A Survey of Mobile Database Caching Strategies

Earl Oliver
David R. Cheriton School of Computer Science
University of Waterloo
eaoliver@uwaterloo.ca

ABSTRACT

The continued growth of portable, wireless enabled devices with massive storage capacity and powerful CPUs are making the wide spread use of mobile databases a reality. Mobile devices are increasingly used for database driven applications such as product inventory tracking, sales order entry, and customer relationship management. These applications have changed the way mobile applications access and manage data. Instead of storing data in a central database, data is being moved closer to applications to increase efficiency and autonomy. This trend leads to many interesting problems in mobile database research. In this paper I survey recent research in mobile database caching strategies. I overview systems built using avoidance-based approaches to cache invalidation. With each system I describe incremental improvements over previous work. I also briefly survey recent related work on conflict avoidance schemes.

1. INTRODUCTION

Everyday mobile devices are becoming cheaper, more powerful, storing more information, and lasting longer. These devices, which include handheld devices and their larger counterpart, the laptop, are becoming an increasingly useful tools for mobile users. Connectivity to the Internet has become ubiquitous, and mobile devices are increasingly used for database driven tasks such as product inventory tracking, sales order entry, and customer relationship management. These applications have changed the way mobile applications access and manage data. Instead of storing data in a central location, i.e. on a server on the Internet, data is being moved closer to applications. This enables data processing to be performed more efficiently and autonomously.

While mobile database systems are commonly regarded as a form of distributed system, there are several characteristics that uniquely constrain solutions to the problem.

- **Limited Resources:** The CPU power, memory, and persistent storage of mobile devices is continuously increasing; however, they still significantly lag behind non-mobile systems such as servers on the Internet. Depending on the sized of the database, limited CPU, memory capacity could prevent a mobile device performing even simple operations on local data. Limited storage capacity also makes it commonly infeasible to cache entire databases to a mobile devices.
- **Limited energy:** Mobile devices have a limited amount of energy. CPU intensive operations, disk I/O, and network connectivity all have an energy cost that must be minimized in a mobile database system.
- **Disconnection:** Mobile devices may roam in and out of network connectivity and experience variable bandwidth in different geographical locations. Interaction between a mobile device and a database is directly affected by the device's network connectivity.
- **Bandwidth asymmetry:** Wireless bandwidth in the downstream direction is often much greater than bandwidth in the upstream direction.

We begin by considering the standard design of a mobile database system. Mobile devices (clients) containing data-centric applications roam between wireless cells and access a centralized database(s). The cell could be cellular network coverage or a wireless local area network. A common goal in mobile database systems is to minimize the amount of data transmitted over the wireless link between the server to the mobile client [4].

Today, it is common for mobile database applications to query a central database directly. Direct (synchronous) calls to a database guarantee a consistent up-to-date view of the database; however, they require full connectivity, constrain mobility, and drain the battery extraneously. As have seen in [32], high data rate wireless access using as EVDO or 802.11 are fast enough to allow for synchronous database access for current mobile workloads. In [32], Tolia defines the *MobileSales* benchmark based on TCP-App that represents the workload of a travelling sales person entering orders, creating customers, etc. Using this workload, the average latency was less than one second over an EVDO connection. For current workloads, many of today's wireless networks are fast enough to provide reasonably fast synchronous access, but mobile clients must still minimize wireless communication to save both energy, and when transmitting over cellular networks, money.

Querying the database directly is a common, robust method for accessing databases on the Internet. As high-speed wireless networks continue to proliferate, the limitations of bandwidth asymmetry and disconnection have a diminishing effect on mobile database design. However, the bulk of mobile database research continues to consider all four of the above device characteristics. The common assumption is that accessing central databases for every operations can be prohibitively expensive and can significantly impact that response time and throughput of a mobile application. Caching data and then performing local queries on the cache has become a common use case. Caching increases the performance of local queries, but comes at a cost of complicated cache maintenance strategies.

The above characteristics coupled with a growth in data-centric applications on the devices leads to an old research challenge. *How do we maintain consistency of cached data on a resource constrained, nomadic, periodically disconnected device, while minimizing power consumption?* This paper surveys answers solutions to this classic problem.

This paper is organized as follows. In the next section I provide background to mobile database research by outlining previous surveys in the area. In Section 3 I survey recent, widely cited research in avoidance-based mobile database caching. I conclude in Section 4.

2. PREVIOUS WORK

The topic of mobile database systems was introduced by Imielinski and Badrinath in 1993 in [14] and [15]. This paper categorizes new research issues introduced by mobile computing into three areas: mobility, disconnection, and new data access modes.

Mobility of database users created problems surrounding location dependent queries. For example, a query of local hospital locations could explicitly or implicitly consider the current location. Mobility also lead to problems within dynamic replication schemes. To maintain performance transparency, data can be replicated on the basis of changing "centers of activity". In practice, the challenge of relocating data closer to a mobile user did not take off in the database community. The bottleneck to mobile database operations continues to be the wireless link and the database itself, not the backhaul connection.

Disconnection from a master database introduces the problem of cache consistency. The trade off between connectivity and cache consistency has been the foundation of many mobile database systems. Strongly connected mobile users may afford strict consistency, while weak connectivity requires a more flexible approach of weak consistency. Hand off and recovery was also introduced as a challenge area with many open questions. What should happen if a mobile switches off or loses connectivity in the middle of a transaction? What is the appropriate transaction model or concurrency control scheme? What trade offs between connectivity, power consumption, and resources on the mobile device should be considered when interacting with a database system? As we will see, these questions are still explored in today's mobile database research.

Issues surrounding new access methods were motivated by the wireless transmission medium and by power restrictions of mobile devices. This area considers trade offs between remote and local execution of queries. For example, transmitting data over a wireless medium consumes an order of magnitude more power than receiving [15]; however, the power cost can be mitigated by faster execution on a remote server and less power consumed during local processing.

In [1], Alonso and Korth introduce other issues relevant to mobile database systems: query optimization and transaction models. They hypothesize that wireless networks will be heterogeneous, with varying monetary and energy costs. Query optimization on mobile devices would need to consider monetary network access costs as well as energy consumption. Fault tolerance due to frequent disconnection was predicted to be a major issue in mobile databases. Servers would need to seamlessly take over a transaction when a mobile disconnects from a database server. Disconnection prediction would need to be exploited to download data that may be needed (proactive cache injection).

In 1995, Dunham and Helal [7] argue that data management in mobile computing environments is similar to that of a distributed system. They present open challenges in mobile database systems and argue that many of the research challenges are the same; they simply have different solutions. Through comparison of distributed and mobile data management systems they conclude by categorizing mobile computing database environments as a mobile heterogeneous multi-database system.

The most recent mobile database survey was in 1999 by Barbará [3]. This paper surveys each distinctive feature of mobile computing and how they impact the implementation of databases for mobile computers.

Barbará outlines several key challenge areas remaining in mobile database research: prototyping, effective bandwidth utilization, transactional properties, optimization of location dependent query processing, and data visualization. Since Barbará's 1999 survey the body of mobile computing research has grown dramatically. Mobile database caching is a prime research area within the challenge area of effective bandwidth utilization.

3. MOBILE DATABASE CACHING

Conventional database caching designs assume both strong connectivity [9] and symmetric bandwidth. In these systems, caching is used primarily as a performance optimization. In mobile environments, where network disconnection and asymmetric bandwidth is a standard mode of operation, caching is essential to maintain service. Mobile database literature has well proven that caching of frequently accessed items is a useful technique for reducing wireless bandwidth requirements and coping with disconnection [4, 6, 8, 16, 36]. However, the use of mobile caching systems raises the challenge of ensuring that transaction semantics are not violated as a result of the creation and destruction of cached data. In [10] these algorithms are defined as *transactional cache consistency maintenance algorithms*. These algorithms can be divided into two categories [10]:

- **Avoidance-based:** Servers either send invalidation messages to signal that a cached item must be updated or, depending on usage patterns, send the update itself.

- **Detection-based:** Clients send queries to the database server to validate cached data.

Avoidance-based approaches have largely dominated mobile database literature. In this survey I will focus on this approach only; however, it is easy to see that many avoidance-based techniques can be modified to fit a detection-based scheme. Most of the caching scheme discussed ignore the problem of transaction conflicts that arise naturally when utilizing a local cache. In 3.2 I briefly examine recent cache conflict avoidance schemes in mobile databases.

3.1 Avoidance-based Approach

Conventional avoidance-based approaches are designed to enforce consistency by making it impossible for transactions to ever access stale data. Database servers enforce consistency by directly manipulating the contents of clients' caches. One way to accomplish this is through the use of broadcasting of updates [24, 27], invalidation messages, and invalidation reports [15, 28, 4][1].

Naively broadcasting database updates to mobile clients [24, 27] is widely considered unpractical and not considered in recent literature. Invalidation messages are sent to clients caching particular data. This approach requires that the server maintain the state of each client. When clients become disconnect, upon reconnection they must connect to the server to receive any pending invalidation messages. This approach is efficient from the perspective of client cache consistency and minimizes communication between client and server; however, maintaining the state of each client's database cache does not scale well.

The broadcasting of invalidation reports is designed to solve the scalability problem. Invalidation reports accommodate occasional disconnection by mobile clients and contain aggregate information about data items that have changed. Reports remove the need for the server to maintain the state of each client. Reports are broadcast to all clients and the report itself represents the state of data that could be cached at each client. When receiving an invalidation broadcasts, mobile clients compare the set of invalidated objects to the objects in its cache. Invalidated objects may be evicted or refreshed (if needed). Generally, each invalidation report carries a timestamp from its issuing service. The validity of a client's cache is therefore valid as of the last invalidation report received. The simplicity of invalidation reports has been adopted in many recent papers [16, 19, 34]. Based on the timing of the reports being sent, broadcast techniques come in two forms:

[1]The use of the word *broadcast* is loosely defined in mobile database caching literature. Literature predating the year 2000 commonly refer to broadcasts over the wireless channel. In practice, many networks do not support broadcasts (flooding). In recent literature, broadcast are generally a server initiated push of information as either as network unicast or multicast.

- **Synchronous:** In a synchronous approach [4, 33, 35] cache invalidation reports are broadcast periodically. Synchronous servers aggregate updates over a fixed period of time and then broadcast these invalidation reports with the time of the updates all in one message. The draw back to this approach is that latency is introduced between the time of the update and the invalidation broadcast.

- **Asynchronous:** In an asynchronous approach [17, 21, 33, 34] the database server broadcasts an invalidation report as soon as an item in the database changes. The biggest problem with this approach is the unpredictable amount of time a client must wait for an invalidation message.

There are many trade offs when designing systems that use invalidation reports. These trade offs are discussed in detail in [4] that served to motivate subsequent research. One obvious research direction was to minimize the size of the invalidation report. Yuen et al. in [38] propose the invalidation scheme, *Invalidation by Absolute Validity Interval* (IAVI). This method exploits pre-defined patterns in how data changes. The predicted period of validity of a data item is defined to be the *Absolute Validity Interval* (AVI). The AVI is distributed to mobile clients as an invalidation. A mobile client can verify the validity of a cached data item by comparing the last update time with the AVI. The client invalidates items in its cache if the current time exceeds the validity period of the AVI. The use of AVIs allows mobile clients to estimate the validity of its cache during periods of disconnection. For example, suppose a database is known to update hourly with new price information. AVIs on data allows a mobile sales agent that is disconnected between updates to query its database cache with a reasonable certainty that the information is valid. When the mobile client is connected, the database server sends invalidation reports to inform clients of changes to AVIs rather than data item itself. Performance studies show that the use of IAVI significantly reduces the mean response time and invalidation report size over previous bit-sequence [16] and timestamp [4] mobile cache invalidation schemes.

A key characteristic that should be considered in a mobile caching scheme systems is power consumption. While it is understood that techniques that minimize wireless communication prolong battery life, the magnitude of savings is rarely quantified in mobile database literature. Wu et al. in [37] was one of the first papers to consider power consumption on mobile cache invalidation. Their invalidation scheme is called *Grouping with cold update-set retention* (GCORE). This scheme also uses invalidation reports to inform clients of updated objects in the database. This scheme differs from previous approaches in that it salvages as many cached objects as possible after a reconnection, if they are still valid. Upon reconnection, GCORE invalidates its cache based on the first broadcast invalidation report it receives. GCORE then checks the validity of the remaining items by querying the database server[2].

[2]It should be noted that this approach implies that GCORE utilizes both the avoidance and detection-based approaches.

The power optimization in GCORE comes from reducing the communication overhead from client to server when validating its cache and from the server to clients when broadcasting updates. This reduction is achieved through grouping of objects. Each group within GCORE maintains two sets of objects: the hot update set and the cold update set. The hot update set represents objects that are frequently updated by transactions on the server. The cold update set contains objects that are infrequently updated. In GCORE, when a mobile client attempts to validate its cache, instead of invalidating an entire group, the cold update set of objects is retained if all of the updated objects (most likely objects in the hot set) had been included in the most recently broadcast invalidation report. The GCORE server maintains a simple group update history that dynamically excludes the hot update set from response to the client's cache validation request. By efficiently retaining the cold update sets of groups on a mobile client, GCORE requires less energy consumption than previous techniques.

Tan outlines another novel energy optimization in [30]. In this paper he presents an improved bit-sequence scheme that allows clients to listen only the portion of the report that is of interest to them. Ignoring invalidation reports known to be irrelevant allows a mobile device to disable its wireless antenna and thus save power. Unfortunately, this approach relies on strict time synchronization between the mobile device and the broadcasting server and interferes with other network applications on the mobile device. The approach is novel, but difficult to practically deploy.

In [20], Lee et al. present *Opportunistic Concurrency Control with Update Timestamp* (OCC-UTS). OCC-UTS again uses the wireless broadcast medium to propagate invalidation reports to mobile clients. In OCC-UTS, cache consistency checks differ from previous techniques. In [4, 36] read operations on the database cache were deferred. Applications were blocked on the database operation until the next broadcast is received and the consistency of the cache can be verified. This technique introduces serializability issues, which are not discussed in this survey; however, it is clear that applications accessing the mobile database would have the freshest data available. The major drawback to [4, 36] is that blocking on the next invalidation broadcast or until the device enters connectivity loses the benefit of client caching (especially if the client's cache is valid). In OCC-UTS, mobile database queries are returned immediately if the requested data is in the cache. If the data is not cached, OCC-UTS connects to the central database to fetch the requested data. In the cached case, the returned may be stale. OCC-UTS relies on the fact that conflicts derived from stale data will be detected on the client when an invalidation report is received, or by the server when the mobile client attempts to commit an update.

If the mobile transaction is an update that needs to commit, the client sends a request to commit to the server. OCC-UTS uses optimistic concurrency control to determine if a transaction is serializable. When a mobile wants to commit a transaction, instead of purely comparing the read sets with the write sets of other transactions[3], the OCC-UTS server compares timestamps on each mobile update with the last update stored on the server to determine of the transaction is serializable. If the transaction is serializable, then it is committed; otherwise the transaction is aborted at the client. In terms of transaction performance, OCC-UTS is compared to WoundCertifier [2] (the state of the art in 1999). OCC-UTS has higher transaction throughput and significantly less wireless communication than WoundCertifier. I will revisit WoundCertifier as a conflict avoidance scheme in 3.2.

We end this topic with the most recent work by Su et al. [29] that defines a technique for composing near optimal invalidation reports using bit-sequence algorithms [16]. This work follows earlier work by Hou et al. on optimal invalidation report construction [13] and is based on the bit-sequence approach presented in [11]. Two techniques are commonly used to reduce the size of an invalidation report: bit-sequence mapping and update aggregation. Update aggregation is often used in conjunction with bit-sequence algorithms to reduce the number of timestamps in a report. Instead of having one timestamp for each updated item, the report uses only one timestamp, which is the time when the report is to be delivered. Update aggregation is successful in reducing the size of the report; however, the accuracy of the report is reduced and valid items count be falsely invalidated. This paper considers the problem of false invalidation on cached items when there is insufficient time detail in the invalidation report. In this approach they exploit clients' disconnection patterns (i.e. the distribution of the time spans between disconnection and reconnection) to reduce the rate of false invalidations. They analyze the relationship between false invalidations and reconnection patterns and formulate a technique that reorganizes the hierarchy of bit-sequences to take into account clients' reconnection patterns. Through simulation they show that their technique has significant improvements over [13], but in practice the technique has an enormous overhead and does not scale to many many clients.

3.2 Conflict Avoidance

Actively maintaining a database cache improves local application performance when performing read only operations. However, the complexity of maintaining a mobile database cache increases dramatically when clients are allowed to perform updates. The naive approach to the database update problem is to forward all updates directly to the central database. This restricts database updates to when the device is within wireless coverage. We require a means for mobile devices to perform independent operations on the database.

An approach that nicely illustrates the trade off between mobile computing characteristics is Barbará's *WoundCertifier* protocol [2]. The WoundCertifier protocol is a slight variation of the invalidation report caching schemes described in 3.1. This work follows from [4]. In the WoundCertifier protocol, the database server periodically broadcasts certification reports over the wireless channel. Certification reports contain a list of read and write sets of current active transactions that have requested to commit and have been "certified" by the database server. When a mobile client receives an invalidation report it checks if any of its current

[3]The protocol for pure backward validation [12].

transactions conflict the read and write sets of the certification report. A major advantage of WoundCertifier over previous cache schemes is that mobile clients do not commit a transaction unless it is guaranteed to commit. If a transaction on the mobile client's read set intersects with a committing transaction's write set, then the mobile client can abort the transaction itself.

WoundCertifier exploits excess CPU and memory resources on the mobile to offload workload from the database server. Only when mobile clients cannot self verify that a local transaction will commit will they contact the database server. WoundCertifier therefore scales to a much larger pool of concurrent clients at the cost of increased wireless communication pushing certification reports to clients. In this paper it is shown that for write intensive workloads, WoundCertifier reduces communication between the client and the server by as much as 90% by detecting conflicts at the mobile client.

Another common approach to share data efficiently over a wide area of mobile environment with less communication overhead is optimistic replication [26]. In [31] mobile clients each maintain a local copy of the database[4]. Each client makes modifications to its local replicated copy of the database. During periods of connectivity updates are integrated into the common database. The reconciliation process may result in a different final database state that the mobile clients cannot independently determine. Mobile clients may be required to abort transactions that had previously been committed during local transactions. Although this technique is proven to work reliably, aborting a mobile transaction after the user believes it is committed is generally not feasible.

Phatak and Badrinath offer a preliminary solution to the mobile-client-server reconciliation problem [23]. Their approach is based on multiversioning of mobile replicas. Multiversioning allows them to use snapshot isolation, (as explained in [5]), to commit multiple transactions as long as the data items in its write set are not overwritten after these data items were read by the transaction. Their technique is also shown to increase the probability of reconciliation. This method makes an incremental contribution over previous work; however, if the preconditions of the read and write dataset are not met, a condition that the mobile database cannot control, the mobile transaction would still be required to abort.

In [25], Preguiça et al. take an alternative approach. In their scheme, portions of a central database can only be modified if a mobile client holds a reservation. They present a middleware library called *Mobisnap* that allows mobile clients to independently guarantee that their updates will be executed on a central database server without conflicts. Mobisnap operates by caching a database snapshot on the mobile device. During periods of disconnections, Mobisnap tentatively applies mobile transactions to the local database cache. Applications running on mobile devices submit updates to the central database by sending it small PL/SQL "mobile transactions". When the client enters network coverage, these transactions are propagated to the server and re-executed against the shared database. Local tentative transactions are considered final if the mobile has acquired enough reservations from the database server. Mobile clients obtain reservations from the server before disconnecting. A reservation is given by the server to allow a mobile client specific state altering operations on the database. Reservations allow mobile clients to guarantee the final state of the database independently because reservations guarantee that no conflict will arise when the mobile transaction is re-executed on the central database server. Unlike previous systems [22, 18], verification of available reservations is performed transparently to the application. No special function is required to access or modify reserved data.

[4]The literature commonly considers the replica a full replica; however, a partial replica of the database could be used on a storage constrained device.

4. CONCLUSION

As the technology underlying today's mobile platforms continues to improve, the everyday use of mobile databases will surely increase. Although wireless networks continue to increase in speed, efficient and robust mobile database caching will continue to be an essential component of any mobile database system.

In this paper I have surveyed the most widely cited work in mobile database caching strategies. The amount of research in this area has become enormous. This survey only skims the surface of what has become, and will continue to be, a large and fertile area of research.

5. REFERENCES

[1] Rafael Alonso and Henry Korth. Database systems issues in nomadic computing. In *Proceedings of ACM SIGMOD Conference*, page 388, Washington, DC, May 1993.

[2] D. Barbará. Certification reports: supporting transactions in wireless systems. In *ICDCS '97: Proceedings of the 17th International Conference on Distributed Computing Systems (ICDCS '97)*, page 466, Washington, DC, USA, 1997. IEEE Computer Society.

[3] Daniel Barbará. Mobile computing and databases-a survey. *IEEE Transactions on Knowledge and Data Engineering*, 11(1):108–117, 1999.

[4] Daniel Barbará and Tomasz Imieliński. Sleepers and workaholics: caching strategies in mobile environments. In *SIGMOD '94: Proceedings of the 1994 ACM SIGMOD international conference on Management of data*, pages 1–12, New York, NY, USA, 1994. ACM Press.

[5] Hal Berenson, Phil Bernstein, Jim Gray, Jim Melton, Elizabeth O'Neil, and Patrick O'Neil. A critique of ansi sql isolation levels. In *SIGMOD '95: Proceedings of the 1995 ACM SIGMOD international conference on Management of data*, pages 1–10, New York, NY, USA, 1995. ACM Press.

[6] J. Cai, Kian-Lee Tan, and Beng Chin Ooi. On incremental cache coherency schemes in mobile computing environments. In *ICDE '97: Proceedings of the Thirteenth International Conference on Data Engineering*, pages 114–123, Washington, DC, USA, 1997. IEEE Computer Society.

[7] Margaret H. Dunham and Abdelsalam Helal. Mobile

computing and databases: anything new? *SIGMOD Rec.*, 24(4):5–9, 1995.

[8] Cedric C. F. Fong, John C. S. Lui, and Man Hon Wong. Quantifying complexity and performance gains of distributed caching in a wireless mobile computing environment. In *ICDE '97: Proceedings of the Thirteenth International Conference on Data Engineering*, pages 104–113, Washington, DC, USA, 1997. IEEE Computer Society.

[9] Michael J. Franklin, Michael J. Carey, and Miron Livny. Global memory management in client-server database architectures. In *VLDB '92: Proceedings of the 18th International Conference on Very Large Data Bases*, pages 596–609, San Francisco, CA, USA, 1992. Morgan Kaufmann Publishers Inc.

[10] Michael J. Franklin, Michael J. Carey, and Miron Livny. Transactional client-server cache consistency: alternatives and performance. *ACM Trans. Database Syst.*, 22(3):315–363, 1997.

[11] C. Gray and D. Cheriton. Leases: an efficient fault-tolerant mechanism for distributed file cache consistency. *SIGOPS Oper. Syst. Rev.*, 23(5):202–210, 1989.

[12] Theo Härder. Observations on optimistic concurrency control schemes. *Inf. Syst.*, 9(2):111–120, 1984.

[13] Wen-Chi Hou, Meng Su, Hongyan Zhang, and Hong Wang. An optimal construction of invalidation reports for mobile databases. In *CIKM '01: Proceedings of the tenth international conference on Information and knowledge management*, pages 458–465, New York, NY, USA, 2001. ACM Press.

[14] Tomasz Imieĺinski and B. R. Badrinath. Data management for mobile computing. *SIGMOD Rec.*, 22:34–39, 1993.

[15] Tomasz Imielinski and B. R. Badrinath. Mobile wireless computing: challenges in data management. *Commun. ACM*, 37(10):18–28, 1994.

[16] Jin Jing, Ahmed Elmagarmid, Abdelsalam Sumi Helal, and Rafael Alonso. Bit-sequences: an adaptive cache invalidation method in mobile client/server environments. *Mob. Netw. Appl.*, 2(2):115–127, 1997.

[17] Anurag Kahol, Sumit Khurana, Sandeep K.S. Gupta, and Pradip K. Srimani. A strategy to manage cache consistency in a disconnected distributed environment. *IEEE Trans. Parallel Distrib. Syst.*, 12(7):686–700, 2001.

[18] Narayanan Krishnakumar and Ravi Jain. Escrow techniques for mobile sales and inventory applications. *Wirel. Netw.*, 3(3):235–246, 1997.

[19] Kwong Yuen Lai, Zahir Tari, and Peter Bertok. Cost efficient broadcast based cache invalidation for mobile environments. In *SAC '03: Proceedings of the 2003 ACM symposium on Applied computing*, pages 871–877, New York, NY, USA, 2003. ACM Press.

[20] SangKeun Lee, Chong-Sun Hwang, and HeongChang Yu. Supporting transactional cache consistency in mobile database systems. In *MobiDe '99: Proceedings of the 1st ACM international workshop on Data engineering for wireless and mobile access*, pages 6–13, New York, NY, USA, 1999. ACM Press.

[21] Pavan Nuggehalli, Vikram Srinivasan, and Carla-Fabiana Chiasserini. Energy-efficient caching strategies in ad hoc wireless networks. In *MobiHoc '03: Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, pages 25–34, New York, NY, USA, 2003. ACM Press.

[22] Patrick E. O'Neil. The escrow transactional method. *ACM Trans. Database Syst.*, 11(4):405–430, 1986.

[23] Shirish H. Phatak and B. R. Badrinath. Multiversion reconciliation for mobile databases. *icde*, 00:582, 1999.

[24] Evaggelia Pitoura. Supporting read-only transactions in wireless broadcasting. In *DEXA '98: Proceedings of the 9th International Workshop on Database and Expert Systems Applications*, page 428, Washington, DC, USA, 1998. IEEE Computer Society.

[25] Nuno Preguiça, J. Legatheaux Martins, Miguel Cunha, and Henrique Domingos. Reservations for conflict avoidance in a mobile database system. In *MobiSys '03: Proceedings of the 1st international conference on Mobile systems, applications and services*, pages 43–56, New York, NY, USA, 2003. ACM Press.

[26] Yasushi Saito and Marc Shapiro. Optimistic replication. *ACM Comput. Surv.*, 37(1):42–81, 2005.

[27] Jayavel Shanmugasundaram, Arvind Nithrakashyap, Rajendran Sivasankaran, and Krithi Ramamritham. Efficient concurrency control for broadcast environments. In *SIGMOD '99: Proceedings of the 1999 ACM SIGMOD international conference on Management of data*, pages 85–96, New York, NY, USA, 1999. ACM Press.

[28] Antonio Si and Hong Va Leong. Query processing and optimization for broadcast database. In *DEXA '96: Proceedings of the 7th International Conference on Database and Expert Systems Applications*, pages 899–914, London, UK, 1996. Springer-Verlag.

[29] Meng Su, Chih-Fang Wang, and Wen-Chi Hou. An approach of composing near optimal invalidation reports. In *MDM '05: Proceedings of the 6th international conference on Mobile data management*, pages 116–124, New York, NY, USA, 2005. ACM Press.

[30] Kian-Lee Tan. Organization of invalidation reports for energy-efficient cache invalidation in mobile environments. *Mob. Netw. Appl.*, 6(3):279–290, 2001.

[31] D. B. Terry, M. M. Theimer, Karin Petersen, A. J. Demers, M. J. Spreitzer, and C. H. Hauser. Managing update conflicts in bayou, a weakly connected replicated storage system. In *SOSP '95: Proceedings of the fifteenth ACM symposium on Operating systems principles*, pages 172–182, New York, NY, USA, 1995. ACM Press.

[32] Niraj Tolia, M. Satyanarayanan, and Adam Wolbach. Improving mobile database access over wide-area networks without degrading consistency. In *MobiSys '07: Proceedings of the 5th international conference on Mobile systems, applications and services*, pages 71–84, New York, NY, USA, 2007. ACM Press.

[33] Agustinus Borgy Waluyo, Bala Srinivasan, and David Taniar. A taxonomy of broadcast indexing schemes for multi channel data dissemination in mobile databases. In *AINA '04: Proceedings of the 18th International Conference on Advanced Information Networking and Applications*, page 213, Washington, DC, USA, 2004. IEEE Computer Society.

[34] Zhijun Wang, Sajal K. Das, and Hao Che. A scalable asynchronous cache consistency scheme (saccs) for mobile environments. *IEEE Trans. Parallel Distrib. Syst.*, 15(11):983–995, 2004. Senior Member-Mohan Kumar.

[35] Kevin Wilkinson and Marie-Anne Neitmat. Maintaining consistency of client-cached data. In *Proceedings of the sixteenth international conference on Very large databases*, pages 122–134, San Francisco, CA, USA, 1990. Morgan Kaufmann Publishers Inc.

[36] Kun-Lung Wu, Philip S. Yu, and Ming-Syan Chen. Energy-efficient caching for wireless mobile computing. In *ICDE '96: Proceedings of the Twelfth International Conference on Data Engineering*, pages 336–343, Washington, DC, USA, 1996. IEEE Computer Society.

[37] Kun-Lung Wu, Philip S. Yu, and Ming-Syan Chen. Energy-efficient mobile cache invalidation. *Distrib. Parallel Databases*, 6(4):351–372, 1998.

[38] J. Chun-Hung Yuen, E. Chan, K.-Y. Lam, and H. W. Leung. An adaptive avi-based cache invalidation scheme for mobile computing systems. In *DEXA '00: Proceedings of the 11th International Workshop on Database and Expert Systems Applications*, page 155, Washington, DC, USA, 2000. IEEE Computer Society.