IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 16, Issue 3, Ver. VI (May-Jun. 2014), PP 32-35
www.iosrjournals.org

Client Based Cache Consistency Scheme in Wireless Mobile Networks

Aswathi.P[1], Abey Abraham[2]
[1](*Information Technology, Rajagiri School of Engineering and Technology, India)*
[2](*Information Technology, Rajagiri School of Engineering and Technology, India)*

Abstract: *This paper proposes a client based cache consistency scheme for maintaining cache consistency in wireless mobile networks using a distributed cache invalidation method. This is implemented on top of a previously proposed architecture for caching data items in mobile ad hoc networks (MANETs), namely COCAS. We have also previously proposed a client and server based consistency scheme, named SSUM and DCIM. Client based cache consistency scheme is a pull-based algorithm that implements adaptive time to live (TTL), piggybacking, and prefetching, and provides strong consistency capabilities. We assign an adaptive TTL value to the cached data items in the data source depending on their update rates, where items with expired TTL values are grouped in validation requests to the data source to refresh them. But the unexpired ones with high request rates are prefetched from the server.*

Index Terms: *Cache consistency, Cache invalidation, Distributed cache invalidation method, time to live, wireless mobile networks.*

I. Introduction

A MANET is a type of ad hoc network that can change locations and configure itself on the fly. Because MANETS are mobile, they use wireless connections to connect to various networks. As Mobile Ad Hoc Networks (MANETs) are becoming increasingly widespread, the need for developing methods to improve their performance and reliability increases [2]. One of the biggest challenges in MANETs is management of data in the highly dynamic environments of MANETs. So in MANET environments, data caching is essential. Because it reduces contention in the network, increases the probability of nodes getting desired data, and improves system performance [3].

MANETs are dynamic in nature, and therefore, a reliable caching scheme is more difficult to achieve. Links between nodes may constantly change as nodes move around, enter, or leave the network. This can make storing and retrieving cached data particularly difficult and unreliable. The use of mobile devices adds even more complexity due to their relatively limited computing resources (e. g., processing power and storage capacity) and limited battery life [4]. It follows that an effective caching system for MANETs needs to provide a solution that takes all of these issues into consideration. An important policy of such a solution is not to rely on a single node but to distribute cache data and decision points across the network. With distribution, however, comes a new set of challenges. The most important of which is the coordination among the various nodes that is needed in order to store and find data.

The major issue that faces cache management is the maintenance of data consistency between the client cache and the server in a MANET, all messages sent between the server and the cache are subject to network delays, thus, impeding consistency by download delays that are considerably noticeable and more severe in wireless mobile devices. 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. Consequently there exist different consistency levels describing the degree to which the cached data is up to date. These levels, other than strong consistency, are weak consistency, delta consistency probabilistic consistency, and probabilistic delta consistency [1].

Cache consistency can be grouped in to three main categories: push-based, pull-based, and hybrid approaches .Push-based [2] is a server-based scheme in which server informs the cache about the updates. Pull-based [5] approaches are client-based in which the client asks the server to update or validate its cached data. In hybrid mechanism the server pushes the updates or the client pull them. In this paper, we propose a cache consistency scheme which is a pull-based scheme [5] and it use a distributed invalidation method for maintaining cache consistency in wireless mobile networks. 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.

In the rest of this paper, section 2 discusses related work and reveals the contributions of the proposed system, which we explain in section 3. Section 4 finishes the paper with concluding remarks. Paper should explain the nature of the problem, previous work, purpose, and the contribution of the paper. The contents of each section may be provided to understand easily about the paper.

II. Related work

Much work has been done in relation to cache consistency in MANETs. All cache consistency algorithms are developed with the same goal in mind: to increase the probability of serving data items from the cache that are identical to those on the server. A large number of such algorithms have been proposed in the literature, and they fall into three groups: server invalidation, client polling, and time to live (TTL). With server invalidation, the server sends a report upon each update to the client. Two examples are the Piggyback server invalidation [9] and the Invalidation report [3] mechanisms. In IR approach include stateless scheme where the server stores no information about the client caches. This is a push-based approach. In client polling, like the Piggyback cache validation of [2], a validation request is initiated according to a schedule. If the copy is up to date, the server informs the client that the data have not been modified; else the update is sent to the client. Client poll algorithms have relatively low bandwidth consumption; their access delay is high considering that each item needs to be validated upon each request.

TTL algorithms, a server-assigned TTL value (e. g., T) is stored alongside each data item d in the cache. The data d are considered valid until T time units pass since the cache update. Usually, the first request for d submitted by a client after the TTL expiration will be treated as a miss and will cause a trip to the server to fetch a fresh copy of d. Many algorithms were proposed to determine TTL values, including the fixed TTL approach, adaptive TTL, and Squids LM-factor [8]. Adaptive TTL provides higher consistency requirements along with lower traffic and is calculated using different mechanisms. TTL-based consistency algorithms are popular due to their simplicity, sufficiently good performance, and flexibility to assign TTL values for individual data items [4]. However, TTL-based algorithms, like client polling algorithms, are weakly consistent, in contrast to server invalidation schemes that are generally strongly consistent. The first mechanism in [7] calculates TTL as a factor multiplied by the time difference between the query time of the item and its last update time. This factor determines how much the algorithm is optimistic or conservative. In the second mechanism, TTL is adapted as a factor multiplied by the last update interval. In the third mechanism in [6] TTL is calculated as the difference between the query time and the kth recent distinct update time at the server divided by a factor K, and the server relays to the cache the k most recent update times.

In [9], three related caching schemes also there: Cache Path, Cache Data, and Hybrid Cache. The main idea behind these schemes is to analyze passing requests and cache either the data or the address of the node in which it is stored. Later, if the same request for that data passes through the node, it can provide the data itself or redirect the request to its location. Cache Path saves space by storing locations where data should be stored, while Cache Data saves time by storing the data instead of the path. The third scheme, Hybrid Cache, is a middle solution where queries are cached by path or by data, depending on what is optimal.

III. Architecture and Operation of Cache Consistency Scheme

This section describes the design of cache consistency scheme and the interactions between its different components.

3.1. 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 multi-hop communication. For example, Node CN4 in Fig. 1 is accessing the server through N4 and then CN1, which in turn acts as a gateway by connecting to the Internet via the AP. The proposed system builds on top of COACS, which we introduced in [2] and did not include provisions for consistency. Briefly, the system has three types of nodes: caching nodes (CNs) that cache previously requested items, query directories (QDs) that index the cached items by holding the queries along with the addresses of the corresponding CNs, and requesting nodes (Rns) that are ordinary nodes. Any node, including a QD or a CN, can be a requesting node, and hence, an RN is not actually a special role, as it is only used in the context of describing the system. One, therefore, might view the employed caching system as a two layered distributed database. The first layer contains the QDs which map the queries to the caching nodes which hold the actual items that are responses to these queries, while the second layer is formed by the CNs.

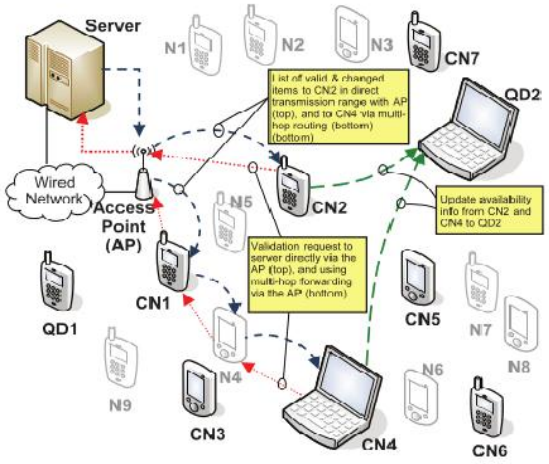

Fig. 1. Overview of the basic design

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).

3.2. Basic Design

We use three messages which are already introduced in COACS. Its description can be shown in the table.1.

Table. 1. Packets used in DCIM

Packet	Function	Description
CURP	Cache Update Request	Sent from CN to server 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 server to CN. It includes updated data items and timestamps

Fig. 2 shows the basic interactions of distributed cache invalidation mechanism through a scenario in which more than one RN is submitting a data request packet (DRP) for a query indexed in the QD.QD maintain two FIFO queues for store the DRP .One queue consist the DRP messages that become forward request to CN in the case of a hit.

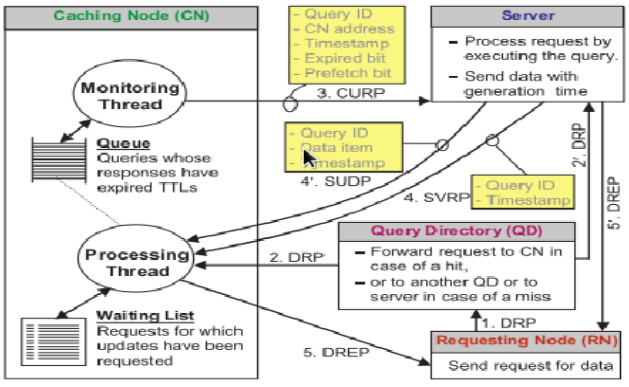

Fig. 2. Packets used in DCIM Interaction between 2 nodes in the system

The second queue consist the DRP messages that become forward to the sever in the case of a miss. This makes the system simpler and saves traffic. If there are many QDs, then do an election procedure among the QDs depending on their resource capabilities. 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 prefetched bit (if set, instructs the server to send the actual item if updated), and the expired bit (tells if an item is expired). 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. 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.

3.3. Caching Nodes

Caching node plays an important role in distribute cache invalidation mechanism for maintaining cache consistency.CN has the role for TTL adaptation. The CN stores the last update time of each item from the last validation request, and uses this information to predict the next update time. Alternatively, we use a running average to estimate the interupdate interval, using timestamps of the items from the server's responses to issued validation requests. The CN can then calculate its own estimation for the interupdate interval at the server, and utilizes it to calculate the TTL of the data item. The CNs stores 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.

IV. Conclusion

This paper presented a client based cache consistency scheme based on distributed cache invalidation method in wireless mobile networks. This scheme is based on pull-based approach. This 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.

Acknowledgements

We give all honor and praise to the lord who gave us wisdom and enable us to create this paper work successfully.

We express our sincere thanks to Dr. J. Issac, Principle, Rajagiri School of Engineering and Technology and Information Technology head of department Prof. Kuttyamma A J for providing the facilities to carry out the paper work successfully. It would be unfair on our part if we do not express our gratitude towards our parents and friends for giving all the moral support in the due course of the work.

References

[1]. Kassem Fawaz and Hassan Artail , DCIM: Distributed Cache In-validation Method for Maintaining Cache Consistency in Wireless Mobile Networks , IEEE Transactions on mobile computing,vol. 12,no. 4,April 2013.
[2]. H. Artail, H. Safa, K. Mershad, Z. Abou-Atme, and N. Sulieman , COACS: A Coop- erative and Adaptive Caching System for MANETS, IEEE Trans. Mobile Comput- ing, vol. 7, no. 8, pp. 961-977, Aug. 2008 .
[3]. K. Mershad and H. Artail , SSUM: Smart Server Update Mechanism for Maintaining Cache Consistency in Mobile Environments, IEEE Trans. Mobile Computing, vol. 9, no. 6 , pp. 778-795,June 2010.
[4]. G. Cao, A Scalable Low-Latency Cache Invalidation Strategy for Mobile Environ- ments, IEEE Trans. Knowledge and Data Eng., vol. 15, no. 5, pp. 1251-1265, Sept./Oct. 2003.
[5]. J. Cao, Y. Zhang, G. Cao, and X. Li, Data Consistency for Cooperative Caching in Mobile Environments , Computer, vol. 40, no. 4, pp. 60-66, 2007.
[6]. J. Jung, A.W. Berger, and H. Balakrishnan , Modeling TTL-Based Internet Caches, Proc. IEEE INFOCOM, Mar. 2003.
[7]. M. Denko and J. Tian , Cooperative Caching with Adaptive Prefetching in Mobile Ad Hoc Networks , Proc. IEEE Intl Conf. Wireless and Mobile Computing, Networking and Comm. (WiMob 06), pp. 38-44, June 2006.
[8]. Q. Hu and D. Lee , Cache Algorithms Based on Adaptive Invalidation Reports for Mobile Environments , Cluster Comput- ing, vol. 1, pp. 39-50, 1998.
[9]. S. Lim, W.C. Lee, G. Cao, and C. Das , Cache Invalidation Strategies for Internet- Based Mobile Ad Hoc Networks , Computer Comm., vol. 30, pp. 1854-1869, 2007.
[10]. Y. Du and S.K.S. Gupta , COOP - A Cooperative Caching Service in MANETs , Proc. Joint Intl Conf. Autonomic and Autonomous Systems and Intl Conf. Network- ing and Services (ICAS-ICNS), pp. 58- 58, Oct. 2005.