IJCAT International Journal of Computing and Technology, Volume 1, Issue 4, May 2014
ISSN : 2348 - 6090
www.IJCAT.org

An Efficient Cache Management using Adaptive Buffer Mechanism in MANET

[1] Rukmini Bhat B.

[1] Computer Science & Engineering Department, VTU, Company
SDIT, Mangalore, Karnataka. India

Abstract- In a mobile environment, as a mobile node moves from one point of attachment to another during an ongoing application it is subjected to packet loss due to network and storage capacity. Such packet loss affects the quality of ongoing communication session. Ability to control the buffer dynamically provides a reasonable trade-off between delay and packet loss, which is within the threshold limit for real-time communications. So first demonstrate that the default use of static buffers in mobile environment leads to either undesirable channel under utilization or unnecessary high delays, which motivates the use of dynamic buffer sizing. We propose an adaptive sizing algorithm which is demonstrated to be able to maintain high throughput efficiency whilst achieving low delay. We evaluate our system using NS2 simulation, and the results demonstrate the feasibility and efficiency of our proposed scheme in terms of consistency ratio, delay, and overhead.

***Keywords*- MANETs, data caching, buffering, packet loss, TTL.**

1. Introduction

Mobile Ad hoc Network (MANET) is a network with dynamic topology and mobile nodes. Due to the dynamic nature of network, there is no central control. Hence, nodes communicate with other nodes through intermediate nodes. The intermediate nodes are normal nodes in the same network and assume the responsibility of forwarding packets on the route from source to destination.

In mobile environments, data caching is essential because it increases the ability of mobile devices to access desired data, and improves overall system performance. The major issue that faces client cache management concerns the maintenance of data consistency between the cache client and the data source. All cache consistency algorithms [3] seek to increase the probability of serving from the cache data items that are identical to those on the server.

However, achieving strong consistency, where cached items are identical to those on the server, requires costly communications with the server to validate (renew) cached items, considering the resource limited mobile devices and the wireless environments they operate in.

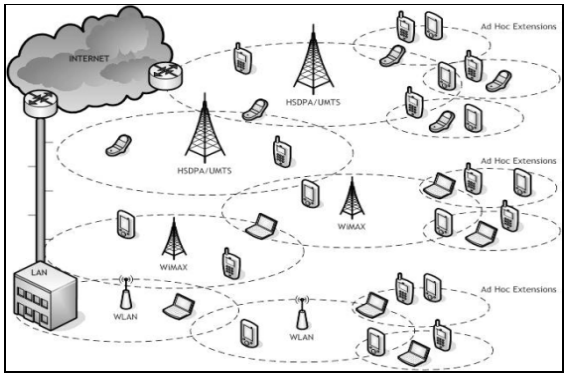

Fig.1 Mobile Nodes

We proposed distributed cache invalidation mechanism (DCIM) [2] is client based approach (pull-based algorithm) that implements adaptive time to live (TTL), piggybacking, and prefetching, provides near strong consistency capabilities. This work is implemented on top of the COACS caching architecture in order to maintain the consistency between the server and the cache node. Cache node using adaptive buffering technique so, we can decrease the dropping of packets MANET and its leads to an efficient packet routing in mobile environment.

2. Literature Review

2.1 Cache Invalidation Scheme

In a mobile computing system, in order to reduce the data access delay, some data items are cached at the client machines. Cache invalidation scheme [4] called Invalidation by absolute Validity Interval(IAVI) for mobile computing system. In IAVI, we define an absolute validity interval(AVI), for each data item based on its dynamic property such as the update interval. A mobile client can verify the validity of a cached item by comparing the last update time and its AVI. A cached

item is invalidated if the current time is greater than the last updated time plus its AVI. With this self-invalidation mechanism, the IAVI scheme uses the invalidation report to inform the mobile clients about changes in AVIs rather than the update event of the data items. As a result ,the size of the invalidation report can be reduced significantly.

2.2 Buffering Techniques

Several buffering techniques are available for efficient packet routing in mobile environment. In static buffering we will declare the buffer space for the data items. If we declare more space than we need, we waste space. If we declare less space than we need, we won't get desired packets. To avoid the drop packets in mobile environment we are using dynamic buffer techniques [5] and [6]. In this buffer size can grow and shrink to fit changing data requirements. It means when additional buffer requires can access from other node or can de-allocate whenever we are done with them, because of this reason probability of dropping packets will decreases.

2.3 Time-To- Live

Many TTL approaches [7] are introduced for MANETs and several mechanisms are introduce for assigning TTL values. First mechanism is adapted by the last update time. This approach is used in dynamic environments. Second approach is the value is calculated by the difference between the query time and last update time. This approach does not give the absolute solution. Finally the TTL value is computed by TCP orientation. Assigning the TTL value by two ways, one is to fix TTL that mean the constant TTL value is assign for a set of item in the cache. Another is adaptive TTL that provide higher consistency along with lower traffic. TTL is a data which stores in cache will be expired if it has not used within threshold time since last update. It provides simplicity, good performance and flexibility to assign TTL values.

2.4 Cooperative and Adaptive Caching System

The idea is to create a cooperative and adaptive caching system that minimizes delay and maximizes the likelihood of finding data that is cached in the ad hoc network, all without inducing excessively large traffic at the nodes. In COACS [1] System , the RN request the particular data item to the nearby QD, the QD maintains the table consist of the id of data item and the address of the cache node which contain the desired data item, if the requested data item is in the particular query directory then it forwards the request to the cache node that contains the desired data item and CN replies to the RN. If the request is not in the QD then it forwards to the nearby QD , These is done by using the MDPF algorithm. The requested data item is not in any of the QD then the message is forwarded to the server and the server reply to the RN directly shown in Fig.2.

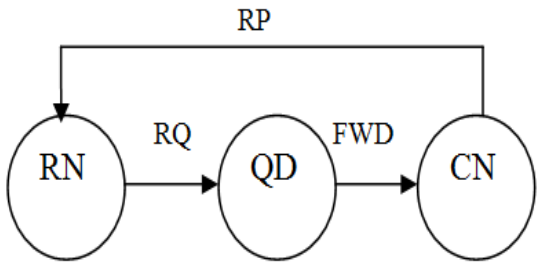

Fig.2: Data Flow

3. System Methodologies

3.1 Existing System Design

In the previous paper introduce server based scheme (SSUM)[8]. It is the process of updating the cache data item by the data source. Here the request gets from the requestor directly to the data source; the data source does not maintain the information about the client nodes and it forwards the request to the all the cache nodes and the nodes update the similar data to the data source. Because of static buffer data source get overhead processing and it leads to network delay and packet loss.

3.2 Proposed System Design

Distributed cache invalidation process, it's totally a client based scheme. Here the Query directory collects all the cache node information along with address in it's the group, and the cache nodes caches the query from the request nodes. In this paper exposes the algorithm called pull-based that implements prefetching, piggybacks [12] and TTL values [7] and [9]. Each data items in the cache nodes has the TTL values which the value same to the data source then the data item is unexpired. Suppose the data item is not similar to data source then it treated as expired. So the caches ask to the data source for updating and prefetch [11] the desired data item from the data source. Cache nodes contains the already asked frequents data. To avoid the overhead processing in data source we propose adaptive buffering technique for Cache Nodes that are present in DCIM system.

Here Cache Node containing previously requested items and their TTL values along with that dynamic buffer. This buffer size can grow and shrink to fit changing data requirements. So we can avoid the packets dropping in mobile environment.

4. Architecture and Operations

This section describes system model and interaction between different components.

4.1 System Model

The system consists of a MANET of wireless mobile nodes interested in data generated at an external data source connected to the MANET using a wired network (e.g., internet) via Wi-Fi Access Points (APs). Nodes that have direct wireless connectivity to an AP act as gateways, enabling other nodes to communicate with the data source using multihop communications showed in fig. 3.

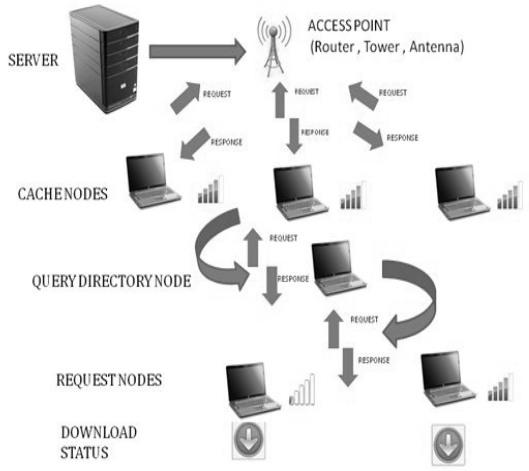

Fig.3: System Architecture

The server acts as the vital source of information to which requests are made to receive and update data. In the above figure it is evident that one of mobile nodes has a weaker connectivity range. Hence in order to overcome the delay and data loss, it forwards the requests through the intermediate node query directory and cache node. Cache node maintains the frequently used data for a temporary memory instance while the query directory maintains the information about those caches maintained in the cache nodes and efficiently directs them to the requesting node. Access point determines the connectivity range of the mobile node.

We list the messages which we are using in DCIM (see Table 1).Fig. 4 shows the basic interactions of DCIM through a scenario in which RN gives the request to the nearest QD's. If this QD finds the query in its cache, it forwards the request to the CN caching the item, which, in turn, sends the item to the requesting node (RN). Otherwise, it forwards it to its nearest QD [1], which has not received the request yet. If the request traverses all QD's without being found, a miss occurs and it gets forwarded to the server which sends the data item to the RN. The server autonomously sends data updates to the CNs, meaning that it has to keep track of which CNs cache which data items. This can be done using a simple table in which an entry consists of the id of a data item (or query) and the address of the CN that

Table 1: Packets used in DCIM

Packet	Function	Description
HELLO	New node's arrival	Broadcasting by newly arriving node
DRP	Data Request	Submitted by an RN to request data
DREP	Data Reply	Submitted by an CN or DB to RN
CURP	Cache Update Request	Sent from CN to sever to validate certain data items
SVRP	Server Validation Reply	Sent from server to CN to indicate which items are valid
SUDP	Server Update Data	Sent from sever to CN

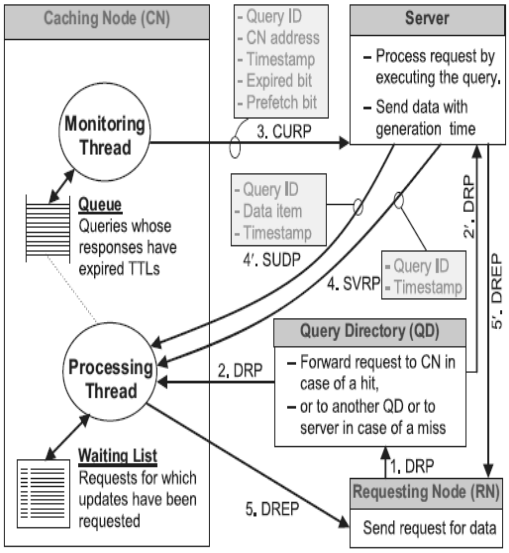

Fig. 4. Interactions between nodes in a DCIM system

caches the data. If any request to the CN caching the item, which, in turn, sends the item to the requesting node (RN).

Algorithm 1: interaction between three nodes (RN,QD and CN)

cache update()

1. *RN* send *DRP* msg to nearest *QD*
2. *QD* checks for data
3. **if** data is present **then**
4. *DRP* msg forward to *CN* (go to step 10)
5. **else if**

6. Forward to nearest *QD*
7. **else**
8. Fetch the data from sever (go to step 15)
9. **end if**
10. *CN* receives the *DRP* request from the QD and checks for the data
11. **if** the data is present
    Check buffer space in CN call **Adaptive buffer()**
12 **else**
13 Fetch the data from server (go to step 15)
14 **end if**
15 return (DERP response to RN)

4.2 Server Operation

When the server receives the CURP message from the CN. The server checks all the items are compared to the last update time. If the item has not changed then it is treated as a valid and sent SVRP is send to the CN. Suppose the data item have not similar to the server then it called as invalidation. So server send SUPR message to the CN fig 3. The server inform about this through the SVRP message. In this approach is client based, the processing at the server is minimal.

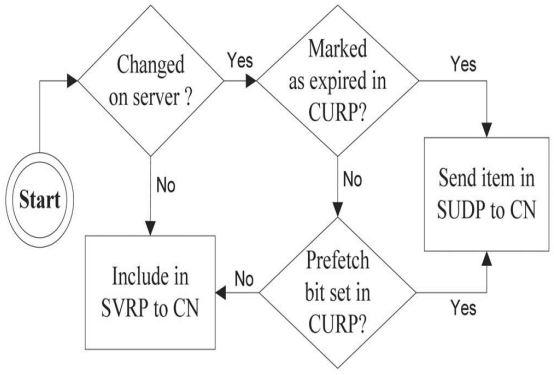

Fig. 5. Decision flow at the server

4.3 QD Operation

This section describes how the QD is assigned in this system depending on the storage capacity. Here the number of QDs are bounded in this based on the limits such as lower and upper limits, where the QD elected [2] it will not yield reduction in average QD load. The upper limits, corresponds to a delay threshold. First we elected the maximum storage capacity node as QD and its store all the address of the cache node information.

After it stores more CN information so the capacity is reduced [7]. Using the minimum distance packet forwarding concept it is easy to reach required destination and it provides efficient way to maintain the system performance.

4.4 CN Operation

The CN store the cached queries along with their responses plus their IDs. A CN maintain all the information in two tables that are cache information table [2] which stores responses of the queries are locally cached and Query information table that stores query specific data. The CN checks for expired data items, validation request, and request updates. CN collects the frequently asked data and the received the request get from the QD. CN assign TTL values for each data items and the value with the query if it is matched it as VALID response to the RN. Suppose the data does not matches then it as INVALID. Then the CN piggyback to server and collect the valid data item. The CN prefetch the item from the server.

These CN's uses the dynamic buffering technique. In the initialization phase, an equal buffer space is allocated to all cache nodes which enable every neighbor to have its share in the buffer. In addition to that, the occurrence of a request, the allocation is dynamically adjusted according to instantaneous share of neighbor's cache nodes buffer. More importantly, we also put limits on maximum and minimum buffer space a single node can occupy. In this way, we can decrease the dropping of packets MANET and its leads to an efficient packet routing in mobile environment.

4.4.1 The Algorithm

Our objective is to simultaneously achieve both high throughput efficiency and low delay. Intuitively, in order to ensure efficient link utilization, the buffer should not lie empty for too long a time. Increasing the buffer size tends to reduce the link idle time. However, to ensure low delays, the buffer should be as short as possible and a trade-off therefore exists. This intuition suggests the following approach. We observe the buffer occupancy over an interval of time. If the buffer rarely empties, we decrease the buffer size to avoid high delay. Conversely, if the buffer is empty for too long a period, we increase the buffer size to maintain high throughput.

Algorithm 2: The adaptive buffer tuning algorithm.

Adaptive buffer()

1. Set the initial queue limit, the maximum buffer limit q_{max} and the minimum buffer limit q_{min}.
2. Set the increase step size a and the decrease step size b.
3. **for** every t seconds **do**
4. Measure the idle time t_i.
5. $q_{new} = q + at_i - b(t - t_i)$.
6. **if** $q_{new} < q_{max}$ **then**
7. **if** $q_{new} < q_{min}$ **then**

8. $q \leftarrow q_{min}$
9. **else**
10. $q \leftarrow q_{new}$
11. **end if**
12. **else**
13. $q \leftarrow q_{max}$
14. **end if**
15. **end for**

5. Simulation Setup

In this section, we are going to reduce the traffic, delay and probability of dropping packet by introducing the concept of the dynamic buffer in cache node. We use the NS2 software to implement the dynamic buffering scheme in the DCIM System.

5.1 Simulation Parameter

A single database server is connected to the wireless network through a fixed access point, while the mobile nodes are randomly distributed. The simulation area was set to 670 x 670m2, populated with 8 nodes that were randomly distributed. Propagation was according to the two-ray model, and the node's bit rate was set to 2 Mbps. Mobility was based on the random waypoint model, with a maximum speed of 2 m/s (Table 2).

5.2 Result

The graphs are plotted by simulation time versus number of cache hits. Fig. 6 clearly demonstrates that the cache hit without adaptive buffer more is observer in the experimentation i.e. due to the static buffer space. Here in Fig. 7, the cache are deleted frequently due to the size constraints. Therefore proposed system uses dynamic size buffer based on the lifetime and frequency of the data. Hence the cache hits at server is observed less.

Table 2: Simulation Parameters

Simulation Parameter	**Default value**
Network size	670 x 670m^2
Node transmission range	100m
number of nodes	8
routing protocol	DSDV
max packet	50
Size of data item	10KB

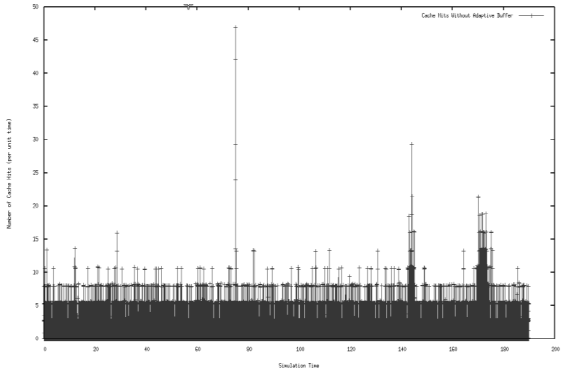

Fig.6: Cache hits without adaptive buffer

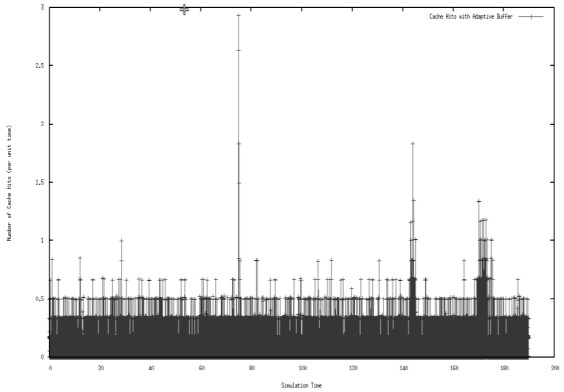

Fig.7: Cache hits with adaptive buffer

6. Conclusion

The proposed scheme can minimize dropping of packets and query delay in the distributed cache invalidation mechanism and reduces the network traffic. In addition, we have adopted the adaptive buffering technique in Cache Node to avoid the dropping of packets in mobile environment , thus improving the performance of the system and minimizes the contention in the network. The extensive results have demonstrated that, in comparison with the existing methods, our proposed scheme is more effective and efficient in accessing the data, reducing response time and improving the Performance of the system.

For future work, we can investigate more sophisticated TTL algorithms for life time of the data and selection of Query directories are dynamic so that requesting node gets the data without delay.

References

[1] 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

[2] Kassem fawaz, student member, ieee, and Hassan artail, senior member, ieee "Distributed Cache Invalidation Method For Maintaining Cache Consistency In Wireless Mobile Networks" IEEE Transactions On Mobile Computing, Vol. 12, No. 4, April 2013

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

[4] G. Cao, "On Improving the Performance of Cache Invalidation in Mobile Environments," ACM/Kluwer Mobile Network and Applications, vol. 7, no. 4, pp. 291-303, 2002.

[5] Tianji Li and Douglas Leith,"Adaptive Buffer Sizin for TCP Flows in 802.11e WLANs".

[6] Rochlani, Yogesh R., and A. R. Itkikar. "Integrating Heterogeneous Data Sources Using XML Mediator." International journal of computer science and network 3 (2012).

[7] T. Hara and S. Madria, "Dynamic Buffer control in Mobile Ad Hoc Networks," Proc. Database Systems for Advanced Applications, pp. 111-136, 2004.

[8] J. Jung, A.W. Berger, and H. Balakrishnan, "Modeling TTL-Based Internet Caches," Proc. IEEE INFOCOM, Mar. 2003.

[9] K. Mershad and H. Artail, "SSUM: Smart Server Update Mechanism for Maintaining Cache Consistency in Mobile Environments," IEEE Trans. obile Computing, vol. 9, no. 6, pp. 778-795, June 2010.

[10] X. Tang, J. Xu, and W-C. Lee, "Analysis of TTL-Based Consistency in Unstructured Peer-to-Peer Networks," IEEE Trans. Parallel and Distributed Systems, vol. 19, no. 12, pp. 1683-1694, Dec. 2008.

[11] L. Bright, A. Gal, and L. Raschid, "Adaptive Pull-Based Policies for Wide Area Data Delivery," ACM Trans. Database Systems, vol. 31, no. 2, pp. 631-671, 2006.

[12] M. Denko, J. Tian, "Cooperative Caching with Adaptive Prefetching in Mobile Ad Hoc Networks," IEEE WiMob'2006, pp.38-44, June 2006.

[13] B. Krishnamurthy, C. Wills, "Study of piggyback cache validation for proxy caches in the World Wide Web,"USENIX, Monterey, CA, December1997.

Rukmini Bhat B. Is a M.Tech Student of Computer Science & Engineering Department of SDIT, Mangalore. She graduated with BE (Honours') in Computer Science & Engineering, affiliated to VTU university. She is currently pursuing her Masters in Computer Science & engineering at SDIT (Shree Devi College of Engineering).Her research Areas are Computer Networks, and Mobile Computing.