Chapter 5

A Coordinated Cache Placement Scheme for Cooperative Caching

Contents

5.1 Introduction .................. 92
5.2 Need for Coordination .......... 93
5.3 Related Works .................. 95
5.4 The Proposed Scheme .......... 99
5.5 Algorithm .................. 101
5.6 Simulation and Performance Evaluation .......... 105
  5.6.1 Performance Metrics .......... 107
5.7 Simulation Results .......... 107
5.8 Conclusion .................. 109
5.1 Introduction

In mobile ad hoc networks, network connectivity, energy constraints, low bandwidth and frequent disconnections pose new problems for efficient data access. The idea of sharing caches among neighboring nodes is an important technique to improve data access efficiency in mobile ad hoc networks. Cache buffer management mainly consists of two components, cache replacement and cache placement. The role of cache replacement was described in the previous chapter. Cache placement component determines where to place the incoming data in order to minimize the average access cost. The effectiveness of a given amount of cache space can be improved by efficient data placement schemes. This chapter focuses on cache coordination issue and discusses the need for coordinated storage decision. Coordinated storage helps for two reasons [Korupolu, 2002]. First, it allows a busy cache to utilize a nearby idle cache. Second, coordination improves the hit rate by having more unique data in the shared nodes. A new cache data placement scheme combined with cache replacement ideally suited for mobile host with limited memory is introduced in this chapter.

Although several cooperative caching protocols have been proposed in literature, only a few studies have examined the coordination of caches from data placement point of view. Many of the studies focus on number of hops for making data placement decision. The remaining works on cache placement mainly focus on selecting cache nodes based on the information about data access frequency and network topology. The main drawback of this approach is that it does not consider the cooperation of caches for data placement. As a result, data may be cached in several caches in the cache group which may lead to inefficient utilization
of available cache space. This may cause significant reduction in hit rate especially for ad hoc networks since the nodes have lower cache storage than other type of networks. This chapter introduces a coordinated data placement scheme that eliminates redundant data stored in multiple neighboring nodes by sharing and coordination of cache state among neighbouring nodes for efficient data storage and retrieval using the concept of cached object’s eviction period. The cache data placement scheme based on eviction period reduces replication of data by assuring that a copy of the data will be present in the cooperate cache, without any extra communication overhead. The simulation results show that the proposed scheme yields better performance in terms of cache hit ratio and average latency compared to independent cache data placement.

5.2 Need for Coordination

In traditional caching scheme, each data request is satisfied by the cache associated with the requesting node. The storage decision made by one cache is independent of those made by other caches in the system. Cooperative caching on the other hand, cooperates in serving one another’s request as well as in storage decisions. By sharing the cache contents, the clients get more chance to fetch data from geographically close nodes instead of remote data server. The placement scheme determines how to coordinate the contents of local cache. Without coordination, the requested object is fetched from the neighbouring node and the local node stores it. This scheme limits the advantage of cooperative caching due to indiscrete replication of objects. In the worst case all the neighbouring nodes can have the same contents and get nothing from cache contents share.
The following example illustrates the above mentioned issues. Consider the above network given in Fig. 5.1 with three nodes $N_1$, $N_2$ and $N_3$ with local cache $C_1$, $C_2$ and $C_3$, which coordinate with each other to share data. When node $N_1$ experiences a local miss, it sends cache requests to nodes $N_2$ and $N_3$. In case of non availability of data in either of the nodes, $N_1$ fetches the data from the server, stores a local copy and serves the client. If after some time, node $N_2$ needs the same data, the data request is satisfied from $N_1$ and $N_2$ stores it locally. A similar case occurs when $N_3$ needs the same data. This uncontrolled placement of data in the cache can result in several performance limitations. As the storage space is limited, each additional copy of data causes the eviction of other data items from the cooperative caches. Hence the number of unique documents available within the cache shrinks. Further the average amount of time the data items remain in the cache before they are evicted decreases. The cumulative effect of these two is the decrease in
the cumulative hit rate of the cooperative cache.

Therefore an independent object placement scheme is not optimal for cooperative caching. Another extreme is to prohibit replication of data items in the neighbouring nodes. This scheme can decrease cache miss ratio by making the number of distinct objects maximum in the group. But the access latency may increase due to a lot of remote fetching data from the neighbor nodes and not from the local cache. Between both extremes there were efforts to find out efficient placement scheme that achieves minimum access latency of objects.

The decision of whether to cache data in each node is made collectively among the neighboring nodes that cache the same data. The coordinated placement of data is based on the cache life time of individual nodes. We use the concept of cached object’s eviction period to measure the contention of caches. The proposed scheme tries to reduce duplicate data in the neighboring nodes by storing the data items only in the cache with lowest eviction rate.

5.3 Related Works

Cache placement schemes assigns data to caches without violating the cache size constraints. As discussed in chapter 3 several cooperative caching protocols have been proposed for MANETs to improve cache hit ratio and to reduce data access delay. The following section presents an overview of different proposals made for cache placement strategies related to the work presented in this chapter. Compared to the amount of work done on cache replacement policies there exists only a few proposals for cache placement. Most of the cache data placement schemes devised so far performs cache placement independently.
Korupolu [2002] defined the cost of cache placement as the sum over all nodes ‘u’ and all objects ‘ψ’ of access frequency for object ‘ψ’ at node ‘u’ times the distance from node ‘u’ to the closest copy of that object. The objective of a cooperative placement algorithm is to compute placement with minimum cost. By optimally placing the cache objects, the network load and server load is not explicitly minimized, but these would be low when the access cost is minimized. This is because the objects will be stored closer to the clients, thereby reducing the load on both the network and the server. In their proposed work, a cost function for document placement was formulated to measure the cost incurred by a cache to obtain the document from its nearest available copy. Access frequency of each object present in the cache is taken as the main parameter to calculate the cost function. The document placement schemes aim at minimizing the sum of the cost functions of all documents over all caches in the group. The arrangements of caches in the cache group are based on clusters and these clusters themselves are arranged in a tree structure. They also proposed a greedy placement algorithm which involves a bottom-up pass of the cluster tree structure to determine the cache where the document has to be stored.

A benefit-based greedy cache placement strategy has been presented in [Tang, 2008], where the cache placement is done based on the total data access cost of all the nodes in the network. The benefit score for a data item $D_j$ in node $i$ is the product of access frequency of $(t_{ij})$ of data item $D_j$ in node $i$ and least distance $(\delta_j)$ to the neighboring node containing the data item.

\[
\text{Benefit Score } B_{ij} = t_{ij} * \delta_j.
\]
When a node caches a data item it broadcasts information about the availability of the corresponding data item to the broker. Similarly, when a data item is deleted it broadcasts the non-availability of the corresponding data item to the broker. The broker periodically broadcasts the metadata of the cache updates to the whole network. When the cache is full, data item with lowest benefit score is replaced. The primary objective of this approach is to cache data at available memory location of all nodes available in the network so that all the available memory are utilized over a period of time.

Du [2005] proposed two schemes for cache placement. In order to cache more data in the cooperative cache the cached data copies are categorized into two groups based on whether they are already present in the neighbouring nodes or not. If the incoming data item is not from any one of the nodes present in the cooperation zone it is taken as a primary copy. The data taken from the neighbouring nodes is taken as secondary. The priority of primary and secondary data is made by the inter- and inter-category rules. The intercategory rule put primary items at a higher priority level and the secondary items are purged to accommodate primary items, but not vice versa. In order to determine whether the data is primary or secondary, a labeling mechanism is used.

When a node receives a data item, it labels the item as primary if the item comes from a node beyond the zone radius and if the data item comes from within the zone radius, then check for the label, to determine whether the data item is primary or secondary. If the data item is already labeled as primary, the new copy would be secondary since there will not be duplicated primary copies in the same cooperation zone. On the other hand, if the data item is tagged as secondary, the provider needs to attach
the information of the primary copy holder. If the primary copy holder is beyond the zone radius, the new copy is primary copy; otherwise, the new copy is a secondary copy. The intra-category rule is used to evaluate the data items within the same category. The LRU algorithm is used to remove the data items present in the same category.

Fan [2013] presented a Gossip based Cooperative Caching (GosCC), which considers the sequential relation among the data items. In this scheme, each mobile node stores the IDs of the data items cached locally and the ID of the data item in use in its progress report. Each mobile node makes use of this progress report to determine whether a data item should be cached locally. The progress reports are propagated within the network in a gossip based way. The progress report contains the progress of each node consuming data items and the contents in cache at each node. Cache digest describes its own cache content to other nodes such that other mobile nodes can send their data request to the closest cache node for data item. In this scheme, the mobile node investigate the progress digest report and send the cache request for the most popular data item in future. The disadvantage of this scheme is the message overhead due to the periodical transfer of cache digests and progress reports. Another disadvantage is that this work mainly focuses on applications with correlated data.

Yin [2006] proposed a hybrid cache scheme in which the incoming data is cached according to certain criteria. The selected criteria include data item size, TTL parameter and number of hops. Based on the value of these parameters, data itself or path to the nearest cache is stored in the cache. Hara [2006] proposed a cache placement scheme based on the access frequency for a particular data item. A mobile host selects data
items to cache based on the access frequency. Nuggehalli [2003] considered the trade-off between query delay and overall energy consumption for cache placement.

From the survey it is found that most of the work related to cache placement involves independent cache placement. Function based cache placement policies require some parameters to calculate the cost function. The disadvantage of this approach is that the relative importance of these parameters can vary from one type of request to another and these policies need to adjust the weights dynamically to achieve the best performance. Function based policies are also limited since they do not consider the redundancy of data. Another category for cache placement scheme is cache placement based on number of hops. The drawback of this approach is, it considers only the number of hops for cache object placement. But the data item present in the neighboring nodes may sometimes get evicted soon from the cache. In order to resolve these issues, a new cache data placement policy based on the eviction period of the data item stored in the cache is introduced.

5.4 The Proposed Scheme

This section presents an overview of cache data placement scheme proposed for cooperative caching in ad hoc networks. In this placement algorithm, each node makes informed and intelligent decisions on whether to cache a particular data. The decision to cache a particular data item is based on two factors: whether the data is already available in the neighboring nodes and if available how long it is likely to remain in the caches where they are currently stored. In order to find the likely time a data item will remain in the cache, the concept of Cache Eviction Period
(CEP) is used [Psounis, 2002].

The eviction period of individual cache can be calculated from the last access time of the data item and the time at which it is removed from the cache. The last access time of each data item can be taken from the replacement policy. The additional parameter required to calculate the eviction period is the time at which the data item is removed from the cache since its last access. The eviction period of a data item is defined as the time duration between the time it was evicted from the cache and the time it was last accessed. This can be denoted as $T_e - T_a$, where $T_e$ is time at which the data item is evicted from the cache and $T_a$ is the time of last access. This period implies how long a data item is placed in the cache after its last access. The average eviction period of a cache in a finite time period is given by the average of the eviction period of the data items that were removed from this cache in this period.

For a given finite time duration $T_i - T_j$, let $S(C, T_i, T_j)$ denotes the set of data items evicted from the cache.

$$S(C, T_i, T_j) = \{D | \forall T < T_i, P(D, C, T) \land \forall T' > T_j, R(D, C, T')\}$$

where $P(D, C, T)$ is the set of data items that remains in the cache $C$ at time $T$ and the set $R(D, C, T')$ denotes the data items already evicted from the cache $C$ at time $T'$.

The cardinality of the set $S(C, T_i, T_j)$ indicates the total number of documents evicted from a cache $C$ during the duration $(T_i, T_j)$. The Cache Eviction Period (CEP) of the cache $C$ for the duration $(T_i, T_j)$ is
Chapter 5. A Coordinated Cache Placement Scheme for Cooperative Caching

\[
\text{CEP} (C, T_i, T_j) = \frac{\sum_{D \in S(C,T_i,T_j)} \text{Eviction perod} (D, C)}{|S(C,T_i,Y_j)|}
\]

The CEP period denotes the average time a data item is expected to live in the cache after the last hit. If the CEP is higher, the data item is expected to stay longer in the cache.

The cache data placement scheme is combined with cache replacement policy E-LRU proposed in the previous chapter. In E-LRU, the time interval between last two references of each cached data item is stored. Using this information the IRT of data items is 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 or the longest time is dropped first. In both cases the time of last hit for each data item is maintained. In such a situation it is straightforward for any node to support the proposed cache placement policy.

5.5 Algorithm

The design rationale of the proposed cache placement algorithm is to optimize the cache usage, for applications with limited memory. The proposed algorithm tries to reduce duplicated data across neighboring nodes. The core idea of this approach is to use the eviction period of a data item as an indicator of the cache space contention at individual
nodes. In the following section we explain the proposed cache data placement algorithm and how each node makes decision on whether to cache the data obtained from another node in the neighbor group.

Whenever there is a local miss, the query node will send a cache control message to the neighboring nodes. The cache control message contains the requested data id and the CEP. If any one of the neighboring node has the requested data, it is sent back to the query node along with its CEP. Upon receiving the data, the query node checks its CEP with the CEP of the responder node. If the CEP of the query node is greater than the CEP of the responder node, it will store a copy of the data item in its local cache. If the CEP of the query node is less than the CEP of the responder, the query node will not store a copy of the data item. This is because the copy of the data item at the responder node is likely to remain in the cache for a longer period. Therefore, under the proposed scheme the query node will not store a copy of the requested data in its local cache, when the CEP is less than the CEP of the responder. At the responder node, if the CEP of the query node is higher, the last access time is not modified, so that the referenced data item will be evicted soon. When the query node receives a negative reply from the neighboring nodes, data is fetched from the server.

The proposed cache placement algorithm involves the following message types.
REQ \((i, d_j)\): the data request message from node \(i\) to the neighbouring nodes for data item \(d_j\).
REPLY \((j, i)\): reply message from node \(j\) to node \(i\) to inform that it has the data item \(d_j\).
ACK \((i, j, CEP_i)\): acknowledgement from node \(i\) to node \(j\) along with
Chapter 5. A Coordinated Cache Placement Scheme for Cooperative Caching

CEP of node ‘i’.

REPDAT \((j, i, d_j, \text{CEP}_j)\): reply message from node ‘j’ to node ‘i’, along with CEP of node ‘j’.

**Algorithm 1 Cache Placement**

```plaintext
for every data request in local cache LC of node i do
  // search for data in the local cache
  if data \(d_i\) is present in LC, then
    Get \(d_i\)
  else
    // Search for data in the neighboring nodes
    Send msg \(\text{REQ} (i, d_j)\) to all neighbours
  end if
end for

if a positive \(\text{REPLY} (j, i)\) from \(N_j\) arrives then
  // Find the CEP
  Send msg \(\text{ACK} (i, j, \text{CEP}_i)\) to node \(N_j\)
  On receiving \(\text{ACK} (i, j, \text{CEP}_i)\) from \(N_j\)
  Send msg \(\text{REPDAT} (j, i, d_j, \text{CEP}_j)\)
  if \((\text{CEP}_i \geq \text{CEP}_j)\) then
    Cache \(d_i\);
  else
    Serve the data request
  end if
else
  // Request is redirected to server
  Send msg \(\text{REQ} (i, d_j)\) to server
  Get \(d_i\);
  Cache \(d_i\);
  Serve the request;
end if
```

A node \(N_i\) that experiences a local miss sends out a \(\text{REQ} (i, d_j)\) to all its neighbouring nodes. If any one of the neighbouring node \(N_j\) has the requested data it sends a \(\text{REPLY}(j, i)\) message to node \(N_i\). Node \(N_i\) on receiving \(\text{REPLY} (j, i)\) message from node \(N_j\), sends an \(\text{ACK} (i, j, \text{CEP}_i)\) back to node \(N_j\). Upon receiving \(\text{ACK} (i, j, \text{CEP}_i)\) Node \(N_j\) sends the
Algorithm 2 Cache Placement at the responding node

// At the responding node \( N_j \)
Upon receiving a msg \( \text{REQ} (i, d_j) \)
begin
Check for the availability of data
if \( d_i \) is available then
Send \( \text{REPLY} \) \((j, i)\) to \( N_i \)
On receiving msg \( \text{ACK} (i, j, \text{CEP}_i) \) from node \( N_i \)
// Check for CEP
if \((\text{CEP}_i > = \text{CEP}_j)\) then
Send msg \( \text{REPDAT} \) \((j, i, d_j, \text{CEP}_j)\)
elser\( \text{IRT} \) \( d_i < -\text{IRT}_{\text{new}} \)
Update \( \text{LC}_j \)
Send msg \( \text{REPDAT} \) \((j, i, d_j, \text{CEP}_j)\)
end if
elesend msg \( \text{REQ} (i, d_j) \) to server
deendif

REPDAT \((j, i, d_j, \text{CEP}_j)\) message back to node \( N_i \). Node \( N_i \) compares its own \( \text{CEP}_i \) with \( \text{CEP}_j \). If \( \text{CEP}_i \) is greater than \( \text{CEP}_j \), \( N_i \) does not store the data locally; it just serves the user with the data. However, if \( \text{CEP}_j \) is greater than \( \text{CEP}_i \), the node \( N_i \) stores a copy of the data locally. On the other hand, node \( N_j \) also compares its own \( \text{CEP} \) with the \( \text{CEP} \) of node \( N_i \). If \( \text{CEP}_j \) is greater than \( \text{CEP}_i \), \( \text{IRT} \) is changed and the data position is updated. Otherwise, \( \text{IRT} \) is not changed and the data position remains un-altered. By this, the placement scheme ensures that the data is cached only if the new copy has a reasonable chance to survive longer than the original copy, which obviously reduces redundant copies in the cooperative cache. The complete sequence of messages transmitted during cache placement is shown in Fig. 5.2.
5.6 Simulation and Performance Evaluation

The performance of the proposed coordinated cache data placement scheme using CEP is compared with independent cache data placement scheme based on number of hops. Following sections explain the simulation model and the performance metrics used for performance evaluation.

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 mobile ad hoc network environment consists of a number of mobile nodes and a fixed data server. Each mobile node
is identified by a node id and a host name. 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. It is assumed that each node knows its current location precisely with the availability of Global Positioning System (GPS). For the simulation model it is assumed that the position of each node is given by the $x$ and $y$ coordinates. The data server contains all the data items requested by the mobile nodes. The database in the data server contains the data items, with each item identified using a data id. The nodes that generate data request are selected randomly and uniformly. 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 recent two data access and the time at which data item is evicted from the cache. This table is updated whenever an event occurs in the cache. The cache space for each node is limited and when it is full, the replacement strategy evicts the unwanted data. 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. When a node receives 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.

The mobile client generates read only queries and the time interval between two consecutive follows an exponential distribution. The queries generated follow a Zipf distribution which is frequently used to model non uniform distribution. The data request is processed in FCFS manner at
the server. An infinite queue is used to buffer the request when the data server is busy. Each miss in the cooperative cache will incur a delay of 4 ms to retrieve data from the data server. 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 $U(V_{\text{min}}, V_{\text{max}})$ and travels with that constant speed $s$. When the node reaches destination, it pauses for 900 seconds. After that it moves to the new destination with speed $s'$. The simulator was implemented in JAVA.

### 5.6.1 Performance Metrics

The following performance metrics are used to evaluate the proposed cache data placement scheme: cache hit ratio and the average latency. The evaluations of these parameters are done by varying the cache size with fixed number of nodes. Cache hit ratio is defined as the percentage of requests that can be served from previously cached data. If a data item is requested by a mobile node at time $T_R$ and the data is served at time $T_C$, latency for data item is defined as $(T_C - T_R)$. Average latency is the average of the latencies experienced for all the data items served.

### 5.7 Simulation Results

To evaluate the performance of the proposed cache data placement scheme, CEP it is compared with an independent hop based scheme (IB scheme) in which the data items from the neighboring nodes are not cached. This scheme is chosen for comparison because most of the existing cooperative caching techniques use this criterion for cache data
placement. The cache hit ratio and average latency for different cache sizes is measured. Fig. 5.3 shows the comparison of cache hit ratio for different cache sizes. From the figure we can see that the difference between cache hit ratio is higher when the cache size is small. This is due to the fact that when the size of the cache memory is small, effective memory utilization can make noticeable improvements in cache hit rates. Fig. 5.4 depicts the comparison average latency of proposed cache data placement for different cache sizes. Since the cache miss rate is relatively low at small cache size the average latency becomes insignificant at large cache sizes.

![Figure 5.3: Cache Hit Ratio](image)

Figure 5.3: Cache Hit Ratio
Chapter 5. A Coordinated Cache Placement Scheme for Cooperative Caching

5.8 Conclusion

The performance of cooperative caching can be improved by proper cache management. This chapter discusses the role of coordinated cache placement for cooperative caching in ad hoc networks. A coordinated cache data placement algorithm based on CEP is presented. The CEP is taken as an indicator of the cache space contention at individual nodes. The decision on caching an incoming data is done coordinately among the neighboring nodes that already have a copy of the data item. The proposed scheme effectively reduces the number of redundant data. This scheme has been proposed for effective memory utilization for mobile clients with limited memory. Simulation results show that the proposed policy can significantly improve the performance compared to independent cache placement schemes especially for applications with limited cache.
5.8. Conclusion