IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 4, April 2016

A Pull Based Method for Maintaining Cache Consistency in Wireless Mobile Networks

T.Jeevanantham[1], C. Suresh Kumar[2]

M.E. Mobile and Pervasive Computing, Anna University, Trichy, TamilNadu[1]

Teaching Fellow - CSE/IT Department, Anna University, Trichy, TamilNadu[2]

Abstract: Distributed Cache Invalidation Method (DCIM) proposed a customer based reserve unwavering quality plan. It is executed on the top of a formerly proposed structural planning for reserving information things in portable impromptu systems (MANETs). In particular COACS, unique hubs store the inquiries and the locations of the hubs that store the reactions to these questions. Even though SSUM is proposed in order to provide a server-based consistency plan, DCIM provides service that is absolutely customer based. DCIM is a force based calculation that actualizes versatile time to live (TTL), piggybacking, and prefetching, and gives close solid consistency capacities. Reserved information things are relegated versatile TTL values that compare to their redesign rates at the information source. The things with lapsed TTL qualities are gathered in approval solicitations to the information source. In order to invigorate them, while unexpired ones however with high demand rates are pre-fetched from the server. In this paper, DCIM is broke down to evaluate the postponement and transfer speed picks up (or costs) when contrasted with surveying each time and push-based plans. DCIM was likewise executed utilizing ns2, and thought about against customer based and server-based plans to evaluate its execution tentatively. The consistency proportion, deferral, and overhead activity are accounted for versus a few variables, where DCIM appeared to be better when analyzed than alternate frameworks.

Keywords: pre-fetch, cache memory, piggy-backing, TTL, DCIM, query direction.

I. INTRODUCTION

The goal of DCIM is to improve the efficiency of the cache updating process in a network of mobile devices which cache data retrieved from a data server, without requiring the latter to maintain state information about the caches.

The proposed system is pull-based, where the CNs monitors the TTL information and accordingly triggers the cache updating and validation process. DCIM is scalable by virtue of the CNs whose number can increase as the size of the network grows (each node can become a CN for an item it requests if not cached suitable to dynamic MANETs than a push-based elsewhere in the network), and thus is more alternative since the server does not need to be aware of CN disconnections. DCIM is also more suitable when data requests are database queries associated with tables and attributes.

In a push-based approach, the server would have to map a cached query to all of its data sources (table attributes) and execute this query proactively whenever any of the sources is updated.

Moreover, DCIM adapts the TTL values to provide higher consistency levels by having each CN estimate the inter-update interval and try to predict the time for the next update and sets it as the item's expiry time.It also estimates the inter-request interval for each data item to predict its next request time, and then pre-fetches items that it expects to be requested soon.

In DCIM, the caching system relies on opportunistic validation requests to infer the update patterns for the data items at the server, and uses this information to adapt the TTL values. Describe the operations of DCIM in details, but first, we list the messages which we added in DCIM to those already introduced in COACs. The reader can refer to [2] for a complete description of all the COACS messages. the basic interactions of DCIM through a scenario in which an RN is submitting a DRP (Data Request Packet) for a query indexed in the QD. The QD forwards the DRP to the CN caching the item assuming there was a hit. At the CN, the requested item may be in the waiting list at the moment if it is being validated.

Validation requests are issued by CNs using CURP messages. Each entry in the message consists of the query associated with this item, a timestamp (last modification time), a "pre-fetch" bit (if set, instructs the server to send the actual item if updated), and the "expired" bit (tells if an item is expired). Upon receiving a CURP message, the server identifies items that have changed and those that have not, and sends the corresponding CNs in SVRP messages the ids of items that did not change and those that changed but do not have the pre-fetch bit set. It also sends the CNs SUDP messages containing the actual items if they were prefetched by the same CN and have changed.

Now the CN releases the request from the waiting list and sends the updated cached response to the RN via a data reply (DREP) message.

Copyright to IJARCCE

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 4, April 2016

II. DISTRIBUTED CACHE INVALIDATION METHOD (DCIM)

In this method, it proposes a pull-based algorithm that implements adaptive TTL, piggybacking and prefetching, and provides near strong consistency guarantees. Cached data items are assigned adaptive TTL values that correspond to their update rates at the data source. Expired items as well as non-expired ones but meet certain criteria are grouped in validation requests to the data source, which in turn sends the cache devices the actual items that have changed, or invalidates them, based on their request rates. This approach, which we call Distributed Cache Invalidation Mechanism algorithm that implements adaptive TTL, piggybacking and prefetching, and provides near strong consistency guarantees. Cached data items are assigned adaptive TTL values that correspond to their update rates at the data source. Expired items as well as non- expired ones but meet certain criteria are grouped in validation requests to the data source, which in turn sends the cache devices the actual items that have changed, or invalidates them, based on their request rates. This approach, which we call Distributed Cache Invalidation Mechanism (DCIM), works on top of the COACS cooperative caching architecture. To our knowledge, this is the first complete client side approach employing adaptive TTL and achieving superior availability, delay, and traffic performance.

A. Block diagram of Distributed Cache Invalidation Mechanism (DCIM)

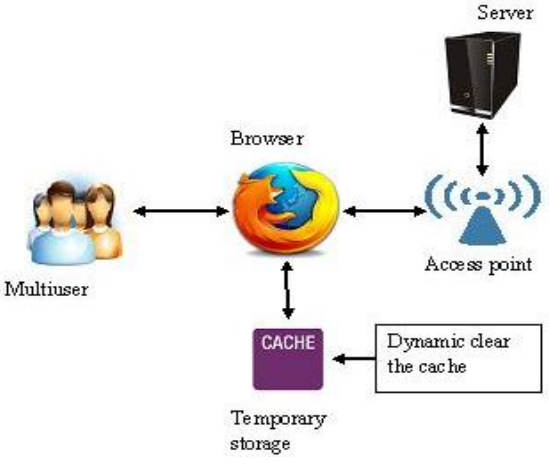

Fig. 1 DCIM Block Diagram

The above diagram described about DCIM (Distributed Cache Invalidation Method) process. In this system the process of this method is easy to search URL. DCIM is a client-side system that able to scale too many types of provided services. DCIM fits more naturally into the current state of the Internet with the prevailing client/ server paradigm, where clients are responsible for pulling the data from the server, which in turn maintains little state information and seldom pushes data to them. On the other hand, push based approaches, like those described in the related work section, rely on the server totally or partially to propagate item changes to the network. Multiuser or mobile node can search the browser using URL. The browser fetches information in server using access point. On another hand query node can perform. These query node store recently processing URL. In this URL are stored in limited time because many URL can stored cache the memory overhead. The user search any URL, browser first browse to cache node then perform the operation. Searching URL not found in cache node, the browser automatically fetch data to server using access point. Cache can performed in temporary storage.

B. DCIM design operation:

This phase describe the operations of DCIM in details, but first, list the messages which added in DCIM to those already introduced in COACs. The reader can refer for a complete description of all the COACS messages. The diagram describes basic interactions of DCIM through a scenario in which an RN is submitting a DRP (Data Request Packet) for a query indexed in the QD. The QD forwards the DRP to the CN caching the item assuming there was a hit. At the CN, the requested item may be in the waiting list at the moment if it is being validated. Validation requests are issued by CNs using CURP messages.

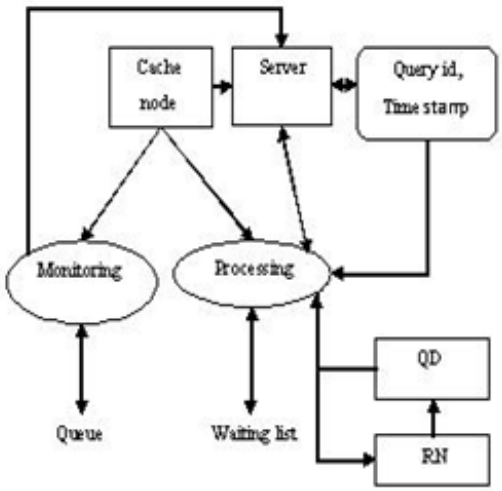

Fig. 2 DCIM operation

Each entry in the message consists of the query associated with this item, a timestamp (last modification time), a "prefetch" bit (if set, instructs the server to send the actual item if updated), and the "expired" bit (tells if an item is expired). Upon receiving a CURP message, the server identifies items that have changed and those that have not, and sends the corresponding CNs in SVRP messages the ids of items that did not change and those that changed but do not have the prefetch bit set. It also sends the CNs SUDP messages containing the actual items if they were prefetched by the same CN and have changed. Now the CN releases the request from the waiting list and sends the updated cached response to the RN via a data reply (DREP) message.

C. Server operation:

This approach is client-based; the processing at the server

DOI 10.17148/IJARCCE.2016.54263

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 4, April 2016

is minimal. When the server receives a CURP message from the CN, it checks if all items have been changed by comparing their last modified times with those included in the request. Items that have not changed are considered valid, and their ids are included in the SVRP response to the CN. On the other hand, items that have changed are treated in two ways: Expired items (those having the expiry bit set in the CN validation request) as well as non-expired ones but having the prefetch bit set are updated by sending SUDP packets (which contain the actual data items and the associated timestamps) to the originating CNs. As to the items whose expiry and prefetch bits are not set (i.e., will not be requested soon), the server informs the CN about them using an SVRP message. This is summarized in the following diagram. As such, the server only reacts to the received CURP messages that do not require it to maintain any state information, and thus it does not need to be aware of the MANET dynamics, including any CN or QD disconnections. Given this, it is not difficult to deploy DCIM in an Internet environment. For example, DCIM can be suited to be deployed on top of HTTP through mapping DCIM's request directives into HTTP header fields and utilizing extended headers, as allowed by HTTP/1.1 and defined in RFC 2616.

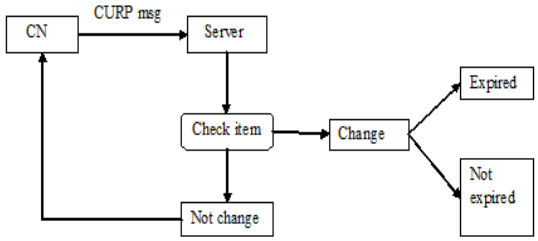

Fig. 3 Server Operation

D. QD operation:

QDs are elected based on their resource capabilities that explain how the number of QDs in the system is bounded by two limits. The lower bound corresponds to having enough QDs, such that an additional (elected) QD will not yield an appreciable reduction in average QD load. The upper bound, on the other hand, corresponds to a delay threshold, since traversing a larger number of QDs will lead to higher response times. Between these limits, the number of QDs can change dynamically depending on how much of the QD storage capacity is used. In the simulations performed in this work, the number of QDs averaged 7 at steady state when the number of nodes was 100.

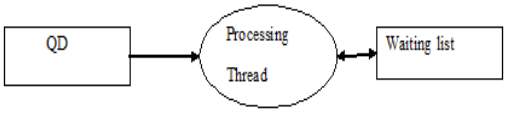

Fig. 4 QD operation

E. CN Processing:

The CNs store the cached queries along with their responses plus their IDs, and the addresses of the QDs indexing them. They are distributed in the network and cache a limited number of items, which makes monitoring their expiry an easy task. A CN maintains two tables to manage the consistency of the cache: the *Cache Information Table* whose data is common to all queries whose responses are locally cached and the *Query Information Table* that stores query-specific dataAs shown, the CN maintains the weighted average of inter-request interval (IRI) for each data item it holds (in a manner similar to the computation of the inter-update interval). The process that runs on the CN includes two threads: a monitoring thread and a processing thread.

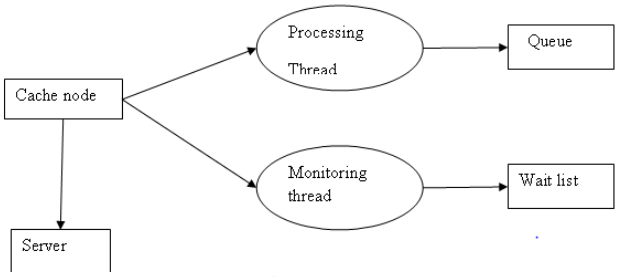

Fig. 5. CN Processing

III. EXPERIMENTAL RESULTS

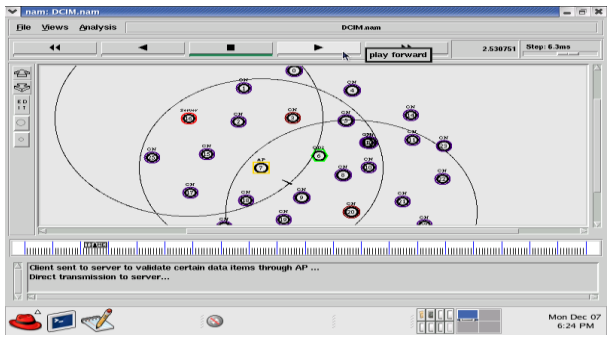

Fig.6 Direct Transmission to Server

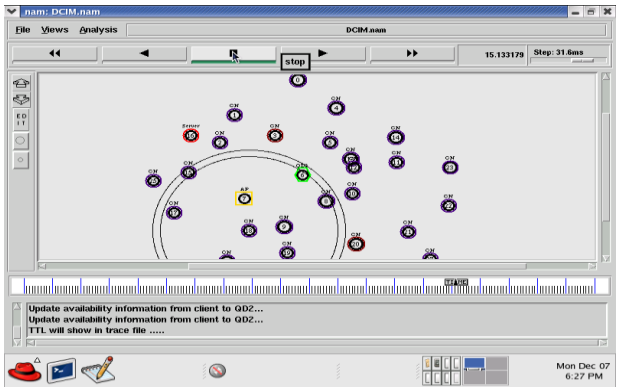

Fig. 7. TTL showing Trace Files

IV. CONCLUSION AND FUTURE WORK

A. CONCLUSION:

The presented a client-based cache consistency scheme for MANETs that relies on estimating the inter update intervals of data items to set their expiry time. It makes use of piggybacking and prefetching to increase the accuracy of its estimation to reduce both traffic and query delays. We compared this approach to two pull-based approaches (fixed TTL and client polling) and to two server-based approaches (SSUM and UIR). This showed

that DCIM provides a better performance than the other client based schemes and comparable performance to SSUM.

B. FUTURE WORK:

In this paper all the process are taken only in client side. In this method, the user send request to browser for finding URL or searching content. Browser accepts the request and send request to server. Server accept request and provide response to browser. Then next time user request same URL or content, the browser send request to cache, not send sever. Cache store content in assigned time, then that automatically removed. In future the client side process all applied to server side.

REFERENCES

[1] T. Andrel, A. Yasinsac, "On credibility of Manet simulations", IEEE Computer, 2006, pp.48-54.
[2] H. Artail, H. Safa, K. Mershad, Z. Abou-Atme, N. Sulieman, "COACS: A Cooperative and adaptive caching system for MANETS", IEEE TMC, v.7, n.8, pp. 961-977, 2008.
[3] D. Barbara, T. Imielinski, "Sleepers and Workaholics: Caching Strategies for Mobile Environments," ACM SIGMOD, pp. 1-12, May 1994.
[4] G.Cao, "A Scalable low-Latency Cache Invalidation Strategy for Mobile Environments," IEEE TKDE, v. 15, n. 5, pp. 1251-1265, 2003.
[5] D. Li, P. Cao, M. Dahlin. "WCIP: Web Cache Invalidation Protocol" IETF Internet Draft, March 2001, http://tools.ietf.org/html/draft-danli-wrec-wcip-01.
[6] J. Cao; Y. Zhang, G. Cao, X. Li, "Data Consistency for Cooperative Caching in Mobile Environments," Computer , v.40, n.4, pp.60-66, 2007.
[7] P. Cao, C. Liu, "Maintaining strong cache consistency in the World-Wide Web," IEEE Trans. Computers, v. 47, pp. 445–457, 1998.
[8] W. Li, E. Chan, D. Chen, S. Lu, "Maintaining probabilistic consistency for frequently offline devices in mobile ad hoc networks," 29th IEEE Int'l Conf. Distributed Computing Systems, pp. 215-222, 2009.
[9] J. Jung, A.W. Berger, H. Balakrishnan, "Modeling TTL-based internet caches," IEEE INFOCOM 2003, San Francisco, CA, March 2003.
[10] B. Krishnamurthy, C. Wills, "Study of piggyback cache validation for proxy caches in the World Wide Web," USENIX, Monterey, CA, December 1997.

DOI 10.17148/IJARCCE.2016.54263