A Virtual Backbone Based Approach for Cooperative Caching in Mobile Ad hoc Networks

Preetha Theresa Joy*, K. Poulose Jacob*

*Department of Computer Science, Cochin University of Science and Technology, Kochi, Kerala, India

preetha@mec.ac.in, kpj@cusat.ac.in

Abstract— **Cache look up is an integral part of cooperative caching in ad hoc networks. In this paper, we discuss a cooperative caching architecture with a distributed cache look up protocol which relies on a virtual backbone for locating and accessing data within a cooperate cache. Our proposal consists of two phases: (i) formation of a virtual backbone and (ii) the cache look up phase. The nodes in a Connected Dominating Set (CDS) form the virtual backbone. The cache look up protocol makes use of the nodes in the virtual backbone for effective data dissemination and discovery. The idea in this scheme is to reduce the number of nodes involved in cache look up process, by constructing a CDS that contains a small number of nodes, still having full coverage of the network. We evaluated the effect of various parameter settings on the performance metrics such as message overhead, cache hit ratio and average query delay. Compared to the previous schemes the proposed scheme not only reduces message overhead, but also improves the cache hit ratio and reduces the average delay.**

Keywords— **Cooperative Caching, Cache Look up, Virtual Backbone, Connective Dominating Set, Cache Replacement**

I. INTRODUCTION

Mobile Ad hoc Networks (MANET) are rapidly deployable, multi-hop, infrastructure less, wireless networks, in which the mobile nodes does not have a prior knowledge of other nodes in the network. They allow the applications running on different wireless devices to share data of different type. This makes it ideal for applications where initial connection setup is not possible or unreliable, like military and disastrous areas. Ad hoc networks have a wide range of commercial applications like personal area networks, sensor networks, emergency networks and vehicular networks. However, due to the highly dynamic nature of ad hoc networks data accessibility in MANET is a challenging problem. One possible solution to this problem is data caching. Caching reduces network traffic and server load by migrating data closer to mobile clients that use the data. However, the cache needs to be large enough to serve the client requests. This can be achieved by sharing the contents of local cache among the clients. Cooperative caching is the sharing and coordination of cache state among multiple clients. It has been recognized as an important technique to reduce data traffic and alleviate network bottlenecks [1]. In MANET, it is likely that multiple clients in the same region will try to access the same service concurrently. So caching such services would be beneficial. As a result, cooperative caching has been used to improve system performance in ad hoc networks.

The two main issues in any caching system are cache placement and cache look up. Cache placement policies decide which nodes should act as cache nodes and cache look up solves the issue of how to find the node that caches data. The effectiveness and efficiency of cooperative caching depends greatly on the cache look up scheme. This paper deals with a decentralized cache look up protocol which reduces the cache overhead by reducing the search space to the nodes in the virtual backbone.

Generally, in cooperative caching there are two main approaches for cache lookup: Search based and directory based. In search based lookup schemes, flooding is the technique used to disseminate data request among the neighbouring nodes. Whenever there is local miss, the search message is propagated to all the neighbouring nodes in the network. The cooperative caching schemes proposed in [2,3] uses this approach for cache discovery. Although time latency is minimal in flooding, it generates a large number of messages which will cause excessive control message overhead as unintended nodes have to receive and process these packets. This will increase the energy consumption and bandwidth utilization of the network. This is an important concern for MANETS as the nodes are energy constrained and the bandwidth available is limited. Moreover, there is a chance of increased collisions, which could degrade the overall performance of the system. On the other hand, directory based technique [5] uses a centralized approach in which a coordinator node will maintain the status of the data present in the neighbouring nodes and the data requester can get the data directly from the coordinator node. Directory based approach greatly reduces the number of messages and latency. However, this approach also imposes several challenges. First of all, maintaining group information is difficult due to the mobility of nodes. The control node may get disconnected which causes excessive overhead [10].The number of entries in the directory increases for higher network density.

This paper proposes a distributed cooperative caching architecture which relies on virtual back bone for locating and accessing data within a MANET. Since MANETs are infrastructure less networks, they do not have a physical backbone infrastructure. A dynamic virtual backbone structure can be formed among the mobile nodes by connected

dominated set (CDS). Our aim is to develop a cache look up protocol that takes the advantage of CDS. The idea in this scheme is to reduce the number of nodes involved in cache look up process, by constructing a CDS that contains a small number of nodes, still having full coverage of the network, while only relaying on local exchange of information and knowledge. Cooperative caching based on virtual backbone reduces the search space for cached data to the hosts in a smaller sub network generated from the CDS.

The proposed cooperative caching architecture consists of two phases: Backbone formation phase and a distributed cache lookup phase. In phase 1, a dynamic virtual backbone is formed such that each node in the network is either a part of the backbone or one hop away from at least one of the backbone nodes. The second phase is the distributed cache look up phase, in which data request message is forwarded to nodes inside the dominating set for data discovery. The cache lookup protocol based on virtual backbone attempts to achieve complete network coverage by ensuring that every node receives the request.

The rest of this paper is organized as follows. Section II surveys previous works related to caching data in mobile ad hoc networks. Section III gives the system outline and assumptions made. Section IV introduces the proposed framework and discusses its design. Section V discusses simulations results and their implications. Finally, the conclusion is given in Section VI.

II. PREVIOUS WORK

Ensuring availability of data is a challenging problem in MANET, because mobile clients need to access the data in an ad hoc manner without the help of any central server. Moreover, the mobility of nodes can result in network partitions and node disconnections. Cooperative caching is an important technique in ad hoc networks to enhance data availability. When considering mobile ad hoc networks, users have a common interest and deals with applications and data that can be shared by all the nodes. Cooperative caching can ensure that most of the data requests can be satisfied from the neighbouring nodes instead of contacting a remote data source. A number of cooperative caching protocols are proposed in the literature in order to improve data accessibility in mobile ad hoc networks. The following section presents some of the cooperative caching techniques related to our work that has been proposed for mobile ad hoc networks.

Yin & Cao [1] proposed a cache frame work where three different caching techniques were used: Cache Data, Cache Path, and Hybrid Cache. In Cache Data, the intermediate nodes between the source and the destination caches the data when it finds that the data item is popular. Cache Path works in a similar manner to Cache Data, but the forwarding nodes cache the data path information for future use, instead of caching the data. Hybrid Cache takes advantage of the above two schemes for further improving the caching performance. Here a mobile node caches the data or path based on data item size and the Time to Live (TTL) parameter. If data item size is small, Cache Data scheme is used, since the data item needs only a small portion of the available cache; otherwise, Cache Path is used. If TTL is small, Cache Data is used because the data item might soon get invalid, using Cache Path can result in changing the path frequently. Here the cache consistency is ensured by TTL mechanism. A routing node considers a cached copy as valid if its TTL hasn't expired. If the TTL has expired, the node removes the cached data item. The drawback of Cache Data scheme is that the forwarding nodes consume lot of cache space. Path becomes obsolete in case of Cache Path algorithm
which causes extra processing overhead.

An aggregate caching scheme to increase the data accessibility in internet based mobile network is presented in [19]. They used a broadcast based information search algorithm called simple search to locate the required data item. Whenever a mobile node needs some data, a request message is broadcast to its adjacent nodes. Upon receiving the broadcast request, the adjacent nodes reply to the request if it has already cached the data, otherwise the request is forwarded to its neighbors until it is acknowledged by an access point or some other nodes which have the requested data. Flooding is the technique used for broadcasting. This algorithm sets a hop limit for the request packet to reduce the traffic in the network. Neighbor caching [20] make use of the neighboring nodes to store data. In this scheme, the data evicted from one node is stored in an idle neighbor node's storage. A ranking based prediction selects the appropriate neighbor to store data. Broad cast based approach is used to find cached data in neighboring nodes. Least Recently Used (LRU) cache replacement policy is used to evict data from the cache when it is full.

COCAS [15] is a distributed caching scheme designed for MANETs to find the requested data from cached nodes. The submitted queries are cached in some special nodes called query directories (QD) and these queries are used as an index to find the previously cached data. Whenever a data item is retrieved, cache nodes (CN) will cache the data and the nearest QD to the cache node will cache the query along with the address of the CNs containing the corresponding data. The assignment of QDs and CNs are done by a service manager. The limitations of this scheme include, broadcasting of requests for searching QDs, and the single point of failure of the service manager. Group caching [4] uses a table based approach for cache discovery and data dissemination. Each node maintains two tables, a group table and self table to maintain the status of neighboring caching nodes. For cache discovery, the local cache table is searched first and if data is not found, the group table is searched to find the location of the cached data. The group table contains cached data *id* and the node *id* which contains the data. LRU is the replacement policy used. The drawback of this approach is the number of control messages exchanged when the node density is high. Also individual nodes have to process these messages which increase the computational overhead. The difference between all other schemes reviewed and the work in this paper is the fact that we use decentralized hierarchical approach for

cooperative caching that overcomes the limitations of previous works.

III. System Outline and Assumptions

We assume that all the nodes in the network have the same transmission range, *R*. Two nodes in the network are neighbors if the Euclidean distance between their coordinates in the network is at most *R*. The Euclidean distance between the nodes are estimated based on the relative position of nodes. The communication channel is shared between the nodes in the network, hence transmitting and receiving messages is not allowed at the same time. The above mention network can be modeled as a unit disk graph G (*V, E*), where *V* is the set of vertices that represents the number of hosts in the network, *E* is the set of edges and unit distance corresponds to the transmission range of the mobile host. We assume an abstract entity called virtual backbone which is a collection of nodes responsible for cache look up. The connected dominating set of a unit-disk graph is used to form the virtual backbone. After the formation of a virtual backbone, the nodes in the network will be either a backbone node or an end node. The nodes in the dominating set are referred as Dominant Nodes and the nodes which are not in the set are referred as End Nodes. The rest of the notations and assumptions are as follows.

The mobile computing environment considered in this paper consists of mobile clients and a data base server connected to a static network. In order to minimize the number of data server request, the client node caches a portion of database in its local memory. The contents of the local cache are shared by its neighboring nodes. When a node fails to find data in neighboring nodes, data is retrieved from the data server. Each node has a unique identifier and location, which it wishes to communicate to all its neighbors. A node *x* is a neighbor of another node *y* if *x* is located within the transmission range of *y*. N(*x*) refers to the set of neighbors of node *x*. Each node maintains two lists, one to store the ID of the neighboring nodes and the second list is to store the neighbor list of each neighboring node. Initially, to find the neighbor node set of node *x* in the transmission range *R*, a short neighbor request control message is disseminated in to the network. The request control message contains the following fields: *the source id, current location and request id*. The *request id* is used to identify the neighbor request control message. When a node receives the request control message, it sends back a reply control message which includes the *node id* and the neighbor list of its neighbors. This is to find the nodes that are two hops away from each node. As the nodes are mobile, the location coordinates of a node may changes in time, as do the connectivity between them. Thus each node's set of neighboring nodes defined within the transmission range will continuously change. In order to maintain the neighbor node set accurately, each node periodically sends a neighbor request control message to its neighbors. To reduce further message overhead the frequency at which update messages are disseminated is determined by the mobility rate of the nodes.

IV. Proposed Cooperative Caching Architecture

In this section, we describe the architecture of our proposed cooperative caching scheme. Our aim is to develop a cache look up protocol which is scalable in terms of network size, as well as efficient in data access. With this design goal, we propose a cooperative caching protocol that uses a virtual backbone based design in which data items are easily located through the indexing performed by the backbone nodes constructed from the connective dominating set.

The proposed solution for cooperative caching works in two steps: In the first step, virtual backbone setup is performed with a subset of nodes from the connective dominating set (CDS). The second step is the cache lookup phase in which the desired data item is searched by sending the request to the nodes in CDS. The detailed description of both the phases is given in the subsequent sections.

A. Virtual Backbone Construction

An ad hoc network topology can be modelled as a unit disk graph G = (*V, E*) in which there is an edge between two nodes if and only if their distance is at most one [12].Virtual backbone in an ad hoc network can be formed by Connected Dominating Set (CDS) of the corresponding unit disk graph [13].

Definition 1: For a graph G (V, E), a dominating Set S of G is defined as a subset of V such that each node in V\S is adjacent to at least one node in S.

Definition 2: A connected dominating set (CDS) for a graph G (V, E) is a subset V' of V, such that each node in V -V' is adjacent to some node in V', and V' induces a connected sub graph of G. The nodes in CDS are called dominators, the others are called dominatees.

The virtual backbone has a an important role in ad hoc networking for routing, broadcasting and mobility [13].The problem of finding minimum dominating set has been proven to be NP–Hard [12]. The heuristics to find minimum CDS falls into two categories: Prune based and maximal independent set (MIS) based. In prune based approach [15], [16], the redundant nodes are pruned from the CDS set according to certain rules. In maximal independent based algorithms [17], [18], first MIS is formed and then connectors are found to form a CDS. In this paper, we implement a prune based algorithm proposed by Wu and Li [15] since it is simple and easily maintainable CDS construction algorithm.

***1) CDS Construction*:** Wu and Li [15] proposed a localized algorithm based on marking process to find CDS. In this process, every node $v \in V$, in the given graph G = (V,E) will be in either marked (T) or in unmarked state(F).The node is in marked state if it has two unconnected neighbours, otherwise it is unmarked. The set of marked nodes form the CDS. The size of the CDS is further reduced by applying two rules for pruning. The open neighbor set of each node v is represented as $N(v) = \{u \mid \{u, v\} \in E\}$.

2) ***Marking Process:*** Initially, all nodes are unmarked, in F state. Every node v exchanges its N (V) with all its neighbours. Mark v to T if it has two unconnected neighbours.

Figure 1, where a connective dominating set is computed illustrates this algorithm. Black nodes represent the dominant nodes while white ones are the end nodes. In this example, node 0 is not dominant because all of its neighbours form only one connected component, while node 2 is dominant because there exists two components in its neighbourhood {3,5} that are not connected.

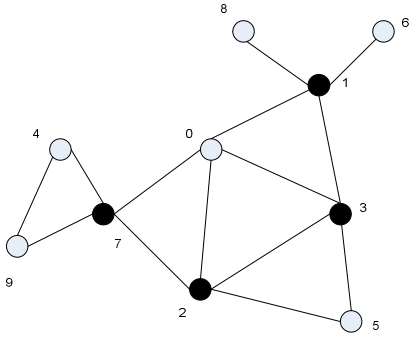

Figure1 A connective dominating set

After the marking process assume that V' is the set of vertices that are marked T in V i.e., V' = {v| v ∈ V, m (v)=T}.The induced graph G' is the sub graph of G induced by V, i.e., G'= G [V'], forms the CDS. To remove redundant nodes from the obtained CDS set, two pruning rules based on node ID are applied. A distinct ID, id (v) is assigned to each vertex in G. N [V] =N(v) ∪ {v} is the closed neighbor set of v.

Rule 1: Consider two vertices v in G' and u in G. If N[v] ⊆ N[u] in G and if id (v) < id (u), the marker of v is changed to F if node v is marked, i.e., G' is changed to G - {v}.

According to Rule 1, a marked node can unmark itself if its neighbor set is covered by another marked node with higher node ID.

Rule 2: Assume that u and w are two neighbors of marked vertex v in G'. If N(v) ⊆ N(u) ∪ N(w) in G and id(v) = min {id(v),id(u),id(w) }, then the marker of v is changed to F.

According to rule 2, a marked node can unmark itself if its neighborhood is covered by two other directly connected marked nodes with higher node IDs. The combination of the two rules reduces the size of a CDS. The message complexity of this algorithm is O (n), where 'n' is the number of nodes in the network.

B. Cache Look up Phase

Now, we have the virtual backbone constructed using the Dominating Set algorithm. We still need mechanisms to locate data present in other mobile clients. Our aim is to develop a cache lookup protocol that minimizes delay and maximizes the probability of finding data without inducing much message overhead with the increase in the number of nodes. When we run the CDS construction algorithm, every node in the network will be either a dominator node (DN) or a dominatee or End Node (EN) node. The DN is the central component of the system and plays a major role in finding the location of the cache data. Each DN has knowledge about its 2 hop neighbors, and maintains an index table, to store the data ID and the corresponding node ID. All the entries in the table are sorted in the increasing order of the data ID.

C. Locating Cached Data

In our proposed system the data search process can be initiated either by the DN or by the EN. To retrieve data, each node initially searches in its local cache. If there is a cache miss, it tries to locate the requested data in other cooperate caches. In order to minimize redundant messages, the nodes present in the virtual backbone perform the data lookup process in the case of primary cache miss. Every DN keeps a list of data items present in their neighboring nodes. When there is a primary cache miss, the query node sends Request Packet (RP) to the nearest DN. Upon receiving the RP packet, the DN node extracts the data ID from the RP packet; performs a binary search using the data ID as the key value. If there is match in data ID, the corresponding node ID is taken from the index table. If the DN does not have a matching data, it adds its address to the RP and sends the modified RP to the nearest DN that has not been checked yet. This continues until a hit occurs or until all the DNs are checked, in which an attempt is made to contact the data source. If a hit occurs, the DN nearest to the query node stores the data id and query node id in the index table.

D. Index Table Management

The DN acts as an index node storing the address of the neighboring nodes which stores the data. Each DN maintains index information of the data item present in its neighboring nodes. An index table is maintained at DN. Each entry in the table consists of a pair of index information $<D_i, n_i>$.The parameter D_i is the data ID and the second parameter is the associated node ID that contains the data. The size of the index table is equal to the number of data items present in the neighboring caches and is sorted based on the key value. Upon receiving a data request from the EN, if the data item is present in any one of the neighboring nodes the table is not updated. Otherwise, the corresponding EN ID and data ID is stored in the index table. This is because after some time the node will cache the requested data. When an item is removed from an EN cache it has to be notified to the corresponding DN. To enable the notification of items being removed from the cache, a Cache Item Remove (CIR) packet is send to the DN. The CIR packet contains the tuple $<n_i, D_i>$, signifying the ID of the node that is going to remove the data item D_i. Upon receiving the CIR packet, the DN will remove the selected entry from the index table to prevent misdirected requests. Whenever an EN leaves the network, the corresponding entry related to that node is removed from the DN index table.

E. Cache Management

Cache management involves cache replacement and cache placement. As in any caching scheme, the capacity of cache in each node is limited and appropriate cache replacement mechanism is needed to evict unwanted data from the cache. The proposed cooperative caching make use of the replacement policy proposed in [20], called Extended-Least

Recently Used (E-LRU) replacement. In E-LRU, the time interval between last two references of each cached data item is stored. Using this information the inter reference time of data items are estimated. Whenever the cache space of the mobile host becomes full and a new data is to be added, the data item with maximum inter reference time is evicted from the cache. If there are data items that are referenced only once, Least Recently Used (LRU) policy is used to select the replacement victim. In LRU policy, the data item that has not been accessed for the longest time is dropped first. Since cache placement is performed through DN duplicate data placement is reduced.

F. Effect of Node Mobility

Due to mobility of nodes there may be several new link connections and disconnections to the nodes within the transmission range. When the topology changes the role of nodes may change. For maintaining the CDS in the case of node mobility the following steps are taken.

Each mobile node updates its neighbor list every *t* seconds. A mobile node *x* compares its new neighbor set N'(*x*) with its original neighbor set N(*x*). If they are same, it simply remains in its state. Otherwise, it performs the following steps.

The new open neighbor set of node *x* is exchanged with its neighbors. Each node in the set will change its state to DN, if it has two unconnected neighbors. If a new DN is formed, the newly formed DN and its DN neighbors apply Rule 1 and Rule 2 to reduce the number of nodes in the CDS. Whenever DN node recognizes a broken link to its neighboring nodes, DN maintains its status as dominating node if it has two unconnected neighbors; otherwise it changes its state to EN.

V. SIMULATIONS

To evaluate and compare our scheme with other cooperative caching schemes, we developed a simulation model. The simulated model consists of mobile nodes which can move freely in an area of 1000X1000m^2. Each mobile node is identified by a node ID. The position of each node is given by the x and y coordinates. The transmission radius is assumed to be the same for all nodes. The nodes move in the given area according to the random waypoint mobility model [22]. Initially, the mobile nodes are randomly distributed in the simulation area. After that each node randomly chooses its destination with a speed *s* which is uniformly distributed U (V_{min},V_{max}) and travels with that constant speed *s*. When the node reaches destination, it pauses for 200 seconds. After that it moves to the new destination with speed *s'*.

Each node periodically generates read only queries. The querying nodes are selected randomly from those nodes that move locally in the defined simulation area. The time interval between two consecutive queries generated from each node follows an exponential distribution with mean of 10sec. The queries generated follow a Zipf distribution [2]. An infinite queue is used to buffer the request when the data center is busy. There is a single data server available in the system. It is implemented in a fixed position in the simulation area. The data server contains all the data items requested by the mobile nodes. The size of each data item is uniformly distributed between s_{min} and s_{max}. The database in the data server contains 1000 data items, with each item identified using a data id. The data request is processed in First Come First Served (FCFS) manner at the server. An infinite queue is used to buffer the request when the data server is busy.

B. Performance Metrics

The performance metrics evaluated include cache hit ratio, message overhead and average query delay. Cache hit ratio defines the number of references made from the cache over the total number of references. If R_i the total number of references made to data item *i*, H_i the number of hit references made to data item *i*, the Cache Hit Ratio (CHR) is defined as, CHR = $\sum H_i / \sum R_i$. For our performance evaluation, H_i includes local hit and neighbor hit. Query Delay is defined as the interval between the query sent and the client node receives a reply for the request. Average query delay is the query delay averaged over all the queries. The number of message processed at each node is taken as the message overhead.

C. Simulation Results

In this section, we present the results obtained by comparing our new cooperative caching protocol called **V**irtual backbone based **C**ooperative Caching (VC) with other cooperative caching schemes, Neighbour Caching (NC) which uses broadcasting for data discovery and Group Caching (GC) that uses a table based approach for cache discovery. We measured the cache hit ratio and average delay for different cache sizes and message overhead for different node densities. Cache size is taken as different percentage of total database. Fig. 2 shows the result when cache size is varied from 10-70% of total database size. The performance of all the cooperative caching schemes improves with increase in cache size. This is because as the cache size increases, it can hold more number of data items. From the figure we can see that the difference between cache hit ratios is higher when the cache size is small. Fig. 3 shows the performance comparison for average delay for different cache sizes. The delay is affected by the cache hit ratio and the data discovery process. As shown in the figure, the VC approach is quite close to the performance of the GC approach when the cache size is high. From Fig. 4, we can see that most of the performance improvement for VC comes in message overhead. This is because in VC, the number of nodes involved in cache discovery process is limited by CDS.

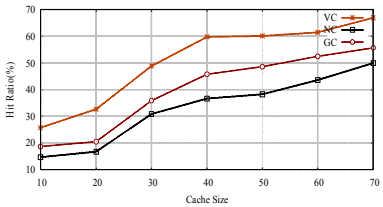

Figure 2 Cache Hit Ratio for different cache sizes

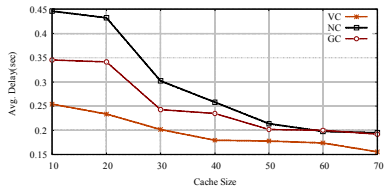

Figure 3 Query delay for different cache sizes

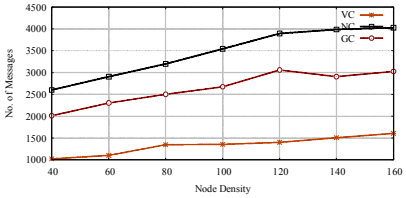

Figure 4 Message overhead for different node densities

VI. CONCLUSIONS

In this paper, we propose a new scheme for cooperative caching which employs a virtual backbone for cache look up. In this scheme the cache lookup is reduced to nodes inside the virtual backbone. Simulation results shows that this approach can greatly reduce the number of messages and the cache overhead while maintaining high cache hit ratio and reduced delay. At the same time, since our scheme uses CDS it can provide good scalability with respect to the number of nodes.

REFERENCES

[1] L. Yin, G. Cao, Supporting cooperative caching in ad hoc networks, *IEEE Transactions on MobileComputing* (1)(2006)77–89.

[2] B. Tang, H. Gupta, S. Das, Benefit-based data caching in ad hoc networks, in *Proc. of ICNP,SantaBarbara,California,2006*,November.

[3] M. Fiore, F. Mininni, C. Casetti, C. Chiasserini, To Cache or not to Cache, in*: Proceeding of IEEE International Conference on Computer Communications*,2009.

[4] Y. Du, S. Gupta, G. Varsamopoulos, Improving on-demand data access efficiency in MANETs with cooperative caching, *Elsevier, Ad Hoc Networks* (2009)579–598.

[5] C. Chow, H. Leong, A. Chan, GroCoca: Group-based peer-to-peer cooperative caching in mobile environments, *IEEE Journal on Selected Areas in Communications*25(1)(2007).

[6] M. Taghizadeh, A. Plummer, S. Bisws, Cooperative caching for improving availability in social wireless networks, in *Proc. of MASS*,2010.

[7] T. Hara, Effective replica allocation in ad hoc networks for improving data accessibility, in *Proc. of INFOCOM*, 2001.

[8] Y. Ma, A. Jamalipour, A cooperative cache-based content delivery framework for intermittently connected mobile ad hoc networks, *IEEE Transactions on Wireless Communications* (2010).

[9] Y.Lin, W.Lai, J. Chen, Effects of cache mechanism on wireless data access, *IEEE Transactions on Wireless Communications* (6)(2003).

[10] Yi-Wei Ting , Yeim-Kuan Chang. A novel cooperative caching scheme for wireless ad hoc networks: Group caching, In NAS '07: *Proceedings of the International Conference on Networking, Architecture, and Storage*, (2007).

[11] A. Valera, W. Seah, and S. Rao, "Improving protocol robustness in ad hoc networks through cooperative packet caching and shortest multipath routing," *IEEE Transactions on Mobile Computing* , v 4, n 5, 2005, pp. 443-457.

[12] Clark,C.Colbourn,andD.Johnson,\Unitdiskgraphs,"*DiscreteMathematics"*, vol.86, no.1-3,pp.165{177,1991.

[13] V. Bharghavan and B. Das, "Routing in Ad Hoc Networks Using Minimum Connected Dominating Sets", *International Conference on Communications'97* ,Montreal, Canada. June 1997.

[14] M.R. Garey and D.S. Johnson, *Computers and Intractability: A guide to the theory of NP-completeness*, W. H. Freeman and Company, San Francisco, 1979.

[15] J. Wu and H. Li, On calculating connected dominating set for efficient routing in ad hoc wireless net-works, *Proc. The 3rd International workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, 1999, pp.7-14.

[16] F. Dai and J. Wu, Distributed dominate pruning in ad hoc wireless networks, *Proc. ICC*, p. 353, 2003.

[17] M. Ding, X. Cheng, and G. Xue, Aggregation tree construction in sensor networks, *Proc. of IEEE VTC*, 2003.

[18] P.-J. Wan, K.M. Alzoubi, and O. Frieder, Distributed Construction of Connected Dominating Set in Wireless Ad Hoc Networks, *IEEE INFOCOM*, pp. 1597-1604, 2002.

[19] S. Lim, W. C. Lee, G. Cao and C. R. Das, "A novel caching scheme for

[20] internet based mobile ad hoc networks," in *Proc. 12th Int. Conf. Comput.Comm. Networks (ICCCN 2003*), Oct. 2003, pp. 38-43.

[21] Joonho Cho, Seungtaek Oh, Jaemyoung Kim, Hyeong Ho Lee, Joonwon Lee, Neighbor caching in multi-hop wireless ad hoc networks, *IEEE Communications Letters*, Volume 7, Issue 11, 525 – 527(2003).

[22] Preetha Theresa Joy and K. Poulose Jacob, A Key Based Cache Replacement Policy for Cooperative Caching in Mobile Ad hoc Networks, 2013 *3rd IEEE International Advance Computing Conference (IACC*), 2013, pp 383-387.