Constructing Efficient Cache Invalidation Schemes in Mobile Environments

Po-Jen Chuang and Yu-Shian Chiu
Department of Electrical Engineering
Tamkang University
Tamsui, Taipei County
Taiwan 25137, R. O. C.
E-mail: pjchuang@ee.tku.edu.tw

Abstract

*Cache invalidation is an effective approach to maintaining data consistency between the server and mobile clients in a mobile environment. This paper presents two new cache invalidation schemes which are designed according to the real situations and are therefore able to comply with the more practical needs in a mobile environment. The **ABI+HCQU** divides data into different groups based on their utilization rates (Hot/ Cold/ Query/ Update) and adapts their broadcasting intervals (ABI) accordingly to suit the actual needs. The **SWRCC + MUVI** (Sleep/ Wakeup/ Recovery/ Check/ Confirm + Modified/ Uncertain/ Valid/ Invalid) aims to solve the validity problem of cached data after a client is disconnected from the server. The new cache invalidation schemes are shown through experimental evaluation to outperform most existing schemes in terms of data access time, cache miss rates and bandwidth consumption.*

1. Introduction

A mobile environment contains a large number of mobile clients and a small number of database servers. The servers (each contains all the data in the system) are connected through a wired network, while the mobile clients (each stores only a part of the data in its cache) are connected to a server through a wireless communication channel. In such a mobile environment, users can use mobile computers to access data without temporal or spatial limitations. To reduce data access time and bandwidth consumption for mobile communications, data items are usually cached at the end of the mobile clients. Thus when a server updates its data items, the corresponding cached data at the mobile clients must be updated as well to maintain data consistency and correctness. Cache invalidation is a popular and effective approach to attain data consistency between the server and mobile clients. In cache invalidation, the server will broadcast, periodically or aperiodically, invalidation reports (IRs) to the mobile clients which then invalidate and update the cached data items according to the reports. Various cache invalidation schemes have been proposed in the literature, including the Timestamp (TS, the simplest method [1]), Bit-Sequence (BS which broadcasts additional bit sequences [2]), Invalidation by Absolute Validity Interval (IAVI in which the cached items can verify their validity using AVIs [3]), Update Invalidation Report (UIR – based on TS but broadcasting additional UIRs between two successive IRs [4]), Counter-based (CONT which broadcasts additional hotly accessed data [5]), Reducing Improving Handling (RIH – based on UIR but the server can answer a query after each UIR packet [6]) and Adaptive Energy Efficient Cache Invalidation Scheme (AEECIS which decides the cache invalidation method – Group/TS/Real-Time – according to the uplink ratio of the clients [7]).

To attain better performance, this paper presents two new cache invalidation schemes which are designed according to the actual situations in a mobile environment and are therefore able to comply with the more practical needs. Our first scheme, **ABI+HCQU**, aims to make IRs carry the most effective information and also to reduce data access time or query latency. In the scheme, we adopt an Adaptive Broadcasting Interval (ABI) approach which first divides data into 5 groups (Hot Query, Hot Update, Cold Query, Cold Update and ReMAinder) based on their utilization conditions and then adapts the broadcasting intervals of the 5 groups according to their utilization frequency. More "popular" data (such as HQ and HU) will be broadcast in shorter intervals to meet the practical demands and to save data access time, while less "popular" data (such as CQ, CU and RMA) will be

978-0-7695-3122-9/08 $25.00 © 2008 IEEE
DOI 10.1109/SITIS.2007.133

broadcast in longer intervals to avoid unnecessary bandwidth consumption.

Our second scheme, **SWRCC+MUVI**, attempts to solve, in a more efficient way, the validity problem of cached data which happens when a client disconnects to the server – actively or passively. In our design for active disconnection (SWR), a client will send the server a **Sleep** message in advance to inform of its intended disconnection and later a **Wakeup** message to announce its return and meanwhile to demand for the first queried item. The server will then broadcast a **Recovery** message to the client, revealing all data items that have been updated during the disconnection and the valid information of the first queried item. In the approach for passive disconnection, a reconnected client can obtain the queried data items from its neighboring clients by exchanging the **Check** and **Confirm** (CC) messages with them via low-power broadcasting and also by classifying the validity of the acquired data into **Modified**, **Uncertain**, **Valid** and **Invalid** (MUVI). The approach allows a reconnected client to obtain most queried data from its neighboring clients, instead of from the server, saving a significant amount of bandwidth consumption.

Extended simulation runs are conducted to evaluate the performance of our new cache invalidation schemes and existing schemes. The results show that the proposed ABI+HCQU not only outperforms the other existing cache invalidation schemes in average data access time and cache miss rates, but is also bandwidth-conserving as it consumes more bandwidth than SWRCC+MUVI and BS only. As to SWRCC+MUVI, it is shown to consume the least bandwidth among all schemes except BS (which has the smallest broadcast bit sequences), mainly because it is able to get most data from neighboring clients, instead of from the server.

2. The New Cache Invalidation scheme

Our study on existing cache invalidation schemes leads to the following three critical questions which, if properly answered, can save remarkable amounts of resources (energy and bandwidth) and as a result improve the quality of mobile communications.

(1) How can a stateless server broadcast IRs to the mobile clients in a more efficient way?

(2) When a client uplinks requests to a server, how can we shorten the server's response time (i.e., the query time)?

(3) After a client reconnects to the server, how can we make the most of the cached data in that client?

2.1 The Adaptive Broadcasting Interval (ABI)

To make IRs carry the most effective information and also to reduce query latency, we adopt an adaptive broadcasting interval (ABI) approach which adjusts the broadcasting intervals of IRs to meet practical communication needs. The main function of ABI is to bring more prompt and critical information to the mobile clients, i.e., to amend the weakness of periodically broadcast IRs. To operate ABI, we first define a threshold value H to decide if the information in an IR is valid:

(1) If the number of updated data exceeds H during 1 broadcasting interval, the information in the IR is considered outdated. The broadcasting interval thus must be adjusted. Assuming the original broadcasting interval is B, it can now be changed to B/2 ($5<B/2<20$). That is, IRs will be broadcast in shorter intervals to satisfy the frequent query requests of clients.

(2) If the number of updated data falls below H during 1 broadcasting interval (i.e., the clients do not file frequent queries to the server), the length of IR broadcasting intervals can be doubled. For instance, if the original interval is B, it can be lengthened into B*2 ($20<B*2<80$), to save resources.

2.2 The Hot/Cold vs. Query/Update (HCQU) Approach

The concept and definition of "hot data" in UIR and CONT inspire us to categorize data into different groups according to their frequency of being queried or updated. To achieve more effective cache invalidation as well as more desirable bandwidth utilization, we divide data into 5 groups: Hot Update (HU), Hot Query (HQ), Cold Update (CU), Cold Query (CQ) and the remainder (RMA), each with a proper broadcasting interval that suits their different demands. The HCQU approach operates as follows.

(1) The server and clients first predefine the five groups, to be redefined after a fixed time interval. To give an example, the five groups can be predefined as

HU：The top 5% of data updated most frequently.

HQ：The top 5% of data queried most frequently.

CU：The bottom 5% of data updated least frequently.

CQ：The bottom 5% of data queried least frequently.

RMA：The remainder of data, belonging to none of HU, HQ, CU or CQ.

(2) The server will broadcast five different IRs (HU-IR, HQ-IR, CU-IR, CQ-IR and RMA-IR) in their own broadcasting intervals.

(3) Repeat steps 1 and 2.

2.3 ABI + HCQU

As stated above, the proposed ABI can adapt the IR broadcasting intervals (based on the amount of updated data) to reduce data access time or query latency, while the proposed HCQU can divide data into different groups (based on utilization frequency) to attain desirable bandwidth utilization. When ABI works together with HCQU, better performance is expectable.

After HCQU divides data (in the server) into groups HU, CU, HQ, CQ and RMA, ABI will check each group's update and query frequency during 1 broadcasting interval and adapt their broadcasting intervals according to the result. The broadcasting intervals of HU/CU/RMA and HQ/CQ will be adjusted according to the update frequency or the query frequency in a fixed period of time. As each group has its own broadcasting interval, we will follow the broadcasting priority HQ>HU>RMA>CQ>CU to avoid overlapped broadcasting time of different groups.

Before answering a query, a mobile client must use the next received IR to check the validity of the cached data. We adjust the broadcasting interval of each group as follows to reduce the response time.

(1) The shorter broadcasting interval is given to HU and HQ (the real data will be broadcast more frequently), and the longer broadcasting interval is given to CU and CQ.

(2) The broadcasting interval for RMA is the average of broadcasting intervals for HU, CU, HQ and CQ.

Thus the "hot" data will be more frequently broadcast and the "cold" data will be less frequently broadcast. Note that such an arrangement consumes only slightly more bandwidth while substantially reduces the query response time.

2.4 The Sleep/ Wakeup/ Recovery/ Check /Confirm (SWRCC) Approach

A mobile client may get disconnected from the server actively (to conserve energy) or passively (due to moving or accidental factors), and thus may take different cache invalidation approaches to verify the validity of its cached data after reconnection.

- **Our Design for Active Disconnection (SWR):**

(1) A client will send a **Sleep** message to inform the server of its intended disconnection.

(2) When the disconnected client reconnects to the server and receives the first query, it will send a **Wakeup** message (which also includes the information of the received first query) to the server.

(3) Receiving such a Wakeup message, the server will move on to send a **Recovery** message to the client. Enclosed are all data items which have been updated during the disconnection period and also the valid information of the first query.

(4) The client then updates its cached items and answers the first query according to the Recovery message. (By allowing a reconnected client to update only items which are invalidated during disconnection, instead of all cached items, the SWR design can avoid unnecessary invalidation of still valid data items to save the valuable bandwidth resources.)

- **Our Design for Passive Disconnection (CC):**

(1) When a client gets reconnected to the server and receives a query request after passive disconnection, it will send a Check message to the neighboring clients (via low-power broadcasting) at every Check Period time interval (to be set as 1/4 or 1/5 of the IR broadcasting interval). The Check message **Check [(id_1,ts_1), (id_2,ts_2),…, (id_n,ts_n)]** includes the id set of queries received during the time interval (n being the total number of the queried data).

(2) Upon receiving such a Check message, a neighboring client first checks its own cache for the requested data item. If the item is in its cache and is still valid, the neighboring client then sends a Confirm message **Confirm [(id_1,ts_1,data_1), (id_7, ts_7, data_7), …]**to the reconnected client and the other neighboring clients.

(3) Receiving the Check and Confirm messages, the other neighboring clients will respond by broadcasting data items which appear in the Check message but not in the Confirm message.

(4) The reconnected client will not answer queries until it receives the next IR from the server (because the data obtained from Confirm messages are only tentative data whose validity must be confirmed by the newest IR).

2.5 Validity States: Modified /Uncertain /Valid /Invalid (MUVI)

When designing the SWRCC approach, we face a problem: How to properly indicate the validity of cached data items? As the validity representation of cached data in a mobile environment resembles that in a multiprocessor, we study the MESI protocol [8] in the multiprocessor and modify it to fit the Client-Server environment and our SWRCC approach. In our way of validity representation, we use two additional bits to indicate the state of each cached data as Modified, Uncertain, Valid or Invalid.

- **Definitions of MUVI**

Modified (M): As mentioned above, when a client reconnects to the server and receives a query message, it will send a Check message to the neighboring clients via lower-power broadcasting and receive possibly valid data from their Confirm messages. At this point, the state of these possibly valid cached data will be changed from Uncertain to Modified.

Uncertain (U): When a client is initially reconnected to the server, the validity of its all cached data is uncertain. Their validity state is thus marked as Uncertain.

Valid (V) and Invalid (I): After reconnection and receiving the first IR, the client moves on to check the validity of its own cached data. The state of data items which are still valid is marked as Valid (V), while the state of data items which become invalid is changed into Invalid (I).

- **State Transition of MUVI**

Upon query requests, the state of the cached data item will change as follows.

Cache Hit and Valid: If the newly received IR shows the requested cached data item is still valid (M/V/U→V), the client can use it to answer the query.

Cache Hit but Invalid: If the newly received IR shows the requested cached data item is invalid (M/V/U →I), the client will uplink the id of the item to the server and answer the query after receiving the broadcast data item from the server.

Cache Hit but Uncertain: A client gets reconnected to the server (M/V/U→U), receives a query and sends a Check message to the neighboring clients who then broadcast Confirm messages with the possibly valid data item to the client (U→M).

(1) After receiving the next IR, if the client finds the requested cached data item is valid (M→V), it will answer the query with the item.

(2) Otherwise (M→I), it will uplink the id of the queried item to the server and answer the query after receiving the broadcast data item from the server.

Cache Hit but Modified: In this situation, the client already receives the possibly valid data item and will wait for the next IR to invalidate the item.

(1) After receives the next IR, if the data item is valid (M→V), it can use the cached data item to reply the query.

(2) Otherwise (M→I), it uplinks the id of the queried item to the server and will reply that query after receives the broadcast data item from the server.

Cache Miss: The queried item is not in the cache of the client which will then broadcast a Check message to the neighbors. If the client receives Confirm messages (along with the queried item) from neighbors, it will set the state to M and wait for the next IR to verify the item. If the client receives no Confirm messages from neighbors, it has to uplink the request to the server.

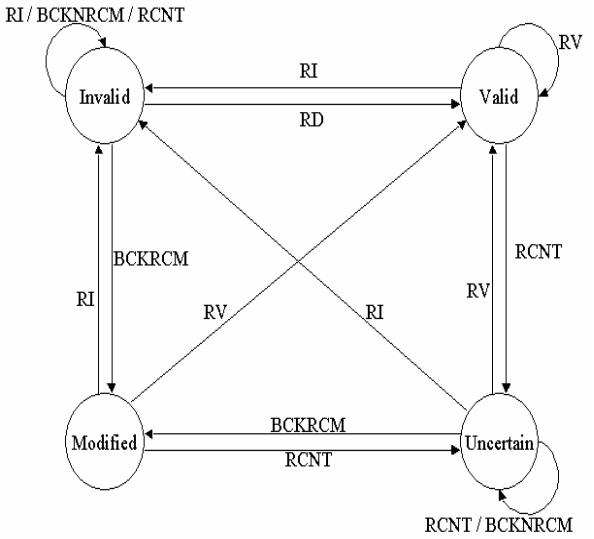

Figure 1. The MUVI state transition diagram for data items in the cache.

Figure 1 presents the MUVI state transition for data items in the cache. The acronyms in the figure are illustrated below.

RD (Receive the Data item from the server): I→V.

RV (Receive the next IR and verify the cached item as Valid): M→V, U→V, V→V.

RI (Receive the next IR and verify the cached item as Invalid): M→I, U→I, V→I, I→I.

RCNT (ReCoNnecT with the server): M→U, U→U, V→U, I→I.

BCKRCM (Broadcast CheckK messages and Receive Confirm Messages): U→M, I→M.

BCKNRCM (Broadcast CheckK messages but do Not Receive Confirm Messages): U→U, I→I.

2.6 SWRCC+MUVI

There are two major advantages for employing SWRCC+MUVI. One is the bandwidth conservation between the clients and the server -- because data transfer mostly happens among the clients. With growing number of clients in the mobile environment, more uplink and downlink messages will be saved, and so will the bandwidth resource. Another advantage is the real-time response after active disconnection. In active disconnection, the client will send a Sleep message to the server before going into the sleep mode. At the end of the Sleep, the server will broadcast a special invalidation report (especially for the disconnection time) to the client. The client wakes up

to receive the invalidation report and goes ahead to invalidate its cached data items. In this way the reconnected client can update the validity of its cache at possibly the earliest time.

3. Performance Evaluation

Extended simulation runs are conducted to evaluate and compare the performance of our new cache invalidation scheme and other existing schemes, including the TS, BS, IAVI, UIR, CONT, RIH and AEECIS [1-7].

3.1 The Simulation Model

We have constructed a simulator similar to that adopted in [6]. It employs one server and a maximum amount of 20 clients. The maximum number of data in the server is 10,000 and the maximum number of cached data in a client is 500. Data will be updated only in the server, not in the clients. The size of each data, each data id and a timestamp is 256 bytes, 17 bits and 64 bits, respectively. The bandwidth for the uplink or between clients is 19.2 kbps, and the bandwidth for the downlink is 100 kbps. For a full cache, we adopt the Least Recently Used (LRU) policy to replace data items (i.e., the oldest cached item will be replaced by the newest one). The database contains 5% hot data and 95% cold data. The probability for querying and updating the hot data and cold data is respectively set as 95% and 5%. The exponentially mean update time and query (generating) time for the data is 1000 seconds and 30 seconds.

3.2 Simulation Results

As mentioned, our invalidation scheme is designed mainly to reduce the data access time and bandwidth consumption in a mobile environment. Thus the performance measures of interest in our experimental evaluation include the average access time, cache miss rate and bandwidth consumption. The involved parameter variations include the disconnection probability, disconnection time, cache size and database size, which will exercise different influences on the targeted performance measures. For instance, growing database sizes and disconnection probabilities will both increase the cache miss rate, and when more cache misses occur, the clients need to uplink more requested data ids to the server, lengthening the average access time and as a result wasting bandwidth resources.

Note that the TS-based cache invalidation strategies, such as TS, IAVI, UIR and CONT, will invalidate all cached data when the disconnection time exceeds broadcasting interval*broadcast window (L*w), despite the fact that some data items may remain valid. These strategies suit better only when the disconnection time falls between the current timestamp T and T-L*w. For longer disconnection time, their performance may degrade. The Bit-Sequence (BS) architecture, by contrast, is more flexible with this issue: If data updates happen less frequently, it can tolerate longer disconnection time. However, when data are updated more and more frequently, the architecture will tolerate shorter and shorter disconnection time. For improvement, we employ SWRCC+MUVI, a new design which allows a reconnected client to acquire queried data by exchanging the Check and Confirm messages with neighboring clients via lower-power broadcasting. Such a design helps reduce the uplinking and downlinking of requested data between the server and clients and thus reduce bandwidth consumption, which is especially important for a resource-constrained mobile environment.

For cache invalidation strategies with periodical IR broadcasting, the server will periodically broadcast data, “hot” or “cold”, to the mobile clients. Periodically broadcasting hot data can be practical, but periodically broadcasting cold data will unnecessarily consume the bandwidth and proves futile. To avoid unnecessary bandwidth consumption, i.e., to attain better bandwidth utilization, our ABI+HCQU approach allows hot data to have higher broadcasting frequency while cold data to have lower broadcasting frequency, duly reflecting the real situations.

- **The Average Access Time**

Figures 2 and 3 demonstrate the average access time for various schemes under different disconnection time and disconnection probability. The obtained results clearly exhibit the effect of different designs among the schemes. For instance, as AEECIS is constantly in the Slow or Fast mode, its average access time will appear longer than that of TS. On the other hand, as IAVI can use the AVI values to predict if data are valid or not and uplink requests to the server ahead of time, its average access time stands shorter than TS. As to UIR, it has shorter access time than TS (due to its high broadcasting frequency), and longer access time than CONT (which is based on UIR but with additional broadcasting of hotly queried or updated data) and RIH (also based on UIR but with additional broadcasting of actual data after each UIR, not each IR).

The results show that the proposed ABI+HCQU approach generates the shortest average access time among all schemes. This is because ABI+HCQU classifies data into different groups based on their

query and update frequency and adapts their broadcasting intervals accordingly to comply with practical needs. Figures 2 and 3 both indicate a nearly unchanged average access time, under all conditions, for the proposed SWRCC+MUVI approach. This is understandable because SWRCC+MUVI allows a client to attain the requested data by exchanging the Check and Confirm messages with neighboring clients. A client will go to the server only when the queried items are not in its cache or its neighboring clients' caches – which is indeed a rare situation. Disconnection time or disconnection probability will influence its average access time only slightly.

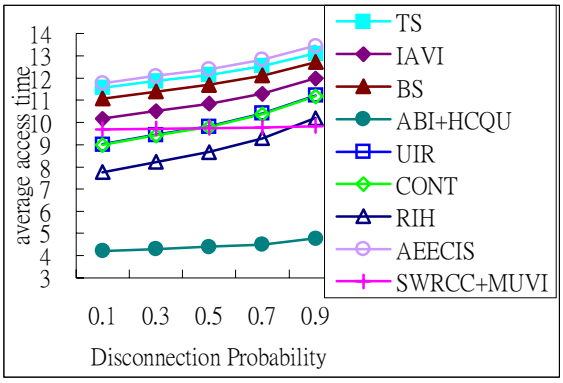

Figure 2. The average access time vs. the disconnection probability for various schemes.

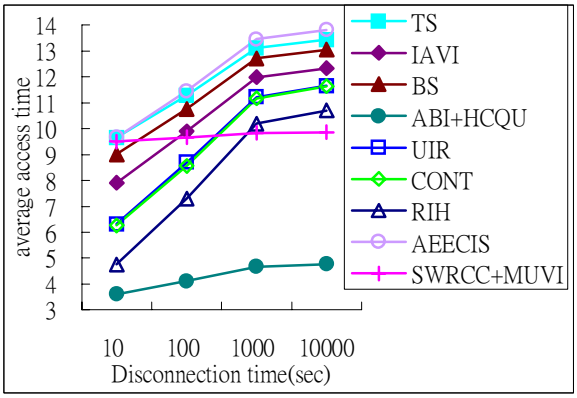

Figure 3. The average access time vs. the disconnection time for various schemes.

- **The Cache Miss Rate**

Figures 4 and 5 depict the cache miss rate obtained under varied disconnection time and disconnection probability. As can be expected, higher disconnection time and probability will generate higher cache miss rate. For TS-based strategies (TS and IAVI), the increase in cache miss rates turns milder when disconnection time exceeds 1000 sec, and so do the UIR-based strategies (UIR and RIH). Without any special management on hot data, the BS strategy yields similar cache miss rates as the TS strategy. It is clear that with special managements on hot data (such as allowing most cached data to be hot data), the probability for cache miss will be reduced when a client receives requests. Among these strategies, CONT, AEECIS and ABI+HCQU are strategies with special handling of hot data. For CONT, each data in the database has a counter to record its frequency of being queried or updated; when the value of the counter gets bigger than the set threshold, it will be broadcast after the IR. AEECIS attains desirable cache miss rates as it changes the mode of invalidation report according to the number of clients that have uplinked requests to the server. The proposed ABI+HCQU achieves the lowest cache miss rate because it first classifies the database into several groups based on their frequency of being used and accordingly adapts their broadcasting intervals to suit practical needs.

- **Bandwidth Consumption**

Figures 6 and 7 display bandwidth consumption vs. different disconnection time and probability. When a client receives a query during disconnection, it has to uplink the query to the server after reconnection. Thus when disconnection probability increases, the demand to uplink queries to the server will also increase. Bandwidth consumption thus escalates.

As the results show here, CONT consumes the most bandwidth because it has to broadcast additional actual data, and BS consumes the least bandwidth because it has the smallest broadcast bit sequences. RIH also consumes considerable bandwidth because it broadcasts actual data after each UIR and may thus repeatedly broadcast some hot data during an IR broadcasting interval. The broadcast information for the TS-based schemes, such as TS, IAVI and UIR, is less than that for RIH and CONT, thus requiring less bandwidth consumption. The results also reveal that AEECIS, which is constantly in the Fast mode or Slow mode, needs less bandwidth consumption than TS. ABI+HCQU consumes even less bandwidth as it dynamically adapts the broadcasting intervals of its classified data groups to assure that hot data are delivered to the needing clients at possibly the earliest instant while cold data are broadcast less frequently to avoid unnecessary broadcasting. Bandwidth consumption for the proposed SWRCC+MUVI stands higher only than that for BS. This is because SWRCC+MUVI first employs the Check and Confirm messages to get data from neighboring clients and as a result reduces the need to uplink requests to the server.

Our simulation results also show that when the size of database grows, the broadcast information grows too

and so does bandwidth consumption. On the other hand, an enlarged cache size will accommodate more cached data and thus reduce cache miss rates. When cache miss rates are reduced, bandwidth consumption can be saved due to decreased query uplinks. (Due to limited space, the results for bandwidth consumption vs. cache and database sizes are not illustrated as figures.)

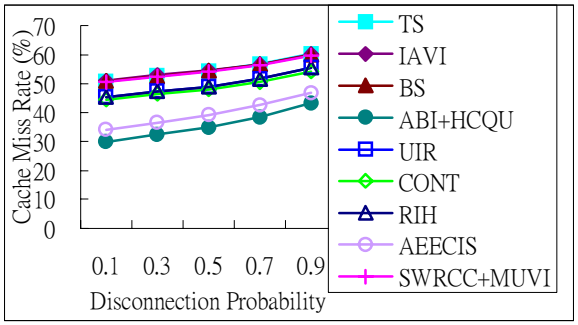

Figure 4. The cache miss rate vs. the disconnection probability for various schemes.

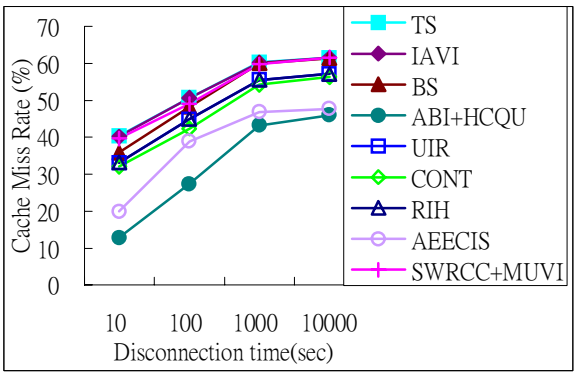

Figure 5. The cache miss rate vs. the disconnection time for various schemes.

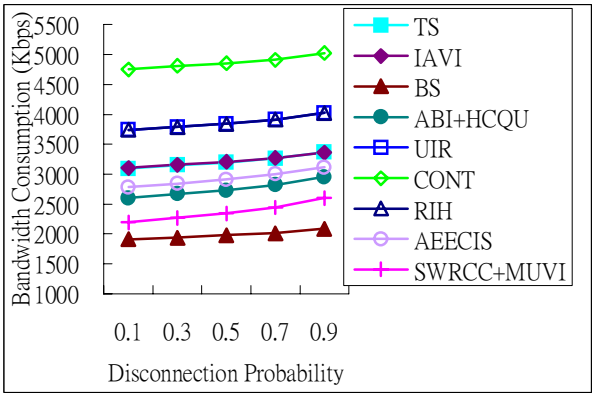

Figure 6. The bandwidth consumption vs. the disconnection probability for various schemes.

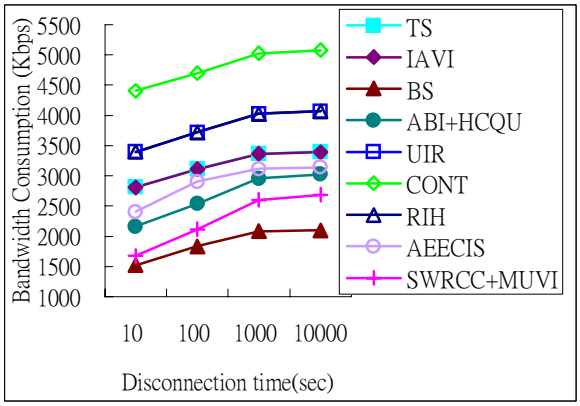

Figure 7. The bandwidth consumption vs. the disconnection time for various schemes.

- **Active / Passive disconnection**

Figures 8 and 9 respectively depict the cache miss rate of our SWRCC+MUVI in dealing with active disconnection and of the TS strategy with SWR. Note that 20% active disconnection indicates 80% passive disconnection in the network. When the active disconnection percentage is 0%, i.e., when all of the disconnection is passive disconnection, the Sleep, Wakeup and Recovery messages will not be sent at the same time. In the original MUVI design (i.e., without SWR), a node will send messages to the neighbors to ask for the data after getting reconnected to the server. But in the SWR approach, a server will send the updated IDs (updated during the disconnection period) to the reconnected client. Thus as the results in Figures 8 and 9 exhibit, SWR casts less influence on SWRCC+MUVI than on the TS strategy, and with increasing active disconnection rates, cache miss rates will drop.

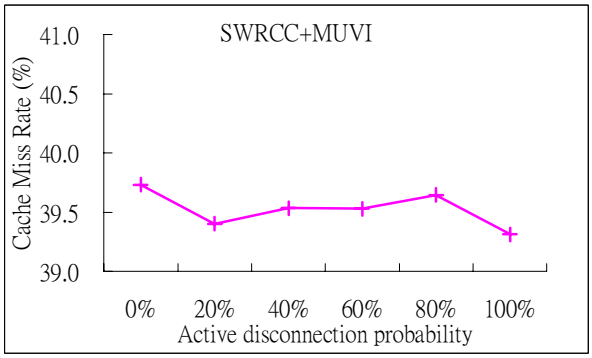

Figure 8. The cache miss rate vs. the active disconnection probability for SWRCC+MUVI.

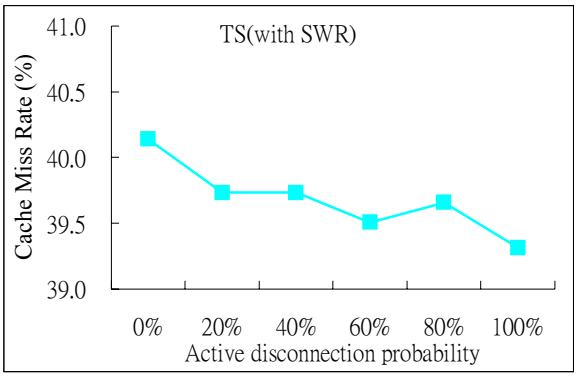

Figure 9. The cache miss rate vs. the active disconnection probability for TS (with SWR).

4. Conclusions

This paper introduces two cache invalidation schemes, ABI+HCQU and SWCC+MUVI, to maintain data consistency between the server and clients in a mobile environment. The proposed ABI+HCQU first divides data into 5 different groups (Hot Query, Hot Update, Cold Query, Cold Update and ReMAinder) based on their utilization conditions or their query and update frequency, and then adapts the broadcasting intervals of the 5 groups according to their "importance" or "popularity". That is, more popular data (such as HQ and HU) will be broadcast to the clients in shorter intervals (i.e., more frequently) to satisfy the actual demands and to reduce the data access time. By contrast, less popular data (such as CQ, CU and RMA) will be broadcast in longer intervals (i.e., less frequently) to avoid unnecessary bandwidth consumption.

The proposed SWRCC+MUVI aims to solve, in a more efficient way, the validity problem of cached data which happens when a client disconnects to the server. Different approaches are adopted to comply with the different situations of active and passive disconnections. In active disconnection, a disconnected client will send the server a Sleep message, before going away, to inform of its intended absence and a Wakeup message, after getting back, to announce its return and meanwhile to ask for the first queried item. The server will respond by broadcasting a Recovery message to the client, which includes all data items that have been updated during the disconnection and also the valid information of the first queried item. For passive disconnection, when a disconnected client gets reconnected to the server, SWRCC+MUVI allows the client to obtain the queried data items from its neighboring clients by exchanging the Check and Confirm messages via low-power broadcasting, instead of directly uplinking the requests to the server, thus saving a significant amount of bandwidth consumption.

Experimental performance evaluation shows that our ABI+HCQU outperforms the other existing cache invalidation schemes in average data access time and cache miss rates and is also bandwidth-conserving (consuming more bandwidth than SWRCC+MUVI and BS only). SWRCC+MUVI is shown to consume the least bandwidth among all schemes except BS as it is able to get most data from neighboring clients, instead of from the server.

5. References

[1] D. Barbara, T. Imielinski, "Sleepers and Workaholics: Caching Strategies in Mobile Environments," *Proc. 1994 ACM Int'l Conf. on Management of Data*, Vol. 23, No. 2, pp. 1-12, May 1994.

[2] J. Jing, A. Elmagarmid, A. Heal, and R. Alonso, "Bit-Sequences: An Adaptive Cache Invalidation Method in Mobile Client/Server Environments," *Mobile Networks and Applications*, Vol. 2, No. 2, pp. 115-127, 1997.

[3] J. C. H. Yuen, E. Chan, K.Y. Lam, and H. W. Leung, "An Adaptive AVI-based Cache Invalidation Scheme for Mobile Computing Systems", *Proc. 2000 Int'l Workshop on Mobility in Databases and Distributed Systems*, Sept. 2000, pp. 155-159.

[4] G. Cao, "A Scalable Low-Latency Cache Invalidation Strategy for Mobile Environments," *IEEE Trans. on Knowledge and Data Engineering*, Vol. 15, No. 5, pp. 1251-1265, Sept./Oct. 2003.

[5] C. C. Wu, J. F. Fang, and P. C. Hung, "A Counter-Based Cache Invalidation Scheme for Mobile Environments with Stateless Servers", *Proc. 2003 IEEE Pacific Rim Conf. on Communications, Computers and Signal Processing*, August 2003, pp. 623-626.

[6] N. Chand, Joshi, R. C., and Misra, M., "Energy Efficient Cache Invalidation in Wireless Mobile Environment", *Proc. 2005 IEEE Int'l Conf. on Personal Wireless Communications*, Jan. 2005, pp. 244-248.

[7] A. Madhukar, R. Alhajj, "An Adaptive Energy Efficient Cache Invalidation Scheme for Mobile Databases", *Proc. 2006 ACM Symp. on Applied Computing*, Apr. 2006, pp. 1122-1126.

[8] W. Stallings, *Computer Organization and Architecture: Designing for Performance*, 4th Ed., Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1996.