

Delay Reduction by Providing Location Based Services using Hybrid Cache in peer to peer Networks

C. Gopala Krishnan¹, A. Rengarajan², R. Manikandan³

¹SRResearch Scholar, St. Peter's University, Department of CSE
Chennai, Tamilnadu, India

[e-mail: skywarekrish@gmail.com]

²Professor, Vel Tech Multi Tech Dr.RR Dr.SR Engg College, Department of CSE
Chennai, Tamilnadu, India

[e-mail: rengu_rajana@yahoo.com]

³Research Scholar, St.Peter's University, Department of CSE
Chennai, Tamilnadu, India

[e-mail: skywarekrish@gmail.com]

*Corresponding author: Gopala Krishnan

*Received November 19, 2014; revised March 28, 2015; accepted May 3, 2015;
published June 30, 2015*

Abstract

Now a days, Efficient processing of Broadcast Queries is of critical importance with the ever-increasing deployment and use of mobile technologies. BQs have certain unique characteristics that the traditional spatial query processing in centralized databases does not address. In novel query processing technique, by maintaining high scalability and accuracy, latency is reduced considerably in answering BQs. Novel approach is based on peer-to-peer sharing, which enables us to process queries without delay at a mobile host by using query results cached in its neighboring mobile peers. We design and evaluate cooperative caching techniques to efficiently support data access in ad hoc networks. We first propose two schemes: Cache Data, which caches the data, and Cache Path, which caches the data path. After analyzing the performance of those two schemes, we propose a hybrid approach (Hybrid Cache), which can further improve the performance by taking advantage of Cache Data and Cache Path while avoiding their weaknesses. Cache replacement policies are also studied to further improve the performance. Simulation results show that the proposed schemes can significantly reduce the query delay and message complexity when compared to other caching schemes.

Keywords: *Hybrid cache, cache management, cache replacement policy, ad hoc networks, data dissemination, simulations*

1. Introduction

Broadcast query processing is becoming an integral part of many new mobile applications. Recently, there has been a growing interest in the use of broadcast queries (BQs), which represent a set of queries that retrieve information based on mobile users' current locations [1,2].

Novel query processing techniques must be devised to handle the following new challenges:

1. Mobile query semantics.

In a mobile environment, a typical BQ is of the form "*find the top-three nearest hospitals.*" The result of the query depends on the location of its requester.

Caching and sharing of query results must take into consideration the location of the query issuer.

2. High workload.

The database resides in a centralized server, which typically serves a large mobile user community through wireless communication. Consequently, bandwidth constraints and scalability become the two most important design concerns of BQ algorithms [2].

3. Query promptness and accuracy.

Due to user mobility, answer to a BQ will lose their relevancy if there is a long delay in query processing or in communication. For example, answers to the query "*find the top-three nearest hospitals*" received after 5 minutes of high-speed driving will become meaningless. Instead, a prompt, albeit approximate, top-three nearest hospitals, may serve the user much better. This is an importance issue, as a long latency in a high workload wireless environment is not unusual.

The wireless environment and the communication constraints play an important role in determining the strategy for processing BQs. In the simplest approach, a user establishes a point-to-point communication with the server so that the user queries can be answered on demand. However, this approach suffers from several drawbacks. First, it may not scale to very large user populations. Second, to communicate with the server, a client must most likely use a fee-based cellular-type network to achieve a reasonable operating range. Third, users must reveal their current location and send it to the server, which may be undesirable for privacy reasons [12]. A more advanced solution is the wireless broadcast model [1, 9, 18]. It can support an almost-unlimited number of mobile hosts (MHs) over a large geographical area with a single transmitter. With the broadcast model, MHs do not submit queries. Instead, they tune in to the broadcast channel for information that they desire. Hence, the user's location is not revealed. One of the limitations of the broadcast model is that it restricts data access to be sequential. Queries can only be fulfilled after all the required on-air data arrives. This is why in some cases, a 5-minute delay to the query "*find the top-three nearest hospitals*" would not be unusual.

Alleviating this limitation, we propose a scalable low-latency approach for processing BQs in broadcast environments. Our approach leverages ad hoc networks to share information among mobile clients in a peer-to-peer (P2P) manner [10, 11].

The rationale behind our approach is based on the following observations:

- As mentioned previously, when a mobile user launches a nearest neighbor (NN) query, in many situations, the user would prefer an approximate result that arrives with a short response time rather than an accurate result with a long latency.
- The results of spatial queries often exhibit spatial locality. For example, if two MHs are close to each other, the result sets of their spatial queries may overlap significantly. Query results of a mobile peer are valuable for two reasons: 1) they can be used to answer queries of the current MH directly and 2) they can be used to dramatically reduce the latency for the current MH relative to on-air information.
- P2P approach can be valuable for applications where the response time is an important concern. Through mobile cooperative caching [5] of the result sets, query results can be efficiently shared among mobile clients. BQs concentrate on two common types of Query searches, namely, kNN queries and window queries (WQs). The contributions of existing study are given as follows:
 1. Identify certain characteristics of BQs that enable the development of effective sharing methods in broadcast environments. We introduce a set of algorithms that verify whether data received from neighboring clients are complete, partial, or irrelevant answers to the posed query.
 2. Utilize a P2P-based sharing method to improve the current approaches in answering on-air kNN queries and WQs.
 3. Evaluate BQ approach through a probabilistic analysis of the hit ratio in sharing. In addition, through extensive simulation experiments, we evaluate the benefits of our approach with different parameter sets.

2. Wireless Data Broadcast

In general, there are two approaches for mobile data access. One is the *on-demand access model*, and the other is the *wireless broadcast model*. For the on-demand access model, point-to-point connections are established between the server and the mobile clients, and the server processes queries that the clients submit on demand. For the wireless broadcast model, the server repeatedly broadcasts all the information in wireless channels, and the clients are responsible for filtering the information. The advantage of the broadcast model over the on-demand model is that it is a scalable approach. However, the broadcast model has large latency, as clients have to wait for the next broadcast cycle broadcasting cycle. If a client misses the packets that it needs, it has to wait for the next broadcast cycle.

To facilitate information retrieval on wireless broadcast channels, the server usually transmits an index structure, along with data objects. A well-known broadcast index structure is the (1, m) [9] indexing allocation method. As we can see in Fig. 1, the whole index is broadcast preceding every $1/m$ fraction of the data file. Because the index is available m times in one cycle, it allows a mobile client easy access to the index so that it can predict the arrival time of its desired data in a timely manner, and once it knows the arrival time, it only needs to tune into the broadcast channel when the data bucket arrives. This mechanism is important for battery-based devices.

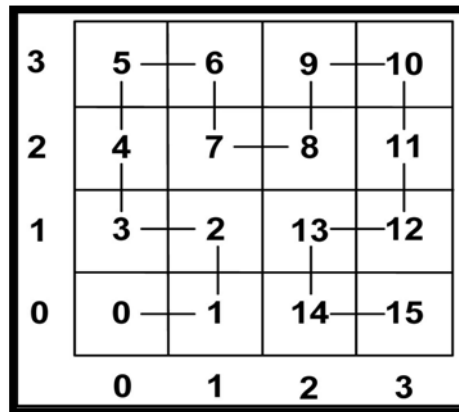


Fig. 1. The Hilbert-curve-based index structure. The numbers represent index values.

Thus, the general access protocol for retrieving data on a wireless broadcast channel involves three main steps [15]:

- The initial probe. A client tunes in to the broadcast channel and determines when the next index segment will be broadcast.
- Index search. The client accesses a sequence of pointers in the index segment to figure out when to tune in to the broadcast channel to retrieve the required data.
- Data retrieval. The client tunes in to the channel when packets containing the required data arrive and then downloads all the required information.

However, nearly all the existing spatial access methods are designed for databases with random access disks. These existing techniques cannot be used effectively in a wireless broadcast environment, where only sequential data access is supported. Zheng et al. [19] proposed indexing the spatial data on the server by a space-filling curve. The Hilbert curve was chosen for this purpose because of its superior locality. The index values of the data packets represent the order in which these data packets are broadcast. For example, the Hilbert curve in Fig. 1 tries grouping data of close values so that they can be accessed within a short interval when they are broadcast sequentially. The MHs use on-air search algorithms [19] to answer LBSQs (kNN and WQs) over data that arrives in the order prescribed by the Hilbert curve.

2.1 Spatial Queries

Existing work focus on two common types of spatial queries, namely, kNNqueries and WQs [17]. With R-tree-based [15] spatial indices, depth-first search (DFS) and best first search (BFS)[8] have been the prevalent branch-and-bound techniques for processing NN queries. The DFS method recursively expands the index nodes for searching NN candidates. At each newly visited nonleaf node, DFS computes the ordering metrics for all its child nodes and applies pruning strategies to remove unnecessary branches.

When a leaf node is reached, the data objects are retrieved, and the NN candidates are updated. Comparatively, the BFS technique utilizes a priority queue to store nodes to be explored through the search process.

The nodes in the queue are sorted according to their minimum distance (MINDIST) to the query point. During the search process, BFS repeatedly dequeues the top entry in the queue and enqueues its child nodes with their MINDIST into the queue.

When a data entry is dequeued, it is inserted into the result set. For WQs that find objects within a specified area, the R-tree families[3,16] provide efficient access to disk-based databases. Basically, an R-tree structure groups objects close to each other into a minimum bounding rectangle (MBR), and a range query only visits MBRs that overlap with the query area.

2.2 Cooperative Caching

Caching is a key technique to improve data retrieval performance in widely distributed environments Hara and Madria proposed three data replica allocation methods in ad hoc networks by considering the access frequency from MHs to each data item and the status of the network connection With the increasing deployment of new P2P wireless communication technologies, P2P cooperative caching becomes an effective sharing alternative With this technique [4, 7]. MHs communicate with neighboring peers in an ad hoc manner for information sharing instead of relying solely on the communication between remote information sources.

P2P cooperative caching can bring about several distinctive benefits to a mobile system: improved access latency, reduced server workload, and alleviated point-to-point channel congestion. In this research, we leverage the P2P caching technique to alleviate the inherent access latency limitation in wireless broadcast environments.

2.3 Sharing-Based NN Queries

Fig. 2 shows an example of an on-air kNN query based on the Hilbert curve index structure. At first, by scanning the on-air index, the k-nearest object to the query point is found, and a minimal circle centered at q and containing all those k objects is constructed. The MBR of that circle, enclosing at least k objects, serves as the search range. Consequently, q has to receive the data packets that covers the MBR from the broadcast channel for retrieving its k-nearest objects.

As shown in Fig. 2, the related packets span a long segment in the index sequence, that is, between 5 and 58, which will require a long retrieval time. The other problem with this search algorithm is that the indexing information has to be replicated in the broadcast cycle to enable twice scanning. The first scan is for deciding the kNN search range, and the second scan is for retrieving k objects based on the search range [19].

Therefore, the Sharing-Based Nearest Neighbor (SBNN) query approach to improve the preceding on-air kNN query algorithm. The SBNN algorithm attempts to verify the validity of k objects by processing results obtained from several peers.

3. Overview

The wireless data broadcast model has good scalability for supporting an almost-unlimited number of clients [9]. Its main limitation lies in its sequential data access: the access latency becomes longer as the number of data items increases. If we can provide (approximate) answers to spatial queries before the arrival of related data packets, we will overcome the limitation of the broadcast model.

A novel component in the methodology is a verification algorithm that verifies whether a data item from neighboring peers is part of the solution set to a spatial query. Even if the verified results constitute only part of the solution set, in which case the query client needs to wait for the required data packets to get the remaining answers, the partial answer can be

utilized by many applications that do not need exact solutions but require a short response time (for example, the query “What are the top-three nearest hospitals?” issued by a motorist on a highway).

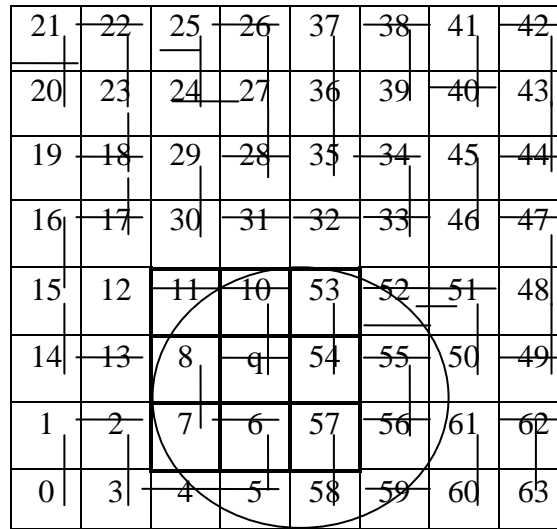


Fig. 2. An on-air kNN query example. The numbers represent index values

3.1 Nearest Neighbor Verification

When an MH q executes SBNN, it first broadcasts a request to all its single-hop peers for their cached spatial data. Each peer that receives the request returns the verified region MBR and the cached POIs to q . Then, q combines the verified regions of all the replying peers, each bounded by its MBR, into a merged verified region MVR (the polygon in Fig. 3). The merging process is carried out by the MapOverlay algorithm [6] (line 4 in Algorithm 1). The core of SBNN is the NN verification (NNV) method, whose objective is to verify whether a POI o_i obtained from peers is a valid (that is, the top- k) NN of the MH q .

Let IP denote the data collected by q from j peers p_1, \dots, p_j . Consequently, the merged verified region MVR can be represented as

$$MVR = p_1.VR \cup p_2.VR \cup \dots \cup p_j.VR$$

Suppose that the boundary of MVR consists of k edges, $IE = \{e_1, e_2, \dots, e_k\}$, and there are l POIs, $O = \{o_1, o_2, \dots, o_l\}$ inside the MVR. Let $e_s \in IE$ be the edge that has the shortest distance to q . An example is given in Fig. 3, where $k=10$, and e_1 has the shortest distance to q .

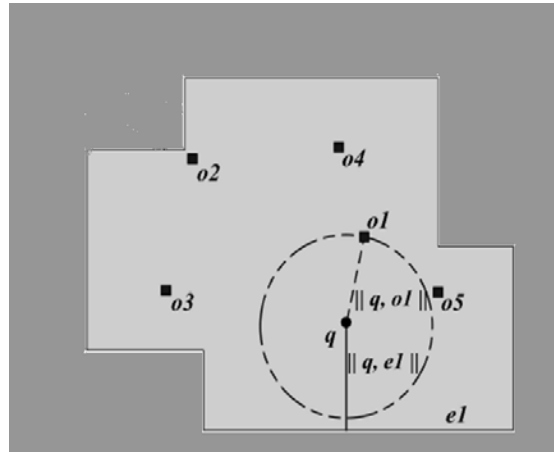


Fig. 3. Because $e1$ has the shortest distance to q and $\|q, o1\| \leq \|q, e1\|$, POI is verified as a valid NN of MH_q .

The NNV method uses a heap H to maintain the entries of verified and unverified POIs. Initially, H is empty. The NNV method inserts POIs to H as it verifies objects from MHs in the vicinity of q . The heap H maintains the POIs in ascending order in terms of their euclidean distances to q . nverified objects are kept in H only if the number of verified objects is lower than what was requested by the query. The NNV method is formalized in Algorithm 1. Since the verified-region merging process dominates the algorithm complexity, the NNV method can be computed in $O(n \log n + i \log n)$ time, where n is the total edge number of the two merged polygons, and i is the number of intersection points.

Algorithm : NNV(q, H, k)

- 1: $P \leftarrow$ peer nodes responding to the query request issued from q .
- 2: $MVR \leftarrow \Phi$
- 3: **for** $\forall p \in IP$ **do**
- 4: $MVR \cup= p.VR$ and $\mathbb{O} \cup p. \mathbb{O}$
- 5: **end for**
- 6: $\forall oi \in \mathbb{O}$, sort according to $\|q, oi\|$
- 7: Compute $\|q, es\|$, where edge es has the shortest distance to q among all the edges of MVR
- 8: $i = 1$
- 9: **while** $|H| < k$ and $i \leq |\mathbb{O}|$ **do**
- 10: **if** $\|q, oi\| \leq \|q, es\|$ **then**
- 11: $H.verified \cup = oi$
- 12: **else**
- 13: $H.verified \cup = oi$
- 14: $i++$
- 15: **end if**
- 16: **end while**
- 17: **return H**

If k elements in H are all verified by NNV, the kNN query is fulfilled. There will be cases when the NNV method cannot fulfill a kNN query. Hence, a set that contains unverified

elements is returned. If the response time is critical, a user may agree to accept a kNN data set with unverified elements, where the objects are not guaranteed to be the top kNNs. However, the correctness of these approximate results can be estimated and will be discussed in the next section. If the result quality is the most important concern, the client has to wait until it receives all the required data packets from the broadcast channel. Nevertheless, the partial results in H can be used to decrease the required data packets and thus speed up the on-air data collection.

3.2 Approximate Nearest Neighbor

We calculate the probability that the unverified i th NN o of a query point q is actually the true i th NN of q . The reason that o cannot be verified is because there is a region that is not covered by q 's neighboring peers. As long as a POI exists in the region, o cannot be q 's i th NN. We denote such a region as o 's unverified region. We assume that the POIs are Poisson distributed in our environment based on our experiments of several common POI types (gas stations, grocery stores, etc.) with chi-square tests [13, 14]. The probability of finding another POI in the unverified region U_i of an unverified POI o_i can be calculated with respect to the area of U_i . We formulate the correctness of an unverified POI based on the probability model

4. Proposed Basic Cooperative Cache Schemes

In this section, we propose two basic cooperative cache schemes and analyze their performance.

4.1 System Model

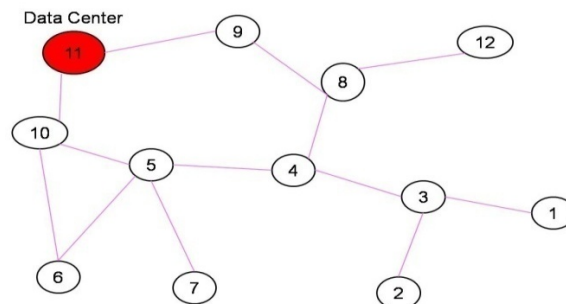


Fig. 4. System Model

Fig. 4.1 shows part of an ad hoc network. Some nodes in the ad hoc network may have wireless interfaces to connect to the wireless infrastructure such as wireless LAN or cellular networks. Suppose node N_{11} is a data source (center), which contains a database of n items $d_1; d_2; \dots; d_n$. Note that N_{11} may be a node connecting to the wired network which has the database. In ad hoc networks, a data request is forwarded hop-by-hop until it reaches the data center and then the data center sends the requested data back. Various routing algorithms have been designed to route messages in ad hoc networks. To reduce the bandwidth consumption and the query delay, the number of hops between the data center and the requester should be as small as possible. Although routing protocols can be used to achieve this goal, there is a limitation on how much they can achieve. In the following, we propose two basic cooperative caching schemes: **Cache Data and Cache Path**.

4.1.1 Cache the Data (CacheData)

In CacheData, the node caches a passing-by data item d_i locally when it finds that d_i is popular, i.e., there were many requests for d_i , or it has enough free cache space. For example, in Fig. 4.1, both N_6 and N_7 request d_i through N_5 , N_5 knows that d_i is popular and caches it locally. Future requests by N_3 , N_4 , or N_5 can be served by N_5 directly. Since CacheData needs extra space to save the data, it should be used prudently. Suppose the data center receives several requests for d_i forwarded by N_3 . Nodes along the path $N_3_N_4_N_5$ may all think that d_i is a popular item and should be cached. However, it wastes a large amount of cache space if three of them all cache d_i .

To avoid this, a conservative rule should be followed: A node does not cache the data if all requests for the data are from the same node. As in the previous example, all requests received by N_5 are from N_4 , which in turn are from N_3 . With the new rule, N_4 and N_5 do not cache d_i . If the requests received by N_3 are from different nodes such as N_1 and N_2 , N_3 will cache the data. If the requests all come from N_1 , N_3 will not cache the data, but N_1 will cache it. Certainly, if N_5 receives requests for d_i from N_6 and N_7 later, it may also cache d_i . Note that d_i is at least cached at the requesting node, which can use it to serve the next query.

This conservative rule is designed to reduce the cache Space requirement. In some situations, e.g., when the cache size is very large or for some particular data that are interested by most nodes, the conservative rule may decrease the cache performance because data are not cached at every intermediate node. However, in mobile networks, nodes usually have limited cache spaces and we do not assume that some data are interested by all nodes. Therefore, the conservative rule is adopted in this paper.

4.1.2 Cache the Data Path (CachePath)

The idea of CachePath can be explained by using Fig. 4.1 Suppose node N_1 has requested a data item d_i from N_{11} . When N_3 forwards the data d_i back to N_1 , N_3 knows that N_1 has a copy of d_i . Later, if N_2 requests d_i , N_3 knows that the data center N_{11} is three hops away whereas N_1 is only one hop away. Thus, N_3 forwards the request to N_1 instead of N_4 . Note that many routing algorithms (such as AODV and DSR) provide the hop count information between the source and destination. By caching the data path for each data item, bandwidth and the query delay can be reduced since the data can be obtained through fewer number of hops. However, recording the map between data items and caching nodes increases routing overhead.

In the following, we propose some optimization techniques. When saving the path information, a node need not save all the node information along the path. Instead, it can save only the destination node information, as the path from current router to the destination can be found by the underlying routing algorithm. In CachePath, a node does not need to record the path information of all passing-by data.

For example, when d_i flows from N_{11} to destination node N_1 along the path $N_5_N_4_N_3$, N_4 and N_5 need not cache the path information of d_i since N_4 and N_5 are closer to the data center than the caching node N_1 . Thus, a node only needs to record the data path when it is closer to the caching node than the data center. Due to mobility, the node which caches the data may move. The cached data may be replaced due to the cache size limitation. As a result, the node which modified the route should reroute the request to the original data center after it finds out the problem. Thus, the cached path may not be reliable and using it may adversely increase the overhead. To deal with this issue, a node N_i caches the data path

only when the caching node, say N_j , is very close. system tuning threshold, called TH, when CachePath is used.

5. Hybrid Caching Scheme

The performance analysis showed that Cache Path and Cache Data can significantly improve the system performance. We also found that Cache Path performs better in some situations such as small cache size or low data update rate, while CacheData performs better in other situations.

5.1 Hybrid Algorithm

To further improve the performance, we propose a hybrid scheme to take advantage of Cache Data and Cache Path while avoiding their weaknesses. Specifically, when a node forwards a data item, it caches the data or path based on some criteria. These criteria include the data item size s_i , the TTL time TTL_i , and the Hsave. For a data item d_i , the following heuristics are used to decide whether to cache data or path

- If s_i is small, CacheData should be adopted because the data item only needs a very small part of the cache; otherwise, CachePath should be adopted to save cache space. The threshold value for data size is denoted as T_s .
- If TTL_i is small, CachePath is not a good choice because the data item may be invalid soon. Using CachePath may result in chasing the wrong path and end up with resending the query to the data center. Thus, CacheData should be used in this situation. If TTL_i is large, CachePath should be adopted. The threshold value for TTL is a system tuning parameter and denoted as T_{ttl} .
- If Hsave is large, CachePath is a good choice because it can save a large number of hops; otherwise, CacheData should be adopted to improve the performance if there is enough empty space in the cache. We adopt the threshold value TH used in Cache Path as the threshold value.

These threshold values should be set carefully as they may affect the system performance. Their effects and how to set them are studied through simulations

ALGORITHM (Hybrid Cache)

When a data item d_i arrives:

```

if ( $d_i$  is the requested data by the current node) then
  cache data item  $d_i$ ;
  return;
If (an old version of  $d_i$  is in the cache)then
  Update the cached copy;
Else if ( $s_i < T_s$  or there is an invalid copy in the cache Or there is a cached path for  $d_i$ ) then
  Cache data item  $d_i$ ;
Else if ( $H_{save} > Th$  and  $TTL_i > T_{ttl}$ )then
  Cache the Path of  $d_i$ ;

```

S_i – size of data item $H \rightarrow$ Hop

T_s, Th – Thresholds levels of size of the data item

TTL – Max Time to Live T_{ttl} - Threshold level of ttl

When a request for data item d_i arrives:

```

If (there is a valid copy in cache) then send  $d_i$  to the requester ;

```

```

Else if (there is a valid path for  $d_i$  in the cache)
    then
Forward the request to the caching node;
    Else
Forward the request to the data center.

```

5.1 Hybrid Cache Algorithm

Section 5.1 shows the algorithm that applies these heuristics in Hybrid Cache. In our design, caching a data path only needs to save a node id in the cache. This overhead is very small. Therefore, in Hybrid Cache, when a data item d_i needs to be cached using Cache Data, the path for d_i is also cached. Later, if the cache replacement algorithm decides to remove d_i , it removes the cached data while keeping the path for d_i . From some point of view, Cache Data degrades to Cache-Path for d_i . Similarly, Cache Path can be upgraded to CacheData again when d_i passes by.

The parameter TTL _{i} and T_s play an Important role in the final performance ,because TTL _{i} (TTL _{i} –Time to Live for data item i) decides when to forward the request to the data center for requested data if it is not available in the nearest neighbor.

T_s is Threshold level for client cache ,if threshold level exceeds in client cache ,the client can forward its request to nearest neighbor and cache replacement policies also done.

6. Performance Evaluation

The performance evaluation includes three sections. The simulation model is given in Section 6.1. In Section 6.2, simulated environments and screenshots are shown. Section 6.3 shows the performance graph of cache data, cache path and hybrid cache from the analytical results.

6.1 The Simulation Model

The simulation is based on ns-2 with Cygwin Environment.. In our simulation, AODV (Ad Hoc On Demand Distance Vector) is used as an underlying routing algorithm. Since it is better than DSDV and I-DSDV(Improved Direct sequence Distance Vector). The aforesaid fact is justified by following results.

Packet Delivery Fraction of AODV

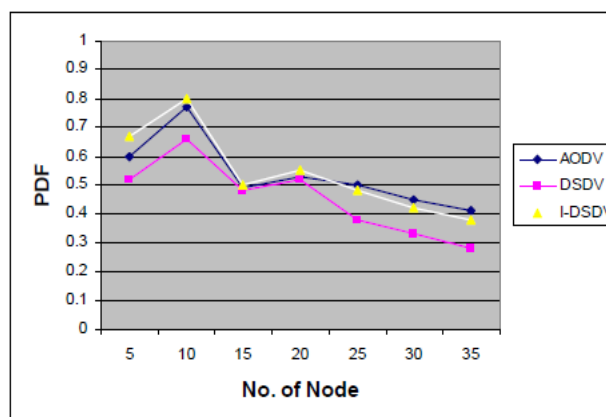


Fig. 6.1 a). Packet Delivery Fraction

From the Figure 6.1a) it is shown that AODV performs better when the number of nodes increases. This is because nodes that become more stationary will lead to more stable path from source to destination. DSDV performance drops as number of nodes increases, because more packets are dropped due to link breaks. I-DSDV is better than DSDV especially when the number of nodes is between 20 and 35. I-DSDV improves the PDF as it finds a new route to the destination when link breaks exist.

End to End Delay of AODV

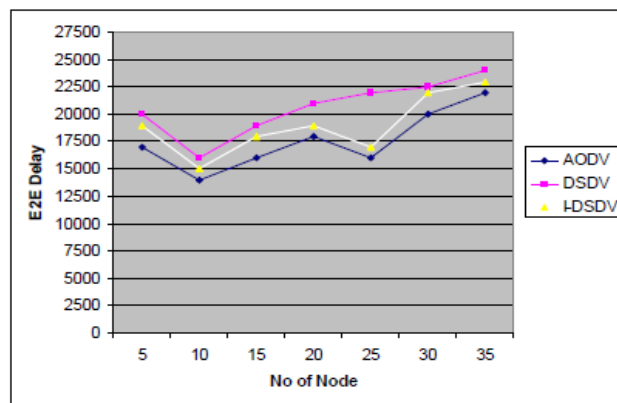


Fig. 6.1 b). End to End Delay

From the Fig 6.1 b) it is clear that AODV didn't produce so much delay even when the number of nodes increased. It is better than the other two protocols. The performance of I-DSDV is slightly better than DSDV especially when the number of nodes is between 15 and 30. It shows that, the I-DSDV protocol improved the DSDV but it was slightly lower than AODV when the nodes are higher. We assumed that the wireless bandwidth is 2 Mb/s, and the radio range is 200m to 250m.

In Fig 6.1b) when we increase the number of nodes initially, the delay gets increased for AODV, DSDV and I-DSDV. Later stage AODV and I-DSDV reach concave point (indicates delay reduction) because AODV supports many nodes .When we Compare DSDV with AODV(Fig 6.1b), AODV produces low delay even when nodes are higher in count.

Two concave spots are observed at 10 and 25 for blue and yellow line. And one concave spot is observed at 10 for purple line. Because at 10, AODV, DSDV and I-DSDV's delay are reduced at 25 when increasing the number of nodes (only AODV and IDSDV delay get reduced) .Finally the AODV performed better in delay reduction and Packet delivery (Fig 6.1 a) and was used as an underlying routing algorithm for our approach.

In paper [8] R tree based spatial data structure is used with k nearest neighbor algorithm for effectively finding nearest mobile node but not to deal with cache path and forward the requests (our proposed algorithm supports). In paper [15] Branch and Bound R -tree traversal algorithm is used to find the nearest neighbor object to a point and not supporting more number of pages accessed (our proposed algorithm supports). In paper [17] Knn is given as a input to SBNN and SBWQ (sharing based window query) which is effective only in caching but the data does not deal with cache path.(our proposed algorithm supports).

The experiment on both real and synthetic data sets are analyzed and corresponding results are plotted (Fig 6.1 c). From the results we can understand that the number of pages accessed is high in Hybrid cache algorithm even when increasing the number of neighbors (120).

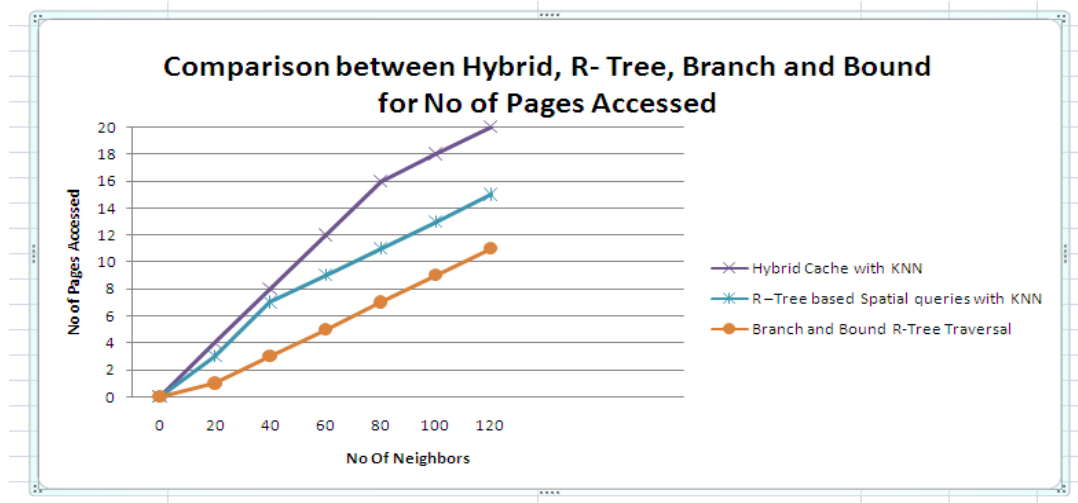


Fig. 6.1 c). Page Access Comparison between Hybrid Cache, R-Tree and Branch and Bound

In Paper [4] COCA (co-operative caching) leads to flood cache, however, as our method uses cache path, flood cache can be avoided.

In Paper [7] discussion is about COCA with Push based information .In this Server sends repeatedly data to client. This also leads to fill the cache of the client.

Experimental results (Fig 6.1 d) shows the comparison between hybrid cache, COCA and COCA with push based information ; we can see the delay percentage which is reduced for hybrid cache.

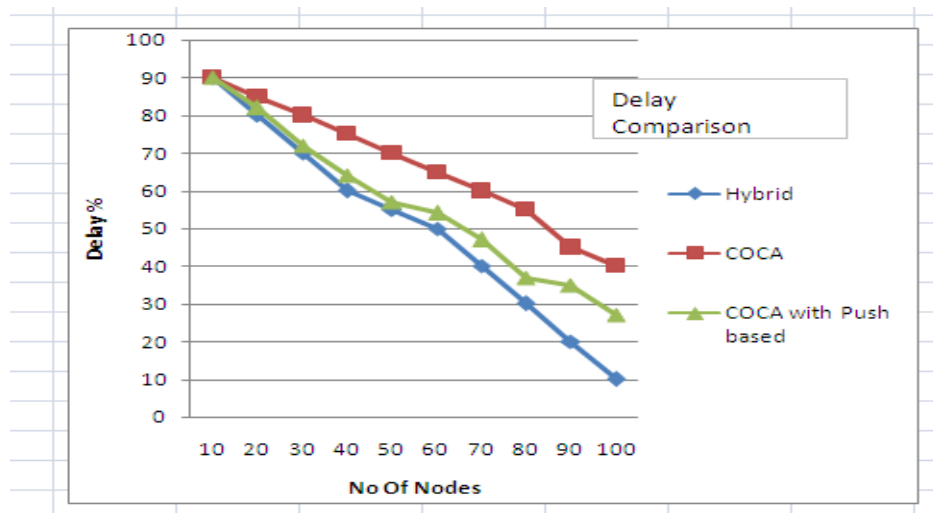


Fig. 6.1 d). Delay Comparison between Hybrid Cache ,COCA and COCA with Push based

6.2 Simulation

Experiments were run using different workloads and system settings. The design of wireless scenario is shown in the Fig 6.2 (a). In this caching of node and caching of path is shown. In Fig 6.2(b) and 6.2(c) the system model of the ad hoc network is shown. In this, seven nodes are shown. In that, one node acts as a Data Center and the remaining nodes have peer to peer connection.

As per the Hybrid cache algorithm Cache Path and Cache Data have been done with the help of ns2 simulator. Caching of data is done when the node 0 requests the data from the data center. If the request is again from the node 0, the node 0 itself analyzes for the data and gets the results .If some other node in this ad hoc network requests for the same data again, a node which has the path id of the node1 will respond to the request.

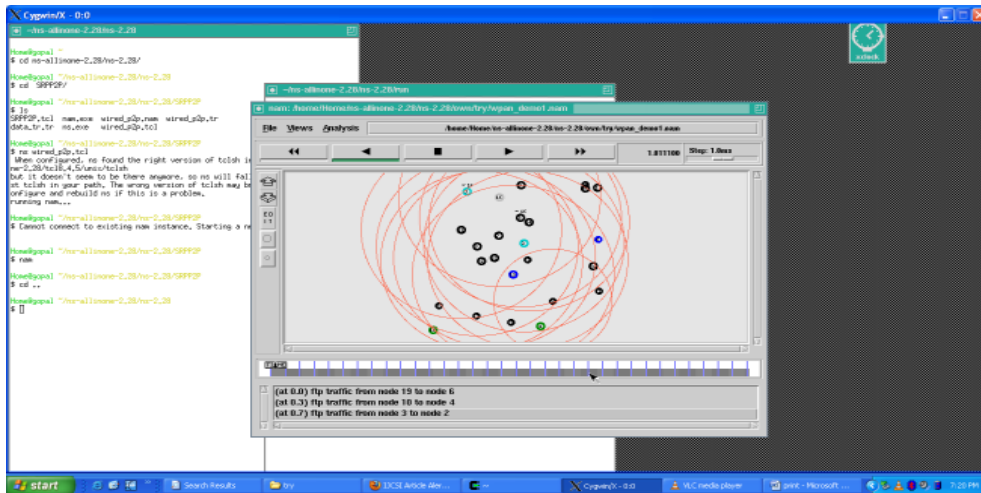


Fig. 6.2 a). Wireless scenario

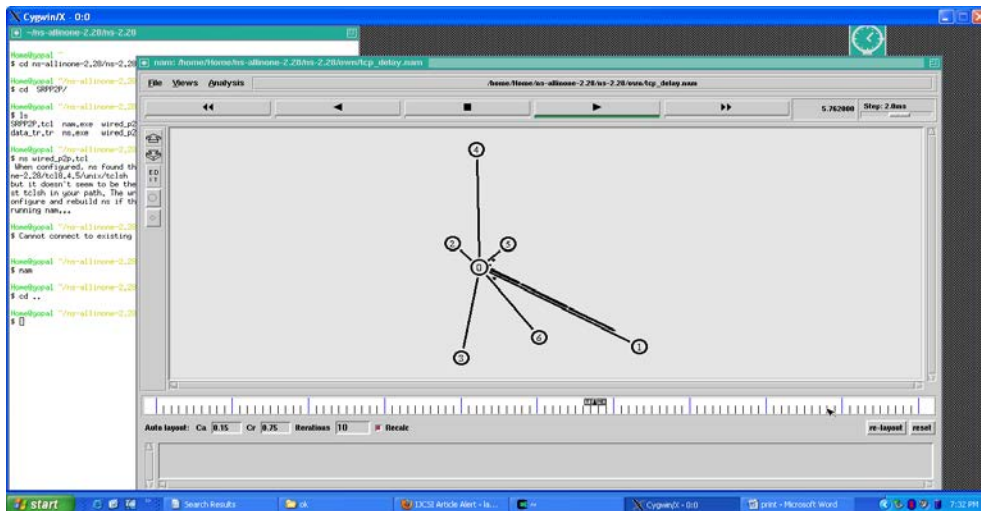


Fig. 6.2 b). Ad Hoc Model

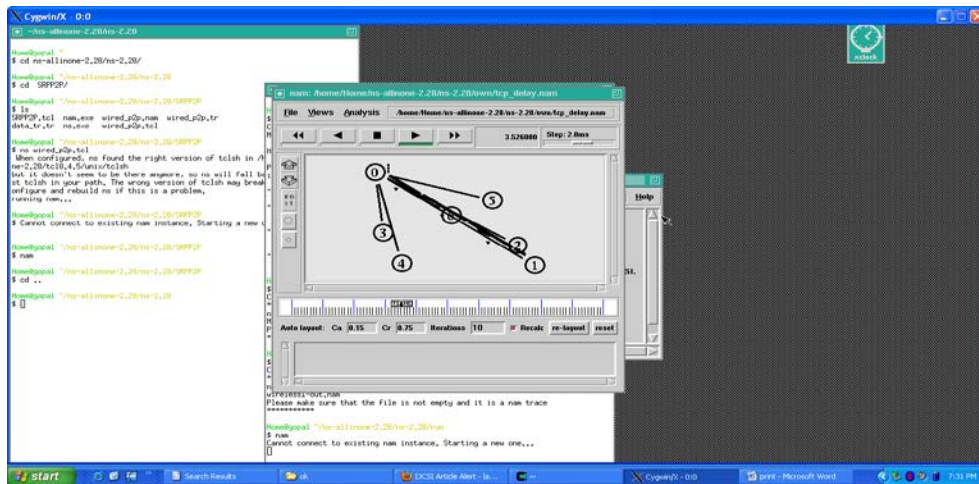


Fig. 6.2 c). Request of node 1 and Response from Data center (node 0)

6.3 Performance Graph

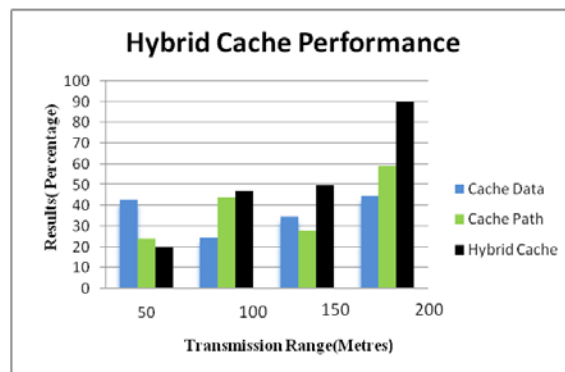


Fig. 6.3. Transmission Range Experiments

The Fig. 6.3 shows the comparison graph of cache data, cache path and hybrid cache. Comparison has been done by taking transmission Range as a parameter. From the Analytical results we can understand that hybrid cache methodology gives gradual growth while increasing transmission range. Hybrid cache is better than cache data and cache path. In this scenario we have considered cache data and cache path. We have considered cache data whenever it is essential (depending upon threshold level and size of a data) and also considered cache path (depending upon threshold of Time to live of a data item).

7. Conclusion

This paper presented a novel approach for reducing the spatial query access latency by leveraging results from nearby peers by applying Hybrid cache Mechanism. Specifically, we proposed three schemes: Cache Path, Cache Data, and Hybrid Cache. In Cache Data, intermediate nodes cache the data to serve future requests instead of fetching data from the data center. In CachePath, mobile nodes cache the data path and use it to redirect future requests to the nearby node which has the data instead of the faraway data center. Hybrid Cache takes advantage of CacheData and CachePath while avoiding their weaknesses. Cache Replacement policies are also studied to further improve the cache performance. Simulation

results showed that the proposed schemes can significantly reduce the query delay when compared to SimpleCache and significantly reduce the message complexity when compared to FloodCache. The experiment results indicate that our method can reduce the access to the wireless broadcast channel by a significant amount, for example, up to 90 percent.

References

- [1] Acharya, R. Alonso, M. J. Franklin, S. B. Zdonik, "BroadcastDisks: Data Management for Asymmetric Communications Environments," in *Proc. of ACM SIGMOD '95*, pp.199-210, 1995. [Article \(CrossRef Link\)](#)
- [2] Barbara, "Mobile Computing and Databases: A Survey, *IEEE Trans. Knowledge and Data Eng.*," vol. 11, no. 1, pp. 108-117, 1999. [Article \(CrossRef Link\)](#)
- [3] N. Beckmann, H.-P. Kriegel, R. Schneider, B. Seeger, "The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles," in *Proc. of ACM SIGMOD '90*, pp. 322-331, 1990. [Article \(CrossRef Link\)](#)
- [4] Chow, H. V. Leong, A. Chan, "Peer-to-Peer Cooperative Caching in Mobile Environment," in *Proc. of 24th IEEE Int'l Conf. Distributed Computing Systems Workshops (ICDCSW '04)*, pp. 528-533, 2004. www.computer.org/csdl/trans/tm/2009/03/ttm2009030384-abs.html
- [5] Chow, H. V. Leong, and A.T.S. Chan, "Distributed Group-Based Cooperative Caching in a Mobile Broadcast Environment, *MobileDataManagement*, pp.97-106, 2005. [Article \(CrossRef Link\)](#)
- [6] de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, "Computational Geometry Algorithms and Applications," second ed. Springer. 2000. [Article \(CrossRef Link\)](#)
- [7] Hara, "Cooperative Caching by Mobile Clients in Push-Based Information Systems," in *Proc. of 11th ACM Int'l Conf. Information and Knowledge Management*, pp. 186-193, 2002. [Article \(CrossRef Link\)](#)
- [8] Hjaltason, H. Samet, "Distance Browsing in Spatial Databases," *ACM Trans. Database Systems*, vol. 24, no. 2, pp. 265-318, 1999. [Article \(CrossRef Link\)](#)
- [9] Imielinski, S. Viswanathan, B.R. Badrinath, "Data on Air: Organization and Access," *IEEE Trans. Knowledge and Data Eng.*, vol. 9, no. 3, pp. 353-372, 1997. [Article \(CrossRef Link\)](#)
- [10] Ku, R. Zimmermann, "Location-Based Spatial Queries with Data Sharing in Mobile Environments," in *Proc. of 22nd IEEE Int'l Conf. Data Eng. (ICDE '06) Workshops*, p.14, 2006. [Article \(CrossRef Link\)](#)
- [11] Ku, R. Zimmermann, and H. Wang, "Location-Based Spatial Queries with Data Sharing in Wireless Broadcast Environments," in *Proc. of 23rd IEEE Int'l Conf. Data Eng. (ICDE)*, 2007. [Article \(CrossRef Link\)](#)
- [12] Mabel, C.-Y. Chow, and W.G. Aref, "The New Casper: Query Processing for Location Services without Compromising Privacy," in *Proc. of 32nd Int'l Conf. Very Large Data Bases (VLDB '06)*, pp. 763-774, 2006. [Article \(CrossRef Link\)](#)
- [13] Montgomery, G. C. Runger, "Applied Statistics and Probability for Engineers," second ed. John Wiley & Sons. 1998. [Article \(CrossRef Link\)](#)
- [14] Ringuest, "A Chi-Square Statistic for Validating Simulation-Generated Responses," *Computers and Operations Research*, vol. 13, no. 4, pp. 379-385. 1986. [Article \(CrossRef Link\)](#)
- [15] Roussopoulos, S. Kelley, F. Vincent, "Nearest Neighbor Queries," in *Proc. of ACM SIGMOD '95*, pp. 71-79. 1995. [Article \(CrossRef Link\)](#)
- [16] Sells, N. Roussopoulos, C. Faloutsos, "The R+-Tree: A Dynamic Index for Multi-Dimensional Objects," in *Proc. of 13th Int'l Conf. Very Large Data Bases (VLDB '87)*, pp.507-518, 1987. [Article \(CrossRef Link\)](#)
- [17] Zimmermann, W.-S. Ku, H. Wang, "Location-Based Spatial Query Processing in Wireless Broadcast Environments," *IEEE Trans. Mobile Computing*, vol. 7, no.6, pp. 778-791, 2008. [Article \(CrossRef Link\)](#)
- [18] Zheng, D. L. Lee, "Information Dissemination via Wireless Broadcast," *Comm. ACM*, vol. 48, no.

- 5, pp. 105-110. 2005. [Article \(CrossRef Link\)](#)
[19] Zheng, W.-C. Lee, and D. L. Lee, "Spatial Queries in Wireless Broadcast Systems," *Wireless Networks*, vol. 10, no. 6, pp. 723-736. 2004. [Article \(CrossRef Link\)](#)



Mr. C. Gopala Krishnan is Pursuing his PhD in Computer science and Engineering from St.Peter's University, Chennai, India. He received his M.E. degree in Computer science and Engg in 2010 from Anna University Tirunelveli, India and B.E. degree in Computer Science and Engineering in 2002 from Madurai KamaRaj University, India. His areas of interest are Mobile computing, Operating systems and Computer Networks and Multi Core Architecture. He has published many papers in International Conferences and journals in various fields. As part of this paper, he is working on developing communication protocols for wireless networks and protocols optimized for wireless and mobility that can support file and database access. He is also investigating operating systems support for mobile hosts He is a member of various Professional bodies.



Dr. A. Rengarajan earned his B.E Degree in Computer Science and Engineering from Madurai Kamaraj University, in 2000 and Master's degree in Computer Science and Engineering from Sathyabama University, Chennai, in 2005 and received his Ph.D. degree in Computer Science and Engineering from Bharath University, Chennai, India in 2011. He is now a Professor at Vel Tech Multi Tech Dr .RR Dr.SR Engineering College in Department of CSE. Chennai. He has more than 25 publications in National Conferences and International journals. He also received funded Projects from SERB. Currently, under his guidance many Research Scholars are pursuing PhD as full time and part time .His areas of interest are Network security, Mobile Computing, Wireless Networks , Data Structures, Distributed systems and Operating systems. He is a member of ISTE.



Mr. R. Manikandan received his B.E degree in Computer Science and Engineering from Anna University,Chennai,India and M.E Degree in Computer Science and Engineering from St.Peter's University, Avadi, Chennai. Currently, he is pursuing his Ph.D degree in Computer Science and Engineering from St.Peter'sUniversity,Chennai.He is now Working as a Teaching Fellow at University College of Engineering Tindivanam (A Constituent College of Anna University Chennai), he presented many papers in National and International conferences in various fields,His research interests include Cloud Security, Mobile Computing,Wirless Networks and Cryptography and Network Security He is a member of various professional organizaion for engineers.