

Maintaining Probabilistic Consistency for Frequently Offline Devices in Mobile Ad Hoc Networks

Wenzhong Li^{1,2}, Edward Chan², Daoxu Chen¹ and Sanglu Lu¹

¹State Key Laboratory for Novel Software Technology, Nanjing University

²Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong

Email: lwz@dislab.nju.edu.cn

Abstract

Maintaining cache consistency in mobile environment is an important issue which is extensively studied in the last decade. In mobile ad hoc networks (MANETs), a large number of nonstationary mobile terminals connect with each other through multi-hop unreliable communication channels, coupling with the fact that disconnections from the network are very frequent. Most existing cache consistency strategies assume reliable communication between mobile terminals, which cannot handle frequently offline devices adequately. In this paper, we introduce the probabilistic cache consistency model for applications not requiring strong consistency. Based on this model, a probability consistency strategy (ProP) for frequently offline devices in MANETs is studied. ProP is a randomized pull-based Strategy. It is demonstrated to guarantee cache consistency with a high probability. A theoretical model is developed to investigate the performance of the proposed cache consistency strategies, and design guidelines are provided for ProP to choose proper system parameters to achieve probabilistic cache consistency.

Keywords: Probabilistic cache consistency, Offline device, Mobile ad hoc network

1. Introduction

Data caching is an efficient way to reduce query delay, save bandwidth, and improve overall system performance, particularly in a mobile system. Many applications have to deal with the challenges of efficient access of rapidly changing data objects such as news, stock prices, sensor data, and traffic information. However, the wireless mobile computing environment is characterized by constraints in bandwidth and battery power, as well as user mobility and unreliable wireless links leading to frequent disconnections. Thus maintaining cache consistency in such environment is a challenging problem.

In MANETs, mobile terminals connect with each other and form a self-configuring network with arbitrary topology. There are several characteristics in MANETs. First, mobile devices are frequently disconnected due to mobility or the

need to conserve power. Second, devices employ multi-hop communication through unreliable links, which may cause long communication delay. Third, broadcast in MANETs is costly, thus traditional cache consistent schemes [1], [2] based on broadcasting invalidation messages is not suitable. It is clear that there are many new challenges in a MANET, and existing approaches [3], [4], [5] assume reliable communication between mobile terminals and it is not clear that they can adequately handle frequent disconnections.

In this paper, we propose a probability consistency model for MANETs. To many applications, such as news, sensor data and traffic information, occasionally inconsistent data is acceptable. This means that maintaining strict consistency between cache and source is expensive and unnecessary. Probabilistic consistency allows inconsistent cache items to be used, so even an offline device can use its cache data to answer a query. However, the cache consistency scheme should guarantee that the answer to a query is consistent with the data source with a high probability. Our purpose is to develop a scheme to support frequently offline devices in MANETs, aiming at reducing the latency of information access and achieving probabilistic cache consistency with minimal overhead.

2. Related Work

In [1], Barbara and Imielinski proposed three cache invalidation schemes, namely, *Broadcasting Timestamps (TS)*, *Amnesic Terminals (AT)*, and *Signature (SIG)*. In these approaches, the server periodically broadcasts an *invalidation report (IR)* in which the changed data items are indicated. The advantage of IR-based solutions is that it need not maintain any state information of the mobile clients and can easily scale to any number of clients who listen to the IR. But it suffers from long query latency; in addition it does not handle long sleep time of client well.

A UIR-based approach is presented by Cao [6] to handle the problem of long query latency, in which a small fraction of the essential information related to cache invalidation is replicated several times within an IR interval.

Many solutions have been proposed to handle the long sleep-wakeup patterns [7], [2], [8], [9], [10]. In the *bit-*

sequence (BS) algorithm [2], the invalidation report consists of a set of binary bit sequences with an associated set of timestamps. However, the size of bit sequence broadcast is large and leads to poor bandwidth utilization. Hu et. al. proposed an adaptive invalidation strategies which combine the TS and BS scheme [8]. The server adaptively broadcast TS or BS invalidation reports according to the update and query patterns and client disconnection time. Tan et al. [9] studied the influence of performance by applying the selective tuning scheme in some known cache invalidation algorithms. Asynchronous and stateful (AS) invalidation strategy is proposed by Kahol and Khurana to support roaming of mobile devices [10]. Invalidation message is buffered at a home location cache of the mobile switch station while a mobile terminal is disconnected from the network, and is redelivered to the device when it gets reconnected. Adaptive *time-to-live (TTL)* strategies are proposed in [11], [12], [13] to support weak consistency and mutual consistency. Location dependent and energy efficient cache invalidation strategies can be found in [14], [15].

Maintaining temporal consistency of frequently changing data is studied in [16], [17], [18]. In [16], pure push is used to disseminate updates through a tree of cooperating repositories. In [17], various schemes are discussed for clients to calculate the time to refresh cache copies. Yu et al. [18] study bounding numerical errors among replicated servers, where every server can store and accept updates. However, in their approaches, each server has to keep state information of other servers. A mixed consistency model combining strong and weak consistency was proposed in [19] to meet different requirements of shared data for heterogeneous users.

The concept of probabilistic consistency is introduced in [20], which guarantees the value returned by the system is temporally consistent with the newest copy with a probability of p . A stochastic consistency approach is proposed in [21]. Stochastic consistency guarantees that cache-source deviation remains within user-specified error tolerance with a certain probability. A Brownian motion model is used to capture the behavior of data changes and to predict when caches should initiate pulls to refresh cached copies.

In MANETs, mobile terminals employ multi-hop communication, thus cache consistency strategies used in single-hop wireless networks are not suitable. Lim et al. [3] proposed a *global positioning system based connectivity estimation (GPSCE)* scheme. However, the high cost of GPS devices restricts the popularization of these strategies. In [5], the combination of push and pull strategies is used to achieve flexible cache consistency for different user requirements. However, it assumed delayed-update in the data source, which may be unacceptable to many applications. However, most existing approaches based on direct message exchange between data source and caches, which may not be realistic when the mobile devices are frequently offline or disconnected from the data source.

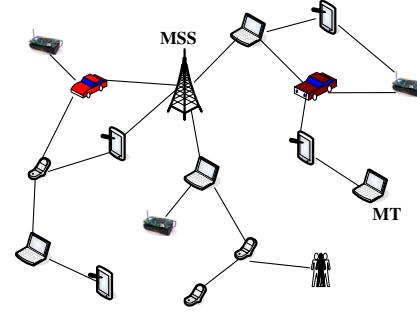


Figure 1. A mobile ad hoc network

3. The Probabilistic Consistency Model

In this section, we present the MANET system model and a formal definition of probabilistic consistency.

3.1. MANET System Model

In mobile ad-hoc network (MANET), mobile terminals (MTs) are connected with each other and form a self-configuring network with arbitrary topology. Figure 1 illustrates our system model of a MANET. A Mobile Support Station (MSS) is a fixed host which connects the MTs with an external wired network (i.e. servers in the Internet). The MSS can be viewed as a data source (or data server) which contains a set of data items and can handle all the requests from the MTs. An MT can move freely and generate queries from anywhere in the covered area. Among the MTs, some can directly communicate with the MSS. If an MT is located out of the communication range of the MSS, information access has to go through multi-hop communication. Each MT can serve as a router to forward message for other MTs.

3.2. Probabilistic Cache Consistency

In our approach, MTs cache and serve data objects using a probabilistic consistency model. A query is satisfied if the data object returned by the MT is consistent with the data source with probability at least Γ , where Γ ($0 \leq \Gamma \leq 1$) is a system pre-defined consistency level. Let S_t^o denote the version of the object o at the source and C_t^o denote the version of the object o returned by the MT, at all time t . The cache is probabilistic consistent of Γ if

$$\forall t, Pr(S_t^o = C_t^o) \geq \Gamma. \quad (1)$$

Practically, we can define probabilistic consistency in a statistical manner. Consider a sequence of visits generated by a MT during a period of time t . Assume the number of visits is $N_v(t)$, and the number of consistent data items returned by the MT is $N_c(t)$. We say the MT satisfy a probabilistic consistency of Γ if

$$\lim_{t \rightarrow \infty} \frac{N_c(t)}{N_v(t)} \geq \Gamma. \quad (2)$$

4. Cache Consistency Strategies

In this section, we propose several pull-based strategies to maintain probabilistic consistency.

4.1. The Pull Source (PS) Scheme

PS is a poll-every-time approach. It is very simple and widely used in maintaining strong consistency. In this approach, when a query is generated, the MT send a $\langle VERIFY, ID \rangle$ (ID indicates the id of the required data item) message to the data source. On receiving a verify message, the data source will send back a $\langle REPLY, T_S(ID) \rangle$ message, where $T_S(ID)$ indicates the timestamp the data object is last updated. When the MT receives a reply message, it compares $T_S(ID)$ with the timestamp of the cache item to decide its validity.

The PS strategy is simple and easy to implement. It maintains strong consistency between source and cache copies. However, it is expensive with high communication overhead. And it also causes long query delay since it needs to exchange message through multi-hop communications. Moreover, it cannot handle disconnected devices.

4.2. The Pull Neighbor (PN) Scheme

Unlike the PS scheme, the PN scheme doesn't communicate with the data source directly. An MT only contacts its one hop neighbors to verify consistency. When a query is generated, the MT broadcast a $\langle VERIFY, ID \rangle$ message to its neighbors. When a neighbor receive a verify message, it sends back a $\langle REPLY, T_N(ID) \rangle$ message, where $T_N(ID)$ indicates the timestamp an update is last seem by the node. The MT collects the reply messages and compares the timestamp of the cache item with its neighbors'. If none of the timestamps is more current than that in the local cache, the MT will use the cached data to serve the query.

The PN scheme has several advantages. Since it only needs one hop communication with its neighbors, it has low communication overhead, and achieve quick response. It also can handle some kind of disconnection like group disconnection devices. However, it only maintain cache consistency with neighbor nodes. As our analytical result shows in a later section, it has a very high stale cache hit rate, which makes the guaranteeing probabilistic consistency extremely difficult.

4.3. The Probabilistic Pull (ProP) scheme

The ProP scheme is a little more complicated than the PS and PN schemes. The basic idea is that in a probabilistic consistency model it is not necessary to pull source every time. For each query, the MT may pull the source with a probability ρ ; otherwise it will contact its neighbors to verify

consistency. If it is lucky and one of the neighbors has just updated its data then the MT can invalidate the cache item using a one hop communication. The ProP scheme can be viewed as a randomized strategy combining the PS and PN schemes: with probability ρ it performs pull source; with probability $1 - \rho$ it performs pull neighbor. It also support probabilistic consistency for frequently disconnected devices.

The details of ProP strategy is as follows.

(1) If the MT is disconnected from the data source: it estimates the probability that the cached data item has not been updated according to update history and its disconnection time, and then decides whether to use the cached item to serve the query.

(3) If the MT is connected to the data source, it performs a randomized pull source strategy:

(a) The MT sends a $\langle VERIFY_{source}, ID \rangle$ message to the source with probability ρ , and broadcast a $\langle VERIFY_{neighbor}, ID \rangle$ message with probability $1 - \rho$;

(b) If the source receives a verify message, it sends back a $\langle REPLY_{source}, T_S(ID) \rangle$ message;

(c) If a neighbor receives a verify message, it sends back a $\langle REPLY_{neighbor}, T_N(ID) \rangle$ message;

(d) The MT compares the timestamp of the cache item with all the timestamps received in the reply messages, if no update is detected, it will use the cached item to serve the query.

ProP is a flexible solution which combines the advantages of PS and PN. By adaptively adjusting the pull probability ρ , ProP can guarantee the probabilistic consistency, yet with low message cost. In Section 5, we will provide theoretical analysis to the performance of the proposed strategies.

5. Analysis of the Proposed Schemes

5.1. Assumptions

We list the assumptions used in our analysis below:

- The data source has N data items.
- The set of cache nodes in the MANET is denoted by G .
- Updates to each data item follows a Poisson process with rate μ .
- In each MT, access to a data item follows a Poisson process with rate λ .
- Only frequently access objects are cached. That is, the access rate to a cache item should be larger than its update rate ($\frac{\lambda}{\mu} > 1$). If $\frac{\lambda}{\mu} < 1$, the data is likely updated before the next access, which is unnecessary to be cached.
- We do not consider the influence of any cache replacement strategy used by the MTs. That is, we assume the cache space of an MT is sufficiently large.

5.2. Performance Analysis

When a data item D is updated at the data source S , assuming there are a set of cache copies in the MANET, we now study how the update is propagated to the other nodes in the network using the proposed cache consistency scheme. Assume R is a cache node. An *update propagation path* is the shortest path from S to R . Let $PA(S, R)$ denote the set of cache nodes in the update propagation path from S to R . If $|PA(S, R)| = k$, we call R a k -hop cache node. The *update propagation radius* Ψ is defined as $\Psi = \max\{|PA(S, R)|, \forall R \in G\}$, which is the length of the longest update propagation path in the MANET.

According to our assumptions, each node generates queries to the item D at a rate of λ . Whenever a query is generated, the node will decide whether use the cached item to serve the query or fetch a new copy from the data source. A cache node is updated if it holds the newest version of D . Let U be the set of cache nodes which is updated. We say an update is k -hop propagated to R if and only if $|PA(S, R)| = k$, and $\forall x \in PA(S, R), x \in U$.

If a requested item scores a hit in the local cache, and the cache item is valid, we call it an *effective cache hit*; otherwise if the request is served by an out-of-date copy, we call it a *stale cache hit*. If an update is detected, the MT has to drop the cache item and fetch a new copy from the source, we call it a *cache miss*.

In our analysis, we assume the interval between two consecutive updates is long enough for an update to be propagated to Ψ hops. After the update propagation, a visit will always be an effective cache hit. We only analyze the average performance during the update propagation phrase, which can be viewed as a lower bound for the performance of the proposed schemes.

The following theorems show the performance of the PS, PN and ProP strategies.

Theorem 1: In the PS scheme, the following statements hold.

(I) When a new update is k -hop propagated to R , the expected number of visits is

$$\delta_{ps} = kH_k, \quad (3)$$

where H_k is the k th Harmonic number [22].

(II) The effective cache hit ratio of PS scheme is

$$h_{ps} = 1 - \frac{1}{H_k}. \quad (4)$$

(III) The stale cache hit ratio of PS scheme is

$$\eta_{ps} = 0. \quad (5)$$

Proof: Let X be a random variable defined to be the number of visits when a new update is k -hop propagated to R . Let C_1, C_2, \dots, C_X denote the sequence of visits, where $C_i \in \{1, 2, \dots, k\}$ denotes the node generated the

request. Call the i th visit C_i a *success* if the node receives the update item for the first time. Clearly C_1 and C_X are always successes.

We divide the sequence into *epochs*, where epoch i begins with the (i) th successful visit and ends with the $(i+1)$ th successful visit. Define the random variable X_i , for $0 \leq i \leq k-1$, to be the number of visits in the i th epoch, so that

$$X = \sum_{i=0}^{k-1} X_i.$$

Proof of (I):

Let p_i denote the probability that a visit is a success in the i th epoch.

$$\begin{aligned} p_i &= Pr\{\text{the visit is from the remaining } k-i \text{ nodes}\} \\ &= \frac{k-i}{k}. \end{aligned}$$

The random variable X_i is geometrically distributed with parameter p_i . Thus the expected value of X_i is $\frac{1}{p_i}$. By linearity of expectation,

$$\begin{aligned} \delta_{ps} &= E[X] = E\left[\sum_{i=0}^{k-1} X_i\right] = \sum_{i=0}^{k-1} E[X_i] \\ &= \sum_{i=0}^{k-1} \frac{k}{k-i} = k \sum_{i=1}^k \frac{1}{i} = kH_k. \end{aligned}$$

By [22] the k th Harmonic number H_k is asymptotically equal to $\ln k + 0.577$.

Proof of (II):

Consider the probability of an effective cache hit in the above epoches. In epoch i , there are i cache copies which have been updated. If a query is generated by one of the i cache nodes, it is an effective cache hit; otherwise it is a stale cache hit or a miss. The probability that a query is an effective cache hit is $\frac{i}{k}$. So the expected number of effective cache hit is

$$\begin{aligned} \delta_{pse} &= E\left[\sum_{i=0}^{k-1} \frac{i}{k} X_i\right] = \sum_{i=0}^{k-1} \frac{i}{k} E[X_i] = \sum_{i=0}^{k-1} \frac{i}{k} \frac{k}{k-i} \\ &= \sum_{i=0}^{k-1} \frac{i}{k-i} = \sum_{i=1}^k \frac{k-i}{i} = kH_k - k. \end{aligned}$$

Hence the average effective cache hit ratio is

$$h_{ps} = \frac{\delta_{pse}}{\delta_{ps}} = 1 - \frac{1}{H_k}.$$

Proof of (III):

This conclusion is straight forward, since the PS strategy maintains strong consistency. The MT verifies its cache item with the data source for each visit, thus the stale cache hit ratio is 0. \square

Theorem 2: In the PN scheme, the following statements hold.

(I) When a new update is k-hop propagated to R , the expected number of visits is

$$\delta_{pn} = k^2. \quad (6)$$

(II) The effective cache hit ratio of PN scheme is

$$h_{pn} = \frac{1}{2} - \frac{1}{2k}. \quad (7)$$

(III) The stale cache hit ratio of PN scheme is

$$\eta_{pn} = \frac{1}{2} - \frac{1}{2k}. \quad (8)$$

The detailed proof of this theorem is omitted due to page limit.

Theorem 3: For each item in an MT, the probability that exactly m visits to the item between two updates is

$$\frac{\mu}{\mu + \lambda} \left(\frac{\lambda}{\mu + \lambda} \right)^m. \quad (9)$$

The conclusion can be derived by the memoryless property of exponential distribution. Due to page limit, detailed proof of this theorem is omitted.

Theorem 4: In the ProP scheme, if each MT pulls the data source with a probability of ρ , the following statements hold.

(I) When a new update is k-hop propagated to R , the expected number of visits is

$$\delta_{prop} = \frac{k}{\rho} H'_k, \quad (10)$$

where $H'_k = \sum_{i=1}^k \frac{1}{\rho - 1 + i}$ and $H'_k \in [H_k, k]$.

(II) The effective cache hit ratio of ProP scheme satisfies

$$h_{prop} \geq \frac{H'_k - \rho}{H'_k} - \frac{\mu}{\mu + \lambda} \frac{1 - \rho}{\rho} \ln(1 + \rho \frac{\lambda}{\mu}) \quad (11)$$

(III) The expectation of stale hit ratio satisfies

$$\eta_{prop} \leq \frac{\mu}{\mu + \lambda} \frac{1 - \rho}{\rho} \ln(1 + \rho \frac{\lambda}{\mu}). \quad (12)$$

The theorem can be proved similar to Theorem 1. The detailed proof of this theorem is omitted due to page limit.

Theorem 4 shows that the stale hit ratio η_{prop} is related to two parameters: the access/update ratio ($\frac{\lambda}{\mu}$) and the pull probability ρ . the $\frac{\lambda}{\mu}$ ratio is decided by data access and update pattern, and ρ is a tunable parameter. According to the theorem, by adjusting ρ , we can restrict the stale cache hit ratio to a small range, thus achieve high probability of cache consistency. In the next section, we will introduce how to choose proper system parameters for ProP to provide probability consistency in MANETs.

6. Guaranteeing Probabilistic Consistency

In this section, we show how the ProP strategy guarantee probability consistency of Γ .

6.1. Maintaining Probabilistic Consistency for Disconnected Devices

If an MT is disconnected from the network, it estimates its cache consistency probability based on local information. Assume the object item D is last updated at time t_u , and the current time is t_c . Let $\Delta t = t_c - t_u$. The following theorem estimates the probabilistic cache consistency level.

Theorem 5: If $\Delta t \leq -\frac{\ln \Gamma}{\mu}$, the cache item satisfies a probabilistic consistency of Γ .

Proof: Consider the probability that D is consistent with the data source:

$$\begin{aligned} & Pr\{D \text{ is consistent with } S\} \\ &= Pr\{\text{no update to } D \text{ in } \Delta t\} \\ &= 1 - \int_0^{\Delta t} \mu e^{-\mu t} dt = e^{-\mu \Delta t}. \end{aligned}$$

Let $e^{-\mu \Delta t} \geq \Gamma$, we have $\Delta t \leq -\frac{\ln \Gamma}{\mu}$. The Theorem is proved. \square

Theorem 5 shows that if the MT's disconnection time is less than $-\frac{\ln \Gamma}{\mu}$ since the last update to D , it can use the cached item to serve the query, with probabilistic consistency guarantee Γ .

6.2. Determining the pull probability

In the ProP strategy, if the pull probability ρ is close to 0, its performance is close to the PN strategy; if ρ is close to 1, the performance is close to PS (achieving near strong consistency). We now study how to determine ρ to guarantee a certain level of probability consistency Γ .

Theorem 6: In the ProP scheme, if the pull probability ρ satisfies

$$\rho \geq \left(1 + \frac{1 + \frac{\lambda}{\mu}}{\ln(1 + \frac{\lambda}{\mu})} (1 - \Gamma) \right)^{-1}, \quad (13)$$

the MT satisfies a probabilistic consistency of Γ .

Proof:

$$\begin{aligned} & Pr\{a \text{ cache item is consistent with the source}\} \geq \Gamma \\ & \iff Pr\{\text{the cache item is a stale hit}\} \leq 1 - \Gamma. \end{aligned}$$

According to Theorem 4, the stale cache hit ratio

$$\begin{aligned} h_{prob} &\leq \frac{\mu}{\mu + \lambda} \frac{1 - \rho}{\rho} \ln(1 + \rho \frac{\lambda}{\mu}) \\ &\leq \frac{\mu}{\mu + \lambda} \frac{1 - \rho}{\rho} \ln(1 + \frac{\lambda}{\mu}) \end{aligned}$$

Let

$$\frac{\mu}{\mu + \lambda} \frac{1 - \rho}{\rho} \ln(1 + \frac{\lambda}{\mu}) \leq 1 - \Gamma.$$

Solving this inequality yields

$$\rho \geq \left(1 + \frac{1 + \frac{\lambda}{\mu}}{\ln(1 + \frac{\lambda}{\mu})}(1 - \Gamma)\right)^{-1}.$$

□

Theorem 6 shows that if the pull probability ρ satisfies the condition in equation (13), the ProP scheme can guarantee a probabilistic consistency of Γ .

7. Performance Evaluation

In this section, extensive simulations are conducted to evaluate the performance of the proposed cache consistency schemes.

7.1. The Simulation Model

We simulate our proposed cache consistency scheme in a MANET environment where a set of mobile terminals move in a rectangular area with size of $area_width * area_height$. A data source (MSS) is located at the upper left corner of the rectangle. The data source contains N data items. The data items can only be updated by the server whereas the queries are made on the client side. We assume updates to each data item is a Poisson process with rate μ .

In the simulation area, there is a set of mobile terminals randomly deployed to form an ad hoc network. Random Waypoint movement model is used to simulate the moving pattern of the mobile clients [23].

Each mobile terminal generates a number of queries periodically. The query to each data item follows a Poisson process with rate λ . Each node and each link in the network has a probability p_d of being disconnected from the network. Disconnection time of a node (link) follows an exponential distribution with mean interval L .

7.2. Metrics

Our cache consistency schemes are evaluated using the following metrics: (i) Fidelity. *Fidelity* is defined to be the degree to which a cache coherency mechanism can provide consistency guarantees to users [12]. It can be measured as: $f = \frac{\text{Number of consistent access}}{\text{Number of queries}}$. (ii) Effective cache hit ratio, which is presented in section 5.2. (iii) Delay. *Delay* is defined to be the time it takes to answer a query.

7.3. Performance Evaluation

In the following, we analyze the performance of PS, PN and ProP strategies under different system parameter settings.

7.3.1. Impact of the access/update ratio. Figure 2 shows the fidelity, effective cache hit ratio and access delay when $\frac{\lambda}{\mu}$ is varied from 2 to 10. It can be seen that the fidelity of PS is always 1, since it maintains strong consistency. The fidelity of ProP is quite close to PS, in most cases larger than 0.95. The fidelity of PN increases from 0.52 to 0.78 as $\frac{\lambda}{\mu}$ increases. The reason is that when $\frac{\lambda}{\mu}$ becomes larger, there are more visits between two updates, which will increase the chance of accessing consistent data. However, PN has the lowest fidelity of the three, and cannot guarantee probability consistency. ProP maintains high fidelity larger than 0.95 all the time.

Figure 2(b) shows the effective hit ratio of the three strategies increases with $\frac{\lambda}{\mu}$. The reason has been explained above: a larger $\frac{\lambda}{\mu}$ attracts more consistent accesses. The performance of ProP and PS are very close. However, the effective hit rate of PN is much lower than the other strategies, which is only half of the PS when $\frac{\lambda}{\mu}$ ratio is small.

Access delay is compared in Figure 2(c). PS has the longest delay, since each visit travels a long path to the data source, and a disconnected device need to block all its queries until the next reconnection. Access delay of PN is almost constant, because it only employs one-hop communication with its neighbor. Performance of ProP is between PS and PN. Its delay is about 20-30% lower than PS.

7.3.2. Impact of network density. Figure 3 presents the experimental results when the number of MTs varies from 100 to 300. It can be seen that PS and ProP are insensitive to network density, both maintain high fidelity (close to 1) and high effective hit ratio (close to 0.7). For PN, as network density increases, there are more neighbors each node, more information is obtained and an update is detected more quickly, thus the fidelity and effective hit ratio both increase. Again, PS has the longest access delay which is about 10 times larger than PN.

7.3.3. Impact of disconnection probability. Mobile terminals' disconnection probability also influence system performance, as shown in Figure 4. The fidelity and effective hit ratio of PS and ProP are barely affected by device disconnections. When device disconnection probability increase from 0.05 to 0.25, ProP remains a high probability consistency larger than 0.95. In PN, its fidelity and hit ratio decreases with increasing device disconnection. Disconnection probability also has a significant impact on access delay. As shown in Figure 4(c), the access delay of PS and ProP both increase when more devices are disconnected, since more queries are delayed until reconnection. ProP allows a small proportion of inconsistent cache access (smaller than 0.05), but achieve a much lower access delay (about 20% lower than PS). If we relax the probability consistency requirement, the overhead

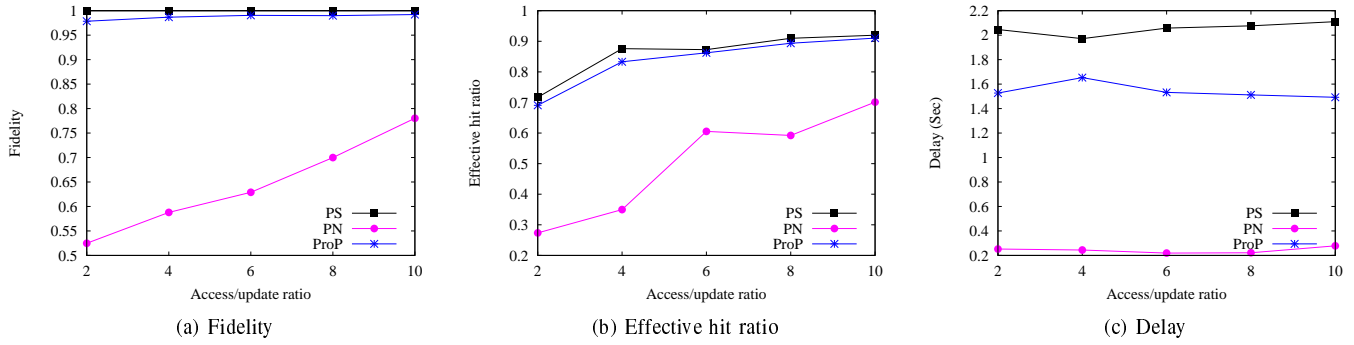


Figure 2. Performance under various access/update ratio.

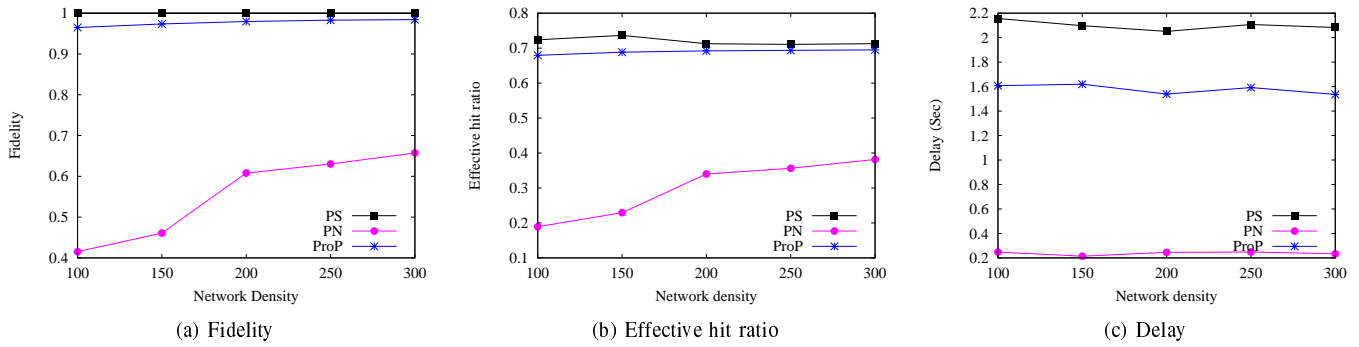


Figure 3. Performance under various network density.

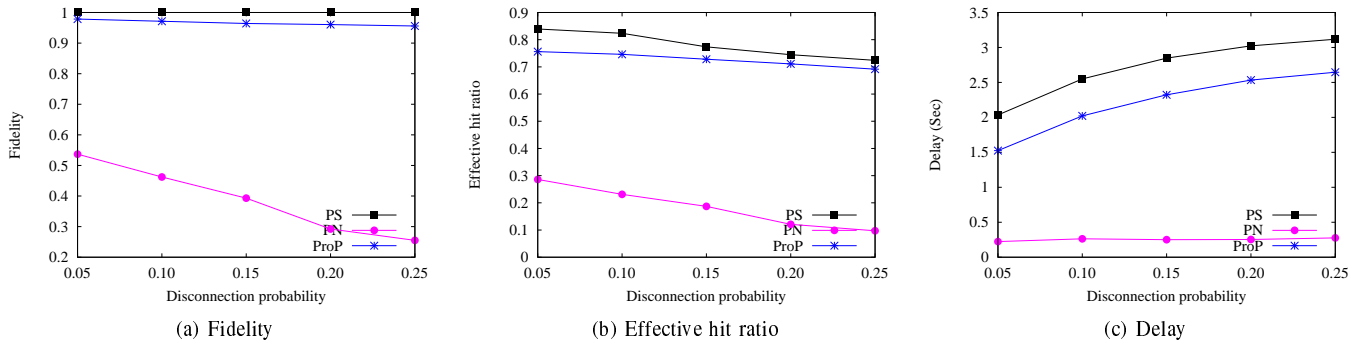


Figure 4. Performance under various disconnection probability.

will be further decreased. Access delay of PN stays a small constant since it always employ one-hop communication with neighbors.

8. Conclusion

In this paper, we introduce a probabilistic cache consistency model to guarantee cache consistency with a high probability. Based on this model, we study three cache consistency strategies: pull source (PS), pull neighbor (PN) and probabilistic pull (ProP). PS maintains strong con-

sistency between data source and cache copies. PN only maintains cache consistency with neighbor nodes. ProP pulls data source with some probability, which is a randomized strategy combining the advantages of PS and PN. A theoretical model is developed to analyze the performance of the proposed schemes. We demonstrate that the ProP scheme achieves probabilistic cache consistency, and we present design guidelines for ProP to choose proper system parameters to guarantee probabilistic consistency. Extensive simulations show the efficiency of our strategies.

Acknowledgment

The work described in this paper was supported by the National Natural Science Foundation of China (No. 60803111, 60573106, 90718031, 60721002), as well as the grant from City University of Hong Kong (Project NO. 7002115), the 863 Program of China (No. 2006AA01Z199), and the 973 Program of China (No. 2009CB320705).

References

- [1] D. Barbara; and T. Imielinski, "Sleepers and workaholics: caching strategies in mobile environments (extended version)," *The VLDB Journal*, vol. 4, no. 4, pp. 567–602, 1995.
- [2] J. Jing, A. Elmagarmid, A. S. Helal, and R. Alonso, "Bit-sequences: an adaptive cache invalidation method in mobile client/server environments," *Mob. Netw. Appl.*, vol. 2, no. 2, pp. 115–127, 1997.
- [3] S. Lim, W.-C. Lee, G. Cao, and D. C.R., "Performance comparison of cache invalidation strategies for internet-based mobile ad hoc networks," in *Mass'04: IEEE International Conference on Mobile Ad-hoc and Sensor Systems*, October 2004, pp. 104–113.
- [4] J. Cao, Y. Zhang, G. Cao, and L. Xie, "Data consistency for cooperative caching in mobile environments," *Computer*, vol. 40, no. 4, pp. 60–66, 2007.
- [5] Y. Huang, J. Cao, Z. Wang, B. Jin, and Y. Feng, "Achieving flexible cache consistency for pervasive internet access," in *PerCom '07: Proceedings of the 5th Annual IEEE International Conference on Pervasive Computing and Communications*, 2007, pp. 239–250.
- [6] G. Cao, "A scalable low-latency cache invalidation strategy for mobile environments," in *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*. New York, NY, USA: ACM Press, 2000, pp. 200–209.
- [7] J. Cai and K.-L. Tan, "Energy-efficient selective cache invalidation," *Wirel. Netw.*, vol. 5, no. 6, pp. 489–502, 1999.
- [8] Q. Hu and D. L. Lee, "Cache algorithms based on adaptive invalidation reports for mobile environments," *Cluster Computing*, vol. 1, no. 1, pp. 39–50, 1998.
- [9] K.-L. Tan, J. Cai, and B. C. Ooi, "An evaluation of cache invalidation strategies in wireless environments," *IEEE Trans. Parallel Distrib. Syst.*, vol. 12, no. 8, pp. 789–807, 2001.
- [10] A. Kahol, S. Khurana, S. K. Gupta, and P. K. Srimani, "A strategy to manage cache consistency in a disconnected distributed environment," *IEEE Trans. Parallel Distrib. Syst.*, vol. 12, no. 7, pp. 686–700, 2001.
- [11] Z. Wang, S. K. Das, and H. Che, "A scalable asynchronous cache consistency scheme (saccs) for mobile environments," *IEEE Trans. Parallel Distrib. Syst.*, vol. 15, no. 11, pp. 983–995, 2004, senior Member-Mohan Kumar.
- [12] B. Urgaonkar, A. G. Ninan, M. S. Raunak, P. Shenoy, and K. Ramamritham, "Maintaining mutual consistency for cached web objects," in *ICDCS '01: Proceedings of the The 21st International Conference on Distributed Computing Systems*. Washington, DC, USA: IEEE Computer Society, 2001, p. 371.
- [13] M. Bhide, P. Deolasee, A. Katkar, A. Panchbudhe, K. Ramamritham, and P. Shenoy, "Adaptive push-pull: Disseminating dynamic web data," *IEEE Trans. Comput.*, vol. 51, no. 6, pp. 652–668, 2002.
- [14] J. Xu, X. Tang, and D. L. Lee, "Performance analysis of location-dependent cache invalidation schemes for mobile environments," *IEEE Transactions on Knowledge and Data Engineering*, vol. 15, no. 2, pp. 474–488, 2003.
- [15] J. C.-H. Yuen, E. Chan, K.-Y. Lam, and H. W. Leung, "Cache invalidation scheme for mobile computing systems with real-time data," *SIGMOD Rec.*, vol. 29, no. 4, pp. 34–39, 2000.
- [16] S. Shah, K. Ramamritham, and P. Shenoy, "Maintaining coherency of dynamic data in cooperating repositories," in *VLDB '02: Proceedings of the 28th international conference on Very Large Data Bases*. VLDB Endowment, 2002, pp. 526–537.
- [17] R. Srinivasan, C. Liang, and K. Ramamritham, "Maintaining temporal coherency of virtual data warehouses," in *RTSS '98: Proceedings of the IEEE Real-Time Systems Symposium*. Washington, DC, USA: IEEE Computer Society, 1998, p. 60.
- [18] H. Yu and A. Vahdat, "Efficient numerical error bounding for replicated network services," in *VLDB '00: Proceedings of the 26th International Conference on Very Large Data Bases*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, pp. 123–133.
- [19] Z. Zhan, M. Ahamad, and M. Raynal, "Mixed consistency model: Meeting data sharing needs of heterogeneous users," in *ICDCS '05: Proceedings of the 25th IEEE International Conference on Distributed Computing Systems*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 209–218.
- [20] H. Zou, N. Soparkar, and F. Jahanian, "Probabilistic data consistency for wide-area applications," in *ICDE '00: Proceedings of the 16th International Conference on Data Engineering*. Washington, DC, USA: IEEE Computer Society, 2000, p. 85.
- [21] S. Zhu and C. V. Ravishanker, "Stochastic consistency, and scalable pull-based caching for erratic data stream sources," in *VLDB '04: Proceedings of the Thirtieth international conference on Very large data bases*. VLDB Endowment, 2004, pp. 192–203.
- [22] S. Baase and A. V. Gelder, *Computer Algorithms: Introduction to Design and Analysis*, 3rd ed. Addison Wesley, 1999.
- [23] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva, "A performance comparison of multi-hop wireless ad hoc network routing protocols," in *Mobile Computing and Networking*, 1998, pp. 85–97.