Volume 5, Issue 8, August 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

A Comprehensive Review on Conserving Data Cache Consistency Approach in MANET for Data Replication

Sivakumar K[1], Premylla Jeremiah[2], Kannan P[3], Anatte Schaffer Anak Juslen[4], Nirmalkumar A[5]
[1]Research Scholar, Faculty of Computer Science and Engineering, Sathyabama University, India
[2]Research Scholar, University Technology Malaysia (UTM), Malaysia
[3]Sr. Lecturer, Faculty of Science, Technology, Engineering and Mathematics, INTI International University, Malaysia
[4]Computing (Hons) Student, University of Greenwich, United Kingdom (SEGI College Kuala Lumpur), Malaysia
[5]Reseaarch Scholar, Faculty of Computer Science and Engineering, Anna University Chennai, India

Abstract— Nowadays, there are Wireless Fidelity (Wi-Fi) enabled devices increases rapidly for various purposes. All such device can communicate with each other in an infrastructure-less network of mobile devices called as Mobile Ad Hoc Network (MANET). In wireless environment, data replication posture challenging concerns in conserving data cache consistency and fault tolerance. This paper proposes reduce the query delay marginally and increase the data cache consistency mechanism for data replication in mobile environments, using an established architecture in mobile computing, which can be used for server update technique for data cache consistency. In this approach, the data stores the queries that are succumbed by query directories, and these stored queries requested by caching nodes and it will be updated by the server using the proposed control algorithm. It maintains data cache consistency in wireless network and this approach was based on a cooperative and adaptive caching system for MANETs architecture for caching data items in wireless network, we then analyze the performance evaluation of the existing system and proposed system using the mathematical model.

Keywords— Mobile Ad Hoc Network (MANET), data replication, server update, data cache consistency, query delay, query update, query directory.

I. INTRODUCTION

A Mobile Ad Hoc Network (MANET) consists of a numerous mobile nodes which are interacting with each other to establish the end-to-end communication through dynamic paths. It's a self-configuring network formed by mobile devices connected via wireless links. The MANET devices are free to move independently in any direction and so they change link to other devices frequently. In a wireless network, all the messages sent between the client and server is subject to delay of the network, as a result download delays that are significantly noticeable in mobile devices and data replication is important as it diminishes conflict in the network system, and it increases the possibility of the nodes receiving the anticipated data. The most important thing is to maintain the data consistency among the server and client-cache. To reduce the contention in the network, cache consistency algorithms were developed. Some of them are server invalidation [1, 2], client polling and time to live algorithm (TTL) [1, 10, 11, 12]. In web server invalidation, the web server normally sends each updates to the client side. In client polling, the update is sent according to the schedule. At regular intervals the server sends update to the client. Normally, A Web server cache checks if the requested data is accessible in its local storage, assuming this is the case, a response is sent back to the client with the requested data; otherwise the cache forwards the request on behalf of the user to either another cache or to the original server [5].

By introducing cache consistency algorithms, numerous duplicates of a same item are made and put away in different caches everywhere throughout the Internet. The estimation of reserve is extraordinarily diminished if cached duplicates are not overhauled when the first data change.

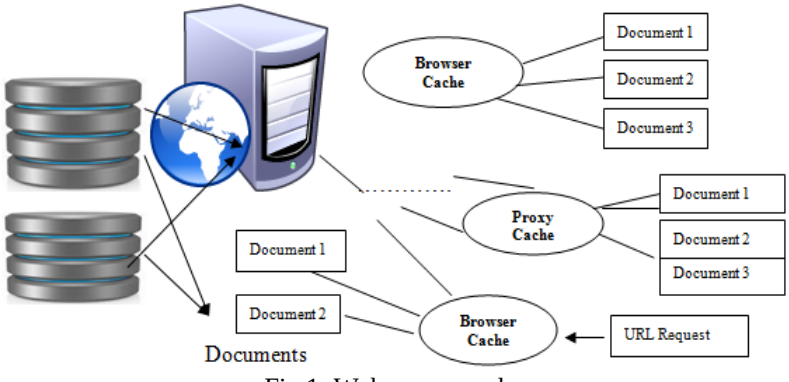


Fig 1: Web server cache

Cache consistency calculations [13] guarantee that cached duplicates of information are in the long run overhauled to keep consistency with the original information. A perfect cache consistency result will implement the consistency to the most extreme degree, while decreasing the system data bandwidth utilization and server load. Under TTL approach, every item is allotted a period to-live esteem, which is an evaluation of the object's lifetime, after which it should change. At the point when the TTL lapses, the information is viewed as invalid, and the following solicitation for the item will bring about the article to be asked for from the first server. TTL-based methodologies are quite easy to implement [12, 13], the "terminates" header field in HTTP format [13].

The Web server [Fig 1] is in charge of staying informed regarding the duplicate of information. Once the information is altered on the server, the server conveys refutation message to each one of those reserves that keep the duplicate. Invalidation guarantees document freshness [12, 13].

Table [1] Results from "Maintaining Strong Cache Consistency in World-Wide Web" by P. Cao & C. Liu [13]

Trace	SASK, 51471 requests		
Modification	1148 files modified		
Approach	TTL	Polling	Invalidation
Hits	16456	16565	16268
Get Requests	35015	34906	35203
If-Modified-Since	922	16565	0
Reply 200	35388	35689	35203
Reply 304	549	15782	0
Invalidations	0	0	6028
Total Messages	71874	102942	76434
File Xfer bytes	185MB	187MB	183MB
Ctrl Msg bytes	3.91MB	7.09MB	4.29MB
Messages bytes	189MB	194MB	187MB
Stale Hits	< 410	0	0
Avg. Latency	0.124	0.138	0.134
Min Latency	0.010	0.039	0.010
Max Latency	32.1	12.2	107
Server CPU	26.0%	30.2%	27.6%
DISK RW/s	.37;2.2	.41;2.3	.41;2.5

Trace	SDSC, 25430 requests		
Modification	57 files modified		
Approach	TTL	Polling	Invalidation
Hits	4907	4907	4905
Get Requests	20523	20523	20525
If-Modified-Since	239	4907	0
Reply 200	20535	20549	20525
Reply 304	227	4881	0
Invalidations	0	0	248
Total Messages	41524	50860	41298
File Xfer bytes	263MB	263MB	263MB
Ctrl Msg bytes	2.39MB	3.38MB	2.36MB
Messages bytes	265MB	266MB	265MB
Stale Hits	< 14	0	0
Avg. Latency	0.16	0.173	0.165
Min Latency	0.010	0.038	0.010
Max Latency	12.2	12.2	12.2
Server CPU	34.1%	35.6%	32.7%
DISK RW/s	.94;2.3	1.4;2.0	1.0;2.2

When the cache gets demand from end-user, it surveys the server to confirm if the information it reserves is still fresh, subsequently likewise ensures freshness. Conceivably there will be a considerable measure of message exchanges. Given a short report lifetime and continuous requests from the client, this is applied [Table 1] [13].

The test in supporting this methodology lies in selecting a fitting TTL value. Under this methodology customer surveying, the customer occasionally returns with the server to figure out whether reserved items are still legitimate. A typical algorithm is called Update Threshold. The upgrade threshold is communicated as a rate of the object's age. For instance, consider a stored document whose age is 30 days and the overhaul threshold is set to 10% [13].

II. BACKGROUND STUDY

Smart server update methodology is a server based issue [1]. Previous systems were not aware of what data is available in which cache node. There was loss of data because of node disconnections. And moreover if the update rate of the server is high then the node request rate needless traffic may occur.

A. CACHE CONSISTENCY IN SMART SERVER UPDATE MECHANISM (SSUM) [1]:

Cooperative and Adaptive Caching System for MANETs (COACS) architecture [1, 2, 14, 5] has been applied in SSUM [1, 2], there are two special nodes named as cache node and the query directory. But this system did not implement the concept of cache consistency and data replication. Hence, several improvements are required to enhance the performance of the data consistency in mobile network. Such as, making the server to be aware of the cache distribution in the wireless network, making the cache data item consistent to the version at the server, making the cache update rate and the server update rate equal reduces traffic in the network.

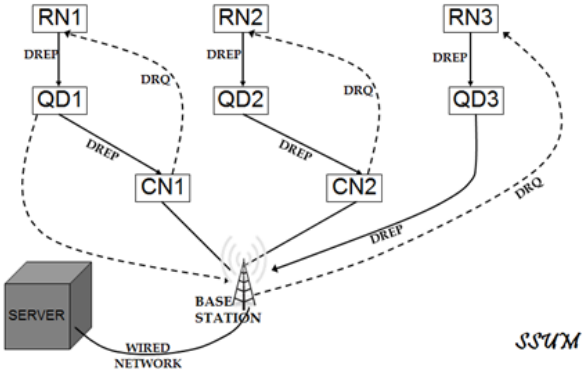


Fig 2: Smart servers update mechanism (SSUM) [1]

In the same instance, the server request if it is quite fast and update rate is slow then too many requests may queue up. To finalize the problem the update rate and the request rate are made equal. In the scheduled work here first a query is sent from a request node. The request is then passed to the Query Directory. The query directory maintains a small table in which all the cache nodes and their particular data are stored. If the data is stored in the cache node automatically the data gets retrieved and is directly send to the request node [16]. If the data was not found in any of the cache node the request is sent directly to the Server. From the server the data are retrieved and used for further purpose. During the second phase the request node acts as the cache node for the upcoming data.

In this SSUM, using a Cooperative and Adaptive Caching System for MANETs (COACS) architecture [14] which consists of Cache Node-CN, Query Directory-QD, Request Node-RN [17]. In this server is main used to send update to the cache node caches which data item so the server must keep track on Cache Node, Query Directory. Request node request for the data the request will send to nearest Query Directory. Query Directory is use to maintain the cache node this is done by using a simple table in which entry consist of ID of data item or it may be store as query. Query Directory is used to find query directory in the cache. After finding it will redirect the request to the Cache Node which has the particular data. If the request data in the nearest query directory then the data will be directly send to Request Node. If the data is not available in the nearest query directory, then it send request the next query directory likewise all the Query Directory in the network will be searched, then the request will send to the server the server will send the required data to the Request Node. This Request Node will act as cache node of other Request Node which is searching for the same data.

B. Request Node and Query Directories:

A node that wishes information sends its request to its closest Query Directory. On the off chance that this Query Directory discovers the question in its cache, it advances the request to the Cache Node, which thus, sends the data item to the Request Node. Else, it advances it to its closest Query Directory, which has not got the request yet. On the off chance that the request navigates all Query Directory's without being discovered, it gets sent to the server.

C. Cache Node maintaining:

The server self-sufficiently sends information redesigns to the Cache Node's, staying informed regarding which Cache Node's store which data items. This is done utilizing a straightforward table as a part of which a section comprises of the id of a data item and the location of the Cache Node that stores the information. On the off chance that any solicitation is gotten by the CN storing the data item, it sends the data item to the requested node.

D. Server Maintaining:

Server updates suspends in SSUM [1] once it seems that they are needless. The tool needs the server to maintain the rate of updates, and the degree of Request Node requests. To avoid cache updates to offline devices, Remove Update Entry Packets [1] is sent to the server.

E. Maintaining Query Replacement and Node Disconnection:

The most critical issue is to abstain from sending the Cache Node upgrades for information that have been either erased or supplanted, or sending the information to the Cache Node that has went out of system. To maintain network traffic and diminish system activity, store overhauls can be ceased by sending the server Remove Update Entry Packets [1]. For example, if the Cache Node went out of the network, the Query Directory, which attempts to send request and it fails immediately the Query Directory which will set the address of the queries whose item are cached by Cache Node which went out of the network are sent to -1 location, and it will send to the server which contain the Identification Code (ID's) of the query.

The server will change address of the cache node to -1 that is it will only has the reference of the queries, and it stop sending update to the Cache Node. If a new node enters into the network, the server will know about the node when it made request and caches one of these item, sometimes Cache Node runs out of space it uses replace mechanism with newly entering nodes and also tell the Query Directory to cache the data according to the query. The Query Directory sends to server to discontinue sending update for ID's for forthcoming. In case, the cache node again enters into the network after disconnection, it will send CICP-Cache Invalidation Check Packet [1], to each of the Query Directory that holds the data according to this CACHE NODE. The QD which receives CICP checks for the items to tell whether it is cached by another node in the network and send CIRP-Cache Invalidation Reply Packet to cache node containing all item not cached by other nodes [1], and it maintain the data which is not cached by other node from the network and it will get update for those data by sending CIRP. By this scenario [Fig 3], Query Directory disconnection and reconnection do not change the cache of the cache nodes and hence the server holds to the cache node remains valid.

F. Invalidation reports & Cost Overhead:

Server based methodologies for the most part uses invalidation reports that are consistently shown by the server. An invalidation report for the most part conveys IDs of the information thing which are overhauled. Whenever the query is created, the node will sit tight for intermittent invalidation report to nullify its reserve and it answers the inquiry on the off chance that it has legitimate thing. At some point the asked for thing may be invalid, it will sit tight for intermittent invalidation report, or as in proposed plan.

In existing framework because of intermittent invalidation substantial postponement happen due to higher redesign rate. So more investigates were made around there for diminishing the time interim between the redesigns or decreasing

the procedure of sending update rate less static in the enhanced invalidation report strategy were executed to contrast SSUM [1] with the time between two sequential refutation reports was isolated into interims. At the beginning of every interim, the UIR [1] will be telecasted by the server, which will be containing the IDs of information things which are overhauled following the last invalidation report.

This decreases the query inactivity time since that must be answer a query holds up until the following UIR to check whether the thing has been redesigned as opposed to holding up until the following invalidation report. The UIR interim may be balanced powerfully as indicated by the normal solicitation rate of the network system. In any case, the inconvenience in this is little overhauling rate however node needs to listen to the recurrence of the channel. Like this few methodologies were endeavoured to make the procedure of sending reports dynamic. At that point after this supreme legitimacy interim is presented it is ascertained by the server based on update rate [1]. In this scheme the item is invalidated after AVI expires. However, there is an improvement in size and interval of invalidation report. Hence this also has disadvantage due to the delay for many data items

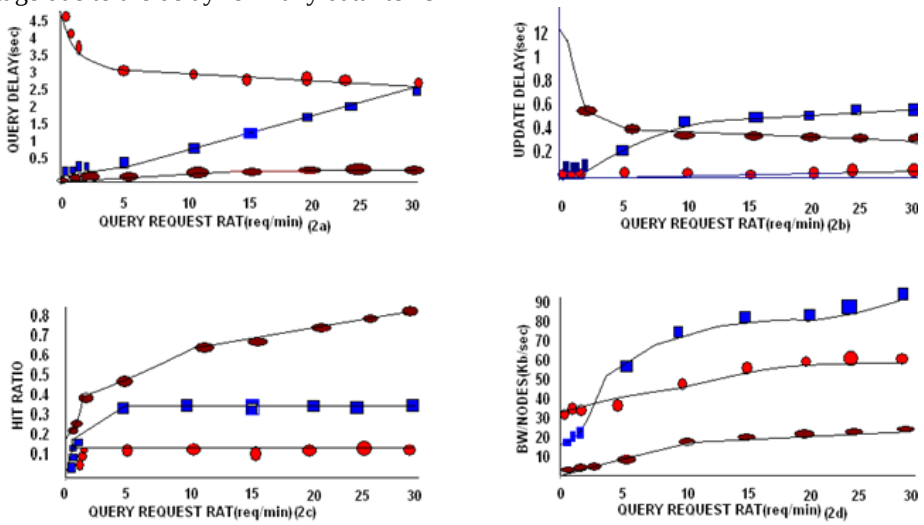


Fig 3: Query Request rate (QR) Versus Hit ratio, Query delay, Update delay and Bandwidth (BW) [1]

At the point when another node goes into the network system, the server will think about it when it made request for information this will be seen by server through Query Directory. The information which is available in the server that is stored in the system is connected with Query ID's, upgrade rate, and request rate, and the location of the reserving node reserving it [1]. Any of the additional data will cost the server around 16 bytes of additional storing per record. Thus as indicated by the capacity limit, this expense may be judged inconsequential when considering the limit of the current server. The server having interface with cache node details by using header data in the exchange packets like RUEP, SCUP, CICP, and CIRP [1, 2, 14]. This packet is used for managing the update process and also indicates the overall traffic in the network. From handling load point of view the server is mainly used to influence the update rate in the network. In common this cache consistency methodology is divided into two types [1]. Such as, Client based - In this cache node will ask for update from server and Server based- In this server send update to the cache node.

In client based query delay occur due the time at the cost of higher traffic load [Fig 3]. Nonetheless in server based there will be low traffic load at the cost of higher query delay. Cooperative is tends half way between both ends. A comparable situation can be made for the Cache Node and here its main concern the effect on the replacement frequency and cache space. For example, when using the value of 5KB data item size with the caching capacity of 200KB, for this data caching capacity the Cache Node can cache 40 data items. Only 8 bytes are needed for addition overhead required for storing the update and the request, so the overhead cost of storing the request and update rate is less than 0.16 percent of the available space. So cache node has only the minimum impact by this additional information and because of this the frequency of cache replacement will not upturn in either way [1].

III. DATA CACHE CONSISTENCY SCHEME AND EVALUATION

In the existing system the update rate is higher than the request rate due to this traffic in the network increased and, data mess occur. We can avoid this server should by stop sending update form the server for unnecessary data. For doing this the server has to observe the local updates (Ru), and also the rate of the request node request (Rr), for each of the data item. When each time the Cache Node receives update from the server, it analyses Ru/Rr and compare with its threshold T. If T is greater than or equal then the data item will deleted from the cache node and sends EDP-Entry deletion packet to the Query Directory. The cache node will also include in the header and a value of Ru, which states the Query Directory that the data has been eliminated because of high update to request ratio.

A. Algorithm

REQUEST NODE:

- ➔ Request Node request for data using data request packet to query directory
- ➔ if data available in Cache Node

Then data reply packet is send to Request Node

➔ else if

Send the request to the Server

➔ Getting data reply packet from server

if don't cache(cache)=false

Then this node will be the Cache node

end if

new_cache=true

end if

QUERY DIRECTORY:

GETTING THE REQUEST

➔ if don't _ cache =false

Then query is not present in cache

else

check in the other nearest query directories

➔ if data not present in Query Directories

then send request to server

end

GETTING ENTRY DELETION PACKET TO QUERIES

➔ if cache_ keep field in the header is true

then keep the entry in the query

else

delete cache entry from query

end if

GETTING CACHE INVALIDATION CHECK PACKET FROM CACHE NODE

➔ for each of the data in cache invalidation check packet

➔ if address of cache node data=-1

send cache invalidation check packet to data id

➔ end if

➔ end for

send cache invalidation check packet to cache node

➔ if cache node not in network

send remove update entry packet to the server

➔ Query comes to online

Delete all queries from query directory

CACHE NODE:

➔ Change data to store keep cache

➔ if keep cache =false

Send entry deletion packet to query directory

GETTING DATA UPDATE FROM SERVER AT TIME Tj

➔ If data sent to the request node in the past Tj-Ti ms

Send details about data to this request node

➔ Then the check routine is called

GETTING DATA REQUEST PACKET FOR DATA FROM QUERY DIRECTORY OR REQUEST NODE

➔ if query directory and cache nodes is waiting for cache invalidation reply packet from server

➔ then

add the data request packet to be in wait list

➔ else

send to request node

➔ if from request node

then return

➔ end if

request rate Rr is updated

➔ check routine is called

CHECK ROUTINE

➔ if Ru/Rr

keepcache =true

send entry deletion packet to query directory

deletes cache entruy for data

➔ end if

CACHE NODE COMES ONLINE

➔ Cached data items are sorted according to query directories

➔ Send cache invalidation check packet to each of above query directory

GETTING CACHE INVALIDATION REQUEST PACKET FROM ALL QUERY DIRECTORIES

➔ Send cache of all data in cache to server

GETTING CACHE INVALIDATION REQUEST PACKET FROM SERVER

➔ Update cache

➔ Respond all data request packet in waiting list

SERVER:

GETTING DATA REQUEST PACKET FROM QUERY DIRECTORY

➔ Compute the result for queries

GETTING REMOVE UPDATE ENTRY PACKET

➔ Delete update entry for data

GETTING CACHE INVALIDATION CHECK PACKET FROM CACHE NODE

➔ Check for version of data

➔ if update is needed

Send cache invalidation reply packet to data

➔ end if

➔ end for

send cache invalidation packet to cache node

updation to data

update Rr

➔ if update entry for data

Send data with time t to cache node

➔ else if Ru/Rr

Respond to data to first request node requesting it with new cache

➔ end if

Algorithm 1: Data cache consistency scheme for data replication by query delay, query update [1]

B. Client Query Model:

At the same time, many processes are running at the server and hence it sends update to the cache node through unicasts, due to this there is a time gap between when an update occurs and at what time the cache node receives the updated data item, let say "DI" is a data item. If a cache node receives the request for Data Item (DI) during this time, it will send a duplicate copy of DI to the request node. In this design uses the time stamp will be maintained for each update send by the server.

For example, if the time stamp is sent with DI is Ts and the time of getting DI buy cache node is Tc. After getting the update the cache node checks whether it had server any request node a copy of DI from its cache in the past Ts-Tc ms. If this is true means, the cache node will send a new data request packet to request node, but at this time it include a new copy of DI.

The client query model is designated such that the node present in the network system will generate another request "CRq" seconds. After the begin of the simulation, every node in the network system will generate a request, and after CRq seconds, it checks if it not receive the response for the request made in which case, it will stop it and generates new request. For example if we select a default value for CRq equal to 20 seconds, but in order to examine the request rate, however access point is frequently used to model non uniform distribution [5]. Consistently, the server will update randomly chosen data items, will be equal to default value of 20. The default value of threshold is 1.25. The default value of disconnection of node is 1 in 2 min with period of 10 seconds, after the node re-enter the network system [1].

C. Comparisons and Performance evaluation

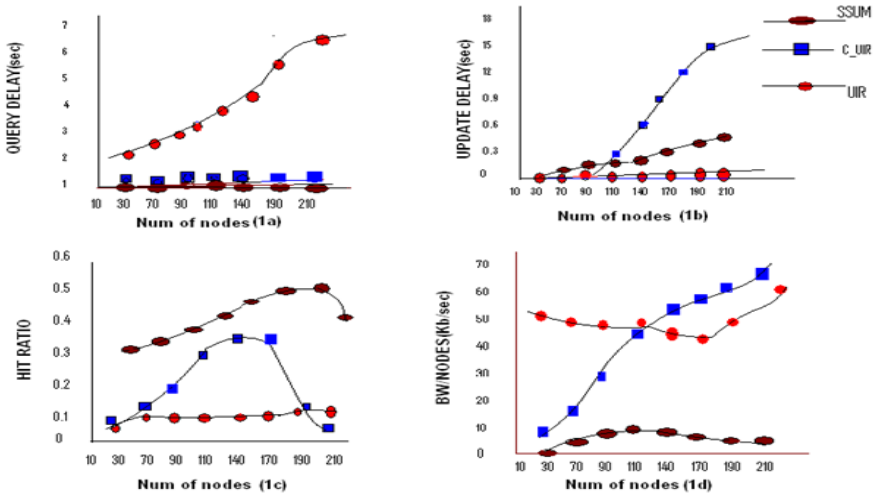


Fig 4: Number of nodes Versus Hit ratio, Query delay, Update delay and Bandwidth (BW) [1]

In this two other systems is simulated in addition to SSUM they are UIR and C_UIR [1]. The version of UIR is C_UIR that are used in top of COACS. Proposed system has more improvement then the existing system. One of the differences is in the existing system the number of data item in the server was set to 1000 but in our proposed system it was 10000. They used parameter for the "zife" parameter [1] was 1 and in the proposed it was 0.7. The size of the data is varied randomly in our simulation between 1 and 10KB, while in existing system it was fixed to 1KB. And also in proposed system the invalidation rate interval is 10 sec and replicate the UIR four times with it each invalidation report interval, where in proposed system it was 20 sec hence there is more stress on the network due to the simulation parameter.

Table 2: Performance evaluation with existing systems (SSUM, UIR and C_UIR)

SNo.	UIR, C_UIR	SSUM	SUGGESTED APPROACH
1.	MIMO Ad Hoc Networks: Medium Access Control, Saturation Throughput and Optimal Hop Distance	COACS: Cooperative Caching in Mobile Environments	Server Update Mechanism
2.	Based on Medium Access Control and Routing Table	Based on the number of cache present in a located zone	Based on Maintaining the consistency of data
3.	Routing Table is used to determine the factors	Time to Live algorithm is used.	Server Invalidation algorithm is used for updation mechanism of server
4.	Average Throughput is given by U = p4E[L]/Σi=1 piTi	-	Threshold value is given by Ru/Rr
5.	Cache not maintained	Cache not consistent	Cache fixed to 200kb. Therein 20 to 200 data items can be cached in it.
6.	-	Implemented using UIR architecture.	Implemented from COACS architecture.
7.	Number of users if assigned to 30	The number of data Item at the server was set to 1,000.	The numbers of stored items are 10,000 in our Simulations.
8.	Throughput is 80%	The parameter value is 1	The parameter value is 0.5
9.		This system occupies between 1 to 10kb parameter values.	It occupies 1kb of parameter value.
10.	Based on distance parameter	Based on time parameter.	Based on update parameter
11.	Routing table is drawn based on the system hop length and services.	A TTL value is assigned and within that limit it is refreshed four times.	A TTL value is assigned and is refreshed at correct interval to reduce stress.
12.	Nodes are seen with a static nature.	Nodes are not afforded to interchange.	Nodes are dynamic.
13.	Nodes are fixed and no variation is seen	Nodes are static but mobility is availed	Nodes are dynamic and easy access is made to maintain data.
14.	Traffic is reduced due to short distance travelling	Traffic is familiar with those systems	Traffic is reduced to a considerable extend.
15.	Routing table illustrates each distance fixed for data travel.	Here cache consistency is not maintained.	Cache consistency is maintained here using replication of data concept.
16.	Server has no knowledge about where a particular data is stored.	Server was made to transmit data by maintaining a table.	Server sends update quite often to maintain the data consistency.

Here we then analyze the performance of existing system such as SSUM, UIR, and C_UIR FROM four different perspectives [1]. Update delay and query delay are first two perspectives. In existing system the server unicast the actual data item to the cache nodes, where in proposed it broadcast the IDs of the item which are updated and not there contents. The calculation was done by recording the overall number of messages of each packet type issued received, or nodes passed by, multiplying it by the respective message size, and then divided by number of nodes and the simulation time [1, 6, 9].

D. Number of nodes and Query request rate

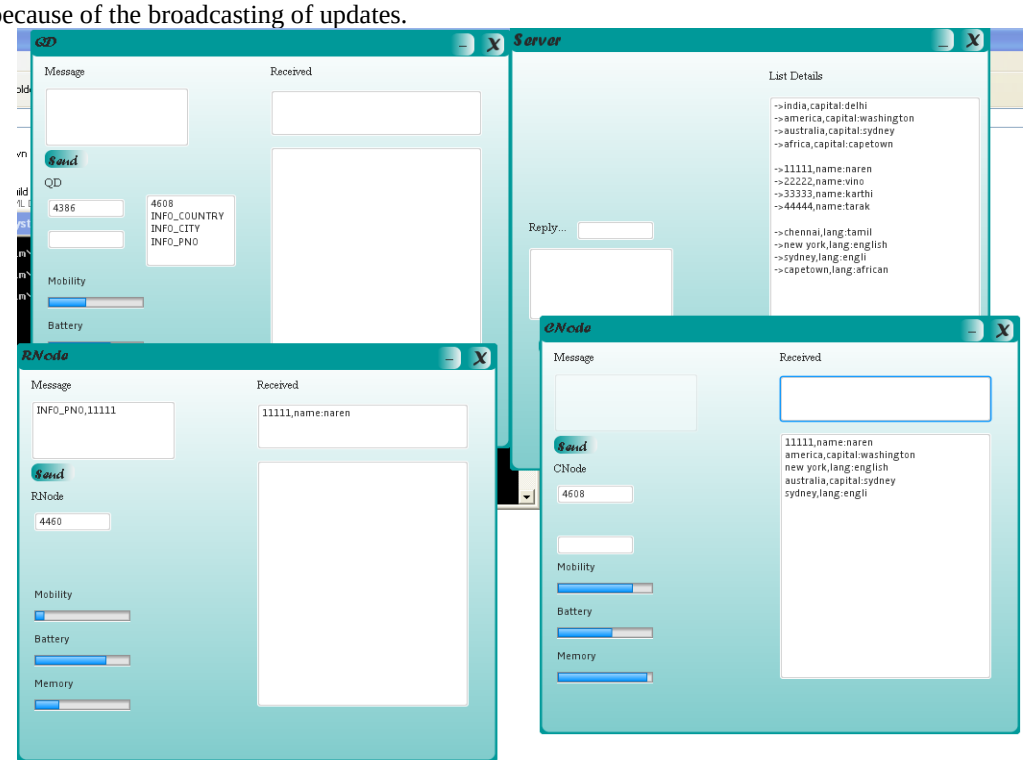
This section deals about varying the number of nodes in a fixed network. The figure shows that the query delay of UIR is much greater than that of SSUM and C_UIR [1, 6]. This is because of issued query in SSUM does not have to wait for any report in the server, it can be server directly after it is issued, this is not happened in UIR it must wait till the next UIR arrives from the server before it is getting processed. In UIR local cache miss will occur, due to this the data item must be fetched from server, but in this COACS AND C_UIR the data are searcher next searched for in the query directory system for possible network cache hit. Fig 1b shows that the update delays of UIR and C_UIR are less than SSUM when there are less than 100 nodes, because update in UIR is simply broadcasted. But when there is increase in nodes, then contention start to build up, which why the hit ratio drop rapidly and its traffic increase are shown in 1c and

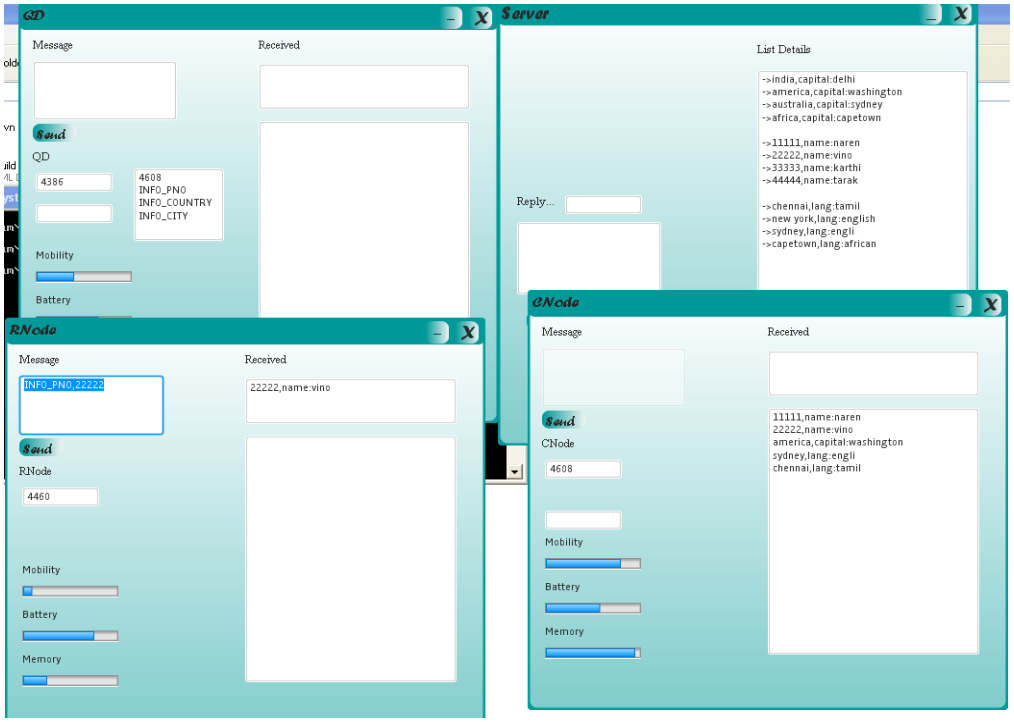
1d [Fig 5]. At last, we know the fact that the update delay presented by UIR and C_UIR is not a total delay sine it measure the time until the items ID reaches the RN/CN, but in SSUM it is the entire delay since it is the until the data item reaches the cache node.

E. Experimental results

The query delay of SSUM increases due to the increase in request rate. When queuing more packets in the node, and the update delay decrease initially and comes down to the bottom because as more item cached, then new caches are set up. This will increase the chance of having more cache nodes closer to the access point; these results to the smaller no of hops, on averages, for the update packet reach their destinations.

The network traffic is increased due to the increased update delay in C_UIR. This large traffic occur due to increase in request rate, update rate, and control packets which leads to contention in the network so there is a delay in the query and update packet of C_UIR. It also leads to decrease in hit ratio due to unsuccessful request. The hit ratio increases due to the increase in request ratio this happens because of two reasons: fist because of increase in cached queries, and the second because of contention in the network. The increase hit ratio drop the UIR query delay while its update delay is unaffected because of the broadcasting of updates.





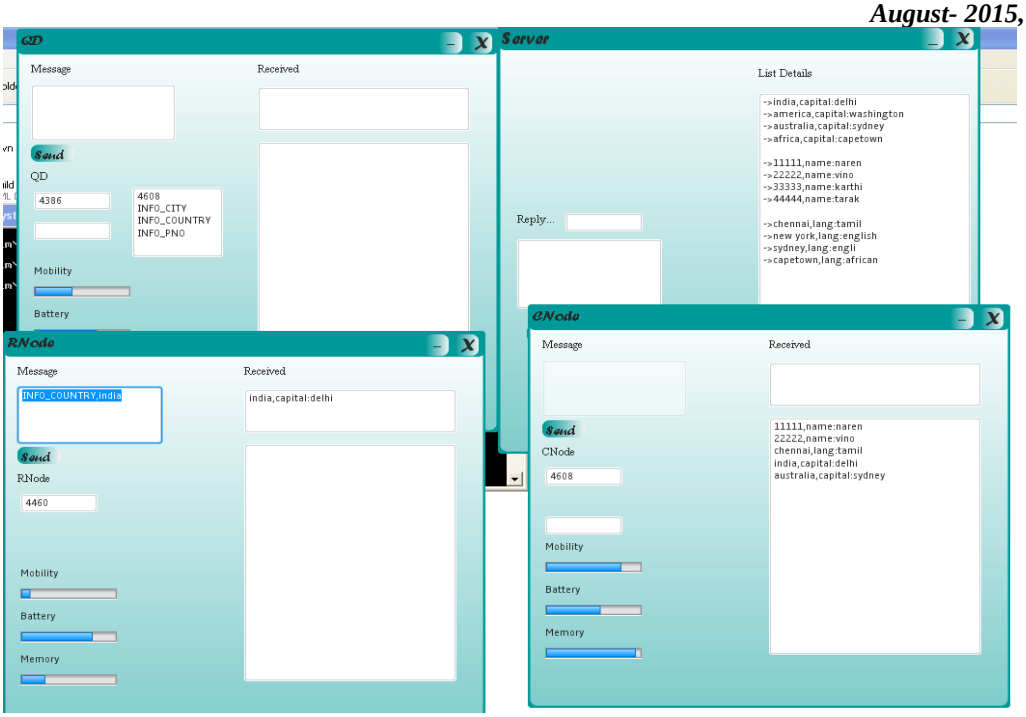


Fig 5: Simulation results

In this system a single database server is in connection with the wireless network through an access point, wherein the mobiles are spread throughout the network. The cache of the client side can hold from 20 to 200 items and the query directory can have about 600 queries that are been searched [3]. In the proposed system the server update rate increases when the traffic increases. If the response is slow it also creates traffic in the network. So to make this equalize the update rate and the sending rate are made constant. Thus wireless traffic is stopped in the network.

IV. CONCLUSSION

This paper deals about maintain cache consistency in wireless network. Our approach was based on Cooperative and Adaptive Caching System for MANETs architecture for caching data items in wireless network. In this updated invalidation report mechanism is used to analyze the performance of the system using the mathematical model. In the proposed system even if the number of nodes increases the request delay is not affected while the updating of the cache is maintained at the same rate. The chance of getting a duplicate data increases if there is a higher update delay occurrence. So in our proposed system we given a provision for keeping track of such nodes and deliver them with a fresh data within a certain time limit. So this architecture can be used in a large network where the number of nodes is high. The most important future work about this system is the security of the network. However, this paper did not deal with network security issues. For our future work we analyze all the threats for our proposed system and build necessary security resources for the system to keep the data secured.

REFERENCES

[1] Khaleel Mershad, Hassan Artail (2010) *"SSUM: Smart Server Update Mechanism for Maintaining Cache Consistency in Mobile Environments",* Mobile Computing, IEEE Transactions, Vol.9, Issue: 6, DOI: 10.1109/TMC.2010.18, ISSN: 1536-1233, pp.778 – 795, June 2010.

[2] L. Yin, G. Gao, and Y. Cai, *"A Generalized Target Driven Cache Replacement Policy for Mobile Environments", Proc. In'l Symp. Application and the Internet (SAINT 2003)*, jan 2003.

[3] J.Yuen, E.Chan, K.Lain, H.Lueng,*"Cache Invalidation scheme for Mobile computing with real-time Data",*vol.29,no.4,pp.34- 39,dec.2000.

[4] H.Maalouf and M.Gurcan,*"Minimization of the Update Response Time in Distributed Database System",* 2002.

[5] Cary G. Gray and David R. Cheriton (1989) *"Leases: An Efficient Fault-Tolerant Mechanism for Distributed File Cache Consistency"* pp.202-210 ACM089791-338-3/89/0012/0202

[6] H.Artail and K.Mershad,*"MDPF: Mnimum Distance Packet Forwarding for Search Applications in Mobile AdHocNetwork,"* IEEETrans. MobileComputing, vol.8, no.10, pp.1412-1426,oct.2009

[7] K. G.Cao,*"A Scalable Low-Latency Cache Invalidation Strategy for Mobile Environments", IEEE Trans. Knowledge and Data Eng.*, vol 15, no.5, pp.1251-1265, sept.2003

[8] P.Cao and C.Liu, *"Maintaining Strong Cache Consistency in the World Wide Web,"* IEEETrans.Computers, vol.47, no.4, pp.445-457, Apr.1998.

[9] M. Nelson, B. Welch and J. K. Ousterhout. *"Caching in the Sprite Network File System".* ACM TOCS 1988.

[10] Beomseok Nam, Kern Koh (1998) *"Periodic Polling for Web Cache Consistency",* Update Policies for Network Caches. World Conference on the WWW and Internet, 1998, Association for the Advancement of Computing in Education.

[11] Michael J. Franklin, Michael J. Carey and Miron Livny (1997) *"Transactional client-server cache consistency: Alternatives and performance"*, ACM Transaction on Database Systems, 1997.

[12] James Gwertzman and Margo Seltzer (1996). *"World-Wide Web Cache Consistency"*, International Conference USENIX, San Diego, CA, 1996.

[13] Leon Cao (2001) Consistency Control Algorithms for Web Caching available at https://cs. uwaterloo.ca/~tozsu/courses/cs748t/surveys/leon-slides.pdf February 28, 2001.

[14] Hassan Artail, Haidar Safa, Khaleel Mershad, Zahy Abou-Atme, Nabeel Sulieman, *"COACS: A Cooperative and Adaptive Caching System for MANETs"*, IEEE Transactions on Mobile Computing, vol.7, no. 8, pp. 961-977, August 2008, doi:10.1109/TMC.2008.18.

[15] Goosen, H., Boyle F (1989) *"Multilevel shared caching techniques for scalability in VMP-MC",* In Proc. 16th Int. Symp. on Computer Architecture (May 1989).

[16] I-Wei Ting , Yeim-Kuan Chang (2013) *"Improved group-based cooperative caching scheme for mobile ad hoc networks"* Journal of Parallel and Distributed Computing, doi:10.1016/j.jpdc.2012.12.013, Volume 73, Issue 5, May 2013, Pages 595–607.

[17] Fawaz K, Abbani N, Artail H (2012) *"A privacy-preserving cache management system for MANETs"* IEEE, Telecommunications (ICT), 2012 19th International Conference on 23-25 April 2012, DOI: 10.1109/ICTEL.2012.6221309, ISSN: 978-1-4673-0746-8, Pages 1-6.

