Caching strategies in Mobile Computing

Vijay Balakumar, M.S. in Computer Science
University of Texas at Arlington

Abstract

The tremendous growth of the mobile users' population coupled with the bandwidth requirements of new cellular services are in contrast to the limited spectrum resources that have been allocated for mobile communications. Three caching strategies have been discussed in this paper. One uses Bit sequence, an adaptive cache invalidation algorithm for client/server mobile environments; the next approach is 1) to use asynchronous invalidation messages and 2) to buffer invalidation messages from servers at the MH's Home Location Cache (HLC) while the MH is disconnected from the network and redeliver these invalidations to the MH when it gets reconnected; and the last approach is to use Data Replication for Caching. The advantages and the limitation of each paper have been identified. In the past few years, there has been a tremendous surge of research in the area of caching in mobile computing. This paper is an effort to survey the techniques and to classify this research in a few broad areas.

Index Terms – Mobile Computing, Caching, Bandwidth, Mobile Hosts, Home Location Cache, Data Replication, MU (Mobile Units), FH (Fixed Hosts).

1. Introduction

Mobile computing has become a reality thanks to the convergence of two technologies: the appearance of powerful portable computers and the development of fast reliable networks. Moreover the growth of the mobile users population coupled with the bandwidth requirements of new cellular services are in contrast to the limited spectrum resources that have been allocated for mobile communications. When developing a

mobile network's infrastructure, wasting bandwidth should be avoided because it is very costly. Thus, the objective is to try and get the most out of the minimum infrastructure.

Caching can reduce the bandwidth requirement in a wireless computing environment as well as minimize the energy consumption of wireless portable computers. Efficient caching schemes for mobile computing should take into account the following factors: data access pattern and update rates, communication/access costs, mobility pattern of the client, connectivity characteristics and location dependence of the data. The major problems in mobile computing are 1) Bandwidth considerations and data transfer rate; 2) Frequent network failures; 3) Limited battery life; 4) Wireless communication is expensive.

Among the three techniques, one uses Bit sequence - an adaptive cache invalidation algorithm for client/server mobile environments; the next approach is 1) to use asynchronous invalidation messages and 2) to buffer invalidation messages from servers at the MH's Home Location Cache (HLC) while the MH is disconnected from the network and redeliver these invalidations to the MH when it gets reconnected. This technique is called AS (Asynchronous and Stateful); and the last approach is to use Data Replication for Caching. The main purpose of this paper is to describe the three strategies mentioned above and throw the limitations of those approaches and also put forward the possible research directions on these approaches.

The rest of the paper is organized as follows: Section 2 describes the bit sequence adaptive cache invalidation algorithm. Section 3 describes the informal overview of the second approach. Section 4 describes the caching strategy using data replication. Section 5 presents the analysis of all the three schemes – their advantages, limitations and possible future research directions. Finally the conclusions are drawn in Section 6.

2. Bit – Sequence algorithm

In [4] an approach to invalidate cache was provided, where the server periodically broadcasts invalidation report in which the changed data items are indicated. Rather

than querying a server directly regarding the validation of cached copies, clients can listen to these invalidation reports over wireless channels. A major challenge for this approach is to optimize the organization of broadcast reports. In general a large report can provide more information and is more effective for cache invalidation. But a large report also implies a long latency for clients while checking the report, given a limited broadcast bandwidth.

The Broadcasting Timestamp(TS)[4] is a good example of an algorithm that limits the size of the report by broadcasting the names and timestamps only for the data items updated during a window of w seconds. Effectiveness of the report cannot be guaranteed for clients with unpredictable disconnection time. This approach presents three optimization techniques. They are:

1) For applications where cached data items are changed less often on the database server, use the bit-sequence naming technique to reference data items in the report.
2) Instead of including one update timestamp for each data item, use an update aggregation technique to group a set with only one timestamp in the report.
3) A hierarchical structure of bit-sequences technique to link a set of bit sequences so that the structure can be used by the clients with different disconnection times.

The algorithm Bit – Sequence(BS) uses these three techniques. This algorithm can be used in applications where frequently cached data items are predictable.

The bits in the sequence represent those data items in the database that are frequently cached and referenced by a majority of clients. The assumption in this model is that the database *can be only updated by the servers.* The database consists of N numbered data items:d1, d2,…, dN and is fully replicated at each data server. Each server periodically broadcasts invalidation reports. To answer a query, the client on a mobile host listens to the next invalidation report and use that report to conclude whether its cache is valid or not. Invalid caches must be refreshed via a query to the server.

In the Bit – Sequence algorithm three techniques are used to optimize the size of report structure while retaining the invalidation effectiveness. To reference data items in the database a technique called ***Bit – Sequence naming*** is applied in the BS algorithm.

The server broadcasts a set of bit sequence. Each bit in a bit sequence represents a data item in the database. For example the nth bit in a size N of sequence represents data item dn. This technique can be applied when both client and server agree upon the mapping of bits to the names of data items in the server database. The client can find the data item that each bit represents in its cache based on the position of the bit in the sequence. The next technique is ***update aggregation.*** In the broadcast report, each sequence is associated with only one timestamp instead of separate timestamp to reach item. A bit "1" in the sequence means that the item represented by the bit has been updated since the time specified by the timestamp. A bit "0" means that the item has not been updated since the specified time. The update aggregation not only reduces the size of the report, but also decreases the invalidation precision of cache invalidation. To adapt to various disconnected clients, a technique called ***hierarchical structure of bit sequences*** is applied.

The BS algorithm can be explained by using the following example.

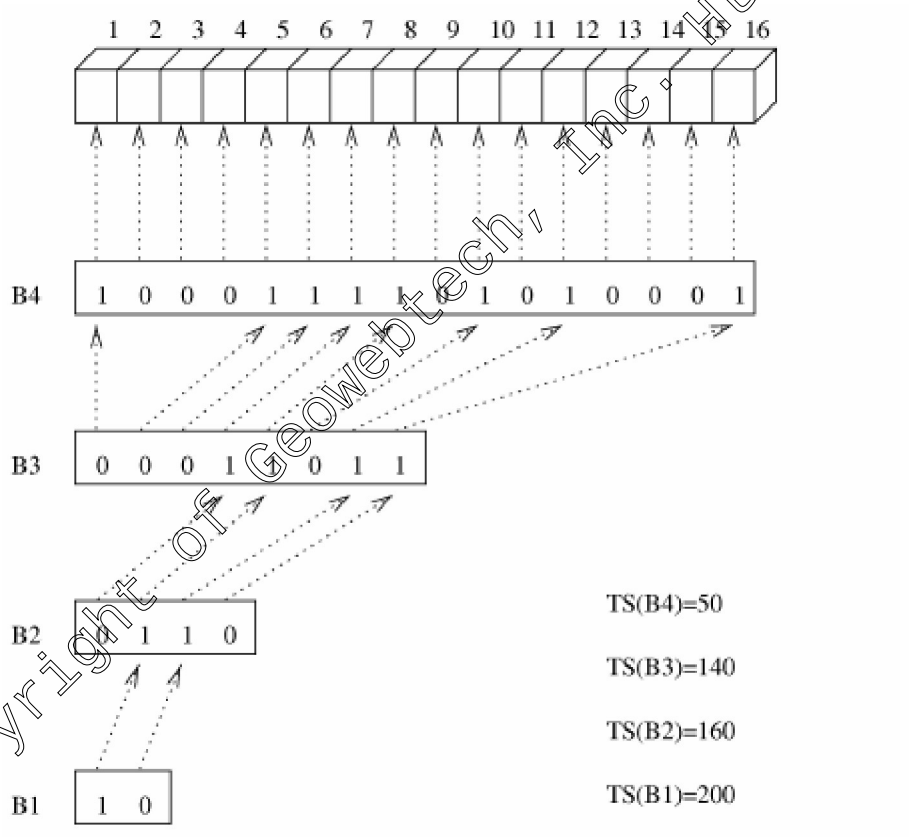


Figure 1. A Bit-Sequences example.

Consider a database consisting of 16 data items. Figure 1 shows a Bit-Sequences (BS) structure reported by a server at time 250. Suppose that a client listens to the report after having slept for 80 time units. That is, the client disconnected at time 170 (= 250−80), which is larger than TS($B2$) but less than TS($B1$). The client will use $B2$ to invalidate its caches. To locate those items denoted by the two "1" bits in $B2$, the client will check both $B3$ and $B4$ sequences, using the following procedure. To locate the second bit that is set to "1" in $B2$, check the position of the second "1" bit in $B3$. We see that the second "1" bit in $B3$ is in the 5th position; therefore, check the position of the 5th "1" bit in $B4$. Because $B4$ is the highest sequence and the 5th "1" bit in $B4$ is in the 8th position, the client concludes that the 8th data item was updated since time 170. Similarly, the client can deduce that the 12th data item has also been updated since that time. Therefore, both the 8th and 12th data items will be invalidated.

The main contributions of this approach include the following:

1) When a static bit mapping scheme is implicitly assumed, the BS algorithm can approach the optimal effectiveness for all data items indicated in the report regardless of the duration of disconnection of the clients. However, such optimization can be achieved only at the cost of about 2 binary bits for each item in the report.
2) The BS algorithm can also be applied to optimize other broadcast based cache invalidation algorithms in which the dynamic bit mapping has to be included explicitly. The optimization reduces the size of the report by about one half while maintaining the same level of effectiveness for cache invalidation.

3. The AS approach

In this paper, we present a caching scheme for wireless networks which uses asynchronous invalidation reports (callbacks) to maintain cache consistency, i.e., reports are broadcast by the server only when some data changes and not periodically. Frequent voluntary and involuntary disconnection of clients makes this a very difficult problem [5], [6], [7]. Each mobile client (host) (MH) maintains its own Home Location Cache (HLC) to deal with the problem of disconnections. The HLC of an MH is

maintained at a designated home Mobile Switching Station (MSS). It has an entry for each data item cached by the MH and needs to maintain only the time-stamp at which that data item was last invalidated.

In figure 2, the mobile hosts (MHs) query the database servers that are connected to a static network. The mobile hosts communicate with the servers via

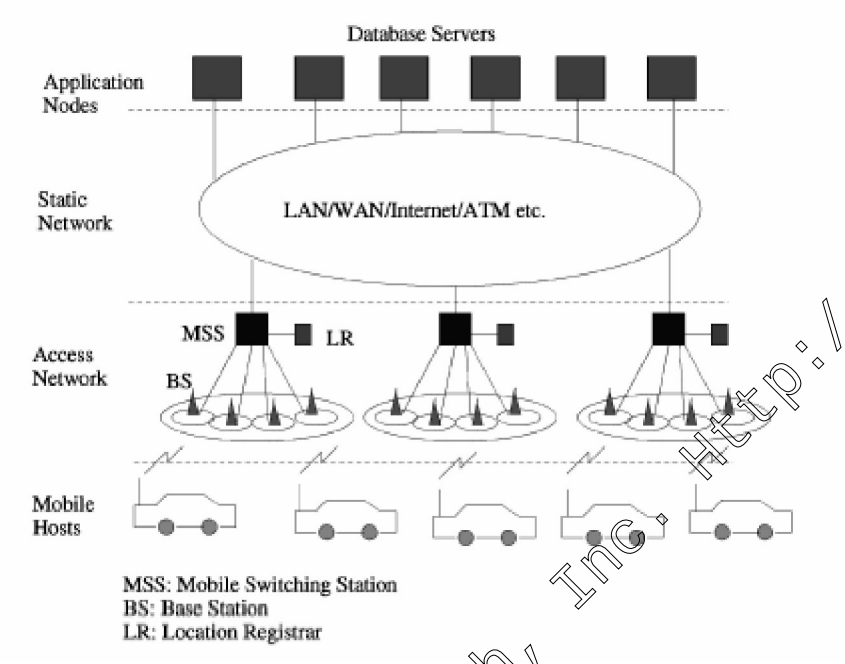


Figure 2

wireless cellular network consisting of mobile switching stations (MSS) and base stations. A mobile host can be in two modes: awake or sleep. When a mobile host is awake (connected to the server), it can receive messages. Hence, this state includes both active and dozing CPU modes. A MH can be disconnected from the network either voluntarily or involuntarily. From the perspective of the mobile host's cache, it is irrelevant whether the invalidation were delayed due to voluntary disconnection (e.g., switching off the laptop) or involuntary disconnection (e.g., wireless link failure, hand-off delay). Hence, for our purpose, a disconnected client is in sleep mode; we use the term wakeup to indicate reconnection. The client sends a uplink request (query) for the data it needs to the database server and the server responds by sending the requested data on the down-link. In order to minimize the number of uplink requests, the client caches a portion of the database in its local memory. The client-cached data is also referred to as

active data [8]. Caching data at clients necessitates a protocol between the client and the database server to ensure that the client cache remains consistent with the shared database. The objective of the proposed scheme is to minimize the overhead for the MHs to validate their cache upon reconnection, to allow stateless servers, and to minimize the bandwidth requirement. The general approach is to buffer the invalidation messages at Home Location Cache (HLC) Our caching scheme for the mobile environment is based on the following assumptions:

1) Whenever any data item is updated anywhere in the network, an invalidation message is sent out to all MSS via the wired network; thus, when a mobile host MH is roaming, it gets the invalidation message if it is not disconnected (we assume no message is lost due to communication failure or otherwise in the wired network).

2) An MH can detect whether or not it is connected to the network.

3) An MH informs its HLC before it stores (or updates) any data item in its local cache.

4) The static host, which is nearest to the MH and maintains the HLC of the MH, forwards the MH any invalidation it receives from the server.

Consider an MSS with N mobile hosts (MHi, 1 <= i <=N) at any given time. For any i, HLCi for MHi, as maintained in the MSS, keeps track of what data has been locally cached at MHi (state information of the MH). In general, HLCi is a list of records (x, T, invalid flag) for each data item x locally cached at MHi, where x is the identifier of a data item and T is the time-stamp of the last invalidation of x. The key feature of our scheme is that the invalidation reports are transmitted asynchronously and those reports are buffered at the MSS (in the HLCs of the mobile hosts) until an explicit acknowledgment is received from the specific MH. The invalid flag (in the HLC record for the specific data item) is set to TRUE for data items for which an invalidation has been sent to the MH but no acknowledgment has been received.

Each MH maintains a local cache of data items which it frequently accesses. Before answering any queries from the application, it checks if the requested data is in a consistent state. We use call-backs from a MSS to achieve this goal. When a MSS

receives invalidation from a server, the MSS determines the set of MHs that are using the data by consulting the HLCs and sends an invalidation report to each of them. When a MH receives that invalidation message, it marks the particular data item in its local cache to be invalid. When an MH receives (from the application layer) a query for a data item, it checks the validity of the item in its local cache. If the item is valid, it satisfies the query from its local cache and saves on latency, bandwidth and battery power; otherwise, an uplink request to the MSS for the data item is required.

In sleep mode, a mobile client is unable to receive any invalidation messages sent to it by its HLC. We use the following time-stamp based scheme by which the MA can decide which invalidations it needs to retransmit to the mobile host. Each client maintains a time-stamp for its cache called the cache time-stamp. Cache time-stamp of a cache is the time-stamp of the last message received by the MH from its MA. The client includes the cache time-stamp in all its communications with the MA. The MA uses the cache timestamp for two purposes:

1. To discard invalidations it no longer needs to keep, and
2. To decide the invalidations it needs to resend to the client.

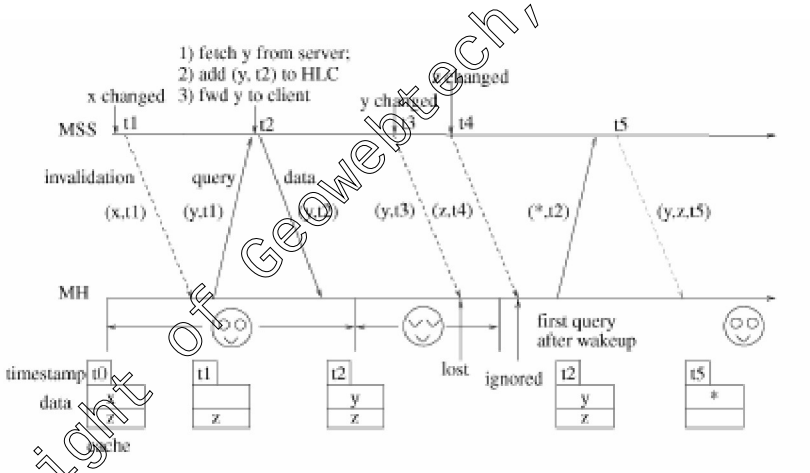


Figure 3

Consider the example scenario shown in Fig. 3. Initially, the cache time-stamp of the MH is t0 and MH's cache has two data items with ids x and z. When MSS receives an invalidation message notifying it that x has changed at the server at time t1, it adds the invalidation message to MH's HLC and also forwards the invalidation message to the

MH with (data-item id, time-stamp), i.e., .x; t1.. On receiving the invalidation message from the MSS, the MH updates its cache time-stamp to t1 and deletes data item x from its cache. Later, when MH wants to access y, it sends a data request with .y; t1. to the MSS. In response to the data request, the MSS fetches and forwards data item associated with y to the MH and adds .y; t2. to the MH's HLC, where t2 is the last updates time-stamp provided by the data server. The MH updates its time-stamp to t2 and adds y to its cache. Now, suppose MH gets disconnected from the network and the invalidation message for y is lost due to this disconnection. When MH wakes up, it ignores any invalidation messages it receives (until the first query), since later, upon the first query after waking up, it sends a probe message (invalidation check message) to the MSS. The MSS uses the time-stamp in this probe message to determine the invalidations missed by the MH and sends an invalidation report with all the missed invalidations by the MH. In this case, the MSS determines, from MH's cache time-stamp t2, that MH has missed invalidation for y and z and so it resends them to the MH.

4. Caching using data replication

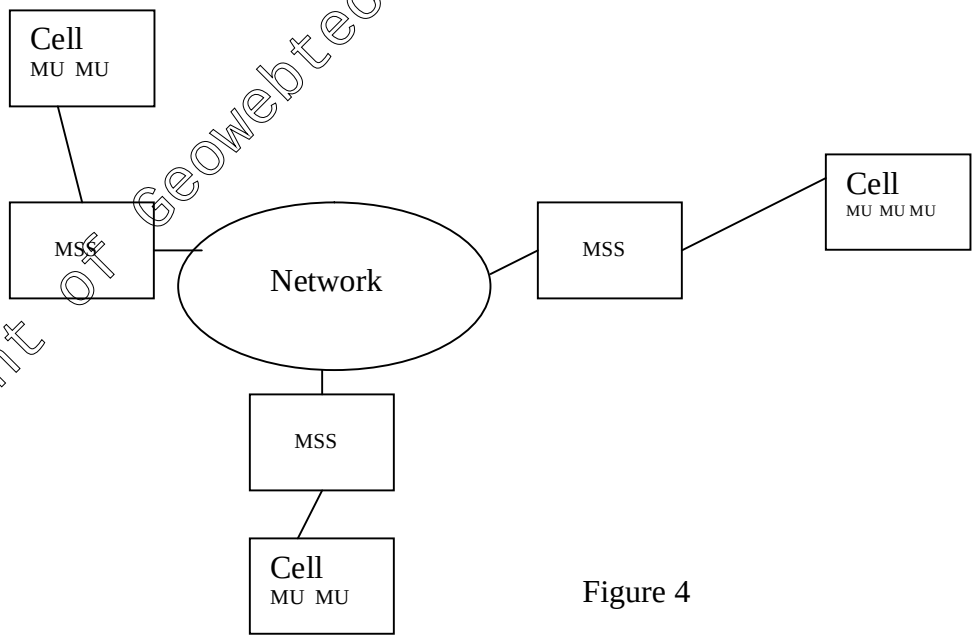


Figure 4

Consider the architecture shown in figure 4. Cell is radio coverage range over which an MSS can communicate with an MU. From the figure we can see two different entities. They are Mobile units(MU) and Fixed hosts – Some are Mobile Support Stations(MSS) that have a wireless interface.

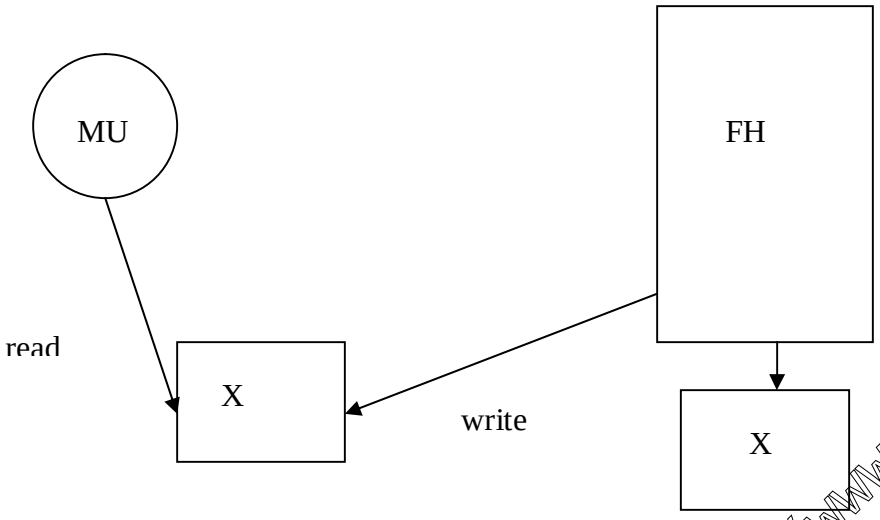
If a MU frequently reads a data-item x, and x is updated infrequently at the server, then it is beneficial for the MU to allocate a copy of x locally at the MU. MU will receive all updates of x.

If a MU reads x infrequently compared to the update rate, then a copy of x should not be stored locally at the MU. Access of that data should be on demand.

The caching allocation strategy is of two types. They are static – allocation scheme does not change over time and dynamic- allocation scheme changes over time.

Assumptions in this model are FH can only request write operations and MU can request only read operations. At any point of time whether or not MU has a copy of x, either the MU or the FH is aware of all the relevant requests. If MU has a copy of x, then all reads at the MU are satisfied locally, and all writes by FH are propagated to the MU. If the MU does not have a copy, then all reads issued by the MU are sent to FH. Two cases arises:

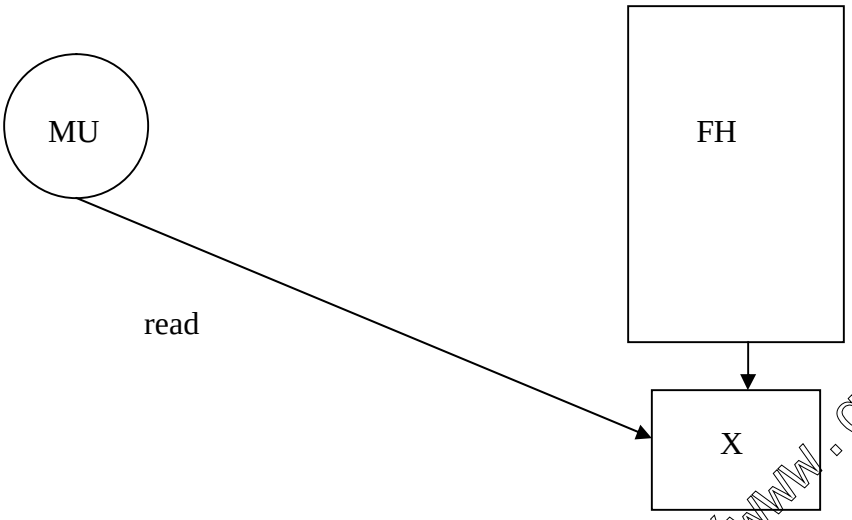
1) MU is in charge or has x in the cache i.e., # reads > # writes.



- If (# reads > # writes) then wait for the next operation.
- If(#writes > #reads) then deallocate copy.
- To deallocate MU sends x to FH

2) FH is in charge or the MU does not have X in the cache i.e. # reads < # writes.

- If (# reads < # writes) then wait for next operation
- If (# writes < # reads) then allocate copy to MU.
- Allocation consists of sending a copy of x to MU and also an indication to save the copy in MU's cache.



The data can be replicated on fixed sites or the fixed hosts (FH) in the network. Now it becomes possible for the MU to access data even after leaving one cell and joining another cell. The invalidation messages from the server will reach the FH which will then check its cache for the data. If the data is found then it is updated in the FH's local cache. If the data is not found then the corresponding MU will receive the invalidation report from the FH.

5. Comparison

This section provides a comparison between all the three schemes discussed in the previous sections. A possible solution to the limitations that arise in each paper was proposed in this section.

TS	AS
Server is stateless (no information about the client cache is maintained)	Server is stateful (HLC maintained)

Invalidation reports sent regardless of whether clients have any data in cache.	Invalidation reports broadcast only if client have valid data in the cache.
Cache restored for sleep limited to a maximum duration of w	Arbitrary sleep patterns can be supported
Mobility is supported by assuming a replication of data across all stationary nodes	Mobility can be transparently supported by using a mobility aware network layer e.g. mobile IP.

The AS scheme overcomes the limitation of the TS scheme where invalidation data report is broadcasted to all the clients , which is a serious problem because of the use of bandwidth for the invalidation messages which may or may not be used by the clients. The TS scheme also does not take into account the arbitrary connection pattern of the clients.

The AS scheme however has one disadvantage where the client even caches data which may be used infrequently. This limitation is taken care in the third strategy data replication where the client caches data only when the data is used more frequently than the update rate of the same data. But this scheme also does not take into account the arbitrary pattern of sleep time of the clients. In the third strategy the data is replicated at different fixed hosts on the network so that the mobile units can still be able to access the data even when moving to a different cell. This was missing in the AS scheme. AS scheme does not take into account the client movement to another cell. Also the TS algorithm does not take into account this factor.

One possible solution to this problem would be to combine the second and the third approach. The MH will cache data only when the # reads > # writes (update from the server). If the # reads < # writes then those data will be stored in the HLC of the MH.

This approach will definitely reduce the load on the mobile hosts. Storing data that may not be used frequently will result in unnecessary wastage of the valuable wireless bandwidth, due to the updates sent from the server. When the invalidation from the server is sent to the MH through the HLC.

6. Conclusion

This paper throwed light on three different strategies for caching in mobile environment. One was a bit-sequence algorithm, in which a periodically broadcast invalidation report is organized as a set of binary bit sequence with a set of associated timestamps. In the second approach the AS technique minimizes the overhead preserving bandwidth, reducing the number of uplink requests and average latency. State information about the local cache at MH with respect to data items is maintained at the home MSS; by sending asynchronous call-backs and buffering them till implicit acknowledgments are received, the cache continues to be valid even after the MH is temporarily disconnected from the network. Maintaining state information at the MSS can be considered as an overhead, but has the capability to provide various other benefits beyond this scheme. For example profiling techniques can be used by the MSS to determine what to cache at the hosts when the cache space is limited [9]. It is expected to provide a platform to enable prefetching of data [10] or hoarding of files [11] at the clients. The third approach which is the data replication strategy where the data is cached at the mobile host only when the number of reads is more than the number of update operations by the server.

Autonomous operation is highly desirable in a mobile computer. This can be achieved by caching - using one of the three techniques discussed in this paper. The limitation of all the approaches was discussed in the comparison section and a possible proposal for the limitation described was also shown.

References

1) Bit Sequences: An adaptive cache invalidation method in mobile client/server environments. Jin Jing, Ahmed Elmagarmid, Abdelsalam (Sumi) Helal and Rafael Alonso
2) A strategy to manage cache consistency in a disconnected distributed environment. Anurag Kahol, Sumit Khurana, Sandeep K.S. Gupta, Senior Member, IEEE, and Pradip K. Srimani, Fellow, IEEE.
3) Data Replication in a mobile environment. Raymond Pon CS244A , Wesley W. Chu.

www.cs.ucla.edu/rpon/Presentations/Data%20Replication%20in%20a%20Mobile%20Environment.ppt

4) D.Barbara and T.Imielinski, Sleepers and workaholics: Caching strategies for mobile environments, in: Proceedings of ACM SIGMOD Conference on management of data.
5) R.Alonso, D. Barbara, and H. Garcia-Molina, " Data caching issues in an information retrieval system", ACM Trans. Database Systems, vol 15, pp. 359-384, Sept. 1990.
6) R. Alonso and H.F. Korth, "Database Systems issues in nomadic computing", Technical Report MITL-TR-36-92, Matsushita information technology laboratory, Princeton, N.J. 08542-7072, Dec. 1992.
7) T. Imielinski and B.R. Badrinath, " Wireless Computing: Challenges in data management", comm.. ACM, vol. 37, no.10, Oct. 1994.
8) K. Wilkinson and M.A. Neimat, "Maintaining Consistency of client cached data", Proc. 16th Int'l Conf. Very Large Data Bases (VLDB '90), pp. 122-133, Aug. 1990.
9) S.L. Tong and V. Bharghavan, "Alleviating the latency and bandwidth problems in WWW browsing", USENIX symp. Internet Technologies and Systems.
10) M. Crovella and P.Barford, "The Network Effects of prefetching", Proc. Conf. Computer Comm. INFOCOM '98.
11) G.H. Kuenning and G.J. Popek, "Automated hoarding for mobile computers", Proc. 16th ACM Symp. Operating System Principles, 1997.
12) Challenges of mobile computing, Prof. Randy H. Katz http://www.cs.berkeley.edu/~randy/Courses/CS294.S96/CS294-7.S96.html.
13) Mobile Computing, Prof. Albert Y. Zomaya. http://www.cs.usyd.edu.au/~zomaya/mobile.html.