A Location Aided Cooperative Caching Protocol for Mobile Ad hoc Networks

Preetha Theresa Joy
Dept. of Computer Science,
Cochin University of Science and Technology,
Kochi ,Kerala ,India.

K .Poulose Jacob,
Dept. of Computer Science,
Cochin University of Science and Technology,
Kochi, Kerala ,India.

Abstract— **Cooperative caching is an attractive solution for reducing bandwidth demands and network latency in mobile ad hoc networks. Deploying caches in mobile nodes can reduce the overall traffic considerably. Cache hits eliminate the need to contact the data source frequently, which avoids additional network overhead. In this paper we propose a data discovery and cache management policy for cooperative caching, which reduces the caching overhead and delay by reducing the number of control messages flooded in to the network. A cache discovery process based on location of neighboring nodes is developed for this. The cache replacement policy we propose aims at increasing the cache hit ratio. The simulation results gives a promising result based on the metrics of studies.**

Keywords- Cooperative caching; Data dissemination; Cache Replacement.

I. INTRODUCTION

Rapid growth of wireless technology has increased the wide popularity of mobile ad hoc networks. However, frequent disconnections, limited energy reserves and bandwidth constraints, makes efficient data access in ad hoc networks a challenging task. Connection refusals due to lack of resources occurs when the server workload is increased. These problems are exacerbated by the dynamic topology of ad hoc networks. Data caching is recognized as feasible approach to improve the performance in many traditional systems. Caching is the process of pre-fetching the needed data and storing it closer to the source. In mobile ad hoc networks, caching can lead to significant bandwidth savings, perceived network latency reduction and ensures higher data availability. Data caching in ad hoc networks are mainly proposed as cooperative caching [6]. Cooperative caching aims to reduce the redundant data transfer using a mechanism which enables the local cache of different mobile clients to be shared in a cooperative manner. In cooperative caching the mobile clients are configured to request the data object from its local set of data items, if not found queries its neighboring nodes. When there is a cache miss in the neighboring nodes queried, the data item is retrieved directly from the server and this procedure continues recursively.

In mobile ad hoc networks, caching can improve mobile client perception in three ways. First, retrieving data from a remote data center involves wireless media network transfers and there is a chance of data loss due to the wireless link characteristics. Second, when the data is served locally, the network latency is hidden from the user. The data center processes the data request only when there is a local and cooperative cache miss. By doing this the server load is balanced, which consequently reduces the latency in serving client request. Thirdly, frequent disconnections which occur in ad hoc networks can be hidden from users, making the network more reliable.

Cooperative caches have different functions: Cache discovery, placement and replacement, consistency maintenance and data dissemination. Discovery refers to how a mobile node locates the cached data. Placement is the processes of selecting appropriate nodes as cache locations and replacement is the strategy used for evicting data when the cache is full. Consistency maintenance is maintaining consistency among the source data and cached copies. Our ultimate goal is to design a cooperative caching technique which minimizes the cache overhead and the average response time of data retrievals. The proposed algorithm investigates cache discovery and cache replacement in cooperative caching for mobile ad hoc networks.

A. Motivation and contribution

The performance of a cooperative cache greatly depends on the overhead and efficiency of data discovery and dissemination. Basically, in cooperative caching there are two main approaches for cache discovery: pull based and push based. In pull based technique [15], [6] when a mobile node tries to locate a cache in the neighboring nodes, it will broadcast a request message to these nodes. Although this method is simple, it relays on flooding to broadcast the data request. Flooding increases network contention and overhead when the network density is high. With push based technique [4], a mobile node maintains information about other node's cache contents. When a cache event occurs, it is informed to all other neighboring nodes. However, this generates a significant overhead not only on the network, but also on other nodes due to message processing. In addition, to locate a cache for a particular data object, a node has to maintain the information about other node's cache contents. This can consume significant amount of resources when the network density is high. The work proposed in [2] tries to solve this by using centralized cache control by selecting one node as coordinator, which

maintains the information about all cooperative caches. The disadvantage of this approach is that it is not scalable. The control node may get disconnected which causes excessive overhead [4].The number of entries in the look up table increases when the network density is high.
Motivated by the weakness of above mentioned schemes, we propose a cooperative caching protocol which uses a distributed cache discovery and dissemination mechanism. The basic idea is to minimize the cache overhead and delay by reducing the number of control messages flooded in to the network. The one hop neighbor list is divided in to two zones based on transmission range and a location based data discovery is adopted. This eliminates not only the flooding of data request into the network, but also reduces the overhead in processing tables. The cache hit ratio of cooperative cache depends mainly on the replacement algorithm. The replacement algorithm used in this scheme is an extension of the basic LRU scheme, which takes in to account the inter arrival time of recent references, size and consistency for page replacement.

The remainder of the paper is structured as follows. Section II reviews the related works in cache resolution, section III presents the system architecture of the proposed cooperative caching scheme, section IV describes simulation model and metrics, section V gives the experimental results and section VI concludes the paper.

II. PREVIOUS WORK

Caching data in mobile nodes is an effective technique to improve performance in a mobile environment. Recently, several schemes for cooperative caching in mobile ad hoc networks have been presented in the literature. The algorithms proposed in [7],[8],[15], focuses more on cache placement and discovery while[6],[9],concentrates more on cache management protocols. Cache management includes cache admission and replacement. The work [2],[4],[5],[11],[10] considers both aspects in cooperative caching. The following section reviews some of the works that are related to the proposed work.

COCAS[15] is a distributed caching scheme designed for finding 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) 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. In [1] the authors try to find the best location for optimal cache placement. Cluster cooperative caching [5] uses a centralized cache management scheme in which a cluster head has the responsibility for maintaining the cached data information within a cluster domain. For a highly dynamic topology, this approach may not be feasible since the cluster head may change frequently. Based on the above observation, a location based cache discovery algorithm is proposed to find the cached data in the neighboring nodes.

Caching in wireless environment has unique constraints like scarce bandwidth, limited power supply, high mobility and limited cache space. Due to the space limitation, the mobile nodes can store only a subset of the frequently accessed data. A good replacement mechanism is needed to distinguish between the items to be kept in cache and that is to be removed when the cache is full. Numerous cache replacement algorithms were proposed in web caching but only a few were proposed for ad hoc networks. Most of the existing replacement policies for cooperative caching uses LRU as the replacement policy [3], [4], [7],[10]. LRU considers only the time of the last access for making the replacement decision and it does not favor the data item has been referenced earlier. The replacement schemes mentioned in [2], [6] make use of a value function based on different parameters to replace the data. Since the relative importance of the parameters used in calculating the value function can vary from one type of request to another, some policies are needed to adjust the weights dynamically to achieve the best performance. Hence, we design a key based algorithm for cache replacement to keep the frequently accessed data items in the cache.

III. SYSTEM ARCHITECTURE

A. Network Model

A mobile ad hoc network is abstracted as a graph G (V, E), where V is the set of nodes and $E \subseteq V^2$ is the set of links which gives the available communication. An edge (u, v) belongs to E means that there is direct communication between two nodes u and v. The elements of E depend on the position and the communication range of nodes. All links in the graph are bidirectional i.e., if u is in the transmission range of v, v is also in the transmission range of u. The maximum communication range is assumed to be same for all nodes and is represented as R, which is given by the Euclidean distance d (u, v) between nodes u and v. The set of neighborhood nodes in the range R are represented as N_R (U) and the set of neighborhood nodes in the range R1, N_{R1} (U) represents the mobile nodes in the primary zone and

$N_{R1}(U) \setminus N_R(U)$ gives the neighbor node set in the secondary zone.

B. System Model

We assume a mobile ad hoc network with a set of nodes which are able to communicate with each other. The transmission radius R determines the maximum communication range of each node and is equal for all nodes in the network. 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. We assume that each node knows its current location precisely with the availability of Global Positioning System (GPS).

Initially, to find the neighbor node set in the transmission range R for node *A* $N_R(A)$, 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 a 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 *current location coordinates*. Upon receiving the location coordinates of neighboring nodes, the source node measures the distance Dij between the source node ni and neighbor nj using the formula,

Dij= $\sqrt{(xi - xj)^2 + (yi - yj)^2}$

When a node is farther away from the source the Euclidean distance will be large. Each node will arrange the neighboring node list in the descending order of their distance to determine the order of neighboring nodes to receive the data request. To reduce the number of messages, the neighboring node set is divide in to two zones, a primary zone and a secondary zone .The primary zone contains nodes which are closer to the source node that comes under transmission radius R1 and the secondary zone contains the nodes that comes in the range R1 and R. To maintain the neighbor node set accurately, each node periodically sends a request control message to its neighbors.

Each node maintains a list which stores the cached data item. The list contains the following fields: cached data id, cached data item, ttl, time difference between the current access and previous access. This table is updated when a new data is placed in to the cache .The cache space for each node is limited and when it is full, a replacement strategy evicts the unwanted data. The contents of the local cache are shared by its neighboring nodes.

The data center is assumed to be a fixed location. The data center maintains a set of data items uniquely identified by means of data item id D_i for1<=i<= n where 'n' is the size of the data base. The size of each data item varies from S_{min} to S_{max}. Each node has a local cache, with certain data items. Each mobile nodes is identified by a distinct <Host id, Name> for 1<=i<=N, where N is the density of the network. Nodes in the network retrieve data items either from the local cache or from the neighbouring cache if there is a local miss. When a node fails to find data in neighboring nodes, data is retrieved from the data center. When a node receives a fresh data directly from the server, it caches a copy of it in the local cache and becomes a provider for that cached content for the neighboring nodes.

When a node wants to access data, it checks in its own local cache. If the requested data is not cached, the node checks whether the data is present in the neighboring nodes. If we are not able to find the data from the neighbor list the request is given to the data center.

A. Cache Discovery

The process of cache discovery is triggered when there is a local cache miss. We use a location aided discovery to reduce the message overhead. The goal of this scheme is to reduce the average number of messages among the cooperative caches while maintaining high cache hit ratio. In this algorithm, the decision to forward the data request is based solely on the location of itself and its neighboring nodes. When the requested data is not found in the local cache the request is forwarded to the nearest neighbors present in the primary zone. After sending the request the node waits for the reply .If the node doesn't receive a positive reply after a period of time t1, which is a predefined threshold, the node searches the data in the secondary zone. The nodes in the secondary zone will be at greater distance compared to the primary zone. The time out period is set to t2, which is higher than time out set for the primary zone. The process of cache discovery is fully distributed and runs in all the nodes in the network.

If the nodes can adjust their transmission power for each node based on distance, the power consumption can be reduced considerably, which leads to increased battery life. In [12] the authors present a general model for the power consumption between two nodes with a distance d .The power consumed P(d) is given by a node in transmitting a request for a distance d is given by

$P(d) = d^{\alpha} + c$ for some constants α and c. If we ignore the constant we can see that the power consumption is directly proportional to distance. Since the proposed cache discovery considers distance as a parameter for finding the cached data the power consumption is reduced.

B. Cache Placement and Replacement

When there is a local miss the data item is fetched either from the neighboring nodes or from the server. The cache placement module is triggered when the data item is brought in, to decide whether to cache or not the incoming data. In order to cache more distinct data the caching decision is done based on two parameters, size and distance. We set a threshold value T, which is 50 % of the cache size for a data item to be admitted to cache. The data coming from the neighboring nodes are also not cached in order to increase the data accessibility.

In cooperative caching if data replacement decision is made by individual nodes by considering only their local cache, the performance is degraded because the data may be present in the neighboring nodes. In order to cache more

distinct data, new data item fetched from adjacent nodes are not cached. When the cache is full, appropriate data from the cache have to be evicted to make room for the incoming data. The replacement policy proposed here considers the number of references for a particular data item and gives more emphasis data items that are referenced more than once. If we have data items referenced only once then that set is given priority for replacement. For this LRU policy is used. If an item is referenced more than once the inter arrival time between the recent two references is considered for eviction.

Let t_c be the current reference time and t_r be the previous reference time then T_{int}, the inter arrival time is given by (1)

$T_{int} = t_c - t_r$ (1)

If tc - t_r = 0 , an item whose last reference time is smaller is replaced .

If Tint > 0, then the replacement decision is made on the value of K (i) which is given as (2)

K(i) = Max $\sum_{i=1}^{k} t_i - t_r$ (2)

The data items with maximum inter arrival time is considered for replacement. In both cases if more than one data item have the same value, the TTL parameter is taken and the one with lower TTL value is removed as the data with lower TTL will be outdated soon.

IV. SIMULATION STUDY

We have developed a simulation model in JAVA. The proposed scheme is a fully distributed scheme, with each node runs an application to request, retrieve and cache data items from other neighboring nodes. Each node caches part of the requested items temporarily. The simulated mobile environment consists of a number of mobile nodes which are randomly placed in an area of 1000X1000m^2. Each node is identified by a node id and a host name. The position of each node is given by the x and y coordinates. The data centre is implemented in a fixed position in the simulation area. The data center 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 center contains 1000 data items, with each item identified using a data id. The nodes that generate data request are selected randomly and uniformly. The time interval between two consecutive queries generated from each node follows an exponential distribution with mean of 10sec. Each mobile node generates a single stream of read only queries. The queries generated follows a Zipf distribution [14] ,which is frequently used to model non uniform distribution. In Zipf distribution, an item ranked i(1≤ i≤ N) is accessed with a probability ,

$$P_N(i) = \frac{1}{i^{\theta} \sum_{k=1}^{N} \frac{1}{k^{\theta}}}, 0 < \theta \leq 1$$

The value of θ ranges between zero for uniform distribution and one for strict zipf distribution. In our study θ is set to 0.8. The data request is processed in FCFS manner at the server. An infinite queue is used to buffer the request when the data center is busy. Each miss in the cooperate cache will incur a delay of 8 ms to retrieve data from the data center.

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 pause for 200 seconds. After that it moves to the new destination with speed *s'*. For the time out mechanism each node maintains two time out period for the primary and secondary zone. The details of the simulation parameters are given in table 1.

A. Simulation Parameters

Parameter	Value
Simulation Time	3600sec
Simulation Area	1000X1000 m^2
Database	1000 items
Cache size	20 -70 % of total database size
S_{min}	1
S_{max}	10
Nodes Density	20-70
Mobility model	Random waypoint
Transmission Range	500m
Transmission range primary zone	250m
Speed of the mobile host	1-10
Pause time	200sec
Mean query generation time	10s

Table.1 Simulation Parameters

B. Metrices

The performance metrics evaluated includes cache hit ratio, message overhead and average query delay. The evaluation of these parameters are done by varying the number of cache locations with respect to number of nodes and the behavior of cache hit ratio for different cache sizes. The hit ratio is defined as the percentage of requests that can be served from previously cached data. Since the replacement algorithm decides whether to cache the data or not, it affects the cache hits of future requests. The query delay is the time interval between the query sent and the data transmitted back to the requester. Average query delay is the query delay averaged over all queries. Message overhead is the overhead messages needed to manage the cache discovery process in cooperative cache.

V. EXPERIMENTAL RESULTS

To evaluate the performance of the proposed cache discovery and cache replacement scheme, we compared the performance of our new cooperative caching scheme (CCN), with Neighbor Caching (NC), a caching scheme which uses broadcasting and LRU for cache replacement.

Fig 1 shows the message overhead for two schemes, as a function of different node densities. The figure shows that CCN outperforms NC at all node densities. As the node density increases, the difference become more significant, this implies that CCN can benefit from larger node densities. Fig 2 depicts the comparison of message overhead for different cache sizes. Again we can see that the message overhead is significantly less for CCN. From figure 3 we can see that the cache hit ratio for CCN is greater than NC for different cache sizes. The relative performance of cache hit ratio remains relatively stable for higher cache sizes. From Figure 4 we can see that the query delay decreases with increased cache size.

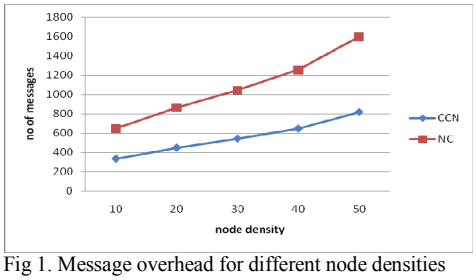

Fig 1. Message overhead for different node densities

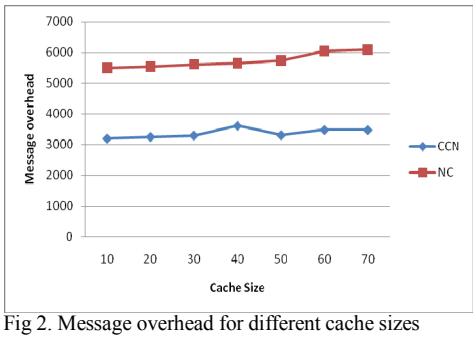

Fig 2. Message overhead for different cache sizes

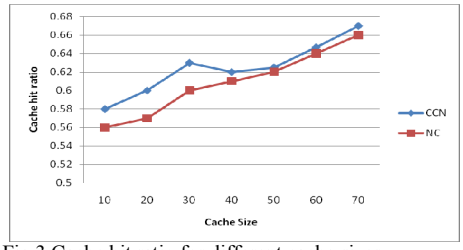

Fig.3 Cache hit ratio for different cache sizes

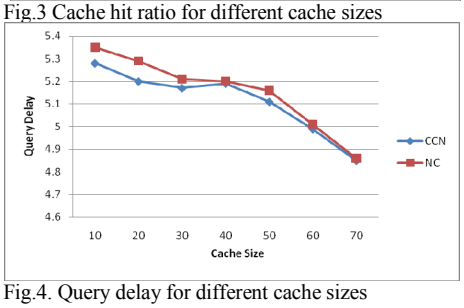

Fig.4. Query delay for different cache sizes

VI. CONCLUSION

In this paper we addressed the problem of data discovery and cache management policies for cooperative caching in ad hoc networks. The objective of our problem was to minimize the number of messages flooded in to the network, which in turn reduces the communication cost and bandwidth utilization. We designed a data discovery process based on the position of the neighboring nodes. The cache replacement policy proposed increases the cache hit ratio compared to that of the most commonly used LRU scheme. Experimental results show that the proposed approach can significantly improve the performance in terms of message over head, cache hit ratio and average query delay.

REFERENCES

[1] Zhao, P Zhang, G Cao, CR Das," Cooperative caching in wireless P2P networks: design, implementation and evaluation", IEEE Trans Parallel Distributed Syst. 21(2), 229 (2010).

[2] N Chand, RC Joshi, M Misra, " Cooperative Caching Strategy in Mobile Ad Hoc Networks Based on Clusters ", Wireless Personal Communications (2007) 43:41–63.

[3] 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).

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

[5] S Lim, WC Lee, G Cao, CR Das, "*A novel caching scheme for improving Internet-based mobile ad hoc networks performance*", Ad Hoc Netw. 4 (2),225 (2006).

[6] L Yin, G Cao," *Supporting cooperative caching in ad hoc networks*", IEEE Trans Mobile Comput. 5 (1), 77 (2006)

[7] M Fiore, F Mininni, C Casetti, DF Chiasserini, "*To cache or not to cache?*", In Proceedings of the IEEE Conference on Computer and Communications (INFOCOM 2009) , Rio de Janeiro, Brazil 235 (2009).

[8] B Tang, H Gupta, SR Das," *Benefit-based data caching in ad hoc networks*",IEEE Trans Mobile Comput. 7 (3), 289 (2008)

[9] Denko, M.K., Tian, J.," *Cross-Layer Design for Cooperative Caching in Mobile Ad Hoc Networks"*, Proc .of IEEE Consumer Communications and Networking Conf(2008).

[10] Y. Du,Gupta, S. K. S. "*COOP-A cooperative caching service in MANETs,"* in Proc. Joint Int. Conf Autonomic and Autonomous Systems and Int. Conf. Networking and Services,. 58-63(2005).

[11] Dan Hirsch, Sanjay Madria " *A Resource- Efficient Adaptive Caching Scheme for Mobile Ad-Hoc Networks* ", 29th IEEE International Symposium on Reliable Distributed Systems.(2010).

[12] González-Cañete et al : "*A cross layer interception and redirection cooperative caching scheme for MANETs* ",.EURASIP Journal on Wireless Communications and Networking 2012.

[13] Niels Sluijs, Frédéric Iterbeke, Tim Wauters, Filip De Turck, Bart Dhoedt and Piet Demeester," *Cooperative Caching versus Proactive Replication for Location Dependent Request Patterns*", Journal of Network and Computer Applications,Vol.34,Issue 2,2011, pages 562-574.

[14] Tiance Wang , Pan Hui, Sanjeev R. Kulkarni, Paul Cuff, "*Cooperative Caching based on File Popularity Ranking in Delay Tolerant Networks*", Proc. Of ExtremeCom , 2012, Zürich, Switzerland.

[15] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Sheker, "*Web Caching and Zipf-Like Distributions: Evidence and Implications*", Proceedings of IEEE INFOCOM, pp.126-134,March 1999

[16] Hassan Artail et.al, "*COACS: A Cooperative and Adaptive Caching System for MANETs*", IEEE TRANSACTIONS ON MOBILE COMPUTING, VOL. 7, NO. 8, AUGUST 2008.

[17] Chi-Yin Chow, Hong Va Leong and Alvin Chan " *Peer-to-Peer Cooperative Caching in Mobile Environments*", Proc. of 24th International Conference on Distributed Computing Systems Workshops2004. (ICDCSW'04) .

[18] Preetha Theresa Joy and K. Polouse Jacob,"*Cooperative Caching Techniques for Mobile Ad hoc Networks*", Proc. of the Intl. Conference on Data Science & Engineering (ICDSE) 2012, Kochi, India ,pp 175-180.

[19] S. Jin, A. Bestavros," *Greedy Dual Web caching algorithm: exploiting the two sources of temporal locality in web request streams*", Computer Communications, vol. 24, no. 2, Feb. 2001, pp. 174-83.

[20] 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).