Dynamic Cache Invalidation Scheme in IR-based Wireless Environments

Yeim-Kuan Chang, I-Wei Ting and Tai-Hong Lin
National Cheng Kung University, Tainan, Taiwan
{ykchang, p7893113}@mail.ncku.edu.tw

Abstract--Traditional cache invalidation schemes are not suitable to be employed in wireless environments due to the affections of mobility, energy consumption, and limited bandwidth. Cache invalidation report (IR) is proposed to deal with the cache consistency problem. However, the main drawback of IR-based schemes is the long latency of data access because the mobile hosts (MHs) need to wait next IR interval for cache invalidation when the cache hit happens. In this paper, we propose a Dynamic Invalidation Report (DIR) to reduce the latency of data access when the MHs query data. DIR contains an early cache validation mechanism by utilizing the validation messages. Therefore, the MHs can verify their cached data as soon as possible. Next, we design a predictive method to dynamically adjust IR interval to further reduce the latency called DIR-AI (DIR with Adjustable Interval) scheme. Finally, we evaluate the performance of the DIR and DIR-AI and compare them with the existing invalidation report schemes by using NS2 (Network Simulator). The experimental results show that DIR reduces averagely 54.3% and 34.3% of latency; DIR-AI reduces averagely 57.35 and 38.6% of latency compared with TS (TimeStamp) and UIR (Updated IR) schemes respectively.

Keywords: cache consistency, dynamic, invalidation report, latency, wireless networks.

1. Introduction

In recent years, the demand for advanced technologies of mobile computing is growing up significantly. Many wireless communication protocols provide different services for accessing resources in the wireless networks. People can obtain their desired application service from Internet at anytime and anywhere. Figure 1 shows a typical wireless architecture. Mobile Support Station (MSS) or called Base Station (BS) is a powerful hardware with large transmission range for communicating with mobile hosts (MHs). A server is a service component connected to MSS. MHs are mobile users which carry devices such as notebooks, PDA's, or cell phones who would like to get information from MSS. A client is a process residing in MH that cooperates with server. A database connected to MSS stores data items needed by server and clients. The main task of a server is to respond the requests from clients, retrieve data from database, and send data to clients.

In most data access model (client-server model), data items are delivered in an on-demand basis. It means server only responses the request from the clients. When the client-server model is applied in the wireless networks, MSS broadcasts data via the wireless channel to all MHs within its transmission coverage. Therefore, broadcast scheme is frequently used by many wireless communication protocols for data access. A broadcast

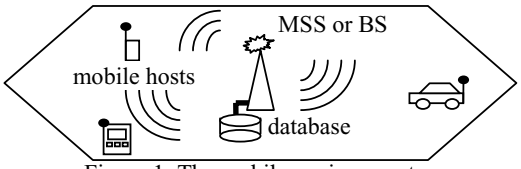

Figure 1: The mobile environments.

data delivery system uses one or more servers to satisfy the requests of users. We first introduce two basic architectures for broadcast delivery systems: push and pull [7]. Then we present the advantages of combining cache techniques with the broadcast system and the challenge of solving the cache consistency problem in the wireless networks.

In a push-based system, data transmission is independent of the actual request patterns. The action of transmitting data is initialized by the server. Clients only wait for their requested data passively. Server broadcasts data to clients continuously and repeatedly. When a user requests a data item, it needs to wait until the requested data item is broadcast by the server. In *broadcast disk access model* [1], the server broadcasts all data items repeatedly. The broadcast channel can be thought of as a disk (broadcasting repeatedly is like disk spinning). MHs can get their desired data by listening broadcast channel in a certain time interval as shown in Figure 2. The fundamental optimization problem is to design broadcast programs for the server such that we can minimize the average waiting time of accessing a data item.

In a pull-based system, the specific data item requested by a user can be observed by the servers. The action of transmitting data is initialized by the client. The servers are always waiting for clients' requests and ready to respond the requested data to clients as shown in Figure 3. The servers know the exact number of pending requests for each data item. We first distinguish the differences between pull-based broadcast systems and the traditional on-demand pull systems. The traditional on-demand pull scheme is a unicast communication protocol. The requested data is sent through the routing path between the source and destination. The data can only be received at the source. However, in the pull-based broadcast system, MSS sends data by wireless channel. The MHs in the transmission range of MSS can share the wireless channels and receive the broadcast data. The fundamental optimization problem is to design a scheduling algorithm for the servers so that we can minimize the average response time.

The main advantage of broadcast scheme is to save more request uplink bandwidth and power consumption of MHs. When a data item (hot data) is queried by many

1550-445X/08 $25.00 © 2008 IEEE
DOI 10.1109/AINA.2008.118

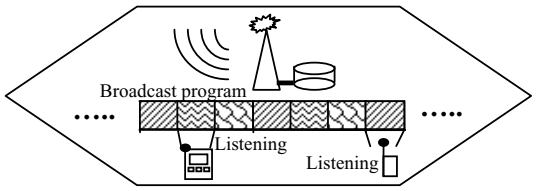

Figure 2: Push-based data delivery model.

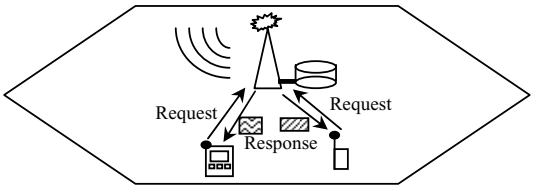

Figure 3: Pull-based data delivery model.

requesters, this data item is only sent once and can be shared by others.

Caching is an efficient technique for increasing the performance in data delivery model. The original idea of caching is that the data accessed by MHs has the properties of temporal and spatial locality. Higher temporal and spatial locality ensures that most accesses will go to the data that were accessed recently in the past and this data resides in cache. It has been widely used in different fields such as CPU design, multi-processor, memory architecture, or router design. Furthermore, Internet uses caches in proxy servers and cooperative caching architecture to reduce the network traffic and average latency of data query significantly. Therefore, the caching techniques are also proposed to combine with the broadcast data delivery systems [2][5][6] for improving the performance of data access.

In wireless networks, D. Barbara and T. Imielinski first proposed a cache invalidation scheme (called IR scheme) in which server periodically broadcasts an invalidation report (*IR*) [2]. Clients only listen to the wireless channel for receiving invalidation report pushed by the server otherwise it is operated in the sleep mode for saving power as shown in Figure 4. Each IR contains data updated information during the certain time interval. Thus, when a client reconnects with the server, it can validate its cache by next IR messages. Indexing technique [3] is also proposed to achieve the effective access of invalidation reports.

The advantages of IR scheme are scalability and energy efficiency. The IRs and data items can be shared by all clients listening to the broadcast channel. MHs can operate in sleep mode most of the time and only be active when the server broadcasts the invalidation report. However, a main drawback of IR-based approaches is the large query delay involved since clients require the current IR to ensure cache consistency. And if a data item is judged to be invalid after examining the invalidation report, the client takes a long waiting time to get the fresh data item back.

In this paper, we propose an improved scheme called *Dynamic Invalidation Report* (DIR) to reduce the data access latency for MHs. DIR contains an early *cache validation mechanism* so that MHs and server can

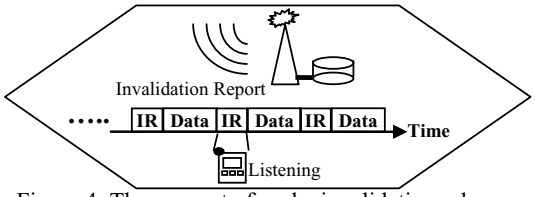

Figure 4: The concept of cache invalidation scheme.

communicate with each other by utilizing the validation messages. Therefore, MHs can verify their cached data as soon as possible. We also define a parameter called virtual hit ratio to record the number of cache hits from the cache validation mechanism. Based on the virtual hit ratio, we design a predictive method to dynamically adjust IR interval to further reduce the latency called DIR-AI (DIR with Adjustable Interval) scheme.

The rest of the paper is organized as follows. Section 2 presents the related works. Section 3 describes the proposed dynamic cache invalidation schemes. The results of performance evaluation are shown in section 4. We conclude the paper in section 5.

2. Related works

2.1 Cache consistency problem

Although caching technique can provide additional performance improvement for data access, it always needs to deal with the cache consistency problem when the original data is updated by clients or servers. If not, MHs may use the invalid cached data and cause the un-predictive conditions. This problem also arises in the broadcast systems.

In a pull-based broadcast system, if a client tries to write the remote data item in the server, it sends a message of write operation to the server by uplink channel. The server performs the sequence of these operations in order, and schedules all requests including write operation and read operation. The new contents of updated data have to be broadcast after the write operation is done, or it will result in stale broadcasting and cache inconsistency.

In a push-based broadcast system, if a client tries to update a data, it first updates its cache, then sends the update message via uplink to server. After receiving the update message, the server writes new content of the data to its local cache and database, schedules data, and disseminates data in next broadcast cycle. If any other update messages come before scheduling, the data item would be changed to latest updated value. The clients will receive data item with value not to their wish in next cycle. Therefore, the assumption is given that data items only can be updated on the server side, and cache consistency can be kept by cache invalidation messages. These massages are sent by the server, and each client uses them to remove invalid data from its own cache.

2.2 Invalidation Report

In general, there are two kinds of IR schemes, stateless and stateful. In the stateless IR schemes [2][6], server is not aware of the states of each entry in MHs' data cache. The states indicate whether each entry of

cached data is valid or not. This scheme is scalable to operate invalidation process of MHs. Because MHs can only wake up in each IR interval to receive the IR message, much power consumption is saved by MHs.

In the stateful schemes [8], server has the knowledge of status of cached data in MHs. When a data object is updated, server can immediately send the invalidation report messages to MHs. MHs do not need to wait for the next IR interval to verify the cache. Compared to the stateless schemes, the stateful schemes can reduce the latency of the data objects. But server needs to maintain complex data structures for the states of the MHs' data cache. Also, MHs will waste power in active mode in order to wait for the IR message at anytime. In wireless networks, the power concern is the most important consideration. Therefore, we focus on the discussion of stateless schemes and briefly present two popular schemes as follows.

2.3 TimeStamp (TS)

The TS [2] algorithm is a stateless IR-based invalidation method. The server generates IRs and broadcasts them periodically at time $T_i = iL$ (i is a time index; L is an IR cycle). An IR is a history that records the data items that have been changed in the last w seconds, $L \leq w$. When an MH receives an IR, for each data item in its cache, said j, it compares the timestamp of $j(t_j^c)$ in the cache to the timestamp of $j(t_j)$ in the invalidation report. If $t_j^c \leq t_j$, the data item j will be removed from the cache.

MHs record a query list $Q_i = \{ j \mid j$ has been queried in the interval $[T_{i-1}, T_i] \}$. When an MH receives an IR, it compares t_j^c with t_j. If some queried data items are valid, they are returned to the requester (i.e., $t_j^c \geq t_j$) and the request is removed from the query list. The remaining query list will be requested via the uplink after all actions are done. The MH also keeps a variable T_l that indicates the last time it received an invalidation report. If the difference between current report timestamp and T_l is bigger than w, the entire cache is dropped.

2.4 UIR (Updated IR)

The UIR algorithm [6] concentrates on reducing the waiting latency and is independent of the cache invalidation strategies combined with IR-based algorithms to deal with long disconnection problem. UIR uses a technique similar to the (1,m) indexing [3] to reduce the query latency. In this strategy, the server broadcasts an invalidation report every L time units. UIR also broadcasts a smaller version of the invalidation report, called *Update Invalidation Report*, every L/m time units, where m is the number of times the smaller reports will be broadcast in each interval. Because the IR contains the update history of the past w broadcast seconds, replicating the complete IR m times will consumes large bandwidth and client's power.

The common drawback of TS and UIR is the long latency when the clients query the data item and the local cache returns the cache hit. It needs to wait next IR interval to check the data consistency as Figure 5 and Figure 6.

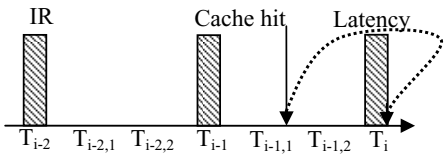

Figure 5: Latency of cache hit in TS scheme.

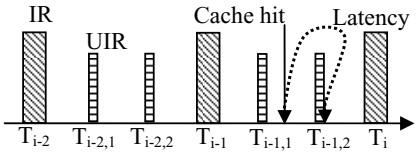

Figure 6: Latency of cache hit in UIR scheme.

3. Proposed Dynamic Invalidation Report Scheme

3.1 Motivation

In a wireless environment, many new factors affect the performance of cache invalidation scheme.

1. Limited bandwidth: The bandwidth in wireless networks is much less than that in wired networks. If the clients always want to keep the cache consistency with the server, the bandwidth will be consumed more due to the communication of the cache consistency messages.

2. Mobility and Disconnection: When an MH moves out of the transmission range of the server, it will disconnect from the server. Thus, it can not receive any invalidation messages during the disconnection time. When an MH reconnects to server, its cache might be inconsistent because it loses invalidation messages sent by server. Therefore, cache invalidation algorithms should consider the problem of reconnecting operation.

3. Power consumption: The MHs can always verify their cache consistency with the server at anytime. However, it will waste more power consumption which is the most importance concern in wireless networks. Thus, in cache invalidation scheme, reducing power consumption is always a goal for MHs.

3.2 Assumptions

We utilize the pull-based on-demand schemes and push-based broadcast schemes to develop a dynamic invalidation report scheme in order to obtain their performance advantages. We assume they are mixed with each other in our schemes. Server broadcasts hot data repeatedly and continuously and transmits some cold data on an on-demand basis if it is a pull-based service. The typical usage of uplink is to request data objects when cache miss happens. All the requests will be observed by the server. The server uses the requests to generate statistic information to schedule data items in the push situation. We also assume that the update operations only act on the server side for simplicity. The server is stateless but it can exactly know the number of requests at anytime.

Invalidation reports are broadcast by the server as in TS scheme. The contents of IRs are the updated information for pass w seconds. And a timestamp is

associated with each data item in IRs. The contents of requested data items that have been requested in last IR interval are continuously broadcast after broadcasting the IRs. This assumption simplifies the problem and not to discuss the scheduling algorithm. When clients receive the IRs, they decide which data have to be moved from their local cache, and compare the timestamps of local cached item with the timestamps in IRs. The bigger value of the timestamp, the fresher the content of the data object. Stale cached items will be removed. If a client doesn't receive an IR over w seconds, all of its cached items will be thrown.

3.3 Early Cache Validation Mechanism

In our proposed Dynamic Invalidation Report (DIR) scheme, the main idea is that the cache-hit data from the local cache can be used as soon as possible. In most IR-based schemes, it takes a long latency to check the consistency of the cached data. Therefore, we design an early cache validation mechanism by using early validation messages to reduce the long latency. This mechanism is performed when an MH requests a data item and responses a cache hit from its local cache. When a client gets a cache hit, it uses the uplink to transmit an early validation request message to the server. After receiving this message, the server decides the request data item is valid or not, and returns an early validation response message immediately. The early validation messages contains early validation request message (*EREQ*) and early validation response message (*EREP*).

The MHs receive the *EREP* from the server to check the cache hit data is a valid or not. The usage of the early validation messages reduces the latency problem obviously. For a hit data item, it might be used immediately after an early validation message is sent by server. Although it increases one uplink usage when the cache hits, the pending query could get response immediately. If the cached item is valid, it can be used without any further delay. If the cached item is invalid, the client would use the uplink to request right away. The TS scheme has another drawback in this situation. The uplink traffic will be burst for those many requests that are sent almost simultaneously when the clients receive IR and the cache-hit data is valid.

Using *EREQ*/*EREP* messages can separate the requests and also avoid the uplink congestion. Instead of the value of data, the structure of *EREP* message just contains a valid bit. Therefore, for power consumption, the overload of receiving an *EREP* is much less than receiving an IR messages. The original invalidation mechanism does not change the data structure of invalidation reports, the character of periodical broadcasting IRs and the way in which the client invalidates its cache. The utility of *EREQ/EREP* messages overlap with the invalidation reports in TS scheme just under cache hit condition. The communication of *EREQ*/*EREP* messages is unicast. So the invalidation report mechanism still works effectively.

The early validation request message ($EREQ_i$) and response message ($EREP_i$) are defined as follows:

(1) When server receives a $DREQ_x$ from a client k
IF d_x is classified as a pull-based service
THEN
Return a $DREP_x$ to the client k
IF d_x is classified as a push-based service
THEN
Queue the request and schedule d_x at the begin in the next IR interval

(2) When server receives a $EREQ_x$ from a client k
Decides that d_x is valid or not by compare the timestamp in the $EREQ_x$ with the last updated timestamp of d_x in the server's database
Return an $EREP_x$ to the client k

(3) When a client queries a data d_x
IF cache hit
THEN
Send an $EREQ_x$ message to the server
IF cache miss
Send a $DREQ_x$ to the server

(4) When a client receives an $EREP_x$ message
IF v_x valid bit is "1"
THEN
Use the cache's value (d_x^c) to answer the query
ELSE
Invalidate d_x^c and send a data request to the server

(5) When a client receives a $DREP_x$ message
Use d_x to answer the request and put d_x into the cache. t_x^c is the timestamp in the $DREP_x$ message

Figure 7: Operations of early cache validation mechanism based on the receiving messages.

$EREQ_i$ = { $[i, t_i]$ | i is the data ID and t_i is a timestamp of data i such that $t_i = \max(t_i^c , T_l)$ } . t_i^c is the timestamp of the data i in the client's cache, and T_l denotes last time the client receives IR.

$EREP_i$ = { $[i, v_i]$ | i is the data ID and v_i is a valid bit that indicates i is valid or not since the timestamp in $EREQ_j$ }

We also define the structure of data request message. The $DREQ_j$ is for a client requesting the data item j and the $DREP_j$ is the response from the server:

$DREQ_j$ = { $[j, t_j]$ | j is the data id and t_j is a timestamp ($t_i = T_l$) }

$DREP_j$ = { $[j, d_j, t_j]$ | j is the data id, d_j is the data object. t_j is the recent updated value of the data j }

Note that the *DREP* messages can only be sent when the requested data item is classified as the cold data. If the requested item is classified as hot data, the server will record the query information, generate statistics, and apply the results to the scheduling algorithms. Figure 7 shows the operations of early validation cache validation

`mechanism according to the receiving message of server and client.

3.4 DIR with Adjustable interval

Based on the early cache validation mechanism, we also propose an improved method to further reduce the latency. For a certain time interval, said *L* (the cycle time to broadcast an invalidation report), the server can maintain counters of the positive *EREP* and negative *EREP* messages (called the "*missed DREP*"), which are sent after cache miss happens. To clarify the description of our scheme, the valid bit of "positive *EREP*" is set to "1" and the valid bit of "negative *EREP*" is set to "0". We define a parameter called "*virtual hit ratio*" (VHR) to be the ratio of the number of positive *EREP* messages to the total amount of three messages. This VHR parameter shows to the server that if the VHR is high, most of clients' queries in pass *L* seconds can finish their process just by sending early validation messages. In contrast, if the VHR parameter is quite low, the clients' queries have to wait for data coming in pass *L* seconds. We expect that the VHR parameter is high. In this case, the invalidation reports may not work very well because most of caches' consistency is kept by *EREQ/EREP* messages. On the other hand, the low VHR indicates that usage of most of *EREQ*s is actually no use, but just a waste of power consumption.

A low VHR arises under two conditions: 1) The update rate is very high on the server. 2) The updated data items were queried in pass *L* seconds. These conditions impact the performance of the early validation mechanism much more than the high hit ratio condition does. Thus, we focus on these conditions and make the server adjusting the IR broadcast interval dynamically. Reducing the IR interval improves the performance because, for a push-based service data, a client has to wait for the next IR. If the next IR comes in quickly, the response time will also be short. The idea is straightforward. We decrease IR interval when the virtual hit ratio is low and increase it when virtual hit ratio is high. Under the low hit ratio situation, reducing the IR interval reduces the latency of cache invalidation. Alternatively, when hit ratio is quite high, adjusting interval actually also reduces the server's load while keeping low latency. Note that we do not discuss how the server schedule IRs. The scheduling problem is out of scope in the paper.

We first formally define the notations and VHR mentioned above:

V_t : the number of "positive *EREP*" in the t_{th} IR interval.

I_t : the number of "negative *EREP*" in the t_{th} IR interval.

M_t : the number of *DREQ* received by the server in the t_{th} IR interval. *DREQ* is the data request message.

PM_t : the number of *DREQ* send after cache miss in the t_{th} IR interval. Thus, $PM_t = M_t - I_t$.

VHR_t : the virtual hit ratio in the server's view (computed in the end of t_{th} IR interval) as Equ. (1)

$$VHR_t = \frac{Vt}{Vt + It + PMt} = \frac{Vt}{Vt + Mt} \quad (1)$$

VHR_i, VHR_{i-1} : defined in section 3.4
LS : the low bound indicates the system load , minimum value of T_i
Mingrep :the minimum guarantee service response time, high bound of T_i
Threshold_h : high threshold of VHR_i
Threshold_l : low threshold of VHR_i
T_i: the time that server construct a IR
(i is an integer that denotes the i_{th} IR interval)

At interval time T_i, server decides to adjust IR interval or not

IF (T_i < SL) or (T_i > Mingrep)
THEN $T_{i+1} = T_i + L_i$ //Not adjust next IR interval
Return

IF VHR_i <= Threshold_l
THEN // decrease next IR interval
$T_{i+1} = T_i + f(VHR_i, VHR_{i-1}, L_i)$

IF VHR_i > Threshold_h
THEN // increase next IR interval
$T_{i+1} = T_i + f(VHR_i, VHR_{i-1}, L_i)$

Figure 8: Operations of adjusting IR interval by server.

The virtual hit ratio indicates the performance of early cache validation mechanism. We expect that the more cache-hit data can be verified as "positive" and used immediately.

Due to the affection of temporal locality, server references the VHRs from the recently IR intervals can obtain better benefit. Thus, we first let the server keeps two *VHR* value from the latest two IR intervals: VHR_i and VHR_{i-1}, VHR_i and VHR_{i-1} are calculated by the server in the end of i_{th} and $(i-1)_{th}$ IR interval respectively. In the end of each IR interval, the server computes VHR and also predicts the (VHR_{i+1}^p) for the next IR interval. The VHR_{i+1}^p is predicted by simply using the concept of exponential average as Equ. (2).

$$VHR_{i+1}^p = \alpha VHR_i + (1-\alpha)\, VHR_{i-1}. \quad (2)$$

The parameter α controls the relative weight of recent hit ratio and past history in our prediction. If $\alpha = 0$, then $VHR_{i+1}^p = VHR_{i-1}$, and recent hit ratio does not included. If $\alpha = 1$, then $VHR_{i+1}^p = VHR_i$, and only the most recent hit ratio VHR_i is included. In general, we can reference the VHR record in each past IR interval from the history and expand the formula Equ. (2) to Equ. (3) for VHR_{i+1}^p by substituting for past VHR as follows:

$$VHR_{i+1}^p = \alpha VHR_i + (1-\alpha)\alpha VHR_{i-1} + (1-\alpha)^2 \alpha VHR_{i-2} + \ldots + (1-\alpha)^j \alpha VHR_{i-j} + \ldots + (1-\alpha)^{i+1} \alpha VHR_0. \quad (3)$$

The VHR_0 is defined as a constant. Both $(1-\alpha)$ and α are less than or equal to 1, so each successive term has less weight than its predecessor.

Due to the temporal locality of cache, some terms of the formula can be cut. If the server considers all pass information since the initial time, it does not match utility and the concept of the invalidation reports since the IRs just contains the updated information in last w seconds. Therefore, we use the hit ratio in pass w seconds to predict the next IR interval. Assume L_i denotes the interval of an IR corresponding to the hit ratio VHR_i. To calculate $VHR_{i+1}{}^p$, the formula is changed as Equ. (4):

$$VHR_{i+1}{}^p = \alpha\, VHR_i + (1-\alpha)\,\alpha\, VHR_{i-1} + (1-\alpha)^2\,\alpha\, VHR_{i-2} + \ldots + (1-\alpha)^c\,\alpha\, VHR_{i-c}, \quad (4)$$

where c is the minimum integer such that $L_i + L_{i-1} + L_{i-2} + L_{i-3} + \ldots + L_{i-c} \geqq w$.

We simply use the ratio of VHR_i to VHR_{i-1} to decide the adjusting rate of last interval. Because VHR_{i-1} is the exponential average in recent w seconds, it is taken as an expected denominator. If the value of VHR_{i-1} is quite low, then server applies L_i as the next IR interval. Otherwise, the server keeps using the old value to broadcast IR. To compute the exact value of next interval, a function is defined as follows:

$$f(VHR_i, VHR_{i-1}, L_i) = \frac{VHR_i}{VHR_{i-1}} \times L_i = L_{i+1} \quad (5)$$

To prevent some excess adjusting, server needs a threshold of low bound and high bound of L_i. The bound can be set according to the history of VHR.

We show an example to explain the association with the latency and the virtual hit ratio. We assume the uplink/downlink bandwidth is 10kbps/200kbps, L_i = 20 second, and the size of early validation messages is 1k bits; the value VHR_{i-1} computed in the $(i\text{-}1)_{th}$ IR interval is 0.5. In the end of the i_{th} interval, the server gets the total query count of all clients is 100, the total count requested early validation message is 40, and computes the VHR_i = 0.2. It means that 80 queries must wait for the i_{th} invalidation report and 20 of them is resulted from an "invalid" early validation message. When ignoring the receiving IR time, the average response time of clients which queried in the i_{th} interval is about $\frac{80 * \frac{L_i}{2} + 40 * (\frac{1k}{10k} + \frac{1k}{200k})}{100}$ = 8.042 seconds. This value is the latency of pure IR with early validation. Then since VHR_i is low, the server decreases the interval applied in the next. $L_{i+1} = L_i * \frac{0.2}{0.5} = 8$. In the end of $(i+1)_{th}$ IR interval, if the value of VHR_{i+1} equals to VHR_i (0.2), and both the count of total queries and total requested early validation messages are the same with the previous interval, the average response time will reduce to $\frac{80 * \frac{L_i}{2} + 40 * (\frac{1k}{10k} + \frac{1k}{200k})}{100}$ = 3.242 seconds. Note that the second term in the denominator is small. So even if both the count of total queries and total requested early validation messages are different, the impact is not large.

Table 1: Simulation parameters.

Definition	Default values
Total data items	1000
Cache size	100 data items
Uplink bandwidth	11Mbps
Downlink bandwidth	11Mbps
Periodic broadcast interval	20 sec
Broadcast window	100s
Data size	16384 bytes
Early validation message size	1024 bytes
Data request message size	1024 bytes
Number of push data	900
Number of pull data	100
Number of hot data (all push data)	120
Number of cold data	880
Probability of access hot data each	80%
α (compute $VHR_i{}^p$ in our scheme)	0.5
LS (low bound of adjusted interval)	10
Mingrep (high bound of adjusted	60
Threshold_h	0.6
Threshold_l	0.5

4. Performance evaluation

4.1 Simulation environment and parameters

To evaluate the efficiency of various invalidation algorithms, we implement the simulation program in NS2 network simulator. A server connects to both the wired and wireless networks, and a database is located inside the server. The database can only be updated by BS while the queries are made on the client side. There are two ways to classify the data items: 1) the hot data and the cold data 2) the push-based service data and the pull-based service data. If a requested data item is in a push-based service, BS will broadcast it after next IR. Alternatively, if a data item is in pull-based service, BS returns the data item. For each request, MHs have a large probability (80 percent) to access a hot data and a low probability (20 percent) to access a cold data. The two classifications do not overlap. Here we assume that all hot data items are in a push-based service. MHs can get the hot data items by just listening to the broadcast channel. Both the updated/requested data IDs and the corresponding event times are generated in an exponential average. Each MH contains a local cache that can store 100 data items. At the initial time of simulation, the cache will be randomly filled with the hot items and the cold items of the ratio mentioned above. Most of the system parameters are listed in Table 1.

4.2 Results

We first measure the latency of requested data under TS, UIR, DIR, and DIR-AI schemes as shown in Figure 9 (a), (b), (c), and (d), respectively. In TS [2] and UIR [5] schemes, a requested data may be gotten after receiving two invalidation reports due to the result of cache consistency. The latency coverage of TS is about 9.9~13.7 seconds, UIR is about 6.8~9.6 seconds, DIR is about 4.2~6.7 seconds and DIR-AI is about 3.6~6.7 seconds. When the query rate decreases, the latency increases substantially. In most situations, the former two algorithms (TS and UIR) have the same deviation such that when the number of clients increases, the latency increases more obviously than DIR and DIR-AI schemes. The reason is that both TS and UIR are constant-interval

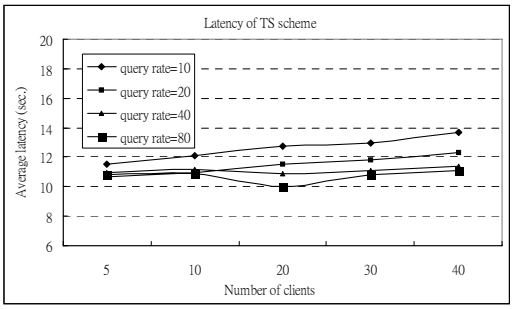

Figure 9 (a): Latency of TS scheme.

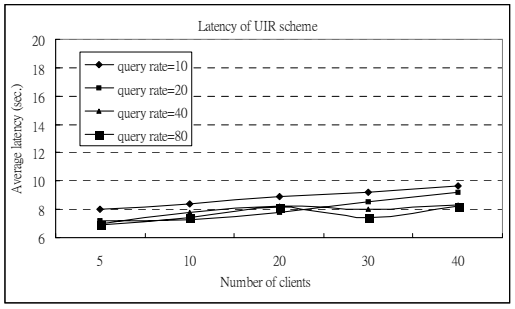

Figure 9 (b): Latency of UIR scheme.

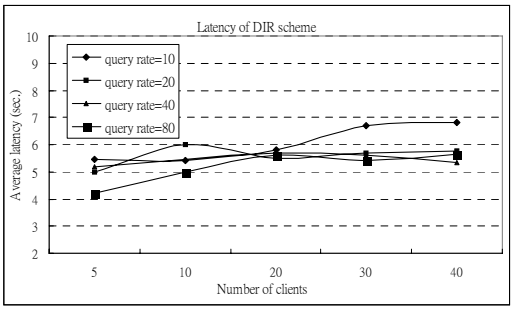

Figure 9 (c): Latency of DIR scheme.

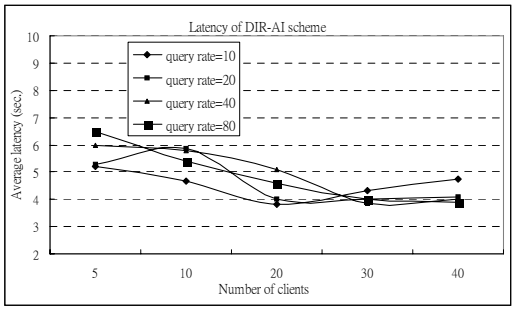

Figure 9 (d): Latency of DIR-AI scheme.

methods. When a client gets a cache hit, it needs to wait for an IR to validate the queried data. So, if the data is invalid, many clients use the uplink at the same time, and these messages become the server's overhead, therefore increases the latency.

Although DIR is also a constant-interval algorithm, it employs the early cache validation mechanism. The queries have not to be pended until the client receives the next IR, so it can reduce the latency obviously. The latency with query rate 10 is suddenly increasing when the number of clients is 20 to 30. This behavior is similar to that in TS and UIR schemes because the overhead of the server increases. In a wireless environment, the bandwidth will decrease when the communication signal becomes week. And this condition can be raised when the communication distant increases or channel conflicts. Even though that is always a barricade, the latency decreases. In the other situation, the latency in DIR is about 5~6.1 seconds.

The performance of DIR-AI is much different from others. We set the low high bounds of the IR broadcast interval that indicate the system load and the minimum guarantee service response time. The low bound depends on the system capacity and the high bound depends on the service type. The low bound is set to 10 while the high bound is 60. By adjusting the interval, the latency is reduced more than the pure DIR scheme in many situations.

In fact, whether an algorithm performs well or not is associated deeply with the value of Threshold_l and Threshold_h. The values of both parameters can be decided by experience when request and update information is enough. At the point when the number of clients is 5 and the query rate is 80, and latency time is just about 6.7 seconds. When there are only 5 clients and the query rate is quite large, the value of VHR_t (defined in section 3) is quite absolute. That is, the sample space is too deficient for the server to make the interval adjustment perform well. Therefore, to use the interval adjusting scheme with 100% efficiency, a premise has to be made that the number of queries in an interval can't be too little. But as the identity of the DIR scheme, the latency is improved even if the interval become large since the hit and valid data could be used immediately by early validation mechanism.

To further evaluate the average response time, we make several experiments with each algorithm. The compared results are shown in Figure 10 (a), (b), (c), and (d). The simulation is made in different combinations where the mean query rate is 20/40. The result shows that the latency reduces by using early validation messages. When the server applies "the virtual hit ratio" to adjusting the interval, the performance is further improved. The main part of the reduced latency is the validation time in DIR while both the validation time and the data getting time is reduced in DIR with adjusting interval scheme. To highlight the performance difference, we choose the mean updated rate of the server to be only 5 and 10. That is, the server updates the data items in the database frequently during the simulation time. This decision is mainly for emphasizing the usage of interval adjusting method of the server, since the TS and UIR schemes are affected more deeply when the mean update rate is high. Actually, if the value of the "virtual hit ratio" (VHR_i) is always stable, the performance of DIR-AI scheme will be close to the DIR. However, in real world, the value of VHR_i might be unstable. Thus, the choice of Threshold_l and Threshold_h play an important role in the algorithm. The decision can be made by server's experience statistics and the value can be changed dynamically according to the system state. To decide the Threshold_l and Threshold_h, we run 10 times of all algorithms that the update rate is 10/20/80/320/640, the query rate is 2.5/5/10 and we get an average VHR_i of about 0.52. So,

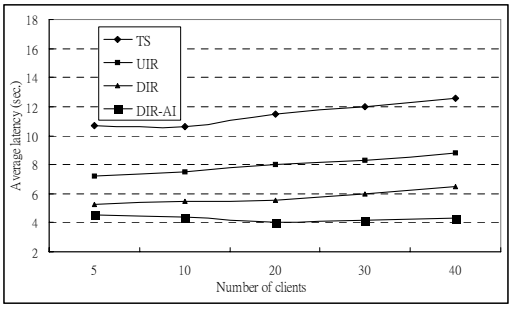

Figure 10 (a): Latency comparison under update rate=5, query rate=20.

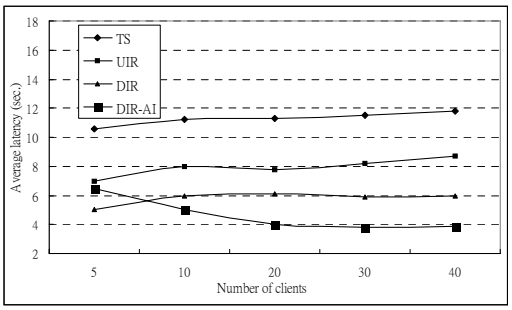

Figure 10 (b): Latency comparison under update rate=5, query rate=40.

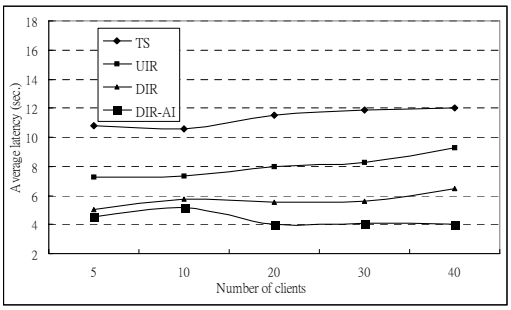

Figure 10 (c): Latency comparison under update rate=10, query rate=20.

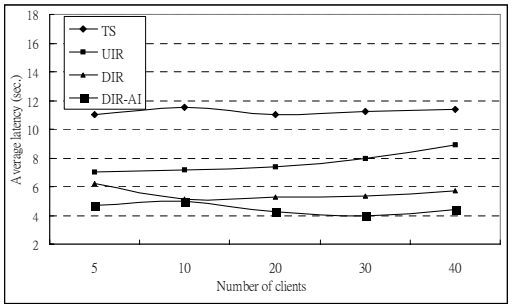

Figure 10 (d): Latency comparison under update rate=10, query rate=40.

the value of Threshold_l and Threshold_h is set to 0.5 and 0.6. In fact, the TS scheme may work well in a pure push-based service environment. But with the advanced wireless techniques, even the push-based service need to add an on-demand based function. Thus using early validation messages is workable to keep cache consistency as fast as possible.

5. Conclusion

In this paper, we propose a dynamic cache invalidation schemes. The main goal is to reduce the long latency in IR-based schemes. The early validation mechanism is employed to make the MH can check the data consistency as soon as possible. Further, when the base station adjusts the IR interval according to the statues of data request, the performance is improved.

References

[1] S. Acharya, R. Alonso, M. Franklin, and S. Zdonik, "Broadcast Disks: Data Management for Asymmetric Communication Environments," *Proc. ACM-SIGMOD, Int'l Conf. Management of Data*, San Jose, pp. 199-210, June 1995.

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

[3] T. Imielinski, S. Viswanathan and B. Badrinath, "Data on Air: Organization and Access," *IEEE Trans. Knowledge and Data Eng*., vol. 9, pp. 353-372, May/June 1997.

[4] J. Jing, A. Elmagarmid, A. Helal, and R. Alonso. "Bit-Sequence: An Adaptive Cache Invalidation Method in Mobile Client/Server Environments," *Mobile Networks and Applications, vol. 31, no. 2*, pp. 115-127.

[5] G. Cao, "A Scalable Low-Latency Cache Invalidation Strategy for Mobile Environments," *IEEE transactions on Knowledge and Data Eng,* vol. 5*,* no. 5 September/October 2003.

[6] G. Cao, "A Scalable Low-Latency Cache Invalidation Strategy for Mobile Environments," *ACM Int'l Conf. on Mobile Computing and Networking (MobiCom)*, pp. 200–209, Aug. 2000.

[7] Joel L. Wolf, Mark S. Squillante, John J. Turek, Philip S. Yu and Jay Sethuraman "Scheduling Algorithms for the Broadcast Delivery of Digital Products" *IEEE transactions on Knowledge and Data Eng,* vol. 13*,* no. 5 September/October 2001.

[8] Z. Wang, S. K Das, H. Che, and M. Kumar, "A scalable asynchronous cache consistency scheme (SACCS) for mobile environments," *IEEE Trans. Parallel Distrib. Syst.*, vol. 15, no. 11, pp. 983-995, 2004.