ConSens: Consistency-Sensitive Opportunistic Data Access in Wireless Networks

Sunho Lim and Yumin Lee
T[2]WISTOR: TTU Wireless Mobile Networking Laboratory
Dept. of Computer Science
Texas Tech University
Lubbock, TX 79409
{sunho.lim, yumin.lee}@ttu.edu

Manki Min
Dept. of Electrical Engineering and Computer Science
South Dakota State University
Brookings, SD 57007
manki.min@sdstate.edu

***Abstract*—**
In order to fulfill the users' insatiable interests in accessing the Internet services and information wirelessly, one of key optimization techniques is caching frequently accessed the data items in a local cache. However, a strong consistency is implicitly assumed in most caching schemes but it may frequently occur a long query delay. In this paper, we propose a consistency-sensitive data access scheme to support applications' diverse consistency requirements, called ConSens, where a user can flexibly set its own consistency level. We also propose a lazy request and opportunistic data access techniques to effectively balance the data accessibility and query delay. Extensive performance studies indicate that the proposed techniques reduce the query delay and increase the number of opportunistic accesses significantly.

***Index Terms*—Cache consistency, cache invalidation, lazy request, opportunistic data access, wireless networks.**

I. Introduction

Due to recent technological advances in high-speed wireless network, mobility support, and portability, wireless mobile device has become an essential tool to access the Internet services and information wirelessly. For example, smart cell phone is receiving a great attention and becoming popular. 45 million users[1] out of 234 million subscribers in the US already own a smart phone in 2010 [1]. 47% and 40% of all adult Americans are the wireless Internet users with laptop and cell phone, respectively in 2010 [2]. Approximately six in ten American adults access the Internet wirelessly anyhow. Typically, the explosive growth in applications running on the smart cell phone (i.e., App[2]) has fueled users to easily access the Internet anytime and anywhere.

In order to fulfill the users' insatiable interests in accessing the Internet wirelessly, one of key optimization techniques is caching frequently accessed data items in a local storage. The caching technique can relieve network traffic and improve information accessibility and availability. In particular, the communication cost in terms of scarce wireless bandwidth and battery energy can be reduced significantly. A great deal of research effort has been devoted in developing various caching methodologies in wireless networks.

In most caching schemes, a strong consistency is implicitly assumed. Thus, a query is answered by the latest updated data item from either a server or local cache. In this paper, we first note that ensuring a strong consistency is not always a prompt and critical requirement. This is because a consistency requirement may be different depending on an update sensitivity of data item. For example, maps, video clips, e-flyers, and weather information, they neither update sensitive information nor need to reflect a current update of the source in an urgent manner. Occasional inconsistency between the source data item and its cached copy would be acceptable. For the update sensitive information, on the other hand, a cached copy must be updated *before* answering a query when the source is updated in the server. For example, news, stock prices, and traffic information, they need to be updated in a real-time manner. Second, ensuring a strong consistency requires a non-negligible long query delay to confirm that both source data item and its cached copy are consistent.

In this paper, we propose a consistency-sensitive data access scheme to support applications' diverse consistency requirements, called *ConSens*, in wireless networks. Under the proposed scheme, each user can flexibly set its own consistency level, called *target consistency*, with the server. This is a quite different approach compared to the conventional caching schemes, where a strong consistency is implicitly assumed and/or entire users have the same consistency level with the server. Our contributions are summarized in three-fold:

- First, we propose a user-defined cache consistency scheme, in which each user can set its own consistency level and maintain the consistency probabilistically. We introduce a consistency condition to decide the current consistency level based on the number of invalid cached data items accessed during a consistency window.
- Second, we also propose both *lazy request* and *opportunistic data access* techniques to balance the data accessibility and query latency efficiently. Based on the

[1]In this paper, we use the mobile node to refer to a wireless mobile device or a person who carries it. Thus, we use the terms users and mobile node (later in short, node) interchangeably.

[2]According to Apple AppStore, more than 250K Apps have been uploaded and more than 6.5 billion downloads have been made so far, September 2010. [3].

consistency condition, each user can judiciously access the cached data items without either sending a request to the server or waiting for the next invalidation report (IR) or updated IR (UIR) from the server.
- Third, we integrate the proposed techniques into the existing IR- and UIR-based caching algorithms. We modify two major cache invalidation schemes [4], [5] to implement the consistency-sensitive data access.

We conduct a simulation-based performance evaluation of the proposed scheme to observe the impact of query interval, update interval, and target consistency on the communication performance. The simulation results show that the proposed scheme not only can reduce the query delay but also can increase the number of opportunistic accesses significantly.

The rest of paper is organized as follows. The prior work is analyzed and the proposed techniques are presented in Sections II and III, respectively. Section IV is devoted to performance evaluation and comparison. Finally, we conclude the paper with future research direction in Section V.

II. Related Work

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

Various invalidation report (IR) based cache invalidation techniques have been proposed for cellular networks [4], [5], [6], [7], where a base station[3] (BS) broadcasts an IR periodically (e.g., every L second, where L is a broadcast interval). A node can answer a query with a cached copy if it is still valid based on the incoming IR. This ensures a strong consistency but a long query delay ($\frac{L}{2}$ in average) is unavoidable, because of periodic IR broadcast. Although an updated IR (UIR) [5] can further reduce the query latency, non-negligible query delay ($\frac{L}{2 \cdot m}$ in average, where m is a number of UIRs) is still expected. The strong consistency guarantees that the latest updated data item is accessed, but the consistency maintenance cost in terms of wireless bandwidth, battery energy, communication overhead, and query delay will be increased.

Time-to-live (TTL) based approaches [8], [9] have been widely used for implementing a weak consistency, where a server predicts and sets a lifetime of each data item. When a node generates a query, a cached copy can be used to answer the query only if its TTL value has not expired yet. Otherwise, the node sends a *request* message to the server for downloading the latest updated data item. The weak consistency can reduce the query latency but predicting the TTL value is not trivial.

[3]From a node's point of view, because a BS is transparent to a database server, we use the terms BS and database server (later in short, server) interchangeably.

Hybrid consistency based techniques [10], [11], [12], [13] have been proposed to support applications that require a certain level of consistency with the server. In [11], a probabilistic delta consistency scheme is deployed in a hybrid wireless network architecture, where both stochastic [13] and delta [10] consistencies are proposed. In this scheme, each node can specify a consistency requirement by adjusting a *timeout* value associated with each data item. When a server updates a data item, it broadcasts an *invalidation* message to the nodes that store the cached copy. If the server does not receive all the *acknowledgment* messages from the nodes, it delays an update operation on the source data item by the timeout. By adjusting the timeout value, various cache consistency schemes can be achieved. Like the TTL value, however, predicting global update information of timeout value is very difficult (if it is not impossible).

In summary, little effort has been devoted in exploring a consistency-sensitive data access methodology that can balance the data accessibility and query latency, in which each node can flexibly set its own consistency level with the server.

III. The Proposed Cache Consistency Scheme

In this section, we first briefly introduce a system model and then present our proposed consistency-sensitive scheme and its related techniques.

A. System Model

We consider a cellular network, where a set of nodes is located in a cell and communicates with a BS by wireless links. Each node can move in any direction, make information 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. A database may be attached to a BS, 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 the nodes for read-only requests. A node can cache a limited number of data items, because the size of the cache space (N_c) in a node is relatively small compared to the total number of data items in the database.

In order to maintain cache consistency, a server periodically broadcasts an IR containing 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 an update history witnessed during the last $L \cdot |w_b|$ broadcast intervals, where w_b is a broadcast window. A set of UIRs can be broadcasted between IRs to further reduce the query delay. Due to the periodic IR or UIR broadcast, nodes can operate in a *doze* mode to save battery energy.

We also assume that each node is able to set its own consistency level with the server. The consistency level can be presented in terms of a threshold value that indicates a tightness of consistency between the source data item and its cached copy. From the implementation point of view, nodes can directly type different threshold values into, for example, a

system configuration to setup the target consistencies depending on the applications.

B. Consistency-Sensitive Data Access

Most of the existing IR- and UIR-based schemes have been focused on solving a long disconnection problem, reducing the query delay, or utilizing the broadcast bandwidth under the following implicit assumptions: (i) A strong consistency; and (ii) All nodes have the same consistency level with the server. To support applications' diverse consistency requirements, in this paper, we relax these assumptions and propose a consistency-sensitive data access scheme.

The basic idea of the proposed scheme, ConSens, is that each node can set its own consistency level with the server flexibly and adjust its data item access and cache invalidation operations adaptively. The ConSens works as follow. First, each node can set its own consistency level, called a *target consistency* (τ, $0 < \tau \leq 1$). Here, τ = 1 implies a strong consistency and τ can have different values depending on the consistency requirements. For example, if a node sets τ = 0.8, eight out of ten queries should be answered by the latest updated data items from either the server or local cache. Each node also maintains a number of invalid data item accesses (N^{inval}) observed during a period, which can be represented in terms of a number of queried data items, called *consistency window* (w_c). Then a cumulative current consistency (τ^{cur}) can be calculated by $\tau^{cur} = 1 - \frac{\sum_{u=0}^{|w_c|} \text{Invalid}[u]}{|w_c|}$, where Invalid$[u]$ is an array for marking (e.g., 1) invalid queries. Also $0 \leq u < |w_c|$ and all array elements are initially set to 0. Here, τ^{cur} changes slowly as $|w_c|$ increases. To reflect changes of the current consistency to both data access and cache invalidation operations promptly, in this paper, we only consider the last $|w_c|$ number of queried data items.

Let say, a node sets τ (e.g., $\tau < 1$) and maintains τ^{cur}. When a query is generated which can be answered by a cached copy, the node examines a consistency condition, $\tau \leq \tau^{cur}$. If the condition is satisfied, then the node can immediately use the cached copy for answering the query without waiting for the next IR (or UIR) (see Figure 1). In this paper, the ConSens is designed to keep τ^{cur} around the user defined τ. Thus, we try to avoid too high or too low τ^{cur} compared to τ but to achieve a τ^{cur} which fluctuates around τ. In the consistency condition, since τ^{cur} is above τ, there is a space that the node can use the cached copy in advance for answering the query without knowing its validity. However, the source data item of the cached copy stored in the server could be updated *a priori*. Thus, after answering the query, the node needs to verify the answered cached copy whether it was valid or not with the incoming IR (or UIR). According to the validity, the node recalculates τ^{cur}. Note that if the cached copy turns out valid, then the node has *opportunistically* accessed the cached data items and reduced the query delay. On the other hand, if the consistency condition ($\tau \leq \tau^{cur}$) is not met, the node should wait for the next IR (or UIR) to validate the cached copy. Since τ^{cur} is below τ, the latest updated data item stored in either the server or local cache should be used for answering the query to increase τ^{cur} closely enough to τ.

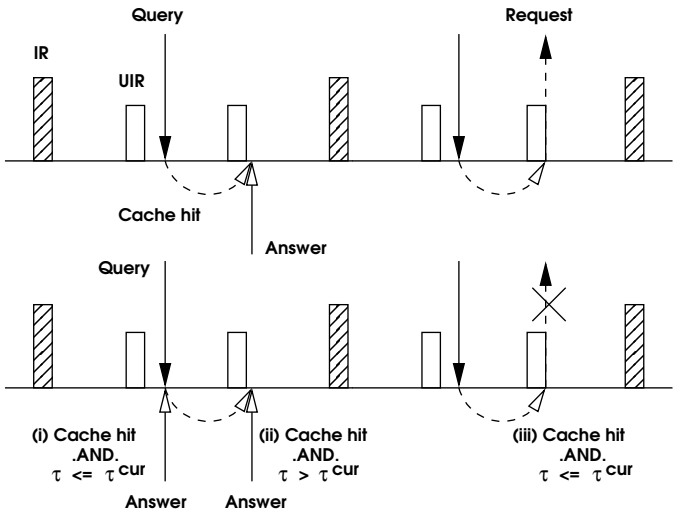

Fig. 1. Unlike to the conventional approach (upper), the ConSens (bottom) can opportunistically access a cached copy for answering a query without waiting for the next IR (or UIR), if the consistency condition ($\tau \leq \tau^{cur}$) is satisfied ((i) case). Each node can also postpone sending an uplink *request* message to the server, if the consistency condition ($\tau \leq \tau^{cur}$) is still met ((iii) case). If the consistency condition ($\tau \leq \tau^{cur}$) is not satisfied, the ConSens follows the conventional approach (i.e., waiting for the next IR (or UIR)) ((ii) case).

Second, in most IR- and UIR-based schemes, when a node generates a query and its cached copy is stored in a local cache, the node should wait for the incoming IR (or UIR) to validate the cached copy before answering the query. Depending on the validity, the node can either access the cached copy directly or send an uplink *request* message to the server for downloading the latest updated data item. In this paper, we propose a *lazy request* approach to reduce the query delay and increase the data accessibility. In this approach, as far as the consistency condition ($\tau \leq \tau^{cur}$) is satisfied, an invalid cached copy can be used for answering a query. Here, the ConSens guarantees that a limited number of invalid cached data items can be accessed before τ^{cur} drops below τ. If τ^{cur} becomes below τ, an uplink *request* message will be sent to the server for downloading the latest updated data item.

For example, in case of a cached copy marked as invalid, the node does not actively download and cache the latest updated data item. The rationale behind this is that even the latest updated data item could be updated again at the server before it is used for answering a query later. Thus, the node tries to postpone sending an uplink *request* until the condition becomes $\tau > \tau^{cur}$ (see Figure 1). If either the consistency condition is not satisfied or invalid cached copy is already used once, then the node sends an uplink *request* message to the server.

Third, 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 was broadcasted and its timestamp, respectively. Unlike to the original scheme [5], we include t' to decide whether an accessed cached copy was valid for the purpose of the proposed scheme. Also a false alarm [5] can be avoided.

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 $0 \leq k < N_c$.
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.
Invalid$_i[u]$: An array for marking (e.g., 1) invalid queries in n_i, where $0 \leq u < |w_c|$ 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 *request* of d to the server;
  else {
    if ($\tau_i \leq \tau^{cur}_i$) {
      if ($d^c_{i,k}$ = invalid) { /* lazy request */
        Invalid$_i[u$++ % $|w_c|]$ = 1;
        $\tau^{cur}_i = 1 - \frac{\sum_{u=0}^{|w_c|} \text{Invalid}_i[u]}{|w_c|}$;
      }
      else /* opportunistic data access */
        Queue d into Q^{opp}_i and wait for the next IR (or UIR);
      Use $d^c_{i,k}$ to answer the q;
    }
    else
      Queue d into Q_i and wait for the next IR (or UIR);
  }

(B) When n_i receives an IR (or UIR),
  for $\forall id(d) \in$ IR (or UIR) **do** { /* check cached data items' validity */
    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 ($t^c_{i,k} < t^s$)
      Invalid$_i[u$++ % $|w_c|]$ = 0;
    else
      Invalid$_i[u$++ % $|w_c|]$ = 1;
    $\tau^{cur}_i = 1 - \frac{\sum_{u=0}^{|w_c|} \text{Invalid}_i[u]}{|w_c|}$;
  }

Fig. 2. The pseudo code of the proposed scheme.

In summary, the proposed scheme trades off 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 validating them. Unlike to the prior IR-based approach, depending on the consistency condition, each node neither wait for the next IR (or UIR) nor send an uplink *request* message to the server blindly. The pseudo code of the proposed scheme is described in Figure 2.

IV. Performance Evaluation

In this paper, we develop a customized discrete-event simulator using CSIM20 [14] to conduct our experiments.

A. Simulation Testbed

In order to examine the proposed idea, we assume that both query and update inter arrival times follow the exponential distribution. The entire data items stored in the database are classified into two subsets, hot and cold data items. 80% of query requests are for hot data items, while 33% of update requests are for hot data items but they are uniformly distributed within the hot subset. Each node caches 10% of the data items in the database. When a cache is full, we use the least recently used (LRU) cache replacement policy. An advanced cache admission or replacement policy such as [15], and a multi-cell case [16] 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 and the consistency window (w_c) is the last 100 queried data items. The important simulation parameters are summarized in Tab. I.

TABLE I
Simulation parameters

Parameter	Value
Number of nodes	100
Database size (N)	1,000 items
Data item size	2,048 bytes
Hot data items	5% of DB
Cold data items	95% of DB
Mean query interval time (t_q)	25 - 300 seconds
Mean update interval time (t_u)	0.5 - 10,000 seconds
Broadcast bandwidth	20,000 bits/s
Broadcast interval (L)	20 seconds
Broadcast window (w_b)	10 intervals
UIR replicate times	4
Target consistency (τ)	0.7 - 0.95
Consistency window (w_c)	100 queries

B. Simulation Results

In this section, we evaluate the performance of proposed scheme in terms of query latency, 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. For performance comparison, we modify two prior cache invalidation schemes, IR- [4] and UIR-based [5] schemes, to aware consistency sensitivity: *ConSens-IR* and *ConSens-UIR*. We also include the base cases represented as *IR* and *UIR*, respectively.

1) The Query Delay: We evaluate the query delay as a function of update and query intervals. In Figs. 3 (a) and (b), both UIR and ConSens-UIR schemes show better performance than the other two schemes, the IR and ConSens-IR schemes. Similar performance trend can be witnessed in Figs. 3(c) and (d). In the UIR and ConSens-UIR schemes, when the server broadcasts a data item requested from a node, all the other nodes can also receive and cache it. Thus, more queries can be answered by the latest updated cached data items without waiting for the next IR (or UIR), and the query delay reduces.

In particular, the ConSens-IR scheme is sensitive to the target consistency. Due to a long IR broadcast period (e.g., 20 seconds), a query has a high probability of being answered by a cached copy before waiting for the next IR in a low target consistency (e.g., τ = 0.7). In the ConSens-UIR, however, the target consistency does not affect the query delay much. In Figs. 3, the ConSens-IR and ConSens-UIR schemes achieves lower query delay than the other two schemes, the IR and UIR schemes, for entire query and update intervals.

2) The Consistency Changes: We monitor the consistency changes over the entire simulation period. In Fig. 4, after a cold state of simulation (i.e., after each node's empty cache is filled up), each current consistency (τ^{cur}) is fluctuating but it is adjusted to its target consistency (τ). Depending on the

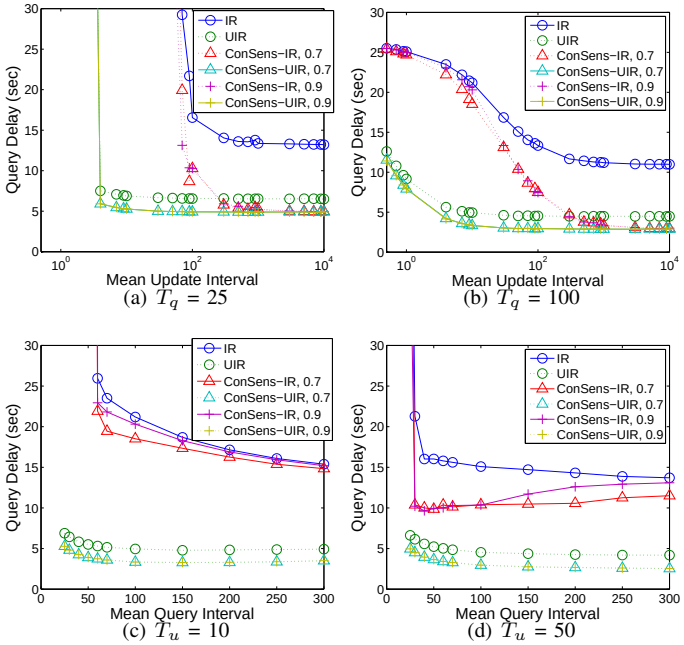

Fig. 3. A comparison of query delay as a function of mean update and query intervals (τ = 0.7 and 0.9).

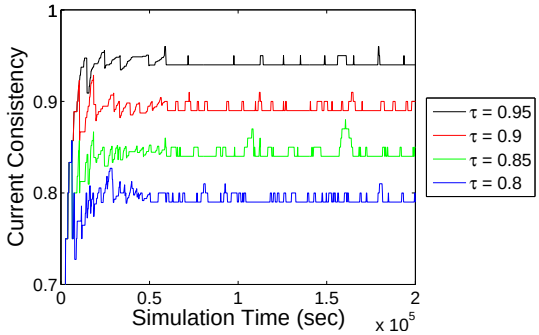

Fig. 4. Consistency changes over the entire simulation period in ConSens-IR (T_q = 100 and T_u = 10). Here, a set of target consistencies is used (τ = 0.95, 0.9, 0.85, and 0.8 from the top).

current consistency, the proposed ConSens-IR and ConSens-UIR schemes can judiciously decide whether to access the cached data items directly or wait for the next IR (or UIR) for validation.

3) The Cache Hit: We evaluate the cache hit as a function of update interval. In the ConSens, a cache hit occurs in the following cases: (i) If the consistency condition ($\tau \leq \tau^{cur}$) is not met, and a cached copy is valid when the next IR (or UIR) arrives; and (ii) If the consistency condition ($\tau \leq \tau^{cur}$) is satisfied, and a cached copy used in advance for answering a query turns out valid when the next IR (or UIR) arrives. In Figs. 5(a) and (b), both ConSens-IR and ConSens-UIR schemes do not affect the cache hit and show almost the same performances with the IR and UIR schemes, respectively, regardless of the different target consistencies and query intervals.

4) The Number of Lazy Requests: We analyze the ConSens in terms of the number of lazy requests. A lazy request is counted, if the consistency condition ($\tau \leq \tau^{cur}$) is met and

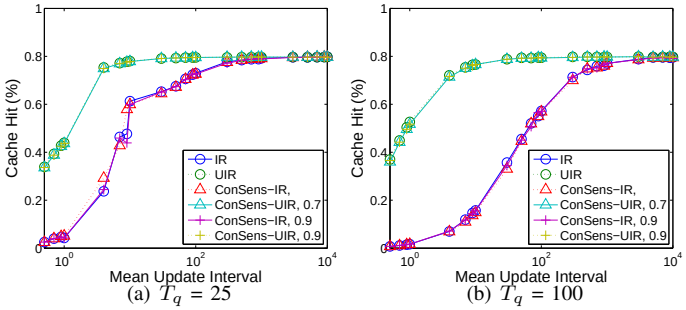

Fig. 5. A comparison of cache hit ratio (%) as a function of mean update interval (τ = 0.7 and 0.9).

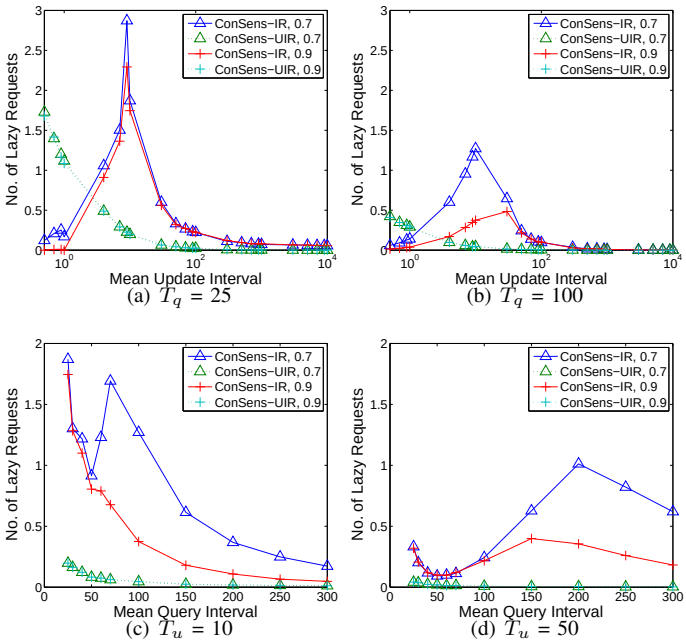

Fig. 6. A comparison of number of lazy requests per IR interval as a function of mean update and query intervals (τ = 0.7 and 0.9).

an invalid cached copy is used for answering a query. In fact, the ConSens guarantees that only a limited number of invalid cached data items are allowed to be accessed before τ^{cur} drops below τ. If τ^{cur} becomes below τ, the invalid cached copy is not used but instead an uplink *request* is sent for downloading the latest updated data item.

In Fig. 6, the ConSens-UIR scheme shows a predictable performance trend 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 in Fig. 6. When the update interval is low, as shown in Figs. 6(a) and (c), more cached data items become invalid and τ^{cur} tends to become below τ frequently. Thus, more uplink *request* messages are sent but less lazy requests are occurred. In Figs. 6(b) and (d), when the update interval is high, most likely more cached data items remain valid and the consistency condition ($\tau \leq \tau^{cur}$) is satisfied. This leads less lazy requests, too. As shown in Fig. 6, the lazy requests are actively occurred at not too low and not too high update intervals. The query interval

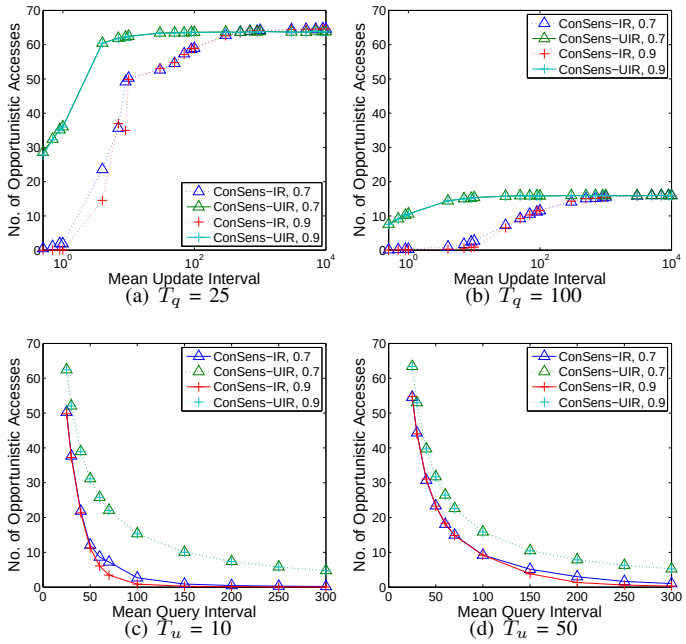

Fig. 7. A comparison of number of opportunistic access per IR interval as a function of mean update and query intervals (τ = 0.7 and 0.9).

also affects the number of lazy requests.

The ConSens-IR scheme shows a consistency-sensitivity in Fig. 6. When the target consistency reduces, the number of lazy requests increases because more invalid cached data items can be used for answering the queries.

5) The Number of Opportunistic Accesses: Finally, we measure the number of opportunistic accesses as a function of update and query intervals. An opportunistic access is occurred, if the consistency condition ($\tau \leq \tau^{cur}$) is met and a cached copy marked as valid is accessed for answering a query. In Fig. 7(a), when the query interval is low, the ConSens-UIR shows more number of opportunistic accesses as the update interval increases. Due to an aggressive caching of the data items broadcasted from the server in the UIR-based approach, each node has a high probability of accessing the valid cached data items. Thus, this can lead high number of opportunistic accesses. When the query interval is high, as shown in Fig 7(b), overall number of opportunistic accesses reduce. In the ConSens-IR scheme, more number of opportunistic accesses are observed in less target consistency (τ = 0.7) because of more chances in accessing valid cached data items. In Figs. 7(c) and (d), the ConSens-UIR shows better performance for the entire query intervals.

In summary, the ConSens provides better performance and balances the data accessibility and query delay through the lazy request and opportunistic access techniques.

V. Concluding Remarks

Most of the existing IR- or UIR-based approaches implicitly assume a strong consistency. However, this assumption may incur high query latency. In this paper, we investigate the techniques to deal with a flexible consistency and support applications with diverse consistency requirements. We proposed a consistency-sensitive data access scheme, called *ConSens*, where each node can define its own consistency level flexibly. In the ConSens, we also proposed both *lazy request* and *opportunistic data access* approaches to effectively balance the data accessibility and query latency. The extensive simulation results indicate that the proposed scheme provides 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 (e.g., mobile ad hoc networks or wireless sensor 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. Under the scenario, we are extending the proposed techniques to see the full potential.

Acknowledgment

This research was supported in part by Startup grant in Dept. of Computer Science at Texas Tech University and US National Science Foundation (CNS-1004210).

References

[1] A. Gonsalves, *Android Phones Steal Market Share*, http://www.informationweek.com, April 2010.
[2] A. Smith, *Mobile Access 2010*, http://pewinternet.org/Reports/2010/Mobile-Access-2010.aspx, Pew Research Center's Internet & American Life Project, July 2010.
[3] Apple Inc., *Apple Special Event*, September 2010.
[4] D. Barbara and 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] M. Yeung and Y. Kwok, "Wireless Cache Invalidation Schemes with Link Adaptation and Downlink Traffic," *IEEE Transactions on Mobile Computing*, vol. 4, no. 1, pp. 68–83, 2006.
[7] S. Lim, W. Lee, G. Cao, and C. R. Das, "Cache Invalidation Strategies for Internet-based Mobile Ad Hoc Networks," *Computer Communications Journal*, vol. 30, no. 8, pp. 1854–1869, 2007.
[8] J. Gwertzman and M. Seltzer, "World-Wide Web Cache Consistency," in *Proc. USENIX Technical Conference*, 1996, pp. 141–152.
[9] Z. Wang, S. K. Das, H. Che, and M. Kumar, "A Scalable Asynchronous Cache Consistency Scheme (SACCS) for Mobile Environments," *IEEE Trans. on Parallel and Distributed Systems*, vol. 15, no. 11, pp. 983–995, 2004.
[10] J. Cao, Y. Zhang, G. Cao, and L. Xie, "Data Consistency for Cooperative Caching in Mobile Environments," *IEEE Computer*, pp. 60–66, 2007.
[11] Y. Huang, J. Cao, Z. Wang, B. Jin, and Y. Feng, "Achieving Flexible Cache Consistency for Pervasive Internet Access," in *Proc. Pervasive Computing and Communications*, 2007, pp. 239–250.
[12] W. Li, E. Chan, D. Chen, and S. Lu, "Maintaining Probablisistic Consistency for Frequently Offline Devices in Mobile Ad Hoc Networks," in *Proc. IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2009, pp. 215–222.
[13] S. Zhu and c. Ravishankar, "Stochastic Consistency and Scalable Pull-Based Caching for Erratic Data Stream Sources," in *Proc. the 30th VLDB Conference*, 2004, pp. 192–203.
[14] *The CSIM User Guides*, http://www.mesquite.com/userguidespage.htm.
[15] S. Lim, W. Lee, G. Cao, and C. R. Das, "A Novel Caching Scheme for Improving Internet-based Mobile Ad Hoc Networks Performance," *Ad Hoc Networks Journal*, vol. 4, no. 2, pp. 225–239, 2006.
[16] S. Lim, C. Yu, and C. R. Das, "Cooperative Cache Invalidation Strategies for Internet-based Vehicular Ad Hoc Networks," in *Proc. The 18th Int'l Conf. on Computer Communications and Networks (ICCCN)*, 2009, pp. 1–6.