IDEA GROUP PUBLISHING
701 E. Chocolate Avenue, Suite 200, Hershey PA 17033-1240, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com

ITJ3478

This paper appears in the publication, **International Journal of Business Data Communications and Networking**, Volume 3, Issue 1
edited by Jairo Gutierrez © 2007, Idea Group Inc.

Cooperative Data Caching and Prefetching in Wireless Ad Hoc Networks

Mieso K. Denko, University of Guelph, Canada

ABSTRACT

This article proposes a cooperative data caching and prefetching scheme for Mobile Ad Hoc Networks (MANETs). In this scheme, multiple hosts cooperate in both prefetching and caching commonly used data. To reduce communication and computational overhead, we use a clustering architecture for the network organization. A weak consistency based on time to live value was used to maintain data consistency. A hybrid cache replacement policy that uses frequency of access and the reference time was employed. The effects of cache size, mobility, and prefetching threshold on the network performance were investigated in a discrete event simulation environment. The contribution of intra-cluster and inter-cluster information to overall data accessibility ratio was also investigated. The simulation results indicate that the proposed scheme improves both data accessibility and query delay at relatively lower prefetch thresholds, larger cache sizes, and moderate mobility.

Keywords: ad hoc networks; cooperative caching; data management; mobile networks; prefetching; wireless networks

INTRODUCTION

In the past few years, most of the research devoted to MANETs has focused on the development of routing protocols to increase connectivity among mobile hosts in a constantly varying topology (Johnson & Maltz, 1996; Perkins & Bhagwat, 1994). Although development of routing protocols is one of the main challenges that must be addressed, improved data accessibility is the ultimate goal of such networks. In order to enable quick deployment of MANETs, development of reliable and efficient data management schemes suitable for this network environment is crucial. Data caching and prefetching techniques used in traditional wireless networks can be extended to be used in MANETs. In this article, we investigate the use of caching and prefetching techniques for improving data accessibility and reducing latency in MANET environments.

Caching has been utilized extensively in wired networks, such as the Internet, to increase the performance of Web services (Fan et al., 1998; Rousskov & Wessels, 1999; Wang, 1999; Wessels & Claffy, 2005). However, existing cooperative caching schemes cannot be implemented directly in MANETs due to host mobility and resource constraints that characterize these networks. Consequently, new approaches have been proposed to tackle these challenges (Cao, Yin, & Das, 2004; Hara, 2002; Lim, Lee, Cao, & Das, 2003; Papadopoui & Schulzrinnr, 2001; Wang, 2005; Yin & Cao, 2006). These approaches have been introduced to increase data accessibility and reduce query delay in MANETs. A cooperative cache-based data access scheme is subsequently proposed for ad hoc networks (Cao et al., 2004; Yin & Cao, 2006). Three caching techniques, namely CacheData, CachePath, and HybridCache, are utilized as caching approaches. In CacheData, the intermediate hosts, which are located along the path between the source host and the destination host, cache frequently accessed data items. In CachePath, the intermediate hosts record the routing path information of passing data. CachePath only records the data path when it is closer to the caching host than the data source. 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. Moreover, the authors did not consider prefetching and multiple data sources in their study. In Lim et al. (2003), a similar approach is proposed for data caching in a network that integrates ad hoc networks with the Internet.

In Hara (2002), a replica allocation scheme with periodic data item updates is proposed. This scheme focused on improving data accessibility with the main goal of decreasing the data access failure in response to network division. The schemes presented in Sailhan and Issarny (2003) and Wang, (2005) are based on a specific routing protocol. The scheme in Sailhan and Issarny (2003) used popularity, access cost, and coherency as criteria to replace cached data items when a mobile host's cache space is full. In Wang (2005), a transparent cache-based mechanism based on a new on-demand routing protocol called dynamic backup routes routing protocol (DBR2P) is proposed. The routing protocol and the cache mechanism allow the caching of data. In order to guarantee data access, this scheme allowed the cached data to be moved to a backup host in response to a link failure. Another study proposed the implementation of an architecture similar to cooperative caching, which defines two protocols to share and disseminate data among mobile hosts (Rousskov & Wessels, 1999). However, the scheme focused on data dissemination in a single-hop rather than cooperative caching in a multi-hop environment. Another study utilized a novel architecture for database caching based on the separation of queries and responses (Artail, Safa, & Pierre, 2005). The experimental results indicated that the scheme improved data accessibility by reducing response time in the presence of host mobility.

Cooperative caching is an effective mechanism for increasing data accessibility in both wired and wireless networks. However, caching alone is not sufficient to guarantee high data accessibility and low communication latency in dynamic systems with limited network resources. In this article, we propose an integrated cooperative caching and prefetching mechanism for MANETs.

This article provides the following contributions to increasing the efficiency of data management in MANETs. First, we use a clustering architecture that allows localized and adaptive data caching and prefetching mechanisms to increase data accessibility and reduce latency in the presence of host mobility. Second, we use a cache replacement policy that combines both frequency of access and latency of access to the cached data. Thus, eviction of data in the cache depends on a metric that combines the optimal combination of access frequency and time of reference with a configurable parameter. Third, the proposed cooperative caching and prefetching architecture is flexible and does not rely on any specific routing protocol. Fourth, the article provides an analysis of the contribution of intra-cluster and inter-cluster information to the data accessibility ratio.

The remainder of this article is organized as follows. Section 2 presents the proposed system architecture. Section 3 presents the cooperative caching and prefetching strategies. Section 4 presents the cache replacement policy and data consistency management. Section 5 presents the results of performance evaluation based on simulation experiments. Finally, Section 6 presents conclusions and future research work.

THE PROPOSED SYSTEMS ARCHITECTURE

Network Model and Assumptions

The network consists of mobile hosts that form clusters. The network connectivity is maintained using a periodic Hello message that is exchanged among one-hop neighbors. Other information such as the data stored at a host, host's role (cluster head, data source, or caching agent) are exchanged among neighbors. The clustering algorithm is used for cluster management. These tasks include cluster head election, monitoring cluster membership changes, and facilitating inter-cluster communications. We assume that each host has a cache of a fixed capacity and cached data can be accessed by any other host. Data caching and prefetching operations are carried out cooperatively to avoid extra communication overhead.

The Architecture

Cooperative caching is particularly attractive in environments where the network is constrained in terms of bandwidth, power, and storage. Cooperative caching offers several benefits since it can enable efficient utilization of available resources by storing different data items and sharing them among themselves. Cooperative caching additionally improves performance by increasing data accessibility and reducing communication latency.

In this study, we consider data management in a large ad hoc network. The network organization is based on a clustering architecture with a cluster head. Hosts that allow communication between two clusters are called gateways. Figure 1 gives an example of the proposed architecture with two clusters. We used cooperative clusters with cluster heads (CH). Each cluster has a CH, data source (DS), caching agents (CAs), and mobile hosts (MHs). The DS generates data items needed by other MHs in the network. Multiple data sources store different data items. The hosts that act as DSs are known to the CHs and local

Figure 1. The cluster architecture

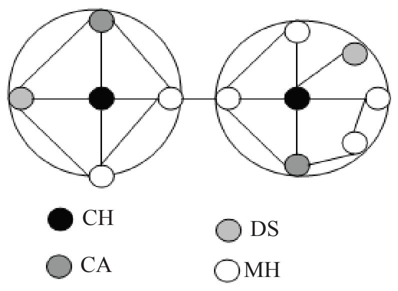


Figure 2. Data request-reply

```
Begin
   When a mobile host, N, requests a data item, D:
   Call cache_avail;
   If a copy of D exists in its cache and valid then
      Return D;
   Else
      Requests D from Neighbors/CA/DS
      If D exists and still valid then
         Return D;
      Else // no data in the cluster
      Requests D through CH;
End
```

CAs. For clustering, we used the lowest ID clustering algorithm proposed in Gerla and Tsai, 1995. However, any existing distributed clustering algorithm can be used. We used a secondary cluster head to cope with cluster head changes and data loses (Lu, 2005).

Cache Placement and Host Organization

Each host has its own caching manger (CM) and prefetching manager (PM). The PM uses information from the CH and other hosts to make prefetching decisions. The PM maintains a list of data items to be fetched based on the implementation of the data prefetching algorithm and sends request to neighbors or to the CH. The CM monitors cache size, cached data lifetime, and cache replacement operation. It uses a cache replacement algorithm to maintain a list of data items to be replaced when other data items are fetched.

When a host does not have the required data, it forwards the request to its neighbors, then to the CA, DS, or CH. Each cluster maintains information on a local data source (DS). When a MH receives the requested data item, it will decide whether the data item should be cached.

A data item is placed at a specific host in the cluster. A fixed threshold distance metric of k-hop will be used for deciding a CA. The number of hops is adjusted based on host mobility and link characteristics. Larger hop lengths are used for less mobile and more stable networks. For example, maintaining three-hop neighborhood information in a dynamic network will cause more update overhead than maintaining one-hop neighborhood information since more transmission is involved. Moreover, the link may be changed before it is used the next time. In general, cache placement at the CA is carried out based on distance metric, cache replacement algorithms, and capacity constraints.

COOPERATIVE CACHING AND PREFETCHING STRATEGIES

Cooperative caching involves cache placement, cache replacement, data request-reply operations, and cache consistency. The data request-reply operation also known as data search can be performed proactively or based on demand. Most existing solutions in data management for MANETs consider cooperative caching without prefetching. In this article, we consider cooperative caching with prefetching.

Types of Cooperative Caching

We consider two types of host cooperation. The first type of cooperation involves cooperative data access and storage. Neighbor hosts store different data items and serve any request made for such data items by any host interested in the data. The second form of host cooperation involves cooperative prefetching and cache placement. When the data item is prefetched from a DS or CA, the data can be cached at any other CA if it is fresh relative to the existing data item. In other words, the existing data item can be updated by a CA in the cluster, even if the request for data was not initiated by it. However, such updates are only performed locally within a cluster.

Data query and reply is forwarded on a hop-by-hop basis. The data request begins at a local cluster. If the request fails, the search will continue through other clusters until the DS or any CA containing fresh data responses to the query. The query reply is forwarded on a reverse route in the same manner. The request will be unsuccessful only when the network is partitioned. The solution for this problem involves data replication. This specific problem was already discussed both in the literature and in our own earlier work (Feldmann, 1999; Hara, 2002; Lu, 2005). Although disconnected operation is a norm (not an exception) in MANETs, permanent failure due to battery power depletion or network partition could result in a complete loss of data if no replicated copy is available.

Prefetching Strategies

Caching becomes more effective if the requested data item is available in the cache when needed. Strategies for proactively prefetching the most frequently accessed data within the cache or prefetching frequently needed data upon the expiry of Time-To-Live (TTL) can significantly improve network performance by reducing latency. To reduce communication overhead, we use the prefetch-on-mis scheme when the value of TTL expires for a particular data item.

We propose implementation of a prefetching strategy that works at two levels. The first level of prefetching is performed between hosts and caching agents, whereas the second level of prefetching is performed between CAs and the CH.

1. **Prefetching between MHs and CAs:** A host that frequently needs a particular data item will prefetch the data from the CA or DS if the data has not already been cached in its neighbor. Under such a circumstance, a host will become a CA. It will notify its CH, its neighbors, and the CAs in its cluster, of its status change and the ID of the data it is storing.
2. **Prefetching between CAs and CHs:** If a CA has an expired data item, it will prefetch the item from other CAs or DSs in its cluster. If the CA cannot find a fresh data item from within its cluster, it will send its request to the CH, which will in turn request the data from other clusters. The pseudocode for Request-Reply is shown in Figure 2.

In the Request-Reply process, any host that receives a data item request will first check its own cache space and its validity and then decide whether to send data to the requester or forward the query to the other host or CH.

The Prefetching Algorithm

The performance of prefeching depends on the quality of the prediction mechanism used. The algorithm should be adaptive and it should use distributed information gathering mechanisms. We used a popularity-based prediction algorithm. The PM maintains the required statistics and implements the prefetching algorithm.

By prefetching frequently accessed data in the local cache within a cluster, latency can be reduced significantly. The frequency of access to data is determined based on the past access history for a particular data item. Based on these statistics, the data items that are most likely to be needed in the near future will be prefetched and cached. To achieve this, hosts maintain frequency of access request statistics for each data item.

Each host maintains a Node Prefetching Index (NPI) for each data item (D_i) as follows:

$$NPI(D_i) = \frac{n_i}{N_k}, i \geq 1 \qquad (1)$$

where N_k is the total number of distinct access requests at the host k, n_i is the total number of distinct requests for the data item (D_i). The value NPI shows the ratio of access to the data. Hence, a higher value of the NPI provides evidence of the popularity of that data at the host.

Each caching agent fetches the data item if the data is popular, which is determined by the cumulative prefetching index (CPI) using a predefined threshold. For each data item (D_i), the value of CPI is computed as follows:

$$CPI(D_i) = \sum_{i=1}^{k} (NPI(D_i)) \qquad (2)$$

where k is the number of hosts. The fetch index is adjusted based on past access history and an update will be sent to all CAs in the cluster. These indexes are updated whenever the data item is accessed.

CACHE REPLACEMENT AND DATA CONSISTENCY

Cache Replacement Algorithm

The primary purpose for cooperative cache placement is to avoid duplicated storage of the same data item at neighbor hosts. This reduces data access costs in terms of the number of hops required for data transmission to obtain the data. A clustering architecture is suitable for partitioning the network into smaller and more manageable groups.

Cache placement determines whether a received data item should be cached. However, when a MH's cache space is full and a new data item should be cached, cache replacement will determine which cached data item should be removed from the cache space.

Cache replacement algorithms play a major role in determining the performance of a caching scheme. There are two scenarios in which cache entries could change. The first scenario occurs when the data stored in the cache becomes invalid. In this case, the invalid data item is replaced. If the data is popular, it will be prefetched and placed in the cluster for future use. The second scenario occurs when a MH's cache space is full and a data item has to be cached. In this case, the existing data should be evicted and replaced by new data. The MH has to decide which data item in its cache space should be removed to make room for the new data item. Thus, our proposal involves prefetching the required data item and distributing it to hosts for caching in designated CAs. The data item would be cached in the CAs by replacing expired TTLs or by evicting existing data using cache replacement algorithms.

If all data items are valid, a cache replacement algorithm is used to evict cache content and store new data items. Most existing cache replacement schemes use policies such as least recently used (LRU) and least frequently used (LFU). In the LRU policy, the least recently accessed cached data is replaced, while in the LFU policy, the least frequently accessed data is replaced. The LRU may replace data that has not been accessed for a long time, even if this data may be needed by multiple hosts later. This is plausible, since hosts can join or leave the network randomly and they do so frequently. Furthermore, LFU alone may not be useful since the frequency of access may not be stable in a dynamic network. Consequently, we used the combined metric that allows replacing the least frequently used data with the least recent references.

The LRFU Cache Replacement Policy

To avoid removing data that may be needed soon, we used a cache replacement algorithm that makes use of both frequency and latency of access information. This hybrid cache replacement policy combines LRU and LFU and is known as the least recently-least frequently used (LRFU) algorithm. LRFU removes data items that have the smallest combined values for both frequency and latency of access. If multiple data items have the same frequency of access, then one of them will be evicted based on their TTL value.

According to the LRFU policy, each host assigns a value called combined recency and frequency (CRF) that estimates the probability that the data will be accessed in the future. Past references to the data contribute to this value based on a weighing function, F(x), where x is the time span between past references and the current time.

The CRF value of a data item, D, at time t_c is computed as follows (Lee et al., 1997):

$$CRF_{tc}(D) = \sum_{i=1}^{k} F(t_c - t_{bi}) \qquad (3)$$

where F(x) is a weighing function and $\{t_{bi}\}$, i, 1… k, are the reference times of data items D and $t_{b1} < t_{b2} < t_{b3} < \ldots \leq t_c$.

The LRFU policy differs from LRU in that it takes other references into account. Furthermore, LRFU differs from LFU in that the contribution of each reference depends on its latency. As the weighting function, we used the function:

$$F(x) = \left(\frac{1}{5}\right)^{\delta x} \qquad (4)$$

where x is the difference between the current time and reference time and δ is a control parameter.

It is evident from the above weighing function that F(x) approaches the LFU as δ approaches zero, and it approaches LRU as δ goes to one. By varying the control parameter δ, the performance of the LRFU policy can be improved.

Cache Consistency Management

Data items stored in the DS are classified as either dynamic data or static data. The value of a dynamic data item changes frequently, while the value of a static data item does not. The cache consistency check ensures that the cached data is consistent with the original data at the DS. Because static data items seldom change, the network traffic caused by an update broadcast is minimal. By dividing data items into static data and dynamic data, the overhead caused by maintaining cache consistency will be greatly reduced.

For dynamic data, a simple weak consistency (Alex, 1992; Wessels & Claffy, 1998; Yin & Cao, 2005) model based on the TTL mechanism is used. The DS assigns a TTL value to all dispatched data items. The TTL value of a data item is computed at the DS as follows:

$$TTL = \min\{\lambda(current - created), \rho\} \quad (5)$$

where λ and ρ are predefined constants. The parameters, *Current and created* refer to the current time and the creation time of the data item respectively. The parameter ρ represents a predefined threshold. To determine whether the TTL value of a data item is valid, a host computes the Current TTL (CTTL) as follows:

$$CTTL = (TTL - (current - initial)) \quad (6)$$

where *current* is the time when this data item was found in cache space, and *initial* is the time when this data item was dispatched from the DS. If the value of CTTL is less than or equal to zero, this data item expires. Otherwise, it is considered valid. When the TTL expires, the data is removed from the cache and the entry is marked with a flag to indicate the invalid status. This information will be sent to neighbors to avoid any request to this data later.

PERFORMANCE EVALUATION

The implementation was run in the Network Simulator (2005) environment in order to evaluate the proposed architecture. Using the hybrid cache replacement policy, we evaluated the performance of integrated cooperative caching and prefetching with cooperative caching. We used the AODV protocol (Perkins, Belding-Royer, & Chakeres, 2005) as the ad hoc routing protocol. Host mobility was modeled using the Random Waypoint model (Broch, Maltz, Johnson, Hu, & Jetcheva, 1998).

Performance Metrics

The performance metrics used in our simulation study were average data accessibility ratio (DAR), average query delay (AQD), and average network traffic overhead (NTO). We investigated the effects of cache size, pause time, number of clusters, and prefetching overhead on these performance metrics. The simulation parameters with their default values are summarized in Table 1. In the simulation, co-operative caching with prefetching, cooperative caching with no prefetching (CCNP), and simple caching schemes (NCC) were investigated at various parameter settings.

1. **Data accessibility ratio (DAR):** The data accessibility ratio is defined as the ratio between the total number of data item requests and the total number of successfully received data items.
2. **Average query delay (AQD):** The average query delay is the average

time interval between the generation of a query and the receipt of the query reply.

3. **The percentage network traffic overhead (NTO):** The percentage network traffic overhead is defined as the ratio of the increase in the total number of data transmitted with integrated caching and prefetching and the total number of data transmitted without prefetching. This metric can be used to measure the efficiency of prefetching.

There is a trade-off between hit ratio and traffic increase. Since each cached data item has an expiry time, the overhead is lower than a mechanism that stores all data for the entire duration until it is replaced. When fetched data can be used by multiple hosts, the increase in data prefetching decreases since data will not be prefetched multiple times. Hence, the hit ratio increases without an increase in network traffic. A good prefetching scheme should result in a high hit ratio without causing much additional traffic.

Discussion of Simulation Results

Figures 3-9 show the results of a performance evaluation from the experiments. The results are summarized in four categories.

1. **Simulation experiments on data accessibility:** The effects of cache size, pause time (PT) and the control parameter (δ) on data accessibility were investigated.

Figure 3 shows that the data accessibility ratio increases with an increase in cache size for all caching schemes. Cooperative caching with prefetching (CCPF) performs better at larger cache sizes due to the increased number of spaces available for caching after data fetching. The DAR includes within cluster data hits (local cache hit, neighbor cache hit, and remote cache hit). The results indicate improvements made by prefetching over cooperative caching (CCNP) and simple caching schemes (NCC).

Figure 4 shows that DAR increases proportionally to pause time. This occurs because the higher pause times indicate lower mobility. Thus, the CA will be relatively static, resulting in better data accessibility. On the other hand, the results show that cooperative caching with prefetching (CCPF) generally outperforms (by an approximate 20% increase in DAR) the scheme with only cooperative caching (CCNP). At higher pause times, data miss is only caused by a lack of the required data in the CA and not by mobility.

Figure 5 shows the effect of the combined frequency and latency control parameter (δ) for the LRFU cache replacement algorithm. The purpose of this experiment was to find a more suitable value for δ that would increase the performance of the cache replacement policy. The smaller value of

Table 1. Simulation parameters

Parameter	Default value/range
Network Area(m)	1500 × 1500
Network Size	200
Transmission Range(m)	250
Number of DS/cluster	3
Number of CA/cluster	3-5
Host Speed (m/s)	0-20
Pause Time (s)	200-2000
Cache Size(KB) in MHs	100 – 1800
Simulation Time(s)	2000

Figure 3. Data accessibility ratio cache size

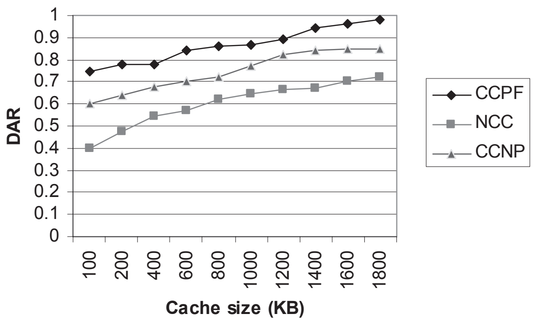


Figure 4. Data accessibility ratio as a function of pause time

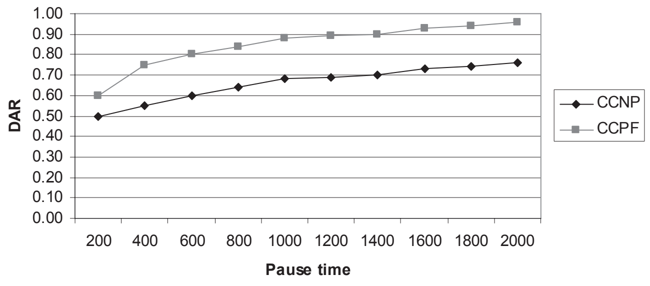


Figure 5. Data accessibility ratio as a function of cache size for different values of δ

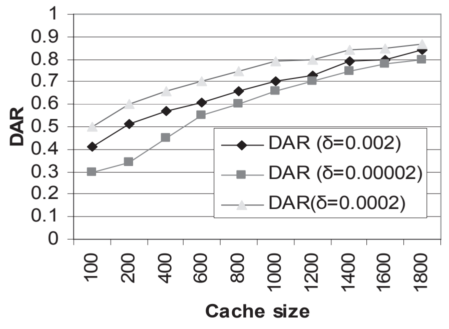
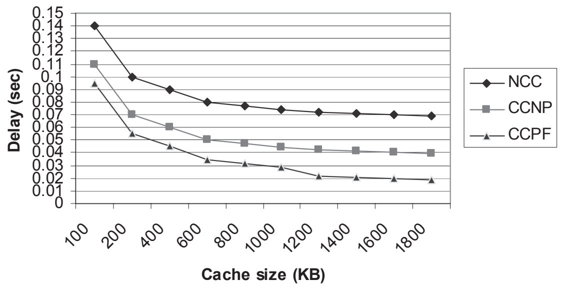


Figure 6. Average query delay as a function of cache size



δ (0.00002) was closer to LFU policy, whereas the larger value of δ (0.002) was closer to the LRU policy. The results in Figure 5 show that when the value of δ is 0.0002, the LRFU policy performs better than the other two cases at all cache sizes. The results also show that at a higher cache size, the rate of increase is low since there is sufficient space for caching and the cache replacement policy has less effect. This occurs because of a decreased cache miss due to the lack of prefetched data. As the value of δ increases, the data accessibility increases until it reaches a maximum level, beyond which the level of increase is negligible. The maximum value of data accessibility depends on both the cache size and the value of δ. Hence, the determination of a more appropriate value for δ depends on the cache size and weighing function.

2. **Simulation experiments on average query delay:** The effects of cache size and pause time on the average query delay were investigated.

Figure 6 shows that the CCPF consistently outperforms other schemes at all cache sizes. However, the difference in the delay is small at both ends of the cache size spectrum (lowest and highest). At lower cache sizes, both policies result in an equally sharp reduction of delay, while at higher cache sizes, the cache replacement policy has little effect since the available space can accommodate the existing or fetched data items. The difference between the query delays is relatively higher at larger cache sizes. This is because more data can be prefetched and cached when the cache size is large than when it is small. The results indicate improvements made by prefetching over cooperative caching and simple caching schemes.

Figure 7 shows that there is a decrease in average query delay associated with increased pause time for both schemes. However, cooperative caching with prefetching performed better than cooperative caching without prefetching. The schemes differ little at higher pause times (lower mobility) and indicate that the delay remains constant. In MANETs, the number of hops is closely related to the communication latency. Therefore, if more requests are fielded by a mobile host's own cache or by its immediate neighbors, then the query delay will be much shorter than if the request is fielded by a remote DS.

3. **Simulation experiments on network traffic:** Although prefetching increases data accessibility, it also introduces network traffic. The effects of prefetch thresholds on network traffic and data accessibility were investigated. The prefetch threshold shows the length of time required to perform prefetching. A threshold value of zero indicates no prefetching, while a threshold value of 1.0 indicates perfect prefetching. Prefetching was performed on the expiry of the TTL value, mobility of immediate neighbor holding the data, and on cache miss. The decision of which data item to fetch also depends on the prefetch index maintained at each host. Figures 8 and 9 show the NTO and DAR for various numbers of clusters (NC), respectively. In all experiments, the NTO was higher for larger numbers of clusters and lower for smaller numbers of clusters at all prefetch threshold levels. This is because data prefetching may involve transmission across multiple clusters, which are located a number of hops away. On the other hand, results in

Figure 7. Average query delay as a function of pause time

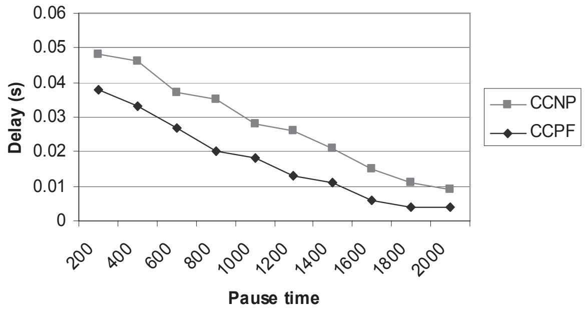


Figure 8. Average network traffic overhead as a function of prefetch threshold

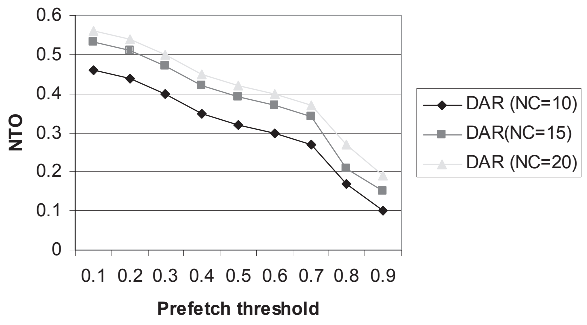


Figure 9. Average data accessibility ratio as a function of prefetch threshold

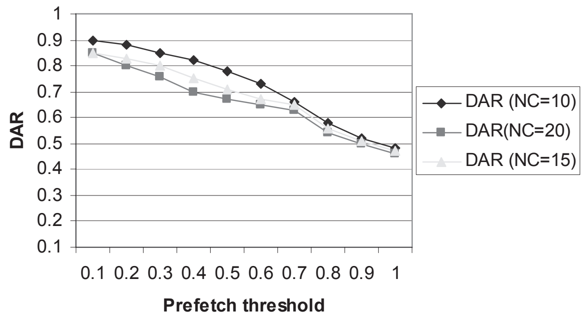


Table 2. DAR by source (internal and external clusters) of successful query and AVG for each caching scheme

Scheme	PT=1400		PT=1800		DAR
	LHR	GHR	LHR	GHR	AVG
CCNP	56%	14%	65%	12%	73.5%
CCPF	81%	9%	88%	7%	92.5%

Figure 9 show that DAR decreased with an increase in the prefetch threshold. However, the rate of decrease was similar at all clusters after the threshold of about 0.6-0.7. Threshold values above 0.7 do not exhibit any significant differences. The fact that higher DAR was observed at relatively lower numbers of clusters indicates that communication delay in inter-cluster communication was reduced, and that more caching agents can be located within the same cluster. These results imply that short prefetch threshold values result in higher data accessibility, but this occurs at the expense of higher network traffic. However, by choosing a suitable threshold value, a balance between both can be achieved. Our experiments indicate that a threshold value below 0.7 and above 0.1 was able to achieve an optimal balance between the data accessibility and the network traffic increase for various numbers of clusters.

4. **Simulation experiments on hit ratio by source:** To see the percentage of queries satisfied from local cluster and external clusters, we compiled separate statistics for intra-cluster and inter-cluster query responses.

We considered both cooperative caching with prefetching and without prefetching. The results are shown in Table 2. In this table, Local Hit Ratio (LHR) refers to the ratio of successful requests replied from local cache. Global Hit Ratio (GHR) refers to the number of requests satisfied from other clusters. This experiment was carried out using PT = 1400KB, PT = 1800KB and MH speed = 18 m/s. The last column of the table gives the average (AVG) data accessibility from both local and global sources.

The results show that with cooperative caching and prefetching (CCPF) schemes, less data items are fetched from external clusters to satisfy queries. The results also show that cooperative caching with prefetching ensures a greater data accessibility ratio than the scheme that uses only cooperative caching. In these simulation experiments, an approximate 25% gain (with PT=1400) and 23% gain (with PT = 1800) was achieved from intra-cluster information (local hit). Also about a 7% gain was observed when the PT value increases from 1400 to 1800. However, due to host mobility and wireless link characteristics, it is difficult to achieve 100% data accessibility in MANETs.

CONCLUSION

In this article, we proposed and evaluated a cooperative caching and prefetching scheme for MANETs. The architecture for enabling cooperative data caching and a prefetching algorithm were both presented. Additionally, a cache replacement policy based on combined metrics for data access frequency and reference time was also presented. A simulation based experimental study was carried out to evaluate the performance of the proposed scheme using average data accessibility, average query delay, and network traffic overhead. The results confirm that caching coupled with prefetching increases the data accessibility ratio and reduces query delay. In future work, we intend to enhance the proposed scheme by adapting it to the integrated ad hoc network and the Internet environment and evaluate performance in an implementation testbed.

ACKNOWLEDGMENTS

This research was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery Grant No. 046940.

The authors are grateful to the anonymous referees for their valuable suggestions, which improved the quality of the article.

REFERENCES

Alex, V. (1992). A good file system. In *Proceedings of USENIX File System Workshop* (pp. 1-12).

Artail, H., Safa, H., & Pierre, S. (2005). Database caching in MANETs based on separation of queries and responses. In *Proceedings of the IEEE International Conference on Wireless and Mobile Computing, Networking, and Communications* (pp. 237-244).

Broch, J., Maltz, D. A., Johnson, D. B., Hu, Y. C., & Jetcheva, J. (1998). A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proceedings of* ACM MobiCom (pp. 85-97).

Cao, G., Yin, L., & Das, C. R.(2004). Cooperative cache-based data access in ad hoc networks. *IEEE Computer Society, 37*(2), 32-39.

Das, S., Perkins, C., & Royer, E. (2000). Performance comparison of two on-demand routing protocols for ad hoc networks. In *Proceedings of IEEE INFOCOM* (pp. 3-12).

Fan, L., Cao, P., & Almeida, J. (1998). Summary cache: A scalable wide area web cache sharing protocol. In *Proceedings of ACM SIGCOMM*, ACM Press (pp. 254-265).

Feldmann, A., Caceres, R., Douglis, F., Glass, G., & Rabinovich, M. (1999). Performance of Web proxy caching in heterogeneous bandwidth environments. In *Proceedings of the 18th Conference of the IEEE Computer and Communications Society* (pp. 107-116).

Gerla, M., & Tsai, J. T. C. (1995). Multicluster, mobile, multimedia radio network. *Wireless Networks, 1*(3), 255-265.

Hara, T. (2002). Replica allocation in ad hoc networks with periodic data update. In *Proceedings of the 3rd International Conference on Mobile Data Management* (pp. 79-86).

Johnson, D., & Malta, D. (1996). Dynamic source routing in ad hoc wireless network. In *Mobile Computing*, edited by T. Imielinski & H. Korth, Kluwer Academic Publishers, Chapter 5, (pp. 153-181).

Lee, D., Choi, J., Choe, H., Noh, S. H., Min, S. L., & Cho, Y. (1997). Implementation and performance evaluation of the LRFU replacement policy. In *Proceedings of the 23rd Euromicro Conference* (pp. 106-111).

Lim, S., Lee, W. C., Cao, G., & Das, C. R. (2003). A novel caching scheme for internet based mobile ad hoc networks. In *Proceedings of the IEEE International Conference on Computer Communications and Networks*

(ICCCN), (pp. 38-43).

Lu, H., & Denko, M. K. (2004). Reliable data storage and dissemination in mobile ad hoc network. In *Proceedings of the International Workshop on Theoretical and Algorithmic Aspects of Wireless Ad Hoc, Sensor and Peer-to-Peer Networks* (pp. 81-86).

Lu, H., & Denko, M. K. (2005). Replica update strategies in mobile ad hoc networks. In *Proceedings of the 2nd IEEE/IFIP International Conference on Wireless and Optical Communications Networks* (WOCN 2005).

Network Simulator. (2005). Retrieved from http://www.isi.edu/nsnam/ns/

Papadopoui, M., & Schulzrinne, H. (2001). Effects of power conservation, wireless coverage and cooperation on data dissemination among mobile devices. In *Proceedings of ACM MobiHoc* (pp. 117-127).

Perkins, C., & Bhagwat, P. (1994). Highly Dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers. In *Proceedings of ACM SIGCOMM* (pp. 234-244).

Perkins, C. E., Belding-Royer, E. M., & Chakeres, I. (2003). Ad Hoc On Demand Distance Vector (AODV) routing. IETF Internet draft, draft-perkins-manet-aodvbis-00.txt, (Work in Progress).

Rousskov, A., & Wessels, D. (1999). Cache digests. *Computer Networks and ISDN Systems*, *30*(22-23), 2155-2168.

Sailhan, F., & Issarny, V. (2003). Cooperative caching in ad hoc networks. In *Proceedings of the 4th International Conference on Mobile Data Management* (pp. 13-28).

Wang, J. (1999). A survey of Web caching schemes for the Internet. *ACM SIGCOMM, Computer Communication Review*, *25*(9), 36-46.

Wang, Y. et al. (2005). A transparent cache based mechanism for mobile ad hoc networks. In *Proceedings of the 3rd International Conference on Inform Tech and Applications (ICITA'05)* (Vol. 2, pp. 305-310).

Wessels, D., & Claffy, K. (1998). ICP and the squid Web cache. *IEEE JSAC*, *16*(1998), 345-357.

Yin, L., & Cao, G. (2006). Supporting cooperative caching in ad hoc networks. *IEEE Transactions on Mobile Computing*, *5*(1), 77- 89.

Mieso K. Denko (denko@cis.uoguelph.ca) is an assistant professor in the Department of Computing and Information Science at the University of Guelph, Canada. He received his PhD and MSc in Computer Science from the University Natal, South Africa and the University of Wales, UK respectively. Dr. Denko has published in the areas of mobile ad hoc networking, integrated wired & wireless networks and wireless network security. His research interests include the design and evaluation of protocols for mobile ad hoc networks, mobile computing, hybrid wired and wireless networks, wireless ad hoc sensor networks and wireless mesh networks. Dr. Denko is a member of ACM, ACM SIGMOBILE and IEEE Communications Society and IFIP WG 6.9.