: *PLWF* (Preemptive Longest Wait First), *SRST* (Shortest Remaining Service Time), *LTSF* (Longest Total Stretch First) and the *Base* algorithm.

In [13], the $R \times W$ metric is proposed, where R stands for the number of requests for a specific data item and W for the waiting time of a request (1st arrival). The objective of this simple algorithm is to provide a balanced treatment of cold and hot items. This algorithm exhibits fair scalability and robustness without increased complexity. Figure 4.1 shows an example of the scheduling function. It maintains a structure containing a single entry for each page that has outstanding requests. This structure is hashed on page identifiers (PID). Obviously, in this example page f is scheduled for the next transmission because its metric ($R \times W = 80 \times 98 = 7840$) has the maximum value.

$R \times W.a$ is an approximate version of $R \times W$. Its objective is to reduce the search overhead of $R \times W$. The a parameter is tunable in percent and controls

the desired level of approximation. The idea is to prune the search space, since the chosen data item will most likely be one with high R-value or high W-value.

[81] introduces a heuristic under the name *ATWT* (Approximate Total Waiting Time), which is an approximate version of the *LWF* (Longest Wait First) algorithm [165]. Although *ATWT* exhibits reduced decision overhead compared to *LWF* and $R \times W$, it fails to provide any substantial performance optimization in terms of access time.

Data Staging

[14] focuses on the *data staging* problem for on-demand broadcast systems. It argues that the implicit assumption, that all items to be disseminated are readily available at the server when they are scheduled to be broadcast, is not always valid. Frequently, the items to be broadcast may reside in secondary, tertiary, or even remote storage. The need to fetch data from such locations produces large variance in service times, which can destroy the performance of the broadcast scheduling heuristics. In this context, the authors argue that the decision on which available data item to broadcast should be based on the *overall* available bandwidth rather than the scheduling heuristics.

The data staging problem is the topic of [157] too. Taking into account how broadcast scheduling, disk scheduling and cache management affect the overall performance, four algorithms are proposed: *ADoRe*, *FLUSH*, *OWeiST* and $R \times W/S$, classified under two categories: those that combine separate, *off-the-shelve* broadcast and disk scheduling algorithms, and those that combine the information available at the broadcast queue and at the disk queue, producing a single scheduling criterion. The $\frac{R \times W}{S}$ metric is obviously based on the $R \times W$ metric, adding the S clause, which represents the disk service time. Thus, the selection of the data item to be broadcast prioritizes items with high $R \times W$ values (as the $R \times W$ algorithm suggests) and low disk access times (as a consequence of the S clause). This metric is used by *STOBS*, a mOLAP architecture presented in 5.3.2.

Transmission Errors - Deadlines

[154] presents the *LDCF* algorithm, which considers not only access time as performance measure, but *request failure* as well. The intuition behind this is that the permission of endless waiting incurs problems. On the one hand, when the server does not receive the access requests because of transmission errors, mobile users wait for responses in vain. On the other hand, responding to obsolete requests also leads to inefficiency because the mobile user might have dropped its interest.

The same issue, also known as the *deadline* problem, is targeted by *MAI* (Multiple Integration Algorithm) [33]. In [59], on-demand scheduling is considered as a problem related to fair queuing [147]. The proposed algorithm takes into account transmission errors. Their solution is extended in order to be applicable to more than one broadcast channels.

The problem of deadlines posed by clients is also known as the problem of time or temporal constraints. Good scheduling algorithm for real-time on-demand broadcast can guarantee as many requests as possible to meet their deadlines with limited bandwidth. *RDDS* [168] and the preemptive *PRDS* [93] are two very similar approaches towards scheduling of real-time requests. The main prioritizing function used is:

$$P_i = \frac{R_i}{D_i \times S_i} \tag{4.2}$$

where R_i is the number of pending feasible requests for data object i, D_i and S_i are the effective deadline and size of i, respectively. The authors argue that taking deadline into consideration improves real-time performance.

4.1.2 Push-based

Broadcast Disks [5] is a pioneer approach in the area of pure push-based broadcast systems. The authors suggest that the design of such systems should consider the broadcast program and the cache management together. The architecture assumes multiple disks spinning at different speeds on a single broadcast channel and addresses two issues: The construction of a broadcast program that satisfies the clients' needs and the cache management given the broadcast program.

[7] identifies that the demand-driven access of Broadcast Disks, does not fully exploit the dissemination-based nature of the broadcast, which is particularly conducive to client prefetching. With a Broadcast Disk pages continually flow past the clients so that, in contrast to traditional environments, prefetching can be performed without placing additional load on shared resources. The authors propose a simple prefetch heuristic called *P7*, which balances the cache residency time of a data item with its bandwidth allocation. The same authors deal with the issue of updates in the broadcast data [9]. They identify a fundamental tradeoff between data currency and performance. In this context, they propose a method to ensure enhanced performance and robustness, even in cases in which updates have to be broadcast immediately.

Naturally, there are many more proposals. The following paragraphs present further related work in the area push-based systems. It is beyond the scope of this document to thoroughly review every approach. Instead we refer to the most representative approaches, categorized according to their specific focus.

Optimization of the Broadcast Schedule

The optimization of the broadcast schedule has attracted the most attention in this area. [158] proposes a *bucketing* scheme that facilitates the tradeoff between time complexity and performance of the scheduling algorithm. Moreover, an algorithm for broadcast scheduling in the presence of errors is proposed. When different clients are capable of listening on different number of broadcast channels, the schedules on different broadcast channels are designed so as to minimize the

access time for all clients. The clients listening to multiple channels experience proportionately lower delays.

[153] considers the broadcast schedule as a deterministic dynamic optimization problem, the solution of which provides the optimal broadcast schedule. By obtaining the properties of the optimal solution, a suboptimal dynamic policy is proposed. The policy has low complexity, is adaptive to changing access statistics and generalizable to multiple broadcast channels.

Typically, it is assumed that the access cost is proportional to the waiting time. [21] examines the best broadcast schedules for access costs that are arbitrary *polynomials* in the waiting time. These may serve as reasonable representations of reality in many cases, where the patience of a client is not necessarily proportional to its waiting time. The authors present an asymptotically optimal algorithm for a fractional model, where the bandwidth may be divided to allow for fractional concurrent broadcast.

Finally, *Cascaded Webcasting* [82] tackles the issue of creating hierarchical Webcasting programs, by exploiting the given skewness in web access probabilities.

Optimization of the Broadcast Disk Array

A system of multiple broadcast channels can be viewed as a *broadcast disk array*. The broadcast disks in an array can be categorized according to their speed, where the speed of a broadcast disk corresponds to the expected delay for the data items in that broadcast disk. In this context, more recent works explicitly concentrate on algorithms that allocate data items to the broadcast disk array according to their access frequencies. [65] focuses on the generation of broadcast programs for multiple channels by using heuristics. A more convincing approach appears in [120]. The authors transform the problem of generating hierarchical broadcast programs into the one of constructing a channel allocation tree with variant-fanout. By exploiting the feature of tree generation with variant-fanout, they develop a heuristic algorithm called *VFK* to minimize the expected delay of data items in the broadcast program.

Dependent Data

Most broadcast systems operate under the premise that each user requests only one data item at a time, and that the requests for all data items are independent. That is, for an arbitrary user, the access probability that the user requests a data item in the *ith* request is predetermined and is independent of what has been previously requested. However, in many real applications, some data items are semantically related, and consequently, there exists dependencies among the requests of these data items.

[91] argues that the problems of deciding the content of the broadcast channel based on clients' requests, and scheduling multiple (dependent) data items to be broadcast are NP-complete. Therefore, different heuristics to these problems are

proposed. In [67], the authors derive the theoretical properties for the average access time in multiple channel environments, which help them develop a genetic algorithm to generate broadcast programs.

Cache Management

Cache management issues have also been extensively addressed in the past years. With caching, clients need only to wait for broadcast if the desired item is not in the cache. However, it has been shown that traditional caching techniques in broadcast environments are inadequate. [6] proposes a cost-based page replacement heuristic, as well as a cost-based page prefetching heuristic, which helps the cache manager to decide what pages, i.e., data items, should be evicted out of cache and what pages should be prefetched into cache when space is available. The idea is that not only pages likely to be requested in the future, as usually thought, but also pages that are not broadcast very often should be kept in cache because the cost of acquisition for those pages is pretty high once cache misses. [167] deals with nonuniform broadcast systems, in which hot data is broadcast more frequently than cold data and proposes a cooperative cache management scheme. The proposal of [32], based on a novel prefetch-access ratio concept, can dynamically optimize performance or power based on the available resources and the performance requirements. *GD-LU* [143] is a further approach that enhances dynamic data availability while maintaining consistency. The proposed utility-based caching mechanism considers several characteristics of mobile distributed systems, such as connection-disconnection, mobility handoff, data update and user request patterns. The main objective is energy efficiency.

Evaluation

Finally, [77] argues that in addition to mean response time, the variance of response time should also be taken into account by the broadcast scheduler. A single transmission of a data item satisfies all pending requests for that item. The response time of a request depends on the broadcast time of the desired data item, which is scheduled by the server according to the overall demands for various data items. Therefore, the response time may vary in a large range. In this context, the authors address the issue of variance optimization in regard to response time. Naturally, this issue is also indirectly addressed by the stretch metric, described in Section 4.1.1.

4.1.3 Hybrid

As already mentioned in Section 3.2.2, hybrid data dissemination is a combination of on-demand and push-based approaches ([15] provides a fair comparison between on-demand and push-based systems). Typical design issues in hybrid systems are channel allocation (number of push and on-demand channels), data classification (cold or hot) and item scheduling (both on on-demand and push channel). In this

Table 4.1: Evaluation symbols

Value	Symbol
Excellent, very, low	++
Good, low, high	+
N/a, medium, no	**0**
Poor, high, some	**-**
Very poor, very high, many	**- -**

context, several approaches trying to minimize the client's access time have been proposed. The authors of [41] investigate two broadcast strategies (constant or variable broadcast size). Requested data items are broadcast in batches, using the indexing techniques of [74]. [8] exclusively deals with the bandwidth allocation problem given a static broadcast schedule. The system proposed in [151] adapts the broadcast content to match the hot-spot of the database. This spot can be accurately obtained by monitoring the broadcast misses and therefore no other implicit knowledge on the actual usage of the broadcast data is necessary. In [56], this hot-spot is analytically derived. In [25] and subsequently in [24], the authors recognize that the problems of channel allocation, data classification and item scheduling are rather interconnected and treat them together.

TC-AHB [48] is a hybrid broadcast model for dynamic and time-critical communication environments. Compared to previous approaches, the notion of deadlines is included. *GDS* [70] is a hybrid system, which correctly identifies that the handled data items have an arbitrary size. Thus, broadcast schedules considering fixed data item size cannot perform optimally, since their optimization policy founds on unrealistic assumptions. The proposed solution is based on two analytical models, modeling both the on-demand and the broadcast channel.

4.1.4 Evaluation of Data Broadcast Modes

Having presented the three broadcast modes, we summarize our discussion by comparing them in Table 4.2 according to specific criteria. The criteria are weighed using the symbols of Table 4.1:

Performance is clearly dependent on the application domain and the client population. Push-based systems exhibit almost unlimited scalability, while as already mentioned on-demand systems are suitable to relatively small client population. However, push-based systems do not receive explicit client requests, and therefore remain unaware of changes in the population or the load. Primarily hybrid and secondarily on-demand systems, introduce increased server and client complexity. Nonetheless, this enables enhanced application generality. Push-based systems typically operate in a specific application domain.

Table 4.2: Evaluation of data broadcast modes

	On-demand	**Push-based**	**Hybrid**
Performance	0	0	0
Scalability	- -	++	+
Self adaptiveness	++	- -	+
Server complexity	-	+	- -
Client complexity	-	+	-
Application generality	++	- -	+

4.2 Database Broadcast Systems

The previous section discussed general data broadcast systems, namely systems that do not consider the content or semantic dependencies between broadcast data items. However, as already described in Section 3.3, in the case of database data items, namely when clients issue queries and not data item requests, efficient data broadcast becomes more complex. The maintenance of the ACID properties becomes a major issue. [107] summarizes the most important issues in mobile transaction management. The following paragraphs describe several proposed solutions.

Consistency and Concurrency

[124] addresses the problem of ensuring consistency and concurrency of client read-only transactions in the presence of updates in push-based broadcast. Its authors propose four scalable techniques. Scalability is achieved as a consequence of the fact that query processing is performed locally. The selection between these techniques depends on the expected behavior in terms of concurrency (percentage of accepted transactions), processing overhead, size, latency, currency (database state seen by the clients) and disconnection tolerance.

BCC-TI [94, 95] is a concurrency control protocol adapted from the optimistic concurrency control with forward validation protocol. The protocol, which is push-based, offers autonomy between the mobile clients and the server such that mobile clients can read consistent data off the air without contacting the server. To reduce the number of unnecessary transaction restarts, so that the timeliness of mobile transactions can be enhanced, the protocol is based on timestamp ordering. By exploiting the semantics of read-only transactions, the timestamp ordering technique profits from the separate processing and flexible adjustment of serialization order.

[119] considers on-demand query processing in mobile environments, which involves join processing among different sites including static servers and mobile computers. Based on the *semijoin* operator, the authors propose query processing methods for both join and query processing. According to the asymmetric features of mobile computing systems, three different join methods are proposed, as well

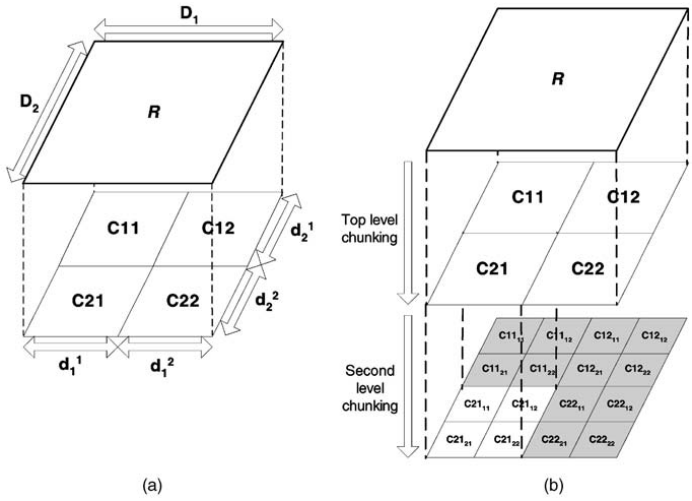

Figure 4.2: Chunking model of a relation (Source: [92])

as specific criteria to identify profitable semijoins. For multijoin query processing three query processing schemes are also proposed.

[127], also in on-demand architectures, examines requests for *multi-items*. This term refers to requests that need more than one data item to be served. The main focus is on keeping transaction consistency, while minimizing access time.

In [137], the family of *MVCC* protocols for hybrid broadcast is proposed. *MVCC* deals with mobile read-only transactions. To achieve consistency and currency guarantees, its authors define four isolation levels (ILs), explicitly suited for the specific domain.

Realizing that serializability as the correctness criterion may be expensive, and more importantly perhaps even unnecessary in such environments, various protocols [96, 124, 125, 138] attempt less demanding correctness requirements. However, the exact semantic and temporal coherency properties associated with them are not always clear. [126] provides adequate clarity by developing a general theory of temporal and semantic coherency.

In [144], the problem of updates is examined under the name *multiversion* data broadcast. The authors identify two basic multiversion organizations, namely *Vertical* and *Horizontal* broadcasts, and propose an efficient compression scheme applicable to both. They also examine the applicability of these schemes in the context of both single disk and multiple disk broadcasts.

Chunking

[92] presents a data broadcast scheme based on *chunking*. Chunks are a regular decomposition of the multidimensional space [44]. Figure 4.2 depicts the chunking of a relation R along two dimensions on domains D_1 and D_2. Accordingly, the

authors propose the decomposition of every query into appropriate chunks and propose an indexing scheme to support it. To evaluate a query, mobile clients listen to the broadcast semantic descriptor and identify qualified broadcast chunks. Due to the fact that a qualified chunk might not cover all tuples required by the query, the client may have to look for the remaining tuples from other chunks.

Despite its strong points, the system should only be considered for relatively small client population because beyond being an on-demand system, it requires additional interaction between server and client than most of the systems. Moreover, there is a complexity overhead for its complicated indexing scheme. Unfortunately, the authors do not provide extensive performance evaluation (especially compared to previous approaches) in order to fully appreciate the merit of the work.

4.3 Mobile OLAP Suitability

Having presented the most important previous approaches, it is important to note that our proposed solution, *FCLOS*, is neither a pure on-demand nor a push-based nor a hybrid architecture. *FCLOS* reacts to explicit client queries and does not employ any fixed broadcast program. In this sense, it could be classified as an on-demand architecture. However, the subsumption based handling of incoming queries and the extensive usage of broadcast resembles a push-based architecture.

Efficient and robust mOLAP cannot be provided by general data broadcast systems presented in Section 4, regardless of the mode for the following reasons:

- *Data semantics*: Content or characteristics of data items are not considered. On the contrary though, multidimensional data cubes are semantically connected to each other.
- *Thin/Fat clients*: Due to the underlying assumption of thin clients, transmitted data is processed for direct end usage. As mobile devices become increasingly powerful, a lot of space for load balancing is open. Client local processing, not only improves the performance of the entire system, but enables offline functionality as well.
- *OLAP end user behavior*: OLAP end users typically navigate through the requested data cubes, performing typical OLAP operations such as roll-ups or drill-downs. This has to be considered in the design of the dissemination system in order to avoid unnecessary transmissions.
- *Offline functionality*: The offline aspect is completely absent, again due to the assumption of a thin client population. However, the entire concept of mOLAP systems is founded on providing offline functionality. This is crucial when considering that mobile devices are not permanently connected to a network.

Beyond the above observations, different broadcast modes exhibit additional shortcomings. On-demand systems are notoriously not scalable, since they fail

to keep up with growing client population and consequently growing number of incoming requests. Although they can efficiently serve a limited number of end users, they are definitely inappropriate as this number increases.

Scalability is addressed by push-based data broadcast. Nevertheless, conventional push-based broadcast is also unsuitable to mOLAP for the following reasons:

- *Data population*: The number of handled data items in mOLAP is not limited, which is a common assumption of existing push-based systems. Data items in mOLAP are query answers. As the number of possible queries is practically infinite, a typical broadcast schedule cannot be employed.
- *Data item size*: Data items do not occupy relatively small size (e.g., web pages), which is also a common assumption of existing push-based systems. Multidimensional data cubes are items order of magnitude bigger than web pages.

Moreover, not only in mOLAP but generally, push-based systems unnecessarily consume bandwidth when the number of incoming requests is relatively low.

Hybrid dissemination systems suffer more or less from the aforementioned observations. Although they are theoretically self adaptive, by dynamically assigning hot data items to push channels and cold data items to pull channels, in practice this incurs increased complexity and maintenance overhead.

mOLAP could borrow some ideas by database broadcast systems presented in Section 4.2, since these systems do consider data content. Nevertheless, the existing solutions focus on issues that are practically of minor importance for mOLAP:

- *Updates-Consistency*: In database broadcast systems, the dependencies between broadcast systems are considered with the objective to ensure data consistency, particularly in the presence of updates. In mOLAP, data is not so dynamic and the control overhead to ensure updated data cannot be justified.
- *Dependencies between data items*: The dependencies between data items in mOLAP is based on subsumption, which means that from one data object (sub-cube), other data objects (sub-cubes) can be computed. In database broadcast systems, data dependencies are considered in the sense that multiple broadcast data items may together form the query answer.

4.4 Summary

Limited wireless bandwidth has been mainly addressed by data broadcast. Data broadcast is a 1-to-n process, enabling enhanced scalability. The capacity of data transfer from the server to the mobile client (downstream communication) is significantly larger than that of the mobile client to the server (upstream communication). Data broadcast can be managed in three modes: on-demand, push

and hybrid. The choice of the broadcast mode is dependent on the application scenario.

General broadcast systems do not consider the semantics of broadcast data. Therefore, if used for mOLAP, they would ignore the semantic connection between sub-cubes, according to the subsumption. Database broadcast systems, dealing with queries and not data item requests, do consider content. However, the main issues of such systems, such as the maintenance of the ACID properties, are of little or no importance for the mOLAP domain.