Asymptotic Analysis of a Peer Enhanced Cache Invalidation Scheme

Julee Pandya[1], Prasant Mohapatra[2] and Dipak Ghosal[3]

University of California, Davis, CA 95616, USA
pandya@cs.ucdavis.edu, prasant@cs.ucdavis.edu, 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. Because the wireless channel is inherently a broadcast medium, a cache coherency scheme in which the server broadcasts cache invalidation reports (IRs) containing data update information is suitable. However, one key issue that IR based techniques must resolve is the intermittent disconnectedness of MSs. Since connecting to the network uses power and because power is a limited resource, MSs connect to the network only when necessary. During periods when the MS is disconnected, IRs are missed, which may result in the MS's cache becoming invalid. This means that upon reconnection, the MS will have to purge its cache if it missed an IR while it was disconnected. In this paper, we propose a Peer Enhanced Cache (PEC) invalidation scheme. This scheme utilizes the ad-hoc mode of operation which is now available with new generation wireless interfaces. PEC exploits the peer-to-peer capability that is enabled by ad-hoc mode by having MSs store IRs on behalf of other MSs. If an MS misses an IR due to disconnection, it can retrieve the missed IRs from its peers. PEC can be used in conjunction with any cache invalidation method since it is an orthogonal scheme. We develop a mathematical model to derive the throughput and hit rate of PEC when it is used in combination with the previously proposed Amnesic Terminal (AT)[1] cache invalidation scheme. Our results show that incorporating PEC significantly improves both the throughput and the hit rate.

1 Introduction

The key characteristics of the wireless network are the mobility of the clients, the scarcity of resources such as power and bandwidth, the varying capabilities of wireless devices, and the disconnectedness of the clients. These characteristics pose significant challenges to achieving fast reliable access for various types of content. The challenges of designing a protocol for wireless access lie in the fact that designs must take all of these attributes of the wireless network into account. One important mechanism that can help overcome these design challenges is caching.

Caching is an important mechanism in the wireless domain because it greatly reduces access time by avoiding unnecessary access across the wireless channel. In order to implement a robust caching scheme, caches must be placed locally on the wireless devices and at the server. In such an organization, the cache coherency scheme used to maintain cache coherency between the wireless devices and the server is a major factor in determining the effectiveness of the caching mechanism. If the cache coherency scheme is not properly designed, the performance of the system could be even worse than a configuration without any caching mechanism.

There are two main types of cache coherency techniques: Time To Live (TTL) based techniques and Invalidation Report (IR) based schemes. In the first type, the server associates a TTL parameter with each data item to be cached. Once the TTL value has expired, the item is no longer valid and the client must check with the server to see if the item has been modified. In the second technique, the server broadcasts IRs to the clients. These reports indicate which data items have been modified during a specified period of time. Clients use the IRs to update their caches.

Although both TTL and IR based schemes can be used for maintaining cached data at the client, TTL based techniques are not ideal for wireless environments. In TTL based schemes, once the TTL of a data item has expired, clients must query the server to determine if the item has been modified. As the number of clients grows, the wireless uplink is likely to become the bottleneck. For IR based schemes, the wireless link bandwidth is not a limiting factor because any client can listen to the IRs. In addition, schemes that use the TTL technique utilize the uplink channel more frequently than schemes that use the IR technique. Due to the fact that the uplink channel consumes more power than the downlink channel, TTL based caching schemes tend use more power than IR based caching schemes. As a result, IR based caching schemes are more suitable for wireless networks.

One of the key issues of IR based techniques arises due to the intermittent disconnectedness of the clients. Because 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, the rest of the network cannot communicate with it. Since the IRs broadcast during these periods are missed, the client's cache may become invalid. This means that upon reconnection, the client will have to purge its cache if it missed an IR when it was disconnected.

Many previously proposed IR methods [1, 5, 6, 4, 8, 3, 2, 9, 7] have addressed this disconnection issue. Many of these methods are variations of the Broadcasting Timestamp (TS) method proposed in [1]. In this paper, we propose a Peer Enhanced Cache (PEC) invalidation method. This cache invalidation method utilizes the ad-hoc mode of operation, which is now available with most new generation wireless interfaces. PEC exploits the peer-to-peer capabilities that are possible with ad-hoc mode by having MSs store IRs on behalf of other MSs. If an MS misses an IR due to disconnection, it can easily retrieve the missed IRs from its peers. Since PEC is orthogonal to other cache invalidation methods, it can be used in conjunction with any cache invalidation scheme. We demonstrate the effectiveness of PEC through a mathematical model in which we derive the throughput and hit rate of PEC when it is used in combination with the Amnesic Terminal (AT) scheme [1]. Our results show that enhancing AT with PEC significantly improves both the throughput and the hit rate.

The remainder of the paper is organized as follows. Section 2 gives the terminology used throughout the paper, and Section 3 reviews related work. In Section 4, we present PEC, which is the proposed cache invalidation method. In Section 5, we present a mathematical analysis of PEC. We conclude the paper in Section 6 with a discussion of future work.

2 Terminology

The terms *client* and *MS (mobile station)* are used interchangeably. These terms refer to the physical wireless device. The terms *BS (base station)* and *server* are also used interchangeably. The reason for this usage will be explained further in Section 4.1. We also use assume that *going to sleep* and *disconnecting* are equivalent as well as *waking up* and *reconnecting.*

3 Related Work

3.1 Broadcasting Timestamp (TS)

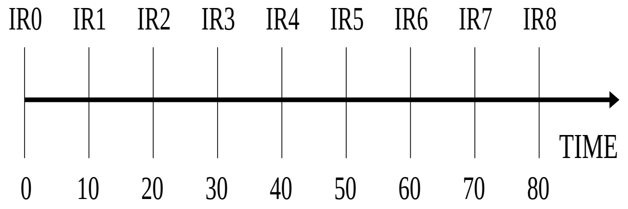

Figure 1. An example of the TS scheme with L=10 seconds and w=20 seconds.

Like other studies, we use the TS scheme as the underlying framework [1]. In this scheme, the server broadcasts IRs every L time units. Each IR indicates the items that have been modified during a specified window of time w, where w is some multiple of L. For example, assume L is 10 seconds and w is 20 seconds. From Figure 1, we can see that IR3, which is broadcast at time 30 seconds, will indicate the data items that have been modified between times 10 seconds and 30 seconds. Therefore, if a client disconnects at time 12 seconds and reconnects at time 25 seconds, the client will be able to update its cache using IR3.

However, if the client is disconnected for longer than w time units (20 seconds in this example), the client will have to purge its cache.

Note that the bigger the w, the longer the client can disconnect without having to purge its cache. However, increasing w also increases the size of the IR. Therefore, when determining the value of w, one must consider how much update history is necessary in each IR as well as consider the bandwidth usage of the IRs.

One important feature of the TS scheme is that MSs 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.

4 Peer Enhanced Caching (PEC)

4.1 Reference Architecture

We assume that the Base Stations (BS) act as servers that broadcast IRs. We define the base station coverage area (BSCA) to be the area that the BS services. In addition, each MS defines a peer coverage area (PCA), which includes all MSs within a certain radius. Thus, MS1 is a peer of MS2 if MS1 is in MS2's PCA. We also assume that each MS has two logical interfaces. One interface is used for peer-to-peer communication, and the other is used for communicating with the BS.

4.2 Algorithm

In our algorithm, we assume that each MS stores IRs when they are awake. If an MS, which was disconnected from the network, reconnects, it will be able to retrieve the IRs it needs to update its cache from its peers. Upon reconnecting to the network, an MS will check if it can use the next IR from the BS to update its cache. If it can, then it will use it. If it cannot, then the MS will send a broadcast to its PCA to check if any MS has the IRs that it needs to update its cache. If no MS in the PCA has the needed IRs, the MS will purge its cache. We also assume that before disconnecting from the network, each MS will listen to the next IR and then wait for a short time δ to respond to peer requests for invalidation reports.

5 Analysis

5.1 Model

In this section we develop a simple model to quantify the benefits of PEC. This model builds upon earlier work reported in [1]. We assume that the underlying cache invalidation scheme used is the Amnesic Terminals (AT) presented in [1]. The AT scheme is similar to the TS scheme discussed in Section 3.1. Recall that in the TS scheme, the server broadcasts IRs every L time units, and each IR indicates the data items that have been modified during a specified window of time, w. The AT scheme is the same as the TS scheme with w always equal to 1. This means that there is no overlap in the information contained in each IR. As a result, if an MS goes to sleep at all, it will have to purge its cache.

The goal of our analysis is to measure the benefit of the peer-to-peer interaction of PEC. More specifically, we will compare the throughput and hit rate of the AT scheme to the throughput and hit rate of the PEC scheme when the AT scheme is used as the underlying cache invalidation method. We denote this combination of the PEC and AT schemes as PEC^{AT}. For the analysis, we consider a tagged MS, denoted by MS_t.

The parameters of the model are as follows. There are M MSs in the BSCA; this does not include MS_t. The database consists of n items. The bandwidth of the wireless channel is W, and the length of the IR interval is L. In ad-hoc mode, each MS has a range of r. We assume that the range of the BS is R. The number of bits per uplink query is b_q, and the number of bits per query answer is b_a.

The models for the query-update pattern and sleep-wake pattern are the same models that were used in [1]. Each MS queries a subset of the database with high locality. This subset is known as the hot spot and each item in the subset is queried at rate λ. Updates to to each data item are negative exponentially distributed with mean rate μ updates per second. The sleep-wake pattern is modeled by assuming that in each interval, an MS has a probability s of sleeping and $1 - s$ of being awake. The behavior of an MS in an interval is independent of the behavior in the previous interval.

For the mobility model, we assume that in each interval, MSs are randomly and uniformly distributed. Therefore, in every interval, an MS can be anywhere in the BSCA. Just as in the sleep-wake pattern, we assume that the behavior in each interval is independent of behavior in the previous interval.

It is important to note that the models used in our analysis are approximations. However, we use the same models to analyze both the AT and the PEC^{AT} schemes. Furthermore, many of the approximations made in the models put PEC^{AT} at a greater disadvantage than it would be if other correlated models were used. As a result, the PEC^{AT} scheme may perform better under other correlated models. For instance, the sleep-wake model used in our analysis assumes that the behavior of an MS in each interval is independent of the behavior of an MS in other intervals. This is often not the case. PEC may perform significantly better with a sleep-wake model that does not make this assumption.

In addition to the above parameters, we use the notation in Table 1, which was taken from [1]. The table shows the symbol, the probability that the symbol represents, and the corresponding event.

Table 1. Summary of notation.

Symbol	Probability	Event
j_0	$e^{-\lambda L}$	No queries in an interval given unit is wake in an interval
q_0	$(1-s)e^{-\lambda L}$	Awake and no queries in an interval
p_0	$s+q_0$	No queries in an interval
$1-p_0$	$(1-s)(1-e^{-\lambda L})$	1 or more queries in an interval
u_0	$e^{-\mu L}$	No updates during an interval
$1-u_0$	$1-e^{-\mu L}$	1 or more updates during an interval

5.2 Throughput

Generic Throughput Equation. To calculate the throughput T, we expand on the throughput analysis presented in [1]. First, we will derive the generic throughput equation, which was given in [1]. This equation will then be used to derive the throughput for both the AT and PEC^{AT} schemes.

In order to derive the generic throughput equation, we assume that each node has an available bandwidth of LW. The bandwidth available for queries that were cache misses is $LW - B_c - B_{P2P}$, where B_c is the bandwidth used to broadcast the IR and B_{P2P} is the bandwidth used for communication between peers. The throughput T is the number of queries per interval processed by MS_t. $T(1-h)$ gives the number of queries per interval that were cache misses, where h is the hit rate. Each cache miss requires $b_q + b_a$ bits; therefore the bandwidth needed to handle all cache misses is $T(1-h)(b_q + b_a)$. Since this quantity must equal $LW - B_c - B_{P2P}$, we obtain the following expression for T;

$$T = \frac{LW - B_c - B_{P2P}}{(1-h)(b_q + b_a)}. \tag{1}$$

An expression for B_c is shown in Equation 2 which is the same as that derived in [1]. The term $n(1-u_0)$ is the mean number of data items that were modified since the last IR was broadcast. The $log(n)$ term is the size of the timestamps that will be used in the IR.

$$B_c = n(1-u_0)log(n) \tag{2}$$

Throughput for the AT Scheme. While $B_{P2P} = 0$ for the AT scheme, the throughput T_{AT} for the AT scheme is given by

$$T_{AT} = \frac{LW - B_c}{(1-h_{AT})(b_q + b_a)} \tag{3}$$

where h_{AT} is the hit rate for the AT scheme and is derived in the next section.

Throughput for the $\boldsymbol{PEC^{AT}}$ Scheme. In order to calculate the throughput, $T_{PEC^{AT}}$, we need to calculate B_{P2P}. To compute B_{P2P}, we consider 2 cases. If we let the current interval be interval l, then in the first case, MS_t was asleep in interval $l-1$, which is the previous interval, and was awake in interval l. In this case, MS_t just woke up and therefore will broadcast a peer request to all MSs in its PCA in order to retrieve the IRs it missed while it was asleep. We denote the bandwidth used for peer communication for this case as B_{P2P_1}. In the second case, MS_t is awake in interval $l-1$, which again is the previous interval, and may be either awake or asleep in interval l. In either case, MS_t will respond to requests that it receives from its peers. We denote the bandwidth used in this case as B_{P2P_2}.

CASE 1: Calculating B_{P2P_1}. As stated above, B_{P2P_1} is the bandwidth used when MS_t sends a peer request to all MSs in its PCA. Let the peer broadcast issued by MS_t use b_{req} bits, and let the number of MSs in $MS_t's$ PCA that respond to the request be N_{res_1}. In addition, we let the size of each response be b_{res}. For this case, MS_t was asleep in interval $l-1$ and awake in interval l. Therefore, we need to compute the probability that MS_t woke up which is $s(1-s)$ and as a result we obtain

$$B_{P2P_1} = s(1-s)(b_{req} + N_{res_1} b_{res}). \tag{4}$$

Since b_{req} is fixed, we need to compute $Nres_1$ and b_{res}.

In our model, we assume that MSs are either awake or asleep for an entire interval. We also assume that before disconnecting from the network, an MS will listen to the next IR and then wait some small time δ to answer any peer requests for invalidation reports. Therefore, N_{res_1} is equal to the number of MSs in the tagged MS's PCA that were awake in interval $l-1$.

In order to calculate N_{res_1}, we define W to be a random variable representing the number of MSs in the BSCA that were awake in interval $l-1$. Since the probability of an MS going to sleep in any interval is independent of the previous interval, W has a binomial distribution. Therefore, the expected value of the number of MSs in the BSCA that were awake in interval $l-1$ is given by

$$\begin{aligned} E[W] &= \sum_{j=0}^{M} jP(W=j) = \sum_{j=0}^{M} \binom{M}{j} j(1-s)^j s^{M-j} \\ &= M(1-s). \end{aligned}$$

Only a fraction of the $M(1-s)$ MSs in the BSCA that were awake in interval $l-1$, will be in $MS_t's$ PCA. If we approximate the PCA to be a sphere of radius r and the BSCA as a sphere of radius R, then the number of MSs in $MS_t's$ PCA is proportional to r^2/R^2 since the MSs in the BSCA are randomly and uniformly distributed in each interval. Therefore,

$$N_{res_1} = \frac{r^2}{R^2} M(1-s). \tag{5}$$

In order to compute b_{res}, we first note that each response is made up of some number of invalidation reports. Therefore $b_{res} = CB_c$, where C is the number of IRs in each peer response. Let MS_p be $MS_t's$ peer, and let C represent the number of intervals that MS_p was awake during the period that MS_t was asleep. If we let X be a random variable representing the number of consecutive intervals that MS_t was asleep, then

$$\begin{aligned} E[X] &= \sum_{j=0}^{\infty} jP(X=j) = \sum_{j=0}^{\infty} j(1-s)s^j \\ &= \frac{s}{1-s}. \end{aligned}$$

If Y is a random variable that denotes the number of intervals MS_p was awake while MS_t was asleep, then

$$\begin{aligned} E[Y] &= \sum_{j=0}^{E[X]} jP(Y=j) \\ &= \sum_{j=0}^{E[X]} j \binom{E[X]}{j} (1-s)^j s^{E[X]-j} \\ &= E[X](1-s) = s. \end{aligned}$$

Thus, $C = s$ and $b_{res} = sB_c$ and we obtain

$$B_{P2P_1} = s(1-s)(b_{req} + \frac{r^2}{R^2}M(1-s)sB_c). \tag{6}$$

CASE 2: Calculating B_{P2P_2}. In the second case, MS_t is awake in interval $l-1$, and will respond to requests it receives from its peers. Let N_{req_2} be the number of requests that MS_t receives, and let N_{res_2} be the number requests that MS_t responds to. Since the probability that MS_t is awake in interval $l-1$ is $(1-s)$, the bandwidth, B_{P2P_2}, for this case is given by

$$B_{P2P2} = (1-s)(N_{req_2}b_{req} + N_{res_2}b_{res}). \tag{7}$$

Note that b_{req} and b_{res} are the same as they were for the calculation of B_{P2P_1}. Also, because we assume that MS_t will respond to all requests, $N_{res2} = N_{req2}$. Therefore, we only need to compute N_{req_2} in order to derive B_{P2P_2}.

Since N_{req_2} is the number of peer requests that MS_t receives, it is equal to the number of MSs in $MS_t's$ PCA that just woke up. In order to compute this value, we define a random variable Z that denotes the number of MS's in the BSCA that just woke up. Since Z has a binomial distribution with $p_{success} = s(1-s)$, we obtain

$$\begin{aligned} E[Z] &= \sum_{j=0}^{M} jP(Z=j) \\ &= \sum_{j=0}^{M} j\binom{M}{j}(s(1-s))^j(1-s(1-s))^{M-j} \\ &= Ms(1-s). \end{aligned}$$

$E[Z]$ gives the number of MSs in the BSCA that just woke up; however, N_{req_2} is the number of MSs in $MS_t's$ PCA that just woke up. Using the same reasoning that was used to derive N_{res_1}, we get $N_{req2} = \frac{r^2}{R^2}E[Z] = \frac{r^2}{R^2}Ms(1-s)$ which leads to

$$B_{P2P_2} = \frac{r^2}{R^2}Ms(1-s)^2(b_{req} + sB_c) \tag{8}$$

and

$$\begin{aligned} B_{P2P} &= B_{P2P_1} + B_{P2P_2} \\ &= s(1-s)b_{req} + \frac{r^2}{R^2}Ms(1-s)^2(b_{req} + 2sB_c) \end{aligned} \tag{9}$$

Finally, the throughput of the PEC^{AT} is given by

$$T_{PEC^{AT}} = \frac{LW - B_c - (s(1-s)b_{req} + \frac{r^2}{R^2}Ms(1-s)^2(b_{req} + 2sB_c))}{(1-h_{PEC^{AT}})(b_q + b_a)} \tag{10}$$

The hit rates h_{AT} and $h_{PEC^{AT}}$ are derived in the next section.

5.3 Hit Rate

In order to calculate the hit rate, we assume that MS_t has made at least two queries, which occurred i intervals apart from each other. From Figure 2, we can see that if an update occurred during any of the i intervals, there would be a cache miss. Furthermore, because MS_t generated a query in the i^{th} interval, it must have been awake in that interval. This means that there are $i-1$ intervals in which MS_t may have any sleep pattern. There are two cases that would ensure that the second query is a hit. In the first case, we assume that MS_t does not sleep during any of the $i-1$ intervals. We will show that the hit rate in this case is simply the hit rate achieved with the AT scheme, which is denoted h_{AT}. For the second case, MS_t may have any arbitrary sleep pattern during the $i-1$ intervals, except for the pattern when MS_t is awake for the entire $i-1$ intervals. This pattern is handled in the first case. We denote the hit rate for this case as h_{P2P}.

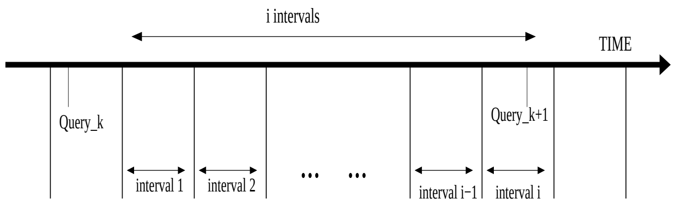

Figure 2. MS_t generates 2 queries that are i intervals apart.

CASE 1: Calculating h_{AT}. In the first case, MS_t does not sleep during any of the $i-1$ intervals. In this case, no peer requests are ever sent and therefore, the hit rate is simply equal to the hit rate for the AT scheme. The hit rate for the AT scheme, h_{AT}, was derived in [1] and is given by

$$\begin{aligned} h_{AT} &= (1-p_0)\sum_{i=1}^{\infty} q_0^{i-1}u_0^i \\ &= \frac{(1-p_0)u_0}{1-q_0u_0} \end{aligned} \tag{11}$$

The first term $1-p_0$ refers to the probability of MS_t generating a query in interval i. The term q_0^{i-1} in the sum is the probability of not generating any queries during the $i-1$ intervals between the queries, and the u_0^i term is the probability of not having any updates during the i intervals. In order to calculate h_{AT}, we sum over all possible values of i.

CASE 2: Calculating h_{P2P}. In the second case, MS_t may have any arbitrary sleep pattern during the $i-1$ intervals, except for being awake the entire $i-1$ intervals. This pattern was handled in the first case. Therefore, the probability of this case gives us the hit rate achieved when MS_t went to sleep at least once during the $i-1$ intervals. We call this hit rate h_{P2P} because MS_t must send at least 1 peer request in order to validate its cache. We assume that N is the number of MSs in $MS_t's$ PCA, not including MS_t. Using the same reasoning from Section 5.2, $N = r^2/R^2M$ and thus the hit rate is given by

$$\begin{aligned} h_{P2P} &= (1-p_0)\sum_{i=1}^{\infty} u_0^i(1-s^N)^{i-1}(p_0^{i-1}-q_0^{i-1}) \\ &= (1-p_0)u_0\left(\frac{1}{1-(1-s^N)u_0p_0} - \frac{1}{1-(1-s^N)u_0q_0}\right) \end{aligned} \tag{12}$$

In the above equation, the first term $1-p_0$ is the same as before; it refers to the probability of MS_t generating a query in interval i. The term u_0^i in the sum is the probability of not having any updates during the i intervals. The term $(1-s^N)^{i-1}$ in the sum is the probability that for each of the $i-1$ intervals, there was at least 1 MS awake. This term allows us to support any sleep pattern that MS_t may have during the $i-1$ intervals. If at least 1 MS in MS_t's PCA is awake in each of the $i-1$ intervals, then MS_t will always be able to retrieve the needed IRs when it wakes up. To see this, assume that there is only 1 MS awake in MS_t's PCA for each of the $i-1$ intervals. The MS that is awake in interval $i-k$ will be able to pass along the IR for that interval to the MS that is awake in interval $i-k+1$. Similarly, the MS awake in interval $i-k+1$ will be able to pass along the IRs from interval $i-k$ and interval $i-k+1$ to the MS awake in interval $i-k+2$. Note that $(1-s^N)^{i-1}$ gives the probability that at least 1 MS is awake. To see this first note that the s^N gives the probability that all MSs in MS_t's PCA are asleep in an interval. Therefore, $1-s^N$ yields the probability that at least 1 MS is awake in an interval. In order to have at least 1 MS awake for $i-1$ intervals, we must raise this probability to the $i-1$ power.

The last term in Equation 12 is the sum of $p_0^{i-1} - q_0^{i-1}$. This term gives the probability that MS_t went to sleep at least once in the $i-1$ intervals and did not generate any queries in the $i-1$ intervals. The first term p_0^{i-1} is the probability that there were no queries in the $i-1$ intervals. We subtract q_0^{i-1} to remove the event when MS_t is awake the entire $i-1$ intervals because this was already handled in the first case. We must sum over all possible values of i in order to calculate h_{P2P}.

The hit rate for the PEC^{AT} scheme, denoted $h_{PEC^{AT}}$, is simply $h_{AT} + h_{P2P}$. So we get

$$h_{PEC^{AT}} = h_{AT} + h_{P2P}$$

$$= (1 - p_0)u_0(\frac{1}{1 - q_0u_0} + \frac{1}{1 - (1 - s^N)u_0p_0} - \frac{1}{1 - (1 - s^N)u_0q_0}) \tag{13}$$

5.4 Results

To numerically compare the AT and PEC^{AT} schemes, we considered the following system parameters. The length of the IR interval, L, is 10 seconds, and the bandwidth of the wireless channel, W, is 1 Mbps. The query rate, λ, is 0.1 queries per second, and for the data update rate, μ, we considered two values 0.0001 and 0.01 updates per second. There are 100 items in the database and 50 nodes (not including MS_t) in the BSCA. Also, $b_a = 1024$ bytes, $b_q = 512$ bytes, and $b_{req} = 512$ bytes. The range of the BS is $R = 200$ meters and the range, r, of the MSs is 40, 50, or 60 meters.

Figures 3 and 4 show $T_{PEC^{AT}}$ normalized to T_{AT}. From the graphs, we can see that the throughput for the PEC^{AT} scheme can have up to a 100 percent improvement over the AT scheme. This improvement is due to the fact that MSs do not have to purge their caches as much as in the AT scheme. As a result, MSs answer data requests using their caches, which is much more efficient than sending uplink requests to the server. As we increase s, the improvement of the proposed scheme over the AT scheme increases up to a certain value of s, beyond which there is a sharp decrease in the throughput improvement.

Figures 5 and 6 give $h_{PEC^{AT}}$ normalized to h_{AT}. We can see from the figures that the hit rate for the PEC^{AT} scheme can be greater than two times the hit rate for the AT scheme. As s increases, the improvement of h_{PEC} over h_{AT} increases because more MSs are sleeping. As a result, there are more MSs that can benefit by retrieving IRs from other MSs. However, beyond a certain value of s, which is around 0.6, the improvement in $h_{PEC^{AT}}$ decreases. This is due to the fact that an MS is less likely to be able to retrieve IRs from the other MSs in its PCA because they are either asleep or were asleep and therefore do not have the IRs the MS needs.

In addition to s having an impact on the performance improvement, the range of an MS also influences this improvement. As the range of the MSs increase, both $T_{PEC^{AT}}$'s and $h_{PEC^{AT}}$'s improvement over T_{AT} and h_{AT}, respectively, also increases. This is due to the fact that as r increases, the number of MSs in a given PCA also increases; therefore, the likelihood that an MS will be able to retrieve the IRs it needs also increases. Another interesting parameter is the data update rate, μ. From the figures, we can see that as μ increases, the benefit of PEC decreases. The reason is that as data is being updated more frequently, the chances of a cache miss increase. When an MS goes to sleep, there is a greater chance that the data in its cache will be updated. However, PEC is beneficial only if an MS goes to sleep. If an MS does not sleep, then $T_{PEC^{AT}} = T_{AT}$ and $h_{PEC^{AT}} = h_{AT}$. The improvement of $T_{PEC^{AT}}$ and $h_{PEC^{AT}}$ does not decreases because an MS is unable to validate its cache; it decreases because the data was updated while an MS was asleep.

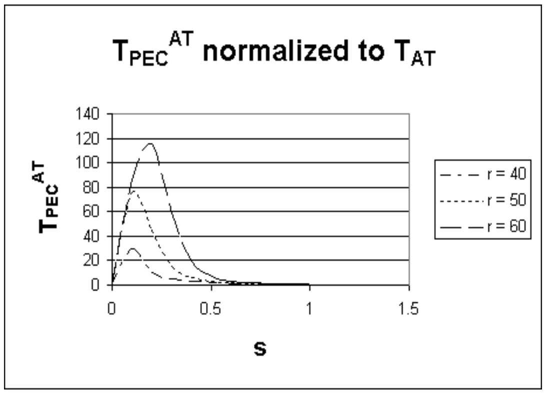

Figure 3. $T_{PEC^{AT}}$ normalized to T_{AT} with $\mu = 0.0001$ (The asymptotic value of $T_{PEC^{AT}} = 1$.)

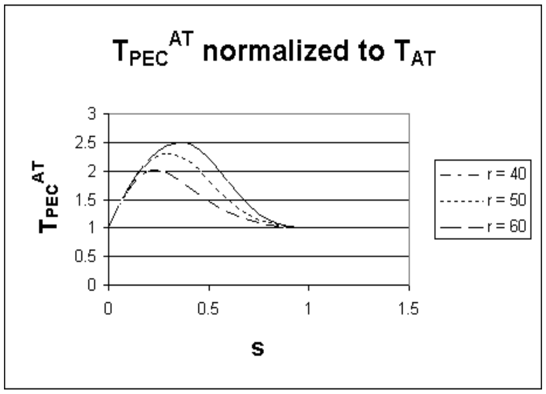

Figure 4. $T_{PEC^{AT}}$ normalized to T_{AT} with $\mu = 0.01$ (The asymptotic value of $T_{PEC^{AT}} = 1$.)

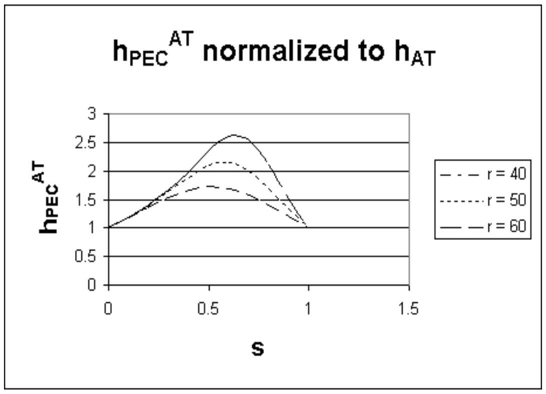

Figure 5. $h_{PEC^{AT}}$ normalized to h_{AT} with $\mu = 0.0001$

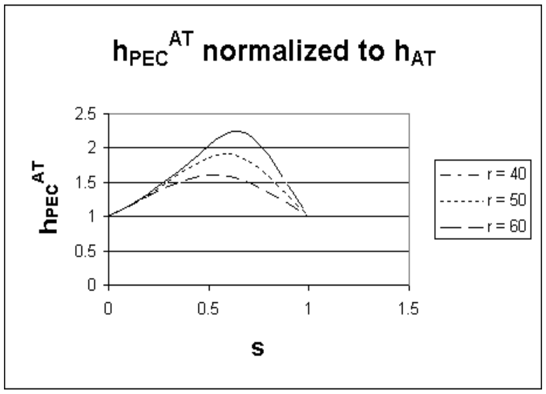

Figure 6. $h_{PEC^{AT}}$ normalized to h_{AT} with $\mu = 0.01$

6 Conclusion

In this paper we proposed a new cache invalidation scheme which exploits peer-to-peer interaction among mobile nodes. In the new scheme, referred to as Peer Enhanced Caching (PEC), mobile stations (MSs) store invalidation reports and can forward them to peers when they reconnect to the network when they wake up from sleep. We have shown that PEC can greatly improve the cache hit rate when it is used in combination with other cache invalidation methods. In particular, we showed that PEC can provide more than a two-fold increase in the hit rate and up to 100 times improvement in the throughput over the Amnesic Terminal scheme. Future work includes extending the analysis to compare the hit rate and throughput of PEC when it is used in conjunction with the TS scheme discussed in Section 3.1. In addition to a mathematical model, a thorough investigation of PEC via simulation will be used to verify the results of the model and evaluate the performance improvements under realistic mobility models.

References

1. 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.
2. J. Cai and K. Tan. Energy-efficient selective cache invalidation. *Wireless Networks*, 5(6):489–502, 1999.
3. G. Cao. Proactive power-aware cache management for mobile computing systems. *IEEE Transactions on Computers*, 51(6):608–621, 2002.
4. Q. Hu and D. K. Lee. Cache algorithms based on adaptive invalidation reports for mobile environments. *Cluster Computing*, 1(1), 1998.
5. 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.
6. A. Kahol, R. Khurana, S. Gupta, and P. Srimani. An efficient cache maintenance scheme for mobile environment. In *International Conference on Distributed Computing Systems*, 2000.
7. A. Kahol, S. Khurana, S. Gupta, and P. Srimani. A strategy to manage cache consistency in a distributed mobile wireless environment, 2000.
8. K. Tan. Organization of invalidation reports for energy-efficient cache invalidation in mobile environments. *Mobile Networks and Applications*, 6(3):279–290, 2001.
9. 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.