Computer Communications 41 (2014) 55–66

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

User-defined consistency sensitive cache invalidation strategies for wireless data access

Sunho Lim [a],*, Yumin Lee [a], Jongpil Cheon [b], Manki Min [c], Wei Wang [c]

[a] T^2WISTOR: TTU Wireless Mobile Networking Laboratory, Dept. of Computer Science, Texas Tech University, Lubbock, TX 79409, United States
[b] Dept. of Educational Psychology and Leadership, Texas Tech University, Lubbock, TX 79409, United States
[c] Dept. of Electrical Engineering and Computer Science, South Dakota State University, Brookings, SD 57007, United States

ARTICLE INFO

Article history:
Received 23 May 2013
Received in revised form 21 December 2013
Accepted 26 December 2013
Available online 18 January 2014

Keywords:
Cache invalidation
Lazy request
Multiple data transmission
Opportunistic data access
User-defined consistency

ABSTRACT

In order to fulfill users' insatiable interests in accessing Internet services and information wirelessly, one of the key optimization techniques is caching frequently accessed data items in a local cache. A strong consistency is implicitly assumed in most caching schemes but it may cause a long query delay. In this paper, we propose a consistency-sensitive cache invalidation scheme, called ConSens, based on the existing Invalidation Report (IR) and Updated IR (UIR) based cache invalidation frameworks. In the ConSens scheme, each user is able to set its own consistency level with a server independently. This user-defined cache consistency can support diverse consistency requirements of applications. We also propose both lazy request and opportunistic data access techniques to effectively balance the data accessibility and query delay. In addition, we enhance the IR-based cache invalidation mechanism and propose a multiple data transmission scheme, called MDT, to further reduce the query delay. Extensive performance evaluation studies show that the proposed strategies can effectively balance the data accessibility, reduce the query delay, and significantly increase the number of opportunistic accesses according to the user-defined consistencies.

1. Introduction

Recent technological advances in high-speed wireless network, mobility support, and portability enable mobile users to access the Internet wirelessly. For example, smart phones have been receiving a tremendous attention and are becoming popular. 115.8 million users in the U.S. already own a smart phone in 2012 and it is expected to increase upto 192.4 million by 2016 [1]. 47.7% of all mobile subscribers in the U.S. are smart phone users in 2012 and it is expected to increase upto 74.1% by 2016 [2]. Approximately seven out of ten American adults access the Internet wirelessly anyhow. Typically, the explosive rise in the number of application software running on smart phones, called apps, has fueled mobile users to easily and frequently access the Internet wirelessly anytime and anywhere. According to Apple AppStore, more than 800K apps are active and more than 50 billion downloads have been made as of May 2013 [3].

In order to fulfill mobile users' rapidly growing interests in accessing Internet services and information wirelessly, one of the key optimization techniques is caching frequently accessed data items in a local cache. Caching techniques can relieve the network traffic and improve information accessibility and availability. Thus, the communication cost in terms of wireless bandwidth, battery energy, and latency can be reduced significantly. A great deal of research effort has been devoted in developing various caching strategies in wireless and/or mobile environments. Here, we use a mobile user to refer to a wireless mobile device or a person who carries it. Thus, we use the terms mobile users and mobile nodes (later in short, nodes) interchangeably.

Most caching strategies implicitly assume a strong consistency between the original data item and its cached copy [4–12]. When a node generates a query, the query should be answered by the recently updated data item stored either at a server or a local cache. However, ensuring a strong consistency requires a non-negligible long query delay to confirm that both the source data item and its cached copy are consistent. For example, update-sensitive weather information such as tornado, hurricane, or tsunami requires a real-time update operation. News, stock prices, and traffic information are also update-sensitive. A cached copy must be updated before answering a query when the source is updated at the server. However, ensuring a strong consistency is not always a prompt and

* Corresponding author.
E-mail addresses: sunho.lim@ttu.edu (S. Lim), yumin.lee@ttu.edu (Y. Lee), jongpil.cheon@ttu.edu (J. Cheon), manki.min@sdstate.edu (M. Min), wei.wang@sdstate.edu (W. Wang).

critical requirement. This is because a requirement of data consistency (later in short, consistency) can be different depending on the update sensitivity of information. For example, maps, video clips, e-flyers, and weather information are not update-sensitive. They may not need to reflect the current update of the source stored at the server in an urgent manner. Occasional inconsistencies between the source data item and its cached copy would be acceptable. In the Bing™ [13], the weather information encapsulated in a Web browser toolbar is not always latest updated (e.g., 45F as of 25 min or 1 h ago). Some mobile apps are not urgent to update their contents, such as mobile education game, travel, or magazine.

Under the consideration of diverse information and different update sensitivities, we propose a *consistency-sensitive cache invalidation* scheme, called *ConSens*. In the ConSens scheme, each node is able to set its own consistency level, called *target consistency*, with the server independently. This user-defined cache consistency is implemented by a minor modification of the existing Invalidation Report (IR) and Updated IR (UIR) based cache invalidation frameworks [4,5]. We also propose a *multiple data transmission* (MDT) scheme to enhance the fundamental IR-based cache invalidation mechanism [4].

Most cache invalidation strategies [4–12] often assume that (i) the validity of cached data items is determined by a strong consistency and/or (ii) the entire nodes have the same consistency with the server. However, the proposed approach is quite different from the conventional cache invalidation strategies, and our contributions are summarized in four-fold:

- First, we propose a consistency sensitive cache invalidation scheme, in which each node can flexibly and independently set its own consistency with the server. A consistency condition is developed to decide the current consistency based on the number of invalid cached data items accessed during a consistency window.
- Second, both *opportunistic data access* and *lazy request* techniques are also developed to improve the data accessibility but reduce the query latency. Depending on the consistency condition, each node can judiciously access the cached data items even without sending a request message to the server and waiting for the incoming IR (or UIR) from the server.
- Third, we also propose a multiple data transmission (MDT) scheme to further reduce the query delay in the IR-based cache invalidation mechanism. In the proposed scheme, the server repeatedly broadcasts a set of requested data items separately. Then each node that has sent a *request* message is able to individually answer a query as soon as receiving the requested data item.
- Fourth, we integrate the proposed techniques with the existing IR- and UIR-based cache invalidation frameworks [4,5]. We modify two major IR- and UIR-based cache invalidation schemes to implement the proposed consistency-sensitive data access and multiple data transmission techniques, ConSens-IR, ConSens-UIR, ConSens-IR-MDT, and ConSens-UIR-MDT.

We conduct extensive simulation-based studies of the proposed schemes to observe the impact of query interval, update interval, and target consistency on the communication performance. The simulation results indicate that the proposed schemes not only can increase the data accessibility but also can reduce the query latency significantly.

The rest of paper is organized as follows. The prior work is analyzed and the proposed strategies are presented in Sections 2 and 3, respectively. Section 4 is devoted to performance evaluation and comparison. Finally, we conclude the paper with future research directions in Section 5.

2. Related work

Most cache invalidation strategies deploy one or combination of the following design aspects: (i) Whether to maintain cache status or not (e.g., stateful or stateless); (ii) Who initiates cache validity (e.g., push, pull, or hybrid); and (iii) Level of cache consistency (e.g., strong, weak, or probabilistic). In this paper, we focus on the consistency dependent cache invalidation techniques.

2.1. Stateful or stateless

A server may or may not maintain any cache status of connected or disconnected nodes. In a stateful approach [6–9], the server keeps track of which node caches which data item (s). Whenever a data item is updated, the server broadcasts an IR to inform nodes of the cache update. In [8], a server-based poll-each-read (SB-PER) is proposed under the assumption of available global access and update information in the server. However, obtaining global update information is practically hard, if it is not impossible. For example, data items such as News, stock prices, or traffic information are not updated on a regular basis but are updated by an event, which is not pre-scheduled. In this paper, we do not consider the data items characterized by a scheduled update. In a stateless approach [4,5,10], however, the server is not aware of any cache status and thus, it periodically broadcasts an IR (or UIR) containing the limited amount of prior update histories. Here, the query delay is affected by the IR (or UIR) size and broadcast interval.

2.2. Push, pull, or hybrid

The push, pull, and hybrid techniques specify who initiates cache invalidation operation. In the push approach [4–10], the server initiates the cache invalidation, for example, by broadcasting an IR. In the pull approach [14], however, whenever a query is generated which can be answered by a cached data item, a *request* message is sent to the server for verifying the validity of the cached data item before it can be used for answering the query. In [8], an on-demand approach is proposed for fast moving vehicles in a multi-cell environment, where a query is forwarded to the server either for validating the cached data item or receiving the queried data item. In particular, when a vehicle handoffs, it sends the *ids* of the entire valid cached data items to the server for validity check and receives an *invalidation* message from the server accordingly. Several push- and pull-based schemes [11] and their combined schemes [15,16] have been proposed under the consideration of tradeoffs between query delay and communication overhead.

2.3. Strong, weak, or probabilistic

Various IR-based cache invalidation strategies have been widely used in wireless and/or mobile environments [4–12], where a server broadcasts an IR periodically (i.e., every L second, where L is a broadcast interval). A node can answer a query with a cached copy after validating it with the incoming IR. This ensures a strong consistency but the long query delay (i.e., $\frac{L}{2}$ in average) is unavoidable. Although a set of updated IRs (UIRs) [5] is broadcasted between two successive IRs to further reduce the query delay, non-negligible query delay (i.e., $\frac{L}{2 \cdot m}$ in average, where m is a number of UIRs) is still expected. The strong consistency ensures that only recently updated data item is accessed for answering a query, but the consistency maintenance cost in terms of wireless bandwidth, battery energy, communication overhead, and query delay increases.

A time-to-live (TTL) based approach have been deployed in Web and/or mobile environments [17,18]. In [18], an estimated TTL value is associated with each cached data item. A cached data item becomes an uncertain state when its TTL value expires and thus, it should be verified before answering a query. Since the update time of cached data item can be smaller than the TTL value and a node can fail to receive an IR broadcast, due to unreliable mobile environment, stale data item(s) can exist in the cache. The TTL-based approach provides a weak consistency and has a drawback in estimating a TTL value which is not trivial.

Probability-based consistency control techniques have been proposed to support diverse consistency requirements with the server [15,19–21]. An early probabilistic consistency ensures that the retrieved data item from the server is temporally consistent with the recently updated data item with a probability p [19]. A stochastic consistency guarantees that the deviation between source data item and its cached copy remains within the user-specific error tolerance ϵ with a probability p [20]. The evolution of source data item is modeled by the Brownian motion. In a delta consistency, a node can use a cached data item for answering a query as far as the deviation between source data item and its cached copy is less than δ, which is the user-specific maximum acceptable tolerance [21]. A probabilistic delta consistency integrates both stochastic and delta consistencies to provide a flexible consistency [15]. This approach is deployed in a hybrid wireless network, where (i) either single-hop or multi-hop relay is used to access the server and (ii) frequent disconnections from the server are expected due to the mobility of nodes.

2.4. Other approaches

Various caching techniques have been proposed in wireless mobile peer-to-peer networks, such as Mobile Ad hoc Networks (MANETs) [16,22–29], Wireless Sensor Networks (WSNs) [30], Internet-based MANETs (IMANETs) [11,15], or Internet-based Vehicular Ad hoc Networks (IVANETs) [31], where either MANET or VANET technology is combined with the Internet to provide users with a flexible information accessibility and availability. In [22], a replication scheme is proposed for periodically updated data items based on the previously suggested replica allocation method [32] in MANETs. In [11], several push and pull-based cache invalidation schemes are proposed based on the aggregate cache [33], in which the aggregated local caches of the nodes can be considered as a unified large cache for IMANETs. In [31], a cooperative cache invalidation scheme incorporating with a hierarchical network model and a location management scheme is proposed to reduce the overhead of cache invalidation operations in terms of the number of broadcasts and long query delay for fast moving vehicles in IVANETs.

In summary, little effort has been devoted in exploring consistency-sensitive wireless data access and its cache invalidation strategies that can adaptively balance the data accessibility and query delay. In light of this, we propose the ConSens scheme (i.e., stateless, push, and probabilistic approaches) and its enhancement.

3. The proposed cache invalidation strategy

In this section, we first briefly introduce a system model and then present our proposed cache invalidation strategies.

3.1. System model

We consider a cellular network, where a set of nodes is located in a cell and communicates with a base station (BS) by wireless links. Each node can move in any direction, make information

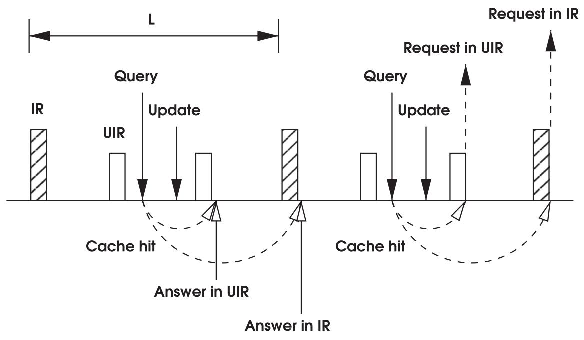

Fig. 1. The basic IR- and UIR-based cache invalidation framework [4,5].

search, and access requests from anywhere in the covered area. The BS is a gateway to the Internet and is assumed to have access to any information. This is because the BS is transparent to a database server from a node's point of view. Thus, we use the terms BS and database server (later in short, server) interchangeably. A database may directly be attached to a BS, or indirectly be attached to a fixed router or a server. We assume that the database consists of a total n data items. A data item (d) is the basic unit of an update or query operation. The database is only updated by the server, while a query is generated by nodes for read-only requests. A node can cache a limited number of data items.

To maintain cache consistency, we consider the IR- and UIR-based cache invalidation approaches [4,5]. In the IR approach, a server periodically broadcasts an IR consisting of a list of tuples, $[d^s, t^s]$, where d^s and t^s are an *id* of updated data item and its most recent timestamp, respectively. The IR contains the update history information witnessed during the last $L \cdot |w_b|$ broadcast intervals, such as $t^s > (t_{cur} - L \cdot w_b)$, where t_{cur} and w_b are the current time and a broadcast window, respectively. In the UIR approach, a set of UIRs can be broadcasted between successive IRs to further reduce the query delay based on the IR framework. Compared to the IR, the UIR does not contain the information of update history. Instead, the UIR contains a set of data items that have been updated after the last IR has been broadcasted. The UIR consists of a list of tuples, $[d', t']$, where $t' > T_i$. Here, T_i is the timestamp of the last IR. The server inserts several UIRs into each IR interval. Due to the periodic IR or UIR broadcast, nodes can operate in a *doze* mode to save battery energy.

For example, as shown in Fig. 1, when a node generates a query, it first searches its cached data items. If a cached copy that can answer the query is stored in the cache, the node should wait for the incoming IR (or UIR) to validate the cached copy. If the cached copy is valid (cache hit), the node directly accesses the cached copy to answer the query. If the queried data item is not cached or the cached copy is invalid (cache miss), the node should send an uplink *request* message to the server for the recently updated data item. The server will broadcast the requested data item(s) immediately after the IR broadcast.

In this paper, we assume that each node is willing to tolerate some inconsistencies and able to set its own consistency level with the server. The consistency can be represented as a threshold value that indicates the update tightness between the source data item stored at the server and its cached copy. For example, if the threshold value is one or less than one, then it is a strong or weak consistency, respectively. Thus, each node can directly type the threshold values into, for example, a system configuration to set the target consistencies depending on the consistency requirements of applications.

3.2. User-defined consistency sensitive cache invalidation

Most of the existing IR- and UIR-based cache invalidation strategies have been primarily focused on solving a long disconnection problem, reducing the query delay and/or energy consumption, or utilizing the broadcast bandwidth under the following implicit assumptions: (i) A strong consistency; and (ii) All nodes have the same consistency with the server. In this paper, we relax these assumptions and propose a user-defined consistency sensitive cache invalidation scheme, called ConSens, to support diverse consistency requirements of applications.

A basic idea of the proposed user-defined consistency approach is that each node can set its own consistency with the server flexibly and independently. Then data access and cache invalidation operations are adjusted accordingly. The ConSens scheme is described by the following three major components: user-defined cache consistency, opportunistic data access, and lazy request.

3.2.1. User-defined cache consistency

Each node is able to set its own consistency level, called a *target consistency* (τ, $0 < \tau \leqslant 1$). Here, $\tau = 1$ and $\tau < 1$ imply a strong consistency and a weak consistency with the server, respectively. In case of $\tau = 1$, a query will be answered either by a cached data item after the invalidation check with the incoming IR (or UIR) or the data item downloaded from the server. In case of $\tau < 1$, for example, if a node sets $\tau = 0.8$, then eight out of ten queries are guaranteed to be answered by the recently updated data items stored at the server or local cache, respectively.

To realize the target consistency, each node counts the number of invalid data items (N^{inval}) that has been accessed to answer the queries. N^{inval} is observed in a *consistency window* (w_c), represented as the number of last queries that has been generated. The current consistency (τ^{cur}) is calculated by $\tau^{cur} = 1 - \frac{\sum_{u=0}^{|w_c|}\text{Flag}[u]}{|w_c|}$, where Flag[$u$] ($0 \leqslant u < |w_c|$) keeps a set of flag bits either for valid (e.g., 1) or invalid (e.g., 0) cached data items that have been accessed to answer the queries. Each array element is initially set by 0. Note that since τ^{cur} changes slowly if $|w_c|$ increases and vice versa, the limited number of last queries ($|w_c|$) is chosen to quickly reflect τ^{cur} changes, i.e., 100.

The detail operation is depicted in Fig. 2. Here, we assume that each query (i.e., i, j, k, and l) can be answered by cached data items. A node sets τ and $|w_c|$ to 0.9 and the last ten queries, respectively. Before query i begins, suppose τ^{cur} is 0.9. In each query, the node checks the consistency condition ($\tau \leqslant \tau^{cur}$) and decides either to answer the query directly (i.e., queries i and l) or wait for the incoming IR (or UIR) (i.e., queries j and k). The node updates Flag[u] either with 0 (valid) or 1 (invalid) after receiving the IR (or UIR), and then recalculates τ^{cur}. For example, since query i is answered by a stale cached data item, 1 (invalid) is inserted in the right most bit and the existing bits are shifted to the left in the Flag[u]. Then τ^{cur} is recalculated, i.e., $0.8 = 1 - \frac{2}{10}$. These operations will be repeated for each query.

3.2.2. Opportunistic data access

Suppose a node sets τ (e.g., $\tau < 1$) and maintains τ^{cur}. When a query which can be answered by a cached copy is generated, the node first examines a *consistency condition*, $\tau \leqslant \tau^{cur}$. Depending on the condition, both data access and cache invalidation operations change adaptively. The rationale behind this condition is to adjust the current consistency into the user-defined target consistency, as closely as possible. If $\tau \leqslant \tau^{cur}$ is satisfied, then the current consistency is maintained higher than the target consistency. Thus, an unnecessary long query delay can be occurred. To reduce the query delay and τ^{cur}, the node aggressively uses the cached data items to answer the queries in advance without waiting for the incoming IR (or UIR) (see Fig. 3). However, the source data item stored at the server could be updated *a priori*. Thus, when the node receives the IR (or UIR), it should validate the answered cached data item by comparing the update time of source data item and

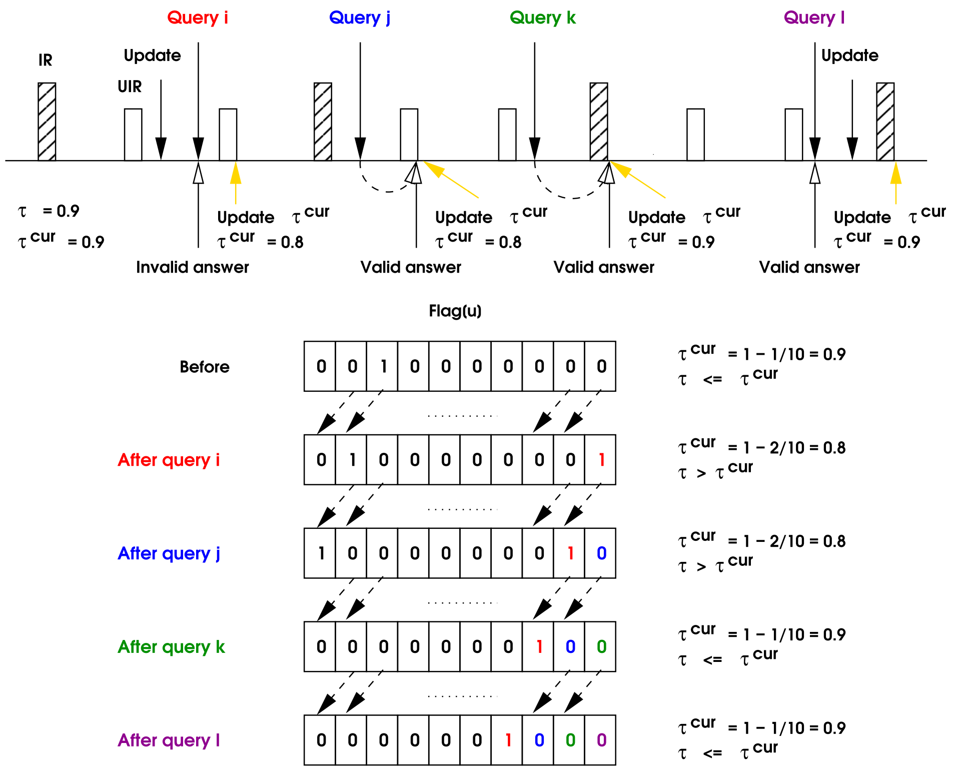

Fig. 2. The calculation of the current consistencies.

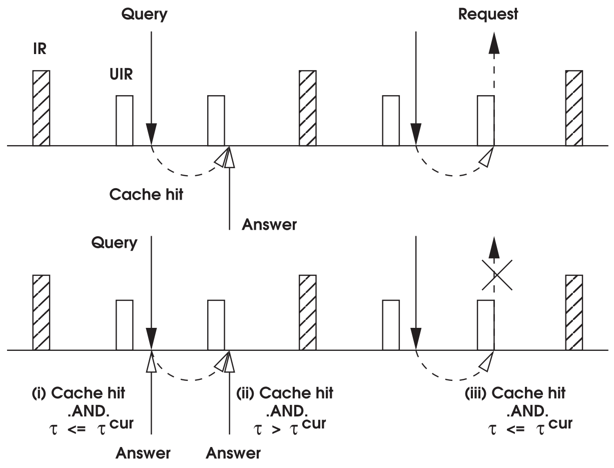
Query
Request
IR
UIR
Cache hit
Answer
Query
(i) Cache hit .AND. τ <= τ cur
(ii) Cache hit .AND. τ > τ cur
(iii) Cache hit .AND. τ <= τ cur
Answer
Answer

Fig. 3. Unlike to the conventional approach (upper), the ConSens scheme (bottom) can opportunistically access a cached copy for answering a query without waiting for the next IR (or UIR), if the consistency condition ($\tau \leqslant \tau^{cur}$) is satisfied ((i) case). Each node can also postpone sending an uplink *request* message to the server, if the consistency condition ($\tau \leqslant \tau^{cur}$) is still met ((iii) case). If the consistency condition ($\tau \leqslant \tau^{cur}$) is not satisfied, the ConSens scheme follows the conventional approach (i.e., waiting for the next IR (or UIR)) ((ii) case).

the access time of its cached copy. If the access time is less than the update time, then the query was answered by the valid cached data item. This implies that the node has *opportunistically* accessed the cached data item and has reduced the query delay significantly. According to the validity, the node updates Flag[u] and recalculates τ^{cur}. If $\tau \leqslant \tau^{cur}$ is not satisfied, however, the query should be answered by the recently updated data item to closely increase τ^{cur} upto τ. Thus, the node should wait for the incoming IR (or UIR) to validate the cached data item.

In summary, the ConSens scheme suppresses too high or too low τ^{cur} compared to the user-defined τ, but keeps τ^{cur} as close as possible to τ.

3.2.3. Lazy request

In this paper, we propose a *lazy request* approach to increase the data accessibility but reduce the query delay. A basic idea is that each node can *postpone* sending an uplink request message to the server but can use an invalid cached data item instead at most once, as far as the consistency condition ($\tau \leqslant \tau^{cur}$) is met. If τ^{cur} is less than τ or an invalid cached copy already has been used before, then the node sends a request message to the server for downloading the recently updated data item (see Fig. 3). After answering the query, the node updates Flag[u] and recalculates τ^{cur}. Due to the consistency condition, this approach ensures that only limited number of invalid cached data items can be used to answer the queries.

In the ConSens scheme, we use an UIR that contains a list of tuple, [d', t'], where d' and t' are an *id* of data item updated after the last IR has been broadcasted and its timestamp, respectively. Unlike to the original scheme [5], we include t' to decide whether the answered cached data item was valid for the purpose of the proposed scheme. The pseudo code of the proposed scheme is also presented for detail data access and cache invalidation operations in Fig. 4.

To capture the dynamic behavior of node, we present a Finite State Machine (FSM) for the ConSens scheme, called *FSMCon*, in Fig. 5. The FSMCon consists of five states (i.e., from s_1 to s_6) beginning with s_1. The state transition is initiated by an event (e) and then its action (a) is followed accordingly. Here, both event and action in state i are represented as $\frac{e_i}{a_i}$. s_1 does not change when a

Notations:
$d^c_{i,k}$, $t^c_{i,k}$: A cached data item stored in the k^{th} slot of local cache and its timestamp in a node (n_i), respectively, where $0 \le k < N_c$, where N_c is the number of cache slots.
C_i, Q_i, Q^{opp}_i: A cache, a query queue for data access, and a query queue for opportunistic data access in n_i, respectively.
Flag$_i[u]$: An array for marking (e.g., 0 or 1) the invalid queries in n_i, where $0 \le u < |w_c|$ and all array elements are initially set to 0.

```
(A) When $n_i$ generates a query $q$ for a data item $d$,
    if ($d \notin C_i$)
        Send an uplink request message to the server for downloading $d$;
    else {
        if ($\tau_i \le \tau^{cur}_i$) {
            if ($d^c_{i,k}$ == invalid) {
                if ($d^c_{i,k}$ has not been used) { /* lazy request */
                    Use $d^c_{i,k}$ to answer the $q$;
                    Flag$_i[u$++ % $|w_c|]$ = 1;
                    $\tau^{cur}_i = 1 - \frac{\sum_{u=0}^{|w_c|} \text{Flag}_i[u]}{|w_c|}$;
                }
                else
                    Send an uplink request message to the server for downloading $d$;
            }
            else { /* opportunistic data access */
                Use $d^c_{i,k}$ to answer the $q$;
                Queue $d$ into $Q^{opp}_i$ and wait for the incoming IR (or UIR);
            }
        }
        else
            Queue $d$ into $Q_i$ and wait for the incoming IR (or UIR);
    }

(B) When $n_i$ receives an IR (or UIR),
    for $\forall id(d) \in$ IR (or UIR) do { /* check the validity of cached data items */
        if ($d \in Q_i \quad \wedge \quad t^c_{i,k} < t^s$)
            Invalidate $d^c_{i,k}$;
    }
    for $d \in Q^{opp}_i$ do { /* dequeue the data item from $Q^{opp}_i$ */
        if ($t^c_{i,k} < t^s$)
            Flag$_i[u$++ % $|w_c|]$ = 0;
        else
            Flag$_i[u$++ % $|w_c|]$ = 1;
        $\tau^{cur}_i = 1 - \frac{\sum_{u=0}^{|w_c|} \text{Flag}_i[u]}{|w_c|}$;
    }
```

Fig. 4. The pseudo code of the ConSens scheme.

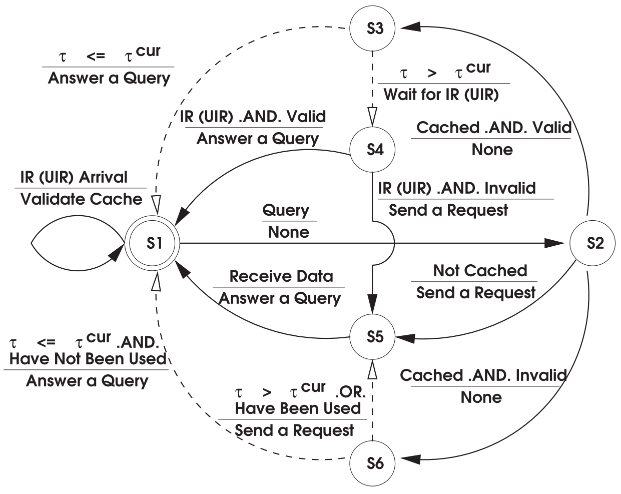
S1
S2
S3
S4
S5
S6
τ <= τ cur / Answer a Query
τ > τ cur / Wait for IR (UIR)
IR (UIR) .AND. Valid / Answer a Query
Cached .AND. Valid / None
IR (UIR) Arrival / Validate Cache
IR (UIR) .AND. Invalid / Send a Request
Query / None
Receive Data / Answer a Query
Not Cached / Send a Request
τ <= τ cur .AND. Have Not Been Used / Answer a Query
Cached .AND. Invalid / None
τ > τ cur .OR. Have Been Used / Send a Request

Fig. 5. A finite-state machine of node behavior in the ConSens scheme.

node receives an IR (or UIR) but change into s_2 when a node generates a query. In s_2, the state transitions $s_2 \Rightarrow s_3$ and $s_2 \Rightarrow s_6$ indicate that the queried data item is cached. If the queried data item is not cached, s_2 changes to s_5 where the node waits for the server to broadcast the queried data item. Note that the state transitions $s_3 \Rightarrow s_1$ and $s_6 \Rightarrow s_1$ indicate opportunistic data access and lazy request operations, respectively. The consistency condition is evaluated in both s_3 and s_6, and both states change (dashed line) accordingly.

3.3. Multiple data transmission

In this paper, we also propose a *multiple data transmission* (MDT) scheme to further reduce the query delay in the IR-based cache invalidation framework. In most IR-based approaches, the server encapsulates all the requested data items into a single data packet. Thus, the nodes who have requested in the previous IR period have to wait to answer the queries until the data packet is received. Note that this can incur an unnecessary long query delay and the nodes will experience the same period of query delay.

In the MDT scheme, however, the server repeatedly broadcasts each requested data item in a separate data packet until no more requests remain in the queue (Q^{bcast}_{server}) as shown in Fig. 6. Thus, each node can individually answer a query as soon as it receives the requested data packet from the server. Due to the number of separate data packet broadcasted from the server, an overhead of packet header may increase. However, the packet header size (i.e., 20 Bytes) is negligibly small compared to the data item size (i.e., 40 KBytes). The pseudo code of the MDT scheme is shown in below:

```
**for** $((d_x = pop(Q^{bcast}_{server})) \neq \text{Null})$ **do**
    Broadcast $d_x$;
```

The proposed MDT scheme is embedded into the existing IR- and UIR-based cache invalidation schemes. The proposed ConSens scheme is also implemented based on the MDT scheme. The performance comparison and analysis with and without the MDT scheme are conducted in Section 4.

In summary, the proposed ConSens scheme balances between the data accessibility and query delay. As far as the current consistency is close to the target consistency, each node can opportunistically access cached data items without waiting for the incoming IR (or UIR). Unlike to the prior IR- and UIR-based approaches,

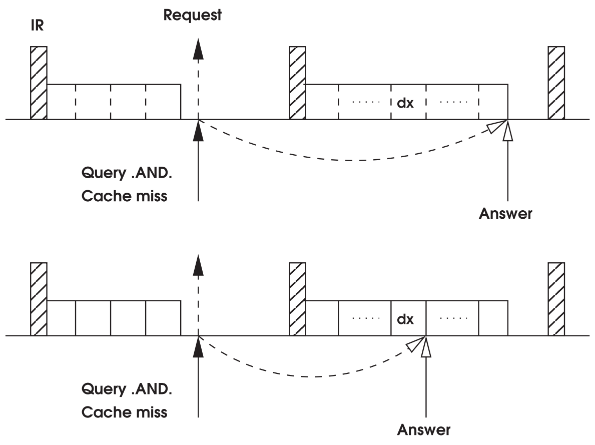

Fig. 6. In the conventional approach (upper), the node who has sent an uplink *request* message can answer a query after receiving the entire data packet broadcasted from the server. In the MDT scheme (bottom), however, the node can answer a query as soon as it receives the requested data item encapsulated in a separate data packet.

depending on the consistency condition, the proposed scheme neither waits for the incoming IR (or UIR) nor sends an uplink *request* message to the server blindly. In addition, the proposed MDT scheme further reduces the query delay, in which the server repeatedly broadcasts each requested data item encapsulated in a separate data packet.

4. Performance evaluation

In this paper, we design and develop a discrete-event driven simulation framework to conduct our experiments and evaluate the performance of proposed techniques. The simulation framework is written by CSIM20 [34] that is a popular development toolkit for discrete-event simulations and modeling.

4.1. Simulation testbed

In order to examine the proposed idea, we assume that both query and update inter arrival times follow the exponential distribution. To model a realistic data access, the entire data items stored in the database are classified into two subsets: hot and cold data items. 80% and 20% of query requests are for hot and cold data items, respectively. 33% of update requests are for hot data items but they are uniformly distributed within the hot subset. Likewise, 67% update requests are for cold subset. Similar data query and update patterns are found in [5,11,31]. Here, we do not consider multiple update frequencies for data items. Each node caches 10% of the data items in the database, i.e., 100 cache slots. When a cache is full, each node uses a simple cache replacement policy, Least Recently Used (LRU). In this paper, advanced cache admission control and replacement policies (e.g., a distance-based cache admission control and a Time and Distance Sensitive (TDS) replacement [33]) are not considered to clearly see the effect of the proposed scheme on the performance. The target consistency ranges from 0.7 to 0.95 with 0.05 steps, but we use 0.7 and 0.9 in most experiments unless otherwise stated to clearly see the performance differences. The consistency window is the last 100 queries. The important simulation parameters are summarized in Table 1.

4.2. Simulation results

In this section, we vary the key simulation parameters: update interval, query interval, and target consistency. Diverse combinations of the simulation parameters are used to conduct extensive performance evaluation studies. We first evaluate the performance of proposed ConSens scheme in terms of query delay, consistency changes, cache hit, number of uplink requests, number of lazy requests, and number of opportunistic accesses as a function of mean update and query intervals. Then the proposed MDT scheme is evaluated in terms of query delay as a function of mean update and query intervals.

For performance comparison, we modify the existing IR- [4] and UIR-based [5] schemes to aware the consistency sensitivity: *ConSens-IR* and *ConSens-UIR*. Also the proposed MDT is embedded into the existing IR and UIR schemes (called *IR-MDT* and *UIR-MDT*) as well as the ConSens scheme (called *ConSens-IR-MDT* and *ConSens-UIR-MDT*). In addition, we include the existing IR and UIR schemes as a base case, represented as *IR* and *UIR*, respectively.

4.2.1. The query delay

We evaluate the query delay as a function of update and query intervals. In Fig. 7(a)–(c), both UIR and ConSens-UIR schemes show better performance than the other two schemes, the IR and ConSens-IR schemes. A similar performance trend can be witnessed in Fig. 7(d)–(f). In Fig. 7, as the query and update rates increase,

Table 1
Simulation parameters.

Parameter	Value
Number of nodes	100
Database size (N)	1000 items
Data item size	40 KBytes
Hot data items	5% of DB
Cold data items	95% of DB
Mean query interval time (t_q)	25–300 s
Mean update interval time (t_u)	0.5–10,000 s
Broadcast bandwidth	384 Kbps [35]
Broadcast interval (L)	20 s
Broadcast window (w_b)	10 intervals
UIR replicate times	4
Target consistency (τ)	0.7–0.95
Consistency window (w_c)	100 queries

overall query delays reduce. In the UIR and ConSens-UIR schemes, when the server broadcasts a data item requested from a node, all the other nodes also receive and cache it. Thus, the query delay reduces because more queries can be answered by the latest updated cached data items without waiting for the incoming IR (or UIR).

In particular, the ConSens-IR scheme is sensitive to the target consistency. Due to the long IR broadcast period (i.e., 20 s), a query has the high probability of being answered by a cached copy before waiting for the incoming IR in the low target consistency (e.g., $\tau = 0.7$). In Fig. 7(a)–(c), as the query interval increases, the ConSens-IR scheme ($\tau = 0.7$) shows lower query delay than the ConSens-IR ($\tau = 0.9$) and IR schemes. This is because more cached data items are accessed to answer queries without waiting for the incoming IR. The query delay gap is also clearly witnessed in Fig. 7(d)–(f), as the update interval increases. In the ConSens-UIR, however, the target consistency does not affect the query delay much because of aggressive cache invalidate operations with a set of UIRs broadcasted in the successive IRs. In Fig. 7, the ConSens-IR and ConSens-UIR schemes achieve lower query delay than that of the IR and UIR schemes for the entire query and update intervals.

4.2.2. The consistency changes

According to the set of target consistencies (i.e., from 0.7 to 0.95), we monitor the consistency changes over the entire simulation period. In Fig. 8(a), after a cold state of simulation (i.e., after each node's empty cache is filled up), each node shows the fluctuated current consistency but it is quickly adjusted within the target consistency. Depending on the consistency condition and current consistency, both ConSens-IR and ConSens-UIR schemes can judiciously decide whether accessing the cached data items to answer the queries directly, or waiting for the incoming IR (or UIR) to validate the cached data items before answering the queries.

In Fig. 8(b), nodes flexibly change their target consistencies either between 0.9 and 0.8 or 0.7 and 0.8. After a cold state of simulation, the current consistency of each node is quickly adjusted to its target consistency with a little fluctuation. In Fig. 8, we only show the consistency changes of ConSens-IR scheme but the similar result is also witnessed in the ConSens-UIR scheme.

We experiment with last 50 or 200 queries for consistency window ($|w_c|$) to see how quickly the current consistency changes. As we can expect, the current consistency changes faster with the smaller consistency window and vice versa. However, if the consistency window is too small, then the gap between the current and target consistencies increases. In this paper, we set the consistency window to the last 100 queries.

4.2.3. The cache hit

We evaluate the cache hit as a function of update interval. In the ConSens scheme, a cache hit occurs in the following cases: (i) The consistency condition ($\tau \leqslant \tau^{cur}$) is not satisfied, and a cached data

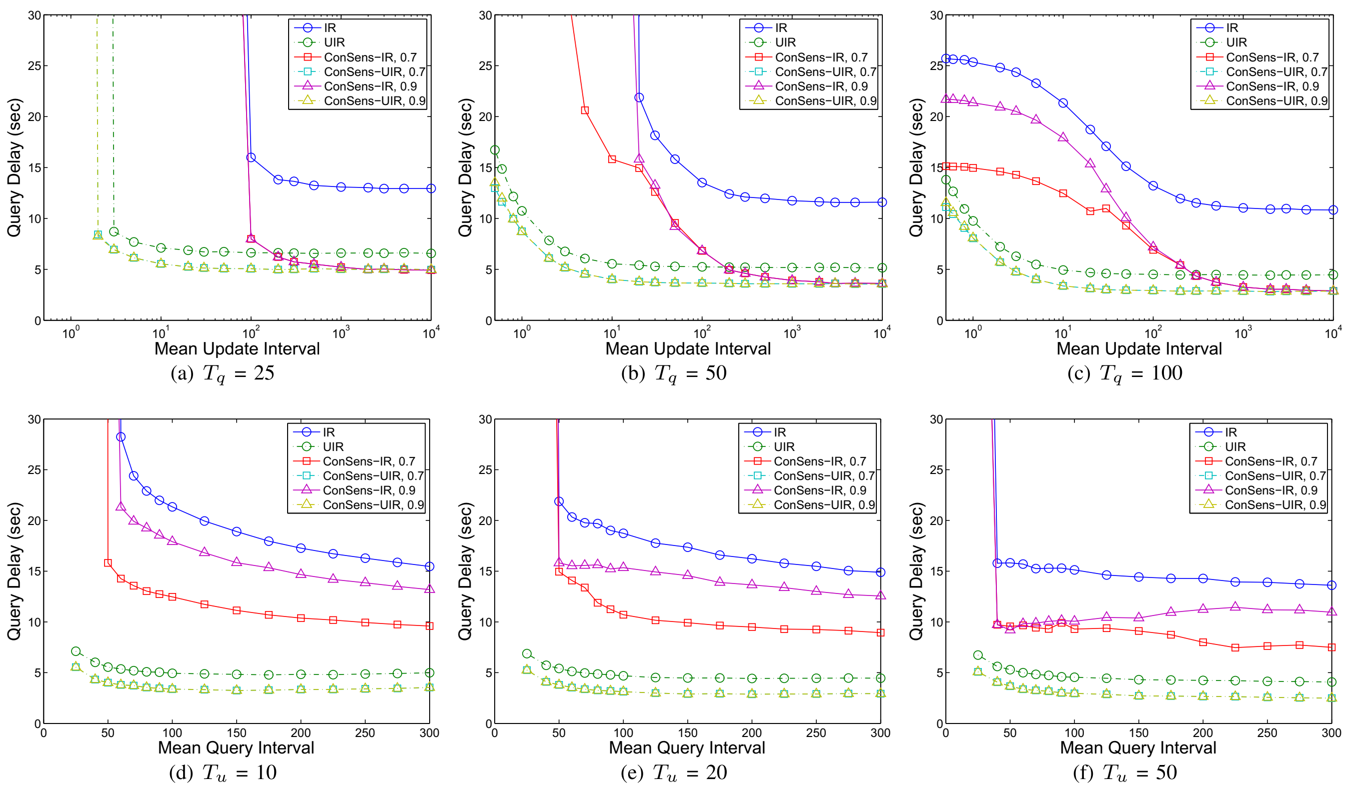

Fig. 7. A comparison of query delay as a function of mean update and query intervals ($\tau = 0.7$ and 0.9).

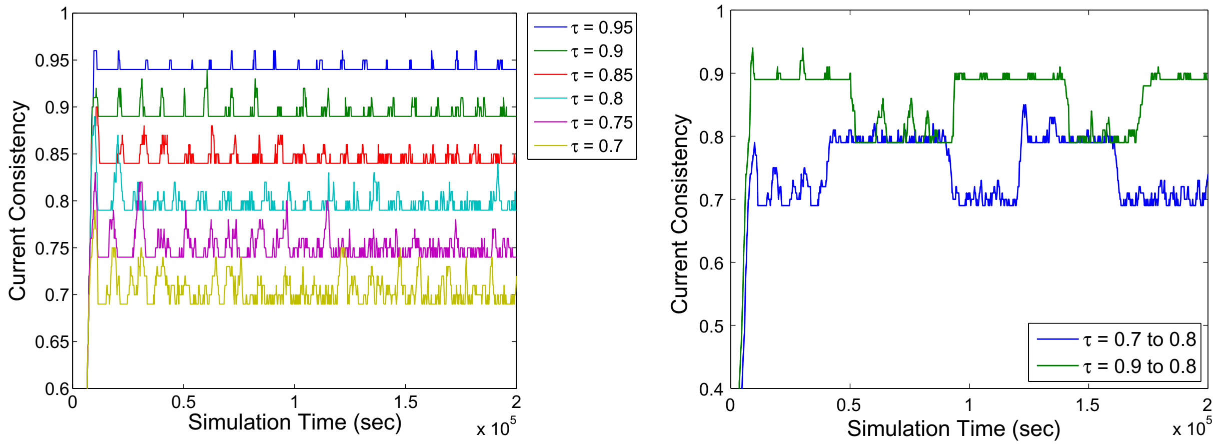

Fig. 8. Consistency changes over the entire simulation period in the ConSens-IR scheme (T_q = 100 and T_u = 10). In (a), a set of target consistencies is used from $\tau = 0.9$ to $\tau = 0.7$ with 0.05 step. In (b), each node flexibly changes its target consistency either between $\tau = 0.9$ to 0.8 or $\tau = 0.7$ to 0.8.

item is still valid when the incoming IR (or UIR) arrives; and (ii) The consistency condition ($\tau \leqslant \tau^{cur}$) is satisfied, and a cached data item used in advance for answering a query turns out valid after validating with the incoming IR (or UIR). In Fig. 9, both ConSens-IR and ConSens-UIR schemes do not affect the cache hit much. They show the similar cache hits with the IR and UIR schemes, respectively, regardless of the query intervals. As shown in Fig. 9(a)–(c), the ConSens-IR scheme shows slightly lower cache hit than the IR scheme depending on the target consistencies.

4.2.4. The number of lazy requests

We analyze the ConSens scheme in terms of the number of lazy requests. A lazy request is counted, if the consistency condition ($\tau \leqslant \tau^{cur}$) is met and an invalid cached data item is used to answer a query. In this paper, the ConSens scheme guarantees that only a limited number of invalid cached data items is accessed before the current consistency drops below the target consistency. If the current consistency becomes below the target consistency, invalid cached data items are not used but instead an uplink *request* is sent to the server to download the latest updated data item.

In Fig. 10, the ConSens-UIR scheme shows predictable performance for the entire update and query intervals. As the update and query intervals increase, the number of lazy requests reduces because more valid cached data items are available. In the ConSens-IR scheme, however, high peak points in the middle of update and query intervals are observed. As shown in Fig. 10(a) and (d), when the update interval is low, more cached data items become invalid and the current consistency tends to become below the target consistency frequently. Thus, more uplink *request* messages are sent but the less number of lazy requests is occurred. In Fig. 10(c) and (f), when the update interval is high, more cached data items remain valid and most likely the consistency condition ($\tau \leqslant \tau^{cur}$) is satisfied. This also leads to the less number of lazy requests. Fig. 10(b) and (e) show the intermediate performance. The lazy request does not occur neither at very low nor very high update interval. The query interval also affects the number of lazy requests.

The ConSens-IR scheme shows a consistency-sensitivity in Fig. 10. When the target consistency reduces, the number of lazy requests increases because more invalid cached data items can be accessed to answer more queries.

4.2.5. The number of opportunistic accesses

We measure the number of opportunistic accesses as a function of update and query intervals. An opportunistic access occurs, if the consistency condition ($\tau \leqslant \tau^{cur}$) is met and a cached data item marked as valid is accessed to answer a query. In Fig. 11(a), when the query interval is low, the ConSens-UIR scheme shows more number of opportunistic accesses as the update interval increases. Due to the aggressive caching of the data items broadcasted from the server in the UIR-based approach, each node has the high probability of accessing valid cached data items. Thus, this can lead to the high number of opportunistic accesses. As shown in Fig. 11(b) and (c), when the query interval is high, the number of opportunistic accesses reduces. In the ConSens-IR scheme, more numbers of opportunistic accesses are observed in the less target

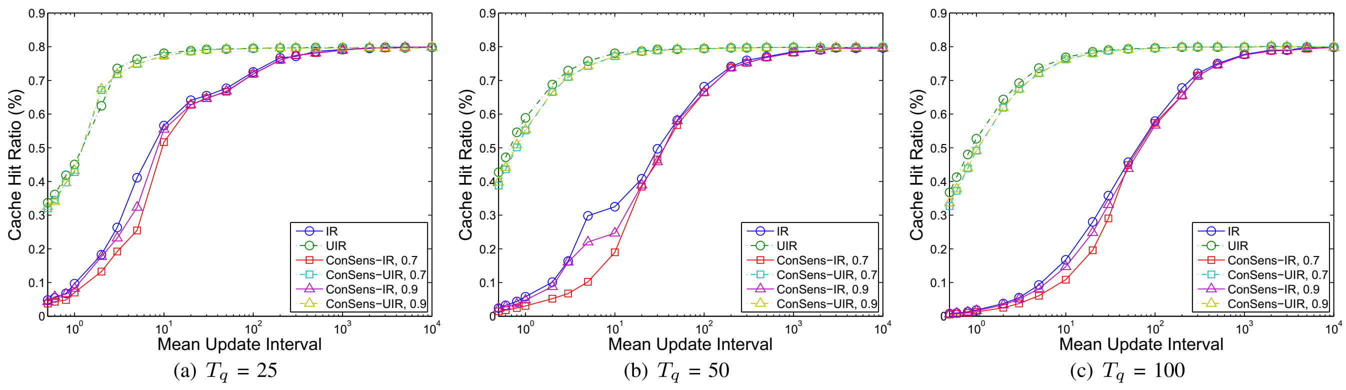

Fig. 9. A comparison of cache hit ratio (%) as a function of mean update interval ($\tau = 0.7$ and 0.9).

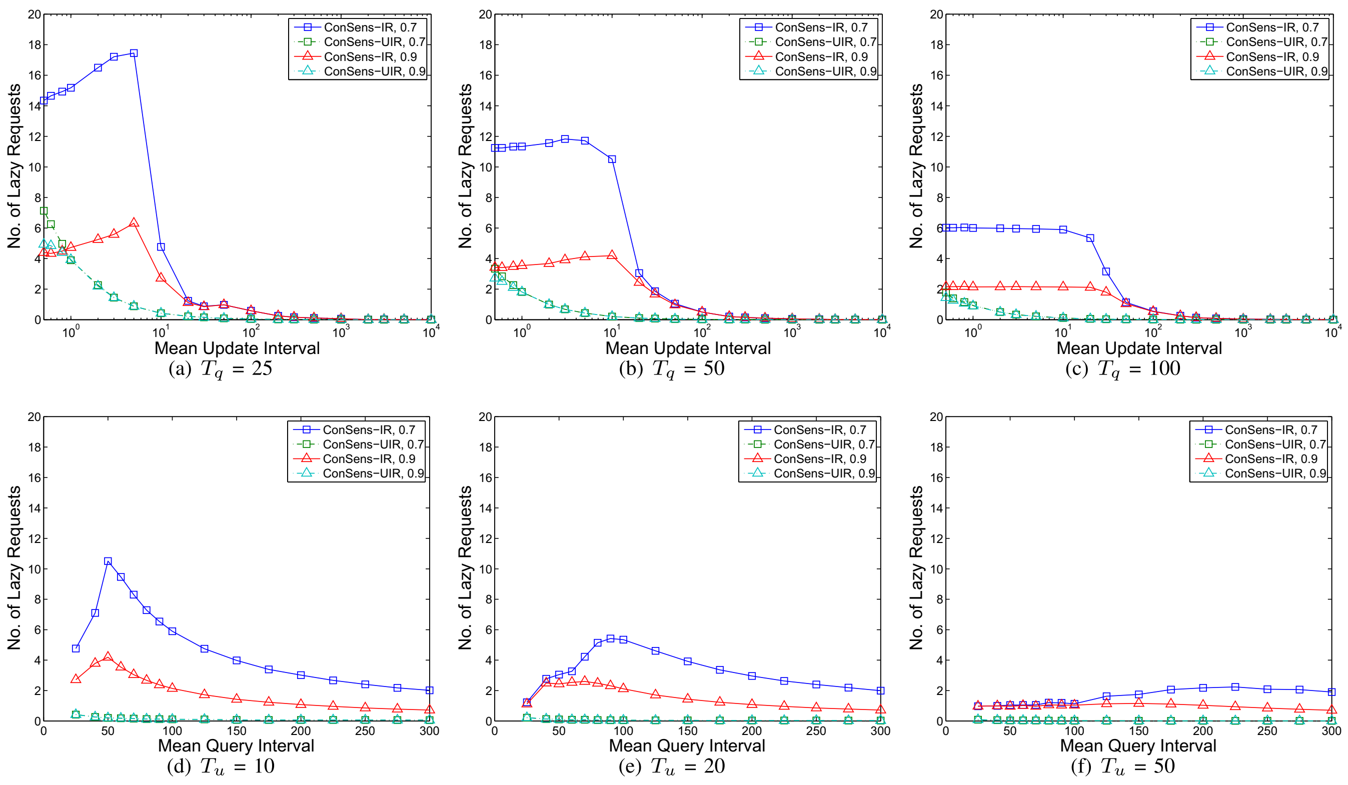

Fig. 10. A comparison of number of lazy requests per IR interval as a function of mean update and query intervals ($\tau = 0.7$ and 0.9).

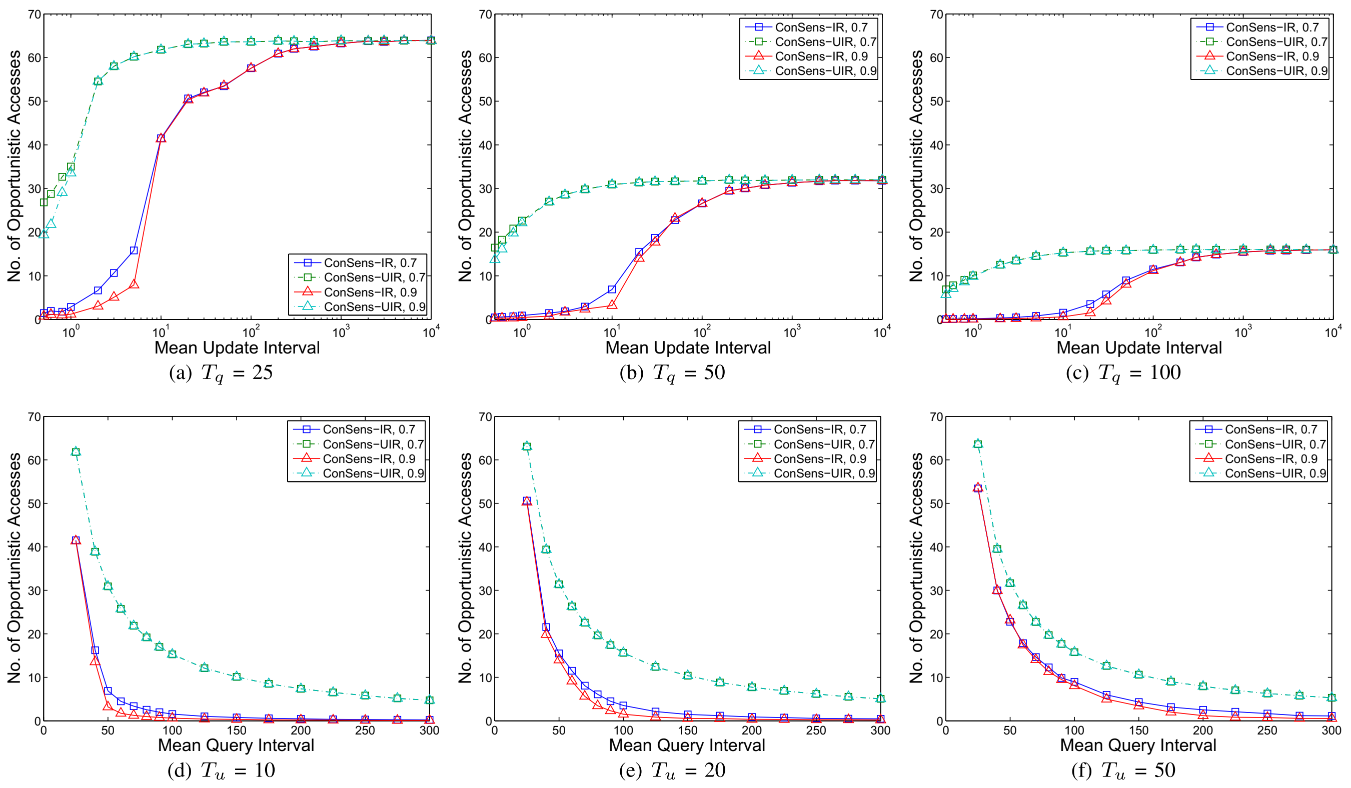

Fig. 11. A comparison of number of opportunistic access per IR interval as a function of mean update and query intervals ($\tau = 0.7$ and 0.9).

consistency ($\tau = 0.7$) because of more chances in accessing cached data items. In Fig. 11(d)–(f), the ConSens-UIR scheme shows the high number of opportunistic accesses for the entire query and update intervals.

4.2.6. The effect of multiple data transmission

Finally, we evaluate the MDT scheme in terms of query delay as a function of mean update and query intervals. Then we investigate the impact of enhancement on the IR, UIR, ConSens-IR, and

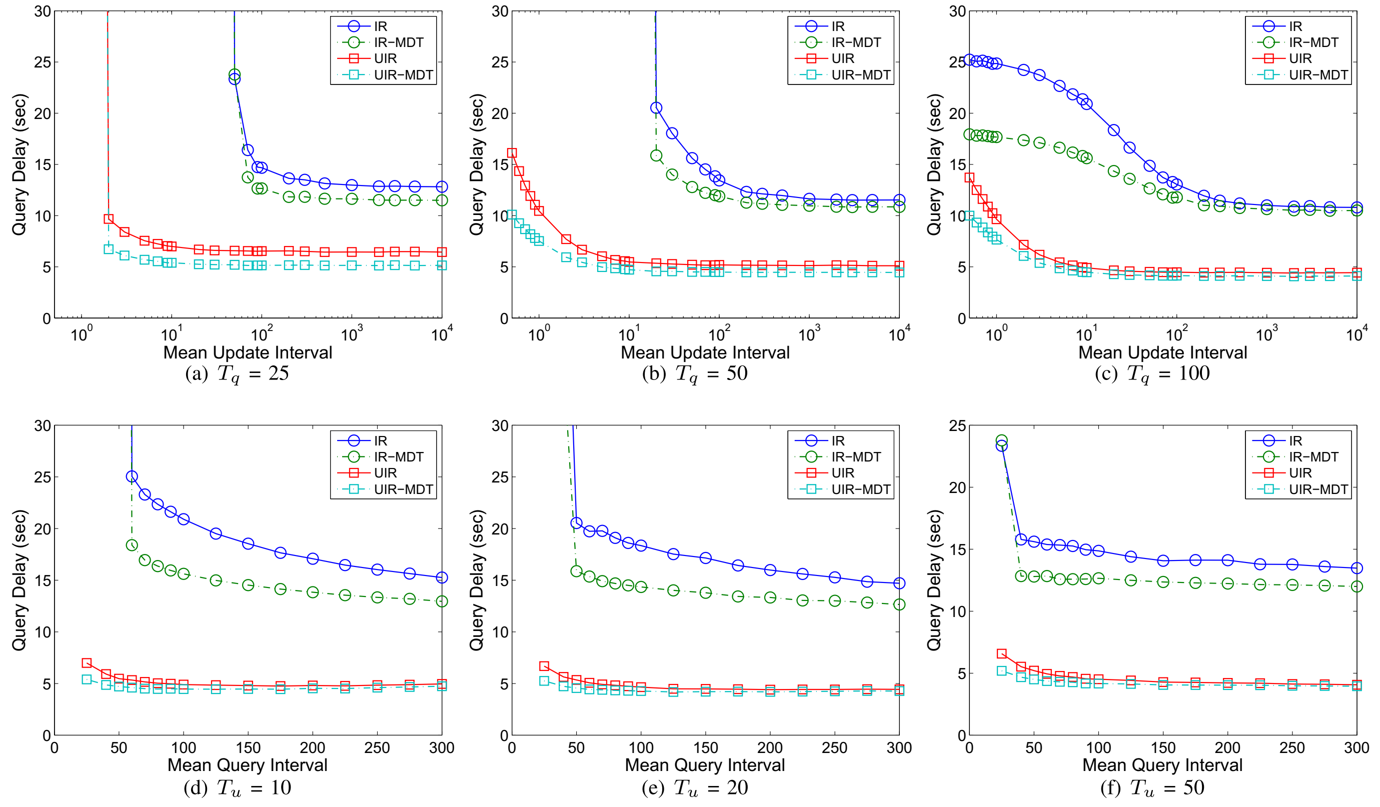

Fig. 12. A comparison of query delay as a function of mean update and query intervals.

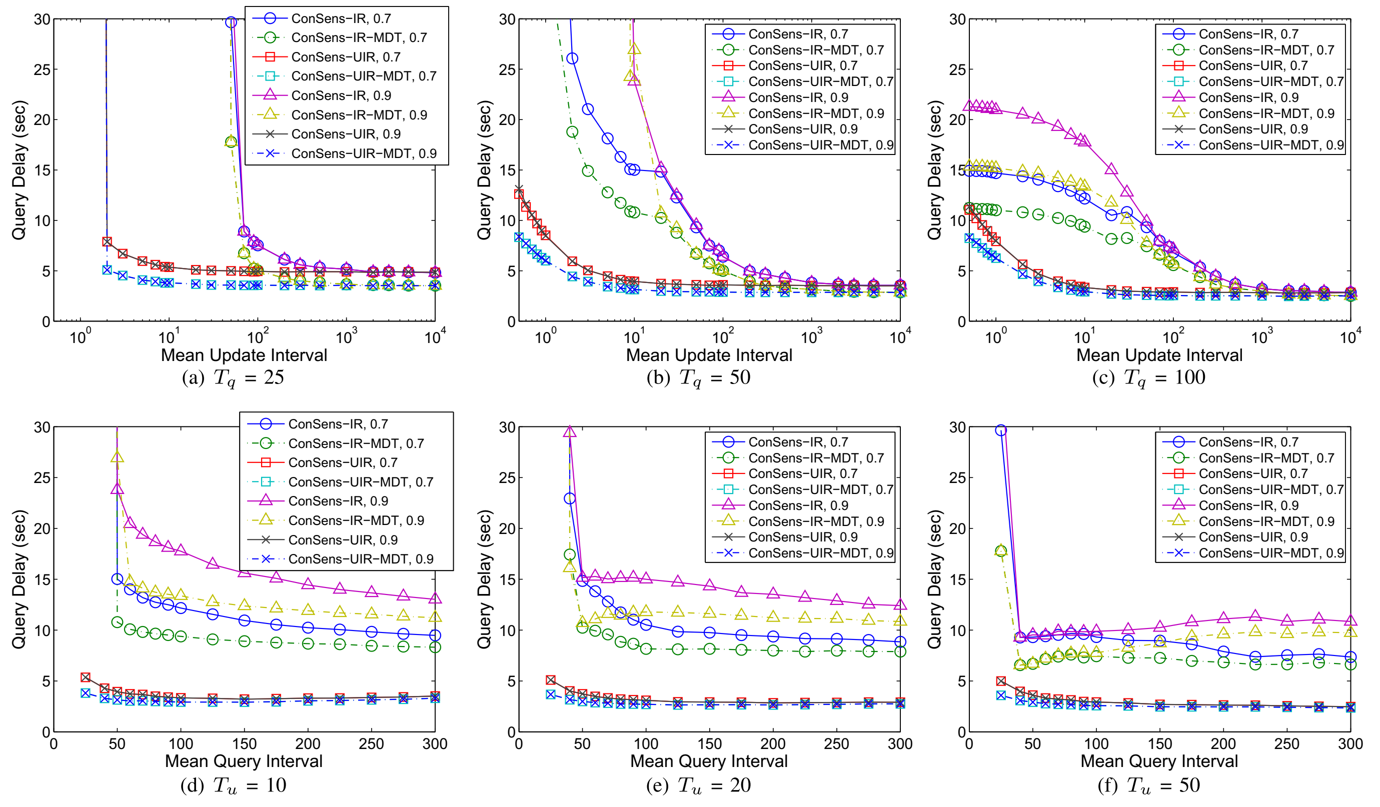

Fig. 13. A comparison of query delay as a function of mean update and query intervals ($\tau = 0.7$ and 0.9).

ConSens-UIR schemes. In Fig. 12(a)–(c), both IR-MDT and UIR-MDT schemes show lower query delay than that of both IR and UIR schemes, respectively for the entire update intervals. In Fig. 12(d)–(f), we can also observe lower query delay of both IR-MDT and UIR-MDT schemes for the entire query intervals. The query delay reduces because each node can immediately answer a query as soon as its requested data item arrives. This is different from the conventional approach, where each node blindly waits until the entire requested data items are received. In Fig. 12, the MDT scheme shows more impact on the IR scheme than the UIR scheme. Due to the aggressive caching of the requested data items, the UIR scheme does not generate many uplink *request* messages compared to the IR scheme. Thus, the UIR scheme has little space to further reduce the query delay with the MDT scheme.

In Fig. 13(b), and (c), the ConSens-IR-MDT scheme shows lower query delay than the ConSens-IR scheme. When the query interval is low, however, the MDT scheme does not affect the query delay in both ConSens-IR and ConSens-UIR schemes as shown in Fig. 13(a). In particular, the MDT scheme does not affect the query delay in the ConSens-UIR scheme regardless of the query and update intervals in Fig. 13. As shown in Fig. 12, this is because the UIR scheme is not affected much by the MDT scheme due to its aggressive caching. In addition, the proposed ConSens scheme deploys the lazy request approach and discourages the uplink request messages. Thus, there is no space to further reduce the query delay in the ConSens-UIR-MDT scheme, regardless of the target consistencies.

In Fig. 13(d)–(f), we can also observe lower query delay of ConSens-IR-MDT scheme than the ConSens-IR scheme for the entire query intervals. As aforementioned, the MDT scheme does not affect the query delay in the ConSens-UIR scheme. The ConSens-IR-MDT scheme with $\tau = 0.7$ and 0.9 shows lower query delay than the ConSens-IR scheme with $\tau = 0.7$ and 0.9, respectively. Thus, the MDT scheme can further reduce query delay in the existing IR and UIR schemes as well as the ConSens-IR scheme with different target consistencies.

In summary, the proposed ConSens scheme can adaptively balance the data accessibility and query delay based on the user-defined consistency through the opportunistic access and lazy request techniques. Also the proposed MDT scheme can further reduce the query delay.

5. Concluding remarks

Most of the existing IR- or UIR-based cache invalidation schemes implicitly assume a strong consistency. However, this assumption may incur high query delay. To deal with a flexible consistency and support applications with diverse consistency requirements, we proposed a consistency-sensitive cache invalidation scheme, called *ConSens*. In the ConSens scheme, each node can define its own consistency level flexibly. We also proposed both *lazy request* and *opportunistic data access* techniques to effectively balance the data accessibility and query latency. In addition, we proposed a *multiple data transmission* (MDT) scheme to further reduce the query delay. The extensive simulation results indicate that the proposed schemes can provide better performance than two prior cache invalidation schemes with respect to the query delay and opportunistic data access.

We envision that the proposed techniques can be integrated into various tactical multi-hop networks in the presence of mobility. For example, a military platoon is exercising and each soldier communicates with other soldiers or a leader. Then the problem becomes how each soldier can maximize its data accessibility and minimize its query delay against a temporal network disconnection. The proposed techniques can also be embedded into mobile apps (e.g., e-textbook, multimedia play, or file sharing service) that require different levels of data synchronization to save scarce wireless resources.

Acknowledgments

This work was published in part at the Military Communications Conference (MILCOM 2011) – Track 2: Network Protocols and Performance [36].

References

[1] Statista Inc., Number of smartphone users in the U.S. from 2010 to 2016, <http://www.statista.com/statistics/201182/>.
[2] Statista Inc., Smartphone users as percentage of all mobile users in the U.S. from 2010 to 2016, <http://www.statista.com/statistics/201184/>.
[3] Apple Inc., Apple Hits 50 Billion Apps Served, May 2013.
[4] D. Barbara, T. Imielinksi, Sleepers and workaholics: caching strategies for mobile environments, in: Proc. ACM SIGMOD, 1994, pp. 1–12.
[5] G. Cao, A scalable low-latency cache invalidation strategy for mobile environments, in: Proc. ACM MOBICOM, 2000, pp. 200–209.
[6] S. Khurana, A. Kahol, S. Gupta, P. Srimani, A strategy to manage cache consistency in a distributed mobile wireless environment, in: Proc. IEEE International Conference on Distributed Computing Systems (ICDCS), 2000, pp. 530–537.
[7] Z. Wang, M. Kumar, S.K. Das, H. Shen, Investigation of cache management strategies for multi-cell environments, in: Proc. Fourth International Conference on Mobile Data Management (MDM), 2003, pp. 29–44.
[8] H. Chen, Y. Xiao, X. Shen, Update-based cache access and replacement in wireless data access, IEEE Trans. Mobile Comput. 5 (12) (2006) 1734–1748.
[9] W. He, I. Chen, B. Gu, A proxy-based integrated cache consistency and mobility management scheme for mobile IP systems, in: Proc. Advanced Networking and Applications, 2007, pp. 354–361.
[10] Q. Hu, D. Lee, Cache algorithms based on adaptive invalidation reports for mobile environments, Cluster Comput. (1998) 39–48.
[11] S. Lim, W. Lee, G. Cao, C.R. Das, Cache invalidation strategies for internet-based mobile ad hoc networks, Comput. Commun. 30 (8) (2007) 1854–1869.
[12] M.K.H. Yeung, Y. Kwok, Wireless cache invalidation schemes with link adaptation and downlink traffic, IEEE Trans. Mobile Comput. 4 (1) (2005) 68–83.
[13] Bing™ Official site – make better decisions with Bing, <http://www.bing.com/>.
[14] Y. Lin, W. Lai, J. Chen, Effects of cache mechanism on wireless data access, IEEE Trans. Wireless Commun. 2 (6) (2003) 1240–1246.
[15] Y. Huang, J. Cao, Z. Wang, B. Jin, Y. Feng, Achieving flexible cache consistency for pervasive internet access, in: Proc. Pervasive Computing and Communications, 2007, pp. 239–250.
[16] W. Li, E. Chan, D. Chen, S. Lu, Maintaining probabilistic consistency for frequently offline devices in mobile ad hoc networks, in Proc. IEEE International Conference on Distributed Computing Systems (ICDCS), 2009, pp. 215–222.
[17] J. Gwertzman, M. Seltzer, World-wide web cache consistency, in: Proc. USENIX Technical Conference, 1996, pp. 141–152.
[18] Z. Wang, S.K. Das, H. Che, M. Kumar, A scalable asynchronous cache consistency scheme (SACCS) for mobile environments, IEEE Trans. Parallel Distrib. Syst. 15 (11) (2004) 983–995.
[19] H. Zou, N. Soparkar, F. Jahanian, Probabilistic data consistency for wide-area applications, in: Proc. International Conference on Data Engineering (ICDE), 2000, pp. 85–85.
[20] S. Zhu, C. Ravishankar, Stochastic consistency and scalable pull-based caching for erratic data stream sources, in Proc. the 30th VLDB Conference, 2004, pp. 192–203.
[21] J. Cao, Y. Zhang, G. Cao, L. Xie, Data consistency for cooperative caching in mobile environments, IEEE Comput. (2007) 60–66.
[22] T. Hara, S. Madria, Data replication for improving data accessibility in ad hoc networks, IEEE Trans. Mobile Comput. 5 (11) (2006) 1515–1532.
[23] G. Karumanchi, S. Muralidharan, R. Prakash, Information dissemination in partitionable mobile ad hoc networks, in: Proc. IEEE Symposium on Reliable Distributed Systems (SRDS), 1999, pp. 4–13.
[24] J. Luo, J. Hubaux, P.T. Eugster, PAN: providing reliable storage in mobile ad hoc networks with probabilistic quorum systems, in: Proc. ACM MobiHoc, 2003, pp. 1–12.
[25] Y. Sawai, M. Shinohara, A. Kanzaki, T. Hara, S. Nishio, Quorum-based consistency management among replicas in ad hoc networks with data update, in: Proc. ACM Symposium on Applied Computing, 2007, pp. 955–956.
[26] W. Zhang, G. Cao, Defend against cache consistency attacks in wireless ad hoc networks, in: Proc. Mobile and Ubiquitous Systems (Mobiquitous), 2005, pp. 12–21.
[27] L. Yin, G. Cao, Supporting cooperative caching in ad hoc networks, IEEE Trans. Mobile Comput. 5 (1) (2006) 77–89.
[28] H. Artail, H. Safa, K. Mershad, Z. Abou-Atme, N. Sulieman, COACS: a cooperative and adaptive caching system for MANETs, IEEE Trans. Mobile Comput. 7 (8) (2008) 961–977.

[29] H. Safa, H. Artail, M. Nahhas, A cache invalidation strategy for mobile networks, Comput. Commun. (2010) 168–182.

[30] N. Dimokas, D. Katsaros, Y. Manolopoulos, Cache consistency in wireless multimedia sensor networks, Ad Hoc Networks (2010) 214–240.

[31] S. Lim, C. Yu, C.R. Das, Cache invalidation strategies for internet-based vehicular ad hoc networks, Comput. Commun. 35 (3) (2012).

[32] T. Hara, Effective replica allocation in ad hoc networks for improving data accessibility, in: Proc. IEEE INFOCOM, 2001, pp. 1568–1576.

[33] S. Lim, W. Lee, G. Cao, C.R. Das, A novel caching scheme for improving internet-based mobile ad hoc networks performance, Ad Hoc Networks 4 (2) (2006) 225–239.

[34] The CSIM User Guides, <http://www.mesquite.com/userguidespage.htm>.

[35] M.K.H. Yeung, Y. Kwok, A game theoretic approach to power aware wireless data access, IEEE Trans. Mobile Comput. 5 (8) (2006) 1057–1073.

[36] S. Lim, Y. Lee, M. Min, ConSens: consistency-sensitive opportunistic data access in wireless networks, in: Proc. MILCOM, 2011, pp. 804–809.