Efficient Cooperative Caching Schemes for Data Access in Mobile Ad Hoc Networks

Cheng-Ru Young[1,2], Ge-Ming Chiu[1], and Fu-Lan Wu[3]

[1] Department of Computer Science and Information Engineer, National Taiwan University of Science and Technology
{chiu, D8815005}@mail.ntust.edu.tw
[2] Department of Electronic Engineering, Chin Min College of Technology and Commerce
chengru@ms.chinmin.edu.tw
[3] BenQ Corp. Taipei, Taiwan
fulan@totoro.ee.ntust.edu.tw

Abstract. We study cooperative caching technique for supporting data access in ad hoc networks. Two protocols that are based on the notion of zone are proposed. The IXP protocol is push-based in the sense that a mobile node would broadcast an index message to the nodes in its zone to advertise a caching event. A data requester can fetch a needed item from a nearby node if it knows that it has cached the data. The second protocol, called DPIP, is explicitly pull-based with implicit index pushing property. A data requester may broadcast a special request message to the nodes in its zone asking them to help satisfy its demand. However, this is done only if its own caching information does not result in a successful fetch. Performance study shows that the proposed protocols can significantly improve system performance when compared to existing caching schemes.

1 Introduction

Recent advances in wireless communication technology have greatly increased the functionality of mobile information services and have made many mobile computing applications a reality. It provides users with ability to access information and services through wireless connections that can be retained while the users are moving. A number of novel mobile information services, such as mobile shopping aids in a large shopping mall and financial information distribution to users via mobile phones and palmtop computers, have already been implemented.

While many infrastructure-based wireless networks have been deployed to support wireless communication for mobile hosts, they suffer from such drawbacks as the need for installing base stations and the potential bottleneck of the base stations. The ad-hoc-based network structure (or MANET) alleviates this problem by allowing mobile nodes to form a dynamic and temporary network without any pre-existing infrastructure. This can be highly useful in some environments. For example, in a large shopping mall there may be an info-station that stores the prices of all goods for querying. Due to limited radio range, an info-station itself can only cover a limited geographical area. If shoppers' mobile devices are able to form an ad hoc network, they can have access to the price list even if they are beyond the radio range of the info-station. In such an environment, when a request for a price is forwarded toward the info-station, it is likely that one of the nodes along the path has already cached the

requested item. This node can send the data back to the requester so that the access time, the channel bandwidth, and the battery power can be saved.

In light of the above example, we see that caching of data offers significant benefits for applications in ad hoc networks. Cooperative caching, which allows sharing and coordination of cached data among multiple nodes, has been widely used to improve web performance in wired networks [9, 22, 25]. These protocols usually assume fixed network topology and often require high computation and communication overheads. However, in an ad hoc network, the network topology changes frequently, and mobile nodes typically have resource (battery, CPU, and wireless bandwidth) constraints, thus cannot afford high computation or communication overheads. Moreover, ad hoc networks are based on wireless communications that are unreliable in nature. Being able to access data from nearby nodes is important from performance point of view. Hence, there exists a need for new techniques that may be applied to ad hoc networks.

In this paper, we design and evaluate cooperative caching techniques for supporting data access in ad hoc networks. Two protocols that are based on the notion of zone are proposed. In the protocols in-zone broadcasts of small-sized messages are exploited to assist the locating of required data items. The first protocol, called *IXP*, is push-based in the sense that a mobile node broadcasts an index message to the nodes in its zone to advertise a caching event. A data requester can fetch a needed item from a nearby node if it knows the node has cached the item. Otherwise, it issues a request to the data center to ask for it. Any nodes along the path can redirect the request to a nearby node, instead of faraway data center, if it knows the node has cached the item. The second protocol, called *DPIP*, is explicitly pull-based with implicit index pushing property. A data requester may broadcast a special request message to the nodes in its zone asking them to help satisfy its demand. However, this is done only if its own caching information does not result in a successful fetch. *DPIP* allows a wider scope of local caching cooperation without incurring extra communication overhead. In particular, unlike previous techniques [5, 6, 7], *DPIP* exploits the implicit index pushing property in locating data items existing in nearby nodes. In addition, the proposed protocols achieve a greater level of caching cooperation through employing an appropriate cache replacement mechanism. Simulation results show that the proposed protocols may improve system performance in terms of the request success ratio and the data access time.

2 Related Works

To facilitate data access in ad hoc networks, some data replication schemes [11, 12, 13, 24] and caching schemes [18, 23, 26] have been proposed in the literature. Data replication addresses the issue of allocating replicas of data objects to mobile hosts to meet access demands. These techniques normally require a priori knowledge of the operation environment and are vulnerable to node mobility.

Unlike data replication schemes, caching schemes do not rely on distributing data objects beforehand to facilitate data access. In the conventional caching scheme, referred to as *SimpleCache* [26], a data requester always caches the received data. If subsequent requests for the cached data arrive before the cache expires, the node may use the cached copy to serve the requests. In case of a cache miss, it has to get the

data from the data center. Getting data from faraway data center will increase the response times for the requests. Recently, a cooperative caching strategy, called *CoCa*, was proposed in [5, 6]. In *CoCa*, mobile hosts share their cache contents with each other to reduce both the number of server requests and the number of access misses. In addition, a group-based cooperative caching scheme, called *GroCoCa*, has been presented in [7], in which a centralized incremental clustering algorithm is adopted by taking node mobility and data access pattern into consideration. *GroCoCa* improves system performance at the cost of extra power consumption. In the *7DS* architecture [19], users cache data and share with their neighbors when experiencing intermittent connectivity to the Internet. However, the above researches focus on the single-hop environment rather than the multi-hop ad hoc networks addressed in this work.

In ad hoc networks, finding the location of a cached copy of a data item is the core of a cooperative caching mechanism. In [17], when an object is requested, the protocol relies on flooding to find the nearest node that has maintained a copy of the object. Using flooding may potentially reduce the response time since the request can be served by a nearby node, instead of the data center faraway. However, flooding can be problematic for network communication. To reduce the overhead, flooding is limited to nodes within k hops from the requester in [17], where k is the number of hops from the requester to the data center. The overhead is still excessive especially when k is large or the network density is high. In [18], flooding is limited by imposing a threshold on route existence probability. Based on the definition of route stability, as a query packet is forwarded by hopping, its route existence probability becomes smaller. By loading the threshold of route probability into the header of a request packet beforehand, the range of cache querying can be limited. However, choosing appropriate threshold for route existence probability is challenging.

Hence options other than flooding are desirable for finding a needed data item in mobile ad hoc networks. In [23], a cooperative caching scheme has been proposed to reduce the communication and energy costs associated with fetching a web object. When a terminal M wants to get a web data W that was not cached locally, M requests W through the base station only if the base station is in the zone of M. Otherwise, M will broadcast a request message for W in its zone. If W is not cached by any of the mobile nodes in the zone, a peer-to-peer communication scheme is realized with the mobile nodes that are known to share interests with M and are at a distance that is less than the one between M and the nearest base station. The communication is based the notion of terminal profile. However, if the data correlation between mobile terminals is small, the effect of terminal profile will be lost. In [26], two caching schemes, called *CacheData* and *CachePath*, had been presented. With *CacheData*, intermediate nodes may cache data to serve future requests. In *CachePath*, a mobile node may cache the path to a nearby data requester while forwarding the data and use the path information to redirect future requests to the nearby caching site. A hybrid protocol *HybridCache* was also proposed, which improves the performance by taking advantages of *CacheData* and *CachePath* methods while avoiding 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 the time-to-leave value of the item. One problem with these methods is that caching information of a

node cannot be shared by a data request if the node does not lie on the path between the requester and the data source.

3 Zone-Based Cooperative Caching Schemes

Our research is motivated by ZRP, a zone-based routing protocol [1, 10]. In general, routing protocols for MANETs can be classified into two categories: proactive and reactive. Proactive protocols (e.g., OLSR [8], DSDV [21]) update their routing tables periodically. Reactive protocols (e.g., AODV [20], DSR [16]), on the other hand, do not take any initiative in finding a route to a destination until a routing demand arises, thus a priori reduce the network traffic. ZRP is a hybrid protocol that combines reactive and proactive modes. In ZRP, a zone is associated with each mobile host and consists of all the nodes that are within a given number of hops, called *radius* of the zone, from the host. Each node proactively maintains routing information for the nodes in its zone. In contrast, a reactive protocol is used to reach any node beyond its zone.

We use the notion of zone as ZRP in this research. The basic idea of our scheme is to have each mobile host share its caching contents with those in its zone or beyond without the need for group maintenance. Our design rationale is twofold. As stated previously cooperative caching is possible among neighboring nodes, and zone reflects such notion of vicinity. Second, aided by the underlying routing protocols such as ZRP, a zone can be readily formed and maintained for a mobile host even if the node is on the move. In the following, we present a simple protocol called *Index Push* (or *IXP*) and a more sophisticated one called *Data Pull/Index Push* (or *DPIP*) for implementing the zone-based cooperative caching.

3.1 System Model

We consider a MANET where all mobile nodes cooperatively form a dynamic and temporary network without any pre-existing infrastructure. There exists a *data center* that contains a database. Each mobile host may send a request message to the data center for accessing a data item. When a node fetches a data item, it always stores the item in its local cache for future use like conventional caching scheme. We assume that each node has limited cache space, so only a portion of the database can be cached locally. If the cache space of a node is full, the node needs to select a victim data item from its cache for replacement when it wants to cache a new one. To reduce access latency and to ease the load of the data center, an intermediate node on the forwarding path between the requester and the data center can directly deliver the requested data to the requester if it has a copy in its local cache, or redirect the request to some nearby node that it knows has cached the data item.

In our system, we also make the following assumptions:

(1) All data items are of the same size.
(2) For sake of simplicity and standing out the salient features of our proposed schemes we do not consider updates of the data items.
(3) We assume that ZRP is the underlying routing protocol used in the MANET although this is not indispensable for our schemes.

3.2 *Index Push* (*IXP*) Protocol

The idea of *Index Push* is based on having each node share its caching content with those in its zone. To facilitate exposition, we call the neighboring nodes in the zone of a node M the *buddies* of M. If M_1 is a buddy node of M, M is also a buddy of M_1. To this end a node should make its caching content known to its buddies, and likewise its buddies should reveal their contents to the node. One way of arriving at this is for a node that has cached a new data item to advertise such a caching event. *Index Push* (*IXP*) takes this approach. It broadcasts an index message to its buddies whenever it caches a data item. The id of the data item that has been cached is included in the index message. A node may receive multiple index messages from different buddies that are associated with the same data item. Each node maintains an index vector, denoted as *IV*. An *IV* has N elements, where N is the number of data items; each element is associated with a distinct data item. Each element in *IV* has three entries that are used to record the caching information of the corresponding item. Consider the *IV* of a node M. The first entry associated with data item x is of type binary and is represented by *IV[x].cached*. This entry indicates whether x is cached locally. If the entry is TRUE, it means that x is locally available; otherwise x has to be obtained from some remote site. The second entry, denoted as *IV[x].cachednode*, is used to record a nearby node that has cached x. For sake of saving storage space M only records the last node that has broadcast an index message associated with x. The third entry, represented by *IV[x].count*, contains a count of *M's* buddies that are known to have cached x since x is cached by M. As described later, this count will be used for cache replacement purpose. Initially, any *IV[x].cached* is set to FALSE, *IV[x].cachednode* is set to NULL, and *IV[x].count* is set to zero.

(a) Data Accessing/Caching

Consider that a node M wants to access a data item x. M first checks its *IV[x].cached* to see whether the data item has been cached locally. If the entry is FALSE, M proceeds to examine *IV[x].cachednode* expecting someone in the neighborhood may offer some help. If the entry is NULL, M sends a request message to the data center directly. If the entry is not NULL, M issues a request message to the node, say M_1, if M_1 is still in the zone. Due to node mobility, it is possible that M_1 may no longer stay in *M's* zone. With ZRP, if M_1 is not inside *M's* zone M has no routing information about how to reach M_1. To avoid the overhead for searching for a path to M_1, M would send a request message to the data center directly. An intermediate node I on the path to the data center can redirect the message to a buddy node that I knows has cached the data item according to its *IV[x].cachednode* entry. When M eventually receives x, it caches x. In doing so it may possibly need to discard another cached data item, say y, if its cache is full. M sets its *IV[x].cached* to TRUE and *IV[y].cached* to FALSE. Moreover, M will reset its *IV[x].count* to 0. As described in the next section, this is performed in order to avoid having x be chosen for replacement soon after it is cached. M then broadcasts an index message to its buddies. Included in the index message are ids of the newly cached item x and the replaced data item y. Upon receiving the index message *M's* buddies update their *IV[x].cachednode* entries with M, increase *IV[x].count* by one and decrease *IV[y].count* by one. The last two operations, i.e. updating *IV[].count* entries, need to be performed by a buddy node only if the corresponding data items are cached by the

node. Furthermore, if a buddy has recorded *M* in its *IV[y].cachednode*, it needs to set the entry to NULL because *y* is no longer cached by *M*.

(b) Cache Replacement

If a node *M* accesses a data item when its cache space is full, some cached item must be removed to make room for the new one. In *IXP*, we use *IV[].count* entry for cache replacement. Recall that this entry indicates the number of *M's* buddies that have cached a data item since *M* cached the same item. *IXP* replaces the data item that has the maximum *IV[].count* among all cached ones. Replacing such a data item tends to induce less impact on *M*'s buddies because there are less buddies relying on *M* for fetching the data item when the associated count becomes bigger. In addition, *M* and its buddies tend to have greater chance in finding the replaced data item in their neighborhood than in finding the other cached items. Moreover, doing so can limit caching duplicates. This may also explain why *M* needs to set the *IV[x].count* to 0 when it first caches a data item *x*. At this time, all of *M's* buddies are supposed to have their *IV[x].cachednode* entries point to *M*, hence we do not want to have *x* replaced too soon. Notice that once a data item *x* is chosen for replacement, the values of *IV[x].count* maintained by *M*'s buddies will also be decremented, implying less chance for these buddies to have *x* replaced. This can alleviate the problem of concurrently replacing the same item by all the nodes in the neighborhood.

3.3 *Data Pull/Index Push (DPIP)* Protocol

IXP is essentially push-based in the sense that a caching node "pushes" the caching information to its buddies. Each node has a view of the caching status in its zone only. However, due to mobility of the nodes and some limitations of mobile devices such as transient disconnection, the caching status reflected by *IV* may be obsolete or not up-to-date. For example, suppose that, according to node *M's IV*, none of *M's* buddies caches *x*. If a new node that has cached *x* moves into *M's* zone now, this caching information cannot be captured by *M's IV* with *IXP*. In the following, we propose a more sophisticated protocol called *Data Pull/Index Push* (*DPIP*) to deal with this problem.

Similar to *IXP*, each node maintains the *IV* vector. When a node *M* wants to access a data item *x* that is not cached by itself, it first examines the entry *IV[x].cachednode* to see if some buddy node in the zone has cached *x*. If such buddy node exists, *M* sends a request message to the node asking for a copy of *x* in the same way as *IXP*. However, unlike *IXP*, if *IV[x].cachednode* entry is NULL, *M* broadcasts a special request message *srg* to all of its buddies. The *srg* message carries the ids of the requested data item, *x*, and the data item that will be replaced if the cache space is already full. Upon receiving the *srg* message a buddy node will reply to *M* if either of the following conditions is met: (1) it has cached *x*, (2) it knows some of its buddies has cached *x* (as per its *IV*). To reduce the number of reply messages, only peripheral nodes, i.e. the nodes at the perimeter of *M's* zone, are required to reply under the second condition, but any node for which the first condition is met should reply. In contrast with *IXP*, *DPIP* increases the chance for the data requester *M* to obtain a copy of *x* from nodes in its vicinity. This can be argued as follows. In addition to the fact that *M* can fetch a copy of *x* from its buddies if the first condition is met, it may possibly obtain caching status of the nodes that are beyond its zone but within the

zones of its buddies as specified by the second condition. Consequently, the scope of local cooperation is essentially extended by a factor of two, in terms of radius, at the data requester site. In addition, the in-zone broadcast, which initiates the "data pulling" operation, allows *DPIP* to use the latest caching information. From above description, we see that in-zone broadcast of the *srg* message implicitly advertises the fact that the requester node will cache a copy of the requested data item, and it will soon be able to help others in satisfying their demands for the item. In other words, *srg* messages serve two functions: data pulling and index pushing (in implicit manner). Note that when a *srg* message is broadcast, a timer is started. If no reply is received from *M's* buddies before the timer goes off, *M* sends a request message toward the data center. *IV* vectors are updated in the same way as *IXP* at both the requester site and its buddies. However, the update of *IV* is done by the buddies at the time when they receive either a broadcast *srg* message or a direct (unicast) request message from the requester site. Cache replacement is performed in the same way as *IXP*.

4 Simulation Study

In this section we present simulation results for the proposed protocols and compare them to *HybridCache*.

4.1 Simulation Configuration

We have developed a simulation model based on the ns-2 simulator [27] with CMU wireless extension. The simulated network consists of 50 nodes spread randomly in an area of 1500m×320m, similar to the model used in [26]. One node is designated as the data center, and it is fixed at the upper left-hand corner of the area throughout the simulation. For ease of implementation, we use the DSDV [21] as the underlying routing protocol. It is assumed that the wireless transmission range of the nodes is 250 meters and the channel capacity is 2Mbps. A node moves according to the random waypoint model [2], in which each node selects a random destination in the specified area and moves toward the destination with a speed selected randomly from the range (1m/s, 10 m/s). After the node reaches its destination, it pauses for a period of time and then repeats the movement pattern. The pause time is used to represent node mobility. The default pause time is 300 seconds.

The data center contains 3000 data items. Data updates are not considered in the model. Data requests are served on FCFS basis at all nodes. The size of each data item is 1000 bytes, other messages such as request messages and index messages are all assumed to be 20-byte long. Each node generates a sequence of read-only requests. The inter-arrival times of data requests follows exponential distribution with mean 20 seconds. The data items are accessed uniformly. We consider a data request failed if the requested data item is not returned within a given amount of time. This is employed to account for packet loss in the unreliable wireless network. In addition, a data request that takes an excessive amount of response time is, in most cases, abandoned by the client.

To capture the performance of the protocols several metrics are examined in the simulation. The first principal metric is the request failure ratio which gives the ratio

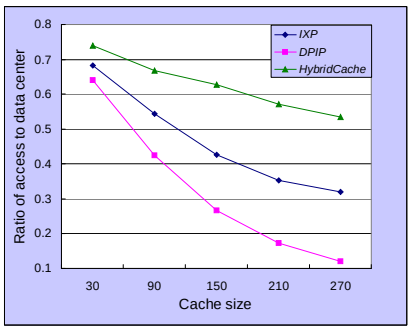

Figure 1: Request failure ratio vs. cache size

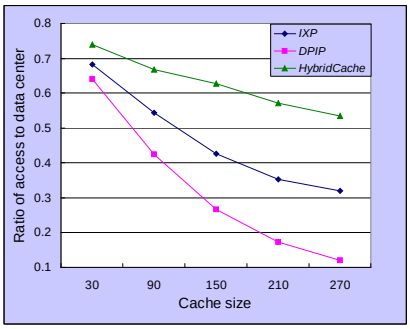

Figure 2: Ratio of access to data center vs. cache size

of data requests that fail to receive the requested data items. The second metric we measure is the average data access time for a successful data request. The other metric of interest is the ratio of data requests that are served by the data center. Reducing such ratio mitigates the workload of the data center, and better load balance results.

As we are interested in the steady state of the system we allow the simulated network to warm up for 1000 seconds. The simulation results are collected for 4000 seconds. In our simulations the radius of a zone is set to one hop for both *IXP* and *DPIP* protocols.

4.2 Simulation Results

Figure 1 illustrates the request failure ratios for *IXP*, *DPIP* and *HybridCache* with cache size varying from 30 to 270 items. In this simulation the pause time is set to 300 seconds. Apparently, both *IXP* and *DPIP* outperform *HybridCache* by significant margins. In addition, traffic congestion near the data center may cause request failures as well. Note that our schemes, *DPIP* in particular, demonstrate more evident improvement, in comparison with *HybridCache*, as the cache size increases. This result indicates that our protocols offer great capability for exploiting localized cooperative caching.

Figure 2 shows the ratios of the requests messages that are eventually addressed to the data center. Reducing the chances of having data requests satisfied by the data center is essential to the efficiency of the cooperative caching techniques. *DPIP* is least likely to lead the requests to be sent to the data center, followed by *IXP* and then *HybridCache*. A smaller such ratio implies, to some extent, better load balance among the data center and the nodes that have cached the requested data items. Again, *DPIP* is most sensitive to the cache size with respect to this metric.

In Figure 3 we illustrate the average access times of the successful data requests for the same setting as used in Figure 1. Since the access times do not consider the failed data requests, a straightforward comparison of the three protocols using this figure may not be appropriate. From Figure 3 we see that *DPIP* does not offer advantages in terms of the average access time when the cache size is very small. This is mainly due to the fact that *srg* broadcasts in *DPIP* are not effective enough to compensate for the

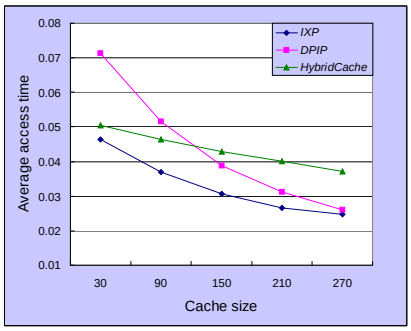

Figure 3: Average data access time vs. cache size

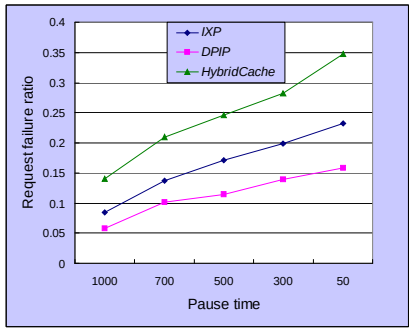

Figure 4: Request failure ratio vs. pause time

timeout intervals experienced by the data requesters when no cache hit among their buddies results. In this situation, both *IXP* and *HybridCache*.

To evaluate the effect of node mobility on the performance of the protocols we have performed simulation with different pause times. The result is shown in Figure 4, in which the request failure ratios are plotted against the pause times. Note that as the pause time decreases the node mobility becomes higher. All three protocols are affected when node mobility increases. *DPIP* is least sensitive to node mobility because it provides a comparatively wide scope of caching cooperation for neighboring mobile nodes. *HybridCache* is the most sensitive with respect to the metric.

5 Conclusion

In this paper, we have presented two zone-based cooperative caching protocols for MANETs. Both protocols demonstrate sensitivity with respect to the cache size. This indicates their capability for exploiting localized cooperative caching. Owing to this observation the benefits of the proposed protocols should be evident for MANETs of large scale.

References

1. R. Beraldi, R. Baldoni, “A Caching Scheme for Routing in Mobile Ad Hoc Networks and Its Application to ZRP,” IEEE Transactions on Computers, Vol. 52, Issue 8, August 2003, pp. 1051-1062.
2. J. Broch, D. Maltz, D. Johnson, Y. Hu, and J. Jetcheva, “A Performance Comparison of Multi-Hop Wiress Ad Hoc Network Routing Protocols,” ACM MobiCom, October 1998, pp. 85-97.
3. G. Cao, L. Yin, C. R. Das. “Cooperative Cache-Based Data Access in Ad Hoc Networks,” Computer, vol. 37, Issue 2, Feb 2004, pp. 32-39.
4. C. C. Chiang, H.K. Wu, W. Liu, and M. Gerla, “Routing in Clustered Multihop Mobile Wireless Networks with Fading Channel,” Proc. of IEEE Singapore International Conference on Networks, 103 Singapore, April 1997, pp. 197-211.

5. C.-Y. Chow, H.V. Leong, A. Chan, "Peer-to-Peer Cooperative Caching in Mobile Environments," Proc. of the 24th ICDCS Workshops on MDC, pp. 528-533.
6. C.-Y. Chow, H.V. Leong, A. Chan, "Cache Signatures for Peer-to-Peer Cooperative Caching in Mobile Environments," Proc. of AINA 2004, pp. 96-101.
7. C.-Y. Chow, H.V. Leong, A. T. S. Chan, "Group-based Cooperative Cache Management for Mobile Clients in Mobile Environments," Proc. of ICPP 2004, pp. 83-90.
8. T. Clause, P. Jacquet, and A. Laouti, P. Minet, P. Muhlethaler, A. Qayuam, L. Viennot, "Optimized link state routing protocol," draft-ietf-manet-olsr-07.txt, IETF, http://www.ietf.org, November 2002.
9. L. Fan, P. Cao, J. Almedia, and A. Broder, "Summary cache: A scalable wide area web cache sharing protocol," ACM SIGCOMM 1998, pp. 254-265.
10. Z. J. Haas and M. R. Pearlman, "The Zone Routing Protocol (ZRP) for Ad Hoc Networks," Internet Draft, draft-ietf-manet-zone-zrp- 01.txt, 1998.
11. T. Hara, "Effective Replica Allocation in Ad Hoc Networks for Improving Data Accessibility," Proc. of IEEE INFOCOM 2001, pp. 1568-1576.
12. T. Hara, "Replica Allocation in Ad Hoc Networks with Periodic Data Update," Proc. of International Conference on Mobile Data Management, 2002, pp. 79–86.
13. T. Hara, "Replica Allocation Methods in Ad Hoc Networks with Data Update," Mobile Networks and Applications, Vol.8, No.4, August 2003, pp. 343-354.
14. M. Jiang, J. Li, and Y.C. Tay, "Cluster based routing protocol (CBRP) functional specification," Internet Drafe, draft-ietf-manet-cbrp-spec-00.txt., 1998.
15. Jie Wu and Fei Dai, "A Generic Distributed Broadcast Scheme in Ad Hoc Wireless Networks," IEEE Transactions on Computer, Vol. 53, No. 10, October. 2004, pp. 1343-1354.
16. D. B. Johnson and D. A. Maltz, "Dynamic Source Routing in Ad Hoc Wireless Networks," Mobile Computing, edited by Tomas Imielinski and Hank Korth, Kluwer Academic Publishers, ISBN: 0792396979, Chapter 5, 1996, pp. 153-181.
17. W. Lau, M. Kumar, and S. Venkatesh, "A Cooperative Cache Architecture in Supporting Caching Multimedia Objects in MANETs," The Fifth International Workshop on Wireless Mobile Multimedia, 2002.
18. T. Moriya and H. Aida, "Cache Data Access System in Ad Hoc Networks," Vehicular Technology Conference, Vol. 2, April 2003, pp. 1228-1232.
19. M. Papadopouli and H. Schulzrinne, "Effiects of power conservation, wireless coverage and cooperation on data dissemination among mobile devices," ACM MobiHoc, Oct. 2001, pp. 117-127.
20. C. E. Perkins and E. M. Royer, "Ad-hoc On-Demand Distance Vector Routing," Proc. of IEEE WMCSA 1999, pp. 90-100.
21. C. E. Perkins and P. Bhagwat, "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers," Proc. of ACM SIGCOMM 1994, pp. 234-244.
22. A. Rousskov and D. Wessels, "Cache Digests," Computer Networks and ISDN Systems, Vol. 30, No. 22-23, 1998, pp. 2155-2168.
23. F. Sailhan and V. Issarny, "Energy-aware Web Caching for Mobile Terminals," Distributed Computing Systems Workshops, July 2002, pp. 820-825.
24. M. Tamori, S. Ishihara, T. Watanabe, and T. Mizuno, "A Replica Distribution Method with Consideration of the Positions of Mobile Hosts on Wireless Ad-hoc Networks," Distributed Computing Systems Workshops, July 2002, pp. 331-335.
25. D. Wessels and K. Claffy, "ICP and the Squid Web Cache," IEEE Journal on Selected Areas in Communication, Mar. 1998, pp. 345-357.
26. L. Yin and G. Cao, "Supporting Cooperative Caching in Ad Hoc Networks," Proc. of IEEE INFOCOM 2004, pp. 2537-2547.
27. ns Notes and Documentation. (http://www.isi.edu/nsnam/ns/)