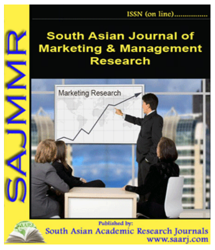
SAJMMR

ISSN: 2249-877X Vol. 11, Issue 10, October 2021, Impact Factor: SJIF 2021= 7.642

South Asian Journal of Marketing & Management Research (SAJMMR)

(Double Blind Refereed & Peer Reviewed International Journal)



DOI: 10.5958/2249-877X.2021.00092.8

COOPERATIVE CACHE MANAGEMENT PERFORMANCES WITHIN MANETS

Gulista Khan*

*Faculty of Engineering, Teerthanker Mahaveer University,
Moradabad, Uttar Pradesh, INDIA
Email id: gulista.engineering@gmail.com

ABSTRACT

Mobile Ad hoc Network are autonomously structured multi-hop wireless links in peer to peer fashion without aid of any infrastructure network. Due to lack of infrastructure support, each node in network act as router, coordinating to forward data packets to other nodes. Caching of frequently accessed data in ad hoc networks is a potential technique that can improve the data access, performance and availability. A cooperative cache-based data access framework lets mobile node cache the data or the path to the data to reduce query delays and improve data accessibility. Due to mobility and resource constraints of ad hoc networks, cooperative caching techniques designed for wired network may not be applicable to ad hoc networks. The objective of cooperative caching is to improve data availability and access efficiency by collaborating local resources of mobile devices. This paper reviews the various cooperative cache management techniques in the mobile ad-hoc networks.

KEYWORDS: *Mobile Adhoc Networks, Cooperative, Caching, Cache Resolution*

1. INTRODUCTION

Rapid progress in portable computer technologies allows MANET to be used in number of areas such as military application, industrial and commercial areas. Example of Ad hoc Network is a battlefield. Several Commanding Officers and group of soldiers form an Ad hoc Network. Each higher Officer have relatively powerful data center all Officers under them have to access data centers of higher officers to get various data needed by them. Soldiers under these lower rank Officers access data from stores of these Officer[1]. If one Soldier access some data, it may be possible that nearby soldiers share common operation and require same data sometimes later. Such scheme saves large amount of bandwidth, time and battery power. Mobile host cooperate with each other to forward data and mobile host have peer to peer connection among themselves.

There are several characteristics of Mobile Ad Hoc network. Firstly, Mobile devices are frequently disconnected due to mobility or the need to conserve power. Secondly, Devices employ multi-hop communication through unreliable links, which may cause long

Communication delay. Third, Broadcast in Mobile Ad Hoc Network is costly thus traditional cache consistent scheme are not suitable for these networks. Mobile Ad hoc networks are ideal in situations where installing an infrastructure is not possible because the infrastructure is too expensive or too vulnerable. This type of network can communicate with external networks such as Internet through a gateway [2].

1.1. Caching

Let consider a scenario in which mobile devices always retrieve data from the data center. This may result in a large amount of traffic in the MANET. This, apparently, is undesirable as traffic directed tothe data center consumes wireless bandwidth as well as power of mobile devices. In addition, a mobile host suffers from high access latency if it is distant from the data center, and packet loss probability for long-distance data access is high[3]. Furthermore, traffic near the data center will be heavy, and this leads to a potential performance bottleneck. These problems are more pronounced when the network size is large, which results in poor scalability of the system. The above observations motivate researchers to investigate data caching techniques for MANETs.

With data cached in mobile nodes, a data request may be satisfied by a nearby caching site, instead of being serviced by the data center. In many applications, mobile nodes in a MANET share common interests. In this scenario, sharing cache contents between mobile nodes offers significant benefits. Typically, nodes cache data items for serving their own needs[4]. Cache sharing, however, allows geographically neighboring mobile nodes to access each other's cache contents. By doing so, the number of long-distance data accesses to the data center can be reduced. The key to this technique is that a node has to know if there is some node in its vicinity that has cached the data it requires and where it is, if any. One approach to deal with this requirement is to let a mobile node record the caching information about a nearby node while forwarding the data requested by the node.

The caching information can subsequently be used to direct requests for the same data to the caching site. If mobile users around infestations, which have limited coverage, form an ad hoc network, a mobile user who moves out of the range of a particular infestation can still access the data it contains. If one of the nodes along the path to the data source has a cached copy of the requested data, it can forward the data to the mobile user, saving bandwidth and power. Thus, if mobile nodes can work as request forwarding routers, they can save bandwidth and power and reduce delays. Since MANETs are mobile and constrained by limited energy, bandwidth, and computation power, which is a big concern when designing protocols for such networks.

1.2.Cooperative Caching in Mobile Adhoc Network

As we have seen that cooperative caching is helpful to reduce the use of network bandwidth and access time to retrieve the data from the data center. Many researchers provide various techniques in order to retrieve the data more efficiently. Some of the techniques are described here.

1.2.1. Push and Pull Approach

The two basic types of cache sharing techniques are push based and pull based. With push-based cache sharing, when a node acquires and caches a new data item, it actively advertises the caching event to the nodes in its neighborhood. Mobile nodes in the vicinity will record the caching information upon receiving such an advertisement and use it to direct subsequent requests for the same item. This scheme enhances the usefulness of the cached contents. The cost we have to pay is the communication overhead for the advertisement; an advertisement is useless if no demands for the cached item arise in the neighborhood. In the push-based scheme, the caching information known to a node may become obsolete due to node mobility or cache replacement. The pull-based approach may overcome this problem.

With pull based cache sharing, when a mobile node wants to access a data item that is not cached locally it will broadcast a request to the nodes in its vicinity. A nearby node that has cached the data will send a copy of the data to the request originator (a pull operation)[5]. Unlike pushing, pulling allows the node to utilize the latest cache contents. However, in contrast with the pushing technique, the pulling scheme has two drawbacks:

- In case the requested data item is not cached by any node in the vicinity, the requester node will wait for the time-out interval to expire before it proceeds to send another request to the data center. This will cause extra access latency, and the pulling effort is in vain.
- Pulling resorts to broadcast to locate a cached copy of an item. In addition, more than one copy will be returned to the request originator if multiple nodes in the neighborhood cache the needed data. This introduces extra communication overhead[6]. Another issue of concern is the limited cache space that is available in a mobile node. Hence, a cache replacement mechanism must be in place for evicting data items from the cache to make room for a newly acquired one, when the cache is full. Since cache contents of a node are shared by other nodes, a good cache replacement policy should take into consideration the access demands from the entire neighborhood.

1.3. Cache Resolution

Cache resolution addresses how to resolve a data request with minimal cost of time, energy, and bandwidth. In cooperative caching, the emphasis of cache resolution is to answer how nodes can help each other in resolving data requests to improve the average performance. In COOP the authors give three cache resolution schemes: 1. Hop-by-hop cache resolution 2. Zone-based cache resolution 3. The cocktail resolution scheme 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 is referred to as Simple Cache[7]. This scheme 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 reduce the cost in terms of increased data access latency and increased 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 caches an unexpired copy of the requested data, it can send a reply to the requester and stop forwarding the data request. The second approach is zone-based cache resolution. This scheme is the extension of the hop by-hop resolution scheme. If a forwarding node does not have the data locally but it knows a closer data source (e.g. by proactive data discovery in its cooperation zone), it can also redirect the request to the closer data source, which also reduces the travel distance of data messages and hence minimizes the energy cost and response delay. COOP uses a cocktail approach based on the basic approaches described above. COOP uses profile based resolution after the local cache misses[8]. If no matching cache is found or the request fails, COOP uses reactive approach to discover the data in its cooperation zone. If this again fails, COOP forwards the data request to the data server, and hop-by-hop resolution is used to resolve the request along the forwarding path.

1.4. *Cache Management*

For cooperative caching, the emphasis of cache management is how to manage an individual cache not only from the local node's point of view, but also from the view of the overall cooperative caching system. To maximize the capacity of cooperative caches, COOP tries to reduce duplicated caching within the cooperation zone, such that the cache space can be used to accommodate more distinct data items. In this paper the authors categorize cached data copies based on whether they are already available in the cooperation zone or not. A data copy is primary if there is no other primary copy within the zone. Otherwise, the data copy is secondary. To decide caching priorities of primary and secondary data the inter- and intra-category rules are used.

- The Inter Category Rule: The idea of inter-category rule is to put primary items at a priority level, i.e. secondary items are purged to accommodate primary items, but not vice versa. If the primary copy holder is beyond the zone radius, the new copy is primary copy; otherwise, the new copy is a secondary copy.
- The Intra Category Rule: The intra-category rule is used to evaluate the data items within the same category. For this purpose, here the authors simply adopt the LRU (least recently used) algorithm.

1.5. *Limitation of COOP*

To improve data availability and access performance, COOP addresses two basic problems of cooperative caching. For cache resolution, COOP uses the cocktail approach which consists of two basic schemes: hop-by-hop resolution and zone-based resolution. By using this approach, COOP discovers data sources which have less communication cost. For cache management, COOP uses the inter- and intra-category rules to minimize caching duplications between the nodes within a same cooperation zone and this improves the overall capacity of cooperated caches[9]. The disadvantage of the scheme is that flooding incurs high discovery overhead and it does not consider factors such as size and consistency during replacement.

1.6.Cache Data and Cache Path

In Cache Data, if a node finds many requests for a particular data item d then data item is cached by the node. For example, in figure 1 both node B and node C request d through node A, node A knows that d is popular and cache it locally. Future request by node D can be served by node A. Suppose the data center receives several requests for d forwarded by node F. Nodes along the path F-C-A may all think that d is a popular item and should be cached. However, it wastes a large amount of cache space if three of them all cache d[10]. To avoid this, authors proposed a conservative rule. That states: A node does not cache the data if all requests for the data are from the same node. As in the previous example, all requests received by node Fare from node C, which in turn are from node A. With the new rule, node C and node A do not cache d. If the requests received by node A are from different nodes such as node C and node D, node A will cache the data.

1.7.Hybrid Cache

In Hybrid Cache, 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-live (TTL) of the item. For a data item d, the following heuristics are used to decide whether to cache data or path: 1. If size of d is small, Cache Data should be adopted because the data item only needs a very small part of the cache; otherwise, Cache Path should be adopted to save cache space.

1.8.Limitations of Cache Data and Cache Path

As we seen in Cache Data, 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 for this approach is that the data could take a lot of caching space in forwarding nodes. In Hybrid Cache, 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-live (TTL) of the item. Because due to the mobility of nodes the collected statistics about the popular data may become useless. One another drawback of these schemes is that if the node does not lie on the forwarding path of a request to the data center the caching information of a node cannot be shared.

1.9. IXP and DPIP Protocols

Two cooperative caching schemes IXP and DPIP. Index Push (IXP) is push based in the sense that a mobile node broadcasts an index packet in its zone to advertise a caching event. The Data Pull/Index Push (DPIP) is a pull based one. DPIP is offers an implicit index push property by exploiting in-zone request broadcasts.

1.10. The IXP Protocol

The idea of IXP is based on having each node share its cache contents with the nodes in its zone. To facilitate exposition, authors call the nodes in the zone of a node M the buddies of M. A node should make its cache contents known to its buddies, and likewise, its buddies should reveal their contents to the node. IXP requires that, whenever a node caches a data item, it broadcasts an index packet to its buddies to advertise the caching event. Each node maintains an index vector, denoted as IV. An IV has N elements, where N is the number of data items in the data set. Each element of IV corresponds to a different data item and consists of three entries that are used to record caching information of the corresponding item.

1.11. The DPIP Protocol

IXP is essentially push based in the sense that a caching node “advertises” the caching information to the surrounding buddies. Each node has a view of the caching status in its zone only. However, due to node mobility and some limitations of mobile devices such as transient disconnections, the caching status represented by IV may become obsolete or not up-to-date. For example, suppose that, according to M’s IV, none of M’s buddie’s caches x. If a new node that has cached x moves into M’s zone, the cache status cannot be captured by M’s IV with IXP. In the following, we propose a more sophisticated protocol, called DPIP, to deal with this problem. DPIP is basically a pull-based protocol. However, it also exploits an implicit index push property. We now describe the details of DPIP. Similar to IXP, each node maintains an 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].

2. DISCUSSION

MANETs are limited by intermittent network connections, restricted power supplies, and limited computing resources. These restrictions raise several new challenges for data access applications with the respects of data availability and access efficiency. In ad hoc networks, mobile nodes communicate with each other using multihop wireless links. Due to a lack of infrastructure support, each node acts as a router, forwarding data packets for other nodes. Most previous research in ad hoc networks focused on the development of dynamic routing protocols that can efficiently find routes between two communicating nodes.

Although routing is an important issue, but the ultimate goal of ad hoc networks is to provide mobile nodes with access to information. In ad hoc networks, due to frequent network partition, data availability is lower than that in traditional wired networks. This problem can be solved by caching data items on mobile hosts. However, the movement of nodes, limited storage space and frequent disconnections limit the availability. By the caching of frequently accessed data in ad hoc networks we can improve the data access, performance and availability. A data management in adhoc network that is based on cooperative caching data access framework lets mobile node to cache the data or the path to the data to reduce query latency and improve data accessibility. Due to mobility and resource constraints of ad hoc networks, caching techniques designed for wired network may not be applicable to ad hoc networks.

3. CONCLUSION

In this paper we have discussed cache sharing issues related to mobile adhoc network environment and give analysis of some popular cooperative caching schemes. These caching schemes are useful in MANET environment. Here we present how these schemes are advantageous in order to find a data item in a MANET by using less resources (e.g. network bandwidth, energy etc.) and improves the performance (data availability and latency time). We also discussed the limitations of these techniques. As the cooperative caching is a useful technique to improve the data availability in the MANET so these analyses will be helpful for the future research.

REFERENCES:

1. E. Nordström, P. Gunningberg, and C. Tschudin, "Robust and flexible Internet connectivity for mobile ad hoc networks," *Ad Hoc Networks*, vol. 9, no. 1, pp. 1–15, 2011, doi: 10.1016/j.adhoc.2010.04.003.

2. Y. Sun, E. M. Belding-Royer, and C. E. Perkins, "Internet Connectivity for Ad hoc Mobile Networks," *Int. J. Wirel. Inf. Networks*, vol. 9, no. 2, pp. 75–88, 2002, doi: 10.1023/A:1015399632291.

3. H. Jin, D. Xu, C. Zhao, and D. Liang, "Information-centric mobile caching network frameworks and caching optimization: a survey," *Eurasip Journal on Wireless Communications and Networking*. 2017, doi: 10.1186/s13638-017-0806-6.

4. P. M. Ruiz, F. J. Ros, and A. Gomez-Skarmeta, "Internet connectivity for mobile ad hoc networks: Solutions and challenges," *IEEE Commun. Mag.*, vol. 43, no. 10, pp. 118–125, 2005, doi: 10.1109/MCOM.2005.1522134.

5. J. Hoebeke, I. Moerman, B. Dhoedt, and P. Demeester, "An overview of mobile ad hoc networks: Applications and challenges," *Journal of the Communications Network*, vol. 3, no. 3. pp. 60–66, 2004.

6. G. M. Chiu and C. R. Young, "Exploiting in-zone broadcasts for cache sharing in mobile ad hoc networks," *IEEE Trans. Mob. Comput.*, vol. 8, no. 3, pp. 384–397, 2009, doi: 10.1109/TMC.2008.127.

7. L. Yin and G. Cao, "Supporting cooperative caching in ad hoc networks," *IEEE Trans. Mob. Comput.*, vol. 5, no. 1, pp. 77–89, 2006, doi: 10.1109/TMC.2006.15.

8. S. Glass, I. Mahgoub, and M. Rathod, "Leveraging MANET-Based Cooperative Cache

Discovery Techniques in VANETs: A Survey and Analysis," *IEEE Communications Surveys and Tutorials*, vol. 19, no. 4. pp. 2640–2661, 2017, doi: 10.1109/COMST.2017.2707926.

9. J. Al-Badarneh, Y. Jararweh, M. Al-Ayyoub, R. Fontes, M. Al-Smadi, and C. Rothenberg, "Cooperative mobile edge computing system for VANET-based software-defined content delivery," *Comput. Electr. Eng.*, vol. 71, pp. 388–397, 2018, doi: 10.1016/j.compeleceng.2018.07.021.

10. A. Mayank and C. V. Ravishankar, "Supporting mobile device communications in the presence of broadcast servers," in *International Journal of Sensor Networks*, 2007, vol. 2, no. 1–2, pp. 9–16, doi: 10.1504/IJSNET.2007.012977.