An Administrative Cluster-Based Priority Cooperative Caching Strategy in MANET for Improving Data Availability and Data Accessbility

Dhivya.S[1], Manikandan.S[2]
[1, 2] Department of Electronics & Communication Engineering
[1, 2] PSN College of Engineering and Technology, Tirunelveli

Abstract-*Recent explosive growth in computing and wireless communication technologies has led to an increasing interest in mobile ad hoc networks (MANET). Among the many challenges for MANET designers and users, data availability and data accessibility are critical issues. Caching is considered as an effective solution for the availability problem. Data accessibility problem can be solved by processing requests based on their classification. In this paper, a new priority cooperative caching strategy for MANETs, which is called Administrative Cluster – Based Priority Cooperative Caching (ACPCC) was proposed. This strategy keeps at most two copies of the cached data items (DI) in each cluster to improve data availability and it serves requests either priority or normal which is to ensure that priority requests are served with minimum cache discovery delay. Experimental results have shown that ACPCC outperforms recent cluster-based Priority caching strategies as it introduces higher better data accessibility as well as data availability.*

I. INTRODUCTION

Rapid advances in wireless communication have created much interest in mobile ad-hoc networks (MANETs). As mobile ad-hoc networks grow widely, there is a need for approaches to increase the availability of data items as well as the performance of processing requests. Dynamic mobility, disconnection, limited bandwidth, and energy are the main characteristics of MANETs. Therefore, accessing distant data items in such a situation is challenging and without an efficient mechanism the performance of the system in terms of average delay, hit ratio, and bandwidth will be affected. Data caching in mobile hosts is widely known as the successful key to enhance the performance in the mobile data base environment. Thus, caching frequently accessed data items by mobile hosts has a direct relation in increasing the availability of data items[1][2][3], reducing the communications with the main database, and enabling accessing cached data during database connection failure.

Requests are served within the neighbor nodes (NBs) without relying on the main database server. Requests are differentiated based on their classifications, either normal or priority which is specified by the requested node. Also, each node will exchange the information of cached data items which is to be stored in the Recent Priority Table (RPT). This classification based serving request increases data accessibility. The ultimate goal of this ACPCC is to make the data available in minimum time and by utilizing fewer resources. As other cluster based cooperative caching techniques, ACPCC divides the network into many overlapped clusters in which each cluster is completely managed by a Cluster Manager (CM) as well as a Cluster Backup (CB). ACPCC integrates between five essential modules which are;

i) Cache Discovery Module (CDM) - cache discovery, which is essential to discover and deliver requested data items from the neighboring MNs which is based on their classification. Moreover, it decides which data items can be cached for future use. ii) Cache Admission Control Module (CACM) - which decides which data item will be cached. This process ensures that, only the valuable items will be cached. iii) Cache Replacement Module (CRM) - which is vital to replace some cached data items to make room for new data items when the caching space is not enough to cache the new ones, iv) Cache Consistency and Maintenance Module (C2M2) - which ensures that the cached data items are up-to-date. v) Cluster Management Module (CM2) – This module aims to promote the system potential by perfectly integrating its modules. CM2 is the responsible of managing, supervising and maintaining the cluster's data by organizing the cluster overall work between its cooperating nodes that are all applying the integrated modules of the ACPCC caching system.

ACPCC is simulated using Java Caching System JCS2 with the efficient platform Java Enterprise Edition EE8 by a client/server model.

II. RELATED WORK

A new cooperative caching technique has been introduced by P. Nagaraju et al. [4] to minimize objects' download cost through using MANET's service which includes search and price models. For the search model, there

are two request generation modes which are homogenous request generation model and heterogeneous request generation model.

Preetha Theresa Joy et al. [5] have proposed an effective cache resolution technique which uses a split table approach to resolve the cached data request which in turn reduces latency, flooding and network traffic. This technique also avoids the drawback of group maintenance by having a distributed approach.

P.Kuppusamy [6] proposed a Cluster Based Update (CBU) algorithm to avoid the stale data in caching nodes. In CBU approach, the source monitors data update rate, Dur, and query access rate Dqr of MNs for every data D. The caching nodes also maintain these values for each data D that it caches. The caching node computes ratio for every data when it obtains data update from the source and compares it to a threshold value. Using this algorithm made the source avoids unnecessary data updates that have less query access rate, so it provides better performance.

In [7] Kuppusamy et al. have proposed Cluster Based Adaptive Push and Pull Algorithm (APPC) and Cluster Based Data Consistency (CBDC) approach based on mobility and connectivity to address the consistency requirements and maintenance then improve the data availability over MANET However, the consistency maintained by updating the DI at variant caching nodes led to an additional overhead for this model.

In [8] Artail etal (2008), nodes rely on the indexing of submitted cached requests to locate the desired dataitems. Nodes in a cooperative and adaptive system (COACS) can be one of the two possible roles, which are cache node (CN) and query directory (QD). QD is responsible for caching requests that are submitted by the requesting mobile nodes, while CN is responsible for caching data items (responses to requests). There are two ways of locating data items in COACS technique. First, there quest traverses the list of QD nodes before the request is redirected to the database server when the data item is not found in the system (miss). Second, the request traverses the list of QD nodes before the request matches one of these QDs' nodes.

III. THE PROPOSED STRATEGY (ACPCC)

ACPCC is a cluster based[9][10] priority cooperative caching technique, which divides the network into a set of overlapping clusters. Each cluster is managed by a Cluster Manager (CM) and a Cluster Backup node (CB). CM is the node that is responsible of maintaining the cluster cache information and management information. Cluster manager is the most powerful node within its cluster, it is considered as the supervisor of its cluster. It accomplishes all the administrative tasks within the cluster besides playing the main role in the caching process. It is also responsible of the communication process with other network cluster managers and the data server. Cluster backup is the second most important node after the CM. It is used to aid the CM in both network administration and caching process.

A. Setting up Network Clusters

ACPCC maintains the network clusters through two stages, which are; (i) cluster formation and (ii) cluster maintenance. During the former; each node broadcasts its unique Node ID (NID) and its selection factor within its transmission range. Hence, each node in the network knows the IDs of its neighbors and its selection factor value. Selection factor value can be calculated by several criteria which is explained below. Selection factor value is mainly used for selecting the CM as well as CB. At first every node should compare its selection factor value with other node's selection factor value. If the node that has the highest value, it is selected to be the cluster manager and the node that has next highest value will be selected to be the cluster backup node. Hence, the cluster is now formed and it consists of the CM, CB and the entire member mobile nodes that are within its transmission range.

B. Calculation of selection factor value

There are factors to be considered while calculating selection factor that are,

1. Battery Power Level (B)
2. Processing Power (P)
3. Memory Size (S)
4. Mobility Degree (M)
5. Processing Overhead (O)
6. Proximity to Cluster Centroid (D)

$$\text{Selection Factor (SF)} = B(x)*P(x)*S(x)*/(M(x)*O(x)*D(x)) \quad (1)$$

Battery Power Level, It's a measure of the battery capacity, When the power capacity decreases, the node performance drops. Processing Overhead, This attribute measures the degree of the processing workload on the MN which describes the maximum CPU utilization. Processing Power, which includes the CPU efficiency and memory size, and Cache Free Space indicates the node's real characteristics. The Degree of Mobility indicates to the MN movement

behavior. The node mobility and connectivity are computed through RSS, which gives somehow approximate distance between any two nodes through received packets exchange.

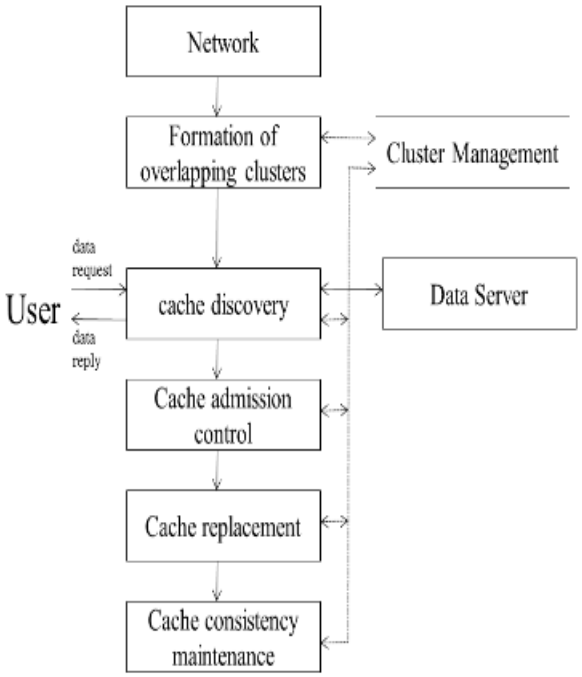


Fig.1 Block Diagram of ACPCC

The below section describes about each modules in the proposed ACPCC algorithm.

C. Cache Discovery Module (CDM)

CDM is used to discover and deliver the required DI whether it is found internally at the MN local cache or externally at another node in the same cluster or at another neighboring one based on its request classification. It uses Node Cache Table (NCT), Recent Priority Table (RPT) and Cluster Cache Table (CCT) in its searching for the requested DI, which leads to reducing the access latency. It is assumed that, every MN has a NCT and RPT. Moreover, CM has a CCT. NCT is a table specific to each node, which contains some important attributes that are related to the locally cached DIs at this MN. This table lists all the cached items at this node as well as the full descriptive information about every cached item like its path, access count, expire date. Also RPT is a table specific to each node, which contains the information about cached data items in the neighboring nodes. CCT is a CM's specific table, which contains essential information about the cached DIs at all nodes within the cluster. This table plays a vital role in the caching process and the management process with the aid of Cluster Management Table (CMT). Every update at any NCT is always followed by an equivalent concurrent update at the CCT.

When a MN requests a specific DI, CDM firstly searches in its NCT to check the existence of the required item at the node local cache (shown in fig 2 (a)). If the DI is found, then it's loaded locally from the node cache. If the data item is not found in the NCT, then the CDM searches it in the RPT, if it is found then CDM loads loaded from the particular node (shown in fig 2 (b)). Again if the data is not found in the RPT, then the request is send to the CM, CM checks the DI in its CCT, if it has, it will send it to the source node shown in fig 2 (c1,c2,c3)). If the DI is not found in the CCT, then the request classification is checked which is specified by the requesting or source node. The request may be either priority or Normal. If the request is priority, CDM directly takes the DI from the database server (shown in fig 2 (d1,d2,d3)). If the request is normal, CDM sends the request to the neighboring cluster manager. Then the neighboring CM's searches the particular DI in its CCT. The request will be passing to all the cluster CM's unless the data item is found. If the DI is not found in any of the CM then the CDM takes it from the database server (shown in fig 2 (e1,e2,e3,e4,e5,e6)). After getting the DI from any one of the ways, the corresponding NCT and CCT tables are updated.

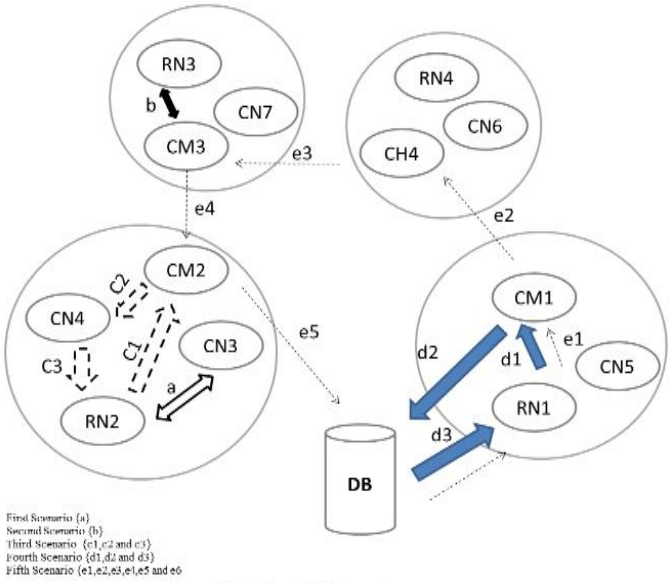


Fig.2. The CDM Scenarios

D. Cache Admission Control Module (CACM)

The main purpose of CACM is to permit accessed DIs to be cached at the node cache or at the neighboring nodes' cache to increase its accessibility without the need of contacting the data server. Most of the previous cooperative caching techniques provide several cached copies for the same DI. ACPCC maintains only one cached copy for the DI per cluster. However, only under some certain conditions it may have another cached copy for the same DI. One of the

conditions that lead to maintain another copy of the cached DI is its DV which can be defined as the total number of DI's access count (N) per recent time interval (tr). DV can be referred as the DI's recent access rate.

$$DV = N / tr \quad (2)$$

Where, N - Is the total number of data item access count for the entire cluster, tr - The recent time interval at which the data item is accessed. Assume tc is the current time and tl is the time of the DI's last access then the tr value can be calculated from the following equation tr = tc - tl ,Where, tc - Is the current time, tl - The time at which the data item is last accessed.

While caching particular DI, it is important to note the DV of that DI only when it has more than one copy in that cluster. If that DI demand value is smaller than the demand value threshold value (DVth) then the requested node only caches the DI and deletes the cache copy in other nodes. In case if the DV is higher than DVth then the requested node also stores the cached copy and the node which has highest recent access count also caches the DI.

E. Cache Replacement Module (CRM)

Cache replacement is used to make a sufficient room for the new DI when a cache miss takes place while there is no sufficient space for hosting the new DI. In order to provide room for the new DI, the old DI should be replaced. Many factors should be considered to elect the victim DI, which will be replaced. Data Item Valve (DIV) should be calculated for every DI. That DIV decides which DI should be replaced. Data Item Value is given as,

$$DIV = DIW * (RUR*UR*TTL) \quad (3)$$

Data Item Weight can be calculated by

$$DIW = LAC/S. \quad (4)$$

Where, Local Access Count (LAC) is the total number of times that DI is accessed by a node. “S” denotes the size of that data item.

Recent Usage Ratio (RUR) can be calculated by

$$RUR = \text{Total Access Count} / tr, \quad (5)$$

Total Access Count is the total number of times that DI is accessed by the cluster. tr is the time interval at which the data item is accessed.

Update Rate (UR) can be calculated by

$$UR = Nu/ tc – tf, \quad (6)$$

Where, Nu is the total number of data item updates, tc is the current time, and tf is the time at which the DI is cached at the current node. Time-To-Live (TTL) that indicates, whether the cached DI is a valid copy. Then, the DI of the minimal DIV is selected as the victim.

F. Cache Consistency and Maintenance Module (c2M2)

C2M2 performs three important tasks, (i) it deletes expired data copies that have small total access, relative to other DIs' access count, to free more cache space, (ii) checks the validity of the cached items and (iii) the new important feature is to update cache important DIs which are frequently accessed periodically in order to always keep recent DI version of the DIs which have high DV values with respect to the other DIs in the cluster. These DI’s should be updated periodically taking in consideration the network circumstances such as; bandwidth availability, traffic overhead, and node battery power. When the network suffers from low bandwidth and high traffic, DI's update process should be postponed till the network circumstances get better. If the MN encounters a shortage in its battery power to a critical value or it has a high load, then DI's update process will be turned off to save node's resources.

G. Cluster Management Module (CM2)

ACPCC uses a cluster management module (CM2), which aims to promote the system potential by perfectly integrating its modules. CM2 administrative tasks can be listed as, (1) Supervising all the transmission and update operation for all the data all over the cluster especially between the CM and CB, (2) Managing and interacting with all the cache operations at the entire cluster's nodes, (3) Checking the MNs' status periodically in order to always keep nodes' recent correct data in their NCTs, and in both the CCT and CMT at both CM and CB. (4) Registering and serving the entering nodes and unregistering the leaving nodes with performing the required updates to the related NCTs at these nodes which leads to CCT updates, (5) Selecting the candidate CM and the candidate CB and performing the switching operation between CM and CB when required or at emergency cases. CM2 keeps its managerial data at CMT which contains all the required information about all the nodes at the entire cluster. This table is always related and integrated with CCT. It's mainly used in supervising the all cluster's nodes and in taking the proper managerial decisions at any tim

IV. SIMULATION MODEL

The simulation of ACPCC model was implemented using Java enterprise application using EE8 platform [11]

integrated with Apache hosting web server [12], MySQL database server and Java Caching System (JCS2) API. This model is implemented as a client / server model, the simulated mobile environment is composed of a data server which serves all the clients' nodes at all the clusters within the entire network. The node acts as a server when it sends data and acts as a client when receiving data. The CM and CB are implemented as general client nodes that are selected by the CM2. Nodes that generate DI request are selected randomly and have Zipf-like [13] distribution pattern. The joining and leaving MNs are also selected randomly at the network with variant rate. The database server that is used for maintaining the data items was "MySQL database server", this server is integrated with the hosting web server "Apache web server" which is mandatory for enterprise java applications.

V. PERFORMANCE METRICS

The used metrics to measure the performance of the system are: hit ratio, access latency, server request ratio.

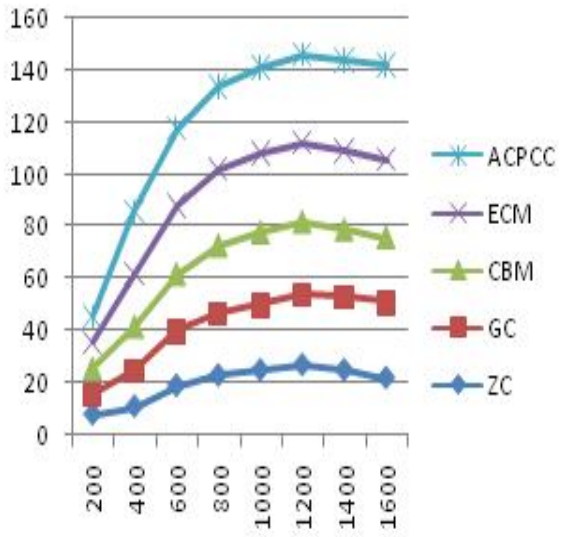
Hit Ratio - The ratio of the number of the requested DIs found in the local or remote cache of a MN over the number of the DIs that queried by the MN.

Access Latency - The time interval between the time of generating a query in the RN and the time of receiving requested DI from the data source.

Server Request Ratio - It measures the rate of requesting data from the data server when the requested DI is not exist at its cluster.

A. Experiment 1: Hit Ratio Vs Cache Size

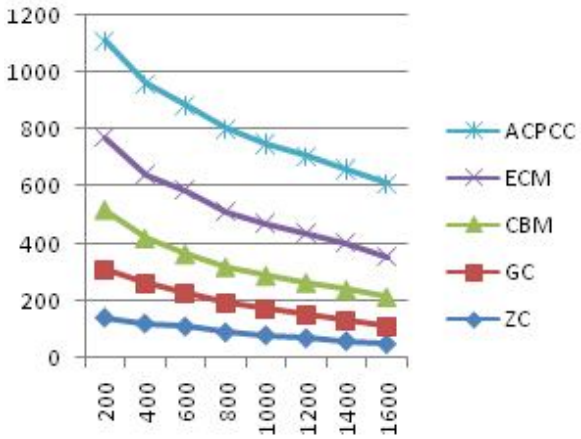
The hit ratio increases by increasing the cache size for all the techniques but just till reaching specific values then, the behavior will differ. At both ZC and GC the cache capacity gets saturated after reaching specific cache size then ZC and GC use the cache replacement to free more space so, the cache hits decrease due to the deletion of some DIs leading to hit ratio reduction. Both ECM and ACPCC have better replacement techniques which use TTL for ECM and DIV mainly for ACPCC model. At both ECM and ACPCC, the hit ratio keeps increasing even when reaching the saturation level of the cache capacity.



Hit Ratio (%) Vs Cache Size (KB)

B. Experiment 2: Access Latency Vs Cache Size

Increasing cache size reduces the access latency. Large cache size can keep more DIs so the nodes can access them locally which decreases the access latency. ACPCC has the best performance due to using new efficient replacement technique, which depends on many realistic parameters.

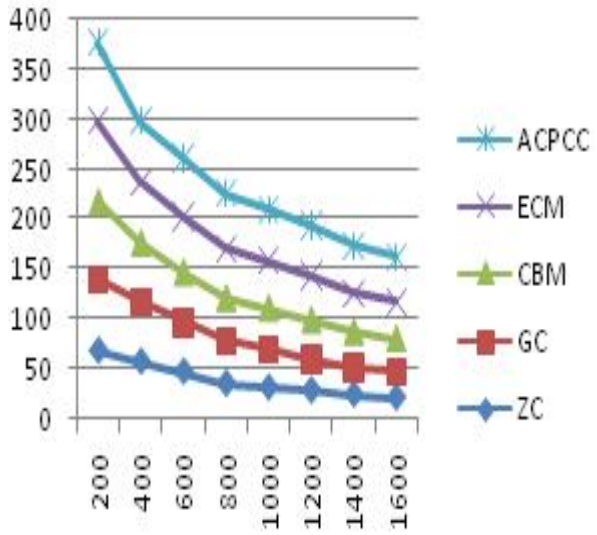


Access Latency Vs Cache Size

C. Experiment 3: Server Request Ratio Vs Cache Size

The server request ratio is decreased by increasing the cache size. ACPCC technique has the lowest level of the server request ratio which means that, when the cache size increases, the cache free space is also increased especially with using efficient cache consistency and maintenance technique. The free cache space is then used to cache more DIs, so there's no need to connect to the server except for requesting a new DI which is not already exist at the entire

cooperative cache that's lead to the reduction of the server request ratio.



Server Request Ratio Vs Cache Size

VI. CONCLUSION

In this paper, a new Administrated Cluster-based cooperative caching (ACPCC) strategy has been introduced. ACPCC has several salient properties that others do not have. It employs a CB node, which assists the CM in managing the underlying cluster. ACPCC ensures that there is only one copy of the cached item per cluster, which can be increased to be at most two copies in case of demand DIs. ACPCC maintains five modules, which integrate together to improve the overall system performance. CDM uses NCT and CCT in its searching for the requested DI, which reduces the access latency. CACM is responsible of caching DIs, determining and controlling their number of copies and updates their information in NCT and CCT tables. CRM on the other hand, utilizes a new metric which is DIV to decide efficiently, which DIs can be replaced in order to have more cache space for caching new DIs. CM2 improves the system reliability by keeping recent updates and getting rid of stale DIs to release system resources. Finally, CM2 is responsible for updating cache information, integrating between cache system modules and supervising the nodes' behavior in order to wisely select the next CM and CB. This whole integral model improves the data availability and the system efficiency by increasing local and remote hit ratio, reducing access delay, overhead, and consumed battery power. Experimental results have shown that ACPCC outperforms recent cooperative caching techniques as it introduces better hit ratio as well as less access latency.

VII. FUTURE WORK

For the future work, two important issues which have to be considered in MANETs. The first issue is the energy consumption which has to be reduced to its minimum level especially in connection operations which absorb most of the energy of the mobile node that affects badly its overall performance and sometimes leads to node disconnection due to the reduction in its battery power level. While the second issue is about cache contents' protection as sometimes sensitive data which needs some privileges to be accessed (i.e. the server can prevent nodes from caching some data or limit the number of nodes that can cache it), also these data may need to be protected from modifying or duplicating its contents by malicious node. The work framework will use Information Centric Networking (CCN) technology which provides valuable performance in terms of service reliability, latency, flexibility and scalability.

REFERENCES

[1] Safa H, Artail H, Nahhas M. A cache invalidation strategy for mobile networks. Elsevier Journal of Network and Computer Applications 2010;33(2):168–82.

[2] Radhamani G, Umamaheswari S. Comparison of cooperative caching strategies in mobile ad-hoc network (MANET). Recent Trends in Networks and Communications 2010:439–46.

[3] Kuppusamy P, Thirunavukkarasu K, Kalaavathi B. Are view of cooperative caching strategies in mobile AdHoc networks. International Journal of Computer Applications 2011;29:22–6.

[4] P. Nagaraju, S. Praneetha, "Propagate Cooperative Caching on MANETS", (IJCSIT) International Journal of Computer Science and Information Technologies, Vol(5), No(4), pp.5371-5374, 2014.

[5] Yu Huang, Beihong Jin, Jiannong Cao, Guangzhong Sun, YulinFeng, "A Selective Push Algorithm for Cooperative Cache Consistency Maintenance over MANETs", Embedded and Ubiquitous Computing (EUC 2007), Volume 4808 of the series Lecture Notes in Computer Science pp 650-660. (Springer database).

[6] P. Kuppusamy and B. Kalaavathi, "A Novel Cache Update Algorithm for Consistency Maintenance over Cluster Based MANET" International Journal of Computer Science & Engineering Technology (IJCSET), Vol(3), No(7), pp. 235- 242, 2012.

[7] P. Kuppusamy and B. Kalaavathi, "Cluster Based Data Consistency for Cooperative Caching over Partitionable Mobile Adhoc Network", American Journal of Applied Sciences Vol(9), No(8), pp.1307-1315, 2012.

[8] Artail H, Safa H, Mershad K, AbouAtme Z, Sulieman N. COACS : a cooperative and adaptive caching system for MANETs. IEEE Transactionon Mobile Computing, IEEE Computer Society 2008;7(8):61–977.

[9] S. Chandra, I. Saha, P. Mitra, B. Saha and S. Roy, "A Brief Overview of Clustering Schemes Applied on Mobile Adhoc Networks", International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE), Vol(5), No(2), pp.667-675, 2015.

[10] A. Bentaleb, A. Boubetra, and S. Harous, "Survey of Clustering Schemes in Mobile Ad hoc Networks ", Communications and Network, Vol(5), No(1), pp. 8-14, 2013.

[11] Available:http://www.oracle.com/technetwork/java/index.html

[12] Available: http://apache.org/

[13] Breslau, L.; Pei Cao; Li Fan; Phillips, G.; Shenker, S., "Web caching and Zipf-like distributions: evidence and implications," INFOCOM '99.