Analysis of Windowing and Peering Schemes for Cache Coherency in Mobile Devices

Sandhya Narayan, Julee Pandya, Prasant Mohapatra, Dipak Ghosal *

Department of Computer Science
University of California, Davis, CA 95616, USA,
{ghosal}@cs.ucdavis.edu

Abstract. A major factor in determining the effectiveness of caching in wireless networks is the cache coherency scheme which maintains consistency between mobile stations (MSs) and the server. Since the wireless channel is inherently a broadcast medium, an appropriate cache coherency scheme is one in which the server broadcasts cache invalidation reports (IRs) that contain data update information. However, in a wireless environment, since MSs may connect to the network only intermittently (e.g., to save power), IRs may be missed. This would cause the MS's cache to become invalid and in turn the cache would have to be purged resulting in higher query-delay and lower throughput. One approach to improving the cache coherency for mobile devices is the Time Stamp (TS) method [2] which uses a *windowing* scheme. In this scheme, the IR in a particular interval contains the IRs for a number of previous intervals determined by the window size. Another orthogonal approach to improving cache coherency is the *peering* scheme [10] where an MS can query neighboring peers to retrieve IRs that it may have missed while it was disconnected. In this paper, we present a unified mathematical model based on Discrete Markov Models (DMMs) to study the effectiveness of these orthogonal schemes both individually as well as their relative importance when they are implemented together. The results show that both schemes are comparable for the most part. Since they are orthogonal, they can be combined in ways that is tailored for the particular environment to achieve significant improvement in performance.

1 Introduction

For wireless networks and applications, caching data at the clients reduces unnecessary network accesses and use of wireless channels, saving energy as well as wireless bandwidth [9, 13]. Benefits of caching can be maximized through robust and effective placement of data objects, maintaining coherency, replacement algorithms of cached objects, and various timing issues. This paper addresses the

* S. Narayan, P. Mohapatra, and D. Ghosal are with the Department of Computer Science, University of California, Davis. J. Pandya is with the MIT Lincoln Labs.

problem of maintaining cache coherency specifically in the context of intermittently disconnected devices.

The underlying cache invalidation method studied in this paper is based on periodic server broadcasts of Invalidation Reports (IRs) to the clients. These reports indicate which data items have been modified during the last interval. Clients use the IRs to update their caches. One of the key issues of IR based techniques arises due to the intermittent disconnectedness of the clients. Since connecting to the network uses power and because power is a limited resource, wireless devices may connect to the network only when necessary. During periods when the device is disconnected, IRs will will be missed. As result, upon reconnection, the client will not be able to guarantee that the cache data is valid and will have to purge its cache. This method, known as the TS scheme, was first discussed in the well known paper [2] in which they also proposed an windowing scheme as an enhancement to the basic approach. Under this scheme, each IR in a particular interval contains the IR for a number of previous intervals defined by the window size. The paper provided analysis based on asymptotic bounds and studied the impact of the window size on the cache hit rate and throughput.

An orthogonal approach for improving cache coherency is to exploit peering among MSs [10]. In this approach, called Peer Enhanced Cache (PEC), MSs utilize the ad-hoc mode of operation, which is now available with most wireless interfaces. They store IRs on behalf of other MSs which are disconnected. If an MS misses IRs due to disconnection, it can attempt to retrieve them by querying all the peers within its direct transmission range. Note that PEC is an orthogonal scheme and can be implemented in conjunction with other cache invalidation methods, in particular with the TS scheme.

In this paper we present a unified mathematical model to study the performance of the windowing and peering schemes. The analysis is based on Discrete Markov Model (DMM) [1]. The model allows us to study the windowing and the peering schemes individually and when they are implemented together. Based on a thorough analysis of the system, the following are the main contributions of this paper:

- While the cache consistency using windowing in TS approach was analyzed in [2], the authors only provided upper and lower bounds for the cache hit rate and throughput. In this paper, we provide an accurate analysis of the TS scheme.
- The peering scheme analyzed in [10] modeled only the basic scheme, i.e., with window size of 1. This model extends the study to the case when both peering and windowing are used.
- The results show that both schemes provide significant improvements to cache coherency, and that their performance is comparable for the most part. However, both windowing and peering consume bandwidth and thereby reduce the gain inthroughput. Since they are orthogonal, they can be combined in ways that is tailored for the particular environment to achieve significant improvement in performance.

The remainder of this paper is organized as follows. Section 2 presents a reference architecture to describe the peering and windowing schemes. Section 3 outlines the model and presents a mathematical analysis of the windowing and peering schemes and when they are implemented together. Section 4 presents the derivation of some metrics such as hit rate and throughput. Section 5 discusses the results and Section 6 provides a brief description of related work. Finally conclusions are drawn in Section 7.

2 Reference Architectures

An application program running on a Mobile Station (MS) uses a local cache for performance, bandwidth savings, and energy efficiency. An application query is handled by the cache if possible, otherwise, it is sent to the base station (BS) which also acts as a server. The Base Station (BS) responds with the data objects for that query which are then cached by the MS. The BS periodically (every L time units) broadcasts an Invalidation Report (IR), which indicates the data items have been modified during the past interval. All connected MSs within the base station coverage area (BSCA) receive the IR. They use this to validate their cache contents and discard data objects that have been invalidated by the IR. During periods when an MS is disconnected, it does not receive IRs. Therefore, if disconnection times is greater than the IR interval, and one or more IRs is missed, when the MS reconnects, the cached data cannot be guaranteed to be valid and will have to be purged.

2.1 TS Scheme

In the TS scheme (also called broadcast time stamp method) [2], the BS broadcasts IRs every L time units as before. However, each IR indicates the items that have been modified during a specified window of time w, where w is some multiple of L. The larger the w, the longer the client can disconnect without having to purge its cache. However, increasing w also increases the size of the IR which implies higher bandwidth for each broadcast. Note that in all IR based schemes, the MS has wait to answer queries until the next IR is broadcast. Queries generated between broadcasts are queued until the next IR is received. This is done so that data items that became invalid during the IR interval are not retrieved from the cache and used to answer the queries. Therefore, if the IR interval can be decreased, the query latency will also decrease.

2.2 Peer Enhanced Caching (PEC)

An orthogonal scheme to improve the IR-based cache coherency mechanism is PEC which exploits peering between MSs [10]. We denote peer coverage area (PCA) to be the coverage area defined by the transmission range of the peer. In order to assist other peers, each MS stores the IRs it receives when connected to the network. When a MS reconnects after being disconnected, it broadcasts a

query in its PCA requesting for the IRs that it has missed. All the MSs that are within the PCA respond with the subset of the requested IRs that they have. The requesting MS then waits until the next IR and based on the IRs received from its peers, it determines if it can validate the cache. If not, the cache is purged. Note that the larger the number of peers, the longer the client can disconnect without having to purge its cache. However, the more the number of peers the higher the bandwidth consumption.

3 A Unified Model

We consider that time is slotted and the slot length is one IR interval denoted by L. The sleep-awake cycle is modeled by a first-order Discrete Markov Model [1] consisting of two states - a sleeping (disconnected) state and 2) an awake (connected) state. When awake, the MS can get disconnected in an interval with probability d. The duration of being in the $awake$ state is negative exponentially distributed with rate $1/d$. We assume that the disconnection time is also negative exponentially distributed with a mean value of T_d. Since L is the length of the IR interval, the probability that a sleeping MS would continue to sleep in the next IR is given $s = e^{-L/T_d}$. Similar to the two papers of interest ([2] and [10]), we assume that the probabilities s and d are equal.

The models for the query and update are same as that used in [2]. Each MS queries a subset of the database with a high locality. This subset is known as the hot spot and each item in the subset is queried at rate λ. Updates to each data item are negative exponentially distributed with mean rate μ updates per second. We consider a random mobility model in which at the end of each IR interval, the MS moves to a random location within the BSCA. Under this model, the number of peers within the PCA will be proportional to the ratio of the areas of the PCA and the BSCA. We will let N denote the average number of peers in the PCA.

In addition to the above parameters, we use the following notation adopted from [2]. The probability of being awake and having no queries in an interval is $q_0 = (1-s)e^{-\lambda L}$. The probability of no queries in an interval is $p_0 = s + q_0$. The probability of no updates during an interval is $u_0 = e^{-\mu L}$.

3.1 Analysis of the Windowing Scheme

Let k denote the window size. As in [2], the goal of the analysis is to find the probability that a tagged MS, denoted by MS_t, has slept for k or more consecutive intervals between two consecutive queries that are i intervals apart. Based on the model, the states of MS_t can be modeled by a DMM shown in Figure1. There are $k+1$ states. State 0 indicates that the MS is awake and it has all the required IRs and that its cache is in a consistent state. State m for $0 < m < k$, indicates that the MS has slept for $m-1$ consecutive intervals and that it can still get all the missed IR data if it wakes up at the next interval. State k indicates that MS_t has missed some IR data as a consequence of sleeping for

k or more consecutive intervals. The cache has become inconsistent as a result and needs to be purged. It should be noted that the system will not remain in this state forever. In fact, it will get back to state 0 as soon as the cache is purged and valid data is retrieved from the server. The diagram also shows the state-transition probabilities.

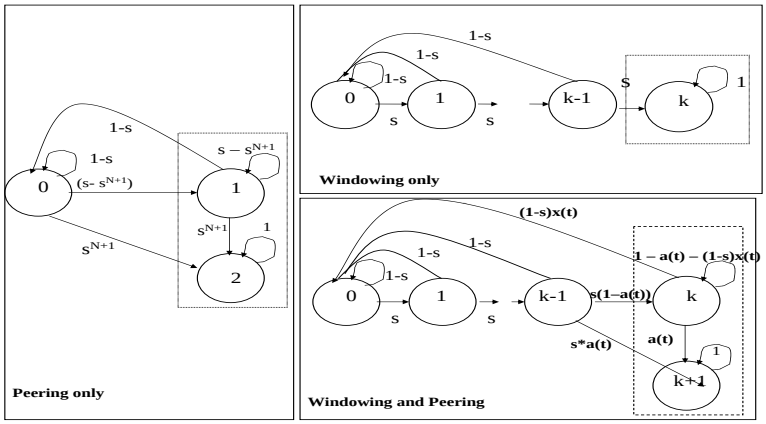

Fig. 1. The Discrete Markov Model for windowing only, peering only, and the combined schemes.

Let $P_m(t)$ be the probability that the DMM is in state m at time t. Without loss of generality, we assume the following initial conditions: 1) $P_0(t) = 0$ for $t < 0$ and $P_0(0) = 1$ and 2) $P_m(t) = 0$ for $m > 0$ and $t \leq 0$. For $m = 0$, $P_0(t) = \sum_{j=0}^{k-1} P_j(t-1)(1-s)$. Since, $\sum_{j=0}^{k} P_j(t) = 1$, $P_0(t) = [1-P_k(t-1)](1-s)$. For $m = 1, 2, ..k-1$, $P_m(t) = sP_{m-1}(t-1)$ and for $m = k$,

$$P_k(t) = sP_{k-1}(t-1) + P_k(t-1) = s^k P_0(t-k) + P_k(t-1). \quad (1)$$

Substituting for $P_0(t-k)$ we get

$$P_k(t) = P_k(t-1) + s^k(1-s)(1-P_k(t-1-k)), \quad (2)$$

for $t \geq k$ and $P_k(t) = 0$ for $t \leq k$. This difference equation can be used to compute the probability at time t that a MS has slept for k or more consecutively intervals over a period of t intervals.

3.2 Analysis of the Peering Scheme

Let N denote the number of peer in the PCA. The state transition diagram of the DMM is shown in Figure 1. State 0 indicates that it has all the required IRs and that its cache is in a consistent state. State 1 indicates that the MS has

slept for one or more consecutive intervals and missed some IR data. However, it can still get the missed IR from other peers that have it. State 2 indicates that none of the MS has valid IR. The diagram also shows the state-transition probabilities. Note that there is a transition from both State 0 and State 1 to State 2. This corresponds to the case when MS_t and all its N peers are asleep in the same interval. As a result no MS will have valid IR for that interval.

It can be shown that the probability of having incomplete IR data is given by

$$P_2(t) = s^{N+1} \sum_{k=0}^{t-1} (1 - s^{N+1})^t = 1 - (1 - s^{N+1})^t. \quad (3)$$

Thus the probability of having valid IR at time t is $1 - P_2(t) = (1 - s^{N+1})^t$ [1]. The effect of peering is evident from the above equation; the effective sleep probability is reduced from s to s^{N+1}, thereby significantly improving cache performance.

3.3 Analysis of the Combined Scheme

In this case we consider both peering and windowing. As before, k is the window size for the IR and N is the number of peers of MS_t. A DMM to describe this case will have $(N+1)^k$ states. We consider an approximate state machine with $(k+2)$ states shown in Figure 1. State k indicates that the MS_t has slept for k or more consecutive intervals and its local IR is incomplete. However, at least one of the peers has required IR and using which the MS could complete the required list of IRs. State $k+1$ indicates that all the MSs have incomplete IRs and the union of all the IRs list is also incomplete. There is a transition from State k to State 0 since MS_t can obtain all the IRs from other peers even after having slept for k or more consecutive intervals. The term $a(t)$ which indicates no peer-help is available is given as:

$$a(t) = (sP_{k-1}(t-1) + P_k(t-1))^N. \quad (4)$$

The term $x(t)$, which indicates that MS_t can get the IRs it missed from other peers, is given by

$$x(t) = 1 - (1 - (sP_0(t-1) + (1-s)(1 - P_k(t-1) - P_{k+1}(t-1)))^N. \quad (5)$$

Using this model, we can compute the probability that the MS_t has incomplete IRs if it slept for k or more consecutive intervals between two queries that are i intervals apart in a situation where there are N other peers helping the MS_t to maintain valid IR. As before, the probability $P_{k+1}(t)$ can be solved recursively.

[1] In [10], this probability of obtaining valid data at a particular interval t is given as $(1 - s^N)^t$. The derivation had ignored the case where the MS_t also contributes through peering. That is, the IR is valid because MS_t is awake and has a valid IR while all other N peers are asleep. The derivation here accounts for this case.

4 Performance Metrics

The improvement in cache coherency due to windowing and peering is measured by using cache hit rate and throughput as in [2] and [10]. Note that we only consider the hit rate due to cache coherency; effects of cache size and replacement policies are not considered in this study.

4.1 Hit Rate

To determine the cache hit rate, we consider two consecutive queries that are i intervals apart. Using the notations defined earlier, the following conditions must be satisfied for a cache hit for the second query: 1) probability that there are no updates during the i intervals which is given by ${u_0}^i$, 2) probability that there are no queries during the $i-1$ intervals between queries which is equal to ${p_0}^{i-1}$, 3) probability that MS_t has a valid IR at interval i which is equal to $P_v(i) = 1 - P_{k+1}(i-1) - (1 - x(i-1))P_k(i-1)$, and 4) the first query arrived at interval 1. Taking all these four conditions, the cache hit probability is given by

$$h_{gen} = (1 - p_0) \sum_{i=1}^{\infty} {u_0}^i {p_0}^{i-1} P_v(i). \tag{6}$$

For the case of $k = 1$, the term $P_v(i) = 1 - P(2) = (1 - s^{N+1})^i$. For the case of $N = 0$, the term $P_v(i) = 1 - P_k(i-1)$, as state $k+1$ is renamed as k.

4.2 Throughput

To compute the throughput we follow the analysis done in [2] and [10]. As mentioned before, we assume that window w is a multiple of L and thus $w = kL$ and the bandwidth of the wireless network is W. Therefore, bandwidth available for each interval is LW. We assume that the number of bits per up-link query is b_q, and the number of bits per query answer is b_a. Each timestamp takes b_T bits and there are n items in the database.

Without Peering. The bandwidth available for queries is $LW - B_c$, where B_c is the bandwidth needed to broadcast the IR and is computed as follows: The expected number of items that changed in window w denoted by $nc = n(1 - e^{-\mu kL})$. Total size of the IR denoted by $B_c = b_T + n_c(log(n) + b_T)$, where $log(n)$ is the number of bits needed to transmit an object's ID, and b_T bits are needed to transmit its timestamp. The first b_T term represents the bits needed to send the IR report's timestamp.

Throughput T is the number of queries processed per interval by MS_t. Throughput due to cache misses $= T(1-h)$, where h is the cache hit rate. Traffic due to queries that resulted in cache misses $= T(1-h)(b_q + b_a) = LW - B_c$ from which we get

$$T = (LW - B_c)/(1-h)(b_q + b_a). \tag{7}$$

We can substitute for the hit rate derived in the previous section and compute the throughput.

With Peering. In this case the bandwidth available for queries will be lower because the communication between the peers uses up some of the bandwidth. Let B_{P2P} be the bandwidth used for communication between peers which consists of two components, i.e., $B_{P2P} = B_{P2P1} + B_{P2P2}$ where B_{P2P1} is the bandwidth required to query other other peers and B_{P2P2} is the bandwidth required to respond to a peer query.

Calculating B_{P2P1}. Let b_{req} be the fixed number of bits required to broadcast request to peers. Let N_{res1} be the number of peers that respond and b_{res} be the size of each response. Let p_{reqh} denote the probability that MS_t will send out query to its peers and let p_{resh} denote the probability that a peer receiving the query will respond with a subset of of the IR requested by MS_t. Then

$$B_{P2P1} = p_{reqh} * (b_{req} + N_{res1} * b_{res}). \tag{8}$$

where $p_{reqh} = (1-s)s^k$. Further, $N_{res1} = N * p_{resh}$ where $p_{resh} = (1-s)(1-s^k)$, as it must be awake and has not slept for k consecutive intervals. Thus,

$$N_{res1} = N * (1-s)(1-s^k). \tag{9}$$

The expected number of intervals asleep is equal to $s/(1-s)$ as derived in [10]. Therefore, conditional expectation of number of IRs needed given that the MS_t has slept for k or more intervals is still $s/(1-s)$, although the probability of this occurring is much smaller (s^k). Thus the size of this response $b_{res} = s/(1-s) * B_c$, where B_c is the mean bandwidth needed to transmit an IR.

Calculating B_{P2P2}. Let b_{req} be the bits used to broadcast request to peers. This has a fixed and known value. Let N_{req2} be the number of peers that request for peer help and N_{res2} be the number of responses sent by MS_t. As before, let b_{res} be the size of each response. Finally, let p_{vIR} denote probability that the MS receiving the query has at least one valid IR. Let B_{P2P2} denote the bandwidth required to respond to a query and is given by

$$B_{P2P2} = (1-s)(N_{req2} * b_{req} + (p_{vIR}) * N_{res2} * b_{res}) \tag{10}$$

where, $p_{vIR} = (1-s^k)$, $N_{req2} = N * Pr(MS\ requesting\ help)$ which is equal to $N * ((1-s) * s^k)$. Finally, $N_{res2} = N_{req2}$ because the MS_t is assumed to respond to all requesters, if it has valid IRs.

5 Results and Discussions

All the experiments were run with the following parameter values: the length of the IR interval L was set to 10 seconds, two different query rates were used, i.e., λ was set to 0.001 or 0.1 queries per second, and the update rate μ was set to 0.0001 updates per second.

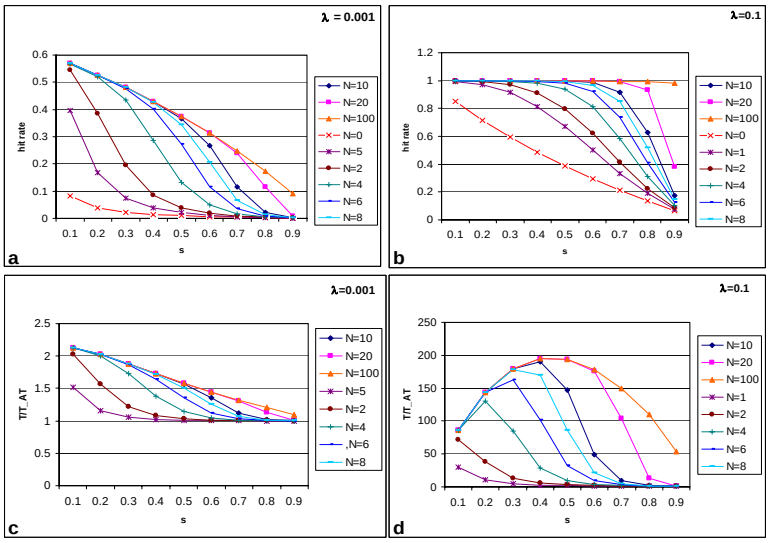

Fig. 2. Hit rate (a and b) and throughput (c and d) for peering only scheme (Note that normalizing throughput T_{AT} is for $k = 1$ and no peering).

5.1 Peering Only

The hit rate and improvement to throughput in the peering only scheme are shown in Figures 2. As expected the hit rate decreases with increasing s. Also the hit rate increases with increasing query rate λ. This is because when the query rate is high, queries occur at fewer intervals apart, where the probability of having incomplete IR data is lower. When queries occur far apart, the hit rate is lower as the probability of having incomplete IR data is higher. Therefore, at a low query rate ($\lambda = 0.001$), the hit rate is very low. As queries come farther and farther apart, the probability of having incomplete IR data will eventually be 1 (for any value of s except 0).

The hit rate increases with increasing N. The benefit of having N peers is to reduce the effective sleep probability from s to s^N (see Equation(3)).

As expected, the throughput improvement is much higher for high query rate ($\lambda = 0.1$) as compared to low query rate ($\lambda = 0.001$). The benefit of peering levels off at higher values of s because the increased peer-to-peer traffic consumes bandwidth.

5.2 Windowing Only

The results of windowing scheme is shown in Figures 3. For high query rates, the benefit of windowing is more at higher values of s. This is because when ss is high, the hit rate is low for the case without windowing. Windowing reduces the effective sleep probability and thus increases the hit rate. For low query

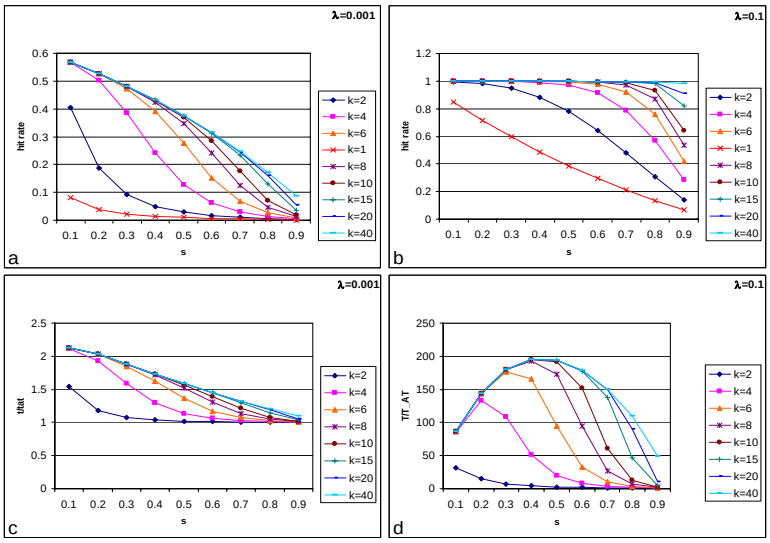

Fig. 3. Hit rate (a and b) and throughput (c and d) for windowing only scheme (Definition of T_{AT} same as before).

rates, the effect of windowing is more significant at lower values of s. At higher values of s, windowing is not as effective because the queries are too far apart for any windowing to make a difference. With respect to the throughput, as in the case of peering, the improvement drops at higher values of s because of higher bandwidth overhead due to larger window size.

5.3 Combined Scheme

The improvement to hit rate and throughput due to a combination of windowing and peering is shown in Figures 4. To study the relative effectiveness of windowing versus peering, we chose N and k values such that the product $(N+1) * k$ is constant denoted by C and set equal to 10 for results shown here. For high query rates increasing k improves the hit rate better. When $k = C$ (i.e., only windowing), the probability of having incomplete IR data for the first $(C-1)$ intervals is equal to zero and is equal to s^C at interval C. In contrast, when $N+1 = C$ (only peering), the conditional probability of having incomplete IR data is s^{N+1} in each of the C intervals and the probability of having incomplete IR data is $1-(1-s^C)^C$ at interval k. When query rate λ is high, the first few intervals are more important in the analysis, as the second query is expected in that duration. However, during this same period, as shown above, probability of having incomplete IR data is higher in the case of peering. Therefore, the case for windowing performs better than the peering.

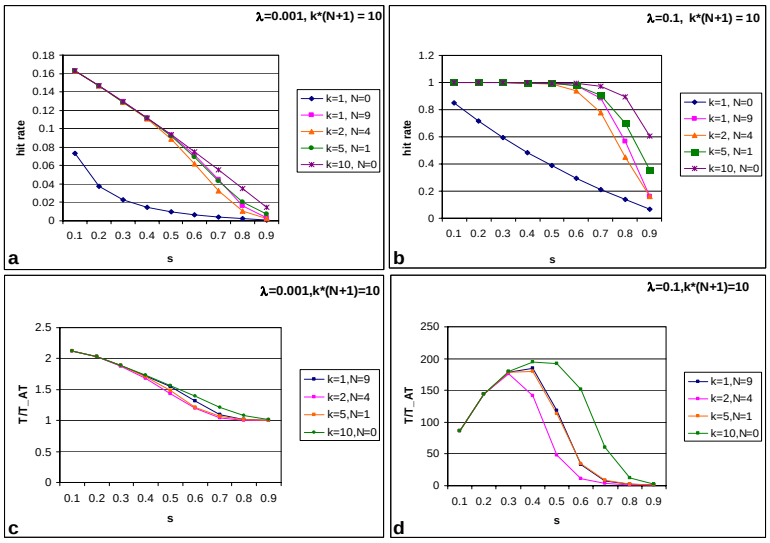

Fig. 4. Hit rate (a and b) and throughput (c and d) for the combined (Definition of T_{AT} same as before).

6 Related Work

In this paper we studied two methods to improve cache invalidation upon update using IRs ([2], [10]). Many other works ([5], [4], [11], [3], [12], and [8]) also dealing with cache invalidation have been published. Group-based invalidation schemes were proposed in [8] to retain as many valid objects as possible by checking for cache validity with the server upon reconnection. A different type of IR was proposed in [6] where the IR contains a set of bit sequences, each associated with a timestamp. The bits represent data objects in the database and indicate change to objects. The scheme reduces the number of cached objects discarded, but increases the size of the IR report. An asynchronous and stateful approach was proposed in [7] to reduce query latency and number of discarded objects. In [5] the authors have proposed an adaptive algorithm which combines the TS and Bit-Sequence approaches based on current query and update rate to achieve better performance. Another approach to reduce query latency and improve bandwidth utilization was proposed in [4] by repeating a small fraction of the IR information within the IR interval. In [11] the authors evaluate the cache performance when using some of the above schemes.

7 Conclusions

In this paper we proposed a unified mathematical model for studying the improvement to cache coherency using windowing, peering and a combination of

both. From the results we conclude that both methods provide significant improvements to cache performance, and that their performance is comparable for most part. Both windowing and peering consume bandwidth and thereby reduce throughput. The extra bandwidth consumed by windowing is of the order of k, while for peering it is of the order of N^2. Since the two schemes are orthogonal, they can be combined to improve cache performance greatly. Depending on the relative costs of bandwidth and power consumption for Base Station to Mobile Station and peer-to-peer communications, one can choose to combine the two methods in a way to optimize the cache performance. Future work can include studying the performance of the three schemes using different cost models for bandwidth and power consumption.

References

1. A.W.Drake. Discrete state markov processes. *Chapter 5: In Fundamentals of Applied Probability Theory*, 1967.
2. D. Barbará and T. Imieliński. Sleepers and workaholics: caching strategies in mobile environments. *MOBIDATA: An Interactive journal of mobile computing*, 1(1):1–12, 1994.
3. J. Cai and K. Tan. Energy-efficient selective cache invalidation. *Wireless Networks*, 5(6):489–502, 1999.
4. G. Cao. A scalable low-latency cache invalidation strategy for mobile environments. In *Proceedings of the sixth annual international conference on Mobile computing and networking*, pages 200–209. ACM Press, 2000.
5. Q. Hu and D. K. Lee. Cache algorithms based on adaptive invalidation reports for mobile environments. *Cluster Computing*, 1(1), 1998.
6. J. Jing, A. Elmargarmid, S. Helal, and R. Alonso. Bit-sequences: An adaptive cache invalidation method in mobile client/server environments. *ACM/Baltzer Mobile Networks and Applications*, 2(2), 1997.
7. A. Kahol, S. Khurana, S. Gupta, and P. Srimani. A strategy to manage cache consistency in a distributed mobile wireless environment, 2000.
8. P.S.Yu K.L.Wu and M.S.Chen. Energy-efficient caching for wireless mobile computing. In *Proceedings of the 12th International Conference on Data Engineering*, pages 336–343. IEEE, February 1996.
9. P. Nuggehalli, V. Srinivasan, and C. Chiasserini. Energy-efficient caching strategies in ad hoc wireless networks. In *The Fourth ACM International Symposium on Mobile Ad Hoc Networking & Computing MOBIHOC 2003*, 2003.
10. J. P. Pandya, P. Mohapatra, and D. Ghosal. Asymptotic analysis of a peer enhanced cache invalidation scheme. In *Proceeding WiOPT: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, pages 200–209. IEEE Press, 2004.
11. K. Tan, J. Cai, and B. Ooi. An evaluation of cache invalidation strategies in wireless environments. *IEEE Transactions on Parallel and Distributed Systems*, 12(8):789–807, 2001.
12. B. Zheng, J. Xu, and D. Lee. Cache invalidation and replacement strategies for location-dependent data in mobile environements. *IEEE Transsactions on Computers*, 51(10):1141–1153, 2002.
13. R. Zheng, J. Hou, and L. Sha. Asynchronous wakeup for ad hoc networks. In *The Fourth ACM International Symposium on Mobile Ad Hoc Networking and Computing MOBIHOC 2003*, 2003.