***I. J. Computer Network and Information Security*, 2013, 10, 24-29**
Published Online August 2013 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijcnis.2013.10.04

Performance Improvement of Cache Management In Cluster Based MANET

Abdulaziz Zam, N. Movahedinia
Computer Department, Faculty of Engineering University of Isfahan, Isfahan, Iran
abduzam2000@yahoo.com, naserm@eng.ui.ac.ir

Abstract — Caching is one of the most effective techniques used to improve the data access performance in wireless networks. Accessing data from a remote server imposes high latency and power consumption through forwarding nodes that guide the requests to the server and send data back to the clients. In addition, accessing data may be unreliable or even impossible due to erroneous wireless links and frequently disconnections. Due to the nature of MANET and its high frequent topology changes, and also small cache size and constrained power supply in mobile nodes, the management of the cache would be a challenge. To maintain the MANET's stability and scalability, clustering is considered as an effective approach. In this paper an efficient cache management method is proposed for the Cluster Based Mobile Ad-hoc NETwork (C-B-MANET). The performance of the method is evaluated in terms of packet delivery ratio, latency and overhead metrics.

Index Terms — MANET, Cache Management, Cluster Based Caching System

I. INTRODUCTION

One of the key issues in mobile ad-hoc networks (MANETs) is cache management, which improves the transmission capacity of the network. Moreover, perfect placement and control of caching system declines the power consumption.

Two basic concerns in MANET's cache management are how to maintain stability, and the scalability of the cache system. One solution to these problems is cluster based MANET as shown in Fig.1 . Cache management approaches consist of three phases [1 and 15] as shown in Fig.2 . 1- Replacement: this algorithm is responsible for evicting less important or expired data, with time to live (TTL) equal to zero, when the node cache is full and a new data is to be fetched on a request from client. 2- Consistency: this algorithm guarantees that all the copies of a data are identical to the original one on the server.

3- Prefetching: this mechanism fetches the most important data items (that have high probability to be requested in the near future) into the caching node and stores them in the cache memory for responding to the future requests and queries.

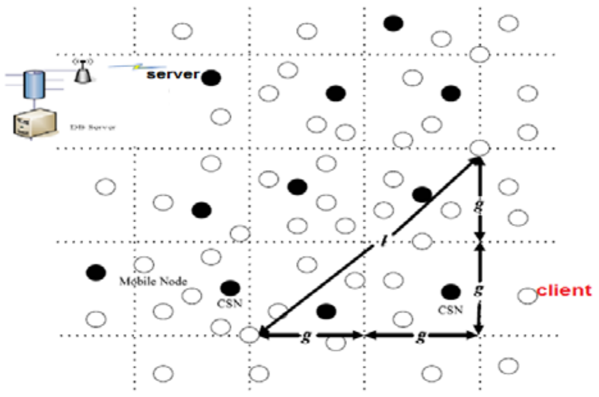


Figure 1. Cluster based MANET

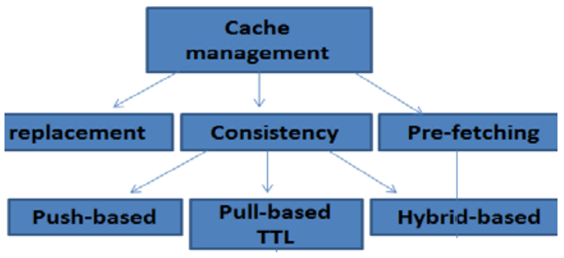


Figure 2. Cache management & consistency phases

Cache management components are all implemented in the middleware layer that takes place under the application layer and above the transport and network layers in which Cluster Based Routing Protocol (CBRP) can be run [1 and 15]. The main goal of this paper is increasing the Cache hit ratio (the percentage of accesses resulting in cache hits) and increasing cluster hit ratio (the percentage of accesses causing cluster hits). We will attain this target by estimating adaptive TTL for the most requested data items and by using hybrid-based consistency maintenance algorithm that provides weak consistency level. Adaptive TTL calculation forms are trying to estimate the next Inter Update Interval (IUI) for a data on the server. The ideal TTL value is very close to IUI and its counter must reach zero on clients when the data is updated on the server. Both Cluster Head (CH) and the server will collaborate to implement our proposed method. In MANET networks, clustering will provide more stability by implementing hierarchical search in which all the cluster members have to send their requests to their CH first. Clustering also can overcome topology changes and mitigate the cache size challenges. All MANET nodes have random mobility, so the distribution of the members in the clusters can be assumed uniform. All cluster members are assumed able to reach each other by one hop jump and thus, every member node can use other member's caches as virtual memory and only one existing copy of any data in a cluster is needed.

The rest of this paper is organized as follows. Section 2 presents the related works. The motivations of our proposed mechanism are described in section 3, and the contributions are elaborated in section 4. Section 5 presents the simulation results and the discussion on the choice of our method's threshold value and finally the concluding remarks are given in section 6.

II. Related works

A. Consistency

Consistency maintenance techniques are categorized into three approaches: 1- Pull or client-based, where a client requests data updates from the source [2]. 2- Push or server-based, where the server sends updates to the clients every time the data items are to be updated [2 and 3]. 3- Hybrid-based, where the data source and the clients collaborate to maintain the data up to date [4, 5 and 12].

Cache management and consistency maintenance mechanisms in usual distributed environments seem to be simple and not proper to be instantly applied to MANET due to its limited cache size and bandwidth, dynamic topology and energy constraint. The common consistency level algorithms are Strong Consistency (SC) and Weak Consistency (WC) [17 and 18]. In SC, all cached copies in the nodes would be up to date at all time but this approach will cause high bandwidth and energy consumption due to the flooding of update messages. In WC, consistency of cached copies is kept between the servers and caching nodes with some latency or deviation, it does not provide warranty that all copies are identical to the original data at the same time, but it conserves energy and avoids unnecessary validation reports. In [3], one new consistency maintenance algorithm was presented, in which both Time To Refresh (TTR) value and query rate of any data are used to help data source to make decision and to push updates selectively to the most requested data in clients. Yu Huang in [16] has presented a new algorithm that provides weak level of consistency maintenance by utilizing hybrid-based approach, in which server will push updates if the most caching nodes need them whereas; caching node will pull updates for a cached data item if it has high probability to be updated on server. In this approach, server uses self-learning method based on a history of the clients-initiated queries.

B. Replacement

When the cache memory of a caching node is full and client needs to fetch a new data, client node must run replacement algorithm that choose invalid or least accessed data to delete and fetch the new data. Time To Live (TTL) is a counter that indicates how long a cached copy is valid for. When counter value reaches zero, the data will be invalid. In MANET networks, TTL of cached copy of a data may reach zero whereas the original data is still up-to-date and should not be changed on data source. This matter often occurs when TTL value is fixed. Proposed TTL calculation algorithms for MANETs can be categorized into fixed TTL approaches in [6], and adaptive TTL approaches in [7, 8 and 19]. Adaptive TTL provides higher consistency, and can enhance replacement algorithm to be effective [7]. Adaptive TTL is calculated using different calculation forms [7 and 19]. The first mechanism in [10], calculates adaptive TTL by multiplying the difference between the request time of the data item and its last update time, by a factor.

In [19], the server sends back the previous IUI to the requesting node that saves the system defined fudged factor (CF). Next, the adaptive TTL is calculated by multiplying the previous IUI by CF. The mechanisms in [11] have taken advantages from the source IUI history to estimate the next IUI and to use it in calculating adaptive TTL. In the latest approach [12], Caching Node (CN) calculates its own estimation for the inter-update interval on the server, and utilizes it to calculate the adaptive TTL value of the data. In this approach, estimating of the inter-update interval requires saving the Last inter-Update Interval (LUI) and the previous estimated inter-update interval, mobile caching node is responsible for calculating TTL by saving IUIs and LUIs for all data items and it can lead to some power and cache consumption. Adaptive TTL plays a vital role in determining validity of the data and thus, replacement algorithm can accurately choose invalid data items to evict. In [1], one new replacement algorithm is proposed in which the cached copy of a data item that has lowest local cache hit number will be replaced, this algorithm will evict the least frequently used data that has been accessed minimum times during its presence in the cache.

C. Pre-fetching

Prefetching is the less attended component of cache management scheme, due to its need to an accurate prediction and because of mobile nodes cannot tolerate high miss prediction penalty. Many prefetching algorithms such as in [12 and 20], have been presented to reduce data access latency in MANET. These algorithms rely on the importance of the data to prefetch the important data items proactively into the cache. The importance of a data item is determined based on some criteria such as access rate and update-to-query rate ratio (Dur/Dqr). Data items that have low Dur/Dqr ratio will be selected as pre-fetched data, if the caching node fetch these pre-fetched data items into the cache before they are actually needed, it will reply to a large number of requests with minimum latency and it will reduce the number of update requests sent to the data source. In [12], the authors have presented a prefetching algorithm in which CN sets pre-fetched bits for the subset of data items that have high access rate. Then CN sends update request to server every short time namely polling interval (Tpoll), whereas it sends update request for the rest of non-pre-fetched data items every relatively long time cycle interval (N×Tpoll), where N, is a configurable number of polling intervals.

III. Motivation

A. The relationship between access rate and update rate

All data items in network can be categorized to: long TTL data like video, PDF and image files and short TTL data such as purse, weather and news files. It is rational that the long TTL data has less update rate whereas; the short TTL data has high update rate, fortunately; there is an extreme proportionality between the access rate from client nodes and update rate on server. In [13], clients have high probability (75 percent) to access the data in short TTL set and low probability (20 percent) to access the data in the Long TTL set. In addition, it is shown that the data source updates are randomly distributed and compatible with the accessing distribution, in which (65.5 percent) of the updates are applied to the short TTL data subset. In this paper, we will use this Proportionality to give more attention to the set of data that have high access rate and high update rate.

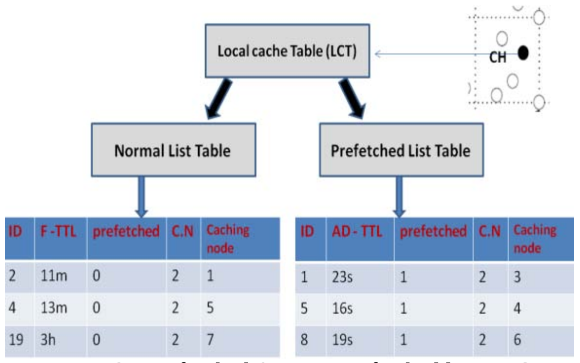
IV. Efficient Cache Management Method

A. system model and assumptions

In this method, we assume that the network topology is divided into non-overlapping clusters as shown in Fig.1, The cluster head (CH) is elected by the cluster members based on power level, cache size and mobility. CH is assumed to have less or no mobility, high power level and large cache size as presented in [14 and 18]. CH of any cluster has two catalogue tables namely: the Local Cache Table (LCT) and the Global Cache Table (GCT). The LCT contains details of all cached data items presented in the respective cluster and the GCT contains details of the cached data items presented in the adjacent clusters.

B. CH phase

In our proposed method, the local cache table of any cluster head is divided into two lists as shown in Fig.3, namely: 1). Pre-fetched List Table (CH-PLT), that contains details of the most accessed data items presented in the respective cluster and 2). Non pre-fetched List Table (NLT), that contains details of the non pre-fetched data items presented in the cluster. In cluster based routing protocol, every client node in the cluster must send its request to CH if the requested data is not cached in its local cache. Then, CH can calculate the access rate for every data in its cluster and compare it with a threshold. For any data, if access rate is greater than threshold, then CH sets its pre-fetched bit, adds this data to the CH-PLT list, and sends report to server to add it to the Server Pre-fetched List (SPL).



ID	F-TTL	prefetched	C.N	Caching node
2	11m	0	2	1
4	13m	0	2	5
19	3h	0	2	7

ID	AD-TTL	prefetched	C.N	Caching node
1	23s	1	2	3
5	16s	1	2	4
8	19s	1	2	6

Figure 3. Pre-fetched & Non-pre-fetched lists in CH

All the pre-fetched data items should have adaptive TTL and push-based updating mechanism. The CH-PLT is a catalogue and is a part of the Local Cache Table (LCT) and thus, CH will save a copy of pre-fetched data only in two conditions:

1) When the file is requested from CH itself.

2) If the requesting node in cluster cannot save it, could not be a caching node.

Otherwise, CH saves only details of the data. All of the CH-PLTs have limited and equal size indicates to the number of pre-fetched data items. The size of any CH-PLT is equal to the relative SPL size. If the CH-PLT is full and one new data item exceeds the threshold, it will be replaced with the least accessed data item. Server must be reported by this replacement and evicted data would take place in CH-NLT. Other data items that their access rate is lower than the threshold are added to the second part of the LCT namely CH-non pre-fetched list table CH-NLT. This list should have fixed TTL and pull based updating mechanism, and its data items have to be sorted according to their access rate for replacement mechanism purpose. By the way; CH is only responsible for dividing its cluster data items into two lists and its response to out of cluster requests is normal. As assumed in the later works, CH will reply to out of cluster requests if it has the details of the requested data in its LCT.

C. Server phase

Server has to save last ten inter update intervals (IUIs) for every data to be used for estimating of the current IUI.

For all data in server pre-fetched list (SPL), server calculates the estimated inter update interval by using the

Following EQ.1:

$$IUI(t) = \frac{a*IUI(t-10)+2a*IUI(t-9)+\ldots+10a*IUI(t-1)}{(1+2+\cdots+10)a} \quad (1)$$

This form means that the current IUI has high probability to be closer to the later inter update interval than to the older one. Server calculates the adaptive TTL of any pre-fetched data every time it would be updated using this form {AD-TTL = CF×IUI (t)}, where CF is system fudged factor. By receiving reports from CHs, Server forms many equal SPLs, every list is related to one cluster and indicates to the most accessed data items presented in the cluster. Server must send validation messages frequently for the SPLs. To avoid an unnecessary messages propagation in network, validation reports propagation must be controlled and we suggest the (average time-validation-sending algorithm) to control it. Once, a TTL of any data in any SPL list would be changed, server calculates the TTL average for this list. For example, if SPL(i) has N data items, then the average TTL of SPL(i) is defined as:

$$TTL - AVG(i) = \frac{TTL1+TTL2+\cdots+TTLn}{N} \quad (2)$$

Server should send validation report to cluster i every TTL-AVG(i) time cycle. There is high probability that the most of pre-fetched data items in list i are requested by the members of cluster i or updated by server through this {TTL-AVG(i)} time cycle. If a pre-fetched data would be updated on server between two successive time cycles, server sends updated data only to its related cluster head. In non pre-fetched data list, when a data is updated, server sends an invalidation report to all CHs that have this data in their non pre-fetched list table (NLT). when this data is requested in a cluster, then the head of this cluster CH reply with ACK message to the server in which, sends updated data with fixed TTL to all related CHs, to keep all cached copies of this data consistent.

D. Server algorithm

Every Server Pre-fetched List is related to one cluster and has its own respective timer; all timers are set to zero at the beginning. The server will estimate the adaptive TTL of any pre-fetched data every time it is updated. To update the pre-fetched data lists in all clusters, the server should send validation reports periodically by implementing the flowchart shown in Fig.4. Validation report contains updated data items with adaptive TTL.

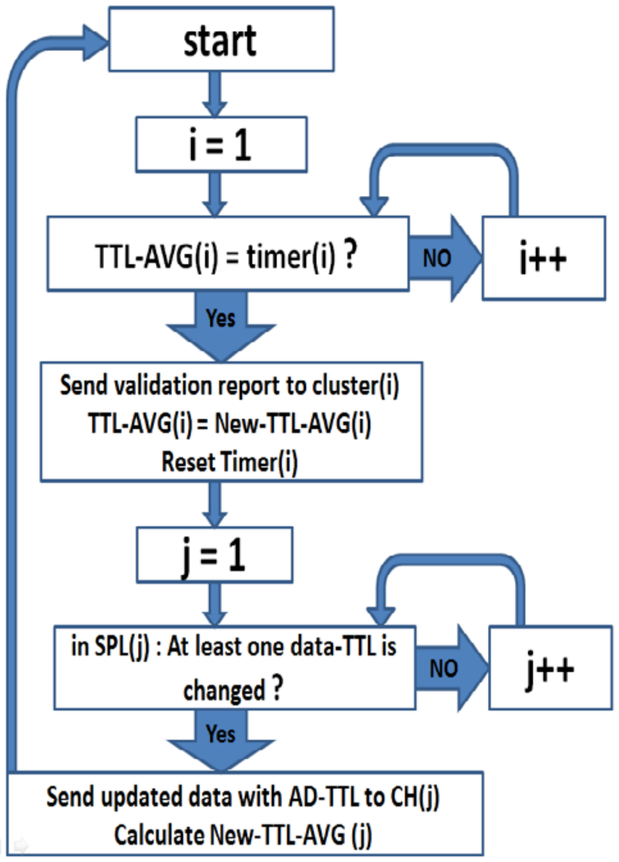

start
i = 1
TTL-AVG(i) = timer(i) ?
NO
i++
Yes
Send validation report to cluster(i)
TTL-AVG(i) = New-TTL-AVG(i)
Reset Timer(i)
j = 1
in SPL(j) : At least one data-TTL is changed ?
NO
j++
Yes
Send updated data with AD-TTL to CH(j)
Calculate New-TTL-AVG (j)

Figure 4. Flowchart of the pre-fetched data lists update

E. Scenario

Fig.5 shows two adjacent clusters, where cluster (2) has seven node members and one CH. Node (1) sends data (d) access request to its CH. The CH has not the details of this data in its pre-fetched and non pre-fetched lists, but it finds data (d) details in its Global Cache Table (GCT). Therefore, it forwards the request to the server according to its adjacent clusters.

The server responses to the request by sending the data (d) with fixed TTL (20s) to CH (2). CH (2) adds this data to its non pre-fetched list table, and forwards it to the node (1). If other nodes in cluster (2) request the data (d), CH Responses that this data is cached in node (1) and it is valid for 20s. CH also calculates the access rate of the data (d). When access rate (d) exceeds the threshold, CH sets its pre-fetched bit and adds it to the (CH -PLT). CH also must send the report to the server to add data (d) to its SPL, and then the server has to calculate its adaptive TTL and also recalculate the pre-fetched list TTL-AVG of cluster (2).

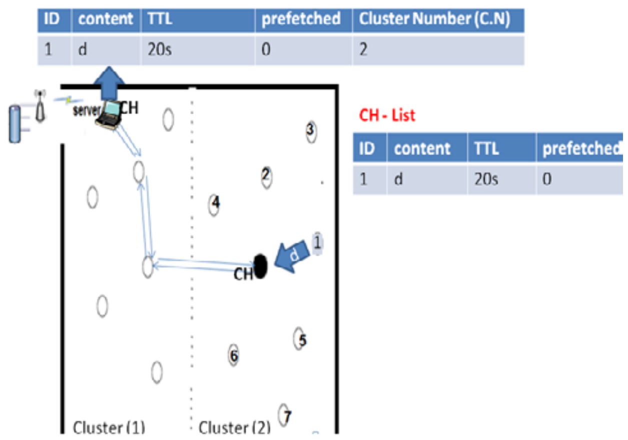

ID | content | TTL | prefetched | Cluster Number (C.N)
1 | d | 20s | 0 | 2
server
CH
CH - List
ID | content | TTL | prefetched
1 | d | 20s | 0
CH
d
Cluster (1)
Cluster (2)

Figure 5. Forwarding request and forming (CH non pre-fetched list Table)

V. SIMULATION

We have used the network simulator (NS2) to simulate our efficient cache management method by using wireless Standard IEEE 802.11, Network size = 50 nodes in a 1000×1000 m zone, time of simulation = 100 seconds, The number of connections is 30, and we have used the Constant Bit Rate (CBR) traffic model. In these simulations, we have compared the proposed method (ECM) with the existing cluster based method (CBM). As the results shown in fig.6, 7 and 8 infer, the proposed method demonstrates superior performance in terms of the imposed overhead, the packet delivery ratio and the data access latency.

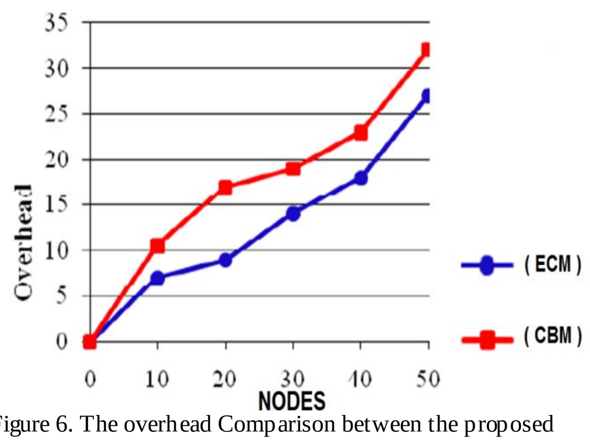


Figure 6. The overhead Comparison between the proposed cache management method (ECM) and the cluster based method (CBM)

Determining the caching threshold value is very important factor in the performance of the proposed method. This value depends on the average number of cluster members. Any important data is to be requested from the most of cluster members in a short time period. In addition, the threshold value should be adaptive and adjusted to the network traffic characteristics. If the network is going to be congested, the threshold value must be set to a lower value to reduce the number of validation messages propagated in the network. Contrarily, if the network has low traffic, the threshold value must be higher to increase the number of pre-fetched data items in the lists, and then to improve the network performance.

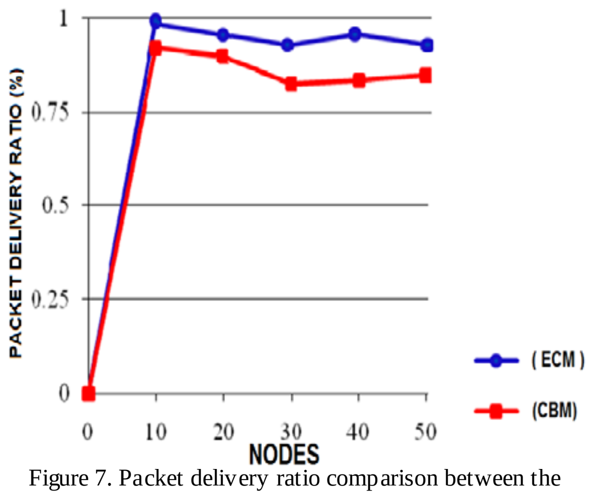


Figure 7. Packet delivery ratio comparison between the proposed cache management method (ECM) and cluster based method (CBM)

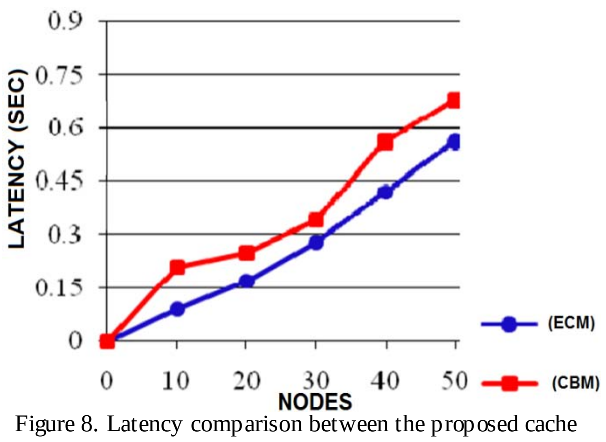


Figure 8. Latency comparison between the proposed cache management method (ECM) and cluster based method (CBM)

VI. CONCLUSIONS

In this paper, we have focused on the prefetching scheme according to pushing the validation reports to all the pre-fetched data every TTL-AVG time cycle. In addition, we have presented an efficient consistency method that uses hybrid-based consistency maintenance approach to switch between pulling and pushing updates, based on the importance of a data. Replacement scheme can be easily implemented here in which the CH evicts and replaces only data items presented in NLT by giving priority to the less accessed data. Other nodes in cluster can utilize the existing least frequently used (LFU-MIN) replacement algorithm, and can get help from (CH-LCT)-details to determine the less accessed data to delete. Our main goal (increasing cluster and local cache hit ratio) can be obtained by maintaining the most accessed data up to date in cluster. Adaptive TTL can also increase local cache hit ratio. If adaptive TTL would be estimated well, data will be up-to-date for the most possible time in the local cache, and through this time, all requests for this data hit. By increasing cluster-hit ratio, access delay and power consumption will be reduced, and accessing data will be more reliable in case of non-reliable links.

REFERENCES

[1] Madhavarao Boddu and K. Suresh Joseph. improving Data Accessibility and Query Delay in Cluster based Cooperative Caching (CBCC) in MANET using LFU-MIN. in International Journal of Computer Applications (0975 – 8887) Volume 21– No.9, May 2011.

[2] D. Barbara and T. Imielinski. Sleepers and Workaholics: Caching Strategies in Mobile Environments. In MOBIDATA: An Intractive journal of mobile computing. Vol. 1 No.1. 1994.

[3] H. Jin, J. Cao and S. Feng. A Selective Push Algorithm for Cooperative Cache Consistency Maintenance over MANETs. In Proc.Third IFIP Int'l Conf, Embedded and Ubiquitous Computing, Dec. 2007.

[4] P. Kuppusamy, K. Thirunavukkarasu and B. Kalaavathi. Review of cooperative caching strategies in mobile ad hoc networks. in International Journal of Computer Applications, 2011.

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

[6] J. Jung, A.W. Berger and H. Balakrishnan. Modeling TTL-based internet caches. in IEEE INFOCOM 2003, San Francisco, CA, March 2003.

[7] P. Cao and C. Liu. Maintaining strong cache consistency in the World-Wide Web. In IEEE Trans. Computers, v. 47, pp. 445–457, 1998.

[8] Chankhunthod, P. Danzig, C. Neerdaels, M. Schwartz and K. Worrell. A hierarchical internet object cache. in USENIX, pp. 13, 1996.

[9] Y. Sit, F. Lau and C-L. Wang. On the cooperation of Web clients and proxy caches. in 11th Int'l Conf. Parallel and Distributed Systems, pp. 264-270, July 2005.

[10] V. Cate. Alex. A Global File system. In USENIX, pp. 1-12, May 1992.

[11] D. Li, P. Cao and M. Dahlin. WCIP: Web Cache Invalidation Protocol. in IETF Internet Draft, March 2001.

[12] Kassem Fawaz and h.Artail. DCIM: Distributed Cache Invalidation Method for maintaining cache consistency in wireless mobile networks. in IEEE, 2012.

[13] N. Sabiyath Fatima and P. Sheik Abdul Khader. Efficient Data Accessibility using TTL Based Caching Strategy in Manet. in European Journal of Scientific Research ISSN 1450-216X Vol.69 No.1 pp.21-32 © Euro Journals Publishing, Inc, 2012.

[14] P. Kuppusamy, K. Thirunavukkarasu and B. Kalaavathi. Cluster Based Cooperative Caching Technique in Mobile Ad Hoc Networks. in European Journal of Scientific Research ISSN 1450-216X Vol.69 No.3 (2012), pp. 337-349 © EuroJournals Publishing, Inc, 2012.

[15] Anand Nayyar. Cross-Layer System for Cluster Based Data Access in MANET'S. in Special Issue of International Journal of Computer Science & Informatics (IJCSI), ISSN (PRINT), 2001.

[16] Yu Huang . Cooperative cache consistency maintenance for pervasive internet access. In State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China, 2009.

[17] Wenzhong Li, Edward Chan, Daoxu Chen and Sanglu Lu. Performance analysis of cache consistency strategies for multi-hop wireless networks. Published in Springer Science Business Media, LLC, 8 June 2012 .

[18] Kuppusamy and B. Kalaavathi. Cluster Based Data Consistency for Cooperative Caching over Partitionable Mobile Adhoc Network. in American Journal of Applied Sciences 9 (8): 1307-1315, 2012.

[19] Yuguang Fang, Zygmunt Haas, Ben Liang and Yi-Bing Lin. TTL Prediction Schemes and the Effects of Inter-Update Time Distribution on Wireless Data Access. in Department of Electrical and Computer Engineering University of Florida, 2004.

[20] Mieso K. Denko, University of Guelph, Canada. Cooperative Data Caching and Prefetching in Wireless Ad Hoc Networks. In Idea Group Inc. Volume 3, Issue 1 edited by Jairo Gutierrez © 2007.

AbdulAziz Zam, received his B.Sc. from Isfahan university, Isfahan, Iran in 2009. He started Master's degree in 2010 in the same university. Currently he is completing his thesis at computer department, faculty of engineering. His research interests are wireless networks and telecommunications.

N.Movahedinia, received his B.Sc. from Tehran University, Tehran, Iran in 1987, and his M.Sc. from Isfahan University of Technology, Isfahan, Iran in 1990 in Electrical and Communication Engineering. He got his PhD. degree from Carleton University, Ottawa, Canada in 1997, where he was a research associate at System and Computer Engineering Department, Carleton University for a short period after graduation. Currently he is an associate professor at the Computer Department, University of Isfahan. His research interests are wireless networks, signal processing in communications and Internet Technology.