International Journal of Computer Science and Informatics

Volume 2 | Issue 3 Article 5

January 2013

Cross-Layer System for Cluster Based Data Access in MANET'S

Anand Nayyar
Department of Computer Applications & IT, KCL Institute of Management and Technology, Jalandhar,
anand_nayyar@yahoo.co.in

Cross-Layer System for Cluster Based Data Access in MANET'S

Anand Nayyar

Department of Computer Applications & IT, KCL Institute of Management and Technology, Jalandhar
E-mail : anand_nayyar@yahoo.co.in

Abstract **-** The objective of cooperative caching is to improve data availability, improve access efficiency and reduce query delay in mobile Ad-Hoc networks. Many types of cache replacement algorithms like LRU, LFU, LFRU, LRU-MIN and LFU-MIN are used to improve data accessibility and reduce query delay in cluster based cooperative caching in Mobile Ad-Hoc networks. But they have some limitations such as accessing remote information station via multi hop communication leads to longer query latency and causes high energy consumption, many clients frequently access the database server they cause a high load on the server and reduce the server response time .Multi hop communication causes the network capacity degrades when network partition occurs. The Research Paper gives an overview of Cooperative Cache Management Techniques and caching policies and propose a new algorithm can be regarded as a LRFU-MIN (least recently frequently used information with minimal number of page replacements). It discover a data source which induces less communication cost of moving cache blocks into the most recently frequently used position and minimizes caching duplications between neighbour nodes. In this paper we utilize a cross-layer design approach to improve the performance of combined cooperative caching and prefetching schemes. The paper examines the performance using NS-2 simulation environments. The proposed LRFU-MIN enhances the performance of cross-layer cluster based cooperative caching in mobile Ad-Hoc networks when compared with LRU and LFU-MIN.

Index Terms—*Adhoc networks, cooperative caching, cluster, Data caching, information search*

I. INTRODUCTION ON PROBLEM OF DATA ACCESS IN MANET

Mobile Ad hoc Network are autonomously structured multi-hop wireless links in peer to peer fashion without aid of any predefined network infrastructure. Due to lack of infrastructure support, each node in network act as router, coordinating to forward data packets to other nodes Mobile devices are frequently disconnected due to mobility or the need to conserve power. Second, 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 network. Four, accessing remote information station via multi hop communication leads to longer query latency and causes high energy consumption. Five, when many clients frequently access the database server they cause a high load on the server and reduce the server response time. Six, multi hop communication causes the network capacity degrades when network partition occurs. This problem can be solved by caching data items on mobile hosts. 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. In this paper we propose a LRFU-MIN cache replacement algorithm for cross- layer cluster based cooperative caching (CCBCC) in MANETs. The rest of the paper organized as follows: section 2 describes the related works on cooperative caching. Section 3 describes the system architecture on cross-layer cluster based cooperative caching (CCBCC). Section 4 describes the CCBCC approach using LRFU-MIN. Section 5 describes the performance evaluation of the proposed approach and section 6 concludes the paper and suggests possible future work.

II. BACKROUND AND RELATED WORKS ON COOPERATIVE CACHING

Cooperative caching is helpful to reduce the use of network bandwidth and access time to retrieve the data from the data centre. 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 neighbourhood. 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 the cooperation of caching nodes is twofold. First, a caching node can answer the data requests from other nodes. Second, a caching node stores the data not only on behalf of its own needs, but also based on other nodes' needs.

COOP addresses two basic problems for cooperative caching in MANETs: 1.Cache resolution – how does a mobile device decide where to fetch a data item requested by the user? 2. Cache management – how does a mobile device decide which data item to place/purge in its local cache? For cache resolution, COOP tries to discover a data source which induces less communication cost by utilizing historical profiles and forwarding nodes. For cache management, COOP minimizes caching duplications between neighbour nodes and allows cooperative caches to store more distinctive data items to improve the overall performance. Cache resolution schemes:

1. Hop-by-hop cache resolution,
2. Zone-based cache resolution,
3. The cocktail resolution scheme.

In [3, 4] Yin and Cao propose three schemes: CachePath, CacheData, and HybridCache. In CacheData, intermediate nodes cache the data to serve future requests instead of fetching data from the data centre. In CachePath, mobile nodes cache the data path and use it to redirect future requests to the nearby node which has the data instead of the faraway data centre. To further improve the performance, we design a hybrid approach (HybridCache), which can further improve the performance by taking advantage of CacheData and CachePath while avoiding their weaknesses.

In [4-8] new cache replacement algorithms are used to make the best use of cache space. But the used traditional replacement algorithms like LRU, LFU, and LRFU have problems. However caching alone is not sufficient to guarantee high data accessibility and low communication latency in dynamic system. To overcome these draw backs, a new approach is proposed in cluster based cross layer design for cooperative caching in MANETs [12]. In the above proposal [12] for cache replacement mechanism they used LRU-MIN as the cache replacement algorithm. But the used LRU-MIN has certain limitations. Among those first, it prefers only small objects to raise the hit ratio. Second, it doesn't exploit the recent frequency information of memory accesses. Third, the overhead cost of moving cache blocks into the most recently frequently used position each time when a cache block is accessed. In this paper we propose a cache replacement algorithm called LRFU-MIN which makes use of recent frequent information with minimal number of page replacements for evicting the objects from the cache. The proposed cache replacement algorithm further enhances the performance of cross-layer cluster based cooperative caching (CCBCC) in MANETs.

III. SYSTEM ARCHITECTURE ON CROSS-LAYER CLUSTER BASED COOPERATIVE CACHING (CCBCC)

CCBCC is a cluster-based middleware which stays on top of the underlying network stack and provides caching and other data management services to the upper layer applications in MANETs environment .In [14], the instances of CCBCC run in each mobile host. CCBCC consists of clustering, stack profile, information search, cache management and prefetching modules. The stack profile module provides cross-layer information exchange. Using the stack profile, cached items can be fetched by the network layer. The network traffic information which is in the data link layer can be retrieved by the middleware layer for Prefetching purposes.

Application layer: It is responsible for providing an interface for users to interact with application services or networking services. Application layer uses HTTP, FTP and TELNET etc.

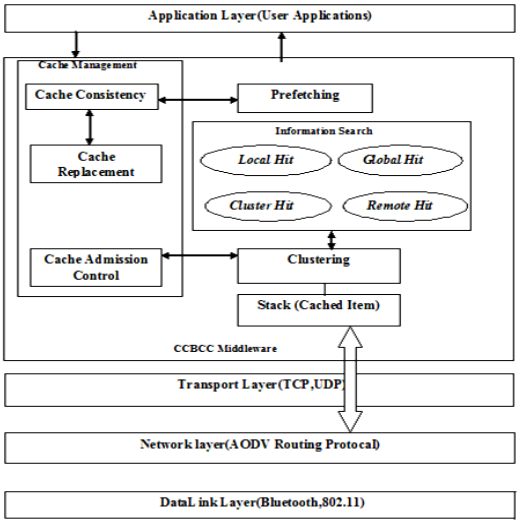

Fig.1: CCBCC Architecture

Middleware layer: It is responsible for service location, group communications shared memory. Middleware layer consists of various blocks such as cache management, information search, Prefetching and clustering.

*Cache admission control***:** In this, a node will cache all received data items until its cache space full. After the cache space becomes full, the received data item will not be cached if the data item has a copy within the cluster

Cache replacement: The various cache replacement algorithms used for this mechanism are LRU, LFU-MIN etc.

*Cache consistency***:** The cache consistency strategy keeps the cached data items synchronized with the original data items in the data source.

Information search: Deals with locating and fetching the data item requested by the client.

Prefetching: Responsible for determining the data item to be prefetched from the Data Centre for future use.

Transport layer: It is responsible for providing data delivery transportation between the applications in the network by using the protocols like TCP, UDP. It includes the functionalities like Identifying the services, segmentation, sequencing and reassembling and error correction.

Network layer: It is responsible for providing logical addressing and path determination (routing).The routing protocols such as AODV, DSR, DYMO, etc. are responsible for performing path determination (routing). The current system architecture uses AODV protocol for path determination.

Data link Layer: Provides apparent network services so that network layer can be ignorant about the network topology and provides access to physical networking media. It includes error checking and flow control mechanism.

Cluster Formation

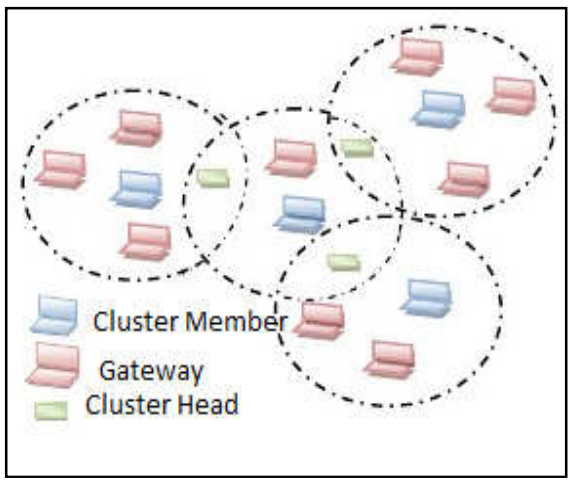

Fig. 2 Clustering Formation

Clustering is a method used to partition the network into several virtual groups. For the cluster formation we use least cluster change algorithm [12] which is an improvement of lowest ID algorithm. Each mobile node has a unique id. The node which has least id in the group is elected as a cluster head. In [14], Cluster head maintains a list which maintains the information of all other nodes in the group. In a cluster, the number of hops between any two nodes is not more than two. Cluster member is just like a mobile node it does not have any extra functionality. The node which is common to two cluster heads is elected as a gateway.

Gateway is used for providing the communication between two cluster heads. Whenever a node requests for the data, first it has to be checked in the cluster head list. If it is not available in the list of cluster head then the cluster head forwards the requested data item to the other cluster via gateway. By using LCC we can reduce the frequent changes of cluster head formation. LCC adopts LID to create clusters. If a cluster member moves out of the cluster it won't affect the existing clustering architecture. If two cluster heads exist within the cluster, the lowest id mobile node is elected as a cluster head and if more number of nodes moves out of the cluster will form a new cluster.

Information Search Operation: Information search operation [12], mainly deals with locating and fetching the data item requested by the client from the cache. This Information search includes 4 cases.

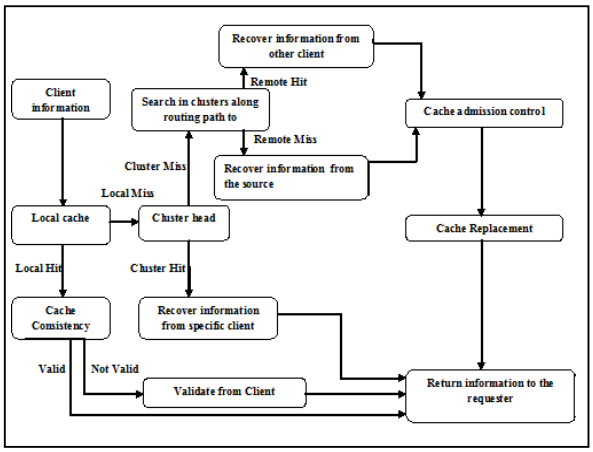

Fig.3 Information Search Operation

Case 1: Local hit: When copy of the requested information item is ordered inside the hard disk of the requester, the information item is recovered to serve the query and no cooperation is necessary.

Case2:Cluster hit: When the requested information item is stored in a client within the cluster of the requester, the requester sends a request to the Cluster head

and the Cluster head returns the address of a client that has cached the information item.

Case3: Remote hit: When the information is found with a client belonging to a cluster, other than home cluster of the requester, along the routing path to the data source.

Case 4: Global hit: Information item is recovered from the server. When the client information request comes to the mobile node, first it checks in the local hard disk of mobile node i.e. local cache of mobile node. If it is available in the local cache it sends back the reply to the client. Otherwise the request is forwarded to the neighbours based on the cache current state information in the cluster head. If the cluster head has the requested cache state information cluster head gives back to the requester by giving the cluster member id. If it is not available within the cluster then the request is forwarded to the other cluster through gateway.

The request is processed the same way and sends back the reply to the requester. Otherwise the request is reached to the data centre, the data centre processes the information request and sends backs the requested information to the client via multi hop communication then the client uses the cache admission control for the consistency check in the cluster. If the same data is available within the cluster, then it won't cache the objects information. If it is not available it will cache the information objects and sends back the cached information to the cluster head for updating in the cluster cache state.

IV. CCBCC APPROACH USING LRFU-MIN

LRFU-MIN uses a technique called least recently frequently used information with minimal number of page replacements. Our alteration is that a node will cache all received information item until its cache space becomes full. LRFU-MIN is an advancement of LRFU replacement algorithm. LRFU-MIN repetitively removes the least recently frequently referenced data items from the cache space until there is sufficient space for the newly arrived item.

Algorithm: LRFU-MIN

Begin

1.Check the cache space is sufficient or not
2.If there is sufficient cache space then cache the information into cache space
3.Else Call LRFU-MIN algorithm for cache replacement process
4.Ensure the size of newly arrived information item
5.Remove the obsolete information item
7.Insert newly arrived information item into cache space
8.If there is still no sufficient cache space after all obsolete items are removed
9.Find the recently frequently referred information items in the cache space
10.Eliminate least recently frequently used item from cache space
11.Insert newly arrived information items into cache space until it becomes full

End

V. PERFORMANCE EVALAUATION OF THE PROPOSED APPROACH

5.1 *Simulation Environment*

The proposed approach was evaluated in an NS-2 simulation environment [10]. In the simulation, we have evaluated the performance of LRU, LFU-MIN, LRFU-MIN cache replacement algorithms in CCBCC approach. The simulation was carried out in a grid of 3500 m x 500 m with 40 to 80 nodes. The time interval between two consecutive queries generated from each node/client follows an exponential distribution with mean node query delay Tq is taken as 4 sec. The node density can be selected by selecting the number of nodes; here we considered the number of nodes as 60by default.

The bandwidth selected for the transmission is 2mbps and the total transmission range of 200m is considered. A single stream of read only queries can be generated by every client. After a query is sent out, the client does not generate new query until the pending query is served. Each client generates accesses to the data items following Zipf distribution [13] with a skewness parameter (θ) 0.7. If θ = 0, clients uniformly access the data items. The AODV routing protocol was used in the simulation. The nodes/clients moves in the simulation area according to the random waypoint mobility model. The Zipf-like parameter [13] can be expressed as

$$\frac{P}{N}(i) = \frac{\Omega}{i^{\theta}} \qquad - \quad (1)$$

Where

$$\Omega = \left(\sum_{i=1}^{N} \frac{1}{i^{\theta}}\right)^{-1} \qquad 0 \leq \theta \leq 1$$

Here N is the total number of data items. And θ is the skewness parameter.

5.2 Performance *Metrics*

The performance metrics is used to determine the effectiveness of access time. It depends on the Hit ratio (HR) and average frequency ratio (AFR) at successive levels.

Hit ratio: successive hit ratios are independent random variables with values between 0 and 1.It is defined as the ratio of number of successive requests to the total number of access frequency requests.

$$Hit\ ratio = \frac{successive\ requests}{total\ number\ of\ access\ requests} \qquad (2)$$

Average frequency ratio: The corresponding temptation for accessing frequent data is to concentrate on miss rate because

it is independent of the speed.

$$Average\ frequency\ ratio = \frac{\sum_i (Hit\ ratio - Miss\ rate)}{\sum_i Access\ requests} \quad (3)$$

5.3 *Comparison of Cache Performance*

We compared the performance of LRU, LFU-MIN and LRFU-MIN.LRFU-MIN improves the average frequency ratio (AFR) on average by 34.13 percent compared with LRU and 10.57 percent compared with LFU-MIN. And it improves the cache hit ratio performance by 42.60 percent over LRU, and 10.05 percent over LFU-MIN. LRFU-MIN also improves the consistency of the cached documents in addition to improving performance of Cache. The performance evaluations of various parameters are plotted by using the graphical representation. In Fig. 4, X-axis represents cache size and Y-axis represents AFR and for fig. 5, X-axis represents cache size and Y-axis represents Hit-Ratio.

Table. 1. Obtained computational values

	LRU		LFU-MIN		LRFU-MIN	
Cache Size(KB)	**AFR**	**HR**	**AFR**	**HR**	**AFR**	**HR**
60	0.12	0.07	0.23	0.13	0.26	0.16
90	0.17	0.10	0.26	0.17	0.29	0.20
140	0.21	0.14	0.28	0.23	0.34	0.25
260	0.25	0.18	0.34	0.27	0.37	0.28
300	0.29	0.21	0.36	0.32	0.39	0.34
350	0.33	0.27	0.39	0.40	0.43	0.46

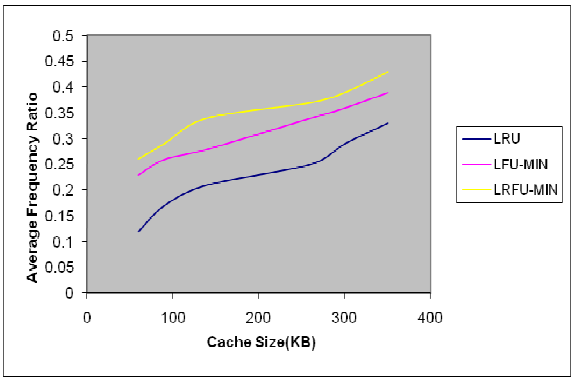

Fig. 4. Performance comparison of AFR

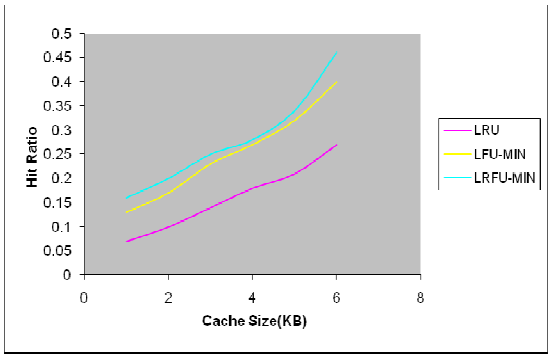

Fig. 5. Performance comparison of HR

VI. CONCLUSION

In this paper, we proposed a new cache replacement algorithm (LRFU-MIN) which works on the principle of least recently frequently used information with minimal number of page replacements. The proposed new cache replacement algorithm is compared with other cache replacement algorithms using simulation modelling (NS-2). The Simulation results shows that the Proposed LRFU-MIN cache replacement algorithm enhances the performance of cross-layer cluster based cooperative caching (CCBCC) in MANETs by improving the cache hit ratio and by reducing the Client query response time while accessing the data items from the cache memory as compared with LRU and LFU-MIN cache replacement algorithms. For future work there is a need to find out an efficient Prefetching technique which further improves the data accessibility and reduce query delay to compliment the cooperative caching scheme.

REFERENCES

[1] M. K. Denko and J. Tian, "Cross-layer design for cooperative caching in Mobile adhocnetworks," in Proc.5th IEEE, Consumer Communications and Networking Conf. (CCNC), 2008, pp. 375–380.

[2] G. Cao, L. Yin and C.R. Das. "Cooperative cache-based data access in adhoc networks," IEEE Computer Society, vol.37, 2004, pp.32-39.

[3] L. Yin and G. Cao, "Supporting Cooperative Caching in Ad Hoc Networks," Proc. IEEE INFOCOM '04, pp. 2537-2547, 2004.

[4] L. Yin and G. Cao, "Supporting Cooperative Caching in Ad Hoc Networks," IEEE Trans. Mobile Computing, vol. 5, no. 1, pp. 77-89, Jan. 2006.

[5] J.Zhao, P.Zhang and G.Cao, "On cooperative caching in wireless P2P networks", inProc.28thInt.Conf.DistributedComputingSystems(ICDCS), 2008.

[6] N.Chand,R.C.Joshi,andM.Misra,"Cooperative caching strategy in mobile ad hoc networks based on clusters,WirelessPerson.Commun" pp. 41-63, Dec. 2006.

[7] H.Artail, H.Safa, K.Mershad, Z.Abou-Atme, and N.Sulieman, "COACS: A cooperative and adaptive caching system for MANETs,"IEEE Trans. Mobile Computing., vol. 7, no. 8, pp. 961-977, Aug. 2008.

[8] J. Tian and M. K. Denko, "Exploiting clustering and crosslayer design approaches for data caching in MANETs," in Proc. 3rd IEEE Int. Conf. Wireless and Mobile Computing, Networking and Communications, (WiMob), 2007, p. 52.

[9] HuapingShen, Sajal K. Das, Mohan Kumar and Zhijun Wang, "Cooperative Caching with Optimal Radius in Hybrid Wireless Networks," NETWORKING, pp. 841-853, 2004.

[10] NetworkSimulator2 [Online]. Available: http:// www.isi.edu/ nsnam/ns/

[11] J. Li, C. Blake, D. S. J. D. Couto, H. I. Lee, and R. Morris, "Capacity of ad hoc wireless networks," in Proc. 7th Annu. Int. Conf. on Mobile Computing and Networking(MobiCom'01), 2001, pp. 61–69.

[12] Mieso K. Denko, Jun Tian, Thabo K. R. Nkwe, and Mohammad S. Obaidat, "Cluster –Based Cross-Layer Design for Cooperative Caching in Mobile Ad Hoc Networks," IEEE Systems Journal, vol. 3, no. 4, Dec 2009.

[13] L. Breslau, P. Cao, L. Fan, G. Phillips and S. Sheker, "Web Caching and Zipf-Like Distributions: Evidence and Implications," IEEE INFOCOM, pp. 126-134, March 1999.

[14] K.Suresh Joseph, Madhavarao Boddu, "Improving data accessibility and query delay in CBCC in manet using LFU-MIN," IJCA, vol. 21, no. 9, May. 2011.

[15] S. Lim, W. Lee, G. Cao, and C. Das, "A Novel Caching Scheme for Internet Based Mobile Ad Hoc Networks Performance," Ad Hoc Networks, vol. 4, no. 2, pp. 225-239, 2006.

❑❑❑