► Data Compression in Sensor Networks
► Data Fusion in Sensor Networks

Recommended Reading

1. Considine J., Li F., Kollios G., and Byers J. Approximate aggregation techniques for sensor databases. In Proc. 20th Int. Conf. on Data Engineering, 2004, pp. 449–460.
2. Cormode G., Garofalakis M., Muthukrishnan S., and Rastogi R. Holistic aggregates in a networked world: distributed tracking of approximate quantiles. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2005, pp. 25–36.
3. Cormode G., Tirthapura S., and Xu B. Time-decaying sketches for sensor data aggregation. In Proc. ACM Symposium on Principles of Distributed Computing, 2007, pp. 215–224.
4. Deligiannakis A., Kotidis Y., and Roussopoulos N. Hierarchical in-network data aggregation with quality guarantees. In Advances in Database Technology, In Proc. 9th Int. Conf. on Extending Database Technology, 2004, pp. 658–675.
5. Gandhi S., Hershberger J., and Suri S. Approximate isocontours and spatial summaries for sensor networks. In Proc. 6th Int. Symp. Inf. Proc. in Sensor Networks, 2007, pp. 400–409.
6. Gao J., Guibas L.J., Milosavljevic N., and Hershberger J. Sparse data aggregation in sensor networks. In Proc. 6th Int. Symp. Inf. Proc. in Sensor Networks, 2007, pp. 430–439.
7. Greenwald M. and Khanna S. Power-conserving computation of order-statistics over sensor networks. In Proc. 23rd ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, 2004, pp. 275–285.
8. Hellerstein J.M., Hong W., Madden S., and Stanek K. Beyond average: toward sophisticated sensing with queries. In Proc. 2nd Int. Workshop Int. Proc. in Sensor Networks, 2003, pp. 63–79.
9. Intanagonwiwat C., Govindan R., and Estrin D. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In Proc. 6th Annual Int. Conf. on Mobile Computing and Networking, 2000, pp. 56–67.
10. Kempe D., Dobra A., and Gehrke J. Gossip-based computation of aggregate information. In Proc. 44th Annual Symp. on Foundations of Computer Science, 2003, pp. 482–491.
11. Luo H., Y. Liu, and S. Das. Routing correlated data with fusion cost in wireless sensor networks. IEEE Transactions on Mobile Computing, 11(5):1620–1632, 2006.
12. Madden S., Franklin M.J., Hellerstein J.M., and Hong W. TAG: a tiny aggregation service for ad-hoc sensor networks. In Proc. 5th USENIX Symp. on Operating System Design and Implementation, 2002.
13. Manjhi A., Nath S., and Gibbons P.B. Tributaries and deltas: efficient and robust aggregation in sensor network streams. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2005, pp. 287–298.
14. Nath S., Gibbons P.B., Seshan S., and Anderson Z.R. Synopsis diffusion for robust aggregation in sensor networks. In Proc. 2nd Int. Conf. on Embedded Networked Sensor Systems, 2004, pp. 250–262.
15. Silberstein A., Braynard R., Ellis C., and Munagala K. A sampling-based approach to optimizing top-k queries in sensor networks. In Proc. 22nd Int. Conf. on Data Engineering, 2006.
16. Silberstein A., Munagala K., and Yang J. Energy-efficient monitoring of extreme values in sensor networks. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2006.
17. Xue W., Luo Q., Chen L., and Liu Y. Contour map matching for event detection in sensor networks. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2006, pp. 145–156.

Data Analysis

► Data Mining

Data Anomalies

► Data Conflicts

Data Broadcasting, Caching and Replication in Mobile Computing

Panos K. Chrysanthis[1], Evaggelia Pitoura[2]
[1]University of Pittsburgh, Pittsburgh, PA, USA
[2]University of Ioannina, Ioannina, Greece

Synonyms

Data dissemination; Push/pull delivery; Data copy

Definition

Mobile computing devices (such as portable computers or cellular phones) have the ability to communicate while moving by being connected to the rest of the network through a wireless link. There are two general underlying infrastructures: single-hop and multi-hop ones. In *single-hop* infrastructures, each mobile device communicates with a stationary host, which corresponds to its point of attachment to the wired network. In *multi-hop* infrastructures, an ad-hoc wireless network is formed in which mobile hosts participate in routing messages among each other. In both infrastructures, the hosts between the source (or sources) and the requester of data (or data sink) form a dissemination tree. The hosts (mobile or stationary) that form the dissemination tree may store data and participate in computations towards achieving *in network* processing. Challenges include [14], (i) *intermittent connectivity*, which refers to both short and long periods of network

unavailability, (ii) *scarcity of resources*, including storage and battery life, and (iii) *mobility* itself.

To handle these challenges, data items may be stored locally (cached or replicated) at the requester or at the intermediate nodes of the dissemination tree. Cache and replication aim at increasing availability in the case of network disconnections or host failures as well as at handling intermittent connectivity. Mobility introduces additional challenges in maintaining cache and replica consistency and in replica placement protocols. Wireless data delivery in both infrastructures physically supports broadcasting. This broadcast facility has been used for providing a push-mode of data dissemination where a server broadcasts data to a large client population often without an explicit request from the clients. Issues addressed by related research include broadcast scheduling and organization (i.e., which items to broadcast and in which order), indexing broadcast data, and update propagation.

Historical Background

Mobile computing can be traced back to file systems and the need for disconnected operations in the late 1980s. With the rapid growth in mobile technologies and the cost effectiveness in deploying wireless networks in the 1990s, the goal of mobile computing was the support of AAA (anytime, anywhere and any-form) access to data by users from their portable computers, mobile phones and other devices with small displays and limited resources. These advances motivated research in data management in the early 1990s.

Foundations

Data Broadcasting

Many forms of wireless network infrastructures rely on broadcast technology to deliver data to large client populations. As opposed to point-to-point data delivery, broadcast delivery is scalable, since a single broadcast response can potentially satisfy many clients simultaneously. There are two basic modes of broadcast data delivery: pull-based and push-based. With *push-based* data delivery, the server sends data to clients without an explicit request. With *pull-based* or *on-demand* broadcast delivery, data are delivered only after a specific client request. In general, access to broadcast data is sequential with clients monitoring the broadcast channel and retrieving any data items of interest as they arrive. The smallest access unit of broadcast data is commonly called a *bucket* or *page*.

Scheduling and Organization

A central issue is determining the content of the broadcast or *broadcast scheduling*. Scheduling depends on whether we have on demand, push or hybrid delivery. In on-demand broadcast, there is an up-link channel available to clients to submit requests. The item to be broadcast next is chosen among those for which there are pending requests. Common heuristics for on-demand scheduling include First Come First Served and Longest Wait First [7]. The $R \times W$ strategy selects the data item with the maximal $R \times W$ value, where R is the number of pending requests for an item and W the amount of time that the oldest pending request for that item has spent waiting to be served [3]. More recent schemes extended $R \times W$ to consider the semantics of the requested data and applications such as subsumption properties in data cubes [15]. Push-based broadcast scheduling assumes a-priori knowledge of client access distributions and prepares an off-line schedule. Push-based data delivery is often periodic. In hybrid broadcast, the set of items is partitioned, so that some items are pushed, i.e., broadcast continuously, and the rest are pulled, i.e., broadcast only after being requested [2]. Commonly, the partition between push and pull data is based on popularity with the most popular items being pushed periodically and the rest delivered on demand. One problem is that for push items, there is no way to detect any changes in their popularity. One solution is to occasionally stop broadcasting some pushed items. This forces clients to send explicit requests for them, which can be used to estimate their popularity [16]. An alternative that avoids flooding of requests requires a percentage of the clients to submit an explicit request irrespective of whether or not a data item appears on the broadcast [5].

The organization of the broadcast content is often called *broadcast program*. In general, broadcast organizations can be classified as either *flat* where each item is broadcast exactly once or *skewed* where an item may appear more than once. One can also distinguish between *clustered* organizations, where data items having the same or similar values at some attribute appear consecutively, and *non-clustered* ones, where there is no such correlation. In skewed organizations, the broadcast frequency of each item depends on its popularity. For achieving optimal access latency or

response time, it was shown that (i) the relative number of appearances of items should be proportional to the square root of their access probabilities and (ii) successive broadcasts of the same item should be at equal distances [7]. It was also shown that the *Mean Aggregate Access* (MAD) policy that selects to broadcast next the item whose access probability × the interval since its last broadcast is the highest achieves close to optimal response time [17]. Along these lines, a practical skewed push broadcast organization is that of *broadcast disks* [1]. Items are assigned to virtual disks with different "speeds" based on their popularity with popular items being assigned to fast disks. The spin speed of each disk is simulated by the frequency with which the items assigned to it are broadcast. For example, the fact that a disk D_1 is three times faster than a disk D_2, means that items assigned to D_1 are broadcast three times as often as those assigned to D_2. To achieve this, each disk is split into smaller equal-sized units called chunks, where the number of chunks per disk is inversely proportional to the relative frequence of the disk. The broadcast program is generated by broadcasting one chunk from each disk and cycling through all the chunks sequentially over all the disks.

Indexing

To reduce energy consumption, a mobile device may switch to *doze* or *sleep* mode when inactive. Thus, research in wireless broadcast also considers reducing the *tuning time* defined as the amount of time a mobile client remains active listening to the broadcast. This is achieved by including index entries in the broadcast so that by reading them, the client can determine when to tune in next to access the actual data of interest. Adding index entries increases the size of the broadcast and thus may increase access time. The objective is to develop methods for allocating index entries together with data entries on the broadcast channel so that both access and tuning time are optimized. In *(1, m) indexing* [18], an index for all data items is broadcast following every fraction $(1/m)$ of the broadcast data items. *Distributed indexing* [18] improves over this method by instead of replicating the whole index m times, each index segment describes only the data items that follow it. Following the same principles, different indexing schemes have been proposed that support different query types or offer different trade-offs between access and tuning time. Finally, instead of broadcasting an index, *hashing-based techniques* have also been applied.

Data Caching and Replication

A mobile computing device (such as a portable computer or cellular phone) is connected to the rest of the network through a wireless link. Wireless communication has a double impact on the mobile device since the limited bandwidth of wireless links increases the response times for accessing remote data from a mobile host and transmitting as well as receiving of data are high energy consumption operations. The principal goal of caching and replication is to store appropriate pieces of data locally at the mobile device so that it can operate on its own data, thus reducing the need for communication that consumes both energy and bandwidth. Several cost-based caching policies along the principles of *greedy-dual* ones have been proposed that consider energy cost.

In the case of broadcast push, the broadcast itself can be viewed as a "cache in the air." Hence, in contrast to traditional policies, performance can be improved by clients caching those items that are accessed frequently by them but are not popular enough among all clients to be broadcast frequently. For instance, a cost-based cache replacement policy selects as a victim the page with the lowest p/x value, where p is the local access probability of the page and x its broadcast frequency [1]. Prefetching can also be performed with low overhead, since data items are broadcast anyway. A simple prefetch heuristic evaluates the worth of each page on the broadcast to determine whether it is more valuable than some other page in cache and if so, it swaps the cache page with the broadcast one.

Replication is also deployed to support *disconnected operation* that refers to the autonomous operation of a mobile client, when network connectivity becomes either unavailable (for instance, due to physical constraints), or undesirable (for example, for reducing power consumption). Preloading or prefetching data to sustain a forthcoming disconnection is often termed *hoarding*. Optimistic approaches to consistency control are typically deployed that allow data to be accessed concurrently at multiple sites without a priori synchronization between the sites, potentially resulting in short term inconsistencies. At some point, operations performed at the mobile device must be synchronized with operations performed at other sites. Synchronization depends on the level at which correctness is sought. This can be roughly categorized as replica-level correctness and transaction-level correctness. At the *replica level*, correctness or coherency requirements are

expressed per item in terms of the allowable divergence among the values of the copies of each item. At the *transaction level*, the strictest form of correctness is achieved through global serializability that requires the execution of all transactions running at mobile and stationary hosts to be equivalent to some serial execution of the same transactions. With regards to update propagation with *eager replication*, all copies of an item are synchronized within a single transaction, whereas with *lazy replication*, transactions for keeping replica coherent execute as separate, independent database transactions after the original transaction commits.

Common characteristics of protocols for consistency in mobile computing include:

- The propagation of updates performed at the mobile site follows in general lazy protocols.
- Reads are allowed at the local data, while updates of local data are tentative in the sense that they need to be further validated before commitment.
- For integrating operations at the mobile hosts with transactions at other sites, in the case of replica-level consistency, copies of each item are reconciled following some conflict resolution protocol. At the transaction-level, local transactions are validated against some application or system level criterion. If the criterion is met, the transaction is committed. Otherwise, the execution of the transaction is either aborted, reconciled or compensated.

Representative approaches along these lines include *isolation-only transactions* in Coda, *mobile open-nested transactions* [6], *two-tier replications* [8], *two-layer transactions* [10] and *Bayou* [9].

When local copies are read-only, a central issue is the design of efficient protocols for disseminating server updates to mobile clients. A server is called *stateful*, if it maintains information about its clients and the content of their caches and *stateless* otherwise. A server may use broadcasting to efficiently propagate update reports to all of its clients. Such update reports vary on the type of information they convey to the clients, for instance, they may include just the identifiers of the updated items or the updated values themselves. They may also provide information for individual items or aggregate information for sets of items. Update propagation may be either synchronous or asynchronous. In *asynchronous* methods, update reports are broadcast as the updates are performed. In *synchronous* methods, the server broadcasts an update report periodically. A client must listen for the report first to decide whether its cache is valid or not. This adds some latency to query processing, however, each client needs only tune in periodically to read the report. The efficiency of update dissemination protocols for clients with different connectivity behavior, such as for workaholics (i.e., often connected clients) and sleepers (i.e., often disconnected clients), is evaluated in [4].

Finally, in the case of broadcast push-data delivery, clients may read items from different broadcast programs. The *currency* of the set of data items read by each client can be characterized based on the current values of the corresponding items at the server and on the temporal discrepancy among the values of the items in the set [13]. A more strict notion of correctness may be achieved through transaction-level correctness by requiring the client read-only transactions to be serializable with the server transactions. Methods for doing so include: (i) an *invalidation* method [12], where the server broadcasts an invalidation report that includes the data items that have been updated since the broadcast of the previous report, and transactions that have read obsolete items are aborted, (ii) *serialization graph testing* (SGT) [12], where the server broadcasts control information related to conflicting operations, and (iii) *multiversion broadcast* [11], where multiple versions of each item are broadcast, so that client transactions always read a consistent database snapshot.

Key Applications

Data broadcasting, caching and replication techniques are part of the core of any application that requires data sharing and synchronization among mobile devices and data servers. Such applications include vehicle dispatching, object tracking, points of sale (e.g., ambulance and taxi services, Fedex/UPS), and collaborative applications (e.g., homecare, video gaming). They are also part of embedded or light versions of database management systems that extend enterprise applications to mobile devices. These include among others Sybase Inc.'s SQL Anywhere, IBM's DB2 Everyplace, Microsoft SQL Server Compact, Oracle9i Lite and SQL Anywhere Technologies' Ultralite.

Cross-references

▶ Concurrency Control
▶ Hash-Based Indexing

► MANET Databases
► Mobile Database
► Replicated Database Concurrency Control
► Transaction Management

Recommended Reading

1. Acharya S., Alonso R., Franklin M.J., and Zdonik S.B. Broadcast disks: data management for asymmetric communications environments. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1995, pp. 199–210.
2. Acharya S., Franklin M.J., and Zdonik S.B. Balancing push and pull for data broadcast. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1997, pp. 183–194.
3. Aksoy D. and Franklin M.J. RxW: a scheduling approach for large scale on-demand broadcast. IEEE/ACM Trans. Netw., 7(6):846–860, 1999.
4. Barbará D. and Imielinski T. Sleepers and workaholics: caching strategies in mobile environments. VLDB J., 4(4):567–602, 1995.
5. Beaver J., Chrysanthis P.K., and Pruhs K. To broadcast push or not and what? In Proc. 7th Int. Conf. on Mobile Data Management, 2006, pp. 40–45.
6. Chrysanthis P.K. Transaction processing in a mobile computing environment. In Proc. IEEE Workshop on Advances in Parallel and Distributed Systems, 1993, pp. 77–82.
7. Dykeman H.D., Ammar M.H., and Wong J.W. Scheduling algorithms for videotex systems under broadcast delivery. In Proc. IEEE Int. Conf. on Communications, 1986, pp. 1847–1851.
8. Gray J., Helland P., Neil P.O., and Shasha D. The dangers of replication and a solution. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1996, pp. 173–182.
9. Petersen K., Spreitzer M., Terry D.B. Theimer M., and Demers A.J. Flexible update propagation for weakly consistent replication. In Proc. 16th ACM Symp. on Operating System Principles, 1997, pp. 288–301.
10. Pitoura E. and Bhargava B. Data consistency in intermittently connected distributed systems. IEEE Trans. Knowl. Data Eng., 11(6):896–915, 1999.
11. Pitoura E. and Chrysanthis P.K. Exploiting versions for handling updates in broadcast disks. In Proc. 25th Int. Conf. on Very Large Data Bases, 1999, pp. 114–125.
12. Pitoura E. and Chrysanthis P.K. Scalable processing of read-only transactions in broadcast push. In Proc. 19th Int. Conf. on Distributed Computing Systems, 1999, pp. 432–439.
13. Pitoura E., Chrysanthis P.K., and Ramamritham K. Characterizing the temporal and semantic coherency of broadcast-based data dissemination. In Proc. 9th Int. Conf. on Database Theory, 2003, pp. 410–424.
14. Pitoura E. and Samaras G. Data Management for Mobile Computing. Kluwer, Boston, USA, 1998.
15. Sharaf MA. and Chrysanthis P.K. On-demand data broadcasting for mobile decision making. MONET, 9(6):703–714, 2004.
16. Stathatos K., Roussopoulos N., and Baras J.S. Adaptive data broadcast in hybrid networks. In Proc. 23th Int. Conf. on Very Large Data Bases, 1997, pp. 326–335.
17. Su C.J, Tassiulas L., and Tsotras V.J. Broadcast scheduling for information distribution. Wireless Netw., 5(2):137–147, 1999.
18. T I., Viswanathan S., and Badrinath B.R. Data on air: organization and access. IEEE Trans. Knowl. Data Eng., 9(3):353–372, 1997.

Data Cache

► Processor Cache

Data Cleaning

Venkatesh Ganti
Microsoft Research, Redmond, WA, USA

Definition

Owing to differences in conventions between the external sources and the target data warehouse as well as due to a variety of errors, data from external sources may not conform to the standards and requirements at the data warehouse. Therefore, data has to be transformed and cleaned before it is loaded into a data warehouse so that downstream data analysis is reliable and accurate. *Data Cleaning* is the process of standardizing data representation and eliminating errors in data. The data cleaning process often involves one or more tasks each of which is important on its own. Each of these tasks addresses a part of the overall data cleaning problem. In addition to tasks which focus on transforming and modifying data, the problem of diagnosing quality of data in a database is important. This diagnosis process, often called data profiling, can usually identify data quality issues and whether or not the data cleaning process is meeting its goals.

Historical Background

Many business intelligence applications are enabled by data warehouses. If the quality of data in a data warehouse is poor, then conclusions drawn from business data analysis could also be incorrect. Therefore, much emphasis is placed on cleaning and maintaining high quality of data in data warehouses. Consequently, the area of data cleaning received considerable attention in the database community. An early survey of automatic data cleaning techniques can be found in [14]. Several companies also started developing domain-specific data cleaning solutions (especially for the customer address domain). Over time, several generic data cleaning techniques have been also been