Wireless Cache Invalidation Schemes with Link Adaptation and Downlink Traffic

Mark Kai Ho Yeung, *Student Member*, *IEEE*, and Yu-Kwong Kwok, *Senior Member*, *IEEE*

Abstract—Providing on-demand data access in client-server wireless networks is an important support to many interesting mobile computing applications. Caching frequently accessed data by mobile clients can conserve wireless bandwidth and battery power, at the expense of some system resources to maintain cache consistency. The basic cache consistency strategy is the use of periodic invalidation reports (IRs) broadcast by the server. Recently, IR-based approaches have been further improved by using additional updated invalidation reports (UIRs) (i.e., the IR+UIR algorithm) to reduce the long query latency. However, the performance of the IR+UIR approach in a practical system is still largely unknown. Specifically, previous results are based on two impractical simplifying assumptions: 1) broadcast traffic is error-free and 2) no other downlink traffic (e.g., voice) exists in the system. The first assumption is clearly unrealistic as signal propagation impairments (e.g., multipath fading) and, hence, packet reception failures are inevitable in a practical situation. The second assumption is also inapplicable in real life because mobile devices are usually multipurposed (e.g., a mobile phone equipped with a browser may be used for Web surfing while having a phone conversation). In this paper, we first study the performance of the IR+UIR approach under a realistic system model: The quality of the wireless channel is time-varying, and there are other downlink traffics in the system. Our simulation results show that query delay significantly increases as a result of broadcast error and the additional downlink traffics experience longer delay due to extended broadcast period. Exploiting link adaptation (i.e., transmission rate is adjusted dynamically according to channel quality), we then propose three schemes to tackle these two problems. Our results indicate that the proposed schemes outperform IR+UIR under a wide range of system parameters.

Index Terms—Cache invalidation, invalidation report (IR), link adaptation, channel adaptive protocols, wireless networks, client-server computing, system design, simulations.

1 Introduction

With the increasing popularity of different mobile devices such as laptops, hand-helds, cellular phones, etc., we are expecting more services to be delivered in a wireless environment. In particular, people are interested in retrieving up-to-date information at any time and any place using their mobile devices in client-server wireless networks [5], [6], [10], [16], [19], [24]. These services include, but are not limited to, financial information, news and weather forecast, traffic schedules and conditions, online shopping catalogs, etc. We need an efficient on-demand data access mechanism to support such services. However, the wireless bandwidth is still severely limited and mobile devices are inevitably battery-operated. Caching frequently accessed data at the client side allows some queries to be served locally. Indeed, caching is widely reckoned as a promising technique to conserve both wireless bandwidth and battery power [4], [22], [18], [19], [20]. However, cached data will eventually become invalid due to asynchronous updates and removals at the data source. As a result, queries cannot be directly served by the local cache. To guarantee correctness, clients should instead ensure the cached data is consistent before using it to serve a query. This relies on the server to inform all its clients to discard any obsolete data in their caches. To efficiently utilize the precious wireless bandwidth, the server only delivers the invalidation information via the broadcast channel, but *not* the contents of the updated data items. This inevitably introduces extra overheads at the server and possibly longer query delay to the clients. It is usually assumed that the processing power of the server is not the bottleneck and the design goal is to achieve cache consistency without a significant increase in query delay observed by the clients.

The basic cache invalidation strategy is the use of *invalidation reports* (IRs) [1]. Specifically, the server periodically broadcasts IRs, each of which indicates those data items that are recently updated at the source. Clients use IRs to keep their caches consistent by discarding any obsolete data. If a query cannot be served locally, i.e., a cache miss, the client issues an uplink query request for the data item. The server aggregates the query requests from all its clients and broadcasts query replies only once every IR broadcast period. In this manner, a query reply can be shared by more than one client via the broadcast channel. The major advantages of this cache invalidation strategy are high scalability and energy efficiency [4]. First, the size of IR is independent of the number of clients. Second, clients can exploit the periodicity of server broadcast to save power in that mobile devices can operate in doze mode most of the time and only activate during server broadcast.

However, if the disconnection time of a client is longer than a fixed period of time, the client should discard its entire cache, even if some of the cached data may still be valid. This issue is addressed in [1], [11], [12]. Another drawback common to IR-based approaches is the large query delay involved since clients require an IR to ensure cache consistency. The addition of updated invalidation reports (UIRs) to the original IR-based approach (IR+UIR)

• *The authors are with the Department of Electrical and Electronic Engineering, The University of Hong Kong, Pokfulam Road, Hong Kong. E-mail: khyeung@eee.hku.hk and ykwok@hku.hk.*

Manuscript received 20 Aug. 2003; revised 23 Dec. 2003; accepted 27 Jan. 2004; published online 1 Dec. 2004.
For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number TMC-0129-0803.

[2], [3], [4] aims to tackle this problem. In IR+UIR, the server broadcasts a number of UIRs between successive IRs. Specifically, each UIR only contains information about the most recently updated data since the last IR. As a result, the use of UIRs requires that the cache is consistent up to the last IR. Clients use UIRs to keep their cache consistent to the source. Thus, for cache hits, there is no need to wait for the next IR and query delay is further reduced.

Unfortunately, previous research results on wireless cache invalidation are based on two simplifying assumptions: 1) the broadcast channel is error-free and 2) there is no other downlink traffic in the system. As such, the applicability of previous approaches under a realistic environment, where the above assumptions are usually invalid [16], [25], is largely unknown. Indeed, the first assumption is clearly unrealistic as signal propagation impairments [17] (e.g., multipath fading due to user mobility, long term shadowing due to terrain configurations, interferences due to other wireless devices, etc.) and, hence, packet reception failures are inevitable in a practical situation. The second assumption is also inapplicable in real life because mobile devices are usually multipurposed (e.g., a mobile phone equipped with a browser may be used for Web surfing while having a phone conversation). In this paper, based on extensive simulations, we present our study on how the IR+UIR approach performs under such a realistic environment, in which other downlink traffic exists and a *channel adaptive* physical layer is incorporated (a commonly used facility in many practical wireless systems, such as UMTS, IEEE 802.11x WLAN, etc. [8]). Link adaptation is an effective strategy [9], [15] which works by dynamically adjusting the transmission rate through varying the amount of forward error correction (FEC) bits included in a packet, according to the channel quality.

First, we find that the performance of the IR+UIR algorithm critically depends on the integrity of IRs. Second, fixed-rate broadcast (i.e., the channel adaptation facility is not used) is clearly undesirable in that it leads to either 1) significant underutilization of the wireless channel or 2) severe transmission errors. Furthermore, despite the fact that IR+UIR improves query delay by minimizing the delay in cache hits, the penalty due to cache miss is still large. In view of these problems, we also propose three different schemes for enhancing the cache invalidation strategies. Our simulation results indicate that our proposed schemes can reduce: 1) the effect of broadcast error to IR+UIR and 2) the impact of broadcast overhead on other downlink traffic.

The rest of this paper is organized as follows: Section 2 provides some preliminaries on the cache invalidation problem and gives a brief review on some representative approaches. The system model used in our study is described in Section 3. In Section 4, we explain our three proposed cache invalidation strategies. Simulation results are presented and discussed in Section 5. Finally, we provide some concluding remarks in Section 6.

2 Background

2.1 Issues in Cache Invalidation

To support data applications in a client-server based wireless environment, such as a cellular network, it is crucial to provide short-delay on-demand information access services. Due to the unavoidable channel quality deteriorating effects—fast fading and shadowing [17], the communication channels between the base-station (server) and its mobile terminals (clients) are error-prone with time-varying effective bandwidth. Moreover, battery-powered clients may voluntarily switch to doze mode to prolong the operation time; or may involuntarily be disconnected due to poor channel conditions. In other words, communication links between the server and clients are only temporary and subject to transmission errors. Caching frequently accessed data on the client side is an effective technique to mitigate these limitations. In particular, caching can: 1) reduce the query latency due to cache hits, 2) conserve power in uplink transmissions, and 3) better utilize the wireless channel via broadcast. However, cached data items will eventually become invalid due to updates and removals at the source. A cache invalidation strategy is employed to maintain consistency among clients' caches such that no obsolete data would be used to serve queries.

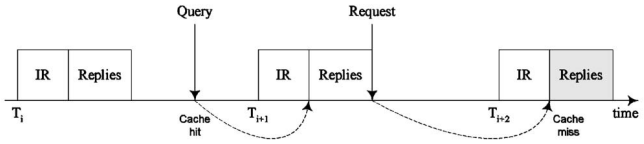


Fig. 1. The IR cache invalidation approach.

2.2 Existing Cache Invalidation Strategies

Existing cache invalidation strategies can be broadly classified into two types: stateful and stateless servers, distinguished by whether clients' cache information is kept at the server or not.

Barbara and Imielinski [1] have proposed the basic stateless server approach. The server broadcasts an invalidation report (IR) every L seconds, indicating which data items are updated in the last ωL seconds, where ω is the broadcast window size. Formally, each IR contains the current timestamp, T_i, and a list of pairs (d_x, t_x) such that $t_x > (T_i - \omega L)$, where t_x represents the most recent timestamp of the data item, d_x. To answer a particular query, a client is required to wait for the next IR to determine whether its cache is valid or not. If a valid cached copy exists, the query can be served locally; otherwise, the client issues an uplink query request to the server and waits for the reply in the next IR broadcast period. For example, in Fig. 1, a new query arrives at a time instance between T_i and T_{i+1}. If the client has a valid copy in its cache, the query can be served locally after the IR broadcast at T_{i+1}; otherwise, an uplink request will be issued to the server. The client is expected to obtain the requested data item in the IR broadcast period at T_{i+2}. Advantages of IR-based approaches include high scalability and energy efficiency: The size of each IR is independent of the number of clients, and IRs are scheduled to be broadcast periodically. As such, clients can switch to doze mode operation between successive IRs to save battery power. The major drawbacks, however, are: 1) clients must flush their entire caches after long disconnection ($> \omega L$), even if some of the cached items may still be valid. Since each IR contains only invalidation information in the last ωL seconds, a client cannot tell whether its cache is valid or not beyond that period; 2) clients must at least wait for the next IR before answering a query to ensure consistency, which potentially causes large query delay.

Various approaches have been proposed to address the long disconnection problem [1], [11], [12]. Jing et al. [12] designed the bit-sequences to replace the original IRs.

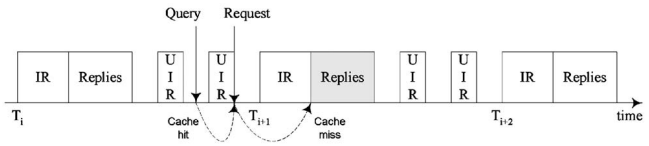


Fig. 2. The IR+UIR cache invalidation approach.

Similar to an IR, each bit-sequence contains invalidation information about the database, but without the time constraint (i.e., ωL). This approach is able to handle clients with arbitrary long disconnection time. However, the assumption is that the update rate to the database is not high. Hu and Lee [11] proposed an adaptive scheme in that the server either broadcasts IRs or bit-sequences in different situations to take advantage of both approaches, which would perform better in a more dynamic environment.

To reduce the long query delay associated with the basic IR cache invalidation approach, Cao [4] has done some pioneering work in that he proposed the addition of Updated Invalidation Report (UIR) to the original IR scheme (IR+UIR). Each UIR contains invalidation information for those data items that have been updated since the last IR. Formally, a UIR consists of the timestamp T_i of the last IR and a list of pairs (d_x, t_x) such that $t_x > T_i$. In other words, UIRs reflect the most up-to-date view of the database. Clients use UIRs to invalidate their cache and either answer queries locally for cache hits or, otherwise, issue uplink requests. As shown in Fig. 2, the same query arrives at a time instance between T_i and T_{i+1}. For cache hit, the query can be served locally after the next UIR broadcast; otherwise, the client issues an uplink request to the server and waits for the reply after the next IR at T_{i+1}. In this manner, IR+UIR reduces the average query latency at the expense of slightly more broadcast overhead in UIRs [2], [3], [4]. However, the use of UIRs during the time period between T_i and T_{i+1} assumes data items in the cache are consistent up to time T_i. This condition is satisfied if the last IR is correctly received, i.e., IR at T_i. Due to disconnection problems or erroneous transmissions, some clients may miss this last IR. They cannot use any of the successive UIRs before receiving the next correct IR at some later time. Equivalently, these clients are operating in the basic IR invalidation approach for that period. Thus, the effectiveness of the IR+UIR scheme largely depends on the integrity of IRs.

Yin et al. [22], [23] proposed a power-aware prefetching scheme based on the IR+UIR approach. The goal is to save clients' battery power expended in downloading data items. In the IR+UIR approach, clients cache all the replies broadcast by the server. In their proposed scheme, however, clients only prefetch the most important data items to their local cache. A *d*-value function is designed to quantify the importance of the broadcast replies, in which the current energy level of a client is taken into consideration. In addition to the energy factor, the *d*-value function also captures other parameters, including update rate, query rate, and data size. Simulation results show that this power-aware prefetching scheme is effective in prolonging the overall system runtime.

Tan et al. [18] presented a comprehensive survey on this topic. They reviewed some design issues in different existing strategies. Specifically, they proposed several cache invalidation strategies that exploit selective tuning to reduce power consumption in downloading data from server.

Kahol et al. [13] proposed a stateful server approach, in which a Home Location Cache (HLC) is responsible for keeping clients' cache status. A client needs to inform its HLC before storing any data item in the local cache. The server sends an invalidation report whenever data is updated. However, this involves a higher complexity at the server side. Besides, a client should listen to the wireless channel all the time whenever it is connected, which also consumes a considerable amount of power [7].

Most importantly, these previous results were based on two simplifying assumptions: 1) the broadcast channel is error-free and 2) there is no other downlink traffic (e.g., voice service as in a cellular network) in the system. Thus, the performance of these approaches is largely unknown in a realistic environment, in which the above assumptions are unfortunately invalid. In this paper, we study: 1) the impact of error-prone broadcast on the IR+UIR approach and 2) the effect of broadcast overhead on other downlink traffic.

3 System Model with Adaptive Physical Layer

In this section, we describe the system model used in our simulations. For the detailed theoretical analysis, please refer to [21].

3.1 Frame Structures

To study the channel characteristics and the effect of other downlink traffic on existing cache invalidation strategies, we employ the system model with an adaptive physical layer presented in [14]. This model is based on the concept of exploiting the synergy between physical layer and medium access control, instead of strictly following the traditional layered approach in protocol design. Fig. 3 shows the uplink and downlink frame structures. In the uplink, a frame is divided into three subframes: request subframe, information subframe (i.e., data payload), and the pilot symbol subframe. Similarly, a downlink frame is divided into four subframes: acknowledgment subframe, polling subframe, information subframe (i.e., data payload), and announcement subframe. There are N_r slots in the request subframe for accommodating access requests. The information subframe is further partitioned into N_i information slots, for transmitting voice or data packets. The capacity of an information slot ranges from 50 to 500 bits (see Table 1). If a data packet cannot be accommodated in an information slot, the packet will then be fragmented and transmitted over several slots. The schedule of the information slots is announced in the announcement subframe. There are N_b slots in the pilot symbol subframe. The functions of pilot symbol subframe and polling subframe are discussed in Section 3.2 below. The frame duration is 2.5ms[1] for both uplink and downlink. N_r, N_i, and N_b are chosen to be 13, 8, and 12, respectively [14]. If a voice user enters a talkspurt, it contends for the uplink information slots by sending requests using the request minislots [14]. If the request is successful, the server

1. It should be noted that such a "short" frame duration is practicable. Indeed, in an IEEE 802.11x-based wireless LAN system, the typical MAC layer frame duration is 0.8ms.

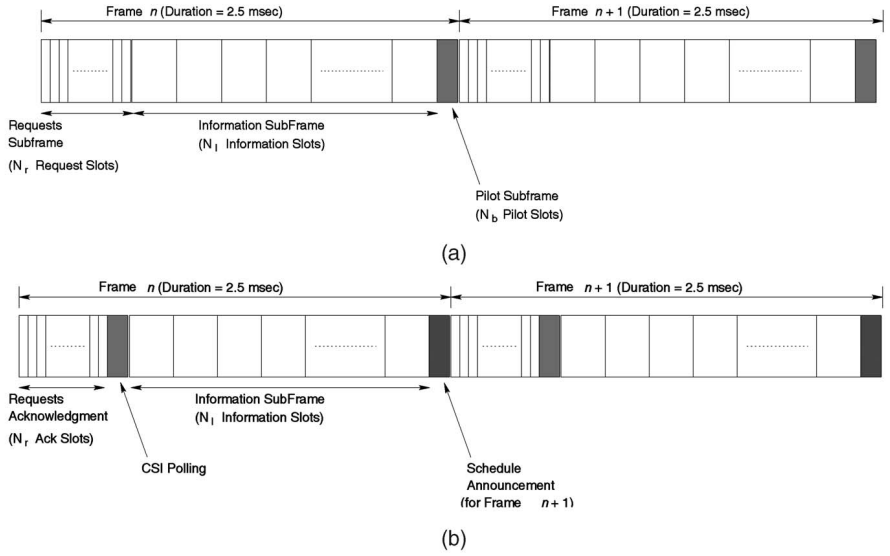


Fig. 3. Frame structures. (a) Uplink and (b) downlink.

reserves information slots in both the uplink and the downlink frames.

3.2 Variable Throughput Physical Layer

The wireless link between a mobile device and the base-station is characterized by two signal propagation components, namely, the *fast fading* component and the *long-term shadowing* component [17]. Fast fading is caused by the superposition of multipath components and is therefore fluctuating in a very fast manner (on the order of a few milliseconds). Long-term shadowing is caused by terrain configuration or obstacles and is fluctuating only in a relatively much slower manner (on the order of one to two seconds). The combined channel fading $c(t)$ is given by $c(t) = c_l(t)c_s(t)$, where $c_l(t)$ and $c_s(t)$ are the long-term and short-term fading components, respectively. In this paper, we assume that the mobility of the users is of pedestrian speed only (i.e., $<$ 5km/hr). Thus, the *coherence time* T_c [17] (time separation between two uncorrelated fading samples) is about 60ms. Most importantly, that means the channel quality stays constant within a frame duration (i.e., 2.5ms). Fig. 4a shows a sample of measured channel fading, i.e., $c(t)$. This is used as the channel state information (CSI) at the transmitter for varying the instantaneous throughput. Since mobile devices are scattered geographically across the system and are moving independently of each other, we assume the channel fading experienced by each mobile device is independent of one another.

To exploit the time-varying nature of the wireless channel, a variable-throughput channel-adaptive physical layer is employed. The receiver estimates the CSI and informs the transmitter via a low-capacity *feedback channel*, a facility commonly provided in practical cellular systems (e.g., cdma2000 [8]). In this system model, the base station obtains the CSI by polling a particular client via the polling subframe. The client then responds with a known pattern in the pilot subframe. Based on the feedback CSI from the receiver, the level of redundancy and the modulation constellation applied to the information packets are adjusted accordingly by choosing a suitable transmission mode. A six-mode[2] (mode-0 to mode-5) variable-throughput adaptive bit-interleaved trellis coded modulation[3] (ABICM) [15] is used. Transmission modes with *normalized throughput*[4] varying from $\frac{1}{2}$ to $\frac{5}{6}$ are available depending on the channel condition.

The ABICM performs "burst-by-burst" adaptation based on the CSI measurement on the physical layer. Specifically, when the channel condition (CSI) is good (fading attenuation is small), the physical layer employs high order modulation and high rate error correction coding so as to boost the instantaneous throughput. On the other hand, when the channel condition is poor, the physical layer employs low-order modulation and low rate error correction coding so as to better protect the packet transmission at the expense of lower throughput. In fact, due to the salient performance in terms of system utilization, such channel adaptation techniques have been widely employed in practical systems such as 3G1x, UMTS-HSDPA, and wireless LAN systems [8].

As mentioned above, since the coherence time of short-term fading is around 60ms which is much longer than an information slot duration, CSI is approximately constant within a frame and it follows that the transmission mode for the whole frame is determined only by the current CSI level. Specifically, transmission mode q is chosen if the feedback

TABLE 1
Performance of the Variable Throughput Physical Layer

Transmission Mode	Constellation	Transmission Rate (Kbps)	Target Bit Error Rate	Data Payload Capacity (bits)
0	BPSK	160	0.00002	50
1	QPSK	320	0.0001	100
2	8PSK	640	0.0005	200
3	16QAM	960	0.002	300
4	32QAM	1280	0.007	400
5	64QAM	1600	0.05	500

2. The number of modes available depends on the number of different modulation/coding methods used.

3. We use the ABICM scheme for illustration purposes only and our proposed cache invalidation schemes can also be used with other channel adaptive physical layer designs such as MQAM [9].

4. Normalized throughput refers to the number of information bits carried per modulation symbol.

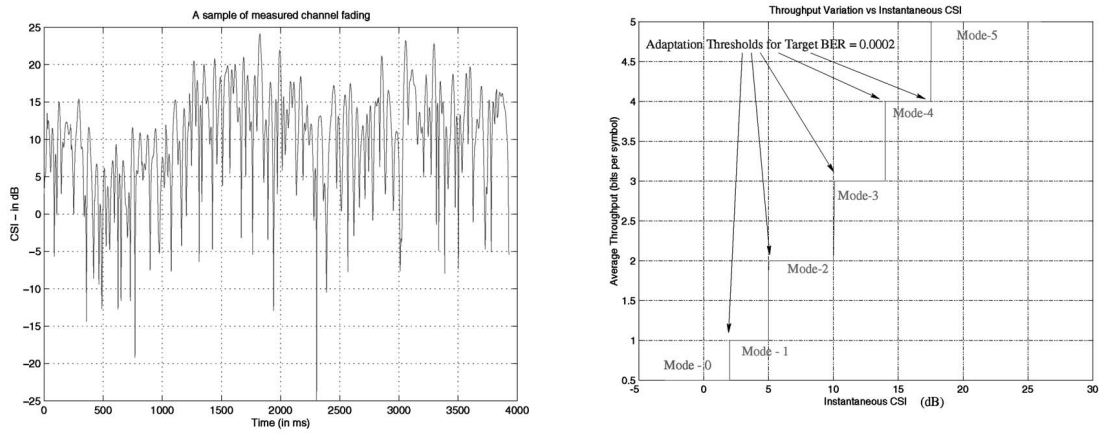


Fig. 4. CSI and throughput of ABICM scheme. (a) Measured CSI: fast fading superimposed on long-term shadowing. (b) Instantaneous throughput versus CSI.

CSI, $\hat{c}$, falls within the *adaptation thresholds*, (ζ_{q-1}, ζ_q) [15]. The adaptation thresholds are set optimally to maintain a target transmission error level over a range of CSI values as illustrated in Fig. 4b. When the channel condition is good, a higher mode could be used and the system enjoys a higher throughput. On the other hand, when the channel condition is bad, a lower mode is used to maintain an acceptable error level at the expense of a lower transmission throughput. The performance details of the variable throughput physical layer are depicted in Table 1.

4 Proposed Cache Invalidation Strategies

In this section, we describe our three proposed cache invalidation strategies targeted at: 1) reducing the probability of corruption in IRs, 2) improving the broadcast channel utilization, and 3) reducing the average delay in other downlink traffic, respectively.

4.1 System Architecture

Fig. 5 depicts a typical cellular network, which adopts the system model with adaptive physical layer presented in Section 3. In addition, base-stations are connected to a common database via a wired network. Each base-station (server) serves the users within its covered area (cell). There are two types of users in the system, namely: voice and data. A voice session requires r_{speech} Kbps. Each data user (client) is making a file transfer session at r_{file} Kbps and also generating a stream of exponentially distributed read-only requests with mean arrival time T_q. The database is divided into hot and cold sets. Updates to each data item follow an exponential distribution with mean arrival time T_u, from which hot items have an update probability of p_u. We assume that each server has a consistent view on the common database and broadcasts the same set of IRs and UIRs. Thus, it suffices to maintain cache consistency among all clients within one cell. As in most previous work [4], [11], [18], [23], we adopt the latest value consistency model, i.e., the most recent value of a data item is used to serve a query. Apart from the server and clients, in our approach, another entity is introduced: the broadcast scheduler, which determines, from Table 1, the transmission rate for the broadcast traffic (see Section 4.3).

4.2 Proposed Scheme 1: Reducing the Probability of Corruption in IR

Existing cache invalidation strategies broadcast each IR in a single packet. However, this is vulnerable to transmission errors. Dividing the IR into a number of segments, with each segment separately transmitted, can reduce both the corruption probability and power consumption.

Server broadcasts invalidation reports (IRs) every L seconds. Each IR contains the current timestamp (T_i) and a list of pairs (d_x, t_x), where d_x is the id of a data item and t_x is the timestamp at which the data item is last updated. To support a long disconnection period, data items that are updated in the past ωL seconds are included in each IR, i.e., $t_x > (T_i - \omega L)$, where ω is the broadcast window size. The server also broadcasts $(m-1)$ updated invalidation reports (UIRs) between successive IRs, where $(m-1)$ is the UIR replicate times. Each UIR indicates those data items that are updated since the last IR. Clients make use of IRs and UIRs to keep their caches consistent by discarding obsolete data such that some queries can be served locally. However, the use of UIRs requires that the cache is consistent up to the last IR. If a client misses the last IR, subsequent UIRs before the next IR cannot be used. Thus, the effectiveness of UIRs depends on the integrity of each IR.

Depending on the update arrival rate to the database, an IR may include a large number of id-timestamp pairs. It is crucial to ensure that clients can successfully receive an IR. However, broadcasting the entire IR at once over the error-prone wireless channel is inefficient due to the fact that such a long transmission is highly vulnerable to channel errors,

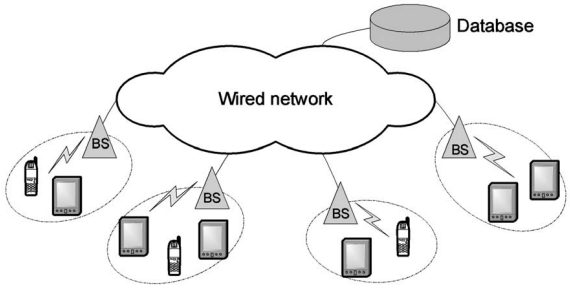


Fig. 5. Cellular network with voice and data users.

Algorithm 1 IR Partitioning

1: INPUT: D (the set of data items); d_x (id of a data item); t_x (last update timestamp); L (broadcast interval); ω (broadcast window); T_i (the current broadcast time);
2: OUTPUT: IR_j for $j = 1 \ldots \omega$ (IR segments);
3:
4: **for** j = 1 to ω **do**
5: $\text{IR}_j = \{\langle d_x, t_x\rangle | (d_x \in D) \wedge (T_i - jL < t_x \leq T_i - (j-1)L)\}$;
6: Broadcast IR_j;
7: **end for**

Fig. 6. IR Partitioning.

especially for those clients with particularly unfavorable channel conditions (e.g., with a high mobility or situates in a poor terrain). Since there may be a large number of clients listening to the same IR in the broadcast channel, it is impractical for the server to retransmit an IR to completely avoid corruption. Instead, to reduce the chance of corruption, an original IR is divided into ω segments, IR_j for $j = 1$ to ω. Each IR segment then contains only id-timestamp pairs in a represented broadcast period, i.e., (d_x, t_x) is included in IR_j if and only if $(T_i - jL) < t_x \leq (T_i - (j-1)L)$, for $j = 1$ to ω. The packet overhead increases from one to ω. However, if an id-timestamp pair is included in the original IR, it appears in one and only one IR segment. The total number of id-timestamp pairs in the ω segments remains the same as that in the original IR.

Algorithm 1 formalizes how the server constructs and broadcasts the IR segments. If a client's cache is consistent up to T_{i-1} (using IR at T_{i-1}), the client is required to retrieve only the latest segment at T_i, i.e., IR_1. The complete IR at T_i can be reconstructed locally using IR_1 at T_i and IR at T_{i-1}. Similarly, if a client misses both the IRs at T_{i-2} and T_{i-1}, then IR at T_{i-3} and the latest two IR segments at T_i are required for local reconstruction. Algorithm 2 formalizes how the local reconstruction process is performed at the client.

Dividing an IR into ω segments can reduce both the corruption probability and power consumption. Each IR contains cache invalidation information for ωL seconds, while an IR segment contains cache invalidation information for L seconds only. Let the size of an IR and an IR segment be $S_{\omega L}$ and S_L, respectively. Suppose the broadcast channel seen by a particular client has a bit error rate, P_e, and the overhead of each packet is x bits. In IR+UIR, the client should receive the complete IR. Thus, the probability of successfully receiving the IR is $(1 - P_e)^{(S_{\omega L}+x)}$. In the proposed scheme, the last IR segment is required. Thus, the probability of successfully receiving the last IR segment is $(1 - P_e)^{(S_L+x)}$. Since updates to a data item follow exponential distribution with mean update arrival rate of $\frac{1}{T_u}$, S_L would be smaller than $S_{\omega L}$ (please refer to [21] for details). The success probability to receive an IR segment would be much higher than a complete IR. Thus, the client has a greater chance to benefit from subsequent UIRs. The power consumed in downloading the required cache invalidation information is also reduced.

The idea is further illustrated in Fig. 8. In both cases, the client correctly receives the IR at T_{i-1}, but some contents of IR at T_i are corrupted due to transmission errors. In the IR+UIR scheme, the client should discard the IR at T_i. There are two adverse consequences: 1) the client cannot answer any query until the next successful IR, which is at least L seconds later, and 2) the client cannot make use of subsequent UIRs. This increases the query delay and the client is effectively operating in the basic IR scheme. In the proposed scheme, the server divides an IR into ω segments using Algorithm 1 (Fig. 6) such that each IR segment contains invalidation information for a represented broadcast interval. An id-timestamp pair appearing in the latest segment at T_i indicates the data item is updated at least once during the time period from T_{i-1} to T_i. The client can use this latest segment, together with the IR at T_{i-1}, to reconstruct the IR at T_i using Algorithm 2 (Fig. 7). As a result, the client can use subsequent UIRs to invalidate its local cache and answer queries locally, as if there were no corrupted content in the IR.

4.3 Proposed Scheme 2: Improving Channel Utilization

In the wireless environment, clients experience the same broadcast channel differently [17]. Some clients perceive a good channel (e.g., the client is near to the server) while others may simultaneously perceive a poor one (e.g., far away from the server, deep fading, etc.). If the traffic is scheduled to broadcast at a fixed low rate in the hope of improving the success probability, the wireless channel is sometimes underutilized; on the other hand, with a fixed high rate broadcast, some clients may be prone to frequent errors, which leads to an increase in average query delay. Thus, dynamically adjusting the transmission rate according to the channel conditions could improve both channel utilization and overall performance.

The work in [14] is mainly about leveraging the time-varying effective channel bandwidth property to improve the performance of individual point-to-point unicast traffic. However, in the cache invalidation scenario, we are concerned about the traffic delivered over the broadcast channel. Specifically, the server cannot optimize the transmission to each particular client by using the "best" transmission mode for the client. Instead, the server has to use only a single transmission mode in a broadcast message for all clients. Indeed, if a higher mode is used in the broadcast (i.e., more aggressive), some clients suffering from poor channel conditions may not successfully receive the broadcast message, despite the fact that a shorter delay is achieved through such a high data rate transmission. On the other hand, if a lower mode is used in the broadcast (i.e., more conservative), some clients enjoying good channel conditions may not fully benefit from the good channel quality with such a low data rate transmission, resulting in a wastage of bandwidth resources.

To improve the success probability of broadcast traffic, the optimal transmission rate should depend on: 1) the current channel status of all clients and 2) the importance of the information being delivered. This idea is illustrated as follows: deliver more important information using a low-rate (i.e., higher level of error protection) to improve the success probability; deliver less important information

Algorithm 2 IR Reconstruction

1: INPUT: L, ω, T_i (defined in Section 2.2); T_{last} (timestamp of the last correct IR); IR_{last} (last correct IR);
2: OUTPUT: IR' (IR at T_i);
3:
4: $j = \textbf{min}\ \{\ \omega,\ \frac{T_i - T_{\text{last}}}{L}\}$ (since there are only ω segments)
5: **if** all the latest j segments at T_i are uncorrupted **then**
6: $IR' = \text{Reconstruct}(\text{IR}_{\text{last}}, \text{all the latest } j \text{ segments})$;
7: $T_{\text{last}} = T_i$;
8: Normal operation using IR';
9: **else**
10: Suspend operation and wait for the next IR at $T_i + L$;
11: **end if**
12:
13: Function Reconstruct(IR_{last}, all the latest j segments)
14: $IR' = \emptyset$;
15: **for** each (d_x, t_x) in the latest j segments **do**
16: $IR' = IR' \cup (d_x, t_x)$;
17: **end for**
18: **for** each (d_x, t_x) in IR_{last} **do**
19: **if** $(d_x, t_x) \notin IR'$ **then**
20: $IR' = IR' \cup (d_x, t_x)$;
21: **end if**
22: **end for**;
23: Return IR';

Fig. 7. IR Reconstruction.

using a high-rate (i.e., lower level of error protection), to better utilize the wireless channel.

For active clients, the latest IR segments (e.g., IR_1, IR_2, IR_3), UIRs and query replies are necessary for correct operations: maintain cache consistency and serve queries. Thus, they are more appropriate for low-rate broadcast to provide better protection against transmission errors. The other IR segments, which contain old cache invalidation information, are only necessary for some clients that have been disconnected for some period of time ($< \omega L$). As such, these old segments are more suitable for high-rate broadcast to better utilize the channel. However, these low-rate and high-rate broadcasts are not kept unchanged for all IR broadcasts. What transmission rates are suitable for low-rate and high-rate broadcast is a function of the channel conditions of all clients. A broadcast scheduler is introduced at the server side to determine the optimal transmission rates for different broadcast traffic. The broadcast scheduler collects the channel status information (CSI) from all clients with the aid of the polling subframe in the downlink and pilot subframe in the uplink. As discussed above, it is not good to use either an aggressive (i.e., high rate) or a conservative (i.e., low rate) broadcast strategy. Thus, the average data rate, computed based on all CSIs, is used to indicate the aggregate channel conditions of all clients. The average rate is used to determine, from Fig. 4b, the transmission mode in broadcasting more important information. From our simulation results (detailed in Section 5), it is observed that the latest three IR segments are more important to mitigate the effect of transmission errors. The less important one is then transmitted with one higher mode (see Table 1). The algorithm at the broadcast scheduler is outlined in Algorithm 3 (Fig. 9).

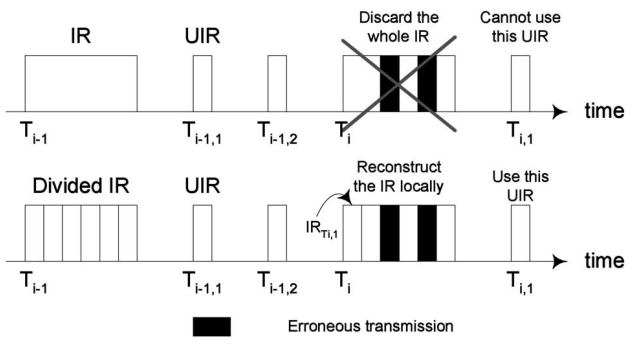


Fig. 8. Reducing the probability of corruption in IR.

4.4 Proposed Scheme 3: Reducing the Average Delay in Other Downlink Traffic

In IR-based approaches, server aggregates query requests from all its clients over L seconds and broadcasts the corresponding replies after each IR. The rationale is to let more clients share the same reply via broadcast to improve efficiency, at the expense of increase in query delay among all clients. However, the channel is blocked for the whole broadcast period and other downlink traffic has to be suspended. If query replies are distributed more evenly, the impact of broadcast on other downlink traffic would be reduced.

Depending on the client size and the query generation rate of each client, there may be a long list of query replies following each IR. As discussed in Section 4.2, the size of each IR can be very large in order to support long disconnection period. These two factors cause the broadcast traffic to occupy the channel for a long time, resulting in a large delay in other downlink traffic. The severity of the large delay depends on the nature of other downlink traffic. For elastic traffic, there is little harm of suffering from slightly larger delay. However, a large delay is unacceptable for some delay sensitive traffic such as real-time voice or video. To reduce the adverse effect of broadcast on other downlink traffic, in our approach, the server broadcasts query replies after both IRs and UIRs instead of just IRs. This can shorten the longest broadcast time, in turn, reducing the average waiting time of other downlink traffic spent at the server.

This proposed scheme can also improve the query delay by reducing the penalty due to cache miss. As illustrated in

Algorithm 3 Broadcast Scheduling

1: INPUT: CSI_i (the instantaneous channel state information of client i); CSI_{mean} (the nominal CSI based on average rate);
2: OUTPUT: M_{basic} (basic broadcast mode for better protection); M_{enhanced} (enhanced broadcast mode for higher transmission rate);
3:
4: **for** each downlink frame **do**
5: Calculate CSI_{mean} from all CSI_i;
6: M_{basic} = using CSI_{mean}, lookup the corresponding transmission mode from Figure 4(b).
7: M_{enhanced} = **min** $\{5, M_{\text{basic}} + 1\}$; (since the highest mode is 5)
8: **if** the current information slot carries data for (i) the latest 3 IR fragments (IR_1, IR_2 and IR_3) or (ii) UIRs or (iii) query replies **then**
9: Use M_{basic} to broadcast;
10: **end if**
11: **if** the current timeslot carries data for the remaining old IR fragments (IR_4 to IR_ω) **then**
12: Use M_{enhanced} to broadcast;
13: **end if**
14: **end for**

Fig. 9. Broadcast Scheduling.

Fig. 10, after receiving an UIR and invalidating any obsolete data items, the client notices that a query cannot be served locally, i.e., a cache miss. The client then issues an uplink query request to the server. In the IR+UIR scheme, the query reply is scheduled to be broadcast after the next IR at T_i. Equivalently, the expected query delay due to cache miss is half the IR broadcast period, i.e., $\frac{L}{2}$ seconds. In our proposed scheme, however, the expected delay for a cache miss is the same as that for a cache hit, i.e., $\frac{L}{2m}$ seconds.

However, the trade-off is the reduction in replies aggregation effect due to the shorter broadcast cycle. A broadcast data item is cached by all clients. When a query for the data item arrives at a later time, depending on the status of the cache, clients will have different actions. This leads to two conflicting effects: if the cached copy is still valid, the query can be served locally and the client can conserve battery power in uplink transmission; otherwise, an uplink request will be issued and the resources expended in previous broadcast of the data item are wasted. Which effect outweighs the other depends on the update arrival rate, query distribution in the database, and also the cache size of each client. Our simulation results in Section 5 suggest that our proposed scheme achieves an overall performance improvement in different settings.

4.5 Summary

We propose three different cache invalidation schemes to tackle the two simplifying assumptions made in most previous work: error-free broadcast channel and no other downlink traffic in the system. The first scheme divides an IR into different segments to reduce the corruption probability. The second scheme takes advantage of the time-varying channel characteristic to work under different channel conditions. The third scheme distributes the query replies more evenly to reduce the waiting time of other downlink traffic at the server. Our three proposed schemes can be used separately or combined to mitigate the effect of transmission errors and reduce the impact of broadcast on other downlink traffic.

5 Performance Results

5.1 Simulation Model and Parameters

We performed extensive simulations to evaluate the performance of the three proposed cache invalidation strategies under error-prone broadcast with other downlink traffic. We employed the system model with adaptive physical layer as discussed in Section 3. Fig. 11 shows the block diagram of the simulation set-up, which involves three entities: server, client, and broadcast scheduler. Most simulation parameters are the same as in [4] (see Table 2). We evaluate the basic IR, the IR+UIR, and the three proposed cache invalidation schemes, based on three metrics: 1) average query delay, 2) number of uplink requests per successful query, and 3) average delay experienced by other downlink traffic (e.g., voice). To study how the different approaches perform under various situations, the above metrics are studied by varying six system parameters:

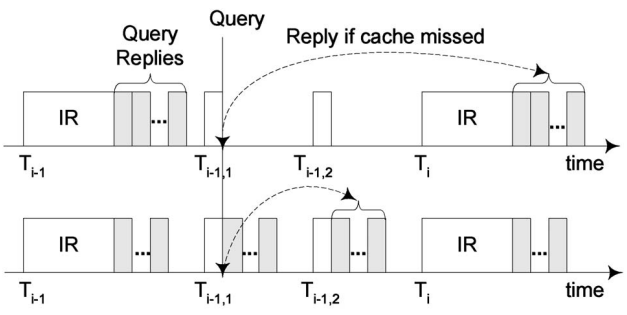


Fig. 10. Distributing query replies after IRs and UIRs.

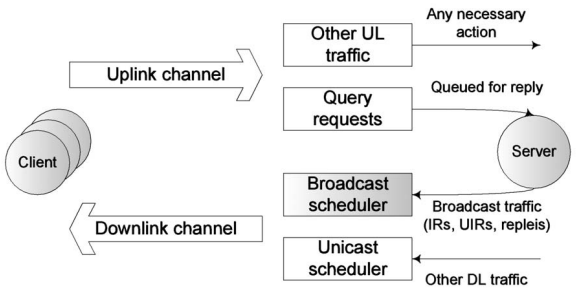


Fig. 11. The simulation set-up (shaded blocks are involved in the cache invalidation process).

TABLE 2
Simulation Parameters

Parameter	Value
Database size (N)	1000 items
Data item size (S_{data})	2048 bytes
Broadcast interval (L)	20 seconds
Broadcast window (ω)	10 intervals
UIR replicate times ($m-1$)	4, i.e., $(5-1)$
Hot data items (D_h)	5% of DB
Cold data items (D_c)	95% of DB
Cache size (c)	10% of DB
Hot data access probability (p_q)	0.8
Hot data update probability (p_u)	0.33
Packet overhead (x)	160 bits
Unique number size (U_{id})	32 bits
Mean query generation time (T_q)	10–10000s
Mean update arrival time (T_u)	10–10000s
Mean connection time	1000s
Mean disconnection time	10–10000s
No. of voice users	10
No. of data users (clients)	10–50
Speech bit rate (r_{speech})	8Kbps
File data rate (r_{file})	19.2Kbps
Frame duration	2.5ms
No. of frames	2×10^6

1. number of clients,
2. mean query generation time,
3. mean update arrival time,
4. number of UIRs,
5. access skew (hot access probability), and
6. mean disconnection time.

If a query request is not served until the next IR broadcast, the client assumes its request is corrupted and retransmits it. To avoid severe impact on the query delay due to successive transmission failures, queries that are not served within 60s are assumed to be lost, which could be up to about 50 percent of the total queries [21]. There are six transmission modes available (see Table 1). Out of these six modes, we select three of them to represent different broadcast conditions: 1) mode-0 (*Conservative*), 2) mode-2 (*Normal*), and 3) mode-5 (*Aggressive*), which are associated with low, moderate, and high error rate, respectively. In the second proposed scheme, adaptive broadcast achieves an average transmission rate of approximately 864Kbps, which corresponds to the performance between transmission modes 2 and 3. The values are summarized in Table 3. For comparison purposes, results based on the impractical error-free broadcast channel model (*Error-free*) are also presented as ideal cases. In the result data plots, *Divide-IR*, *Adaptive*, and *UIR-Reply* correspond to the three proposed scheme, respectively, while *All-3* represents the results obtained from the system in which the three schemes are combined.

5.2 Simulation Results

5.2.1 Effect of Number of Clients

First, we study how different performance metrics are affected by the number of clients in the system. This set of simulations measure the scalability of the underlying cache invalidation strategy. Client size is varied from 10 to 50. The results are shown in Fig. 12.

In terms of query delay, all approaches show a gradual improvement with client size. This trend is caused by more cache hits among different clients. Specifically, consider the case when a data item is updated, the first client that queries for it will result in a cache miss. When the server broadcasts the reply, other clients can cache the updated data item. Subsequent queries for the same data can be served locally without going through the server, resulting in the decrease in average query delay. This effect would be more profound if the data is in the hot data set. When there are more clients in the system, with other things being equal, the performance gain of sharing the data in the broadcast channel is larger. Thus, query delay improves with increasing number of clients.

TABLE 3
Transmission Rate Achieved in Different Broadcast Schemes

Broadcast scheme	Transmission rate (Kbps)
Conservative	160 (fixed)
Normal	640 (fixed)
Aggressive	1600 (fixed)
Adaptive	160–1600 (Average $\approx$ 864)

With a perfect broadcast channel (which is unrealistic), IR+UIR shows significant performance improvement over the basic IR scheme. The use of UIRs is effective in reducing the query delay. This observation is consistent with previous research results [2], [3], [4]. However, performance gain in IR+UIR degrades with transmission errors because corrupted IRs are discarded and clients have to wait for the next IR, which is at least one broadcast interval later, i.e., 20s. Although the query delay only slightly increases in the normal broadcast schemes, the performance degradation due to transmission errors is most prominent when the server broadcasts at an aggressive rate (i.e., IR+UIR (Aggressive)). In this case, the query delay in IR+UIR is actually worse than the basic IR scheme.

Divide-IR aims to reduce the IR corruption probability and performs better than IR+UIR as shown in Fig. 12a. A more significant difference occurs in aggressive broadcast. On the other hand, normal broadcast only shows some slight improvement. Notice that when the client size reaches 50, Divide-IR with normal broadcast is comparable to the ideal error-free case. For the proposed adaptive broadcast scheme (see Fig. 12c), it achieves a higher transmission rate (see Table 3), but involves a smaller query delay compared with the normal broadcast. This suggests that the adaptive broadcast scheme is more applicable than the fixed rate normal broadcast.

The performance of UIR-Reply in terms of query delay is shown in Fig. 12b. UIR-Reply always achieves a smaller query delay than IR+UIR using the same broadcast rate. In particular, its performance using normal broadcast is even better than that of IR+UIR in an ideal error-free channel. This is because clients do not need to wait for the next IR compared with the original schemes, which significantly reduces the query delay caused by cache misses. However, as illustrated in the UIR-Reply (Aggressive) curve, if corrupted IRs are frequent, the improvement gradually decreases with client size. In such cases, clients are mainly penalized by corrupted IRs but not by cache misses.

The number of uplink requests per successful query reflects the effectiveness of the caches. Due to cache hits, some queries can be served locally without any uplink request. On the other hand, due to transmission errors, more than one uplink request may be required. If clients do not employ any caching, at least one uplink request is required for each successful query. Thus, this metric measures both

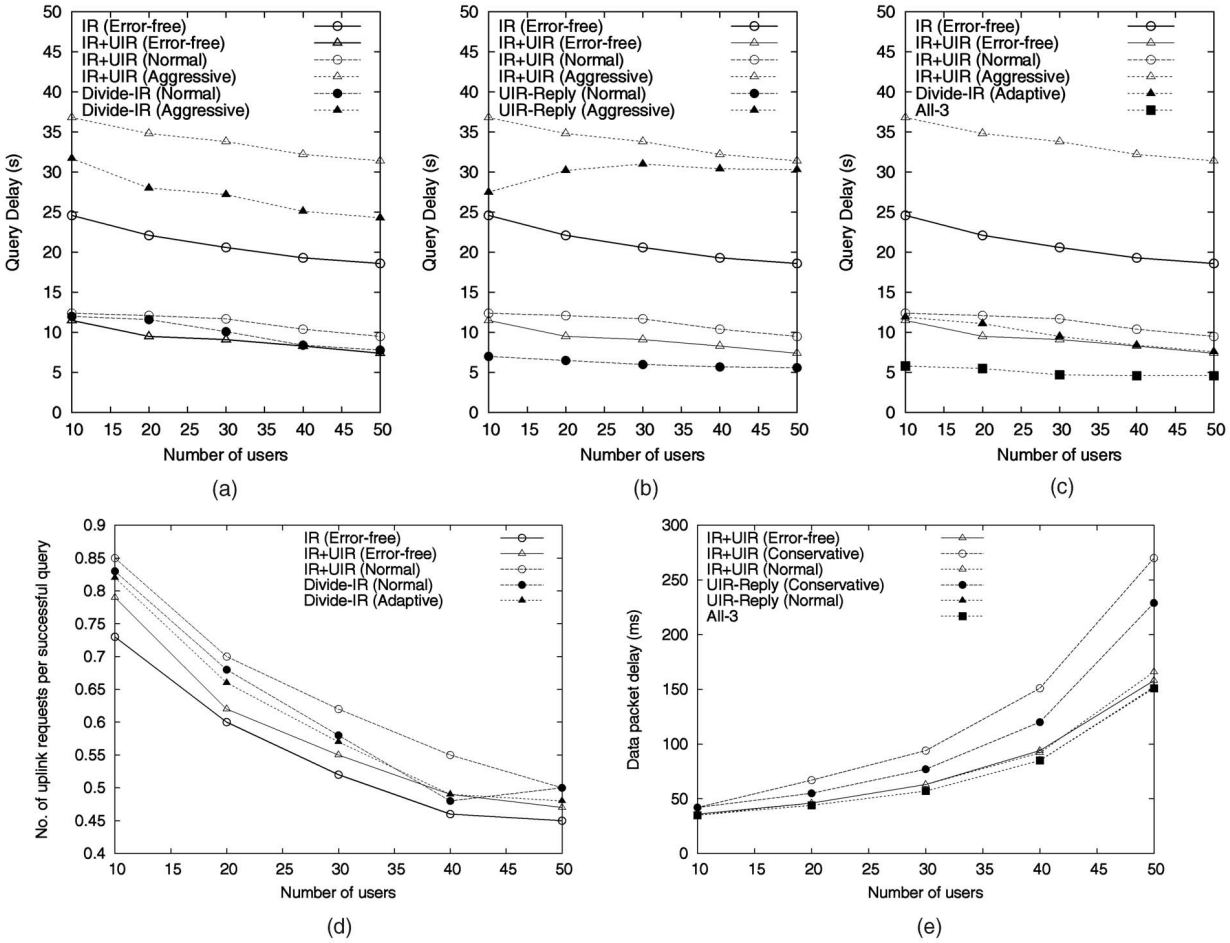


Fig. 12. Effect of number of clients ($T_q = 100$s; $T_u = 100$s). (a) Query delay: IR+UIR versus Divide-IR. (b) Query delay: IR+UIR versus UIR-Reply. (c) Query delay: Fixed versus Adaptive. (d) Number of uplink requests per successful query. (e) Delay in other downlink traffic.

the effect of cache hit and broadcast error. As can be seen in Fig. 12d, the number of uplink requests required decreases as the client size increases. The results for aggressive scheme are not shown in the figure because the uplink requests are very large (> 10) for each successful query, resulting from severe broadcast errors. The IR+UIR scheme requires slightly more uplink requests per query than the basic IR algorithm in an error-free broadcast environment. The reason is that clients issue uplink requests only after IRs in the basic scheme, while clients send requests after IRs and UIRs in the IR+UIR scheme. The larger aggregate effect in IR requires slightly lower uplink cost given the same query pattern. Divide-IR with constant rate broadcast rates reduces the number of retransmissions and shows some improvement. Under any constant rate broadcast schemes, the broadcast channel may sometimes be underutilized or overexploited. The adaptive scheme varies the broadcast rate according to the current channel conditions and shows better results. Hence, adaptive broadcast achieves further improvement using a higher broadcast rate when compared with the normal broadcast case.

The performance gains in query delay and uplink requests come with a cost. Fig. 9e shows the data packet delay experienced by other downlink traffic sharing the same wireless channel. There is little difference in data delay between basic IR and IR+UIR strategies. This suggests that data delay in other downlink traffic is largely due to long broadcast time involved in IRs and query replies. When more clients are in the system, there are more uplink requests. Although IR size is independent of client size, the server needs to broadcast more data items. Since other downlink traffic is blocked during broadcast, the long broadcast time causes large data packet delay. This is clearly reflected by the result of the conservative scheme, i.e., slowest broadcast. While conservative broadcast achieves the least transmission error, its impact on other traffic is the largest. Thus, there is a trade off between broadcast rate and the delay in other downlink traffic. To mitigate the effect of large delay, the longest broadcast period is reduced by distributing the query replies after IRs and UIRs. Other downlink traffic can then resume transmission earlier. Thus, UIR-Reply shows an improvement in different broadcast schemes, with the most encouraging performance achieved in the conservative one.

5.2.2 Effect of Query Generation Time

The effect of mean query generation time is shown in Fig. 13. This parameter is related to client size, but queries from different clients and the corresponding replies have different corruption probability. Another difference is that larger client size means more downlink traffic in the simulation environment. Here, we would like to study how different query rates affect the performance metrics.

Fig. 13 shows an increasing trend in query delay as the mean query generation time increases. The trend follows from the fact that fewer queries made means less replies

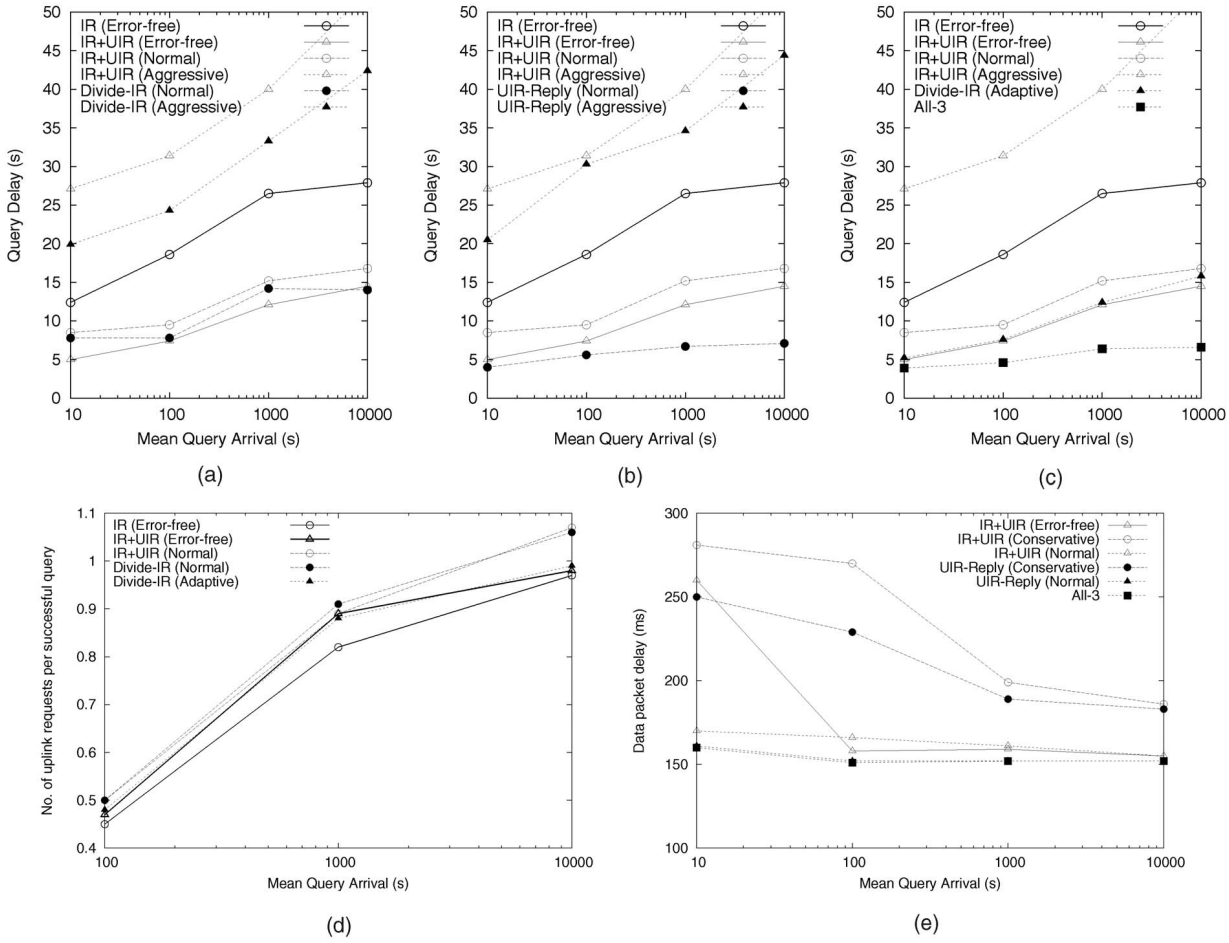


Fig. 13. Effect of mean query generation time (number of clients = 50; $T_u = 100\text{s}$). (a) Query delay: IR+UIR versus Divide-IR. (b) Query delay: IR+UIR versus UIR-Reply. (c) Query delay: Fixed versus Adaptive. (d) Number of uplink requests per successful query. (e) Delay in other downlink traffic.

broadcast by the server. This makes cached items more likely to be invalidated, leading to more frequent cache misses. There is little to do with corrupted IRs. As can be seen in Fig. 13a, Divide-IR is not very effective in reducing query delay. On the other hand, UIR-Reply aims to reduce the penalty due to cache miss and, therefore, performs better: the query delay increases at a slower pace (see Fig. 13b). UIR-Reply using aggressive broadcast gives poor performance because corruption in IR is the dominant factor. We also observe a significant improvement in IR+UIR over the basic IR approach, which justifies the use of UIRs. However, the inherent error-prone wireless channel causes data corruption among different clients. This increases the query delay compared with the ideal cases with the perfect channel assumption. With frequent broadcast errors (poor channel quality), IR+UIR performs even worse than the perfect IR, as reflected from results using aggressive broadcast. Thus, the channel quality, as well as the broadcast scheme, significantly affect the query delay.

Fig. 13d shows the results for the number of uplink requests per successful query. The same performance difference is also observed: IR requires fewer uplink requests per successful query at different query rates. As explained in the previous section, this difference results from the larger aggregate effect in basic IR than that in IR+UIR. Excessive errors at $T_q = 10\text{s}$ make this metric very large and, thus, the results are not shown here. At $T_q = 10^4\text{s}$, most of the data items cached by clients become invalid at the time when a query is generated, leading to frequent cache misses. As a result, the performance of IR and IR+UIR becomes comparable. In such a case, the number of uplink requests per successful query is about one—client caches are effectively not functioning. The use of UIRs is ineffective at such a low query rate.

Fig. 13e shows the delay in other downlink traffic against mean query generation time. The use of UIRs increases the broadcast overhead, comparing with the basic IR. Thus, IR+UIR shows a slightly larger delay in other downlink traffic. The result shows a downward trend from high to low query rates. At $T_q = 10^4\text{s}$, the server only needs to broadcast a very small number of replies. In this case, there is little difference between whether to distribute the data replies or not. Thus, the performance of UIR-Reply and the original schemes converges. The delay experienced at that point, i.e., about 160ms, which represents the lower bound for this update rate, $T_u = 100\text{s}$ is largely due to IRs and UIRs. The improvement by UIR-Reply is more significant in conservative broadcast. This confirms that delay in other downlink traffic is mainly due to the long broadcast time of IRs and data replies.

5.2.3 Effect of Update Arrival Time

Fig. 14 shows how performance metrics are affected by various mean update arrival time ($T_u = 10\text{s}$ to 10^4s). This set

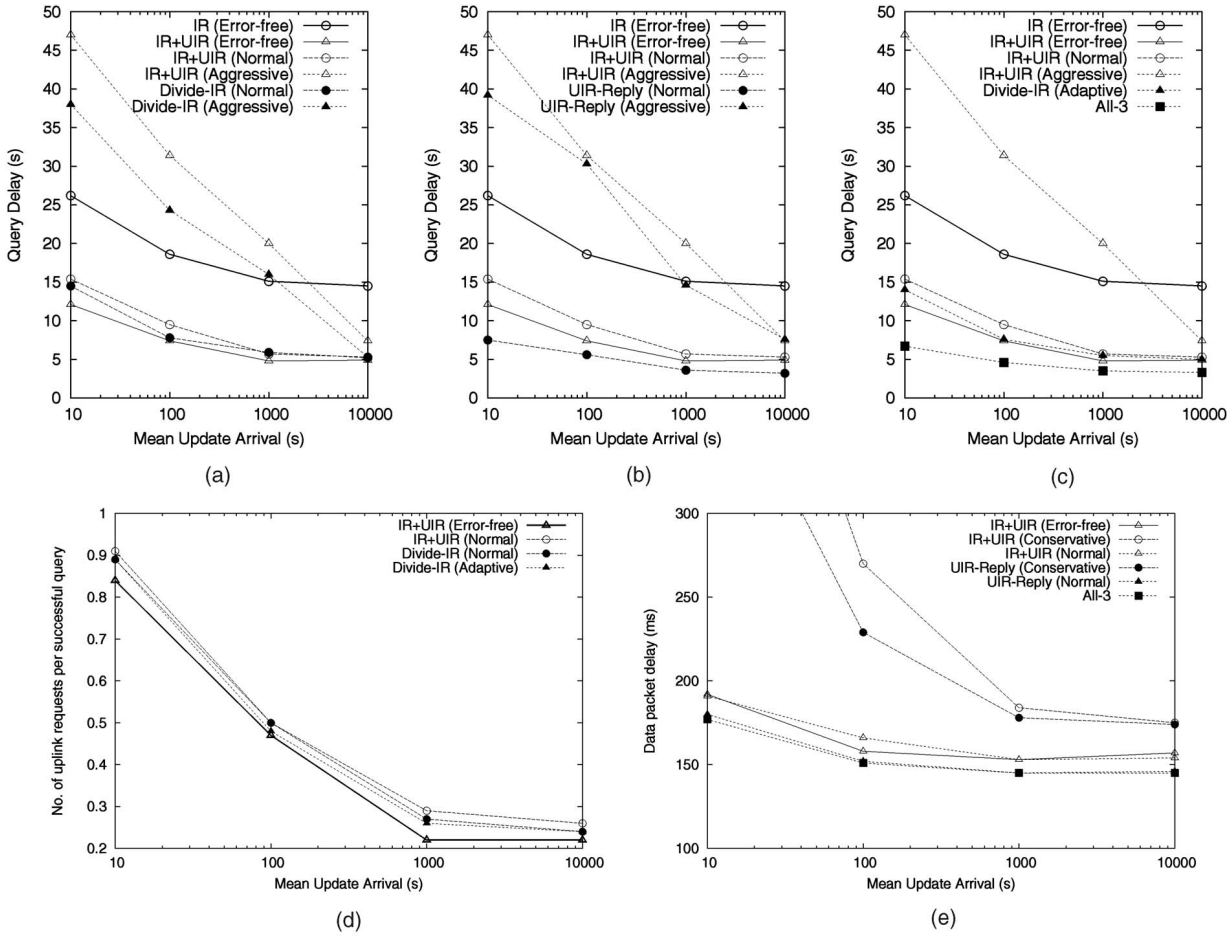


Fig. 14. Effect of mean update arrival time (number of clients = 50; $T_q = 100\text{s}$). (a) Query delay: IR+UIR versus Divide-IR. (b) Query delay: IR+UIR versus UIR-Reply. (c) Query delay: Fixed versus Adaptive. (d) Number of uplink requests per successful query. (e) Delay in other downlink traffic.

of results aims to study the appropriate cache invalidation strategy under different update conditions.

Update arrival time is directly related to the number of obsolete data items. When data items are updated more frequently, cached items are more likely to be invalidated. Subsequent queries would result in cache miss. This is reflected by the downward trend in Fig. 14. IR+UIR outperforms the basic IR approach, similar to previous cases. However, the performance of IR+UIR deteriorates in different error-prone conditions. For aggressive broadcast, IR+UIR is worse than the basic IR for $T_u = 10\text{s}$ to 10^3s. As shown in Fig. 14a, Divide-IR shows slight improvement in normal broadcast case, while more significant results are observed in using aggressive broadcast. This is because when IRs are divided into segments, clients have a greater chance of receiving IRs correctly. As there are more frequent IRs being corrupted when the server performs aggressive broadcast, the performance gain from Divide-IR is more prominent. Fig. 14b shows the results using UIR-Reply. The improvement is generally larger, particularly at $T_u = 10\text{s}$. The large query delay at this high update rate is mainly due to cache miss but not corruption in IRs. As a result, UIR-Reply outperforms Divide-IR in this case. When the update rate is very low, e.g., $T_u = 10^4\text{s}$, the performance of all approaches using UIRs converge, but still outperforms the basic IR.

In Fig. 14d, we can see that the number of uplink requests per successful query decreases as a result of lowering the update rate. This result confirms that clients' cache is more effective in low to moderate update rate. When updates occur more frequently, cached data items will usually become invalid before being used. Thus, the advantage of caching is reduced: Each successful query requires an average of about 0.9 uplink request. High update rate also indirectly affects other downlink traffic. Frequent updates induce more cache misses, which results in more uplink requests from clients. Thus, the server has to broadcast a large number of replies to serve the queries. This large broadcast traffic has a profound adverse effect on other downlink traffic. As illustrated in Fig. 14, the impact becomes more significant in conservative broadcast. If there is some delay sensitive traffic in the system, such a high data delay is obviously unacceptable. UIR-Reply alleviates this effect by distributing the broadcast traffic and results in the improved performance. To summarize, high update rate or, equivalently, low update arrival time, increases both query delay and data delay in other downlink traffic, making caching less favorable.

5.2.4 Effect of Number of UIRs

Fig. 15 shows the effect of different number of UIRs on the performance metrics. If there are more UIRs, clients can invalidate their cache more frequently and answer queries locally with smaller delay. However, having more UIRs mean larger broadcast overhead, which has a negative effect on other downlink traffic. On the other hand, if the

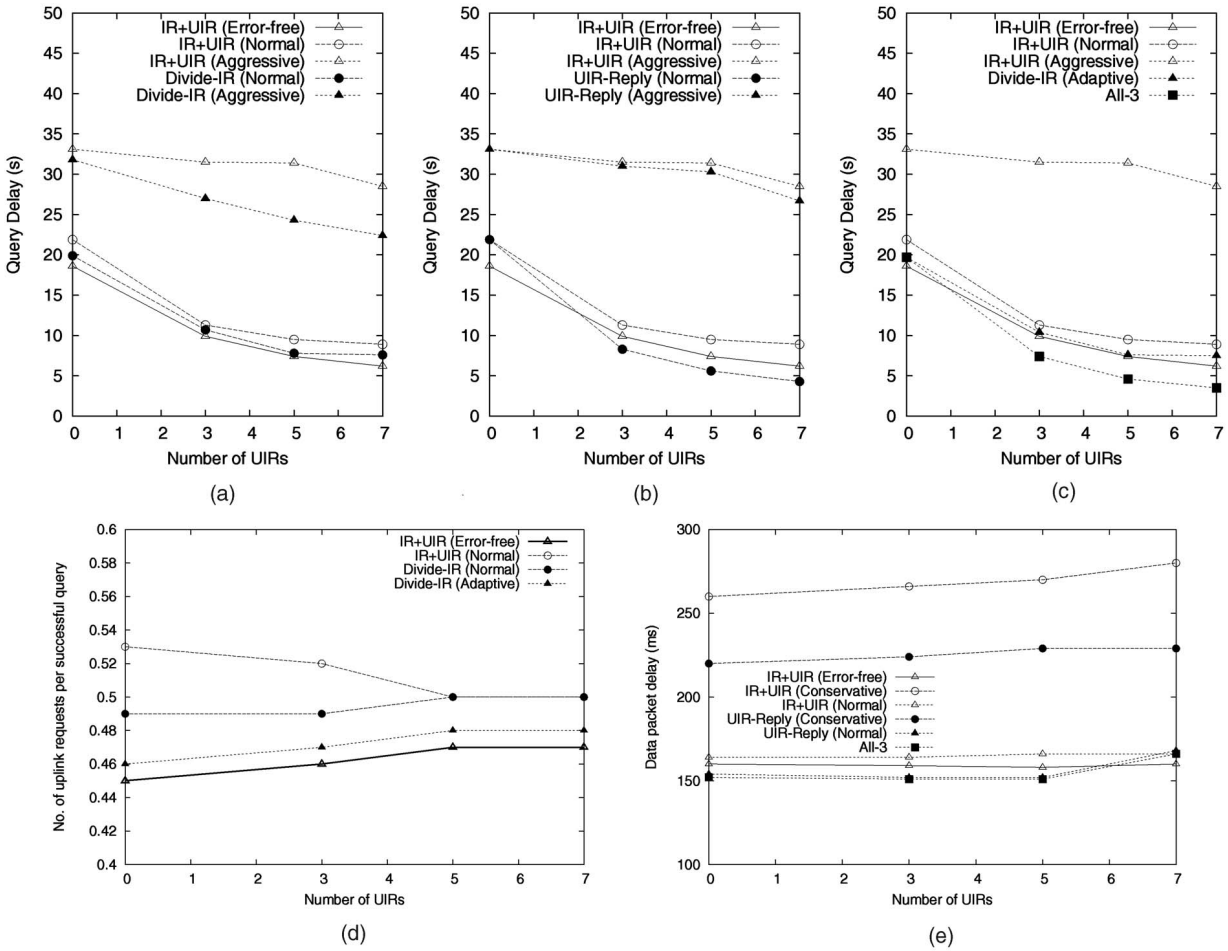


Fig. 15. Effect of mean number of UIRs (number of clients = 50; $T_u = 100$s; $T_q = 100$s). (a) Query delay: IR+UIR versus Divide-IR. (b) Query delay: IR+UIR versus UIR-Reply. (c) Query delay: Fixed versus Adaptive. (d) Number of uplink requests per successful query. (e) Delay in other downlink traffic.

number of UIR replicates is too small, the performance gain from using UIRs is not significant. Thus, these simulations study the optimal number of UIR replicates between successive IRs such that we can achieve performance improvement without sacrificing other downlink traffic too much.

The downward trend in Fig. 15 confirms that we can achieve better performance with more UIR replicates. However, the marginal performance gain decreases and begins to saturate starting at UIR count = 5. From the results using aggressive broadcast schemes, the improvement by using Divide-IR also increases with the number of UIRs. This is consistent with the result that the integrity of IRs plays an important role on the effectiveness of UIRs. As shown in Fig. 15d, the uplink requests per successful query decrease with the number of UIRs. The uplink cost in different schemes starst to converge from UIRs = 5. Fig. 15e depicts the cost to increase the number of UIR replicates. The slight upward trend in the figure is due to the broadcast overhead introduced by UIRs. Since UIRs are of a small size, the result shows a gradual increase only. Considering the above performance metrics, the appropriate number of UIR replicates would be around 5. Indeed, the above results further explain the default value of UIR replicates used in previous research results and the simulations described earlier.

5.2.5 Effect of Access Skew

The effect of access skew on different cache invalidation strategies is shown in Fig. 16. Specifically, the hot data access probability (p_q) is varied from 0.4 to 1.0 while maintaining other parameters unchanged. The range of p_q value represents different query patterns. If most clients have similar query characteristics, p_q is close to 1; otherwise, p_q would be just a small value. The simulation results show how the performance metrics are affected under different query patterns.

In terms of query delay in Fig. 16, performance only improves gradually, but not proportionally as the access skew is increased. This suggests that IR-based cache invalidation strategies work well in low access skew setting, though the performance at large p_q is better. The channel error factor affects the performance more at small p_q, while different schemes give similar query delay at high access skew. At low to medium broadcast errors, UIR-Reply generally outperforms Divide-IR as illustrated in Fig. 16a and 16b. Using aggressive broadcast (see Fig. 16c), Divide-IR shows a larger improvement even all queries fall on the hot data set ($p_q = 1$). This again confirms that the integrity of IRs is crucial to the query delay.

The uplink cost is largely affected by the access skew. As shown in Fig. 16d, the uplink cost decreases sharply from 0.8 to 0.3 as the hot data access probability is increased from 0.4 to 1.0. This downward trend is mainly due to the increased

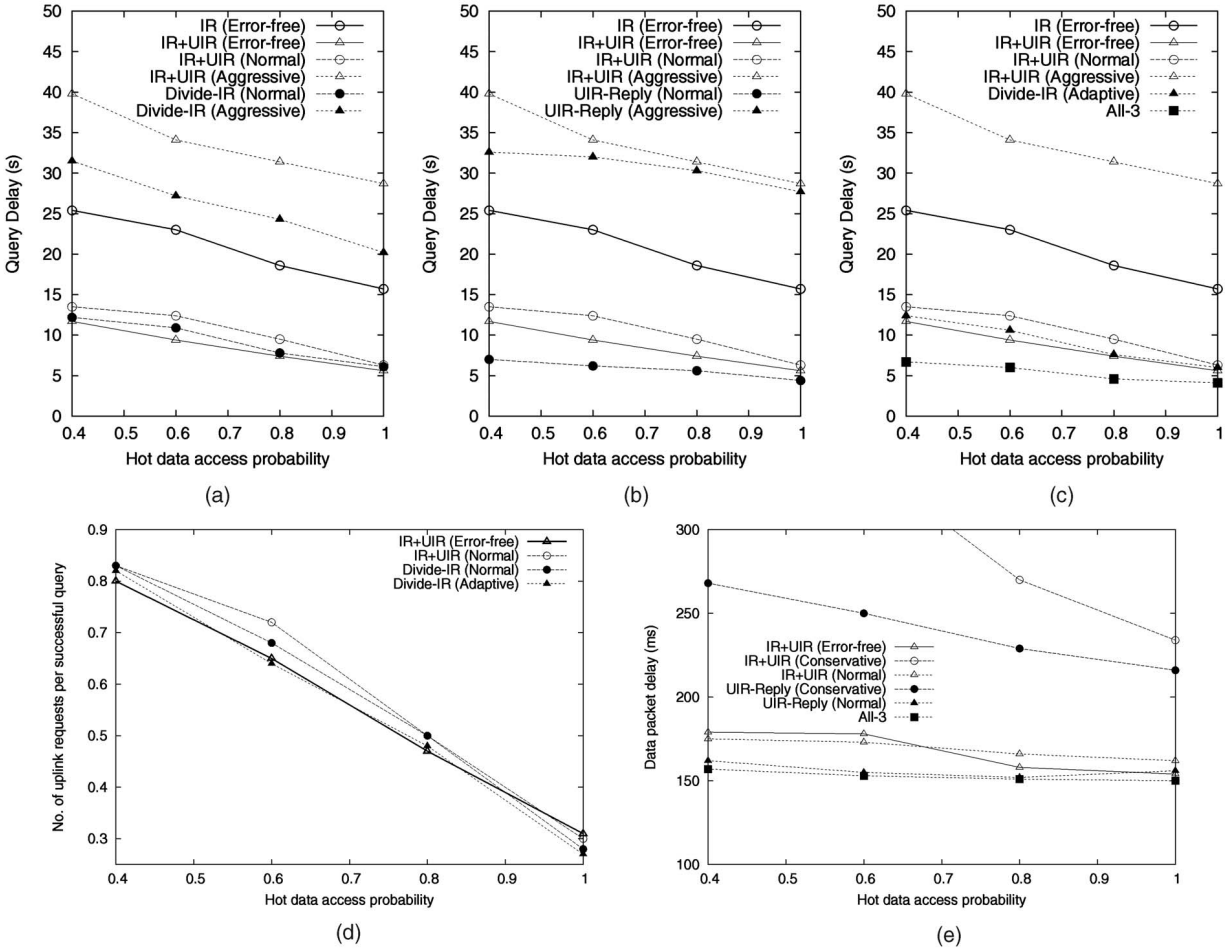


Fig. 16. Effect of access skew (number of clients = 50; $T_u = 100\text{s}$; $T_q = 100\text{s}$). (a) Query delay: IR+UIR versus Divide-IR. (b) Query delay: IR+UIR versus UIR-Reply. (c) Query delay: Fixed versus Adaptive. (d) Number of uplink requests per successful query. (e) Delay in other downlink traffic.

usage of cached data items in highly skewed query patterns. As can be seen in Fig. 16e, other downlink traffic, except the aggressive cases, experiences similar delay at different access probabilities. To summarize, access skew is mainly concerned with query delay and uplink cost, while delay in other downlink traffic is comparatively not affected.

5.2.6 Effect of Disconnection Time

The last set of simulations examines the effect of mean disconnection time on different performance metrics. Each client has a mean connection time of 1000s. Mean disconnection period varies from 10s to 10^4s. The results are shown in Fig. 17. In effect, short disconnection period, e.g., 10s, is similar to hard handoff. We observe that different performance metrics are not much affected. Even the basic IR cache invalidation strategy does not show large deterioration in performance at that disconnection period. This suggests that IR-based strategies perform well even when the mobile clients move across cells provided that each cell broadcasts the same IRs and UIRs. In practice, depending on the level of synchronization, there exists some time difference between broadcast in different cells. When the disconnection period is longer, query delay increases, as shown in Fig. 17.

If the disconnection period is shorter than the limit, ωL (200s), a client can wait for the next IR to invalidate its cache. The effect is as if the client is connected but the IRs in that period are all corrupted. Thus, the upward trend is observed from Fig. 17d when mean disconnection time increases from 10s to 10^2s. However, if the disconnection period is longer than 200s, the client should flush its entire cache. As a result, a significant increase in the number of uplink requests per successful query occurs from 10^2s to 10^4s. When a client reconnects, it requires a correct IR for invalidating the local cache. This confirms the result that UIR-Reply schemes only show little improvement, while the improvement in Divide-IR schemes is slightly better. After a long disconnection, a client has to flush its entire cache and then issue uplink requests for every new query. Thus, the number of uplink requests for query grows to around 0.9, reducing the cache effectiveness. On the other hand, a long disconnection period effectively decreases the query rate. Although the size of IRs and UIRs are not affected, the server broadcasts a smaller number of query replies. As shown in Fig. 17e, this reduces the delay experienced by other downlink traffic.

5.3 Discussion

Based on the simulation results in previous sections, we discuss the applicability of the three proposed cache invalidation schemes below.

The first proposed scheme divides each IR into segments to salvage invalidation information from transmission errors at the expense of slightly increase in broadcast traffic. This scheme leverages the fact that most clients only require the up-to-date invalidation information in IRs. From the simulation results, it is more applicable in a more

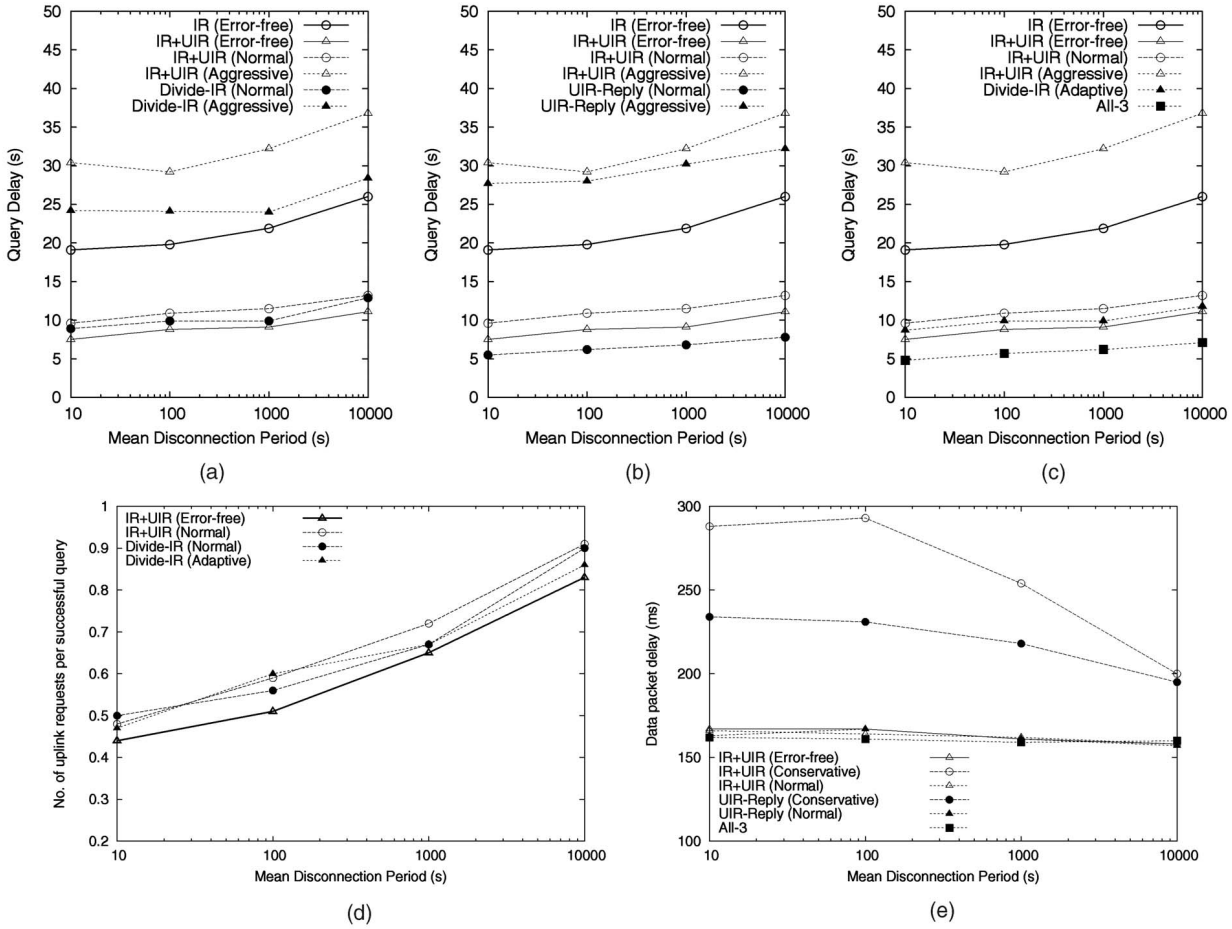


Fig. 17. Effect of mean disconnection time (number of clients = 50; $T_u = 100$s; $T_q = 100$s). (a) Query delay: IR+UIR versus Divide-IR. (b) Query delay: IR+UIR versus UIR-Reply. (c) Query delay: Fixed versus Adaptive. (d) Number of uplink requests per successful query. (e) Delay in other downlink traffic.

error-prone environment and a moderate to high update arrival rate condition. The effectiveness largely depends on the size of an IR and the frequency of transmission errors.

The second proposed scheme takes advantage of the time-varying effective bandwidth characteristic. In the simulation, three fixed rate broadcast schemes are used to represent different levels of transmission errors. A low transmission rate (fewer transmission errors) induces larger delay in other downlink traffic, while a high transmission mode is unfavorable for most broadcast traffic. It is observed that no fixed rate broadcast scheme is more preferable in all settings. A server should dynamically adjust the transmission mode to balance the two conflicting factors. In a practical system, clients are inevitably suffering from time-varying transmission errors, it is more desirable to adopt an adaptive broadcast scheme.

The third proposed scheme aims to reduce delay in other downlink traffic at the expense of an increase in total broadcast traffic volume. It is observed that data queries mainly account for the queueing delay in other downlink traffic. By distributing the data queries to both IRs and UIRs, queueing delay is shown to be reduced. The increase in broadcast traffic would then be justified if delay-sensitive downlink traffic is present in the system. Since the server answers queries more frequently, the proposed scheme also serves to reduce the query delay. Thus, the appropriateness depends on the delay requirement of both other downlink traffic and the database access application itself.

6 Concluding Remarks

IR-based cache invalidation strategies have received much attention due to its salient scalability and energy efficiency properties. However, there are still some drawbacks. One of the drawbacks is the problem with long disconnection, which is addressed in [1], [11], [12]. Another drawback is the long query delay involved. The addition of UIRs to the IR-based schemes (IR+UIR) [2], [3], [4] can significantly reduce the query delay over the basic IR scheme. However, existing algorithms may not be practicable because they are designed based on several unrealistic simplifying assumptions: 1) error-free broadcast and 2) no other downlink traffic in the system. In this paper, we study the effect of lifting these two assumptions on the basic IR and the IR+UIR in terms of various performance metrics. Our simulation results show that: 1) error-prone broadcast channel increases the query delay and 2) broadcast traffic increases the delay in other downlink traffic.

We proposed three cache invalidation schemes to address the above two issues. The first scheme (Divide-IR) is to divide an IR into a number of segments. If a client receives the last IR successfully, it only requires downloading the latest fragment in the next IR. The corruption probability in IR is therefore reduced. The second scheme (Adaptive) is to vary the broadcast transmission rate according to the current channel conditions. If most clients perceive a good channel, the server broadcasts at a higher

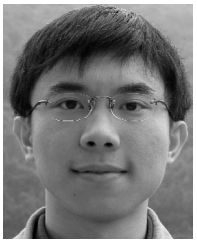
rate, to better utilize the channel; otherwise, the server broadcasts at a slower pace, reducing transmission errors. The third proposed scheme (UIR-Reply) aims to reduce the adverse impact of broadcast on other downlink traffic and the penalty due to cache misses, through distributing query replies after IRs and UIRs. These three proposed schemes can be used separately or concurrently. Our simulation results showed that they perform better than the original IR+UIR by alleviating the effect of transmission errors on broadcast traffic and the impact of broadcast traffic on other downlink traffic in the system.

ACKNOWLEDGMENTS

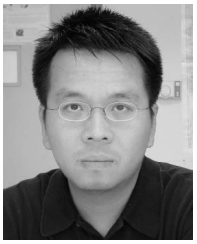
The authors would like to thank the anonymous reviewers (in particular, Reviewer 3) for their careful reading and constructive comments about the paper. A preliminary version of portions of this paper was presented at the *IPDPS 2004 International Workshop on Algorithms for Wireless, Mobile, Ad Hoc, and Sensor Networks (WMAN'2004)*, Santa Fe, New Mexico, US. This research was supported by a grant from the Research Grants Council of the HKSAR Government under project number HKU 7162/03E.

REFERENCES

[1] 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.
[2] G. Cao, "On Improving the Performance of Cache Invalidation in Mobile Environment," *ACM Mobile Networks and Applications,* vol. 7, no. 4, pp. 291-303, Aug. 2002.
[3] G. Cao, "Proactive Power-Aware Cache Management for Mobile Computing Systems," *IEEE Trans. Computers,* vol. 51, no. 6, pp. 608-621, June 2002.
[4] G. Cao, "A Scalable Low-Latency Cache Invalidation Strategy for Mobile Environments," *IEEE Trans. Knowledge and Data Eng.,* vol. 15, no. 5, pp. 1251-1265, Sept./Oct. 2003.
[5] B.Y.L. Chan, A. Si, and H.V. Leong, "A Framework for Cache Management for Mobile Databases: Design and Evaluation," *J. Distributed and Parallel Databases,* vol. 10, no. 1, pp. 23-57, July 2001.
[6] S.K. Das, E. Lee, K. Basu, and S.K. Sen, "Performance Optimization of VoIP Calls over Wireless Links Using H. 323 Protocol," *IEEE Trans. Computers,* vol. 52, no. 6, pp. 742-752, June 2003.
[7] L.M. Feeney and M. Nilsson, "Investigating the Energy Consumption of a Wireless Network Interface in an Ad Hoc Networking Environment," *Proc. INFOCOM 2001,* vol. 3, pp. 1548-1557, Apr. 2001.
[8] V.K. Garg, *Wireless Network Evolution: 2G to 3G.* Prentice-Hall, 2002.
[9] A.J. Goldsmith and S.-G. Chua, "Variable-Rate Variable-Power MQAM for Fading Channels," *IEEE Trans. Comm.,* vol. 45, no. 10, pp. 1218-1230, Oct. 1997.
[10] R. Han, P. Bhagwat, R. LaMaire, T. Mummert, V. Perret, and J. Rubas, "Dynamic Adaptation in An Image Transcoding Proxy for Mobile Web Browsing," *IEEE Personal Comm.,* vol. 5, no. 6, pp. 8-17, Dec. 1998.
[11] Q. Hu and D. Lee, "Adaptive Cache Invalidation Methods in Mobile Environments," *Proc. High Performance Distributed Computing,* pp. 264-273, Aug. 1997.
[12] J. Jing, A. Elmagarmid, A.S. Helal, 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, Oct. 1997.
[13] A. Kahol, S. Khurana, S.K.S. Gupta, and P.K. Srimani, "A Strategy to Manage Cache Consistency in a Disconnected Distributed Environment," *IEEE Trans. Parallel and Distributed Systems,* vol. 12, no. 7, pp. 686-700, July 2001.
[14] Y.-K. Kwok and V.K.N. Lau, "A Novel Channel-Adaptive Uplink Access Control Protocol for Nomadic Computing," *IEEE Trans. Parallel and Distributed Systems,* vol. 13, no. 11, pp. 1150-1165, Nov. 2002.
[15] V.K.N. Lau, "Performance Analysis of Variable Rate: Symbol-By-Symbol Adaptive Bit Interleaved Coded Modulation for Rayleigh Fading Channels," *IEEE Trans. Vehicular Technology,* vol. 51, no. 3, pp. 537-550, May 2002.
[16] D.L. Lee, W.-C. Lee, J. Xu, and B. Zheng, "Data Management in Location-Dependent Information Services," *IEEE Pervasive Computing,* vol. 1, no. 3, pp. 65-72, July-Sept. 2002.
[17] J.D. Parsons, *The Mobile Radio Propagation Channel,* second ed., John Wiley & Sons, 2000.
[18] K. Tan, J. Cai, and B.C. Ooi, "An Evaluation of Cache Invalidation Strategies in Wireless Environments," *IEEE Trans. Parallel and Distributed Systems,* vol. 12, no. 8, pp. 789-807, Aug. 2001.
[19] J. Xu, X. Tang, and D.L. Lee, "Performance Analysis of Location-Dependent Cache Invalidation Schemes for Mobile Environments," *IEEE Trans. Knowledge and Data Eng.,* vol. 15, no. 2, pp. 474-488, Mar./Apr. 2003.
[20] J. Xu, Q. Hu, W.-C. Lee, and D.L. Lee, "Performance Evaluation of an Optimal Cache Replacement Policy for Wireless Data Dissemination," *IEEE Trans. Knowledge and Data Eng.,* vol. 16, no. 1, pp. 125-139, Jan. 2004.
[21] M.K.H. Yeung, "On Channel Adaptive Cache Invalidation Schemes in Wireless Client-Server Systems with Heterogeneous Downlink Traffic and Game Theoretic Considerations," M.Phil. thesis, The Univ. of Hong Kong, Aug. 2004.
[22] L. Yin, G. Cao, C. Das, and A. Ashraf, "Power-Aware Prefetch in Mobile Environments," *Proc. Int'l Conf. Distributed Computing Systems,* pp. 571-578, July 2002.
[23] L. Yin and G. Cao, "Adaptive Power-Aware Prefetch in Wireless Networks," *IEEE Trans. Wireless Comm.,* accepted for publication and to appear.
[24] B. Zheng, J. Xu, and D.L. Lee, "Cache Invalidation and Replacement Policies for Location-Dependent Data in Mobile Environments," *IEEE Trans. Computers,* vol. 51, no. 10, pp. 1141-1153, Oct. 2002.
[25] M. Zorzi, "Packet Dropping Statistics of a Data-Link Protocol for Wireless Local Communications," *IEEE Trans. Vehicular Technology,* vol. 52, no. 1, pp. 71-79, Jan. 2003.



Mark Kai Ho Yeung received the BEng degree (1st class honors) in electrical and electronic engineering from The University of Hong Kong in 2002. He will receive the MPhil degree, also from The University of Hong Kong, in August 2004. His research interests include wireless cache invalidation, wireless data access, and ad hoc mobile networks. He is a student member of the IEEE.



Yu-Kwong Kwok received the BSc degree in computer engineering from the University of Hong Kong in 1991, and the MPhil and PhD degrees in computer science from the Hong Kong University of Science and Technology (HKUST) in 1994 and 1997, respectively. He is an associate professor in the Department of Electrical and Electronic Engineering at The University of Hong Kong. Before joining The University of Hong Kong, he was a visiting scholar in the Parallel Processing Laboratory in the School of Electrical and Computer Engineering at Purdue University. His research interests include distributed computing systems, wireless networking, and mobile data access. He is a senior member of the IEEE. He is also a member of the ACM, the IEEE Computer Society, and the IEEE Communications Society.