An Effective Hybrid and Adaptive Caching Network Framework Approach In Time-to–Live (TTL) Based Data Item for Peripatetic Computing Environment

Srimanchari P (srimancharieasc@gmail.com)
Erode Arts and Science College https://orcid.org/0000-0002-2271-8254
Anandharaj G
Adhiparasakthi College of Arts and Science

Research Article

Keywords: Hybrid and adaptive caching, storage systems, data availability, cache replacement technique, cache invalidation technique.

Posted Date: June 15th, 2021

DOI: https://doi.org/10.21203/rs.3.rs-521325/v1

An Effective Hybrid and Adaptive Caching Network Framework Approach In Time-to –Live (TTL) Based Data Item for Peripatetic Computing Environment

[1]Dr. P. Srimanchari and [2]Dr. G. Anandharaj
[1]Assistant Professor, Department of Computer Science
Erode Arts and Science College, Erode – 638009
srimanchari@gmail.com

[2]Associate Professor, Department of Computer Science
Adhiparasakthi College of Arts and Science, Kalavai – 632506, Vellore (Dt)
younganand@gmail.com

Abstract
Caching is a well established technique to improve the efficiency of data access. This research paper introduces a Hybrid and Adaptive Caching (HAC) approach to cache the data item based on the varying size, and, Time-to-Live (TTL) based invalidation of the data item in a mobile computing environment. Mobile nodes establish single-hop communication with the base station and ad-hoc peer to peer communication with other neighbor nodes in the network to access data items. The proposed work adjusts the caching functionality level based on the size of the data item and stores the cached data item in two different storage systems. The cache of each node is separated into Temporary Buffer (TB) and Permanent Buffer (PB) to improve the data access efficiency. This approach is based on the fact that the smaller size data (e.g. stocks) are updated for shorter Time-to-Live (TTL) whereas the larger size data (e.g. video) are updated only for longer TTL. This proposed work also suggests an adaptive cache replacement and cache invalidation technique to resolve the issues regarding bandwidth utilization and data availability. In cache replacement technique, the cached data item is effectively replaced based on the size of the data item and TTL value. A timestamp-based cache invalidation strategy where the cached data is validated according to the update history of the data items has also been introduced in this paper. The threshold values have greater impact on the system performance. Therefore, the threshold values are fine tuned such that they do not affect the system performance. The proposed approaches significantly improve the query latency, cache hit ratio and efficiently utilize the broadcast bandwidth. The simulation result proves that the proposed work outperforms the existing caching techniques.

Keywords: Hybrid and adaptive caching, storage systems, data availability, cache replacement technique, cache invalidation technique.

1. Introduction

The performance level of the mobile data access can be significantly improved by caching frequently accessed data items at each mobile node's buffer. Designing an effective mobile caching, cache replacement, and cache invalidation are the challenging

task because of dynamic mobility and frequent disconnection of mobile nodes. The mobile environment considered is as hybrid, which is a combination of single hop connection with the base station and multi hop connection with other nodes using the peer-to-peer communication. The caching technique must be adaptive to both these communication systems. Nowadays, the mobile computing technology and computing power has improved significantly. However, the mobile computing environment includes frequent disconnection of mobile nodes, constricted bandwidth, and limited battery resources as a serious constraint [28].

The major components of the mobile computing environment are mobile nodes and base station (or Mobile Service Stations). Each mobile node is made to communicate with one base station. A mobile node may be under the control of more than one base station. A disseminated data can be stored on both the mobile nodes and base stations, and they are queried and accessed over the wireless network. In such a hybrid environment, limitation on bandwidth restricts the amount of data that could be transmitted. Bandwidth utilization is the key issue in the mobile computing environment due to the expensive wireless medium. The disconnectivity rate and noise level are too high in a mobile computing environment when compared to conventional environment. A novel caching policy is presented especially for heterogeneous mobile environment in which each node computes their own caching policy independently regardless of central authority[24]. The prominent way to improve the mobile computing system performance is to adopt a caching technique [5]. The query access latency is significantly reduced as the data can be retrieved from the cache rather than forwarding the query request to the server. A caching technique must be always accompanied with an effective cache consistency [6], cache replacement and cache invalidation. Several caching techniques have been proposed for web environment [17] [10] and ad hoc networks [1].

This work has proposed novel hybrid architecture to resolve the drawbacks in existing caching issues (data availability, proper bandwidth consumption, etc). The proposed mobile system is constructed through combining the mobile infrastructure based architecture with the ad hoc based P2P communication. This communication is being referred as an ad hoc based P2P data dissemination model. In order to reduce the bandwidth utilization and the access latency, the communication pattern is properly selected according to the size of the requested data item. In this approach, caching system is designed to use the less storage resources in the mobile communication network. In this approach, each mobile node is available with two storage systems, namely, Temporary buffer (TB) and Permanent buffer (PB). Based on the size of the data item, the mobile node caches the data item in two different local buffers. The temporary buffer caches smaller size data items (stock data) are updated for every short Interval of time (less TTL), whereas the permanent buffer with large size data items (video) is updated for large interval of time (high TTL).

1.1 Aim and Objectives

- To design an efficient and adaptive caching mechanism for a hybrid mobile computing environment on considering data items of different sizes.
- To propose a hybrid caching mechanism that optimizes the bandwidth utilization effectively. This objective is achieved on caching the data items based on the size of the data item and stores appropriately on TB and PB.
- To suggest an effective and adaptive cache replacement technique and cache invalidation technique.

1.2 Paper Organization

This paper is organized as follows: Section 2 investigates the previous works that are related to caching, invalidation and replacement. Section 3 discusses the overview of the proposed hybrid and adaptive caching in the mobile environment. Section 4 suggests a hybrid and an adaptive caching technique. This technique ensures data availability in the hybrid mobile environment and reduces bandwidth consumption. Section 5 explains the experimental set up, simulation analysis, and simulation results. Section 6 concludes the paper.

2. Previous Work

This section provides a survey on the previous works based on the adaptive cache concepts. It also includes the survey of adaptive cache invalidation and cache replacement techniques.

2.1 Adaptive caching techniques

The cache management in a distributed mobile environment is not an easy task due to disconnected architecture, resource and energy constraints. An effective and hybrid distributed storage system is discussed in [21]. It is a combination of page cache and object cache. It caches page and object adaptively based on the locality. According to the application, it regulates the available cache space and decides whether to cache a page or an object adaptively. Hybrid adaptive caching considerably reduces the cache misses ratio and retaining the locality.

Semantic caching also supports mobile caching that answers only spatial queries on caching the required data items along with query descriptions. The drawback in semantic caching is that it can answer only to certain types of queries. It is much difficult to share the requested data items between various query types. This issue is resolved in [9] that propose a proactive caching scheme. This scheme caches the index addition to the requested object. The main idea behind caching the index is that it allows reuse of cached objects in order to answer all types of queries. The caching of the index considerably reduces the query response time. The demerit of this approach is that it does not support all types of queries, and it finds difficult to distribute all these types.

The cooperative caching in the mobile environment specifies that a mobile node can access the data in its neighbor's cache. The concept of Cooperative Caching (COCA) has been well explained in [2]. COCA broadly classifies the mobile nodes into low activity and high activity and correspondingly they are known as Low Activity Mobile nodes (LAM), and High Activity Mobile nodes (HAM). COCA considerably reduces the system complexity and reduces the cache miss ratio. The cooperative caching is efficient only when all the cooperative caches the valuable data and thus, improves the cache hit ratio. The Group based Cooperative Caching (GroCOCA) [3] meet the requirement of each node along with its mobility model. In GroCOCA, a group of nodes that have a common mobility model are referred to as a Tightly Coupled Group (TCG).

2.1.1 Other caching techniques

An effective cooperative caching was suggested in [16] to enhance system performance in wireless P2P networks including ad hoc and mesh networks. The suggested

cooperative caching is asymmetric that places the data in cache effectively. The data queries are broadcasted to the cache layer on all nodes, but the requested data are broadcasted only to the cache layer of the nodes that actually require the data. The asymmetric cooperative caching reduce the overhead caused by copying data between the requester space and kernel space and also reduces average end to end delay.

Video data are cached in cellular network through small base stations called helpers. Cellular users requests video files and receives them through short range transmission i.e. they retrieve video file from helpers. If the helper does not have the corresponding file, it receives data from the base station. Femto caching optimizes the network architecture such that high rate backhauls are replaced by low rate backhauls [23]. It also designs architecture to perform coded and uncoded Femto caching. Uncoded Femto caching caches entire video file whereas coded Femto caching caches a segment of the video file. Femto caching reduces data access delay on using the helpers.

An approach ensuring efficient data access in Disruption Tolerant Networks (DTNs) is presented in [25] . It supports sharing and coordination between several nodes regarding cached data and hence, reduces data access delay. It deliberately places the caches data at a particular location called Network Central Location (NCL). NCL is selected appropriately based on the probabilistic selection metric. It ensures optimization between data access rate and overhead.

2.2 Cache replacement techniques

ACME [12] overcomes the drawbacks in distributed caching. ACME stands for Adaptive Caching design using Multiple Experts. ACME manages the replacement techniques among the distributed caches in order to increase the cache hit ratio. ACME assigns ratings to the existing cache replacement techniques on the basis of weight updated using machine learning algorithm. The caching node adjusts itself on assigning the weight based on the workload. The static cache replacement technique offers superior performance when combined with distributed networks. A target driven cache replacement strategy is suggested in [19]. A generalized value function is used for the cache replacement technique. Rather than introducing multiple parameters, a generalized function or parameter can be used to make the optimization process easier. The generalized parameter can be optimized in several ways according to the requirements. For instance, the generalized parameter is optimized in accordance to the access cost. Two different functions are defined and rated with the set up of two different targets. The targets are achieving minimum query access delay and achieving minimum downlink traffic. The drawback of this approach is the data items in a queue are re evaluated, and their positions are changed frequently. in-SAUD is used to ensure cache consistency before the cached data item is exploited [13]. Min- SAUD is designed on considering the factors like access probability, the size of the data item, query latency, and data access cost. The performance of the Min- SAUD is better compared to LRU and SAIU. The effectiveness of the optimal cache replacement is measured using a parameter called stretch. Stretch is a function of query latency, the size of the data item, and bandwidth. The optimal cache replacement technique replaces the data item with low gain value. It is difficult to manage the position of the data item in the heap. Moreover, it is difficult to estimate the value of the running parameters to maintain the updated status of the attributes of the data item. To overcome the disadvantages of local cache replacement, [22] introduces the global cache replacement technique. It revisits all the existing cache replacement technique and notices the drawbacks in those cache replacement techniques. The global cache replacement policies are of two types namely; communication based and centralized. The caches in the

communication based policy have their own local replacement technique. The disadvantage of the centralized cache replacement technique is a lack of scalability.

2.3 Cache invalidation techniques

The prominent cache invalidation technique is broadcasting Invalidation Reports (IR). The two serious drawbacks of the existing cache invalidation technique are addressed in [7]. The first drawback is, the query latency increases between two consequent IRs. The second drawback is that, if the server updates a data item, the mobile nodes must send a query again to the server increasing the bandwidth utilization. Therefore, [7] suggests a cache invalidation technique that substantially overcomes the abovementioned drawbacks. It decreases query latency and optimizes bandwidth utilization. The Bit Sequence (BS) [15] is an effective cache invalidation technique aims at reducing the bandwidth utilization. BS achieves its aim by optimizing the invalidation report size, and at the same time, it maintains the efficiency of the cache invalidation.

The BS algorithm makes use of three factors to optimize the size of the invalidation report. These factors include BS naming, update aggregation, and BS hierarchical structure. In the BS algorithm, a unique identity to the data items in the database is assigned using BS naming procedure. The UIR based cache invalidation technique is extended to reduce the data access latency [8]. It also proposes a counter based cache invalidation technique to overcome the issue regarding bandwidth usage. Different techniques are proposed to manage failure in server and mobile node, and disconnection between the mobile node and server. The data items that are expected to be used in the near future are prefetched and can be used when it can be reused. If there is no high variation in the traffic pattern, the hot data items remain hot, and the cold data item remains cold. The problem arises when the network traffic changes frequently. When the main server gets failed, there is no guarantee for the freshness of the data item. In order to reduce the data access latency, a Dynamic Invalidation Report (DIR) especially for the mobile computing environment is proposed [27]. DIR holds an early hour's cache validation technique on exploiting the validation texts. It enables the mobile nodes to check the cached data very quickly. This decreases the data access latency further by proposing a method, DIR-AI to make IR adaptive. DIR considers stateless server, and it has knowledge of the number of a request made at any time. A special approach is suggested to maintain the freshness of the cache [26]. Each caching node is designed such that it is responsible for refreshing a particular set of caching nodes. This design maintains the freshness of the cache in both distributed and hierarchical fashion. The freshness of the cached data is maintained throughout entire communication using probabilistic replication.

2.4 Prefetching techniques

A power-aware prefetch scheme is designed to resolve all these issues. This scheme proposes Value-based Adaptive Prefetch (VAP) technique specifically for mobile computing [20]. There is an issue in selecting the data item that has to be prefetched. The VAP suggests the data item for prefetch on the basis of the assigned value of the data item. The value is estimated on the basis of remaining power, query latency, the size of the data item, and the update rate. In most of the prefetching techniques, "stretch" is used as a primary parameter metric to evaluate the performance level of the prefetching technique. The VAP considers the stretch as the performance metric along with the power consumption. The Cache-Miss-Initiated Prefetch (CMIP) mechanism assists the mobile nodes to determine the appropriate data item for prefetching [11]. CMIP has been proposed based on two remarks. Firstly, a mobile node may request a set of data

repeatedly. In that case, if those data are cached, there is no need for the mobile node to make a request repeatedly. Thus, CMIP reduces the uplink traffic and saves the energy level of the mobile node. Therefore, the data items that are expected to be accessed frequently are prefetched. Secondly, data items requested within an interval of time are inter-connected with each other. Two different prefetch sets can be constructed in two steps. In the first step, the history of the mobile node is extracted to describe the association rule. Consequently, confidence parameter is calculated. In the second step, the confidence parameter obtained from the association rule is used for the construction of prefetch sets. The common issues in the IR based cache invalidation technique are addressed in [6], and it suggests a prefetching technique. It prefetches the data items that are expected to be retrieved in the near future. This technique exploits prefetch access ratio to optimize the performance level of the mobile computing environment. This technique improves cache hit ratio, system throughput, and the bandwidth consumption addition to reducing the query access delay and power utilization. When the query delay is considered to be the primary issue, the system should prefetch the data on the basis of the user community. Broadcasting the data on the air seems to be a complete solution to the frequently updating environment. The trade off among the power consumption and query latency are reduced by several indexing techniques.

3. Hybrid and adaptive caching in the mobile computing environment

The proposed hybrid and adaptive caching technique directs a way to use the mobile cache memory effectively. It considerably reduces the access time on using a dual buffer storage system.

3.1 Overview of hybrid and adaptive caching in the mobile environment

In this work, a hybrid mobile computing environment is constructed with the base station and set of mobile nodes within the transmission range. Therefore, there is a no need to maintain fixed infrastructure and somehow manages the network functioning even in the presence of greater mobility [4]. The mobile nodes move randomly, and they must operate in the limited energy constrained resources. In order to manage the dynamic mobility and limited resources of the mobile environment, a hybrid network is constructed through combining the conventional infrastructure architecture with the ad hoc communication architecture. Here, the communication pattern between the base station and a set of mobile nodes is adaptive. It is determined from the size of the requested data item.

It is essential to notice that the mobile nodes could communicate directly with the base station through a single hop connection. Therefore, all mobile nodes can access the data directly from the base station without any intercommunication between any other nodes in the network. In the case of a large data item, request is forwarded to neighbor nodes in the network rather than the base station which is in the single hop transmission range. This is because; high bandwidth is needed to communicate directly with the base station for large size data item. In order to reduce the bandwidth utilization and the access latency, a communication pattern is properly selected according to the size of the requested data item. In order to improve the cache performance, repeatedly accessed data items is cached among the mobile nodes in the network. Caching not only ensures data availability, but also helps to utilize the bandwidth efficiently.

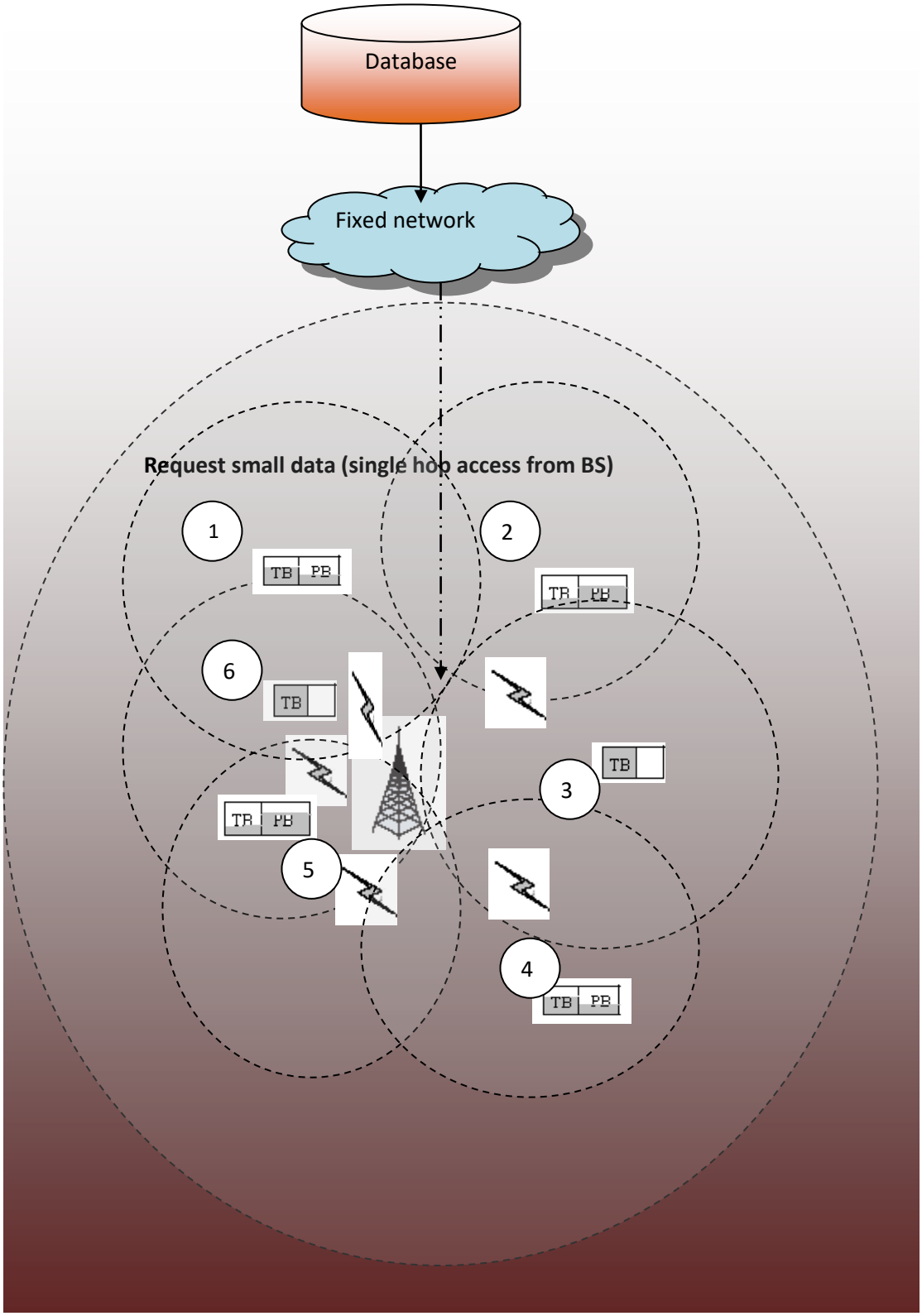

Fig.1. Hybrid Adaptive Caching Mobile Environment

The caching technique must be designed in such a way that it must offer the best service even in the greater mobility and disconnectivity. An effective hybrid caching technique is suggested in [18] that combine the advantages from two different caching techniques called cache data and cache path. In the cache data techniques, the data is cached while in the cache path technique the path for data location is cached. The hybrid

caching technique implements either cache data or cache path based on the size and TTL of the data item.

The Fig. 1 demonstrates the hybrid architecture of a mobile cache system. It comprises of the main database server, Base Station (BS), and several mobile nodes. The mobile nodes are connected to the base station via a wireless link which in turn connected to the main database server through a fixed network.

The base station is responsible for broadcasting three key messages, namely, broadcasting technique, replacement message, and invalidation report. This approach is different from other proposals as it suggests a caching technique with variable data sizes. It is much difficult to design a caching system for the data of different sizes. To achieve this, each node maintains a local cache into two parts. The first part can be referred as temporary buffer and the second part can be referred as permanent buffer. The temporary buffer stores the data items of smaller sizes which is accessed frequently (e.g. stock). Therefore, buffered data in a temporary buffer is updated for every short interval of time (less TTL). The permanent buffer of a cache stores larger data items (e.g. Video file)

3.2 System design

This work has been proposed based on hybrid caching architecture. Depending on the size of the requested data item, the proposed work adaptively supports caching on both hybrid communication pattern, such as conventional infrastructure (single-hop) and a mobile ad hoc environment (multi-hop).

The mobile node generates the request for small size data item the single hop communication scenario is triggered where the Mobile Service Station (MSS) directly communicates to all the mobile nodes within its communication range. On the other hand, the mobile nodes generate the request for larger data item; they establish a mobile ad hoc model using multi hop communication. In this case, there are three possible outcomes of a request generated for large data item (video).

1. Local Cache Hit (LCH). If the requested large data item is available in its own mobile local cache, then local cache hit is occurred; otherwise, local cache miss.
2. Global Cache Hit (GCH). In the case of local cache miss, the mobile node forwards the request to neighbor peers in network using multi hop communication. If the neighbor node contains the requested data item, then constitutes a global cache hit. Otherwise, global cache miss will be considered.
3. Base Station Hit (BSH): In the case of requested data item is not available in both in its local cache and other neighbor nodes, the request is forwarded to the base station through single hop communication.

The effectiveness of the proposed work is proved with some assumptions as follows: Consider a hybrid network ($H_N(V,E)$) with n number of mobile nodes denoted as N_1- Nn. H denotes the hybrid network; V represents the nodes in network, E represents the edges of V, and N represents the number of nodes in the network. The combination of infrastructure and ad hoc network is called hybrid network. In infrastructure network, the nodes are connected with Base station (BS). In ad hoc network, the mobile-to-mobile communication is established. The H(V,E) has several data items denoted as Di; i - varied from 1-n [$H(V,E)\forall\{D_1 \ldots\ldots D_N\}$]. The size of Di is DSi. The intermediate node can cache multiple data items. The node has limited memory capacity (Mi) to cache the Di.

The access frequency for Di is represented as A_{Dn}. The A_{Dn} value is measured for all data items. Since, if the node memory Mi is overflowed (Mi $\in$(DSi) < Mi), the data item which has lowest A_{Dn} is replaced by a new data item Di. Let 'R_{ch}' be the cache hit ratio of a mobile node. 'β' be the total number of bits transmitted during a particular interval of time. Let 'b_{br}' be the number of bits forwarded by the broadcast. Let L be the broadcast interval between the invalidation reports. The throughput (τ) of the caching system can be represented as follows:

$$\tau = \frac{\beta - b_{br}}{(b_r + b_u)(1 - R_{ch})} \quad (1)$$

The calculation of throughput must be normalized so that it is easy to compare the effectiveness of the proposed system with other caching techniques. The effectiveness of the proposed caching system (C_{eff}) can be estimated as:

$$C_{eff} = \frac{\tau}{\tau_{max}} \quad (2)$$

τ_{max} is the throughput value that is determined by an unfeasible caching technique in which it supports an instant cache invalidation technique.

3.3 Storage system

The proposed work is both hybrid and adaptive and hence, the design of a storage system of each mobile node is a bit complex process. The cache memory of each node is divided into TB, and permanent buffer PB. A cache memory of each node is divided into TB and PB. This kind of division in the cache memory is brought to make the caching process efficient as it has heterogeneous data items. Whenever a cache receives a data, HAC initially checks the Data Size (DS) of the Data Item (Di). If the DS is greater than the Threshold value for Data Size (TH_{DS}), it is cached in PB. Otherwise, it is cached in TB. The data in TB are smaller in size and has less TTL value. These data are updated frequently. In contrast, the data in PB are larger in size and has a large TTL value. These data are updated less frequently compared to data in TB. The table 1 describes the characteristics of the data in TB and PB.

Data in cache	Data Size (DS)	TTL	Update Rate
TB	Small	Less	High
PB	Comparatively Large	High	Less

Table 1: TB and PB Characteristics

This approach introduces two storage systems for each mobile node. This storage system makes easier to retrieve the data items according to the update_interval. It is necessary for the server to satisfy the requirements of all the mobile nodes. The time taken to download a file also plays a crucial role in the storage system. For instance, a movie takes longer time to get download. Therefore, it is stored in the PB as it will not expire shortly. The data items that are expected to remain valid for a longer period is stored in PB. On the other hand, the data items that are expected to lose its validity already are stored in TB. The proposed work helps the server to perform the required data before their validity get expires.

3.4 Hybrid and adaptive cache mechanism

The hybrid and adaptive cache mechanism is primarily designed to enhance the cache performance in the hybrid mobile environment. The proposed work is flexible so as to satisfy the application requirements. Since the cache technique is adaptive, both the cache invalidation and cache replacement techniques also must be adaptive. Most of the existing caching scheme works in the assumption that all the data items are equal in size. The proposed work considers the broadcasting of data items according to the size of the data item. Here, a data item is cached in the mobile node according to the size of the requested data item. The cache of each node is designed to accommodate two storage systems. This is because the size of the data item is directly related to the update_interval.

Due to the limited memory capacity Mi, the Threshold value for Data size (TH_{DS}) is fixed. The node memory is divided into TB and PB. If the data size DSi is within the TH_{DS}, the DSi is stored in the TB. Otherwise, the data item is stored in PB. The $D_{TTL\ (i)}$ is varied with respect to the DSi. The $D_{TTL\ (i)}$ is high when it has large size. The TB data items Di should be within the value of TTL threshold (TH_{TTL}). Otherwise, the data item is stored in PB.

$$D_{TTL\ (i)} \alpha\ DSi \text{ ……. (3)}$$

HAC Cache Policy

For data item Di,
{
If ($DSi < TH_{DS}$ && $D_{TTL} < TH_{TTL}$)
{
Cache the data items in TB
}
Else
Cache the data items in PB }

In existing cache techniques, all data items are stored in server node only. If any node requires a data item, it requests the server node via multihop links. This kind of cache technique increases the data delay and traffic generation. The existing cache techniques do not utilize the node memory properly. Only, it considers a scenario with similar data size. Various data size is not considered in the cache technique. In order to make an effective cache, consider a real scenario with various data size and cache nodes. Since, every node retains its own information in its memory. If A_{Dn} is high for a particular Di, the Di is stored in the intermediate nodes itself. These intermediate nodes are called as cache nodes. The main problem associated with this network is that the multiple intermediate nodes cache similar data items. It leads to memory constraint. In order to avoid this problem, only the small size data items are stored in the cache nodes. It leads to proper memory utilization. Consider a scenario with a single small size data item Di is cached in multiple cache nodes Cn (Cn is N_3, N_7, and N_9). In Fig 2, shows the ad hoc connectivity of cache nodes.

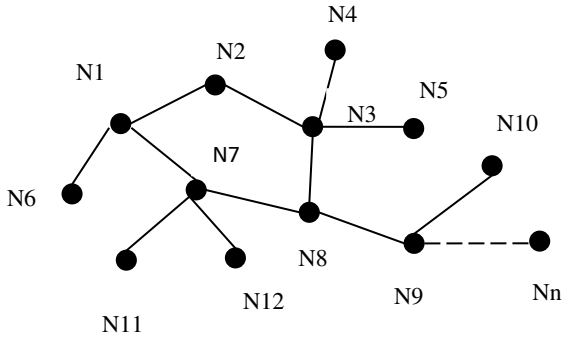

Figure 2: Ad hoc network

Consider a scenario with a single small size data item Di is cached in multiple cache nodes Cn (Cn is N_1, N_3, N_7, and N_9). In Fig 2, the node 1 acts as a destination node. The cache node aids to reduce the delay time. In Fig 2, the nodes 4 and 5 request the data item via node 3. Hence, the data item Di has the highest A_D is stored in the intermediate node 3. Thus, the node 3 acts as a cache node Cn. In this network, the node 3, 7, and 9 act as a cache node. Hence, the delay is reduced.

The nodes in the resource constraint hybrid mobile environment have high mobility. Therefore, cooperative caching mechanisms developed for wired environment is not suitable for hybrid mobile environment. Due to high mobility in the hybrid mobile environment, the node that caches a data item may move somewhere in the network. The node that has changed its position should send query to the server after it recognizes the problem. The cached data may not be consistent and using these data may introduce additional overhead. To overcome this problem, a source node caches the data only if the caching node is much close to the source node. The distance between source and destination must be high compared to that of source and cache node. If the distance between the source node and caching node is less, it can resolve the problem quickly in the case of route failure due to mobility. Therefore, HAC can ensure data availability in the environment with tolerable high mobility.

3.4.1 Delay Analysis

$[\tau_{QRY} + \tau_{DATA}]$ → Forwarding Time of Query and Data between two hops

Total Query Delay (D_{QRY}) between two nodes (N1-N2) is calculated as follows:

$$D_{QRY(N1\text{-}N2)} = [\tau_{QRY} + \tau_{DATA}]_{(N1\text{-}N2)} \text{ ..... (4)}$$

Total Query Delay (D_{QRY}) for a path between Source (Ns) and Destination (N_D):

Total Delay without cache node ($D_{QRY\ (NS\text{-}ND)}$) = $H_C * [\tau_{QRY} + \tau_{DATA}]_{(NS\text{-}ND)}$ …. . (5)

Total Delay with cache node ($D_{QRY\ (NS\text{-}ND)}$) = $(H_C - n) * [\tau_{QRY} + \tau_{DATA}]_{(NS\text{-}ND)}$ …. (6)

n→ Number of hops conserved due to the cache node

$$n = H_{C\ (ND\text{-}NS)} - H_{C\ (ND\text{-}Cn)} \text{ …….. (7)}$$

The equation (5) makes it clear that the reduction of query delay due to the cache node.

In Fig 2, the node 5 queries Di to N_D. The query packet is forwarded via N_5-N_3 - N_2 - N_D. The node 4 also queries the same Di to N_D. It uses the path of N_4-N_3 - N_2 - N_D. The node 3

is linked with two paths. Hence, the node 3 acts as a cache node for that particular Di. Thus, the hop count is reduced.

Total Delay without cache node ($D_{QRY\ (N5-\ ND)}$) = $H_C * [\tau_{QRY} + \tau_{DATA}]_{(N5-\ ND)}$
The two hop count is conserved due to the cache node. Hence, the value of n is 2.
Total Delay with cache node ($D_{QRY\ (N5-\ ND)}$) = $(H_C - 2) * [\tau_{QRY} + \tau_{DATA}]_{(N5-\ ND)}$

3.5 Adaptive cache replacement strategy

The cache replacement is a vital task to achieve a good cache management, especially in the on-demand communication system. In normal cache replacement strategies, access frequency is the parameter that decides the data items for replacement. In this hybrid communication architecture, the cache replacement parameter considered is update_interval (TTL) and the size of the data item in addition to access frequency. The server or the base station generates data updates in which the update interval is distributed exponentially. It is necessary to design an effective cache replacement strategy to achieve a high cache hit ratio. This work planned to design a cache replacement strategy that is adaptive to a hybrid mobile environment. The cache replacement strategy works on the basis of update_interval, DS and Access Frequency (A_D) of the data item.

3.5.1 HAC Cache Replacement Strategy

The TB contains only the data items with smaller sizes. TTL and A_D is calculated for all data items in a cache. If a data item is found with minimum TTL ($[D_{TTL\ (i)}]_{(MIN)}$) or minimum A_D ($A_{Di\ (MIN)}$), it is replaced with new data item (Dnew) in TB. If a data item is found with maximum TTL ($[D_{TTL\ (i)}]_{(MAX)}$) or minimum A_D ($A_{Di\ (MIN)}$), it is replaced with new data item (Dnew) in PB.

3.5.1.1 Access Frequency (A_D) Calculation for Di

The access rate of Di is called Access Frequency (A_D). The number of nodes requires a particular Di is denoted as $N_{Di.}$ The A_D of a Di is calculated as,

$$A_{Di} = N_{Di} / N - N_{Di} \quad \text{..... (8)}$$

3.5.1.2 Algorithm to Identify the Lowest Access Frequency Node

```
if (i=N1, i<N, i++)
        {
                if (j=N1, j >N, j++) {
                if {v (i)> v (j)} {
                Puts "L = N1"
                }}
        }
```

Step 1: The first data item N_1 is assigned as i and j
Step 2: The values of i and j are compared with each other
Step 3: The values are sorted as the highest value and the lowest value and the sorted values are stored in “L”
Step 4: The j value is incremented to the next data item of N_{B2}
Step 5: Then, i and j values are compared and sorted
Step 6: The process is continued until it reaches the Nn
Step 7: The value of i is incremented to the N2
Step 8: Step 2 to Step 5 is performed until i reaches the Nn.

From the list L, the cache node can identify the data item which has the lowest access frequency.

3.5.1.3 Algorithm for Cache Replacement

Adaptive cache replacement strategy

If there is no enough memory space {
Go
For a data item in TB,
{
$\Rightarrow \{N \in Di\}$
Find $(N \in [(D_{TTL\,(i)}) \,\&\&\, A_{Di}])$
If $[D_{TTL\,(i)}]_{(MIN)} \,\|\, A_{Di\,(MIN)} < \{N \in [Di\text{-}(D_{TTL\,(i)})]_{MIN} \,\|\, A_{Di\,(MIN)}\}$
Replace the data item
Dnew $\rightarrow (D_{TTL})_{MIN}$
For a data item in PB,
Find $(N \in [(D_{TTL\,(i)}) \,\|\, A_{Di}])$
If $[D_{TTL\,(i)}]_{(MAX)} \,\|\, A_{Di\,(MIN)} > \{N \in [Di\text{-}(D_{TTL\,(i)})]_{MAX} \,\&\&\, A_{Di\,(MIN)}\}$
Replace $[(D_{TTL})_{(i)}]_{MAX}$ the data item with Dnew
Dnew $\rightarrow (D_{TTL})_{MAX}$
Else
Cache the data item
}}

If the node has no enough memory to store an arrived data item, the cache replacement algorithm is followed.

Step 1: In TB cache replacement, find TTL and A_D for all data items Di
Step 2: Select a data item which has minimum TTL and A_D value
Step 3: If any one of the condition is satisfied, the new Data Item (Di) replaces the selected Data item ($D_{TTL\,(MIN)}$)
Step 4: In PB cache replacement, find TTL and A_D for all data items Di
Step 5: Select a data item which has maximum TTL and minimum A_D value
Step 6: If both the conditions are satisfied, the new Data Item (Di) replaces the selected data item ($D_{TTL\,(MAX)}$)
Step 7: Else continue caching

3.6 Adaptive cache invalidation technique

Cache invalidation technique must be used to verify the freshness of the cached data. To prevent bandwidth usage in downlink broadcasting, and save uplink broadcasting bandwidth, we propose an adaptive cache invalidation technique that adjusts the broadcasting invalidation report (IR) based on the system workload [14]. The invalidation is an important factor for cache process. It validates the freshness of retained data items Di. Every node has Time Stamp (TS) which retains the time related history of all Di. The time related to history of a data item Di is called Update History 'U'. From the equation (8) makes it clear that the TS of a data item Di retain the, 'U(Di)'. The difference between current ($\Gamma_{CURRENT}$) and previous (Γ_{PREV}) update time of a Di is denoted as 'Γ'. The difference between pre-previous ($\Gamma_{PREPREV}$) and previous update time of a Di is denoted as 'Γ′'.

$$TSi \in U(Di) \quad \ldots\ldots (9)$$

$$\Gamma_{CURRENT} - \Gamma_{PREV} = \Gamma \ldots\ldots\ldots (10)$$

$$\Gamma_{PREV} - \Gamma_{PREPREV} = \Gamma' \ldots\ldots\ldots (11)$$

$$\Gamma - \Gamma' = \Gamma_{WAIT} \ldots\ldots\ldots (12)$$

The equations (9) and (10) are denotes the update time of current and previous data item. If Γ' with waiting period (Γ_{WAIT}) is greater than the Γ value, the Di is a valid data item, otherwise the Di is an invalid data item. The equation (12) denotes the condition for valid data item, D_V and the equation (13) denotes the condition for invalid data item, D_{IV}.

$$D_V = \Gamma < \Gamma' + \Gamma_{WAIT} \ldots\ldots\ldots (13)$$

$$D_{IV} = \Gamma > \Gamma' + \Gamma_{WAIT} \ldots\ldots\ldots (14)$$

3.7 Tuning of Parameters

3.7.1 TTL tuning

The TTL value has great impact on the network performance in terms of query delay. TTL is the factor that determines the update rate of the data. Smaller value of TTL indicates the higher data update rate. The data item, Di with higher update rate is invalidated sooner resulting in increasing the average query delay. Smaller TTL value of Di decreases the cache hit ratio. If the cache has data with less TTL value, most of the data are invalidated and hence, cache hit ratio is reduced. The delay in HAC scheme gradually decreases with the increase in TTL value. The cooperativeness of all nodes in the network is increased to attain the maximum benefits of high TTL value (low data update rate). The cache memory of the TB is much effectively used if it caches Di with small TTL. PB is effectively utilized if it caches Di with a large TTL value. Therefore, it is important to fine tune the threshold value to categorize and cache the data effectively. The threshold value of the TTL is denoted as TH_{TTL}. The value of TH_{TTL} should be neither low nor high. The TH_{TTL} is selected such that it achieves the least query delay.

3.7.2 Data item size tuning

Data size is an important criterion for caching. In HAC, if Di is small, it is cached in TB using cache nodes. Similarly, if Di is large, it is cached in PB. The threshold value for data item size is denoted as "TH_{DS}". If TH_{DS} is too small, HAC may fail to cache some small important data. If TH_{DS} is too large, HAC caches all the data including the data with least importance. As the destination holds data with heterogeneous data sizes, it is much important to derive an optimal value for TH_{DS}. The value of TH_{DS} should be neither low nor high. The size of the data item varies from DS_{Min} and DS_{Max}. TH_{DS} is a measure of percentage of summation of DS_{Min} and DS_{Max}. TH_{DS} can also be considered as the function of D_{QRY}.

3.8 Pseudo code of HAC mechanism

Any node (N1, ……., Nn) in the network:

BEGIN

When a Dnew arrives:

if the Mi space is available

Check DS and TTL of Dnew

If (DSnew<TH_{DS} && D_{TTL} < TH_{TTL})

then cache Dnew in TB

else cache Dnew in PB

else there is no space available

when there is a set of data items Di such that (|Di|> $(D_{TTL\,(i)})]_{MAX} \| A_{Di\,(MIN)}$))

replace Di with Dnew

else if check the freshness of the set of cached data items Di

find the update period of Di

if there is cached Di such that ($\Gamma < \Gamma' + \Gamma_{WAIT}$)
validate Di
else invalidate Di and replace Di with Dnew
END

4. Simulation set up

The proposed work is simulated in the network simulator NS-2. The number of nodes considered in the proposed work is 'n'. The internet system or LAN is connected to the base station via wired link. The mobile nodes are connected to the base station via wireless links. During simulation, some of the mobile nodes are set one hop away from the base station and the remaining nodes are set to be multi hop away from the base station. The proposed work is mainly based on the parameter called update_interval, and it ranges from 10 to 1000 seconds. It depends on the accessibility rate of data items. The size of TB is smaller than the PB. The simulation runs for 3000 seconds. In a mobile node, ¼ th of memory is allocated to TB and the remaining memory is allocated to PB.

Destination Node Cache Memory = M_D
Node Memory = 0.002 * M_D = M_i
Size of TB for node N_1 = 0.25 * M_i
Size of PB for node N_1 = 0.75 * M_i

The table 2 explains the data structure of each mobile node's information.

Software for simulation	Network simulator 2.
Channel	Wireless
Simulation run time	100 seconds
Node density Area	600 X 600
Packet size of small data	50bytes
Packet size of large data	5000bytes
Speed	50m/s
Routing Protocol	AODV
Propagation model	TwoRayGround
Mobility Model	Random Waypoint Model
Network Interface Type	Wireless Physical
MAC Type	Mac/802.11
Antenna Type	Omni Antenna
Number of mobile nodes	Any number of nodes
b_{br}	7000bps
BW	100Mbps

Table 2: Simulation Parameters

5. Performance Evaluation

The proposed work is simulated in NS-2. The effectiveness of the proposed HAC is evaluated in this section.

5.1 Effect of update_interval on cache hit ratio

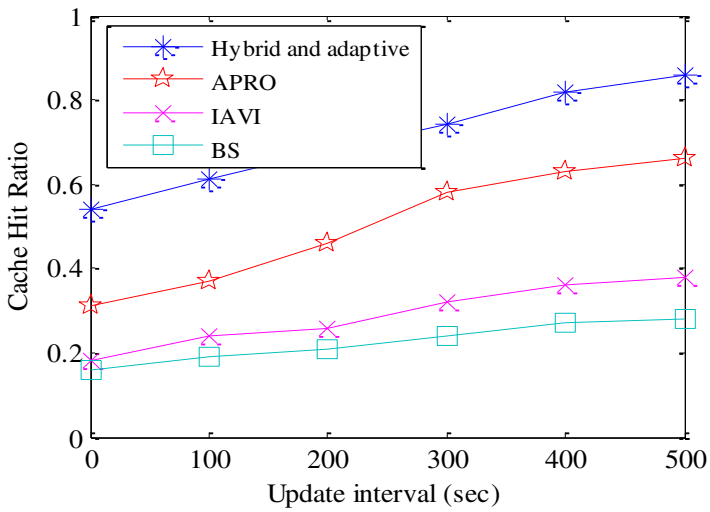

Figure 3: Update_interval Vs Cache hit ratio

The Fig. 3 shows the comparison performed based on the Cache Hit Ratio (CHR) between the proposed Hybrid and Adaptive Caching (HAC) scheme with the already existing scheme called as Adaptive Proactive Caching (APC). It is clearly depicted that the proposed approach achieves high cache hit ratio than existing approaches. The reason is the proposed scheme adaptively caches the data item in two separate buffers according to the size of the data item among the mobile nodes in the network. The cache is updated for two different TTL (longer and shorter) update interval, where the frequently accessed data are most likely to be updated in short TTL. This is because; small data item in a temporary buffer is updated for a short interval time whereas the large data item in permanent buffer is updated for a long interval of time. As the update interval for data item is increases, most of the request is satisfied in cache with less access latency. Due to the proper updation of cached data item adaptively, reduces the cache misses that conventionally improves the cache hit ratio in the proposed approach.

5.2 Effect of Update_interval on query latency

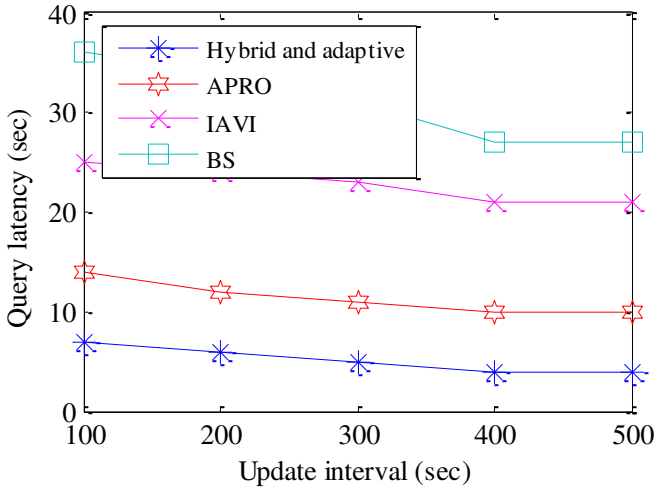

Figure 4: Update_interval Vs Query Latency

The Fig.4 shows the relationship between update interval and the system performance metric in terms of query latency for existing adaptive proactive caching and the proposed hybrid and adaptive caching scheme. In the case of a small data item, query latency decreases according to the increasing update interval. If the update interval for data item is increases, then most of the request is satisfied with less access latency.

The proposed scheme takes only 5 seconds for accessing the data objects for the requested data object which is much less than Adaptive Proactive Caching (APC) scheme that takes greater than 15 seconds. This indicates that caching data objects in two different buffers according to the size of the data item is an effective method to increase system performance.

The performance of proposed scheme is always better than that of APC based schemes for update interval in the range of 100 to 1000secs. The proposed approach schemes have more than 50% performance gain in terms of cache hit ratio and query latency over existing APC based schemes, thus demonstrating their superiority in a mobile environment.

5.3 Effect of database

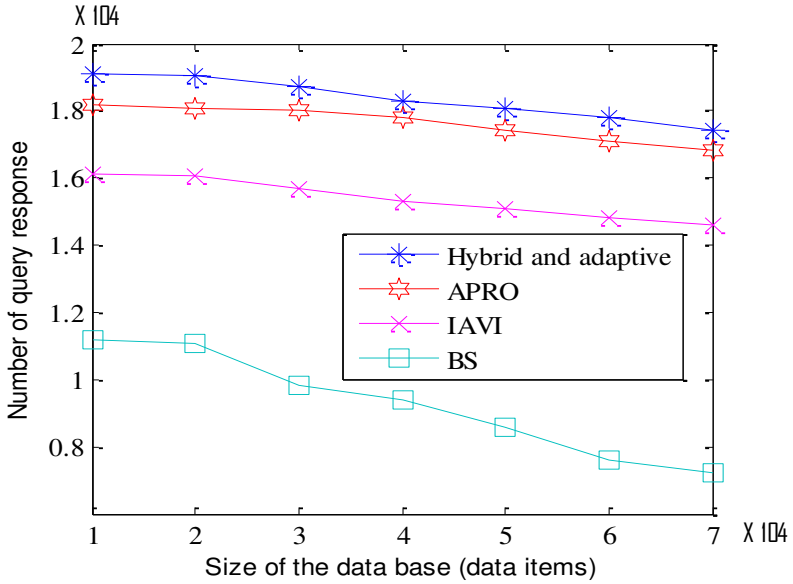

Figure 5: Database size Vs Number of Query response

The Fig.5 shows that the relation between the size of the database and number of queries satisfied. In Existing Bit sequence [15], cache invalidation technique is performed based on the update aggregation technique for the group of data items either a large data item (video) or small data item (stock) and also only one timestamp is associated for all data item in the same report. Therefore, the number of query response is ultimately decreased according to the size of the data item. In the proposed approach, the mobile node adaptively caches the data item in two separate buffers depending upon the size of the data object. Therefore, it certainly improves the number of query response according to the increasing number of data item better than the already existing approaches.

5.4 Effect of disconnection probability

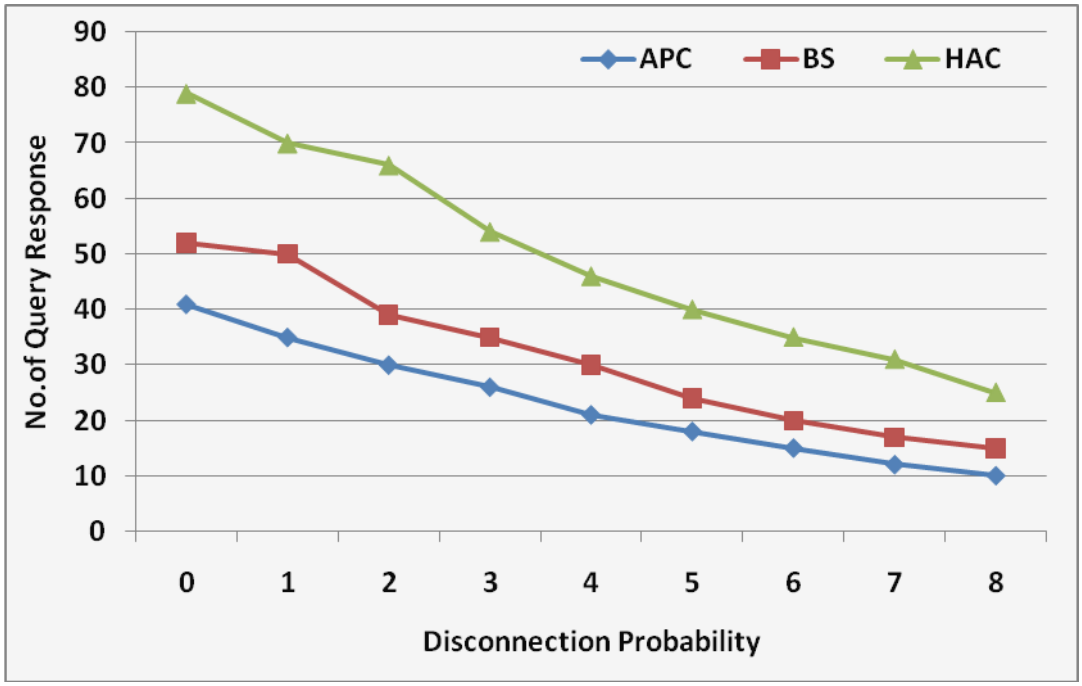

Figure 6: Disconnection probability Vs Number of query response

Finally, the effect of client disconnection is studied by varying the disconnection probability from 0 to 8 as shown in the Fig.6. The network throughput in terms of the number of queries responded decreases when the network disconnectivity rate increases. Due to the random mobility of the mobile nodes, the network gets disconnected frequently. The performance of the proposed hybrid and adaptive cache mechanism is compared with the bit sequence cache invalidation technique. The existing scheme uses the bit sequences to identify the disconnected mobile nodes in a particular time stamp. In the proposed approach, client disconnection handling protocol is newly developed to manage the performance of the system under the increasing rate of disconnection probability. Therefore, the proposed approach improves query responses according to the varying disconnection probability.

5.5 Effect of buffer size

Increment of buffer size in both temporary buffer and permanent buffer on the system performance is evaluated using the graph depicted in figure 7. Increment of buffer size results in high cache hit ratio. When the percentage of buffer size increases, the number of cached data items in the local buffer also increases correspondingly. Therefore increment in buffer size facilitates the high probability of availability of required data item in local buffer which results in high cache hit ratio. Large number of storage of data items with smaller TTL in temporary buffer do not affect the cache hit ratio since the data items are updated with appropriate update interval corresponding to the TTL value of smaller data item. Obviously the increment of buffer size in permanent buffer increases the cache hit ratio due to the availability of data item in the local cache is high.

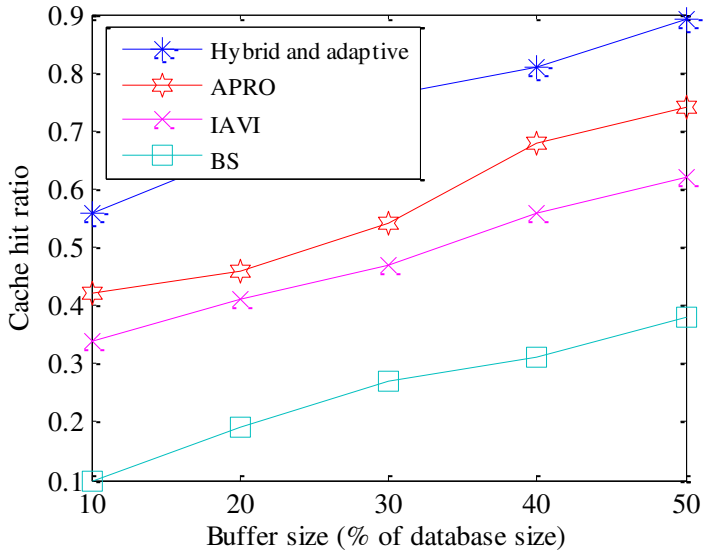

Figure 7: Percentage of increment in Buffer Size Vs Cache hit ratio

CONCLUSION

This paper presents an effective hybrid and adaptive caching technique for the hybrid mobile computing environment. This paper ensures data availability and effective bandwidth utilization. This proposal varies from existing proposals in two ways: i) hybrid architecture is constructed using both the conventional fixed infrastructure and ad hoc communication model. ii) Adaptive caching is performed in two storage systems such as Temporary Buffer (TB) and Permanent Buffer (PB) based on the size of the data item. The mobile node generates the query for the requested data item and decides whether to forward query to the base station or neighbor peers based on the size of the data item. The proposed work uses the cache memory effectively on placing the data on appropriate buffer. The proposed approach effectively uses the bandwidth through adaptive request forwarding in hybrid network architecture. The proposed work ensures data availability. The cache replacement technique is also adaptive and replaces the data effectively based on the TTL value. The proposed work invalidates the cache effectively using the timestamp values of the cached data. The threshold values used in the proposed work are fine tuned. The analytical and simulation result proves that the proposed work outperforms the existing systems.

REFERENCE

[1] Bin Tang, Himanshu Gupta, and Samir R. Das. 2020, "Benefit-based Data Caching in Ad Hoc Networks", IEEE transaction on mobile computing, Volume 7, Issue 3, pp. 289- 304. Digital Object Identifier : 10.1109/TMC.2007.70770

[2] Chi-Yin Chow, Hong Va Leong, and Alvin Chan. 2019, "Peer-to-Peer Cooperative Caching in Mobile Environments" IEEE computer society Proceedings of the 24th International Conference on Distributed Computing Systems Workshops.

[3] Chi-Yin Chow, Hong Va Leong, Member, and Alvin T. S. Chan. 2017, "GroCOCA: Group-based Peer-to-Peer Cooperative Caching in Mobile Environment" IEEE journal on selected areas in communications, Volume 25, Issue 1, pp. 179- 191.

[4] Daniel Barbari and Tomasz Imieliriski. 1995, “Sleepers and Caching Strategies in Workaholics: Mobile Environments”, The VLDB Journal — The International Journal on Very Large Data Bases, Volume 4, Issue 4, pp. 567- 602.

[5] Evaggelia Pitoura and Bharat Bhargava, 1995, “Maintaining Consistency of data in Mobile Distributed Environments”, IEEE proceedings of the 15th international conference on distributed computing system, pp. 404- 413. Digital Object Identifier : 10.1109/ICDCS.1995.500045

[6] Guohong Cao. 2002, “Proactive Power-Aware Cache Management for Mobile Computing Systems”, IEEE transactions on computers, Volume 51, Issue 6, pp. 608-62. Digital Object Identifier : 10.1109/TC.2002.1009147

[7] Guohong Cao 2003, “A Scalable Low-Latency Cache Invalidation Strategy for Mobile Environments”, IEEE transactions on knowledge and data engineering, Volume 15, Issue 5, pp. 1251- 1265.

[8] Guohong Cao and Chita Das. 2001, “On the Effectiveness of a Counter-Based Cache Invalidation Scheme and its Resiliency to Failures in Mobile Environments”, proceedings of the 20th IEEE symposium on reliable distributed systems, pp. 247- 256

[9] Haibo Hu, Jianliang Xu, Wing Sing Wong, Baihua Zheng, Dik Lun Lee, and Wang Chien Lee. 2005, “Proactive Caching for Spatial Queries in Mobile Environments”, ACM proceedings of the 21st international conference on data engineering, pp. 403-414.

[10] Hao Che, Ye Tung, and Zhijun Wang. 2002, “Hierarchical Web Caching Systems: Modeling, Design and Experimental Results”, IEEE journal on selected areas in communications, Volume 20, Issue 7, pp. 1305- 1314, 2002. Digital Object Identifier : 10.1109/JSAC.2002.801752

[11] Hui Song, and Guohong Cao. 2005, “Cache-Miss-Initiated Prefetch in Mobile Environments”, Elsevier transaction on computer communications, Volume 28, pp. 741- 753.

[12] Ismail Ari, Ahmed Amer, Robert Gramacy, Ethan L. Miller, Scott A. Brandt, and Darrell D. E. Long. 2002, “ACME: Adaptive Caching Using Multiple Experts”, proceedings in informatics, Volume 14, Carleton Scientific.

[13] Jianliang Xu, Qinglong Hu, Wang-Chien Lee, and Dik Lun Lee. 2004, “Performance Evaluation of an Optimal Cache Replacement Policy for Wireless Data Dissemination”, IEEE transactions on knowledge and data engineering, Volume 16, Issue 1, pp. 125- 139.

[14] Jiannong Cao, Yang Zhang and Guohong Cao, and Li Xie. 2007, “Data Consistency for Cooperative Caching in Mobile Environments”, IEEE transaction on computer, Volume 40, Issue 4, pp. 60- 66.

[15] Jin Jing, Ahmed Elmagarmid, Abdelsalam (Sumi) Helal and Rafael Alonso. 1997, “Bit-Sequences: An Adaptive Cache Invalidation Method in Mobile Client/Server Environments”, Mobile Networks and Applications, Volume 2, pp. 115–127.

[16] Jing Zhao, Ping Zhang, Guohong Cao, Chita R. Das. 2010, “Cooperative Caching in Wireless P2P Networks: Design, Implementation, and Evaluation", IEEE Transactions on Parallel and Distributed Systems, Volume 21, No 2, pp 229-241.

[17] Li Fan, Pei Cao, Jussara Almeida, and Andrei Z. Broder. 2000, “Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol”, IEEE/ACM transactions on networking, Volume 8, Issue 3, pp. 281- 293. doi>10.1109/90.851975

[18] Liangzhong Yin, Guohong Cao. 2006, "Supporting Cooperative Caching in Ad Hoc Networks" , IEEE Transactions on Mobile Computing, Volume 5, No 2, pp 77-89.

[19] Liangzhong Yin, Guohong Cao, and Ying Cai. 2005, “A Generalized Target-Driven Cache Replacement Policy for Mobile Environments”, Elsevier Transaction on Parallel Distributing Computing, Volume 65, pp. 583- 594.

[20] Liangzhong Yin, Guohong Cao, Chita Das, and Ajeesh Ashraf. 2002, “Power-Aware Prefetch in Mobile Environments”, ACM proceedings of the 22nd international conference on distributed computing systems, page 571, 2002

[21] Miguel Castro, Atul Adya, Barbara Liskov, and Andrew C. Myers. 1997, “HAC: Hybrid Adaptive Caching for Distributed Storage Systems” Proceedings of the 16th ACM Symposium on Operating Systems Principles, pp. 102-115, Volume 31, Issue 5.

[22] Mohamed Zahran. 2004, “Cache Replacement Policy Revisited”, IEEE international symposium on cluster computing and the grid, pp. 182- 189.

[23] Negin Golrezaei, Karthikeyan Shanmugam, Alexandros G. Dimakis,Andreas F. Molisch, Giuseppe Caire. 2012, "FemtoCaching: Wireless Video Content Delivery through Distributed Caching Helpers", Proceedings IEEE INFOCOM, pp 1107-1115.

[24] Stratis Ioannidis, Laurent Massoulie, Augustin Chaintreau. 2010, "Distributed Caching over Heterogeneous Mobile Networks",Proceedings of the ACM SIGMETRICS international conference on Measurement and modeling of computer systems, pp 311-322. doi>10.1145/1811039.1811075

[25] Wei Gao, Guohong Cao, Arun Iyengar, Mudhakar Srivatsa. 2011, "Supporting Cooperative Caching in Disruption Tolerant Networks", 1st International Conference on Distributed Computing Systems, pp 151-161.

[26] Wei Gao, Guohong Cao, Mudhakar Srivatsa, Arun Iyengar. 2012, "Distributed Maintenance of Cache Freshness in Opportunistic Mobile Networks", 32nd IEEE International Conference on Distributed Computing Systems, pp 132-141.

[27] Yeim-Kuan Chang, I-Wei Ting and Tai-Hong Lin. 2008, “Dynamic Cache Invalidation Scheme in IR-based Wireless Environments”, 22nd International Conference on Advanced Information Networking and Applications, pp. 697- 704.

[28] Zhijun Wang, Mohan Kumar, Sajal K Das, and Huaping Shen. 2006, “Dynamic Cache Consistency Schemes for Wireless Cellular Networks” IEEE transactions on wireless communications, Volume 5, Issue 1. Digital Object Identifier : 10.1109/TWC.2006.1611060

Figures

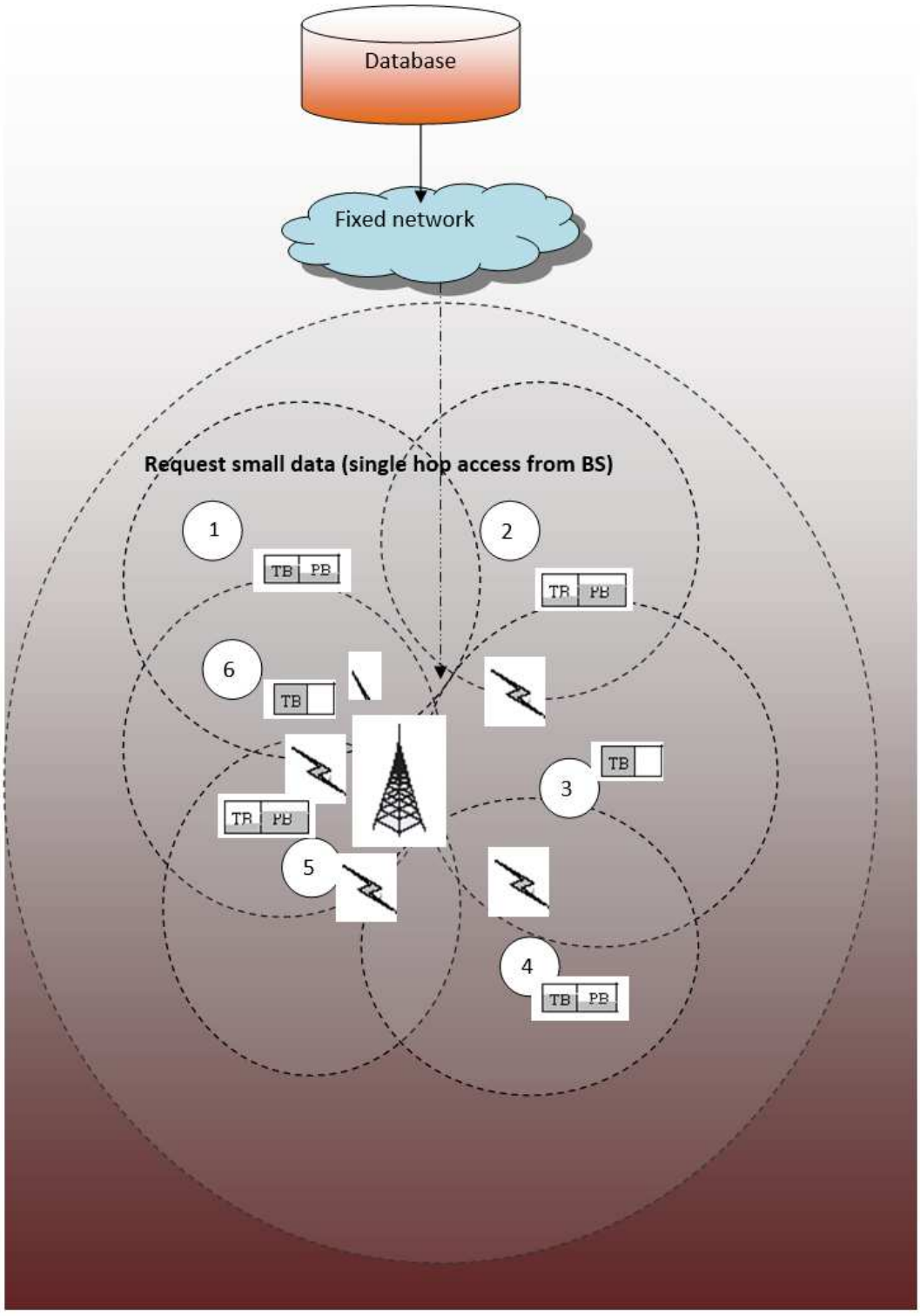

Figure 1

Hybrid Adaptive Caching Mobile Environment

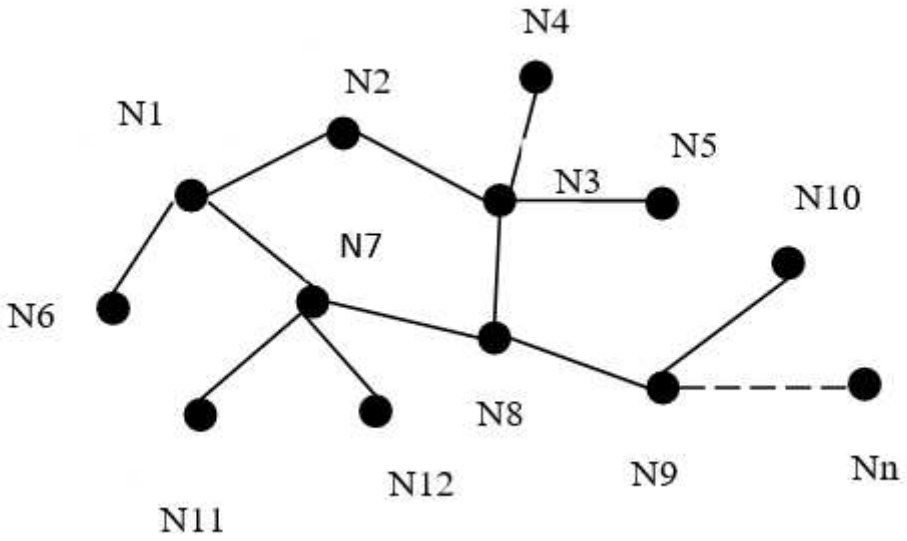

Figure 2

Ad hoc network

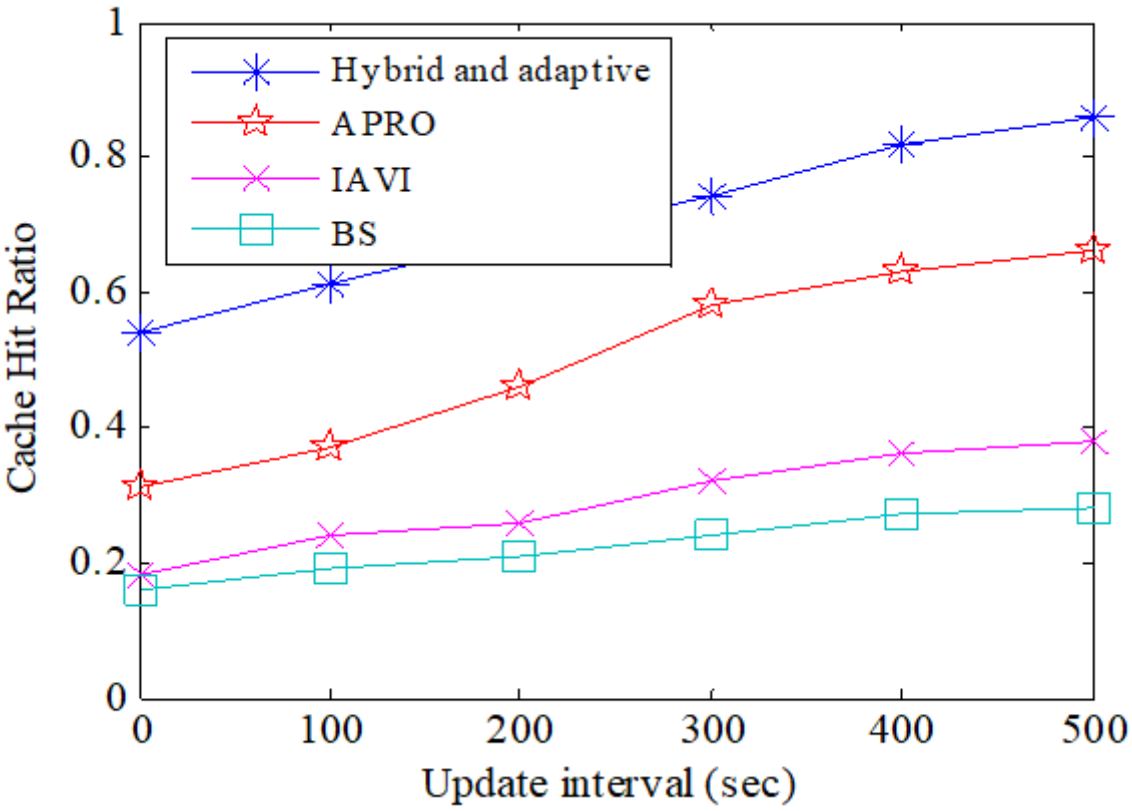

Figure 3

Update_interval Vs Cache hit ratio

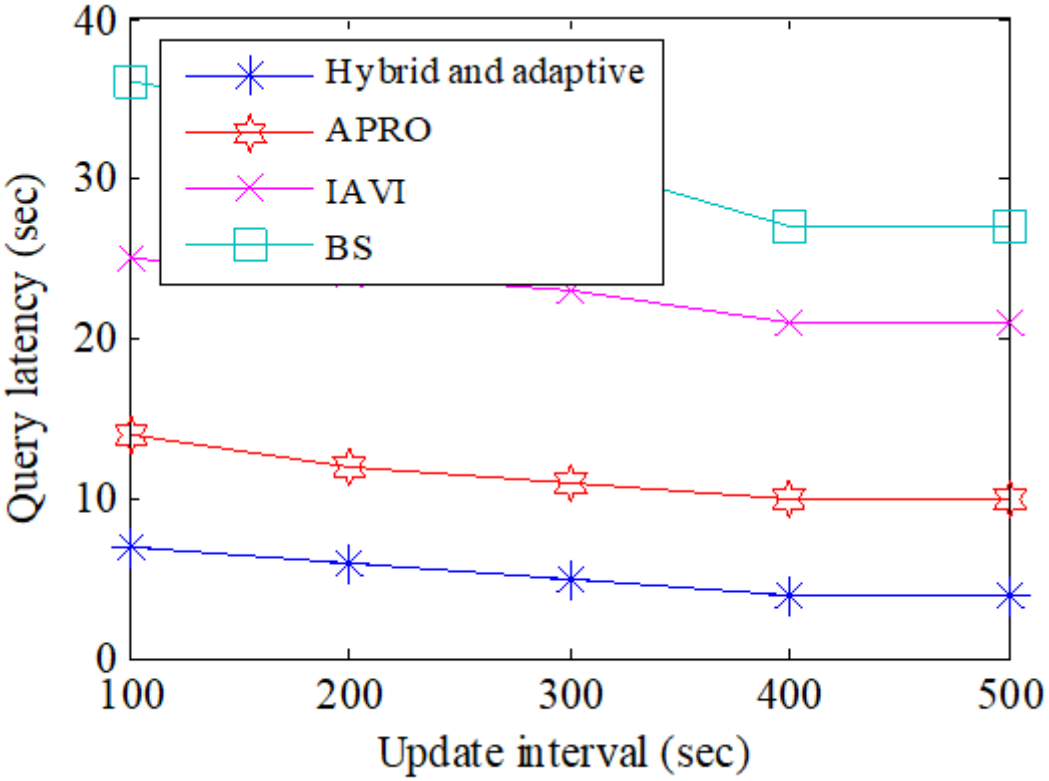

Figure 4

Update_interval Vs Query Latency

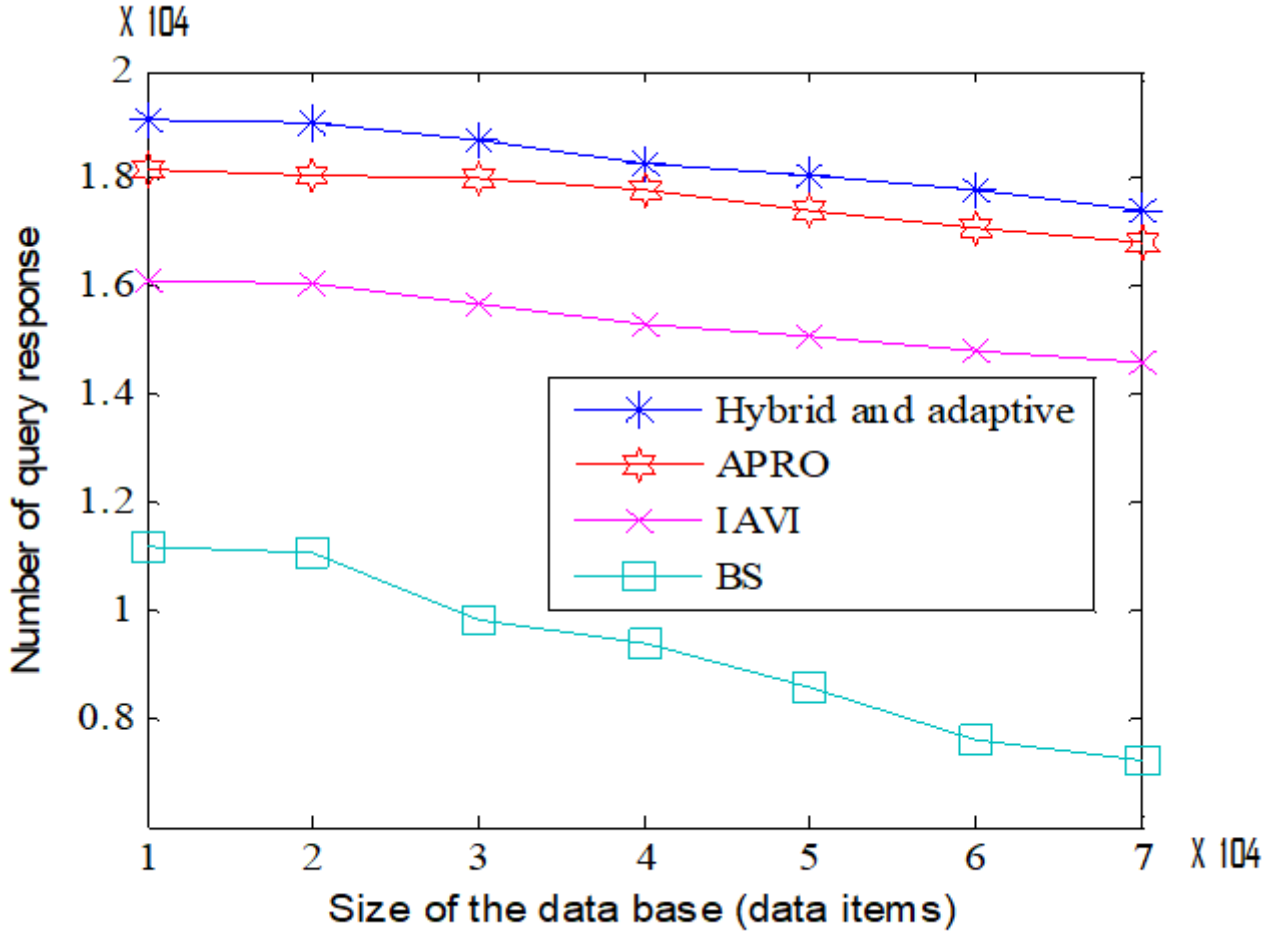

Figure 5

Database size Vs Number of Query response

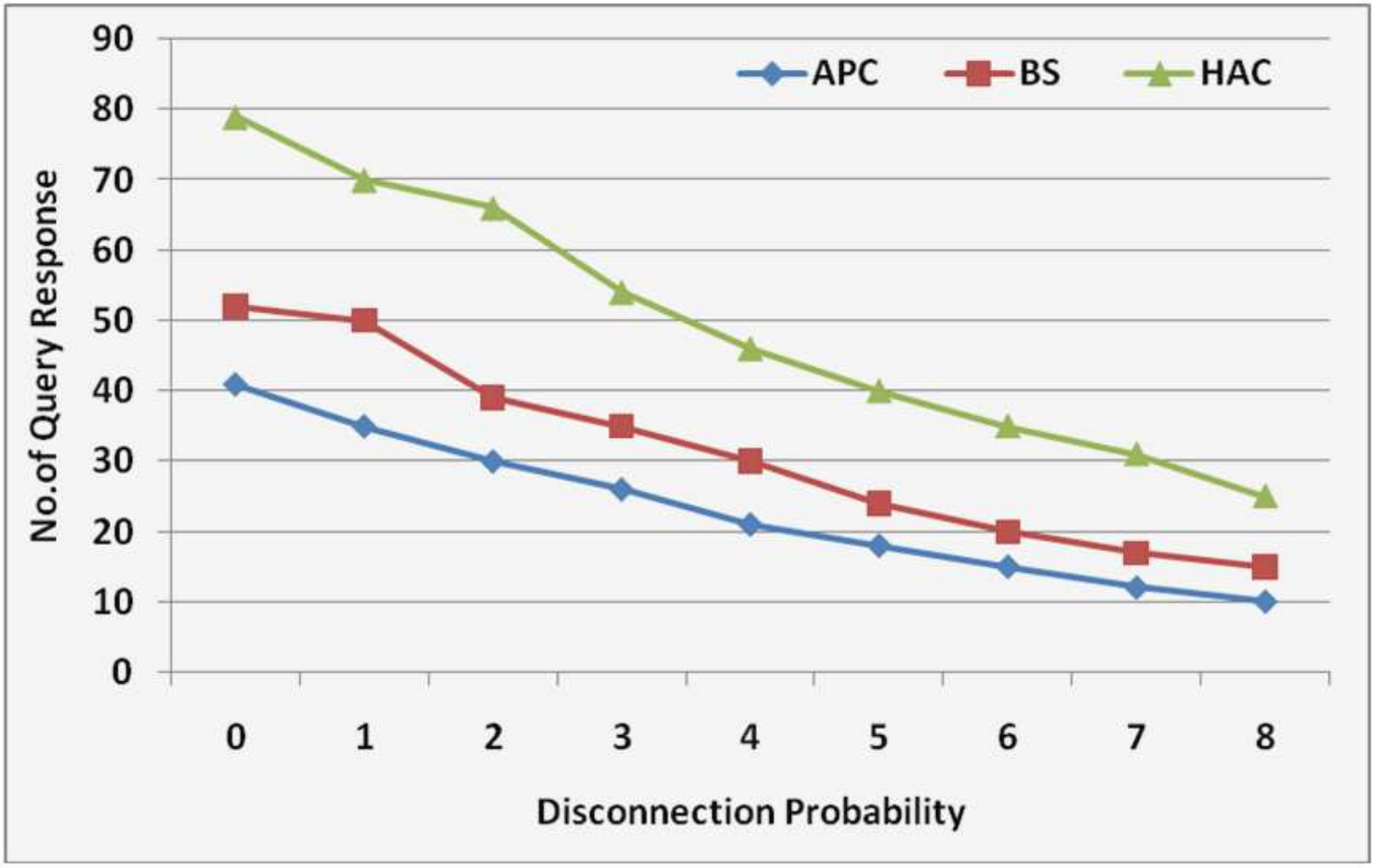

Figure 6

Disconnection probability Vs Number of query response

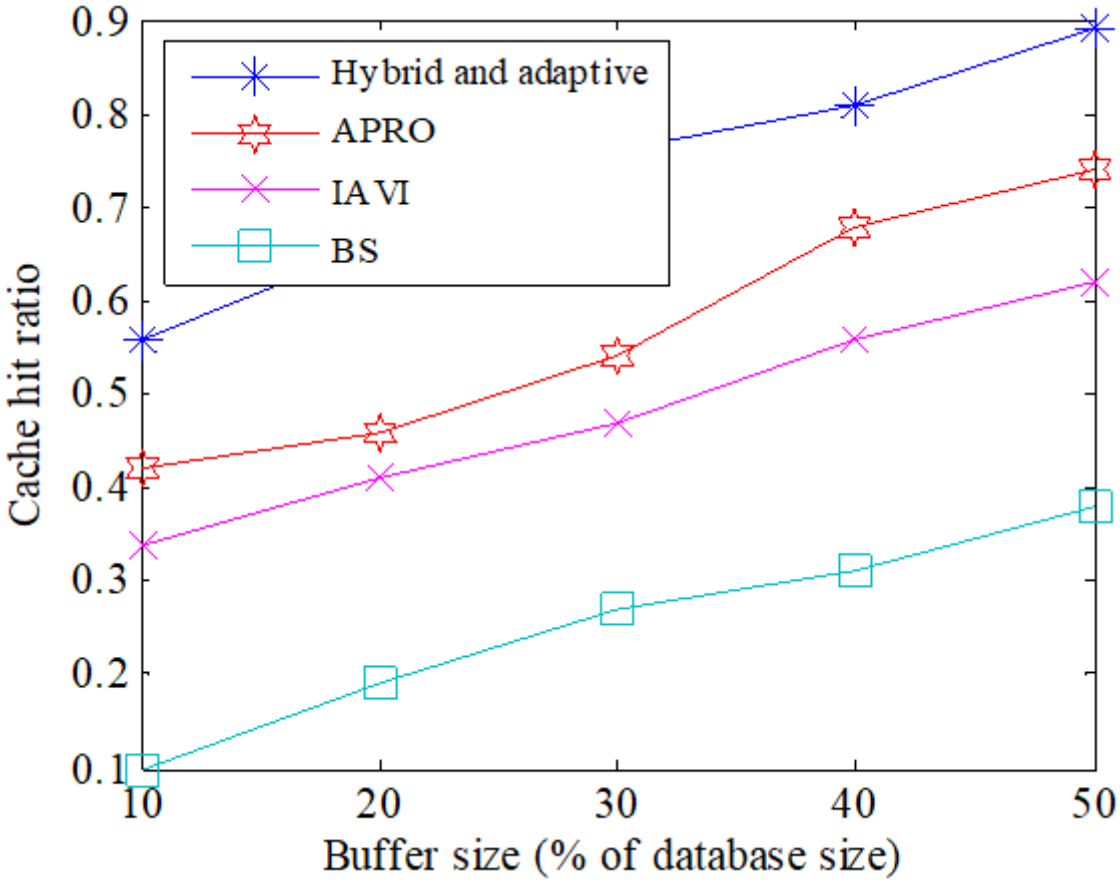

Figure 7

Percentage of increment in Buffer Size Vs Cache hit ratio