Prefetching based Cooperative Caching in Mobile Adhoc Networks

Naveen. Chauhan, and L.K. Awasthi

Abstract—To reduce the query latency and improve the cache hit ratio, prefetching can be used. In the prefetching, access to remote data is anticipated and the data is fetched before it is required. Since caching and prefetching are both well recognized for improving client perceived response time, the integration of both strategies may be exploited to improve the system performance. In this paper we proposed a cooperative caching scheme based on prefetching for mobile adhoc networks (MANETs). With prefetching as a integrated part of data caching, will try to sense the future needs of mobile nodes (MNs). Once it is established regarding the requirement of a particular data item, data is fetched and kept in the cache before it is required. This will reduce the query latency & improve the data availability. Experiments are used to evaluate the performance of proposed scheme. Compared to GCC, Proposed GCCP scheme can greatly improve the system performance in terms of average query latency, mean query generate time and local byte hit ratio.

Keywords—Prefetching, Cooperative Caching, Data Mining, Association Rules.

I. INTRODUCTION

Since the wireless bandwidth is limited, prefetching should be used to improve the cache hit ratio and query latency. If the requested data item is not prefetched earlier, the mobile node has to send a request to the cooperating nodes or server when a query demands the item. This not only increases the query latency but also the wireless bandwidth requirement. However, prefetching consumes a lot of system resources. For example, although a client can prefetch the data item routed through it, the clients still consumes power to process the prefetched data. Since mobile nodes are powered by batteries, it is important to prefetch the right data. To achieve effective prefetching, data mining techniques can be used. Data mining is the research area that deals with finding associations among data items by analyzing a collection of data [19, 20]. With caching, the data access delay is reduced since data access requests can be served from the local cache, thereby obviating the need for data transmission over the scarce wireless links. As mobile nodes in adhoc networks may have similar tasks and share common interest, cooperative caching, which allows sharing and coordination of cached data among multiple nodes, can be used to reduce the bandwidth and power consumption. Since caching and prefetching are both well recognized for improving client perceived response time, the integration of both strategies may be exploited to improve the system performance. In mobile ad hoc networks cache misses are not isolated events, and a cache miss may followed by a series of cache misses. Therefore data mining association rules may be used to find relationship among data items and hence perform the prefetching. In the prefetching, access to remote data is anticipated and the data is fetched before it is required.

In this paper we proposed a novel algorithm “Global Cooperative Caching with Prefetching” (GCCP) that prefetches the data based on association among data items. Our scheme prefetches highly related data items and considers confidence of association rules. To find the relationship among data items, association rules base data mining technique is used [4]. To enhance the caching performance in GCCP, the generated caching rules are used to prefetch the data item(s). Simulation is performed to evaluate the performance of our algorithm under several circumstances. Based on caching rules, prefetching is performed, and confidence value alongwith other caching parameters is used during prefetching. To perform the prefetching, we generate a set of related items called Association Set (AS).

The remainder of this paper is organised as follows. In Section II we briefly review the related studies on cache replacement and prefetching in mobile ad-hoc networks and mobile environment. Section III gives description of system model. Section IV describes the proposed algorithm. Section V evaluates the performance of proposed scheme with GCC. Section VI concludes the paper and describes the scope of future work.

II. RELATED WORK

Please submit Caching frequently accessed data on client side is an effective technique to improve performance in mobile ad-hoc networks [1]. A lot of research has been done on cache invalidation in past few years [3, 11], with relatively little work being done on cache prefetching methods. In the following, we briefly review related studies on prefetching in mobile environments.

Prefetching has been widely employed to reduce the response time in the web environment [12, 13]. Most of these techniques concentrate on estimating the probability of each file being accessed in near future. Since these techniques are designed for point-to-point communication environment, they are not suitable for mobile ad-hoc networks. Also they do not consider the constraints of mobile ad hoc networks. In [14],

Naveen Chauhan. is with the National Institute of Technology, Hamirpur, INDIA (e-mail: naveenchauhan.nith@gmail.com).

L.K. Awasthi Senior Member IEEE is with the National Institute of Technology, Hamirpur, INDIA (e-mail: lalit@nitham.ac.in).
Cao proposed an adaptive prefetch scheme, where clients record the number of times a cached item being accessed and prefetched respectively. The client calculates the Prefetch access ratio (PAR), which is the no. of prefetches divided by the no. of accesses for each item. If PAR is less than one, prefetching item is useful since prefetched data may be accessed multiple times. Gitzenis and Bambos [15] proposed a prefetched scheme considering the quality of wireless channel. Clients prefetch aggressively when channel quality is good but reduce the prefetch rate when the channel quality becomes poor. Yin and Cao [10] proposed a power aware prefetch scheme called value-based adaptive prefetch (VAP) scheme, which can dynamically adjust the no. of prefetches based on the current energy level to prolong the system running time. All the previous schemes have ignored the relationship between the data items. Song and Cao [9] realized that cache misses are not isolated events, and a cache miss is often followed by a series of cache misses. They addressed the prefetching issue among related data items by using a cache-miss-initiated prefetching scheme, which is based on association rule mining technique, but the disadvantage of this scheme is that the item appearing in the consequent of mined prefetched rules can not be prioritized. M. Denko et. al [5] proposed a two level prefetching strategy based on prefetching at the level of caching agent and mobile host and other one between caching agent and a cluster head. A popularity based prediction algorithm for prefetching the data item was proposed. This calculates the popularity of a data item based on past access history. CacheData and CachePath have been proposed in [21]. Unlike the previous methods, these protocols do not rely on flooding or broadcast for discovering a cached copy of the requested data item. With CacheData, intermediate nodes may cache a data item to serve future requests while forwarding the item for another node. In contrast, CachePath caches the information of a path to the request originator and uses the information to redirect future requests to the nearby caching site. The hybrid protocol HybridCache combines CacheData and CachePath in an attempt to avoid their weakness. In HybridCache, when a mobile node forwards a data item, it caches the data or the path based on some criteria. These criteria include the data item size and time-to-live (TTL) of the item. Sailhan et. al [22] have proposed a cooperative caching scheme to reduce the communication and energy costs associated with fetching a web object. The communication is based on the notion of terminal profile. However, if data correlation between mobile terminals is small, the effect of terminal profile will be lost.

III. SYSTEM MODEL

This paper studies the prefetching of requisite data in mobile ad hoc networks. Figure 1 depicts a typical system model used for study. As illustrated, there is a cache management mechanism in a client. The client employs the rule generating engine to derive caching rules from the client’s access log. The derived caching rules are stored in the caching rule depository of the client. In the paper we have assumed Global Cluster Cooperation (GCC) [17] among the mobile nodes of MANETs. In GCC when a client suffers from a cache miss (called the local cache miss), the client lookup the required data item from the cluster members.

Only when the client can not find the data item in the cluster members’ caches (called cluster cache miss), it will request the cache state node (CSN) which keeps the global cache state (GCS) and maintains the information about the node in the network which has copy of desired data item. If a cluster other than requesting nodes’ cluster has the requested data (called remote cache hit), then it can serve the request without forwarding it further towards the server. Otherwise request will be satisfied by server.

The Proposed scheme is based on the observation that data items queried during a period of time are related to each other. The choice of prefetching items can depend, in general, on a number of previously accessed items. For instance, suppose data item dX is cache missed at the client and d1, d2, & d3 are closely related to dX. It is observed that when the client accesses dX, it is more likely that d1, d2, and d3 may be accessed in the near future. So to avoid cache misses the client may prefetch d1, d2, and d3. Depending on the storage size, it may prefetch one, two or all the three items. The priority of items, d1, d2, and d3 depends on their association with the item dX.

A. Prefetching

Whenever a mobile node issues a request, the cache request processing module first logs this request into record and checks whether the desired data item is available in local cache of MN or in any of the MN in the cluster. If it is a cache hit, the cache manager still needs to validate the consistency of the cached item with the copy at the original server. To validate the cached item, the cache manager checks the validation of data item from its TTL value. If the data item is verified as being up to date, it is returned to the MN immediately. If it is a cache hit, but the value is obsolete, the cache manager sends an uplink request to the server and waits
for the data broadcast. When the requested data item appears, the cache manager returns it to the requester and retains a copy in the cache. In the case that a cache miss occurs, the client cache manager checks the caching rule depository to derive the prefetching rules corresponding to the requested item. If this request triggers some prefetching rules, the ids of the item implied by these prefetching rules will also be piggybacked to the server along with id of missed cache item.

B. Generation of Association Set

Number To generate the association set only the rules with one antecedent are used. The generation procedure can be summarized as follows. Assume that the client has a cache miss to item dx. Our scheme finds out the rules whose antecedent is dx. We call these rules the prefetch rules. We store the consequent of each prefetch rule along with the confidence value in a linked list of such tuples. The tuples of this list are then sorted in descending order of their confidence values.

For instance, if there is a cache miss for item dx, then the set Zx of prefetch rules is generated from the caching rule set Z for the item dx. The prefetch set PSx for dx is defined as $PS_x = \bigcup (r_{xy} : r_{xy} \in Z_x)$. The tuples in PSx are stored in linked list in descending order of confidence values.

C. Prefetching based on Association Set

Here we define the proposed prefetching strategy. When a cache miss happens, the prefetch set for the cached miss item is generated. Instead of only requesting the cache miss item, the client also requests the items indicated in the prefetch set. When a cooperating node or server receives the request, it transfers the requested items over the wireless channel. The client downloads the items and stores them in its cache. By prefetching the items, the client can save future requests and reduce the query latency.

IV. PROPOSED ALGORITHM

A. GCCP Algorithm

In this section, we present algorithms for caching and prefetching in mobile ad hoc networks. Figure 2 shows the flowchart for performing prefetching.

Algorithm 1 Caching algorithm
When a MN generates a query for data item di
update access log;
if $d_i$ is available in local cache then
return $d_i$ to MN;
else
check CCS
if $d_i$ available within cluster then
return $d_i$ to MN;
else
check the GCS
if $d_i$ available with some neighboring cluster then
return $d_i$ to MN and update CCS;
else get $d_i$ form data center and update GCS and CCS

Algorithm 2 Prefetching algorithm
generate association set ASi for the item $d_i$
/* there is a cache miss for item $d_i$ */
send out a request for all items $d_j \in AS_i$;
get $d_j$ as it passes through the node;
if $d_j$ is already in the cache then
/* TTL has expired */
update $d_j$ in the cache;
return $d_i$ to MN and update GCS and CCS;
if there is enough free space then
insert item $d_j$ into the cache;
Delete the item from the local cache; /* remove the least profitable item based on PBR strategy */
insert $d_i$ into the local cache;
update GCS and CCS;

Fig. 3 GCCP Algorithms

V. SIMULATION RESULTS

The simulation area is assumed of size 1500 m x 1500 m. The clients move according to the random waypoint model [18]. The time interval between two consecutive queries generated from each client follows an exponential distribution with mean $T_q$. Each client generates accesses to the data items following Zipf' distribution with a skewness parameter $\theta$. There are $N$ data items at the server. Data item sizes vary from $s_{\text{min}}$ to $s_{\text{max}}$ such that size $s_i$ of item $d_i$ is,

$$s_i = s_{\text{min}} + \left\lfloor \text{random}() \cdot (s_{\text{max}} - s_{\text{min}} + 1) \right\rfloor, \quad i = 1, 2, \ldots, N,$$
where random() is a random function uniformly distributed between 0 and 1. The simulation parameters are listed in Table 3. For performance comparison of GCC with prefetching, the other scheme GCC in general [17] is also implemented. In GCC, the scheme does have cooperation among various clusters at global level, but the scheme doesn’t anticipate the future requirements of MNs. In our experiments, we have tried to incorporate the prefetching of data items depending upon their utility.

**Table I**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default Value</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database size (N)</td>
<td>1000 items</td>
<td></td>
</tr>
<tr>
<td>(s_{\text{min}})</td>
<td>10 KB</td>
<td></td>
</tr>
<tr>
<td>(s_{\text{max}})</td>
<td>100 KB</td>
<td></td>
</tr>
<tr>
<td>Number of nodes (M)</td>
<td>70</td>
<td>50–100</td>
</tr>
<tr>
<td>MN cache size (C)</td>
<td>800 KB</td>
<td>200–1400 KB</td>
</tr>
<tr>
<td>MN speed ((v_{\text{min}}\sim v_{\text{max}}))</td>
<td>2 m/s</td>
<td>2–20 m/s</td>
</tr>
<tr>
<td>Bandwidth (b)</td>
<td>2 Mbps</td>
<td></td>
</tr>
<tr>
<td>TTL</td>
<td>5000 sec</td>
<td>200–10000 sec</td>
</tr>
<tr>
<td>Pause time</td>
<td>300 sec</td>
<td></td>
</tr>
<tr>
<td>Mean query generate time (T_q)</td>
<td>5 sec</td>
<td>2–100 sec</td>
</tr>
<tr>
<td>Transmission range ((r))</td>
<td>250 m</td>
<td>25–250 m</td>
</tr>
<tr>
<td>Skewness parameter ((\theta))</td>
<td>0.8</td>
<td>0–1</td>
</tr>
</tbody>
</table>

**A. Effect of Cache Size**

We investigated the performance of various algorithms under different cache sizes. The Simulation results are shown in Figure 4. Our algorithm outperforms the other ones in terms of local byte hit ratio.

**Figure 5** shows the effects of cache size on average query latency on GCC, GCC with PBR, GCCP schemes. For all the algorithms, the average query latency drops when cache size increases since fewer queries are generated and the server can serve queries more quickly.

**B. Effect of Mean Query Generate Time**

This section explores the influence of mean query generate time \(T_q\) on local byte hit ratio and average query latency. Each client generate queries according to the mean query generate time. The generated queries are served one by one. From Figure 6 it is evident that for all the schemes, the local byte hit ratio increases with increase in \(T_q\) since fewer queries are generated. At later stages when the value of \(T_q\) further increases the local byte hit ratio for all the strategies tends to decrease. Figure 7 shows the effects of mean query generate time on the average query latency of GCCP, GCC with PBR and GCC schemes. From Figure 7 it is evident that with increasing mean query generate time the average query latency is decreasing in all the schemes.
algorithm which maximizes the performance improvement only prefetch the right data. In this paper we have proposed an performance in mobile ad hoc networks. However prefetching management in mobile computing environments. cache set. We hope that the proposed techniques will lead to a data item, but also the relationship of this item with the algorithm which not only considers the caching parameters of due to caching in mobile ad hoc networks. We devise an [1]
[3]
[4]
[7]
[5]
[6]
Fig. 6 The effect of mean query generate time on an average query latency

VI. CONCLUSION
Prefetching technique can be used to improve the system performance in mobile ad hoc networks. However prefetching also consumes a large amount of system resources such as computation power and energy. Thus it is very important to only prefetch the right data. In this paper we have proposed an algorithm which maximizes the performance improvement due to caching in mobile ad hoc networks. We devise an algorithm which not only considers the caching parameters of a data item, but also the relationship of this item with the cache set. We hope that the proposed techniques will lead to further research work in several areas related to cache management in mobile computing environments.

REFERENCES

Naveen Chauhan received M.E. degree from Punjab University, Chandigarh, India. He is currently working toward the Ph.D. degree in Computer Science and Engineering at National Institute of Technology, Hamirpur. His research interest includes wireless networks, distributed systems, and mobile computing, with a focus on mobile ad hoc networks.

Dr. Lalit Awasthi received his Ph.D degree from IIT Roorkee in Computer Science and Engineering. Previously he received MTech from IIT Delhi. Presently he is working as Dean (Students & Alumani Affairs) and Head Computer Centre, NIT Hamirpur since 1989 he has been with the Department of Computer Science and Engineering, NIT Hamirpur, where he is currently a Professor. His research interests are distributed fault tolerant computing, mobile computing, wireless sensor networks.