Efficient Cooperative Caching in Mobile Ad-hoc Networks

Po-Jen Chuang, Yu-Yiu Chen, and Hang-Li Chen

Department of Electrical Engineering, Tamkang University
Tamsui, New Taipei City, Taiwan 25137, R.O.C.
Email: pjchuang@ee.tku.edu.tw

Abstract. Cooperative caching can improve the accessibility of data objects in mobile ad hoc networks (MANETs). To lift up cache hit ratios and reduce data access latency in MANETs, this paper introduces a novel cooperative caching scheme, the Regionally Maintained Cooperative Caching (*RMCC*) scheme. *RMCC* allows one node in a region to cache a data item while the other nodes in the same region to cache the path to the node when acquiring the same item. By generating more cache space and wider cached data variety for nodes in one region, *RMCC* is shown through experimental evaluation to outperform related schemes in several parameters and achieve better system performance.

Keywords: Mobile ad-hoc networks (MANETs), cooperative caching schemes, cache hit ratios, access latency, experimental evaluation.

1. Introduction

Similar to content caching used by Internet Service Providers (ISPs) to accelerate web content acquirement, a wireless cache can temporarily store popular content which flows into an ISP's network. If the temporary storage can satisfy a subscriber's data request, we can avoid data transfer via expensive transit links and reduce network congestion. Caching data is also important and useful in MANETs where a node may communicate with others anytime anywhere. Cooperative caching can improve data accessibility and system performance by exercising cache cooperation between mobile nodes in MANETs. But, high node mobility, restricted battery energy and limited wireless bandwidth may decrease the cache hit ratios and increase access latency; or, nodes in a region may fail to cache more hot data when a cooperative caching scheme cannot effectively distribute varied data to caches in that region.

To improve the situation, this paper presents an advanced Regionally Maintained Cooperative Caching (*RMCC*) scheme. Built on certain features of existing caching schemes (such as *Zhao's* and *GroupCache* [1-2]), *RMCC* achieves better performance by specific designs of its own. It lets only one node in a region cache a data item while the other nodes in the same region cache the path to that node when pursuing the same item. Such a design can produce more cache space for nodes in the region to store more data and practically raise the cache hit ratios. To maintain cache validity of adjacent nodes, *RMCC* utilizes hello message broadcasting in on-demand routing for MANETs. It exchanges and maintains the cache status when any node receives a data

© SERSC 2013

reply message. For a data miss in the CacheData space, each node will search the item in its CachePath table before forwarding the request to the next node along the routing path to the server. Experimental evaluation results show that our *RMCC* scheme outperforms other caching mechanisms, including *SimpleCache* [1], *GroupCache* [2] and *Zhao's* [1], in cache hit ratios and amounts of file packets.

2. The Regionally Maintained Cooperative Cache (*RMCC*) Scheme

To enhance the cooperation of node caches in [1], we let nodes broadcast hello messages carrying node cache information. Nodes can cooperate to catch current cache distribution. When a node receives a file forwarding packet and the popularity degree of the file exceeds a predefined threshold, i.e., a *popular* file, it will add the file name, version and popularity degree into the next hello message and broadcast to neighbor nodes. Receiving the hello message, a neighbor node will check its cache space to see if it has the same file: if yes, delete it and cache the path only. By allowing only one node to cache the file and the other nodes in the same region to cache the path, we attain more cache space for storing other files and substantially increase the cache hit ratios.

2.1 The Flow of Our Cache Scheme

2.1.1 The task of each node: Our scheme adopts similar routing as the *AODV* scheme [3-4]. *AODV* uses neighbor tables to store the information of neighbor nodes. Our *RMCC* combines the cache space and neighbor tables. Besides caching data, our node also stores the cache paths to neighbor nodes in its neighbor table, to facilitate future requests for popular data. When a node receives a hello message with the cache information of a data item, it first searches the cache for the item. If it is in the cache: check its validity; if not, check the neighbor table for the corresponding cache path.

2.1.2 Node cooperation: When updating cached data, two nodes in a region, say A and B, may cache the same item. If so, compare the popularity degree of the item: If it is more popular in A, A will keep the item in its cache while B will add the cache path in its neighbor table and delete the item. A then includes the updated information in the next hello message and broadcast to neighbors. Receiving the message, each neighbor will delete this specific item from its cache and store the cache path in the neighbor table. Through such node cooperation, we increase the cache efficiency.

2.1.3 An example: Figure 1 illustrates the basic design of our scheme. Assume node 0 is a wireless node adjacent to a wired network, via which we can fetch a requested item from the network. When neighbors of node 0 repeatedly cache the same item, it will be a waste of the limited wireless resources. Previous schemes cannot avoid such resource waste. By employing the regionally maintained cooperative caching by way of hello messages, our scheme can. Suppose nodes 4 and 5 in Figure 1 will cache the same item, and node 4 is about to broadcast a hello message. It has the information of this item (including the time stamp and popularity degree), adds the information to the

hello message and broadcast out. (Combining cache information with hello messages can help maintain caches and also confirm the presence of neighbor nodes.) Receiving the hello message, node 5 will search its cache for this item: if having the item, delete it and cache the path only (save the cache space for other data). Later, when node 9 queries the same item (by the red line) via node 5, node 5 will attach the queried information (including the time stamp) for the server to check its validity in node 4. The server then sends a control message to node 4 (by the blue line), asking it to return the requested item (by the green line) to node 9. Thus completes the query.

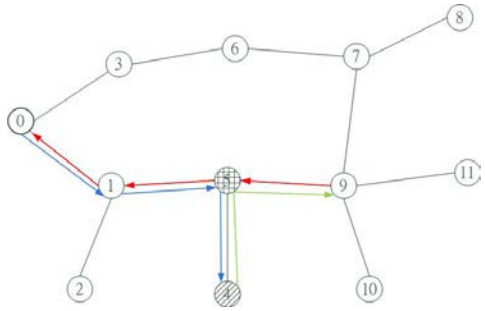

Fig. 1. An example to illustrate our scheme.

2.2 Data Placement and Replacement

In our *RMCC* scheme, when a packet containing a requested file reaches a node, the node will cache the file when cache space allows. If the cache is full with the file, the node adds 1 to the file's popularity degree; if it is full without the file, the node follows the LRU cache replacement mechanism in [5] to cache the file. We can also store cache paths via periodical hello message broadcasting in MANET routing (like *AODV* routing).

2.3 Data Discovery

In *RMCC*, when a node requests for a file, each node on the way of exploration to the server will check if the file is in the cache. If yes, add the file name, version and node name to the exploration packet; otherwise, look for cache path space and append related information to the packet. If a node caches no file or path, it will pass the exploration attempt to the next node which then repeats the same process until reaching the server. In data discovery, if an intermediate node caches the valid information of the requested file and directly returns it to the client, bandwidth resources can be significantly saved due to reduced control packet transmissions. If the requested file cached in intermediate nodes is invalid, the server will send the file to the client.

3. Experimental Evaluation

3.1 Simulation Parameters

Experimental evaluation using NS2 is conducted to check the performance of *RMCC*, *Zhao's*, *GroupCache* and *SimpleCache* (which caches the received data at the query node only). We set client/server models based on [5], exponentially distribute the query interval of nodes and slowly increase the query frequency. A new query takes place after a current query is served. Considering the actual trace of webpage packets, we let 10% of popular pages provide 80% of user requests (the Zipf-like distribution [6]). To simulate data updates in the server, we let the server update data every 5 seconds. The simulation area = 1500m*500m, with 50 nodes. The sizes for the client cache, server database and data item are, respectively, 800kb, 1000 items and 20kb. Note that *RMCC* uses *AODV* routing and its routing parameters: the hello message is broadcast per second; the transmission distance = 250 meters; the simulation area is rectangular – to reflect cache performance in long-distance transmissions.)

3.2 Simulation Results

Figure 2 gives *cache hit ratios* (the ratio of requested data being in the cache, *valid*) vs. *server update ratios*. *SimpleCache* which lets each node check the cache space for the requested item yields undesirable performance, and so does *GroupCache* which does not check validity from the server. *RMCC* yields the best hit ratios mainly because it uses hello messages to maintain caches regionally and to store cache paths.

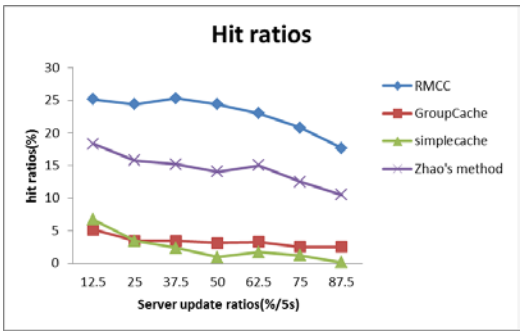

Fig. 2. Cache hit ratios vs. server update ratios.

Figure 3 gives *the amount of file packets* vs. *server update ratios*. The total of file packet transmissions (excluding control packets) indicates the percentage of file packet transmissions for these caching schemes over that for original *AODV* (no caching). The result, which reveals the performance of caching mechanisms, shows that *SimpleCache* and *GroupCache* save limited file packets. *GroupCache* requires even more file packets than original *AODV* especially at high server update ratios –

because it fetches data from the server upon cache miss. Among all, *RMCC* reduces the most file packet transmissions thanks to its regional inter-node cooperation policy.

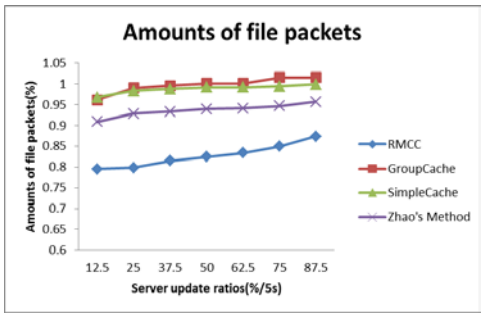

Fig. 3. The amounts of file packets vs. server update ratios.

4. Conclusions

To enhance caching efficiency in MANETs, we build a Regionally Maintained Cooperative Caching (*RMCC*) scheme based on a regional inter-node cooperation concept: A data will be cached in only one node of a region; when the other nodes in the same region need the item, they simply cache the path to that node. This design increases the cache space and variety of cached data, raises cache hit ratios and shortens data access latency. *RMCC* meanwhile uses hello message broadcasting to exchange/maintain cache statuses and maintain cache validity of adjacent nodes. Simulation results show that when compared with related schemes, *RMCC* is able to increase cache hit ratios and reduce file packet transmissions.

References

1. Zhao, J., Zhang, P., Cao, G., Das, C.R.: Cooperative Caching in Wireless P2P Networks: Design, Implementation, and Evaluation. IEEE Transactions on Parallel and Distributed Systems 21 (2), 229--241 (2010)
2. Ting, Y.-W., Chang, Y.-K.: A Novel Cooperative Caching Scheme for Wireless Ad Hoc Networks: GroupCaching. In: 2007 International Conference on Networking, Architecture, and Storage, pp. 62--68 (2007)
3. Perkins, C. E., Royer, E. M.: Ad hoc On-Demand Distance Vector Routing. In: 2nd IEEE Workshop on Mobile Computing Systems and Applications, pp. 90--100 (1999)
4. Abolhasan, M., Wysocki, T., Dutkiewicz, E.: A Review of Routing Protocols for Mobile Ad Hoc Networks. J. Elsevier Ad Hoc Networks 2 (1), 1--22 (2004)
5. Yin, L., Cao, G.: Supporting Cooperative Caching in Ad Hoc Networks. In: 23rd Annual Joint Conference of the IEEE Computer and Communications Societies, vol. 4, pp. 2537--2547 (2004)
6. Breslau, L., Cao, P., Fan, L., Phillips, G., Shenker, S.: Web Caching and Zipf-like Distributions: Evidence and Implications. In: 18th Annual Joint Conference of the IEEE Computer and Communications Societies, vol.1, pp. 126--134 (1999)