A Review of Cooperative Caching Strategies in Mobile Ad Hoc Networks

P.Kuppusamy
Assistant Professor / CSE
Vivekanandha College of Engineering for Women,
Namakkal, India.

Dr.K.Thirunavukkarasu
Principal
T.J.S Engineering College
Chennai, India.

Dr.B.Kalaavathi
Professor / IT
K.S.R College of Technology
Namakkal, India.

ABSTRACT
The Mobile nodes are communicating with each other without centralized administration and data is accessed from the data source through multi-hop environment. The accessed data is stored in the mobile node's cache for its own and neighbor's future use. Caching is a significant process to store the frequently accessed data item in the MANET. Data availability and accessibility is a challenging task due to mobility of nodes, limited battery power and insufficient bandwidth. Cooperative caching addresses these challenges to improve the data availability and efficiency of data access by sharing and coordination among the mobile nodes. These challenges have received a tremendous amount of concentration from researchers and led to development of many different cooperative caching strategies. This paper attempts to provide the review and hypothetical analysis of various cooperative caching strategies in the mobile ad hoc networks based on their performance metrics such as cache hit ratio and average query delay with respect to cache size and number of mobile nodes. The Global Cluster Cooperative caching provides better performance than others in terms of cache hit and average query delay.

Keywords
Mobile ad hoc network, Cooperative Caching, cache hit, query delay.

1. INTRODUCTION
Mobile Ad Hoc Network (MANET) [3], [9] is a self configured temporary network with a set of mobile nodes (MNs). The MN's act as routers, keeping route information to reach other MNs and forward data packets from one MN to another due to lack of infrastructure. The network topology is changed continuously due to the frequent mobility of nodes which reduces the data availability. Each node communicates with another through unreliable multi-hop wireless links which may cause long query delay. Each MN acts as a client or server with local database based on the applications. Many nodes frequently access the data item from a server that results in overload and high response time of a server. Multi-hop communication degrades the network capacity when network partition occurs. These issues are overcome by improving the data accessibility [10] using caching strategies. The previous researches in MANET are mainly focused on the multi-hop routing for efficient data communication, but not on data access and availability in the MNs. Therefore improving the data accessibility [5] by using caching technique is another most important issue. The cache management is also a challenging task in MANET due to nodes mobility, variable data size, limited nodes resources and insufficient bandwidth. Co-operative caching [9] which allows the sharing and coordination of cached data among multiple MNs, can be used to reduce the bandwidth and power consumption. Moreover this scheme reduces the query delay by providing the requested data either from the local cache or forwarding requests to its nearby MNs.

1.1 Caching
When a MN requires a data item, it sends the data request to the server which is in a long distance multi-hop environment. All the MNs along the path forward the request to the server. The server could not tolerate the load when more nodes accessing the data at the same time. It generates the bottleneck with large amount of traffic at the server; as well the data request and reply need to pass through multiple nodes to reach the server. Hence, the communication may consume large bandwidth, power and take long query delay. The above issues motivate the researcher to make investigation on the caching strategies.

The cooperative caching is helpful to solve the use of network bandwidth and query delay to retrieve the data from the server. The MNs are equipped with intel/AMD processor and cache memory. MNs store the requested data items in the cache memory for its future needs [12]. This cached information is shared with the nearby MNs. Hence, every node in the network is aware of the contents of its neighbor's cache. When the MN sends data request to the nearby node which is in the path that request may be fulfilled by its cached content instead of the server. First, it reduces the long query delay and bandwidth by serving nearby nodes cached content. Second, power consumption, traffic at the server is reduced by maintaining the data in the intermediate node's cache.

The cooperative caching addresses two basic caching issues in MANETs:

1.2 Cache resolution
A MN decides where to fetch a data item requested by the user based on cache resolution. Cooperative caching tries to discover a data source which induces less communication cost by utilizing historical profiles and forwarding nodes.

1.3 Cache management
A MN decides which data item to be placed or deleted in its local cache based on cache replacement and cache admission control. Cooperative Caching minimizes caching duplications between nearby nodes and allows the nodes to store more distinct data items to improve the overall performance.

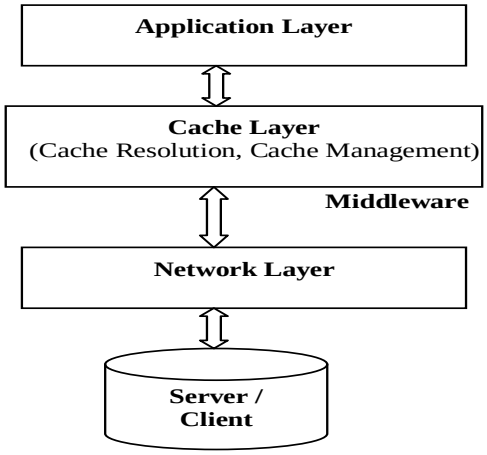

Fig 1. System Architecture in MANET

Each MN has a local database [8] may be either client in some applications or a server to process the requests from other MNs. The fig. 1 shows the MANET's database architecture. It has three layers in which cache layer is the middleware to support the query request processing. The requests processed by the one MNs middleware that results send to other MNs middleware in the network. If the processed results are required, then MN stores the data in its local database.

2. COOPERATIVE CACHING IN MANET

Many researchers provide various techniques in order to access the data item more efficiently. Some of the techniques are described here.

2.1 Simple Cache

For on-demand data access applications, the traditional way of resolving a data request is to check the local cache first and send the request to the data server after local cache misses. This scheme is referred to as Simple Cache [5], [9]. This scheme works well as long as the connection to the data server is reliable and not too expensive; otherwise, it results in failed data requests or request timeouts.

2.2 Cache Data, Cache Path, Hybrid Cache

In Cache Data [5], [9], intermediate nodes check the passing-by data requests. If a data item is found which is frequently requested, intermediate nodes cache the data based on conservative rule. A conservative rule is proposed as follow*:* A node does not cache the data item if all requests for the data item are from the same node. The request is answered by intermediate nodes instead of the database server when the requested data item is in intermediate node's cache. A limitation of this approach is that the data could take a lot of caching space in intermediate nodes. In Cache Path [5], [9], forwarding nodes cache the path to the closest caching node instead of the faraway database server and redirect future requests along the cached path. Cache Path scheme saves caching space compared to Cache Data. But this scheme could introduce more processing overhead when path length is more than 5 hop count. Because of the caching node is dynamic that results recorded path could become obsolete. In Hybrid Cache [3], [5], it avoids the weakness of those two schemes and decides when to use which scheme based on the properties such as passing-by data size and the Time-to-Live value of the data item. Cache Data is adopted in two criteria's. First, if data size is less than the threshold data size 40 KB, the data item only needs a very small part of the cache. Second, if the TTL of data is less than threshold TTL 5000 seconds the data item may be invalid soon. Hence, Cache Path may provide the wrong path to the request and it resending to the data server. If not, Cache path is better. Cache path is adopted in two criteria's. First, when TTL of data item is more than 5000 seconds it is valid for more time. Second, when distance is more than 2 hop count. If not, Cache data is better. Drawback of these schemes is that if the node does not lie on the forwarding path of a request to the data server the caching information of a node cannot be shared.

2.3 Zone Cooperative Cache

In Zone Cooperative (ZC) [7], the set of one-hop neighbors in the transmission range forms a zone. The MNs in the zone forms the cooperative cache. The origin of each data item is held by a particular data source act as server. Data is updated only at server. Each MN stores the frequently accessed data items in its cache. The data items in the cache satisfy not only the node's own requests but also the data requests passing through it from other MNs. Once the data is updates, MNs cached copy becomes invalid. For a data miss in the local cache, the node first searches the data item in its zone (i.e. zone cache hit). After Zone cache miss, the request is forwarded to the neighbor along the routing path. Each MN searches the data in its local cache or zone cache before forwarding the request to the next node that lies on a path towards data server. Once a MN receives the requested data, it cache the data based on server's location. If the server of the data resides in the same zone of the requesting MN, then the data is not cached. Hence, MNs in the same zone can store the different data items that reduce the energy consumption and bandwidth. If there is not sufficient cache for the data, then oldest TTL cached data is replaced. However, the delay may become longer if the neighbors of intermediate nodes do not have a copy of the requested data item for the request. Because requested node wait for the reply before it resend the request to server.

2.4 Group Caching

In Group Caching (GC), each node and its one-hop neighbors form a group [7] within transmission range by sending the "Hello" messages periodically. Each group has a master node which needs to communicate with its members directly through one-hop routing. It increases the communication speed and reduces the delay among nodes in the group. In order to utilize the cache space of each node in a group, the master node checks the caching status of group members using caching control message. Hence, it stores more different data items and increases the data accessibility. In order to record the caching status of group members, each node maintains self table, group table to record caching status of itself and its group members respectively. When a MN receives the request, it checks the self table for local cache hit. After local cache miss, checks the group table to send the request to group members for remote cache hit. After remote cache hit, the request send to the next group along the path to the server. When intermediate group along the path receives the request, it checks its group table for

the data. If yes, it sends the data to the requested MN. Otherwise, the request is forwarded to the server. In GC, place the data in cache based on two schemes. First, MNs store the received data while it has enough cache space. Otherwise, the receiving MN checks the cache status of group members. If the enough cache space is available, it stores the data item in the group member. Second, if the available space of every group member is not sufficient to cache the received data, the receiving MN checks the group table to see either its group member already caches the data or not. If yes, the data item is not cached. Otherwise, the receiving MN selects group member which has the oldest TTL of cached data item in the group table and stores the data item in that group member. When caching placement and replacement need to be performed, the mode selects the appropriate group member to execute the caching task in the group. If MN does not receive the Hello message during predefined cycles from the neighbor, it assumes the neighbor is leaving or shut down. The group table needs to be updated and remove the related records of the leaving neighbor. However all nodes need to maintain the cache status by periodical hello messages. Hence, it consumes energy.

2.5 Global Cluster Cooperative Caching

The Global Cluster Cooperative (GCC) caching [1], [2] partitions the whole network into equal size ($r/\sqrt{8}$) clusters based on the geographical network proximity. Individual MNs interact with each other in a cluster which enhances the system performance in a network. In each cluster area a "super" node is selected to act as Cache State Node (CSN), which is responsible for maintaining the global cache state (GCS) information of different clusters in the network domain. GCS for a network is the list of data items along with their TTL stored in its cache. When a node caches or replaces a data item, its GCS is updated at the CSN. In GCC, when a client suffers from a local cache miss, the client will look up the required data item from the cluster members. If the client cannot find the data item in the cluster member's caches (cluster cache miss), it will request the CSN which keeps the global cache state and maintains the information about the MN in the network which has copy of desired data item. If a cluster other than requesting MN's cluster has the requested data (remote cache hit), then it can serve the request without forwarding it further towards the data server. Otherwise, the request will be satisfied by the data server. The GCC caching scheme reduces the message overheads and enhances the data accessibility. Once a MN receives the requested data, it triggers the admission control based on the server's location to determine whether data need to be cached. If the server of the data resides in the same cluster of the requesting MN, then the data need not be cached. The same data items are cached in different clusters without replication to reduce the delay, bandwidth and energy.

The Cooperative caching schemes uses simple weak consistency than strong consistency model based on TTL. The MN removes the cached data when the TTL expires. In strong consistency, if the cached data is requested by MN, the caching node first asks the server to check the cached data is either valid or invalid. It needs a large bandwidth and energy. In weak consistency, a MN considers a cached copy is up-to-date if the TTL has not expired. The TTL of cached data is refreshed when a fresh copy of that data is passing through the node.

3. PERFORMANCE METRICS

The Performance metrics are average query delay, average hop count and cache hit ratio (include local cache hit, remote cache hit ratio in remote caching node, global cache hit at the server) of data objects.

Cache hit ratio: It is defined as the ratio of number of successful requests to the total number of requests.

Average query delay (T_{avg}): The query delay is the time elapsed between the query is sent and the data is transmitted back to the requester, and average query delay is the query delay averaged over all the queries.

The caching strategies have evaluated with NS2 simulator with AODV routing protocol, database size (N) 1000 items, bandwidth 2Mbps, mean query time (Tq) 5 secs, average node speed (V) 2 m/s, skewness parameter (θ) 0.8, Random WayPoint Mobility model [11].

4. OBSERVATIONS AND RESULTS

As a summary, the comparison of cooperative caching schemes is listed.

4.1 Cache hit Ratio

The following tables have shown the cache hit ratios of MNs under different cache sizes and number of nodes. The cache size is set to 200KB, 400KB, 600KB, 800KB, 1000KB, 1200KB and 1400KB. The size of a data item is 10KB. In general, the cache hit ratio increases while the cache size increased. Table 1 shows the comparison of various caching techniques. The performance of simple cache is very low, because the request is answered either MNs own cache or server. Simple cache does not utilize the neighbor's cache. Cache data is better in small size of data. Hybrid cache utilizes the advantages of cache data and cache path. Hence, it is better than the cache data and cache path. GC uses the group members cache to cache the data. The cumulative cache of group is large which provide better performance. ZC access the more data in its local and zone cache hit than the global hit at the server. GC caching has a higher cache hit ratio than others because it partitions the network into one-hop neighbor clusters. Cluster state node (CSN) maintains the Global cache state information that consist the all cached content's information in the network. The cluster member's stores the distinct data in the cache. These cached data items improve thecache hit ratio based on local hit, cluster hit and remote hit. Table 2 shows the cache hit ratio under the various number of nodes 50, 60, 70, 80, 90 and100. In general, the cache hit ratio increases while the MNs are increased. In simple cache, the increase of MN's results network size will be increased and the MNs local cache could not cache large amount of data. Hence, the request is answered by the server than MNs own cache. In cache data, cache size increases when MNs increased. The cache path stores the path for data with large TTL and hop count. Hybrid cache is better when it avoiding the weakness of cache data and cache path. The cumulative cache size increased in zone and groups with increase of MNs. Hence ZC, GC can cache the more data than others. GCC has more MNs in each cluster with number of MNs which leads to improvement in cluster hit ratio and remote hit ratio. Due to cluster cooperation, GCC performs better than other caching categories by varying number of nodes.

Table 1. Cache hit Ratio Vs Cache Size

Cache Type	Cache Size (KB)						
	200	400	600	800	1000	1200	1400
Simple Cache[5,9]	0.05	0.06	0.15	0.19	0.2	0.21	0.22
Cache Data [5,9]	0.12	0.3	0.5	0.57	0.6	0.62	0.64
Cache Path [4,5,9]	0.14	0.24	0.26	0.28	0.32	0.32	0.28
Hybrid Cache[3,4,5]	0.18	0.25	0.29	0.30	0.31	0.33	0.35
Group Cache [6,8]	0.29	0.49	0.62	0.66	0.71	0.83	0.90
Zone Cache[7]	0.52	0.66	0.8	0.84	0.86	0.88	0.90
Global cluster cache[1,2]	0.59	0.7	0.82	0.9	0.9	0.93	0.94

Table 2. Cache hit Ratio Vs Number of nodes

Cache Type	Number of Nodes					
	50	60	70	80	90	100
Simple Cache[5,9]	0.2	0.23	0.2	0.2	0.2	0.2
Cache Data [5,9]	0.61	0.61	0.62	0.64	0.68	0.7
Cache Path [4,5,9]	0.63	0.62	0.65	0.72	0.76	0.78
Hybrid Cache[3,4,5]	0.74	0.79	0.81	0.83	0.83	0.82
Group Cache [6,8]	0.77	0.81	0.83	0.81	0.85	0.87
Zone Cache[7]	0.84	0.86	0.88	0.9	0.92	0.93
Global cluster cache[1,2]	0.86	0.87	0.9	0.9	0.93	0.95

4.2 Average Query Delay

When cache size is small, less required data items can be stored. The cache data and cache path utilizes the neighbor's cache which reduce the delay. But, simple cache uses its own cache and after local miss it send the request to the server which makes long delay. Hybrid cache is better when it uses the strength of both schemes. In GC, when a MN receives the request it checks its local cache. After local miss it selects group members using group table in the master node. ZC utilizes the local cache and zone member's cache using MNs own table. Hence, ZC reduces the delay than GC. In GCC, each MN maintains its own cluster information and CSN maintains other clusters information. The cached data is frequently received by local hit, cluster hit and remote hit than the server. Table 3 shows the GCC is better than others. As the cache size is large, the MNs can access most of the required data items from local, cluster and remote cache which reduces query delay. Table 4 shows the average query delay under the various numbers of nodes. As MNs increased, the delay also increases because more nodes compete for limited bandwidth. Cache data and cache path is better than simple cache since it uses its local cache only. GC is better than hybrid cache. Because GC utilizes the group members cache to access the data. The query delay of ZC is higher than Global cluster cache scheme. While increasing the number of nodes in the network, the cluster size also is increased. Thus increasing the size of cluster cache and remote caches reduce query delay and increase the cache hit in GCC than other caching types.

Table 3. Average Query Delay (Sec) Vs Cache Size

Cache Type	Cache Size (KB)						
	200	400	600	800	1000	1200	1400
Simple Cache[5,9]	0.34	0.33	0.31	0.30	0.29	0.28	0.29
Cache Data [5,9]	0.29	0.27	0.23	0.20	0.19	0.18	0.17
Cache Path [4,5,9]	0.27	0.24	0.20	0.17	0.16	0.16	0.16
Hybrid Cache[3,4,5]	0.25	0.24	0.18	0.15	0.13	0.13	0.13
Group Cache [6,8]	0.27	0.23	0.23	0.21	0.20	0.20	0.18
Zone Cache[7]	0.26	0.2	0.17	0.16	0.15	0.12	0.10
Global cluster cache[1,2]	0.23	0.16	0.13	0.11	0.11	0.10	0.10

Table 4. Average Query Delay (Sec) Vs No. of nodes

Cache Type	**Number of Nodes**					
	50	**60**	**70**	**80**	**90**	**100**
Simple Cache[5,9]	0.27	0.28	0.29	0.32	0.34	0.35
Cache Data [5,9]	0.18	0.19	0.20	0.22	0.23	0.24
Cache Path [4,5,9]	0.14	0.17	0.17	0.19	0.21	0.22
Hybrid Cache[3,4,5]	0.12	0.16	0.18	0.19	0.19	0.21
Group Cache [6,8]	0.13	0.15	0.17	0.17	0.19	0.22
Zone Cache[7]	0.16	0.16	0.17	0.18	0.18	0.19
Global cluster cache[1,2]	0.09	0.13	0.14	0.15	0.17	0.17

5. CONCLUSION

This paper presents the review and hypothetical analysis of various cooperative caching schemes in mobile ad hoc network. These caching schemes are useful in MANET environment for efficient data access. This paper presents advantageous of these schemes in order to find a data item in a MANET by using less bandwidth, power and improves the performance in terms of cache hit ratio and query delay. The comparison reveals the Global Cluster Cooperative (GCC) caching scheme is better than the other caching schemes. As the cooperative caching is a useful technique to improve the data accessibility and data availability in the MANET, and these analysis will be helpful for the future research in terms of data prefetching, replacement and consistency.

6. REFERENCES

[1] Chand. N, Joshi R.C and Manoj Misra, "A Cooperative Caching Strategy in Mobile Ad Hoc Networks Based on Clusters", International Journal of Mobile Computing and Multimedia Communications, 3(3), 20-35, July-September 2011.

[2] Chand. N and Naveen Chauhan, "Cooperative Caching in Mobile Ad Hoc Networks Through Clustering", ACM Recent Researches in Software Engineering, Parallel and Distributed Systems, 2011.

[3] Bin Tang, Gupta.H, Das. S.R, "Benefit-Based Data Caching in Ad Hoc Networks", IEEE Transactions on Mobile Computing, vol. 7, no. 3, pp. 289-304, March 2008.

[4] Yin. L and Cao. G, "Supporting Cooperative Caching in Ad Hoc Networks", IEEE Transactions on Mobile Computing, vol. 5, no. 1, pp. 77-89, Jan. 2006.

[5] Yin. L and Cao. G, "Supporting Cooperative Caching in Ad Hoc Networks", IEEE INFOCOM, March 2004.

[6] Yi-Wei Ting and Yeim-Kuan Chang, "A Novel Cooperative Caching Scheme for Wireless Ad Hoc Networks: Group Caching", IEEE International Conference on Networking, Architecture, and Storage 2007.

[7] Narottam Chand, R.C. Joshi and Manoj Misra,"Efficient Cooperative Caching in Ad Hoc Networks", IEEE Transactions, pp.1-8, August 2006.

[8] Han Ke, "Cooperative caching algorithm based on grouping nodes in mobile ad hoc networks", ICIA, IEEE June 2010.

[9] Guohong Cao Liangzhong Yin Chita R. Das "Cooperative Cache-Based Data Access in Ad Hoc Networks", IEEE Computer Society, 2004.

[10] Hara.T and Madria.S.K, "Data Replication for Improving Data Accessibility in AdHoc Networks" IEEE Transaction on Mobile Computing Vol.5.No.11pp.1515-1532, 2006.

[11] Bettstetter.C, Resta.G and Santi.P, "The Node Distribution of the Random Waypoint Mobility Model for Wireless Ad Hoc Networks" IEEE Transactions on Mobile Computing Vol.2.No.3, pp 257–269, 2003.

[12] Chow C.Y., Leong H.V. and Chan A. "Peer-to-Peer Cooperative Caching in Mobile Environments", Proceedings of the 24th International Conference on Distributed Computing Systems, pp. 528-533, 2004.

[13] Lim S., Lee W.C., Cao G. and Das C.R, "A Novel Caching Scheme for Improving Internet Based Mobile Ad Hoc Networks Performance", Elsevier Journal of Ad Hoc Networks, Vol. 4, No. 2, pp. 225-239, 2006.

[14] Lim S., Lee W.C., Cao G. and Das, "Performance Comparison of Cache Invalidation Strategies for Internet Based Mobile Ad Hoc Networks", Proceedings of IEEE International Conference on Mobile Ad Hoc and Sensor Systems, pp. 104-113, 2004.

7. AUTHORS PROFILE

P.Kuppusamy is currently working as an Assistant Professor in Computer Science and Engineering, Vivekanandha College of Engineering for Women, Namakkal. He has completed master's degree in 2007. He is a member of CSI INDIA and carrying out PhD research work in association with Anna University of Technology, Coimbatore. His current areas of research interests are ad hoc networks and mobile computing.

Dr.K.Thirunavukkarasu received the B.E (Metallurgy with University Ranking) in Madras University in 1975 and M.Tech(Metal casting) in 1977and Ph.D degree in Mechanical Engineering from IIT Madras in 1983. He is currently working as a principal in T.J.S Engineering College, Chennai. He is a member of MSITE, IWS INDIA. His current areas of research interests are robotics, Sensors, Mobile Computing, Signal Processing and Real-time Controls Integration.

Dr.B.Kalaavathi received the B.E. (Computer Science and Engineering) degree in 1993 from Bharathiyar University, M.Tech from Pondicherry University in 2000 and Ph.D from Periyar University in 2010. She is currently working as a Professor in Information Technology, K.S.Rangasamy College of Technology, Namakkal. She is a member of CSI & ISTE INDIA. Her current areas of interest include Mobile Computing, Data Structures and Algorithms Analysis.