Telecommun Syst (2016) 62:149–165
DOI 10.1007/s11235-015-0070-1

An adaptive cache invalidation technique for wireless environments

Rajeev Tiwari[1] · **Neeraj Kumar**[1]

Published online: 3 June 2015

Abstract With an evolution of Internet and related technologies such as 4G, and 5G, there is a need of fast response time for various queries raised by intermediate nodes and mobile terminals from infra structured-based/less networks. Although there has been many efforts in the past to address this issue by using an invalidation report (IR)-based cache management schemes which reduce the bandwidth requirements, and battery consumptions, but when the update rate of data at the server is high, then most of the existing approaches have long query latency due to large and fixed size of IRs, and broadcast time (BT) interval. To address these issues, in this paper, an adaptive cache invalidation technique (ACIT) is proposed. In comparison to the previous approaches, the proposed scheme uses different thresholds update rates for adaptive IR, and BT intervals. In the proposed scheme, only hot data updates in IR are recorded which results a less query delay, and bandwidth consumption. The performance of the proposed scheme was studied by extensive simulations by comparing it with the other state-of-the-art schemes using various metrics. The proposed scheme yields 1.5 times enhancement in query response time with a reduction in IR size to 25 % in comparison to other existing techniques. Moreover, there is a reduction of 119.89 % in broadcast time interval using the proposed scheme.

Keywords Cache invalidation · Cooperative caching · Cache hit ratio · Wireless networks

✉ Neeraj Kumar
nehra04@yahoo.co.in; neeraj.kumar@thapar.edu

Rajeev Tiwari
errajeev.tiwari@gmail.com

[1] Department of Computer Science and Engineering, Thapar University, Patiala, Punjab 147004, India

1 Introduction

From the last few decades, there has been exponential growth in the usage of Internet and related technologies for various applications such as health care, transportations, monitoring, and surveillance. During this era, various wireless standards have been developed to reduce the access delay of various services accessed by the end user. The popularity of different kinds of portable equipments to access these services from the service providers has also increased with an aim to provide an efficient wireless mobile environment where users can access and use various services with reduced delay. But, the access delay of these services are dependent on the type of network being used- wired/wireless, as both these types of networks have different characteristics which differentiate these from one another.

A Wireless network has mobile terminals (MTs), access points (APs), and base station (BS) to provide services to the end users. But, the MTs are having high mobility, which leads to dynamic topological changes [1] and peers communicate with one another similar to mobile adhoc network (MANET) [2–4]. There exist a hierarchical architecture in this environment in which MTs are connected to an AP, which is connected to a BS, and finally, BSs are connected to server, which contains the overall data repository for all the operations performed in this environment. The MTs are connected via unreliable low-bandwidth communication channels with the server having frequent disconnections due to power saving strategies, and mobility [5,6], i.e., this environment has a slow wireless link having MTs with limited battery power.

From the above discussion, it is clear that access delay is one of the major challenges in accessing various services when mobile clients have high mobility in wireless environment. One of the solutions to reduce the delay is- Caching, in which frequently accessed data items on the client side are

accessed to reduce the network traffic to enhance the network performance. It is one of the efficient techniques to enhance the network performance in which frequently accessed data items are placed either in the MTs memory or to the nearest APs, so that every time MTs need not to ask for data from the remote server which may cause a long access delay, e.g., If data is available in the cache of MTs, then they respond quickly for any query related to that data. However, data consistency must be maintained between MTs and servers to prevent clients from answering to those queries which are outdated by servers, i.e., which are invalidated from the server. A data item is invalidated if a query to access the cache has bee generated from the clients and it has been updated by the server and clients are still using the old copy of the same.

There are various types of cache management schemes for wireless environments which have been described in literature like time stamping technique (TS) [7] which records all updates pages for IR and broadcasts it to mobile nodes (MN).It has fixed size IR. In update invalidation report (UIR)[8] technique, a smaller fixed size m UIRs are sent between BT times and at BT time fixed IR sent. In SAS, uses group invalidation report (GIR) [9] which has longer delays and IRs. Many of these proposals were based on Invalidation Reports (IR) to handle frequent disconnections [7]. The server periodically broadcasts IRs to inform the clients of which data items have been updated during the most recent past time [10].

Most of the above techniques [7–9] suffer from longer query delays and large bandwidth consumptions due to the longer and fixed sized IRs. So in such as case, MTs cant only reply for query till its cached data is validated by IR, sent from server. Hence, longer IRs consume more bandwidths and increase the query response time for MTs. Authors in [8] has proposed an UIR-based cache technique in which a server sends UIRs after every L/mth seconds.This sending of UIRs may decrease query response time to the extent of hit occurrences, due to UIRs on MN. But, this replication of UIR instead of IR consumes a lot of bandwidth and also increases the congestion in the network. As it has been observed from the above discussion that the only update in data should not be the criteria for it to be included in IR, it will increase the size of IR and technique can loose its effectiveness. As hot data are accessed more frequently by the clients, so including hot data in IRs sent from the server can reduce the size of IRs and at same time it can be an effective option for increasing the hit ratio. So, author in [11,12] has proposed another cache invalidation technique (CIT), which was based on counter to identify a hot data item. For hot data identification, a counter is maintained for each data item on server and correct information is maintained so that data item is cached on each MT, but it generates large overhead when update rates are more on server. Therefore, these approaches

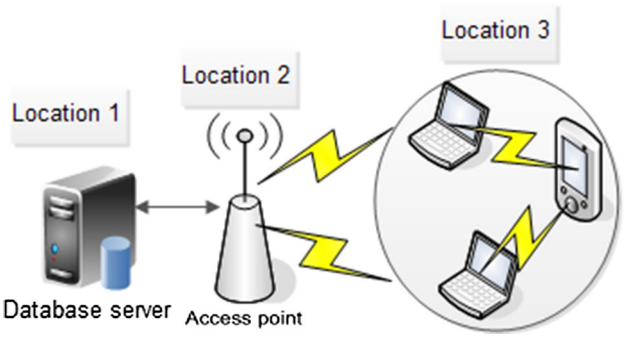

Fig. 1 Generalized architecture for various locations for cache invalidation

may not be suitable when update rate is higher in the wireless environment.

We have been motivated from our earlier solutions [13, 14], in which we have designed cache management scheme for peer-to-peer (P2P) mobile clients in vehicular environments where, the adaptive size of IRs with respect to hot data was not considered. But, in the current solution, we have considered this factor in different environment and tested the performance of the designed scheme using adaptive IR size for hot data items. In the current solution, IR is divided into two blocks as-current IR (CIR), and historical IR (HIR). Depending upon the requirements from MN side, either CIR or HIR is used. So, IR is adapted as CIR or HIR. Both are stored as two different objects, CIR is broadcasted to MN and HIR is stored at BS. If MN is continuously connected and has not missed any of IRs then it can use CIR as IR. Otherwise, if it has missed some IRs of last BTs (from 1 to L intervals only, not more than L) then depending upon the requirements, HIR is unicasted to MN as IR to invalidate its cache of last BTs intervals. Hence, adaptive IR works in the proposal as per the needs of MNs. Figure 1 shows the cache invalidation with three locations in the proposed scheme. Location 1 is the place where all the cache contents are placed in the database repository, location 2 is the place of access point where some database for some pages can be placed, and finally, it can be placed in cache of each individual device MN as shown in Fig. 1.

To address the above discussed problems with respect to various challenges and constraints, in this paper, we propose an efficient cache invalidation scheme for mobile environments with stateless servers having hot data at any time interval. We have used a similar type of approach for identifying hot data as used in [15], but we have used a separate mechanism for checking update rate of data on server, so that BT for server can be regulated. The usage of adaptive size of IR reduces the bandwidth consumption in the proposed scheme. Moreover, we have sent IRs of hot data along with data itself, so it increases the hit ratio, and reduces the query response time. The performance of the proposed scheme

was analysed using extensive simulations which results a decrease in query latency, and bandwidth consumption.

The remainder of the paper is organized as follows. Section 2 discusses the most relavant related work done with similarity and difference to the current proposal. A detailed comparison of existing protocols with respect to various parameters is also described in this Section. Section 3 presents the network model and problem formulation. Then, in Sect. 4, a detailed description of the proposed scheme is provided in the text. An analytical analysis is described in Sect. 5. Simulation results, and discussion are provided in Sect. 6. Finally, conclusion and future insights are provided in Sect. 7.

2 Related work

There are a number of research proposals exist in literature addressing the issue of cache invalidation in wireless environments. Two of most common strategies for IR-based cache invalidation are object-based, and group-based. Various research proposals in these categories are described as follows.

Table 1 gives the comparison of the existing protocols discussed above with respect to various parameters.

Barbra *et al.* [7] proposed first broadcasting timestamp (TS)-based effective CIT technique. This strategy was the first object-based, stateless-server, synchronous IR-based invalidation strategy. In this strategy, at every time interval equal to L seconds, the server broadcasts an IR. Any IR_i at time T_i consists of a list of (Oid,T_d^r) pairs, where O_id is the identifier of the changed data item and T_d^r is the time stamp of the most recent data item changes, such that $Ti - wL < T_d^r < Ti$ holds. MN also maintains a variable T_{lb}, in which it stores the time of the last IR. In addition, for each cached data item O_i, it stores the items identifiers (O_{id}) and the timestamp for the item which was last invalidated (T_d^c) is kept. When MN receives the IR_i report at T_i, then depending upon state of MN, following possibilities may occur: Firstly, If MN has disconnected before the last broadcast time, i.e., $T_{lb}(T_{lb} < T_i - wL)$, then it drops its entire cache because the disconnection time is more than wL, and all the updates may not be covered in the IR_i. Secondly, when $T_{lb}! < T_i - wL$, i.e., disconnection interval is less than wL seconds, then MN starts invalidation process. For all cached data objects, it checks timestamp value with the IR_i timestamp value. If $T_d^r > T_d^c$, then data item is marked as invalid; otherwise, it is marked as valid. The main advantage of this approach was that the MNs can be disconnected for a short duration less than w sec and can reconnect again and can invalidate its cache at next IR broadcast. While It has a larger latency associated, because for query response, client has to wait till next IR which may generate extra congestion in the network. The MNs are forced to scan whole IR regardless of fact that all the cached data are included in report or not. When update rate increases, the report size increases significantly and consumes more bandwidth in this scheme.

Jing *et al.* [16] proposed a technique which was suitable where data update rate is less. It uses a bit sequence which is used to refer a data item. It uses very large IR (object based) and is hierarchical in nature, so consumes most of the bandwidth. It may provide effective validation but, there was a trade off between size & effectiveness in this scheme.

Cai *et al.* [17] proposed a group-based, synchronizes stateless-server IR-based cache invalidation strategy. It improves the communication cost by reducing the size of the IR. It works with disjoint set of data item in different groups identified by Gid. It divides the data items on the server into

Table 1 Comparison table of various existing schemes

Protocols	Report type	Report size	Server type	Disconnection time	False invalidation	Query-response	Bandwidth used
TS[7]	IR(OCI)	Fixed	Stateless	Small	No	Slow	More
BS [16]	Bit Sequence	Fixed-longer	Stateless	Longer	——	Common	Much
GIR [17]	GIR	Fixed-small	Stateless	Unlimited	Yes	Fast	Much
UIR [8]	IR+UIR	Fixed-longer	Stateless	Long	Yes	Common	More
SCI [18]	IR+GIR	Longer	Stateless	Longer	Yes	Fast	Much
CBSFI [11,12]	IR(OCI)	Small	Stateful	Unlimited	No	Fast	Little
RIH [19,20]	IR+UIR	Small	Stateless	Longer	No	Fast	Much
CBSLI [21,22]	IR(OCI)	Longer	Stateless	Longer	No	Fast	Little
SAS [9,23]	IR(OCI)	Adaptive-longer	Stateless	Longer	No	Fast	Little
ADD [24]	IR(OCI)	Adaptive-longer	Stateless	——	No	Very Fast	Little
DVD [25,26]	IR(OCI)	Longer	Stateless	Unlimited	No	Common	Much
CCI [31]	IR(OCI)	Longer	Stateful	Unlimited	No	Common	More
ACIT (proposed)	IR(OCI)	Small	Stateless	Longer	No	Fast	Little

disjoint units called data groups and each group has a unique identifier, G_{id}. At each time interval of L seconds, the server broadcasts GIR in which the reports entity is a data group id instead of a data item. $GIR_i = [G_{id}, TS_{gid}]$ where, TS_{gid} is the timestamp of the most recent update of the group. Discarding the contents are based on invalidation completely. It does selective tuning to reduce battery consumption. It has smaller GIR but has a many false invalidations so, may loose its effectiveness in wireless environment.

Cao *et al.* [8] introduced a new type of IR report generation scheme called as UIR. In UIR, the broadcast interval L is divided into (m−1) sub intervals ($m > 0$), and UIR is broadcasted after every L/m seconds. The server broadcasts an IR_i at T_i containing the history of changes in last wL seconds where, $w > 0$. IR is an OIR based scheme having data id and timestamp $< O_{id}, t^r_{id} >$, such that $T_i - wL < t^r_{id} < T_i$ holds. At time $T_{i,k}$, server broadcasts the k_th UIR report denoted by $UIR_{i,k}$ so, it has different broadcast times for UIRs. This UIR is a list of data items identifiers which have been changed after last broadcasted IR. With UIR, when a client has validated its cache then, it replies fast to the queries raised by other clients so, response time in this scheme is fast but, due to number of UIR along with IR broadcast, it consumes more bandwidth. Main disadvantage of this scheme is that if one IR is lost then, entire scheme collapses and suffers from false invalidations.

Cao *et al.* [11,12] presented a counter based state full cache invalidation (CBSFI) technique.It is a state-full scheme that uses a counter to identify the hot data. The server broadcasts the message to all the clients at updations of hot data items so that it informs new entries of IR to clients, in which clients pre-fetches the data that are most likely to be used in future. Server maintains a counter for each data item for each client. When client requests the data, the counter is increased by one and if counter reaches equal to or more than a threshold value then, that data is considered as hot data. Client has to inform the server when it will delete the data from cache, so that counter can be decreased by one. A cached item register (CIR) is associated in server with each client to maintain the state of the client. CIR stores the id of data items cached to handle server and client failure with disconnections. During hand offs, this information of client is transferred between two servers. This scheme has advantage that hot data changes are recorded in the IRs so, it becomes more effective but, on the other hand, it is difficult to maintain the cache state which generates extra overhead resulted in consumption of most of the resources in this environment.

Cai *et al.* [18] proposed a hybrid scheme called selective cache invalidation (SCI) which uses Object-based and group-based IR. It utilizes the benefits of both object and group-based IRs and was a synchronous and stateless server type scheme but, SCI size in this scheme becomes large in this scheme. Server broadcasts SCI after a fixed time interval which resulted more bandwidth consumption. Moreover, due to group invalidation, it suffers from large false invalidation of data but, due to selective invalidation, query response time can be faster in this scheme.

Chand *et al.* [19,20] proposed a technique which was an improvement over UIR. It was an enhancement of UIR scheme. While UIR was sending the list of requested pages after next IR but, RIH sends the list of requested pages by broadcasting from server in next UIR/IR whichever is earlier. If client is disconnected for some time then in UIR, it drops all the contents of cache of client but this scheme being a state-full approach keeps state of disconnected client so can validate the contents of the disconnected client as well.

Chin and Wu *et al.* [21,22] proposed a counter based stateless cache invalidation scheme (CBSLI). It was a stateless scheme, so there is no need to maintain cache state of clients but, it maintains a counter for the data item whenever a data item is requested by a client. This counter counts only the current requests of a data item during the current broadcast intervals and categorize these as hot or cold as per the threshold values. So, only hot data item is updated during a particular time interval. Clients may pre-fetch the hot data and will piggyback the id of that hot data with the next request of data to maintain a counter more accurately. For the next broadcast interval, counters are again reset to zero. Hot data is kept in IRs and is broad casted to clients which increases the effectiveness of this technique. This type of technique is very effective having smaller IR size and faster query response time.

Safa *et al.* [9,23] proposed a Hybrid technique called as selective adaptive sorted(SAS) Cache Invalidation Technique, which uses both type of IRs-object and group-based. It was an improvement over SCI [18] technique as size of IR increases but, in this scheme, IR was made adaptive and only on demand of disconnected MNs, historical part of IR is broad casted to MNs from BS. Only current broadcast time interval changes are broadcasted to MNs. MNs need to scan only recent updated pages in IR and it supports longer duration of disconnected MNs to validate their cache with the help of HIR stored at BS. Historical changes of interval wL to WL sec. are covered in this scheme and is used as the disconnection time for MNs. Due to its selective, adaptive and sorted nature, it takes lesser response time for the queries reply. This scheme reduces the size of IRs as it is dynamic so, less consumption of bandwidth occurs in this scheme but, when update rate is high on server and MNs are disconnected frequently, this scheme becomes unfavorable due to increase in size of IR and more bandwidth consumptions.

Chuang *et al.* [24] proposed an adaptive data dividing scheme (ADD). This scheme reduces the latency time for query replies. The proposal uses two approaches called as-Adaptive Broadcast Interval and Hot/Cold Query/Update. Adaptive Broadcast Interval divides the size of database into

three different ranges as lower, medium and higher. During one broadcast interval, if the numbers of updates are in lower range (lesser number of updates) then, broadcast interval is increased. When updates are in medium range then, current broadcast interval is suitable and when updates are in higher range then, broadcast interval is reduced to get the faster response time. In Hot/Cold Query/Update approach, Data is divided into 5 different categories as hot query (HQ), hot update (HU), cold query (CQ), cold update (CU) and remainder (RMA). According to this classification, each group of data is assigned a suitable broadcast interval. The server broadcasts 5 different IRs with different broadcast intervals with priorities $HQ > HU > RMA > CQ > CU$ but, it generates an extra overhead of keeping track of large categories of data items and their maintenance.

Fatima *et al.* [24–26] proposed a data validity defining scheme (DVD). This scheme is efficient in maintaining data validity after disconnection of MN voluntarily or non voluntarily. It was a client initiated cache invalidation scheme. Client sends starting and ending time of disconnections as timestamp of query to server and then server sends all updated data ids during this time interval so that MNs can validate their cache as per their requirements. There are some other techniques such as [27–30] for energy efficiency and disconnection handling in the literature.

Suno *et al.* proposed a Cooperative Cache Invalidation Technique [31]. CCI technique works by cooperation among different levels of components and neighbors. So, a variety of Cooperative schemes have been proposed such as-[32–36]. Lim et al. [31] proposed a Cooperative state-full approach for wireless network. Some other techniques such as [37,38] have been proposed for internet-based vehicular ad hoc networks (IVANETs). The architecture proposed by the authors has a server and agents for location management which works together in coordination for cache invalidation operations. The server and network agents maintain a list of data items such that vehicle that acts as mobile client can access the data item as per their requirements. As it is a state-full approach so, it can provide unlimited disconnection times for MTs. So, whenever a data item is updated, it sends an IR to a home agent (HA) rather than blindly and directly broadcasting the IR to multiple cells to increase the congestion. Then, the HA judiciously refines and re-distributes the IR to appropriate gateway foreign agent (GFA), where they can answer the validity of a queried data item, requested from a vehicle. Here, GFAs do not send the IR to an individual vehicle but, reply a vehicles validity request by on-demand basis. Since a vehicles location is implicitly maintained at both GFA and HA, the server can avoid keeping track of the vehicles current location, which is in fact a duplicated operation with the location management. Also, since a vehicle has a low probability of finding common data items in adjacent vehicles, broadcasting the same IRs to different vehicles is inefficient in IVANETs. So, whenever there is query on MN (Vehicle), it would be answered by validating the cache with IR multicasted by GFA so, query response time in this scheme is average but, it has considerably reduces the number of uplink requests to the server.

2.1 Gap analysis

As discussed in above section, CIT were widely investigated from different aspects and many research proposals have been reported in the literature but still following are the research gaps that needs further investigation:

(I) Only few caching algorithms are stateful and some are stateless. Being a stateful, there is an extra overhead of maintaining the cache and client state which continuously consumes more efforts. Also, stateless techniques are limited with the disconnection time of clients (wL seconds).
(II) Size of IRs is a major concern. Techniques reported in the literature have longer IRs which consequently consumes more bandwidth for broadcasting IRs. So, it should be minimized and if required can be increased for making effectiveness of IR and invalidating the cache completely.
(III) Most of techniques have covered the updates of data items of last wL seconds and broadcasting the report after every L seconds which results in checking of redundant updated pages resulted in increasing the response time for queries.
(IV) Almost all techniques have a fixed interval for broadcasting. So, when update rate is very high then due to same broadcast interval, size of IR may become large and limited bandwidth for mobile environments may become bottleneck for efficiency. Hence, these techniques can loose their effectiveness.
(V) It is observed that the size of IRs is fixed to carry updated data items, but recent research proposals used dynamic and adaptive IR reports to utilize them optimally in limited resources environments.
(VI) Only updates in data should not be the only criteria for being a content of IR. If data is much of use and is updated only then it should be included into IR, i.e, if hot data is updated then it must be included into IR while cold data can be ignored to include in IR which optimizes the size of IRs and keep the effectiveness of the techniques.
(VII) Techniques should include some mechanism to invalidate the cache of disconnected clients while actively or passively disconnected by some reasons.

3 System model

This section illustrates the network model used and problem definition.

3.1 Network and mobility model

A wireless mobile environment has been categorized into three groups [39,40] as follows:

- *Pure cellular* In this architecture, gateways/Access points are deployed at every intersection. It supports infrastructure- based services.
- *Pure ad hoc* It contains only MTs communication without any centralized control.
- *Hybrid* It is combination of pure ad hoc, and pure cellular.

In this paper, we have considered a hybrid architecture in which APs and BS are deployed in such a manner so as to connect MTs to clients in different network domains. APs and BSs are the part of distributed service set(DSS) protocol standard with IEEE 802.11, while MTs are included in basic service set (BSS). These two are statically connected to data server (S) as shown in Fig. 2. We also assumed that entire data resides on server and data updates take place on the server only. So, BS and AP facilitate MTs to communicate with clients in different network domains to access various services to exchange data. So, AP and BS act as gateway and router for MTs. The entire data locates at server and on updation of data, server periodically broadcasts an IR report down in network to BS, which in turn, sends IR to lower level device such as- AP to MTs. Finally, MTs fetch IR and validate their cache.

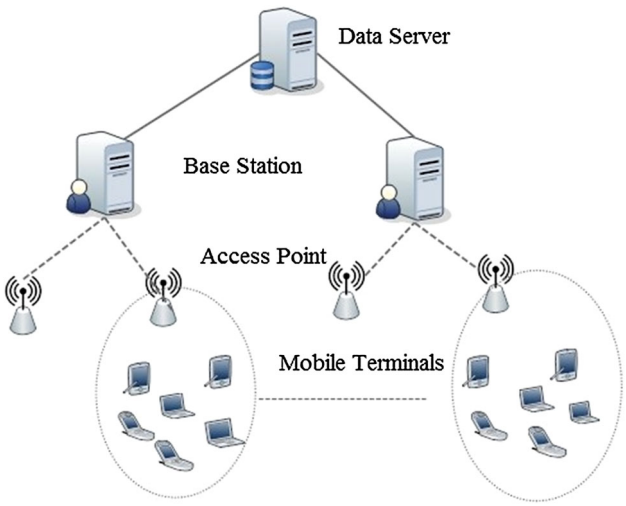

Fig. 2 Network model used

We have considered static components such as- server S, BSs and APs. Although MTs have mobility but, we have not considered any hand offs mechanism in the proposed scheme as same is provided by standard Mobile IPV6 (MIPV6) protocol, i.e., we have restricted MTs mobility in the range of their corresponding APs because, mobility is not a significant parameter for cache invalidation of MTs while taking assumption that MNs are continuously connected to their AP.

3.2 Problem formulation

There are various challenges for validating the cache consistency of MNs in a wireless environment. In a MN initiated cache invalidation approach, MN sends uplink requests to server to invalidate the cache. So, as the number of MNs increases, it generates more uplink requests results in generation of congestion to the uplink channel. For statefull server approach [10], exact state of clients cache is maintained on server. So, on any updation of data page on server, it sends an IR message to client so that it replies to the queries instantly. Hence, such a technique has better query response time but, it suffers considerable overhead of maintaining cache state on server so, lot of energy is consumed in this process. In state less server schemes [21], there is no extra overhead for maintaining exact state of client's cache as IR is broad casted to MNs. When it is broad casted asynchronously, then on every update of data, server broadcasts an IR to MNs which consumes most of bandwidth and keeps MNs active every time so, MNs consume more energy. Contrary to the above, when IR is broadcasted synchronously [7], then after fixed time interval, all updations are sent to MNs through IR so, for small duration between these events, MNs can sleep and save power consumption. But, IR sizes became large if update rate of data is more so, larger IR consumes more downlink bandwidth. Most of the group-based Invalidation schemes suffer from false invalidation of cache contents as they are having larger size of their GIRs which consume more bandwidth. In counter-based stateless schemes [12], IR was effective in terms of invalidating cache contents, as it uses hot updated data only in IR but, their broadcast time was fixed, i.e., they need to be adaptive depending upon the update rate of data. So, for cache invalidation scheme, IR need to be of appropriate size, with less consumption of bandwidth so that query delay can be minimized. For minimizing average query response time (AQRT), we proposed an objective function alongwith constraints defined in Eq.1 as follows.

$$Minimize\{AQRT\}$$
$$SubjectTo:$$

$$\begin{cases} P_{hit} + P_{miss} = 1 \\ P_{hit} \geq P_{miss} \geq 0 \\ 0 < C_{th} \leq Count_{dx} \\ 1 \leq CHR \leq 2^{\frac{w}{2}-1} \end{cases} \quad (1)$$

Table 2 Symbols and their meanings

Symbols	Meaning
P_{hit}	Probability of finding page in cache
P_{miss}	Probability of page missing from cache
IR	Invalidation report
D	Total data on server
H	Hot data %age
C	Cold data %age
d_x	Data page Id
$Page(d_x)$	Data page d_x
CoA_{MN}	Care of address of mobie node
CoA_{AP}	Care of address of AP
CoA_S	Care of address of Server
ts	Time stamp
t_{bt}	Length of current broadcast interval
w	No of broadcast intervals for carrying history
ts_{curr}	Current time stamp
U_{rate}	Update rate of current t_{bt}
$Count_{dx}$	Counter register associated with dx
U_{th}	Update rate threshold value
C_{th}	Counter threshold value
CHR	Control history register of $w/2$ bits
l_{bt}	Last broadcast time
MB	Working mode bit. (0/1: Normal/Fast)
DRP	Data request packet
$Cache_{MN}$	Cache of MN
HIR	History invalidation report
L	Normal mode broadcast time
wL	w Broadcast times
t_q, t_u	Time of query and update
QP_s	Query Processing delay time at server
QP_{bs}	Query Processing delay time at BS
QP_{ap}	Query Processing delay time at AP
QP_{mn}	Query Processing delay time at MN
C_{size}	Cache size on MN
Q_{rate}	Query rate

Where, $count_{dx}$ is greater that and equal to threshold counter value of count and CHR is less than $2^{\frac{w}{2}-1}$ byconsidering higher hit ratio with respect to miss ratio. Table 2 describes various symbols and their meaning used in this paper.

4 Proposed work

Based upon above issues and challenges, this section illustrates the proposed solution to the problems discussed above. In the architecture discussed in Sect. 3, every MN has cache memory with it. A Server has all the data D so that all updates of data items are done only on the server. Server is a stateless and synchronous in nature. So, a hybrid cooperative scheme is proposed with adaptive IRs and BT depending upon the update rate and hotness of the data items on server. Hot data are identified using similar kind of technique as proposed in [15], which in turn, provide faster query response time with less bandwidth consumption. The proposed scheme also has disconnection time of 0 to wL sec., i.e., upto w BT intervals. Proposed scheme improves bandwidth utilization by reducing the report size and the number of uplink requests sent to the server. The report size is made dynamic and adaptive as per the needs of a MN by using the hotness of the data items on the server. Moreover, the proposed scheme overcomes with the problems of more bandwidth usage and high query response time of UIR and SAS schemes. The detailed description of the proposed scheme is illustrated in the following subsections.

4.1 Design and structure

For maintaining the hotness of data on server, we have used a counter register with every data $count_{dx}$ for counting number of accesses of d_x in t_{bt}. So, we will identify the hotness of data using similar kind of technique used in [15]. A counter is maintained to check that how many requests for a data item have been received at the server in last unit time interval. Then, control history register of w bits is maintained to record historical hotness of data. Then, IRs of only that hot updated data is created along with data pages d_x for t_{bt} which is sorted as per time stamps. Similarly, report of last $(w-1)$ BT intervals is maintained without data pages. So IR is divided in to two parts as- history invalidation report (HIR) of last (w-1) BT and current IR (CIR) of t_{bt} sec. This IR is sent to BSs, which divides this IR in two parts of HIR and CIR. Then, HIR is kept with BS and CIR is sent down to MNs using AP.

4.2 Detailed description

4.2.1 Hot data maintenance

On the server side, data hotness is maintained which is accessed frequently by many MNs. So, we have identified the hotness of data in the proposed scheme similar to the technique proposed in [15]. Hotness of the data is decided using two factors- how many requests for a data item have arrived at the server in last BT interval and recent history of hotness of data. we define the following.

– *Threshold Counter* $count_{dx}$ value is compared with a counter threshold value c_{th}. When any data's counter value is larger or equals to c_{th} then, it is included in

current hot data list, i.e., if ($count_{dx} >= C_{th}$) then data item is considered as hot.

– *Historical Recency* Data item remains hot during recent past w/2 intervals. As the hotness of data remains hot for some period, so we have used control history register (CHR) of length w bits to keep track of hotness of d_x of last w/2 BT intervals. So, if data was hot recently, it is kept in hot category. In CHR, every bit shows that it was hot (1) during that broadcast interval or not (0). So, if it was hot during recent half of time interval, we give that data a second chance to become hot, and push a 0 bit in its history control register and use a mask of 0000011111 to check about recent hotness in CHR. If after masking, we get a value between 1 to 31 then, we keep data in hot list for t_{bt}. Hence, a less value indicates more recently data item listed in hot data list. It helps to keep the recency (Locality of reference) for the hot data item. We illustrate this technique using following two examples.

Example 1 Consider a data counter is not upto threshold value with w=10 units, L is 20 s. and CHR is 1111010000. Check if the hot data list or not?

Sol: Mask of 0000011111 is used to check about the recent hotness of the data item. CHR will be masked by mask register value as: 1111010000&0000011111 = 0000010000, it is numeric 16, which is less than 31 so, we will keep this data in hot list and push another 0 in CHR by shifting it towards left. Now, for the next BT if d_x is not frequently accessed then, it will be out of hot data list and as a result after masking, the value is not between 1 to 31, i.e., there is no recent hotness of the data item.

Example 2 Consider a data counter is not upto threshold value with w=10 units, L is 20 s. and CHR is 1111100001. Check if data will be included in hot data list or not?

Sol: Mask of 0000011111 is used to check about the recent hotness of the data item. CHR is masked by mask register value as. 1111100001&0000011111 = 0000000001, it is numeric 1 which is less than 31 so, it is kept in hot data list and push another 1 in CHR by shifting it towards left. As value is very close to 1 so, this is very recent hot data. Hence, it has more chances to be included into hot data list.

4.2.2 Adaptive IR generation

IR records data ids of hot updated data only for a time period of current BT, i.e., t_{bt}. So, for time $(w-1)t_{bt}$, the data ids of such hot data with their time stamps are also included in IR. It keeps historical updations of data on server and also helps disconnected clients to validate their cache upto $w * (t_{bt})$ times. This IR is broadcasted to BS. Because IR has only hot data of interval of t_{bt} sec. so, size of IR is optimal and broadcasting it to BS will not consume more bandwidth.

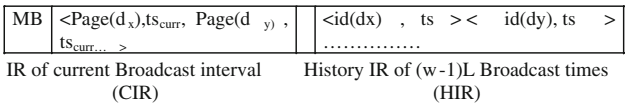

Fig. 3 Invalidation report (CIR+HIR)

IR will contain information in 2 blocks- first block stores information of current BT interval t_{bt}. This block contains information such as- Data Ids of hot updated data, original $page(d_x)$ and time stamp. On the Server, update rate U_{rate} of data pages is tracked which is used to set the mode of operation of scheme using value of mode bit (MB) as 1 or 0. In second block, data ids of last (w−1) BT, hot data ids with their timestamps is stored. These ids are sorted using their timestamps values, so that while checking of most recent changes at client side, only unchecked and recent missed IR information can be checked firstly. When time-stamp of MN exceeds time-stamp of data items of IR, it will not check that further so, it is sorted. This sorting of data ids in IR reduces the validation time to large extent.

The data of Block one is broadcasted to MNs because, it is the hot updated data from the server and is required by MNs in near future. Then, uplink and downlink requests are generated for it. So it is beneficial for MN, if server is sending some updated hot data and which is invalid in local cache of a MN then, MN prefetches that data for future uses, which makes the downlink bandwidth effective and reduces future uplink & downlink requests for that hot data.

Also, IR is divided in two parts on BS as- Current Invalidation Report (CIR) ($wL - t_{bt} < ts <= wL$), and History Invalidation Report (HIR) ($0 < ts < wL - t_{bt}$). Then BS sends first block as IR to MNs and real hot updated data page(d_x) to MNs. Its IR is shown in Fig. 3.

4.2.3 Adaptive BT - fast, or normal mode operations

As server keeps track of update rate of current BT interval as U_{rate}, so, a threshold value for update rate, U_{th} is kept. After this update rate,BT is regulated to smaller intervals and MB is set to Fast Mode. If update rate is higher then, smaller BT is kept; otherwise, earlier BT works fine, i.e., L works by setting mode bit as 0(NORMAL) to keep smaller query delay. Due to this adaptive BT, the tuning size of IR will not increase considerably and remains optimal.

Following advantages can be achieved using this strategy:

– Tuning of BT on update rate gives smaller Size of IR.
– Congestion on the network reduces during high update rates.
– Query reply is faster as BT is reduced.

4.2.4 Disconnection MN support

As MNs are limited with the battery life so, an energy efficient scheme is required to support the disconnected operation. For energy efficiency, MN can disconnect actively (Voluntarily) or passively (Due to some failures such as link failure).

For active disconnections upto t_{bt} An actively disconnected MN reconnects with AP can have its first query for reply, so, it listens to CIR from AP and current updated hot data list and validates some portion of its cache and prefetches that data.

For Passively Disconnected for Longer durations up to wL secs. As data history of last w broadcast times are recorded and is broadcasted at BS so, if any MN has missed some IR in sequence that can demand for History IR from AP which in turn, asks the same from BS and validates the cache of reconnected active MN. When a MN is disconnected it records its disconnection time and its reconnection time on AP. So, after reconnection when MN makes first query for data then, it listens to CIR from AP and current updated hot data list to validate some portion of its cache. After that, it prefetches that hot update data and asks for validating its cache by sending its start and end time of disconnection from BS along with the requested data if not validate yet. Then, server sends all the recorded changes of $wL - t_{bt}$ interval, to MN using which MN validates its cache and serves for subsequent requests.

To show the flow of sequences of various activities in the proposed scheme, the algorithms for network components such as server in Algorithm 1, Base station in Algorithm 2, Access Point in Algorithm 3 and Mobile Node in Algorithm 4 are illustrated below. The flowcharts of server is presented in Fig. 4.Then to understand cache validation process another flowchart is created that depicts validation process in Fig. 5 and finally query reply flowchart is given in Fig. 6.

Algorithm 1 describes the complete behavior and working of Server. As input in line number 1, server requires data like MB bit, BT and BT window w. Then, server generates IR and response of any requested data. From line no 3–5, the working and setting up of MB is shown. As depending upon $U_{rate} >= U_{th}$, the mode bit is set (Fast Mode) or reset (Slow/Normal Mode). Then, lines 6–11 describe how hotness of data is checked along with accounting of recency of data hotness. Lines 6 and 7 are used for checking current hotness based on threshold value of counter and lines from 8–11 are used for checking recency of hotness upto $w/2$ last intervals. In lines 12–13, IR is generated and in 14–16, Data request is full filled and sent to BS.

Algorithm 2 describes the working of BS component in ACIT technique. From lines 3–6, Bs is setting t_{bt} as MB and judiciously dividing IR into CIR and HIR. Then, BS sends CIR with data and keeps HIR for disconnected MNs, who have missed some recent CIRs. Lines 7 and 8 show history request packet demands and response by BS to AP. Then, in lines 9–11 request for query data handling from AP is depicted and is sent to upper level to server. In lines 12 and 13, response to requested data page is sent via AP.

Algorithm 1 Server side sequence of operations

1: **Input**$t_{bt} = \{L, 1\}$, For Normal Mode MB=0 and Fast Mode MB=1, $L > 1$ //Working Mode Bit w: Broadcast Time Window & $w = n * t_{bt}$. Server Keeps Track of Updation of Data and Update Rate U_{rate} of current t_{bt} & counter of data items.
2: **Output:** IR and Requested Data Page.
3: **A.** Setting Mode of Operation
   $U_{rate} >= U_{th}$
4: Set MB=1 & $t_{bt} = 1$ //Fast Mode- Adaptive BT
5: $MB = 0$ & $t_{bt} = L$ //Slow Mode
6: **B.** On Updations, Checking for Hot Data.
   d_x is Updated $count(d_x) >= C_{th}$
7: Include in CIR as (d_x, t_x^r)
8: Maintain CHR by pushing 1 //Checks its CHR Value $((CHR <= w/2) AND (CHR > 0))$
9: Include it in CIR as (d_x, t_x^r)
10: Maintain CHR by pushing 0.
11: Push 0 in CHR.
12: **C.** IR Generation
13: Generate IR after t_{bt} & send to BS with data pages.
14: **D.** Data Request
15: Receive Data Page Request
16: Send Data Page to BS.

Algorithm 2 Base Station side sequences of operations

1: **Input:** IR from Server, DRP from AP
2: **Output:** Requested Data Page, request to server
3: **A.**Receive IR from Server.
4: Set its t_{bt} as MB bit.
5: Separate its CIR & HIR
6: Broadcast CIR with d_x & pages to APs.
7: **B.** Receives HRP request from AP
8: Sends HIR to AP
9: **C.** Receives DRP from AP as $\{< d_x, CoA_m >, CoA_{AP}\}$
   Page Found
10: Replies with Data Page.
11: Forward Request to Server as $\{<< d_x, CoA_m >, CoA_{AP} >, CoA_{BS}\}$
12: **D.**Receives Data Page from Server as $< page, CoA_m >$
13: Sends it to AP.

Algorithm 3 Access Point Side sequences of operations

1: **Input:** CIR, Request to server
2: **Output:** HRP,Data Page
3: **A.** Receives CIR from BS
4: Broadcast IR to MNs
5: **B.** Receives HRP request from MN
6: Forwards HRP to BS
7: **C.** Receives HIR from BS
8: Sends/Unicasts HIR to MN
9: **D.** Receives DRP from MN
10: Sends Request to BS
11: **E.** Receives Data Pages from BS
12: Sends it to MN.

Algorithm 3 describes the steps that AP takes. In lines 3 and 4, CIR is received on AP and is broadcasted by AP to MN. In lines 5 and 6, request for HRP is handled. In lines 7 & 8, a HIR is received from BS and it is sent to MN. Then, in lines 9 and 10, received request for data from MN is forwarded to BS. Lastly in lines 11 and 12, received data is sent to MN.

Algorithm 4 Mobile Node Side

1: **Input:** Query, CIR
2: **Output:** DRP, Query Reply.
3: **A.** MN connects to AP at time t_i $(wL - L <= t_i <= wL)$//Active Disconnection
4: Wait for next CIR & receive pages $(1 <= t_i < wL - L)$//Missed some IR
5: Call HIR from AP //Passive Disconnections
6: Flush Cache Contents & Receive IR with Data.
7: **B.** Query generated for d_x $(d_x \in Cache_{MN})$ //Hit Occurs
8: Reply with $Page(d_x)$
9: Miss Occurs
10: Generate DRP (d_x, CoA_{MN}) & Send to AP.
11: **C.** Receives Data Packet from AP & Replies Query
12: **D.** Receives HIR from AP
13: Validate Its Cache.

Algorithm 4 illustrates the detailed working of Mobile Nodes. MN query and CIR are input for which it works and looks for data request packet or query reply. In lines 3–6, there is a check on reconnection of MN at t_i to confirm that whether it was active disconnection or passive dis connection in case MN has missed some recent IRs broad casted by server. In lines 7–10, steps for query handling are shown. If hit occurs with a rely and miss ,then generated DRP is forwarded to AP. Lines 11–13 show that on arrival of requested data from server, a send request is replied with respect to the generated query from the clients. Moreover, for passive disconnection a HIR is received from BS for validation of cache.

5 Analytical model

In this Section, we analyze the proposed cache invalidation scheme with respect to various parameters such as-average query response time (AQRT) which consists of two types of time delays as- id time spent during hit occurrences and time spent during miss events. Here, the arrival of queries on MN and updation of data on server are considered as Poisson's processes while, their inter-arrival time are taken as exponentially distributed. A set of notations used are listed in Table 2.

5.1 Average query response time

Whenever a query is generated on MN, MN accesses IR for cache validation. The data may be in cache and may be validated by IR received via BS, AP from server. So, let probability of data item to be cached is P_{cache} and probability of data to be updated during t_{bt} be P_{up}. Then, consider P_{hit} as the probability of cached data as valid and P_{miss} as probability of cached data as invalid. So, $P_{hit} = (1 - P_{miss})$. Hence, we define AQRT using Eq. 2 as follows.

$$AQRT = P_{hit} * (TimeSpent1) + P_{miss} * (TimeSpent2) \quad (2)$$

We also assume that total data on the server is D, out of which H % is hot and C % is cold data, i.e., $C = (100 - H)$

Let us assume that cache size is C_{size} % of D. For any data selected from hot set or cold data set, we have probabilities as P_{hot} and P_{cold} where, $P_{hot} = (1 - P_{cold})$ holds. Then probability of data item cached in MN is represented as:

$$P_{cache} = P_{hot} * \left(\left(\frac{C_{size}}{H}\right) * P_{hot}\right) + (1 - P_{hot}) * \left(\left(\frac{C_{size}}{C}\right) * (1 - P_{hot})\right) \quad (3)$$

When the cached data is queried after it is being updated then, it is a invalid data item. Then probability of data item being updated during t_{bt} to query arrival time where, $t_u < t_q$ is given as:

$$P_{up} = \int_{t_q=0}^{\infty} \int_{t_u=0}^{t_q} \mathrm{f(t_q)} * \mathrm{g(t_u)}\, \mathrm{d}(t_u)\mathrm{d}(t_q) \quad (4)$$

$$P_{up} = \frac{U_{rate}}{(Q_{rate} + U_{rate})} \quad (5)$$

So P_{hit} can be represented in terms of P_{cache} and P_{up} as given below.

$$P_{hit} = P_{cache} * (1 - P_{up}) \quad (6)$$

We can write P_{hit} by Eq. 7 as follows.

$$P_{hit} = \left(P_{hot} * \left(\left(\frac{C_{size}}{H}\right) * P_{hot}\right) + (1 - P_{hot}) * \left(\left(\frac{C_{size}}{C}\right) * (1 - P_{hot})\right)\right) * \left(\frac{Q_{rate}}{(Q_{rate} + U_{rate})}\right) \quad (7)$$

Now, as per Eqs. 3,5,& 7, P_{hit}, and P_{miss} have been computed as above. In Eq. 2, we have two time components as-the time delay during query reply when hit has occurred on MN cache so, it includes only query processing time of MN, i.e., QP_{mn}. Hence,

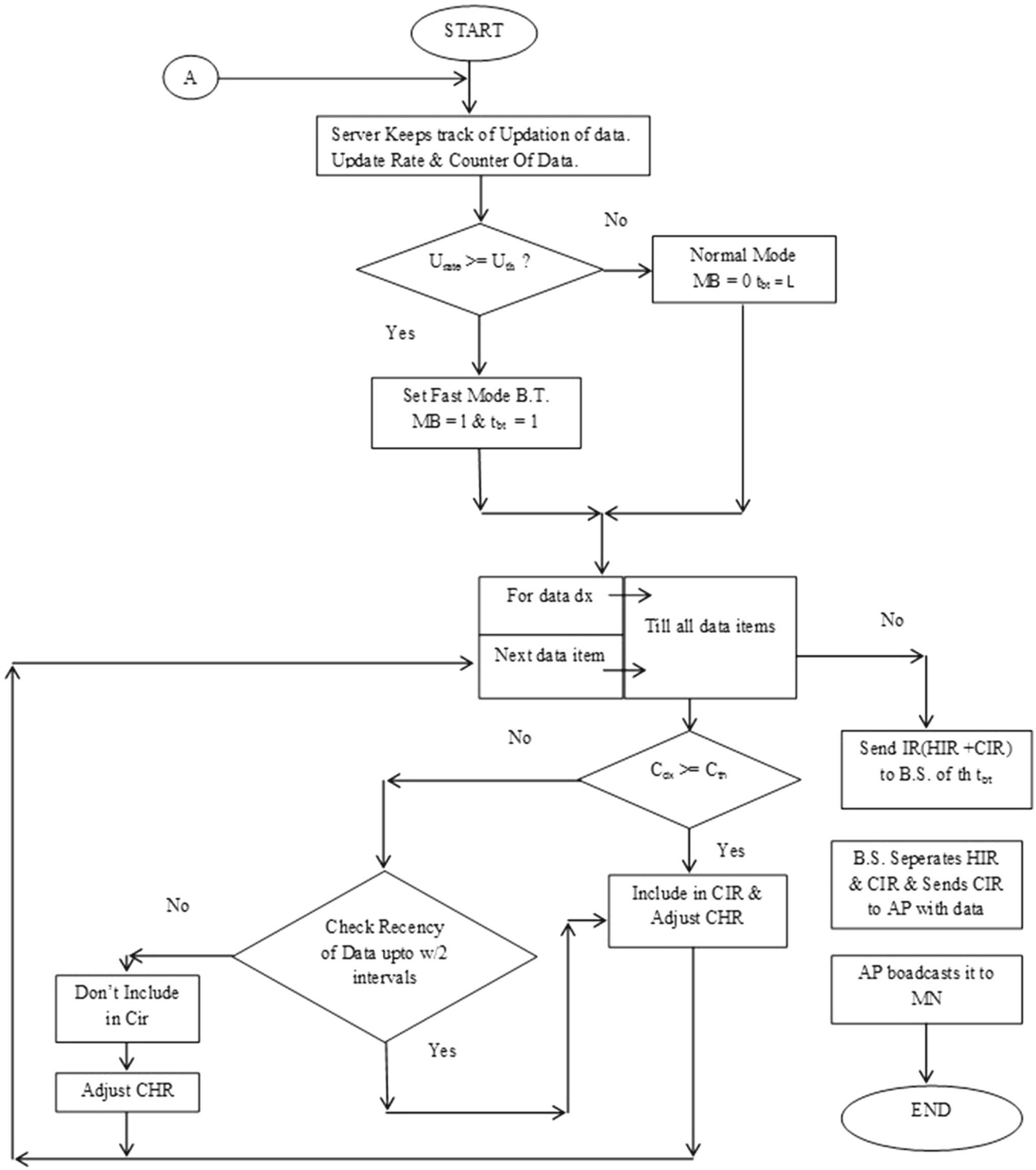

Fig. 4 Server flowchart

$$TimeSpent1 = QP_{mn} \quad (8)$$

The second component is the time delay occurred during query reply when a miss for data has occurred. So, request for data is made to server via AP and BS. Hence, requested data page is brought back to MN via same components and the time spent is represented as follows.

TimeSpent2 = (Request Sent on Server Via AP and BS) + (Response with requested Page via AP and BS).

$TimeSpent2 = (QP_{mn} + QP_{ap} + QP_{bs} + QP_s) + n * (QP_{mn} + QP_{ap} + QP_{bs} + QP_s)$ Hence, we can write

$$TimeSpent2 = (n+1) * (QP_{mn} + QP_{ap} + QP_{bs} + QP_s) \quad (9)$$

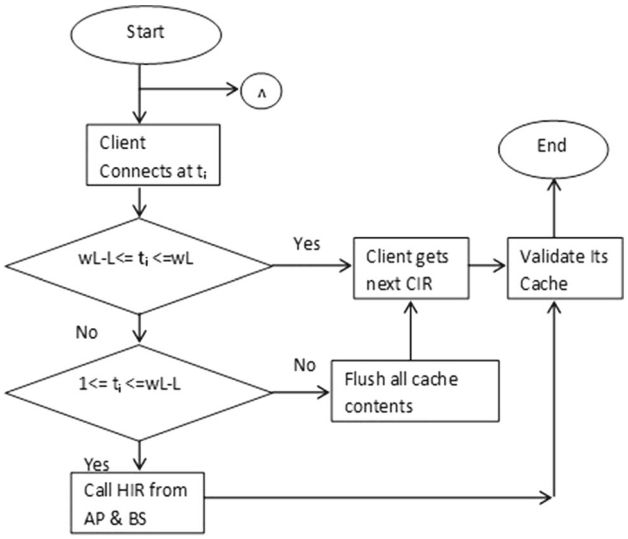

Fig. 5 Cache validation flowchart

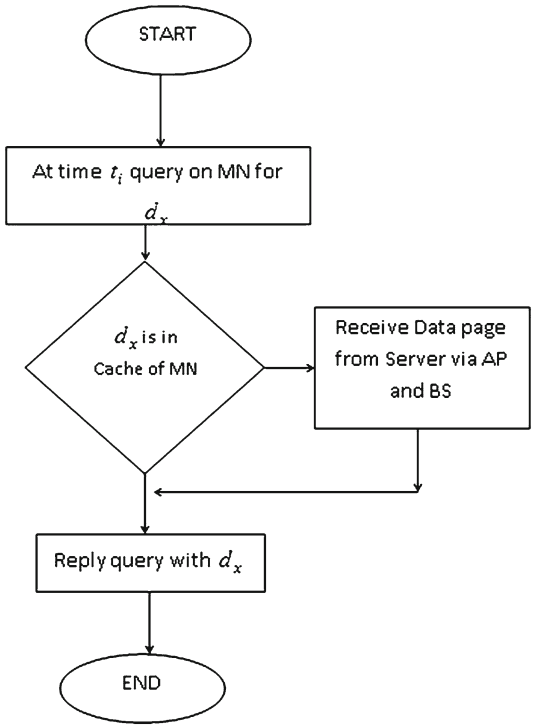

Fig. 6 Query reply flowchart

Using Eqs. 2, 6, 8 and 9, we can write equation for AQRT as given below.

$$AQRT = (P_{cache} * (1 - P_{up}) * QP_{mn}) + ((1 - P_{cache}) * (1 - P_{up})) * ((n + 1) * (QP_{mn} + QP_{ap} + QP_{bs} + QP_s)) \quad (10)$$

Figures 4 , 5, and 6 show flow of sequences of activities at the server side, for cache validation, and query reply.

6 Performance evaluation

6.1 Simulation settings

To evaluate the performance of the proposed scheme, we have used the discrete event simulator ns_2 [41,42]. We used considered a scenario consisting of wired and wireless connection with hierarchical addressing. We have taken 10 MNs, 2 APs, 2 BS and 1 server with all data items, in network topology considered. In our simulation model as shown in Fig. 7, we have neither considered mobility nor location management of MNs.

We have kept the client with default configurations having cache, cache manager, and query generator modules as shown above. Broadcast Manager, Query, uplink Queue use their basic functionality such as broadcast manager broadcasts the IR, query module uses MN query to be answered from server, and uplink queue contain all the up ward traffic from lower components towards server . When a cache is full, Least recent cache replacement algorithm is used instead of any advanced replacement techniques [43]. We have deployed few more modules such as- *update manager module, history register manager*, and *hot data selector* for IR to enhance the functionality of server. Following are the functionalities of different components included in the simulation model.

- *Update manager* It keeps track of update rate of data pages and then matches this rate with a threshold value.
- *History register manager* To keep track of history register for every data item and manages it for every t_{bt}.
- *Hot data selector* Selects only hot updated data to be included in IR.

Depending upon MB value, t_{bt} is set for next broadcast time. Then, IR is generated and broadcasted to MN. Depending upon disconnection time of MN, a hit or a miss of data item occurs. A miss occurs if data item is not found in cache or data item is marked as invalid in cache. Then, data item request is sent on an up link and brought from server to cache. The simulation parameters used in the proposed scheme are listed in Table 3.

6.2 Results and discussion

To check the effectiveness of the proposed scheme, it was compared with other state-of-the-art schemes such as- SAS [9], and UIR [8]. The performance of these three schemes were evaluated and compared using several metrics such as- average query response time, IR size, and uplink requests. Each metric is measured by varying parameters such as object update rate, and query arrival rate. The results obtained are illustrated as follows.

Fig. 7 Simulation model used

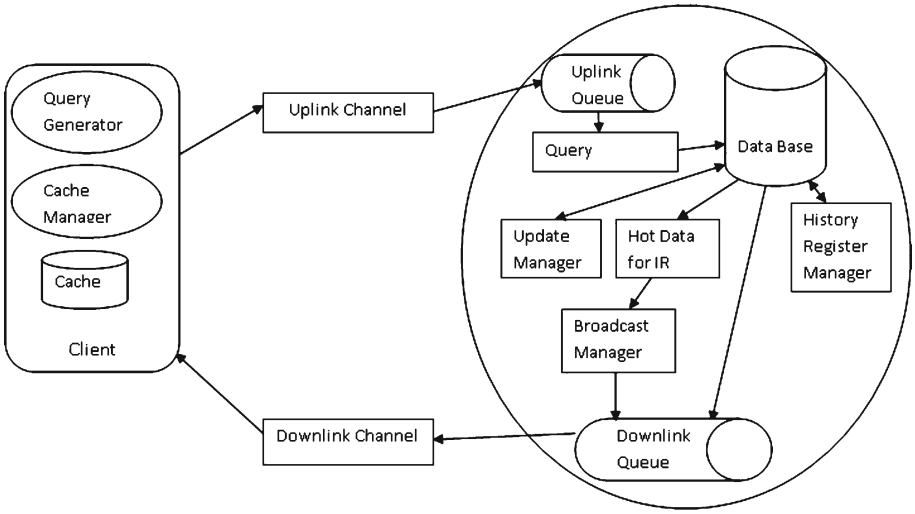

Table 3 Parameters used and their values

Parameter	Default value
Number of mobile nodes, n	10
Database size, D	1000 data items
Cache size, C_{size}	200
Broadcast window size, w	10
Broadcast time, L	20 s
UIR times, m	4
Data id, d_x	32 bits
Data size	2048 bits
Time stamp size	64 bits
Query arrival rate (Poisson's Dist.), Q_{rate}	2 queries/s
Object update rate(Poisson's Dist.), U_{rate}	5 objects/s
Distribution of query arrival times	exponential
Distribution of update arrival times	exponential
%age of hot data items, H	10 % of D
% age of cold data items, C	90 % of D (Remainder)
Topography	670 × 670 m
Simulation time	1000 s
Routing protocol	DSDV
No of BS	1
No of AP	2
No of server	1

6.2.1 Effect of object update rate

When object update rate increases on server, then cache miss ratio also increases on MN side because most of the requested data items stored in the MNs cache are invalid. Also, the number of uplink requests sent to server increases, which results an increase in the average query response time and size of IR reports. Hence, there is an increase in bandwidth consumption. This occurs in both UIR and SAS techniques but, in our proposed scheme, we have update manager module

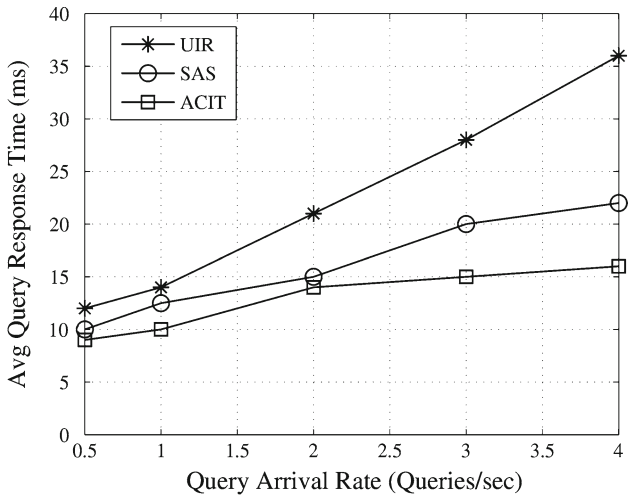

Fig. 8 Variation of average query response time with object update rate

on server which keeps track of updation data rate and set mode bit as MB = 0(for normal operation) and MB = 1(for fast operation). In the proposed scheme, IR is adaptive as it is having only hot data inside it so, size of IR is not increased considerably because, all updated data are not included in IR. Depending upon MB status, broadcast time of proposed scheme is set. Hence, during higher update rate, IR is generated and broad casted after shorter interval of time which results in small query response time. Only updated hot data pages are sent to MN so more hits occurs and fast query response time is achieved. Where HIR is stored on BS which can help disconnected MNs to validate their cache beyond the BT boundaries by making a demand from BS for HIR. The uplink requests are sent only for cold data query requests with reduced frequency in the proposed scheme which results a reduction in uplink requests.

– *Impact on average query response time-(AQRT)* Fig. 8 shows the variation of an average query response Time

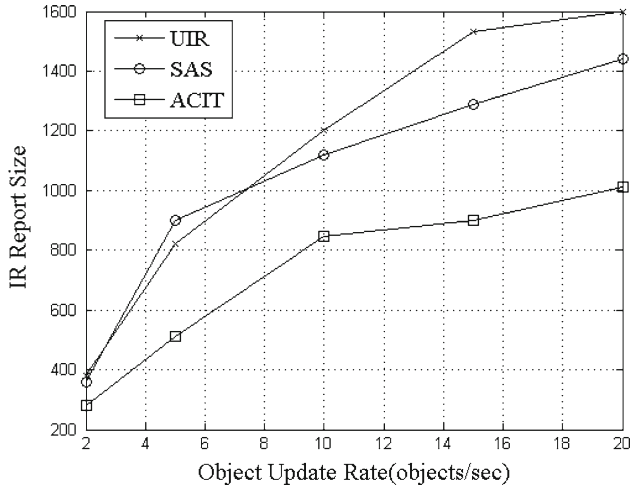

Fig. 9 Variation of size of IR using object update rate

with Object Update rate of UIR, SAS and adaptive cache invalidation technique (ACIT). All the schemes have lower AQRT for smaller values of object update rate. But, it increases with an increase in Object Update rate in all the schemes. AQRT increases sharply in case of UIR because, with such larger update rates, the data in cache becomes invalid and more uplink requests are sent. In SAS, with higher update rates, the AQRT becomes larger due to longer IRs and HIR data. But, in ACIT, the variation of AQRT is not sharp as it has fast travel of IRs to MNs so, queries can be replied instantly with updated page as data is sent along with IRs.

– *Impact on IR report size* Fig. 9 shows an impact of IR size on server. As size of IR becomes large with higher update rates then, there is large consumption of network resources such as bandwidth and memory. In SAS, carrying history of updates of data in IR, to facilitate disconnected MNs, makes size of IR larger which resulted a large consumption of resources of the network.Same is case with UIR, as a number of UIR reports are generated along with IR report which significantly increases the size of IR report. But, in ACIT, only updates of hot data items are kept in IRs along with small historical updates information which resulted a smaller size of IRs. Moreover, when update rate goes higher from threshold update rate then, due to fast mode working, IRs are broadcasted faster so, query access delay is reduced in the proposed scheme.
– *Impact on no of uplink requests* In Fig. 10, with higher update rates, data in cache becomes invalid in case of UIR and SAS. So, more up-link requests are sent on server to validate a cache data. But, in case of ACIT, as update rate becomes higher, its BT is adapted and IRs reached at MN faster so, cache is invalidated and mostly hot data is pre-fetched by MN so, lesser uplink requests are sent to the nodes in the hierarchy.

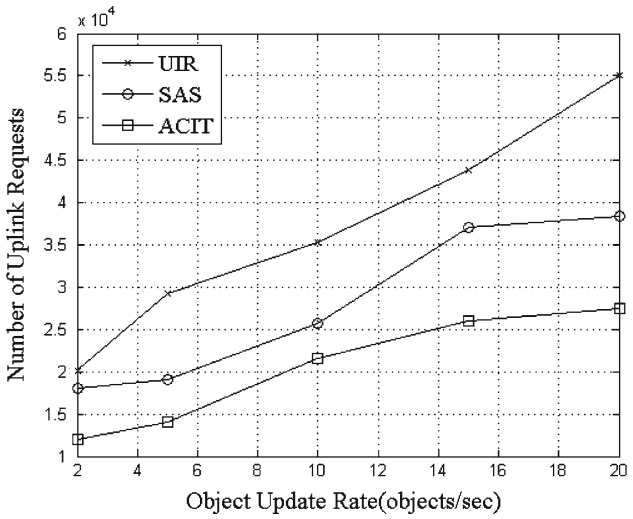

Fig. 10 Number of uplink request using object update rate

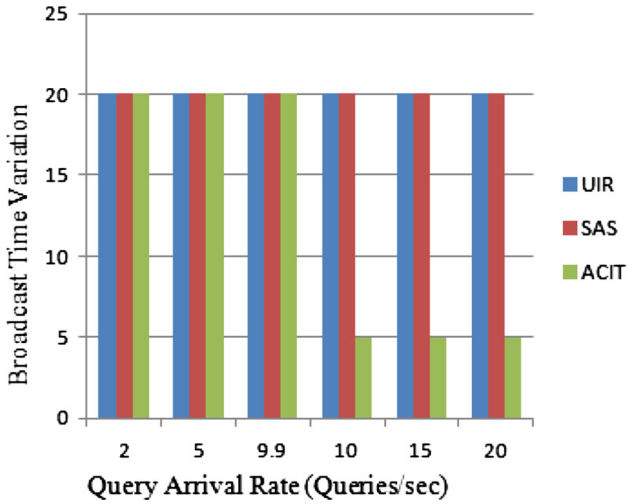

Fig. 11 Variation of BT using data update rate

– *Impact on broadcast times* As shown in Fig 11, BTs are fixed in UIR and SAS schemes but, it is adaptive in ACIT. BT is more when update rate is small and becomes smaller when update rate becomes more. This adaptive nature of of BT helps the proposed scheme to carry smaller IRs with higher update rates.

6.2.2 *Effect of query arrival rate*

To evaluate the performance of the proposed scheme, we have varied the query arrival rate using metrics such as AQRT and Uplink Requests. Results obtained are discussed as follows.

– *Impact on average query response time* Fig. 12 shows impact of query arrival rate on AQRT of UIR, SAS and ACIT strategies. Query arrival rate is varied from

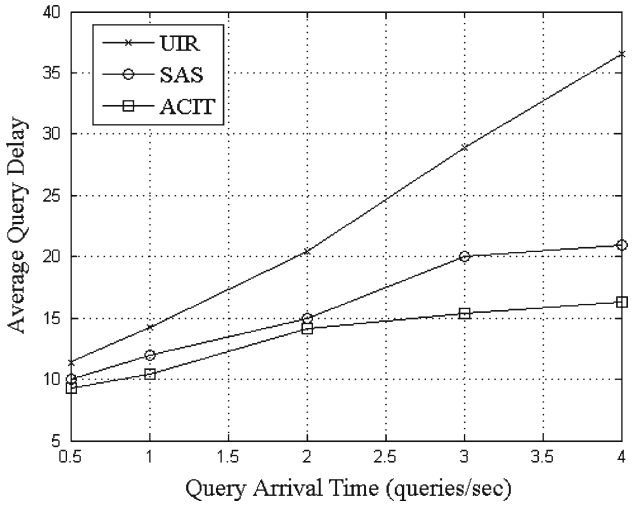

Fig. 12 Average query response time with respect to query arrival rate

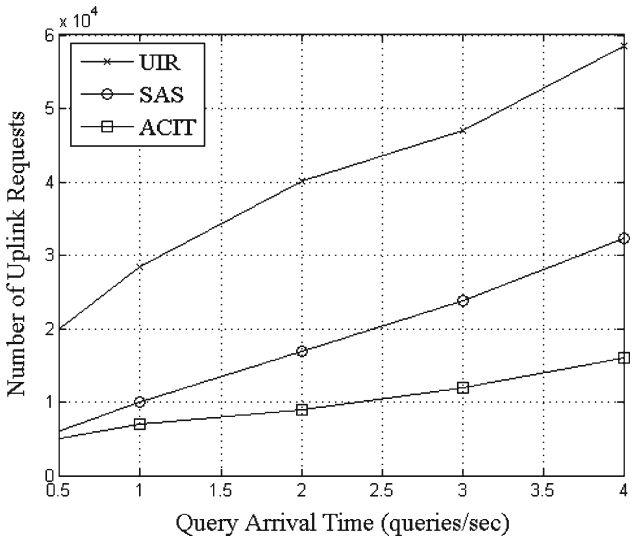

Fig. 13 No of up link request with respect to query arrival rate rate

2 queries per sec. to 4 queries/sec. for each MN. Then, their AQRT is recorded and analyzed. For lower query arrival rates upto 1 query per sec, AQRT of UIR and SAS increases, but due to hotness of the data item, queries are replied by cache copy of data on MN so, smaller query response time occurs in the proposed scheme. As there are less miss in ACIT so, less AQRT achieved with more query arrival rates in the proposed scheme. With lower query arrival rates, the probability of requesting a cached data which is updated on server becomes very less. When query arrival rate reaches 2 queries/sec. then, performance of UIR degrades due to more misses so, queries are replied from server by an uplink request. For SAS, queried data is looked with large IRs and histories so, more time is consumed in getting the reply for a query. Hence, ACIT has outperformed other two schemes as, most of queries are answered from local cache which results a less query response time in the proposed scheme.

– *Impact on uplink Requests* Fig.13 shows the uplink requests increases as the number of queries increases because, there is very less probability of queried data to be found in cache and being validated in UIR and SAS schemes. But, in ACIT, there are higher chances of locating the queried data in cache because, only hot data is kept in IRs and in cache. So, for UIR and SAS, as the need of queried data is raised, number of misses occurs more frequently. So, more uplink requests from MNs are sent for getting the requested data from the server. Therefore, more number of uplink requests in UIR and SAS are generated but, in ACIT number of uplink requests are very less which results an enhancement in the system performance.

Table 4 shows the percentage improvement of proposed scheme in comparison to UIR and SAS schemes with respect to parameters AQRT, IR size, Uplink requests and broadcast times.

Table 4 Percentage improvement of ACIT over other existing schemes

Metrics	In comparison with UIR[8]	In comparison with SAS[9]
AQRT	1.9 times faster	0.7 times faster
IR size	34.2 % smaller	24.69 % smaller
Up Link Requests	44.35 % lesser	20.64 % lesser
Broadcast Times	119.89 % lesser	119.89 % lesser

7 Conclusion

In this paper, we proposed a new cache management scheme called as- *"Adaptive Cache Invalidation Technique (ACIT) for Mobile Environments"* that overcomes the higher query response time and more number of up link requests problem found in the existing state-of-the-art schemes in literature. Our strategy included only hot data updates in IR to reduce the size of IR which helped in better utilization of bandwidth, and number of hits. The proposed ACIT has historical data with timestamps of w BT intervals for disconnected MN. So, MNs can sleep or disconnect to save power consumption in the proposed scheme. The proposed scheme works in fast mode with updated IR and broadcast time when update rate is higher than a predefined threshold value which results in having faster query replies. Due to inclusion of hot data in IR, cache contains only hot data content so, queries are replied from cache which reduces the number of uplink requests. To evaluate the performance of ACIT, exhaustive simulations were performed by varying the parameters such as- update rate, and query arrival rate, and observed their impact on

AQRT, IR report size, uplink requests and broadcast time. Results obtained show the effectiveness of proposed ACIT scheme over other state-of-the-art schemes such as UIR, and SAS.

In the future, we will test the performance of the proposed scheme in environments where mobility rate is higher. Also, security aspects of the proposed scheme would be evaluated with respect to different network threats presented in the network.

Acknowledgments We are thankful to all the anonymous reviewers for their valuable suggestions and comments which improves the overall content, quality, and presentation of the paper.

References

1. Li, Y., & Chen, I. (2011). Adaptive per-user per-object cache consistency management for mobile data access in wireless mesh networks. *Journal of Parallel and Distributed Computing*, *71*(2), 1034–1046.
2. Newman, M. (2003). The structure and function of complex networks. *SIAM Review*, *45*(1), 167–256.
3. Sutton, C. (2008). *Internet began 35 years ago at ucla with first message ever sent between two computers*. Los Angeles: UCLA.
4. Schacham, N. & Westcott, J. (1987). Future directions in packet radio architectures and protocols. In: *Proceedings of the IEEE*, *75*(1), 83–99.
5. Diacui, C., & Berkenbroc, M. (2008). Supporting cache coherence. In: *Mobile cooperative system: Seventh IEEE international symposium on network computing and applications* (pp. 240–243).
6. Chuang, J.P., & Chiu, Y. (2008) Constructing efficient cache invalidation schemes for mobile environments. In: *Third international IEEE conference on signal-image technology and internet based system* (pp. 281–288).
7. Barbara, D., & Imielinksi, T. (1994). Sleepers and workaholics: Caching strategies for mobile environments. *ACM SIGMOD*, *23*, 1–12.
8. Cao, G. (2000). A scalable low-latency cache invalidation strategy for mobile environments. *ACM MOBICOM*, 200–209.
9. Safa, H., Aartail, H., & Nahhas, M. (2010). A cache invalidation strategy for mobile networks. *Journal of Network and Computer Applications*, *33*(2), 168–182.
10. Khurana, S., Kahol, A., Gupta, S., & Srimani, P. (2000). A strategy to manage cache consistency. In: *A distributed mobile wireless environment. IEEE international conference on distributed computing systems (ICDCS)*.
11. Cao, G., & Dar, C. (2001). On the effectiveness of a counter-based cache lnvdidation scheme and its resiliency to failures in mobile environments. In: *Proceeding of 20th IEEE symposium on reliable distributed system (SRDS)* (pp. 247–256).
12. Cao, G. (2002). On improving the performance of cache invalidation in mobile environments. *Mobile Networks and Applications (MONET)*, *7*(4), 291–303.
13. Kumar, N., & Lee, J. H. (2014). Peer-to-Peer cooperative caching for data dissemination in Urban vehicular communications. *IEEE Systems Journal*. doi:10.1109/JSYST.2013.2285611.
14. Tiwari, R., & Kumar, N. (2012). A novel hybrid approach for web caching. In: *Proceeding of IMIS 2012* (pp. 512–517). Palermo.
15. Wu, C.-C., Fang, J.-F., & Hung, P.-C. (2003). A counter-based cache invalidation scheme for mobile environment with stateless servers. In: *IEEE Pacific Rim conference on communications, computers and signal processing PACRIM*.
16. Jing, J., Elmagarmid, A., Helal, A., & Alonso, A. (1997). Bit sequences: An adaptive cache invalidation method in mobile/server environments. *Mobile Networks and Applications*, *2*(2), 115–127.
17. Cai, J., & Tan, K. L. (1999). Energy efficient selective cache invalidation. *Wireless Networks*, *5*(6), 489–502.
18. Cai, T., & Ooi, H. (2001). An evaluation of cache invalidation strategies in wireless environments. *IEEE Transactions On Parallel & Distributed Systems*, *12*(8), 789–807.
19. Chand, N., Joshi, R. C., & Misra, M. (2005). Energy efficient invalidation in mobile environment. *Journal of Digital Information Management*, *3*(2), 119–125.
20. Joseph, M.S., Kumar, M., Shen, H., & Das, S. K. (2005). Energy efficient data retrieval & caching in mobile P2P network. In: *International conference on pervasive computing & communications workshops* (pp. 50–54).
21. Chin, C., Fie, J., & Chun, P. (2009). A counter based cache invalidation scheme for mobile environments with stateless servers. *Communications, Computer & Signal Processing*, *2*, 623–626.
22. Wu, C. C., Fang, J, & Hung C. P. (2003). A counter-based cache invalidation scheme for mobile environments with stateless servers. In: *IEEE Pacific Rim conferences on communications, computers and signal processing* (pp. 623–626).
23. Safa, H., & Aartail, H. (2008). COACS: A cooperative and adaptive caching system for MANETs. *IEEE Transactions on Mobile Computing*, *7*(8), 951–977.
24. Chuang, P., & Chiu, Y. (2011). Efficient cache invalidation schemes for mobile data accesses. *Information Sciences*, *181*, 5084–5101.
25. Fatima, N., & Khader, P. (2011). Enhanced adaptive cache invalidation approach for mobile environments. In: *3rd IEEE international conference on electronic computer technology (ICECT)* (pp. 76–80).
26. Lim, S., Lee, S., Soha, M., & Lee, B. (2013). Energy-aware optimal cache consistency level for mobile devices. *Information Sciences*, *230*(5), 94–105.
27. Paul, P., & Saravanan, N. (2013). Efficient service cache management in mobile P2P networks. *Future Generation Computer Systems*, *29*(6), 1505–1521.
28. Nguyen, T., & Dong, T. (2010). An efficient cache invalidation system in mobile information systems. In: *IEEE international conference on computing and communication technology, research, innovation, and vision for future (RIVF)* (pp. 1–4).
29. Paul, P., Saravanan, N., Baskaran, R., & Dhavachelvan, P. (2013). Efficient service cache management in mobile P2P networks. *Future Generation Computer Systems*, *29*(6), 1505–1521.
30. Chan, E., Li, W., & Chen, D. (2009). Energy saving strategies for cooperative cache replacement in mobile ad hoc networks. *Pervasive and Mobile Computing*, *5*(1), 77–92.
31. Sunho, L., Chansu, Y., & Das, C. R. (2012). Cache invalidation strategies for internet-based vehicular ad hoc networks. *Computer Communications*, *35*, 380–391.
32. Cao, G., Yin, L., & Das, L. C. (2004). Cooperative cache based data access in ad-hoc networks. *Computer*, *37*(2), 32–39.
33. Shen, H., Joseph, M. S., Kumar, M., & Das, S. (2005). A scheme for cooperative caching in mobile Peer to Peer network. *IEEE international parllel & distributed processing symposium* (pp. 57–64).
34. Chan, E., & Liw, L. (2012). Movement prediction based cooperative caching for location dependent information service in mobile adhoc network. *Journal of Supercomputing*, *59*(1), 297–322.
35. Yin, L., & Cao, G. (2006). Supporting cooperative caching in ad hoc networks. *IEEE Transaction on Mobile Computing*, *5*(1), 77–89.
36. Kumar, P., Chauhan, N., Awasthi, L., & Chand, N. (2010). Proactive approach for cooperative caching in mobile adhoc networks. *International Journal of Computer Science*, *7*(8), 21–27.

37. Mrmol, F., & Prez, G. (2012). TRIP: A trust and reputation infrastructure-based proposal for vehicular ad hoc networks. *Journal of Network and Computer Application*, *35*(3), 934–941.
38. Dubey, A., & Sharma, S. (2011). A cache invalidation scheme through data classification in IVANET. *International Journal of Computer Application*, *25*(9), 54–57.
39. Wang, Y., & Li, F. (2005). *Vehicular ad hoc networks*. chapter 20, 503–525.
40. Ahmadifard, N., Nabizadeh, H., & Abbaspour, M. (2014). ISEFF: An id-based scalable and efficient distributed file shaing technique in vehicular ad hoc networks. *Wireless Personal Communications*, *75*(2), 821–841.
41. NS2 simulator, http://www.insi.edu/nsnam/ns (2008)
42. Issariyakul, T., & Hossain, E. (2011). *Introduction to network simulator NS2*. Berlin: Springer.
43. Lim, S., Lee, W., Cao, G., & Das, C. R. (2006). A novel caching scheme for improving Internetbased mobile ad hoc networks performance. *Ad Hoc Networks*, *4*(2), 225–239.

Rajeev Tiwari is working as an Assistant Professor in UPES, Dehradun (India). He is pursuing Ph.D. from Thapar Univeristy, Patiala (Punjab). He has more than 10 years of experience in teaching and research. He has served many institutes of repute in teaching and research during last 10 years. His broad area of research is MANET, VANET, QoS in wireless networks and adaptive cache invalidation techniques. He is also an expert in simulation and design of scenarios on NS2, SUMO and MOVE. He is a member of IEEE,IACSIT and IAENG. He is a reviewer in SCI indexed journal like AHWSN. He is TPC member of many international Springer, IEEE and Elsevier conferences.

Neeraj Kumar is working as Associate Professor in Department of Computer Science and Engineering, Thapar University, Patiala (Punjab), India . He received his Ph.D. in CSE from Shri Mata Vaishno Devi University, Katra(India) and PDF from Coventry University, Coventry, UK. He has more than 100 scholarly research publications in peer reviewed journals and conferences including IEEE, Elsevier, and Springer. Some of his research findings can be found in top-cited journal such as- IEEE TIE, IEEE TDSC, IEEE TITS, IEEE TCE, IEEE Netw., IEEE Wireless Comm., IEEE Netw., IEEE SJ, FGCS, JNCA, TJ, and ComCom. His research is focused on mobile computing, parallel/distributed computing, multiagent systems, service oriented computing, routing and security issues in wireless adhoc, sensor and mesh networks. He is leading the Mobile Computing and Distributed System Research Group. Prior to joining Thapar University, Patiala, he has worked in SMVDU, Katra, HEC Jagadhri and MMEC Mullana, Ambala, Haryana, India. He has delivered invited talks and lectures in various IEEE international conferences in India and abroad. He has organized various special sessions in international conferences in his area of expertise in India and abroad. He is TPC of various IEEE sponsored conferences in India and abroad. He is reviewer/ editorial board of various international journals. He is guest editor of special issue of 6 international journals. He is senior member of ACM, ACEEE and IACSIT.