Journal of the Korean Data & Information Science Society
2010, **21**(3), 605–619

Design and evaluation of a fuzzy cooperative caching scheme for MANETs[†]

Ihn Han Bae[1]

[1]School of Computer and Information Engineering, Catholic University of Daegu

Received 13 April 2010, revised 18 May 2010, accepted 22 May 2010

Abstract

Caching of frequently accessed data in multi-hop ad hoc environment is a technique that can improve data access performance and availability. Cooperative caching, which allows sharing and coordination of cached data among several clients, can further enhance the potential of caching techniques. In this paper, we propose a fuzzy cooperative caching scheme in mobile ad hoc networks. The cache management of the proposed caching scheme not only uses adaptively CacheData or CachePath based on data similarity and data utility, but also uses the replacement manager based on data profit. Also, the proposed caching scheme uses a prefetch manager. When the TTL of the cached data expires, the prefetch manager evaluates the popularity index of the data. If the popularity index is larger than a threshold, the data is prefetched. Otherwise, its space is released. The performance of the proposed scheme is evaluated analytically and is compared to that of other cooperative caching schemes.

Keywords: Cooperative caching, data similarity, data utility, mobile ad hoc networks, popularity index.

1. Introduction

Mobile ad hoc networks (MANETs) consists of mobile hosts (MHs) such as notebooks, PDAs, cell phones and so on. These mobile devices can create a wireless network dynamically without the aid of any network infrastructure. Every MH can move arbitrarily and communicate with other MHs by using multi-hop wireless links. Each MH acts as a router and forwards data packets to other neighbors in the coverage of its transmission range. This type of network can communicate with external networks such as Internet through a gateway. However, MANETs are limited by intermittent network connections, restricted power supply and limited computing resources. These restrictions raise several new challenges for data access applications with respects to data availability and access efficiency. One of the most attractive techniques to improve data availability is caching. In general, caching results in (i) enhanced QoS at the clients, i.e., lower jitter, latency and packet loss, (ii) reduced

[†] This research was supported by the Research Grants of Catholic University of Daegu in 2009.
[1] Professor, School of Computer and Information Communication Engineering, Catholic University of Daegu, Hayang-eup, Gyeongsan-si, Gyeongbuk-do 712-702, Korea. Email: ihbae@cu.ac.kr

network bandwidth consumption, and (iii) reduced data server/source workload. In addition, the reduction in bandwidth consumption can potentially improve battery life of mobile hosts (Ting and Chang, 2007; Chand *et al.*, 2007).

Caching has been proved to be an important technique for improving the data retrieval performance in mobile environments. With caching, data access delay is reduced since data access requests can be served from the local cache, thereby obviating the need for data transmission over wireless links. However, caching techniques used in one-hop mobile environment (i.e., cellular networks) may not be applicable to multi-hop mobile environments since the data request may need to go through multiple hops. As the mobile clients in ad hoc networks may perform similar tasks and share a common interest, cooperative caching, which allows sharing and coordination of cached data among multiple clients, can be used to reduce bandwidth and power consumption (Chand *et al.*, 2007).

In this paper, we propose a fuzzy cooperative caching scheme (FCCS) in mobile ad hoc networks to enhance the performance of the COOP scheme proposed by Du and Gupta (2005). The cache management of the proposed caching scheme not only uses adaptively CacheData or CachePath based on a fuzzy logic according to data similarity and data utility, but also uses the replacement manager based on data profit. Also, the proposed caching scheme uses a prefetch manager. When the TTL (Time to Live) of the cached data expires, the prefetch manager evaluates the popularity index (PI) of the data. If the PI is larger than a threshold, the data is prefetched. Otherwise, its space is released. The performance of the proposed method is evaluated by an analytical evaluation and is compared to that of other cooperative caching methods: HybridCache and COOP.

The remainder of this paper is organized as follows. Section 2 reviews the related works. Section 3 describes the proposed FCCS scheme. Section 4 presents a model for analytical evaluation of FCCS. Section 5 concludes the paper and discusses future works.

2. Related works

A cooperative caching scheme specifies how multiple computing nodes can share and manage cached data in a collaborative manner. This paper focuses on caching of regular objects such as text files and web documents. 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 server after local cache misses. This scheme, referred to as SimpleCache (Yin and Cao, 2005), works well as long as the connection to the server is reliable and not too expensive; otherwise, it results in failed data requests or request timeouts.

To increase data availability and to reduce the cost in terms of data access latency and energy consumption, hop-by-hop cache resolution allows a node on the forwarding path to serve as a proxy for resolving the request. If a forwarding node contains an un-expired copy of the requested data, it can send a reply to the requester and stop forwarding the data request (Du *et al.*, 2009).

A nodes cooperation zone consists of the surrounding nodes within a r-hop range. The parameter r is called the radius of the cooperation zone. If a node does not have information about whether the requested data item is available in the zone, the node can reactively discover this by flooding the request within the zone. To restrict the flooding range, the TTL value of the request is set to the zone radius r. The restricted flooding not only has the potential to discover the closest data location but also can serve as an announcement

in the neighborhood and effectively segment the whole network into clusters, within which caching information is shared (Du *et al.*, 2009).

Yin and Cao (2005) presented three cache resolution schemes: CacheData, CachePath, and HybridCache. In CacheData, forwarding nodes check the passing-by data requests. If a data item is found to be frequently requested, forwarding nodes cache the data, so that the next request for the same data can be answered by forwarding nodes instead of travelling further to the data server. A problem with this approach is that the data could take a lot of caching space in forwarding nodes. CachePath is similar with CacheData, except that forwarding nodes cache the path to the closest caching node instead of the data and redirect future requests along the cached path. This scheme saves caching spaces compared to CacheData, but since the caching node is dynamic, the recorded path could become obsolete and this scheme could introduce extra processing overhead. Trying to avoid the weak points of those two schemes, HybridCache decides when to use which scheme based on the properties (size and the TTL value) of the passing-by data. The HybridCache technique represents a combination of CacheData and CachePath. This technique performs better than either the CachePath or CacheData approach. The cache replacement algorithm in HybridCache is based upon the access frequency of a data item and the distance to the same cached copy or to the data source. However, due to the inherent mobility of the host, such distances can change frequently.

Du and Gupta (2005) proposed COOP, a novel cooperative caching service for data access applications in MANETs. The objective is to improve data availability and access efficiency by collaborating local resources of mobile devices. COOP addresses two basic problems of cooperative caching: cache resolution and cache management, i.e. how to find the requested data efficiently and how to manage local cache to improve the capacity of cooperated caches.

Bae (2009) proposed a fuzzy proxy cache scheme for distributed base stations. The proposed scheme predicts the mobility of mobile hosts and uses various cache methods for neighboring proxy servers inspired by fuzzy-logic-based control rules based on the membership function of the mobile host.

3. Fuzzy cooperative caching scheme

In this paper, we present FCCS, a fuzzy cooperative caching scheme for MANETs which aims to improve data availability and access efficiency. FCCS addresses two basic problems for cooperative caching in MANETs.

- Cache resolution - how does a mobile host decide where to fetch a data item requested by the user or the application.
- Cache management - how does a mobile device decide which data item to place/purge in its local cache?

For cache resolution, FCCS tries to discover a data source, which induces less communication cost by utilizing historical profiles and forwarding nodes. For cache management, FCCS minimizes caching duplication between neighbor nodes and uses fuzzy cooperative caches based on data item utility and access similarity to improve the overall performance.

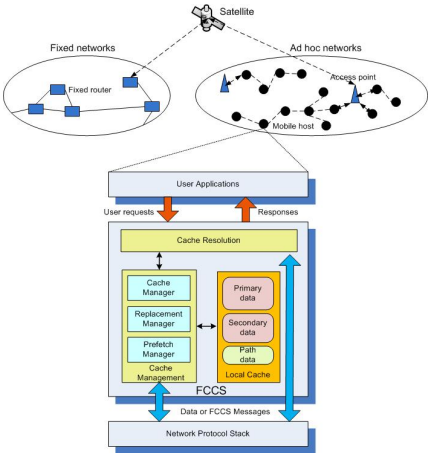

Figure 3.1 FCCS system model.

In this paper, we consider a MANET environment where some of the mobile nodes have interfaces to other networks such as satellite networks (Figure 3.1). Those nodes serve as gateways to allow other nodes to communicate with the outside networks. A data source can reside inside or outside the MANET. The mobile nodes can read data from the data sources. But only the sources have the privilege to write or update the data they host. The mobile nodes can cache a copy of the data in its local storage. We assume the TTL scheme is used to maintain consistency between the client-side cache and the data source. The basic idea of TTL is that each data item is assigned a TTL value, and the data item is considered valid as long as the caching time has not exceeded the TTL value.

The system architecture of FCCS resides at middleware level, acts as a proxy for users applications, and uses the underlying network stack to communicate with FCCS instances running on other nodes. A running FCCS instance receives data requests from users applications, and resolves the requests using the cache resolution. The cache management scheme decides what data to place/purge in the local cache of a mobile device. It discriminates primary data (non-duplicated copy) and secondary data (duplicated copy) with primary data at a higher caching priority, so that cooperating caches can store more distinct data items to improve the overall performance.

3.1. Mobile cache structure

Designing a cache model has to be based on the characteristics of the cached data and the specific caching environment. In mobile caching, the cached data may be migrated to the mobile host from other hosts instead of being directly fetched from the original data servers. To cope with the characteristics of cached data and mobile environments, a mobile cache structure featured in Figure 3.2 is designed. The mobile cache structure consists of two parts. The cache structure body is the cached data item.

The header includes ID, Name, Link, TTL, Dist, Size, Origin and Tags. The ID field contains a UUID (universal unique ID) which the URL of the cached document is hashed.

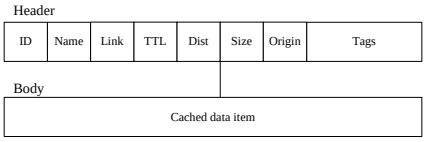

Figure 3.2 Mobile cache structure.

Name is the name of the cached data. Because the cached data might be migrated from other mobile hosts, Link is needed to provide links to mobile hosts that previously owned or searched for this data. The data server assigns a TTL value to all data items. Mobile hosts consider a cached copy up-to-date if its TTL has not expired. Due to TTL expiration, some cached data may be invalidated. Dist is the number of hops to reach the node which forwards the data. Size is the size of the cached data. Origin indicates whether this current data replica is fetched from the original data server or is migrated from another mobile host. Tagging, or labeling content, is part of the collaborative nature of Web 2.0. A tag is any user-generated word or phrase that helps organize web content and label it in a more human way. This kind of meta data helps describe an item and allows it to be found again by browsing or searching (Deitel and Deitel, 2008; Golder and Huberman, 2006). Tags are chosen informally and personally by the item's creator. We assume that each data object has 5 8 tags.

3.2. Cache management

The cache management of FCCS studies how to decide which data item to keep in a nodes local cache. The goal is to increase cache hit ratios, which largely depend on the capacity of the cache. The cache management consists of three components: cache manager, replacement manager and prefetch manager.

3.2.1. The cache manager

The cache manager uses adaptively the CacheData or CachePath based on fuzzy logic according to the similarity and the utility of a data item. In the CachePath of the cache manager, a node does not need to record the path information of all passing data. Thus, a router node only records the data path when it is closer to the caching node than the data source. Due to mobility, the node which caches the data may move. The cached data may be replaced due to cache size limitation. As a result, the node which modified the route should reroute the request to the original data source after it finds out the problem. Thus, the cached path may not be reliable and using it may adversely impact the overhead. To deal with this issue, a node caches the data path only when the caching node is very close. In the CacheData approach, the router node caches the data instead of the path when it finds that the data is frequently accessed. Since the CacheData approach needs extra space to save the data, it should be used prudently.

To maximize the capacity of cooperative caches, FCCS tries to reduce duplicated caching within a short-distance neighborhood, such that the cache space can be used to accommodate more distinct data items. The CacheData approach categorizes cached data copies based

on whether they are already available in the neighborhood or not. A data copy is primary if it is not available within the neighborhood. Otherwise, the data copy is secondary. The range of neighborhood is provided as a customizable option. The reason for discriminating primary and secondary data is that the cache miss cost is proportional to the travel distance of a data request, and primary data usually incur higher cache miss costs than secondary data. The inter-category and intra-category rules are used to decide caching priorities of primary and second data, which are described next.

The idea of inter-category cache is to put primary items at a priority level, i.e., secondary items are purged to accommodate primary items, but not vice versa. Once a node fetches a data item, it labels the item as primary copy if the item comes from a node beyond the neighborhood range. Otherwise, if a data item comes from within the neighborhood, we need to further consider whether the data provider labels the item as primary or secondary. If the provider already labels its copy as primary, the new copy would be secondary since we do not intend to have duplicated primary copies for the neighborhood. On the other hand, if the provider tags its own copy as secondary, the provider needs to attach the information of the primary copy holder. If the primary copy holder is beyond the neighborhood range, the new copy is primary copy. Otherwise, the new copy is a secondary copy.We assume that each mobile host has the tags for preference in the user profile and the user profile also has 5 8 preference tags. The similarity between two mobile hosts is simply the number of common tags between the tags of a mobile host and the tags from caching data. Eq. (3.1) shows the formula we use for computing the ratio of tag sets between the tags of a mobile host and the tags from caching data.

$$Sim_i(A, B) = \frac{2 \times |A \cap B|}{|A| + |B|}, \tag{3.1}$$

where *A and B* represent the sets of the tags of a mobile host and the tags of a data item d_i, respectively. We map the data similarity to the three basic fuzzy sets: H (high), M (medium) and L (low) using the membership function as shown in Figure 3.3. The membership function is a function in [0, 1] that represents the degree of belonging. The membership function fully defines the fuzzy set. Membership functions on *Sim* represent fuzzy subsets of *Sim*. The membership function which represents a fuzzy set DS (Data Similarity) is usually denoted by $\mu_{DS}(Sim_i)$.

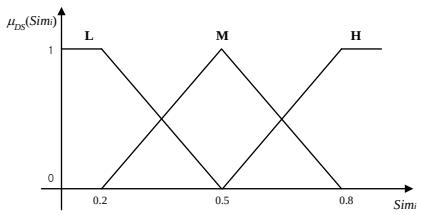

Figure 3.3 Membership degree for the data similarity.

Observe that smaller the size of a data item, the lesser the cache space for the data item needs. The greater the *Dist* value of a data item has, the more the utility for the cached data has. But the larger the *Size* of the data has, the lesser the utility for the data caching

has. Accordingly, the utility for cached data item d_i, $Util_i$ is computed using the expression:

$$Util_i = \frac{Dist_i}{Size_i}. \tag{3.2}$$

We map the data utility to the three basic fuzzy sets: H (high), M (medium) and L (low) using the membership function as shown in Figure 3.4, where the unit of data size is kilobyte (KB). Membership functions on *Util* represent fuzzy subsets of *Util*. The membership function which represents a fuzzy set DU (Data Utility) is usually denoted by $\mu_{DU}(Util_i)$.

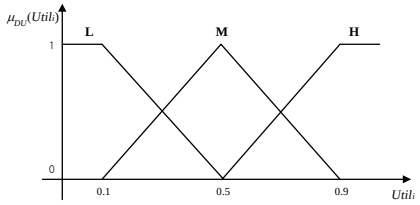

Figure 3.4 Membership degree for the data utility.

To maximize the capacity of cooperative caches, FCCS tries to reduce duplicate caching within short-distance neighborhood, such that the cache space can used to accommodate more distinct data items. The inter-category and intra-category rules are used to decide caching priorities of primary and secondary data. The control rules for FCCS which consider data similarity and data utility are shown in Table 3.1. For a mobile host, if the data similarity is high (H) and the data utility is high (H), CacheData (i) is used for managing the data item. If the data similarity is medium (M) and the data utility is high, CacheData (j) is used. And, if the data similarity is medium (M) and the data utility is medium (M), CacheData (k) is used. Where i, j and k are the input parameters which represent neighborhood range and i<j<k. Also, if the data similarity is low (L) and the data utility is low (L), CachePath is used for managing the data item.

Table 3.1 The control rules for FCCS.

		DU		
		H	M	L
DS	H	CacheData (i)	CacheData (j)	CacheData (k)
	M	CacheData (j)	CacheData (k)	CachePath
	L	CacheData (k)	CachePath	CachePath
(input variables)		DU: H (high), M (medium), L (low) DS: H (high), M (medium), L (low)		
(output variables)		Cache methods: CacheData (i), CacheData (j), CacheData (k), CachePath		

The intra-category rule is used to evaluate the data items within the same category. For this purpose, the cache manager calls the replacement manager. In FCCS, the replacement manager of cache management uses the least profit value (LPV) policy.

3.2.2. The prefetch manager

When the TTL of the cached data expires, the prefetch manager is called. The prefect manager evaluates the popularity index (PI) of the expired data. The PI of a data reflects the access probability of the data item for a mobile host. The popularity index for expired data item d_i, PI_i is

$$PI_i = \frac{A_i}{\sum_{j=1}^{n} A_j}, \tag{3.3}$$

where A_i is the mean access rate to data item d_i and n is total number of data items in the local cache. A_i can be estimated by employing sliding window method of last k access times. FCCS keeps a sliding window of k most recent access timestamps $(t_i^1, t_i^2, ..., t_i^k)$ for data item d_i in the cache (Shim *et al.*, 1999). The access rate is updated using the formula:

$$A_i = \frac{k}{t - t_i^k}, \tag{3.4}$$

where t is the current time and t_i^k is the time of k-th most recent access. If less than k samples are available, all available samples are used. If the computed PI_i is greater than a threshold (δ_{PI}), the data is prefetched. Otherwise, its space is released.

3.2.3. The replacement manager

Because of limited cache size, a replacement policy must be adopted to evict data from the cache when new data arrive. The replacement manager uses the replacement policy based on the least data profit. At a mobile host, the replacement manager evaluates the data profit for all cached data items in the mobile host. The larger the mean access rate, the distance and current TTL of a cached data have, the larger the data profit value of the cached data item has. Also, the larger the size of a cached data item has, the smaller the data profit value of the cached data item has. Accordingly, the profit of a cached data item d_i, Profiti is computed by

$$Profit_i = \frac{A_i \times Dist_i \times CTTL_i}{Size_i}. \tag{3.5}$$

The current TTL of a cached data item d_i, $CTTL_i$ is computed by the following expression:

$$CTTL_i = TTL_i - (T - T_i^{initial}), \tag{3.6}$$

where $T_i^{initial}$ is the time when d_i was dispatched from the data server.

The replacement manager deletes the cached data item with the LPV from all cached data items when the free space is insufficient for accommodating a new data item to be cached.

3.3. Cache resolution

The proposed FCCS uses a composite approach for cache resolution as illustrated in Fig. 3.5. The FCCS forwards the data request to a destination mobile host along the data path after the local cache misses. If the requested data is in the destination mobile host, the data

is returned. If no cache is found using previous schemes, the data request is forwarded to the original data source. Roadside Resolution allows a node on the forwarding path to serve as a proxy to resolve the request, and roadside resolution is used to resolve the request along the forwarding path. That is, if the data is in the local cache of a forwarding node or the local cache of the node, which is in the local path of a forwarding node, the forwarding node can respond to the request then stop forwarding the data request.

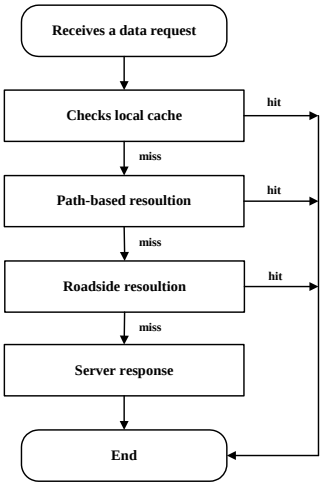

Figure 3.5 Resolution process of FCCS.

4. Performance evaluation

The following assumptions are made for the analytical evaluation for fuzzy proxy caching scheme (Feeney and Nilsson, 2001):

- The response delay and energy cost is 0 if the local cache contains the requested data. This is a reasonable assumption because the local cache access cost can be ignored when compared to inter-node communication.
- If a data request cannot be satisfied from the local cache, we assume that the response delay is proportional to the travel distance in hops of the data request, that is, $Response\_delay = \alpha \times Travel\_distance$ and α is a constant. In this analysis, we normalize α to 1 since this does not change the analysis and comparison results.

If a data request cannot be satisfied from the local cache, we assume that the energy cost is proportional to the number of messages triggered by the data request, that is, $Energy\_cost = \beta \times Number\_of\_messages$. Similarity, we normalize β to 1 since this does not change the results.

P_d is the probability that a mobile host caches a copy of data d. It can be estimated by the percentage of caching mobile hosts in the network from history profiles. P_d can be different for different data items. The average density ρ is used to estimate the total number

Table 4.1 Notations for an analytical evaluation.

Notation	Description
P_d^{Hcd}	In HybridCache, the probability that the requested data is in the local cache.
P_d^{Hp}	In HybridCache, the probability that the requested data is in the local path.
P_d^{Co}	In COOP, the probability that the requested data is in the local cache.
P_d^{Cf}	In COOP, the probability that a forwarding node resolves the data request.
P_d^{Fc}	In FCCS, the probability that requested data is in the local cache.
P_d^{Fp}	In FCCS, the probability that the requested data is in the local path.
P_d^z	The probability that a node in the zone of the requesting node has a copy of data d.
P_d^p	The probability that a requesting data is popular.
P_d^x	The probability that a data along the local path is not usable. This may be caused by TTL expiration or broken paths because of node movement.
P_{xpd}^r	The probability that an expired popular data is requested.
P_{pd}^h	The probability that the prefetched data are requested.
λ	Average arrival rate of data requests per minute.
r	The radius of the cooperation zone of the requesting node.
L_s	The distance in hops between the requesting node and the original data server.
L_p^{Hc}	The average length of the path for a request to reach the node which has. a valid copy of the data in HybridCache.
L_p^{Fc}	The average length of the path for a request to reach the node which has. a valid copy of the data in FCCS.
ρ	The average node density.

of mobile hosts in a two-dimensional space. Accordingly, the number of mobile hosts within r-hop range of the requesting mobile host is estimated by $\rho\pi r^2$.

4.1. HybridCache

For the Hybrid caching, if a copy of the requested data is in the local cache of a requesting node or the local cache of the node, which is in the local path of the requesting node, the data is returned. If the requesting node misses the requested data, the data request is forwarded to the original data source. To calculate the average delay, three cases need to be considered:

1. The requested data is in the local cache.
2. A path is found in the local cache which indicates N_i caches the requested data. Two subcases are possible:
   a. A valid data item is found in N_i.
   b. The data item in N_i is not usable because of a broken path or TTL expiration.
3. No data or path is found in the local cache.

The probabilities of cases 1, 2(a), 2(b), and 3 are P_d^{Hc}, $(1 - P_d^{Hc})P_d^{Hp}(1 - P_d^x)$, $(1 - P_d^{Hc})P_d^{Hp}P_d^x$ and $(1-P_d^{Hc})(1-P_d^{Hp})$, respectively. Therefore, the average response delay is

$$D_{HybridCache} = 0 \times P_d^{Hc} + L_p^{Hc}(1 - P_d^{Hc})P_d^{Hp}\left[(1 - P_d^x) + L_s P_d^x\right] + L_s(1 - P_d^{Hc})(1 - P_d^{Hp}) \tag{4.1}$$

$$= L_p^{Hc}(1 - P_d^{Hc})P_d^{Hp}\left[(1 - P_d^x) + L_s P_d^x\right] + L_s(1 - P_d^{Hc})(1 - P_d^{Hp}).$$

The average energy cost is

$$E_{HybridCache} = L_p^{Hc}(1 - P_d^{Hc})P_d^{Hp}\,[(1 - P_d^x) + L_s P_d^x] + L_s(1 - P_d^{Hc})(1 - P_d^{Hp}). \quad (4.2)$$

4.2. COOP

The cache resolution of COOP uses the cocktail approach that combines hop-by-hop resolution and cooperation zone resolution by first trying to resolve a data request in the zone and, if that fails, the request is sent toward the server allowing intermediate nodes to capture and resolve the request. Let r be the radius of the cooperation zone, and P_d^z the probability of a node in the zone caches d locally. The probability of getting the requested data d from within the cooperation zone is as follows.

$$P(r) = 1 - (1 - P_d^z)^{\rho\pi r^2 - 1}$$

Let P_d^f be the probability of a forwarding node resolves the data request. In COOP, the probabilities of possible situations are as follows. The probability that the request is resolved locally is P_d^z, the probability that the request is resolved within the cooperation zone of the requested node is *P(r)*, the probability that the request is resolved at i (i=1, 2,..., L-r-1) hop beyond the cooperation zone is $((1 - P(r)))P_d^{Cf}(1 - P_d^{Cf})^{i-1}$, and the probability that the request is solved at the data server is $(1 - P(r))(1 - P_d^{Cf})^{L-r-1}$. Therefore, the average response delay is

$$\begin{aligned} D_{COOP} &= 0 \times P_d^{Co} + (1 - P_d^{Co})\{rP(r) + (1 - P(r))[r + P_d^{Cf}(1 + 2(1 - P_d^{Cf}) + \\ &\quad \cdots + (L - r - 1)(1 - P_d^{Cf})^{L-r-2}) + (L - r)(1 - P_d^{Cf})^{L-r-1}]\} \\ &= (1 - P_d^{Co})rP(r) + (1 - P(r)) \\ &\quad [r + P_d^{Cf} \sum_{i=1}^{L-r-1} i(1 - P_d^{Cf})^{i-1} + (L - r)(1 - P_d^{Cf})^{L-r-1}]. \end{aligned} \quad (4.3)$$

Upon a local miss and if the data is found within the zone, the energy cost is taken to be $\rho\pi\gamma^2$. Therefore, the average energy cost is

$$\begin{aligned} E_{COOP} &= (1 - P_d^{Co})(\rho\pi r^2) \times P(r) + (1 - P(r)) \\ &\quad [r + P_d^{Cf} \sum_{i=1}^{L-r-1} i(1 - P_d^{Cf})^{i-1} + (L - r)(1 - P_d^{Cf})^{L-r-1}]. \end{aligned} \quad (4.4)$$

4.3. FCCS

The average response delay of FCCS is calculated based on three possible situations: a data request can be satisfied from the local cache or local path, from a node in forwarding path, or from the data server. The probability that no requested data is found in the local cache or local path is as follows.

$$\bar{P}_d^{Fcp} = (1 - P_d^{Fc})(P_d^{Fp} P_d^x)(1 - P_d^{Fp})$$

The average response delay that the requested data is in the local cache or local path of the i-th forwarding node is as follows.

$$D^i_{Fcp} = i \times (\bar{P}^{Fcp}_d)^{i-1}[P^{Fc}_d + L^{Fc}_p \times (1 - P^{Fc}_d)P^{Fp}_d]$$

Therefore, the average response delay of FCCS is

$$\begin{aligned} D_{FCCS} &= 0 \times P^{Fc}_d + L^{Fc}_p(1 - P^{Fc}_d)P^{Fp}_d + 1 \times \bar{P}^{Fcp}_d[P^{Fc}_d + L^{Fc}_p + (1 - P^{Fc}_d)P^{Fp}_d] \\ &\quad + 2 \times (P^{\bar{F}cp}_d)^2[P^{Fc}_d + L^{Fc}_p \times (1 - P^{Fc}_d)P^{Fp}_d)] + \cdots \\ &\quad + (L_s - 1) \times (\bar{P}^{Fcp}_d)^{L_s - 1}[P^{Fc}_d + L^{Fc}_p \times (1 - P^{Fc}_d)P^{Fp}_d] + L_s \times (\bar{P}^{Fcp}_d)^{L_s} \qquad (4.5) \\ &= L^{Fc}_p(1 - P^{Fc}_d)P^{Fp}_d + \sum_{i=1}^{L_s - 1} i(\bar{P}^{Fcp}_d)^i[P^{Fc}_d + L^{Fc}_p(1 - P^{Fc}_d)P^{Fp}_d] + L_s(\bar{P}^{Fcp}_d)^{L_s}. \end{aligned}$$

The average energy cost is

$$E_{FCCS} = L^{Fc}_p(1 - P^{Fc}_d)P^{Fp}_d + \sum_{i=1}^{L_s - 1} i(\bar{P}^{Fcp}_d)^i[P^{Fc}_d + L^{Fc}_p(1 - P^{Fc}_d)P^{Fp}_d] + L_s(\bar{P}^{Fcp}_d)^{L_s}. \qquad (4.6)$$

The parameters and values of the performance evaluation for FCCS are shown in Table 4.2. In FCCS, the probability that the requested data is in the local cache P^{Fc}_d is computed as follows:

$$P^{Fc}_d = P^{Fcd}_d + P^h_{pd},$$

where P^{Fcd}_d is the local hit ratio by the CacheData method of FCCS. In (Jin and Bestavros, 2000), Jin presented an on-line algorithm that effectively captures and maintains an accurate popularity profile of Web objects requested through a caching proxy and designed a caching algorithm that utilizes such information. The local hit ratio of Jins caching algorithm is higher than that of other caching algorithms to 0.08 degree. Therefore, we consider two cases to evaluate FCCS performance.

- FCCS (A) - the value of P^{Fc}_d is higher than those of P^{Hc}_d and P^{Co}_d to 0.08 degree because the proposed FCCS utilizes not only data similarity and data utility for caching but also data profit for replacement.
- FCCS (B) - the value of P^{Fcd}_d equals to those of P^{Hc}_d and P^{Co}_d.
- The local hit ratio from the prefetched data P^h_{pd} is computed by

$$P^h_{ph} = \frac{\lambda \times P^r_{xpd} \times P^p_d}{TTL}. \qquad (4.7)$$

We assume that access rate of a data is attended with Zipf-like distribution (Zipf, 1949). In case that the number of data is 100 and the threshold of popularity index is 0.02, is

approximately 0.55 and the number of 9 data items of total 100 data items is only popular. Also, we assume that P^r_{xpd} is 0.9 because all prefetched data are popular data, According, the value of P^h_{pd} is

$$P^h_{pd} = \frac{6 \times 0.9 \times 0.55}{300} \approx 0.01.$$

Table 4.2 Parameters and values for performance evaluation.

Parameter	Value
P_d^{Hc}, P_d^{Co}	0.5
P_d^{Fcd}	0.58, 0.5
P^h_{pd}	0.01
P_d^{Hp}, P_d^{Fp}	0.2
P^x_d	0.012
P^z_d	0.07
P_d^{Cf}	0.02
λ (min)	6
TTL (min)	300
r	3 6
δ_{PI}	0.02
L_s	10 25
L_p^{Hc}, L_p^{Fc}	3
ρ	0.6
i, j, k	3, 4, 5

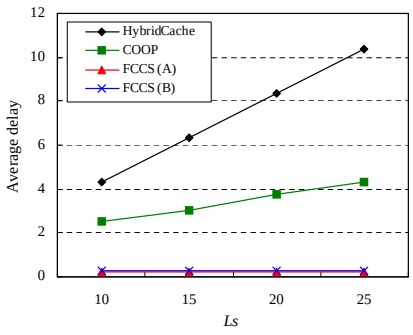

Figure 4.1 Expected travel distance with different Ls.

Fig. 4.1 shows the expected travel distance in hops between the requesting node and the original data server (Ls) in case that r is 3. From this figure, we know that the expected travel distance of FCCS is shorter than those of HybridCache and COOP regardless of Ls, and the expected travel distance of FCCS (B) closely equals to that of FCCS (A). Even though Ls is increased, the expected travel distance of FCCS is nearly constant. But the expected travel distances of HybridCache and COOP increase linearly as Ls increases.

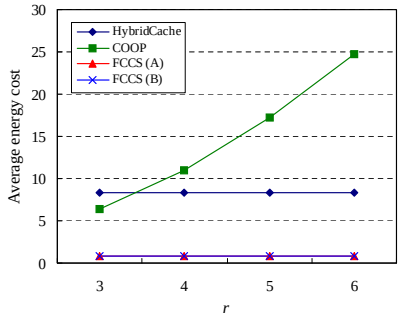

Figure 4.2 Expected energy cost with different r.

Fig. 4.2 shows the expected energy cost according to the radius of cooperation zone (r) in case that *Ls* is 20. From this figure, we know that the expected energy cost of COOP increases linearly as r increases and the expected energy cost of FCCS is smaller than those of HybridCache and COOP regardless of r, and the expected energy cost of FCCS (B) closely equals to that of FCCS (A).

The proposed FCCS is superior to HybridCache and COOP in two performance criteria: average delay and average energy cost. Therefore, we identify that the proposed FCCS is most applicable cache scheme to MANETs.

5. Concluding remarks

In this paper, we proposed FCCS, a fuzzy hybrid caching scheme for mobile ad hoc networks. The cache management of FCCS not only uses adaptively CacheData or CachePath based on fuzzy logic according to data similarity and data utility, but also uses the a replacement manager based on data profit. Also, the FCCS uses a prefetch manager. When the TTL (Time to Live) of the cached data expires, the prefetch manager evaluates the PI of the data. If the PI is larger than a threshold, the data is prefetched. Otherwise, its space is released. The performance of FCCS is evaluated analytically and is compared to that of HybridCache and COOP, where we consider two cases: FCCS (A) and FCCS (B) to evaluate FCCS performance. FCCS (A) is that the local cache hit ratio of FCCS is higher than those of HybridCache and COOP to 0.08 degree because the proposed FCCS utilizes not only data similarity and data utility for caching but also data profit for replacement, and FCCS (B) is that the local cache hit ratio of FCCS equals to those of HybridCache and COOP. From the evaluation results, we know that the proposed FCCS is superior to HybridCache and COOP in two criteria: average delay and average energy cost, and the performance of FCCS (B) closely equals to that of FCCS (A). Therefore, we confirm that the proposed FCCS is most applicable cache scheme to MANETs.

Future work includes a detailed performance analysis of FCCS through changing parameter values and simulations, and studying on semantic caching schemes for MANETs and fuzzy proxy caching schemes for SNs (Sensor Networks) and VANETs (Vehicular Ad Hoc Networks).

References

Bae, I. H. (2008). Design and evaluation of a dynamic continuous media streaming supporting method on the basis of logical grid hierarch for manets. *Journal of the Korean Data & Information Science Society*, **19**, 645-665.

Bae, I. H. (2009). Design and analytical evaluation of a fuzzy proxy caching for wireless internet. *Journal of the Korean Data & Information Science Society*, **20**, 1117-1190.

Bae, I. H. and Kim, Y-J. (2008). Design and evaluation of a fuzzy hierarchical location service for mobile ad hoc networks. *Journal of the Korean Data & Information Science Society*, **18**, 757-766.

Cao, G., Yin, L. and Das, C. R. (2004). Cooperative cache-based data access in ad hoc networks. *IEEE Computer*, **37**, 32-39.

Chand, N., Joshi, R. C. and Mistra, M. (2007). Cooperative caching in mobile ad hoc networks based on data utility. *Mobile Information Systems*, **3**, 19-37.

Deitel, H. M. and Deitel, P. J. (2008). *Internet & world wide web how to program*, Prentice Hall.

Du, Y. and Gupta, S. K. S. (2005). Coop - A cooperative caching service in manets. *Proceedings of the Joint International Conference on Autonomic and Autonomous Systems and International Conference on Networking and Services*, 58-58.

Du, Y., Gupta, S. K. S. and Varasmopolus, G. (2009). Improving on-demand data access efficiency in manets with cooperative caching. *Ad Hoc Networks*, **7**, 579-598.

Feeney, L. M. and Nilsson, M. (2001). Investing the energy consumption of a wireless network interface in an ad hoc networking environment. *IEEE INFOCOM*, 1548-1557.

Golder, S. A. and Huberman, B. A. (2006). Usage patterns of collaborative tagging systems. *Journal of Information Science*, **32**, 198-208.

Jin, S. and Bestavros, A. (2000). Popularity-aware greedy dual-size web proxy caching algorithms. *Proceedings of the 20th International Conference on Distributed Computing Systems*, 254-261.

Shim, J., Scheuermann, P. and Vingralek, R. (1999). Proxy cache design: algorithms, implementation and performance. *IEEE Transactions on Knowledge and Data Engineering*, **11**, 549-562.

Ting, Y. W. and Chang, Y. K. (2007). A novel cooperative caching scheme for wireless ad hoc networks: GroupCaching. *International Conference on Networking Architecture and Storage*, 29-31.

Yin, L. and Cao, G. (2005). Supporting cooperative caching in ad hoc networks. *IEEE Transactions on Mobile Computing*, **5**, 77-89.

Zipf, G. (1949). *Human behavior and the principle of least effort*, Addison-Wesley.