SAJET

South Asian Journal of Engineering and Technology Vol.2, No.15 (2016) 175–183

ISSN No: 2454-9614

Client Based TTL Method for maintaining Cache Consistency in Mobile Ad-hoc Networks

P. Rajakumar[a], M. Selvakumar[a*], R. Sivaraj[a,b] , L. Priyanga Devi[a,b,c]

[a)] *Department Of Computer Science and Engineering, K.S.R. College of Engineering, Tiruchencode, Namakkal, Tamilnadu, India.*
[b)] *Department Of Computer Science and Engineering, K.S.R. College of Engineering, Tiruchencode, Namakkal, Tamilnadu, India.*
[c)] *Department Of Computer Science and Engineering K.S.R. College of Engineering, Tiruchencode, Namakkal, Tamilnadu, India.*
[d)] *Department Of Computer Science and Engineering K.S.R. College of Engineering, Tiruchencode, Namakkal, Tamilnadu, India.*

*Corresponding Author: P. Rajakumar
E-mail: palanirajakumar@gmail.com,

Received: 17/11/2015, Revised: 28/12/2015 and Accepted: 08/03/2016

Abstract

A Mobile adhoc network (MANET) is considered a collection of wireless mobile nodes that are capable of communicating with each other without the use of a network infrastructure. In a MANET, caching is an important technique where the mobile devices can access desired data from caching nodes that increases the system performance. Distributed cache invalidation mechanism (DCIM) is a client-based cache consistency scheme that provides strong consistency capabilities using time to live (TTL), piggybacking, and prefetching mechanisms. But, whenever the caching nodes are disconnected, requesting nodes should access data from the server. In this project, replication of the most requested data is created for the nearest caching nodes and the information is provided to the nearest query directory. In case of cache node disconnections, requesting nodes can access data from the nearby caching nodes.

Keywords: Mobile adhoc network, cache, consistency, TTL, push, pull, and client based.
Reviewed by* **ICETSET'16 *organizing committee*

1. Introduction

Mobile ad-hoc network (MANET) is an independent system of mobile nodes connected by wireless links forming a short, live, on-the-fly network even when access to the Internet is unavailable. Nodes in MANETs generally operate on low power battery devices. These nodes can function both as hosts and as routers. As a host, nodes function as a source and destination in the network and as a router, nodes act as intermediate bridges between the source and the destination giving store-and-forward services to all the neighbouring nodes in the network. Easy

deployments, speed of development, and decreased dependency on the infrastructure are the main reasons to use ad-hoc network. MANET allows rapid deployment because they don't depend on a fixed infrastructure. In a MANET nodes can act as source, destination, or an intermediate router. This feature is helpful for military applications, emergency situations, group meeting where fixed infrastructures may not be available. In a mobile ad hoc network (MANET), data caching is important as it reduces contention in the network, improves the probability of nodes getting desired data from the intermediate nodes, and improves the system performance. The major problem is the maintenance of consistency between the cache client and the server. In a MANET, all messages between the server and the cache are subject to delays it decreases consistency by download delays, which are more dangerous in mobile devices. All cache consistency algorithms are developed with the goal to increase the probability of serving data items from the cache node not from the server. But, achieving consistency between cache and server requires more communications with the server to update the cached data items, considering the resource limited mobile devices and the wireless environments they operate in. Large numbers of cache consistency algorithms have been proposed in the literature, and they classified into three groups. They are,

- Server invalidation,
- Client polling,
- Time to live (TTL).

In a server invalidation technique, the server sends an Invalidation reports (IR) upon each updating to the client. Two examples of server invalidation are the Piggyback server invalidation and the Invalidation report mechanisms. In pull based, like the piggyback cache validation of a validation request is initiated according to schedule. If the copy of the data item is update, the server informs the client node that the data have not been modified; else the update is sent to the client. In a TTL algorithm, a server assigns. TTL value that is stored alongside each data item d in the cache. The data item d is considered valid until T time units since the cache update. The first request for data item d submitted by a requesting node after the TTL value expiration treated as a miss and the server fetches a fresh copy of data item d.

Many algorithms proposed to determine TTL values they are, the fixed TTL, adaptive TTL, Squid's LM-factor. TTL-based algorithms are popular due to their simplicity, good performance, and flexibility to assign TTL values for all kind of data items. But, TTL-based algorithms, like pull based algorithms, are weakly consistent, in contrast to invalidation schemes that are generally strongly consistent. With strong consistency algorithms, users are server strictly fresh data items, while with weak algorithms, there is a possibility that users may get inconsistent copies of the data. A server-based scheme implemented on top of the COACS caching architecture. In COACS, elected query directory (QD) nodes cache submitted queries and use them as indexes to data stored in the nodes that initially requested them (CN nodes). Since COACS did not implement a consistency strategy, the system DCIM provides the following improvements:

1) Enables the server to be aware of the cache distribution in the MANET.
2) Making the cached data items consistent with their version at the server.

3) Adapting the cache update process to data update rate at sever relative to the request rate by the clients.

With these changes, the overall design provides a complete caching system in which the server sends to the client's selective updates that adapt to their needs and reduces the average query response time. It is fair to assume that CNs and QDs will go offline from time to time either temporarily or permanently. In either case, DCIM should react efficiently to keep the system running: it does not attempt to proactively account for CN or QD disconnections, but rather, it reacts to these events by relying on the QDs to detect when CNs go offline. In case of a query hit, the QD will always try to forward the data request to the CN, and consequently any routing protocol will return a route error if no route can be established to the CN. This will indicate that the CN is not reachable or equivalently disconnected. In such a case, the QD instructs the RN to request the item from the external data source as it would do in a case of a data miss.

This paper proposed to Replicate the most requested data to the new cache node and provide the information to nearest QD. The replicated cache information sends to the server in order to maintain the consistency and contention. Whenever the Requesting node access data from the server, the caching node does the following functions:

1. Copies the data coming from the server.
2. Routes the packet to the Requesting node.
3. Replicates packet to the nearby caching nodes.
4. Provides information to the server about replications.
5. Election of Query directory (QD) & Caching nodes (CN) for replication.

2. Literature Review

2.1. DCIM: Distributed Cache Invalidation Method for Maintaining Cache Consistency in Wireless Mobile Networks:

Distributed cache invalidation mechanism (DCIM) is a client-based cache consistency. DCIM is a pull-based algorithm that provides strong consistency capabilities using adaptive time to live (TTL), piggybacking, and perfecting, mechanisms. Cached data items are assigned TTL values that correspond to their update rates at the server, where items with expired TTL values are grouped in validation requests to the data source to refresh them, whereas unexpired ones but with high request rates are perfected from the server. In this paper, DCIM is analyzed to assess the delay and bandwidth gains when compared to polling every time and push-based schemes.

2.2 SSUM: Smart Server Update Mechanism

SSUM is a server-based approach that avoids many issues associated with push-based cache consistency approaches. In SSUM, 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 caches the data.

As shown in Figure. 1, a node that desires a data item sends its request to its nearest QD. 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). A Server sends update reports frequently to the CN's for indicating updating in the cached data items.

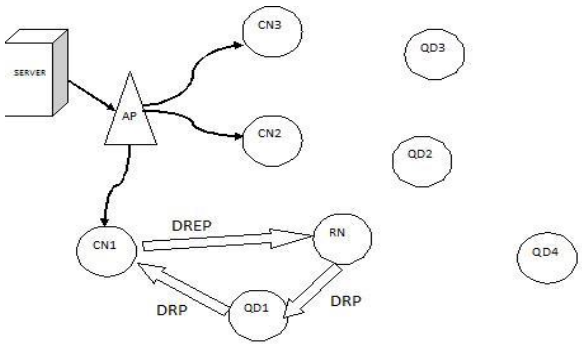


Fig. 1. SSUM Design

2.3COACS: A Cooperative and Adaptive Caching System for MANETS

COACS is a distributed caching scheme. Cache consistency is difficult in Ad-hoc networks compared to the Infrastructure based networks. COACS introduces new architecture for cache consistency. COACS consist three types of nodes: Requesting node (RN), Query directory (QD), Caching node (CN). A QD's task is to cache queries submitted by the requesting mobile nodes, while the CN's task is to cache data items (responses to queries).

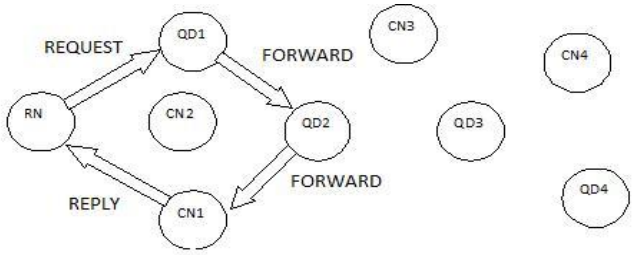


Fig. 2. COACS Design

As shown in Fig. 3, RN sends request, QD forwards that request to the CN. CN checks whether the requested data available or not. If the requested data is available then it forwards reply to the RN.

2.4 Zone based co-operative caching scheme

One hop neighbour of a mobile client form a cooperative cache zone. Mobile client share cache with its neighbours lying in the zone **[10].**

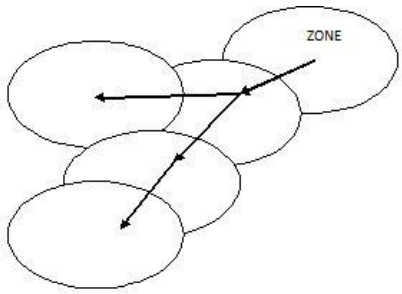


Fig. 3. Cache Discovery Process

During cache discovery process when a data request is initiated it first looks for the data item in its own cache **[10].** If there is a local cache miss, the mobile client checks whether the data item is cached in other mobile hosts within its home zone. When a mobile host receives the request and has the data item in its local cache, it will send a reply to the requester to acknowledge that it has the data item. In case of a zone cache miss, the request is forwarded to neighbour along the routing path **[10].**

3. System Model

3.1 DCI M Architecture

DCIM is a client based cache consistency mechanism that implements adaptive time to live (TTL), piggybacking, and prefetching, and provides near strong consistency capabilities **[11].** DCIM follows the COACS Architecture.

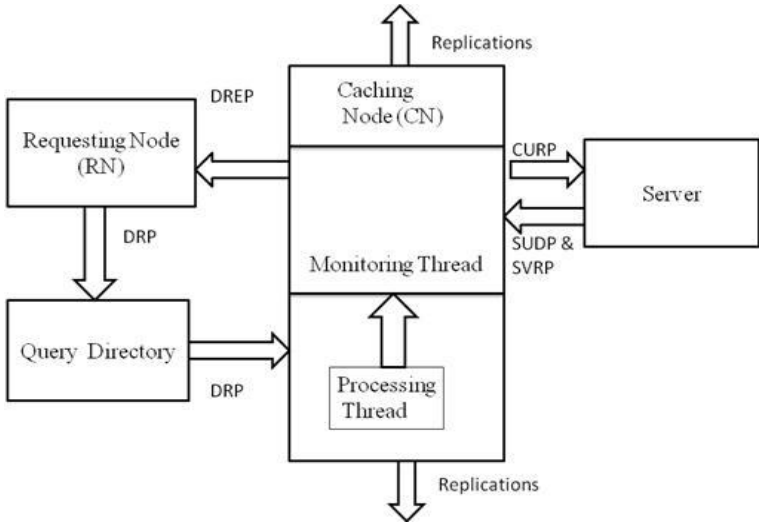


Fig. 4. DCIM Architecture

As shown in Fig. 1, Requesting node (RN) sends request using Data request packet (DRP) packet. Query directory (QD) checks whether the requested data is available in caching node (CN). Then, CN returns reply using Cache update request (CURP) **[11].** Data items are assigned adaptive TTL values that correspond to their update rates at the data source, where items with expired TTL values are grouped in validation requests to the data source to refresh them. This validation requests send using CURP. Server gives validation reply using Server update data (SUDP) and Server validation reply (SVRP) **[11].** Whereas un-expired ones but with high request rates are prefetched from the server. Since COACS did not implement a consistency strategy, the system DCIM (Distributed Cache Invalidation Method) provides the several improvements: Enabling the server to be aware of the cache distribution in the MANET, Making the cached data items consistent with their version at the server, Adapting the

cache update process to the data update rate at the sever relative to the request rate by the clients, With these changes, the overall design provides a complete caching system in which the server sends to the clients selective updates that adapt to their needs and reduces the average query response time **[11].**

3.2 TTL Calculation

The CN polls the server frequently to know about the update times of the items it caches. It also piggybacks requests to refresh the items it caches each time it has reason to contact the server, basically whenever an item it caches expires. Nevertheless, to avoid unnecessary piggybacks to the server, the CN utilizes a two-phase approach. Specifically, at the end of each polling interval ðTpollÞ, every CN issues validation requests for the items it indexes that have expired TTLs and have a high request rate. After a configurable number of polling intervals, denoted byNpoll, theCNissues a validation request for all the items it caches if at least one item has an expired TTL regardless of its request rate. We refer to the interval Npoll _ Tpoll as the piggyback interval, Tpigg. When the server receives a CN's request, it replies with a list of updated as well as nonupdated items. The CN uses this information to adapt the TTL values to the server update rate for each item. The effectiveness of these mechanisms is explained in the experimental evaluation section. Although in principle it achieves weak consistency, DCIM can attain delta consistency when at least one item has a TTL expired by the end of the piggybacking interval, thus effectively causing validation requests to be issued periodically. Hence, the CN ensures that data items are at most one piggybacking interval stale. Fig. 1 shows a scenario for illustration purposes where two CNs are sending cache validation requests to the server (dotted arrows) via gateway nodes and through the Access Point. The server replies back with lists of valid and changed data items (short-dashed arrows) to the CNs, which in turn update the corresponding QDs asynchronously about the items they cache (long-dashed arrows).

4. Proposed Model

4.1 Election of QD & CN

In our existing method only one caching node is allowed to cache for a particular data. In this method if the request rate for the particular data is high then there is more number of hits in a particular QD & CN for that data. It also increases the traffic in the network. In order to avoid this we implement the concept of cache replication.

4.2 Cache Replication

In Cache replication the data which have the request rate higher than the threshold value is distributed to more than one caching node. The threshold value is set on the basis of the network condition. For the higher traffic network we set the Threshold as minimum and for the lower traffic network we tune the threshold to the lower value. The Threshold values are set as 0.50 for normal networks and as 0.75 for congest networks. While distributing the data to the other cache we have to consider the following. The replication cache cannot come under the same QD which stores the information about the present caching node for that data in order to avoid the network congestion in that QD. Basically every time when a node makes a request for a data the ratio Ru/Rr is calculated. When the ratio within the replication's threshold value then the data is allowed for the replication. Once the ratio

reaches below 0.75 the caching node send the request to the server for cache replication and get the permission for the replication. After a new node get that particular data it is allowed for caching that data by sending QCRP message to the new node.

4.3 Data Replication & Intimation to Server

This module is to replicate the most requested data to the new cache node and provide the information to the nearest QD. The replicated cache information is send to the server in order to maintain the consistency and contention. Once a new node get the permission to become a cache node it give the intimation to the server by sending the SCUP message to the server. After verifying the ratio the server grants the permission for the new caching node. Now the new caching node sends the QCRP message to the nearest QD. The QD verifies that there is entry for that particular data in its list. If it so it send the request to their nearest QD to make the entry for the replicated caching node otherwise it add the entry in its list in order to avoid congestion.

4.4 Simulation Results

At different threshold levels of queries the query delay will be reduced and the query delay with respect to different threshold levels is shown in Fig. 2.

At different threshold levels the update delay in the cache node is also reduced when compared to the existing methods and it is shown in Fig 3.

Fig. 5. Query Delay Vs Threshold Rate

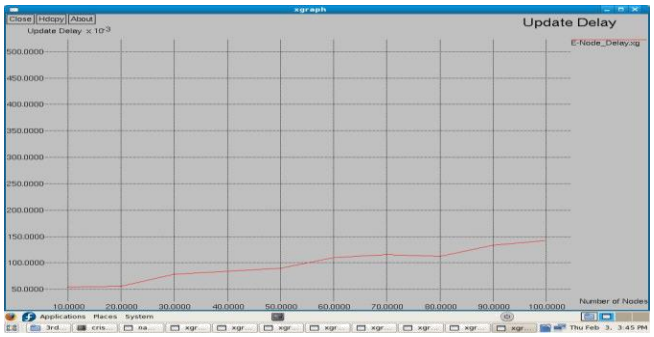
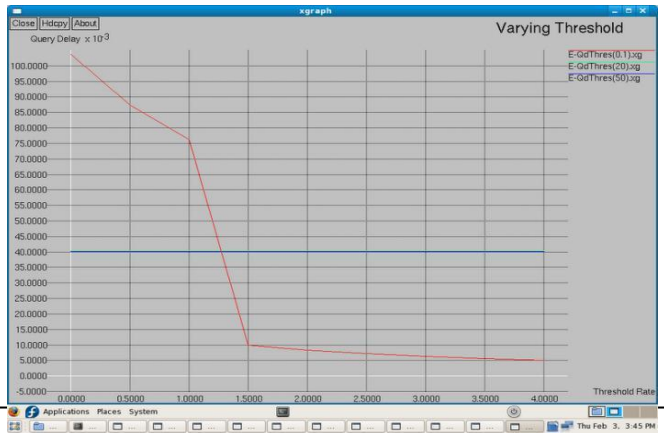


Fig. 6. Update Delay Vs Threshold Rate



Even though we increased the number of nodes the update delay is checked and it is more efficient when compared with the existing methods and it is shown in Fig 4.

Update delay with respect to the query rates were calculated and graph is plotted. The update delay is very much reduced and is shown in Fig. 5.

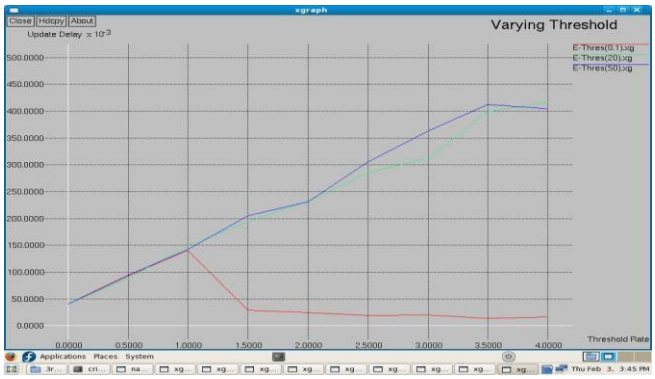


Figure 4. Update Delay Vs Number of Nodes

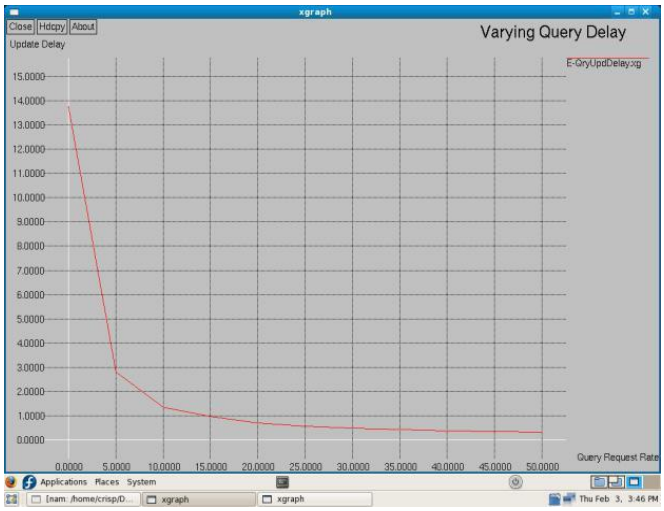


Figure 5. Update Delay Vs Query Request Rate

5. Conclusion

DCIM (Distributed Cache Invalidation Method) provides a stronger cache consistency than the other client based schemes. But it does not concentrate on CN disconnections. Replication among the nearest caching nodes will improve the performance of DCIM. On caching node (CN) disconnections requesting node (RN) need not go for the server, it can access data from the other caching nodes. Replication is distributed to nearby nodes. In case of cache node disconnection data can be accessed from the nearby caching nodes.

References

[1]. K. Fawaz and H. Artail, "DCIM: Distributed Cache Invalidation Method for Maintaining Cache Consistency in Wireless Mobile Networks", IEEE Transactions, Mobile Computing, Vol. 12, No. 4, April 2013.
[2]. 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.
[3]. T. Andrel and A. Yasinsac, "On Credibility of MANET Simulations," IEEE Computer, vol. 39, no. 7, pp. 48-54, July 2006.
[4]. J. Cao, Y. Zhang, G. Cao, and X. Li, "Data Consistency for Cooperative Cache In Mobile Environments," Computer, vol. 40, no. 4, pp. 60-66, 2007.
[5]. 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./Oct. 2003
[6]. 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.
[7]. Y. Sit, F. Lau, and C-L. Wang, "On the Cooperation of Web Clients and Proxy Caches," Proc. 11th Int'l Conf. Parallel and Distributed Systems, pp. 264- 270, July 2005.
[8]. 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.