International Journal of Engineering Research in Computer Science and Engineering (IJERCSE) Vol 3, Issue 6, June 2016

An Efficient Caching scheme to maintain consistency in Hybrid P2P system

[1]Usha Rani J [2]Swetha D [3]Rajath A N
[1][2][3]Assistant professors, Department of Computer science & Engineering
GSSS Institute of Engineering & Technology for Women Karnataka Mysuru -570016
[1]usharani@gsss.edu.in, [2]swetha.d@gsss.edu.in,[3]rajathan@gsss.edu.in

***Abstract* - The paper aims on how efficiently the performance of cache memory can be maintained in disturbed network and also in P2P networks. A Technique of distributed cache invalidation mechanism (DCIM), it is client-based cache consistency scheme is implemented on top of a previously existing architecture, namely COACS, here in COACS a special nodes cache the queries and the addresses of the nodes that store the responses to these queries. DCIM uses a pull-based algorithm that implements adaptive time to live (TTL), piggybacking, and perfecting, and provides near strong consistency capabilities. Cached 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, whereas unexpired ones but with high request rates are perfected from the server.**

***Keywords* — Hybrid, Peer-to-peer systems, structured, unstructured P2P, DCIM, TTL**

I. INTRODUCTION

A peer to peer network (p2p) is a distributed application architecture that partitions tasks between peers. Peers are both suppliers and consumers of resources, in contrast to the traditional client-server model in which the consumption and supply of resources is divided.

The peer joins or leaves the system often; hence, p2p networks [1] are dynamic in nature. Peer-to-peer networks can be divided into two categories: structured peer-to-peer networks where the organization of peers is through regular topology and unstructured peer-to-peer networks where the organization of peers is arbitrary. Hence, neither structured peer-to-peer networks nor unstructured peer-to-peer networks can provide efficient, flexible, and robust assistance independently [2].

We propose a hybrid peer-to-peer system for distributed data sharing which combines the structured and unstructured peer-to-peer networks. In the proposed hybrid system, a structured ring-based core network forms the backbone of the system and multiple unstructured peer to peer networks are attached to the backbone and communicate with each other through the backbone. The core-structured network provides an accurate way to narrow down the queried data within a certain unstructured network, while the unstructured networks provide a low cost mechanism for peers to join or leave the system freely.

We propose a hybrid peer-to-peer networks which is a combination of structured and unstructured peer-to-peer networks. These architectures use special node to provide directory services for regions of the network and are potentially powerful model for developing large scale network.

In this paper, we propose a pull-based algorithm that implements adaptive TTL, piggybacking, and perfecting, 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 which 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) [12] works on top of the COACS cooperative caching architecture.

II. EXISTING SYSTEM

The cache consistency mechanisms can be grouped into three main categories: push based, pull based, and hybrid approaches. Push-based mechanisms are mostly server-based, where the server informs the caches about updates, whereas pull-based approaches are client-based, where the client asks the server to update or validate its cached data. Finally, in hybrid mechanisms the server pushes the updates or the clients pull them. We focus on

peer-to-peer networks for efficient distributed data (file) sharing among peers.

2.1 push/pull caching approaches

Pull caching is client initiated whereas the push caching is web server initiated.

Figure 1: Push technology - A content provider (server) sends new information to proxy and client is notified about it.

As shown in figure, in push-based data delivery, the server tracks all proxies that have requested objects. If web page has been modified, it notifies each proxy and when the client requests for the file it is served from the proxy's cache instead of request going directly to the server. This improves network utilization.

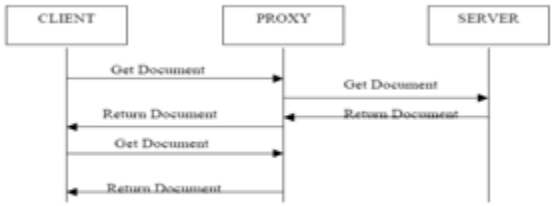

Figure 2: Pull Technology [1]

As shown in figure 2, a proxy explicitly requests data items from the server [4] when the client requests for a document. The proxy instead of the request going directly to the server form the client can service subsequent requests for the same file by clients. In the pull-based approach, the proxy is entirely responsible or maintaining consistency. The proxy maintains data constancy by setting a TTL on the document cached and this copy is served until the TTL expires

2.2 advantages of the push approach

1. The push technology can reduce the burden of acquiring data for tasks in which there is a large information flow. Push technologies improve efficiency by downloading information to the users' system in a scheduled fashion so it can be rapidly viewed, and thereby eliminating the risk of the user being served stale data.

2. Pushing alerts to the user (e.g. In the form of e-mail or the change itself), improves the efficiency of web-based time-sensitive information distribution (such as stock quotes or trouble tickets in a technical support system).

3. Automatic downloading of software upgrades and fixes is a way to deliver software faster, and at the same time, reduce costs.

4. The push technology enables intelligent information filtering based on personalized user profiles describing required information needs.

2.3 push vs pull

In the push approach, the server repetitively transmits (multicasts) the data to the proxy services. In such a system, data items are periodically sent from the server to the proxy service without requiring a specific request from the clients. In contrast, the pull-based approach explicitly requests data items by sending messages to the web server . Pull based access has the advantage of allowing proxy service to play a more active role in obtaining the data, rather than relying solely on push enabled web servers.

III PROPOSED SYSTEM

We propose 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 which meets 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.

3.1 advantages of proposed system

- TTL algorithms are popular due to their simplicity, sufficiently good performance, and flexibility to assign TTL values to individual data items.
- Also, they are attractive in mobile environments because of limited device energy and network bandwidth and frequent device disconnections.
- TTL algorithms are also completely client based and require minimal server functionality. From this perspective, TTL-based algorithms are more practical to deploy and are more scalable.
- This is the first complete client side approach employing adaptive TTL and achieving superior availability, delay, and traffic performance.

IV REVIEW OF PREVIOUS PAPER

A. In [3], they introduced a cooperation-based database caching system for mobile ad hoc networks (manets). The heart of the system is the nodes that cache submitted queries. The queries are used as indices to data

cached in nodes that previously requested them. We discuss how the system is formed and how requested data is found if cached, or retrieved from the external database and then cached. Analysis is performed and expressions are derived for the different parameters, including upper and lower bounds for the number of query caching nodes as well as the average load they experience, generated network traffic, node bandwidth consumption, and other performance-related measures. Simulations with the ns-2 software were used to study the performance of the system in terms of average delay and hit ratio, and to compare it with the performance of two other caching schemes for manets, namely cache path and cache data. The results demonstrate the effectiveness of the proposed system in terms of achieved hit ratio and low delay.

B. In [4], the mobile wireless computing environment of the future a large number of users equipped with low powered palmtop machines will query databases over the wireless communication channels. Palmtop based units will often be disconnected for prolonged periods of time due to the battery power saving measures; palmtops will also frequently relocate between different cells and connect to different data servers at different times. Caching of frequently accessed data items will be an important technique that will reduce contention on the narrow bandwidth wireless channel. However, cache invalidation strategies will be severely affected by the disconnection and mobility of the clients. The server may no longer know which clients are currently residing under its cell and which of them are currently on. We propose a taxonomy of different cache invalidation strategies and study the impact of client's disconnection times on their performance.

C. In [5],caching frequently accessed data items on the client side is an effective technique for improving performance in a mobile environment. Classical cache invalidation strategies are not suitable for mobile environments due to frequent disconnections and mobility of the clients. One attractive cache invalidation technique is based on invalidation reports (irs). However, the ir-based cache invalidation solution has two major drawbacks, which have not been addressed in previous research. First, there is a long query latency associated with this solution since a client cannot answer the query until the next ir interval. Second, when the server updates a hot data item, all clients have to query the server and get the data from the server separately, which wastes a large amount of bandwidth. In this paper, we propose an ir-based cache invalidation algorithm, which can significantly reduce the query latency and efficiently utilize the broadcast bandwidth. Detailed analytical analysis and simulation experiments are carried out to evaluate the proposed methodology. Compared to previous ir-based schemes, our scheme can significantly improve the throughput and reduce the query latency, the number of uplink request, and the broadcast bandwidth requirements.

D. In [6], the trend toward wireless communications and advances in mobile technologies are increasing consumer demand for ubiquitous access to internet-based information and services. A 3d framework provides a basis for designing, analyzing, and evaluating strategies to address data consistency issues in mobile wireless environments. A proposed relay-peer-based cache consistency protocol offers a generic and flexible method for carrying out cache invalidation.

E. In [7], as the web continues to explode in size, caching becomes increasingly important. With caching comes the problem of cache consistency. Conventional wisdom holds that strong cache consistency is too expensive for the web, and weak consistency methods, such as time-to-live (TTL), are most appropriate. This study compares three consistency approaches: adaptive TTL, polling-every-time and invalidation, through analysis, implementation, and trace replay in a simulated environment. Our analysis shows that weak consistency methods save network bandwidth mostly at the expense of returning stale documents to users. Our experiments show that invalidation generates a comparable amount of network traffic and server workload to adaptive TTL and has similar average client response times, while polling-every-time results in more control messages, higher server workload, and longer client response times. We show that, contrary to popular belief, strong cache consistency can be maintained for the web with liTTLe or no extra cost than the current weak consistency approaches, and it should be maintained using an invalidation-based protocol.

V SYSTEM ARCHITECTURE

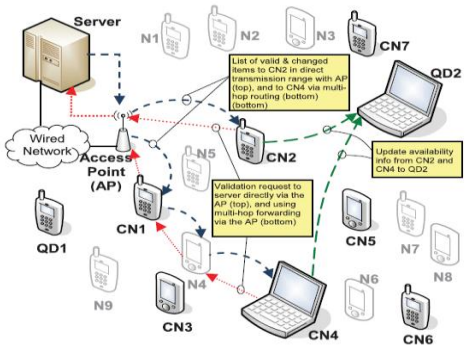

Fig 3: overview of DCIM basic design.

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.

Sequence Diagram

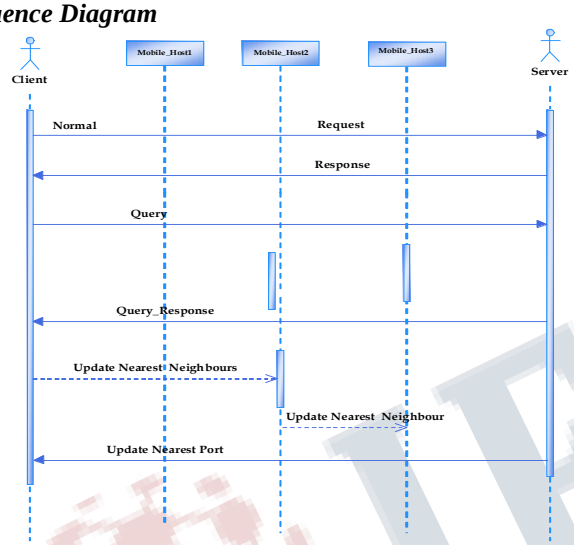

Figure 4.shows the set of activities of between client and server using pull based approach

VI. MODULES

- TTL Adaptation Module
- Server Operations
- CN Processing Module
- QD Operations Module

1. TTL Adaptation Module:

The CN in DCIM has a partial view about the update pattern of each item at the server with the help of piggybacking mechanism [10]. It stores the last update time of each item from the last validation request, and uses this information to predict the next update time. However, the CNs has constraints in terms of processing, storage capabilities, and power and obviously, sophisticated prediction schemes are slow and inadequate to use as they are mobile devices. We use a running average to estimate the inter update 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 inter update interval at the server, and utilizes it to calculate the TTL of the data item [9] [11].

2. Server Operations:

As this approach is client-based, the processing at the server 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 nonexpired 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.

***3. CN Processing Module*:**

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. 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 data.

4. QD Operations Module:

In contrast to the CNs that become caching nodes when they first request non-cached data, QDs are elected based on their resource capabilities, as described. A procedure is included that explains 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.

5. Handling CN and QD Disconnections:

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. As a result, the RN would become a CN for this item in particular, and the QD will mark this entry as invalid. All the QDs behave similarly for each request they receive for items cached in offline CNs. When a CN rejoins the network, it broadcasts a HELLO message as any new node joining the network would do.

VII. CONCLUSION

We 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 overall performance than the other client based schemes and comparable performance to SSUM. For future work, we will explore three directions to extend DCIM. First, we will investigate more sophisticated TTL algorithms to replace the running average formula. Second, we will extend our preliminary work in to develop a complete replica allocation. Third, DCIM assumes that all nodes are well behaved, as issues related to security were not considered. However, given the possibility of network intrusions, we will explore integrating appropriate security measures into the system functions. These functions include the QD election procedure, QD traversal, QD and CN information integrity, and TTL monitoring and calculation. The first three can be mitigated through encryption and trust schemes

REFERENCES

[1] Min Yang, Yuanyuan Yang., "An Efficient Hybrid Peer-to-Peer System for Distributed Data Sharing " IEEE transaction on Computers, Vol. 59, no.9, September 2010.

[2] E. Cohen, S. Shenker," Replication strategies in unstructured peerto-
peer network's", in: Proc. of ACM SIGCOMM, 2002

[3] 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.

[4] D. Barbara and T. Imielinski, "Sleepers and Workaholics: Caching Strategies for Mobile Environments," Proc. ACM SIGMOD, pp. 1- 12, May 1994.

[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] 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.

[7] 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.

[8] Y. Huang, J. Cao, Z. Wang, B. Jin, and Y. Feng, "Achieving Flexible Cache Consistency for Pervasive Internet Access," Proc. IEEE Fifth Ann. Int'l Conf. Pervasive Computing and Comm., pp. 239- 250, 2007.

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

[10] B. Krishnamurthy and C. Wills, "Study of Piggyback Cache Validation for Proxy Caches in the World Wide Web," Proc. USENIX Symp. Internet Technologies and Systems, Dec. 1997.

[11] J. Lee, K. Whang, B. Lee, and J. Chang, "An Update-Risk Based Approach to TTL Estimation in Web Caching," Proc. Third Int'l Conf. Web Information Systems Eng. (WISE '02), pp. 21-29, 2002.

[12] K.S. Khurana, S. Gupta, and P. Srimani, "A Scheme to Manage Cache Consistency in a Distributed Mobile Wireless Environment," IEEE Trans. Parallel and Distributed Systems, vol. 12, no. 7, pp. 686-700, 2001.

[13] 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.

[14] 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.

[15] U.A. Ninan, M. Raunak, P. Shenoy, and K. Ramamritham, "Maintaining Mutual Consistency for Cached Web Objects," Proc. 21st Int'l Conf. Distributed Computing Systems, p. 371, 2001.