AN EFFICIENT CACHE CONSISTENCY SCHEME IN MOBILE NETWORKS

T.Chindrella Priyadharshini, S.Neelakandan, M.Swarnalatha

Abstract— In this paper we proposes a cache consistency scheme based on a previously proposed architecture for caching database data in MANETs. The original scheme for data caching stores the queries that are submitted by requesting nodes in special nodes, called query directories (QDs), and uses these queries to locate the data (responses) that are stored in the nodes that requested them, called caching nodes (CNs). The consistency scheme is server-based in which control mechanisms are implemented to adapt the process of caching a data item and updating it by the server to its popularity and its data update rate at the server. The system implements methods to handle disconnections of QD and CN nodes from the network and to control how the cache of each node is updated or discarded when it returns to the network. Estimates for the average response time of node requests and the average node bandwidth utilization are derived in order to determine the gains (or costs) of employing our scheme in the MANET. Moreover, ns2 simulations were performed to measure several parameters, like the average data request response time, cache update delay, hit ratio, and bandwidth utilization. The results demonstrate the advantage of the proposed scheme over existing systems.

Index Terms— Cache consistency, data consistency, invalidation, Kalmam filter prediction techniques, MANET, server-based approach, and utility theory.

1 INTRODUCTION

In a mobile ad hoc network (MANET), data caching is essential as it reduces contention in the network, increases the probability of nodes getting desired data, and improves system performance [24], [28]. The major issue that faces cache management is the maintenance of data consistency between the client cache and the server [28]. 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. 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 [7] and the Invalidation report [19] mechanisms. In client polling, like the Piggyback cache validation of [8], 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. Finally, with 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 [6], adaptive TTL [32], and Squid's LM-factor [17]. TTL-based consistency algorithms are popular due to their simplicity, sufficiently good performance, and flexibility to assign TTL values for individual data items [27]. However, TTL-based algorithms, like client polling algorithms, are weakly consistent, in contrast to server invalidation schemes that are generally strongly consistent. According to [32], with strong consistency algorithms, users are served strictly fresh data items, while with weak algorithms, there is a possibility that users may get inconsistent (stale) copies of the data. This work describes a server-based scheme implemented on top of the COACS caching architecture we proposed in [25]. 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 described in this paper fills that void and adds several improvements: 1) enabling 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, and 3) adapting the cache update process to the data update rate at the server 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. Next, Section 2 describes the proposed system while Section 3 discusses the experimental results, and Section 5 reviews some previous cache consistency techniques before concluding the paper in Section 6.

- *T.Chindrella Priyadharshini, currently working as a Assistant Professor in R.M.K College of Engineering and Technology, India, E-mail: chindrella17@gmail.com*
- *S.Neelakandan, currently working as a Assistant Professor in R.M.K College of Engineering and Technology, India, E-mail: snksnk07@gmail.com*
- *M.Swarnalatha, currently working as a Assistant Professor in R.M.K College of Engineering and Technology, India, E-mail: swarni23@gmail.com*

2. EMINENT SERVER IMPROVED MECHANISM

ESIM is a server-based approach that avoids many issues associated with push-based cache consistency approaches. Specifically, traditional server-based schemes are not usually aware of what data items are currently cached, as they might have been replaced or deleted from the network due to node disconnections. Also, if the server data update rate is high relative to the nodes request rate, unnecessary network traffic would be generated, which could increase packet dropout rate and cause longer delays in answering node queries. ESIM reduces wireless traffic by tuning the cache update rate to the request rate for the cached data.

2.1 Basic operations

Before detailing the operations of ESIM, we list in Table 1 the messages used in the COACS architecture as we refer to them in the paper. Given that no consistency mechanism was implemented in COACS, it was necessary to introduce four additional messages. In ESIM, 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. 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).

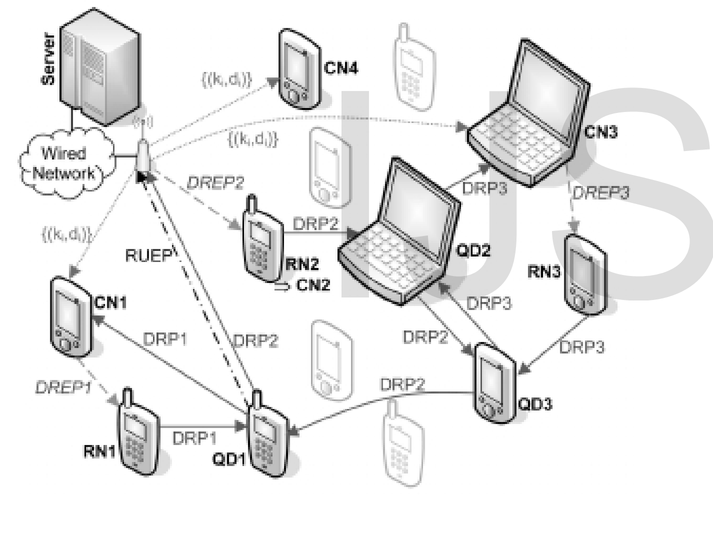

Fig. 1. Scenarios for requesting and getting data in the COACS Architecture

Otherwise, it forwards it to its nearest QD, which has not received the request yet. If the request traverses all QDs without being found, a miss occurs and it gets forwarded to the server which sends the data item to the RN. In the latter case, after the RN receives the confirmation from the last traversed QD that it has cached the query, it becomes a CN for this data item and associates the address of this QD with the item and then sends a Server Cache Update Packet (SCUP) to the server, which, in turn, adds the CN's address to the data item in its memory. This setup allows the server to send updates to the CNs directly whenever the data items are updated. Fig. 1 illustrates few data request and update scenarios that are described below. In the figure, the requesting nodes (RNs) submit queries to their nearest QDs, as shown in the cases of RN1, RN2, and RN3 . The query of RN1 was found in QD1 , and so the latter forwarded the request to CN1, which returned the data directly to the RN. However, the query of RN2 was not found in any of the QDs, which prompted the last searched (QD1) to forward the request to the server, which, in turn, replied to RN2 that became a CN for this data afterward. The figure also shows data updates (key data pairs) sent from the server to some of the CNs.

2.2 Dealing with Query Replacement and Node Disconnections

A potential issue concerns the server sending the CN updates for data that have been deleted (replaced), or sending the data out to a CN that has gone offline. To avoid this and reduce network traffic, cache updates can be stopped by sending the server Remove Update Entry Packets (RUEPs). This could occur in several scenarios. For example, if a CN leaves the network, the QD, which first tries to forward it a request and fails, will set the addresses of all queries whose items are cached by this unreachable CN in its cache to À1, and sends an RUEP to the server containing the IDs of these queries. The server, in turn, changes the address of that CN in its cache to À1 and stops sending updates for these items. Later, if another node A requests and then caches one of these items, the server, upon receiving an SCUP from A, will associate A with this data item. Also, if a CN runs out of space when trying to cache a new item in , it applies a replacement mechanism to replace id with in and instructs the QD that caches the query associated with id to delete its entry. This causes the QD to send an RUEP to the server to stop sending updates for id in the future. If a caching node CNd returns to the MANET after disconnecting, it sends a Cache Invalidation Check Packet (CICP) to each QD that caches queries associated with items held by this CN. A QD that receives a CICP checks for each item to see if it is cached by another node and then sends a Cache Invalidation Reply Packet (CIRP) to CNd containing all items not cached by other nodes. CNd then deletes from its cache those items whose IDs are not in the CIRP but were in the CICP. After receiving a CIRP from all QDs to which it sent a CICP and deleting nonessential data items from its cache, CNd sends a CICP containing the IDs of all queries with data remaining in its cache to the server along with their versions. In the meanwhile, if CNd receives a request from a QD for an item in its cache, it adds the request to a waiting list. The server then creates a CIRP and includes in it fresh copies of the outdated items and sends it to CNd , which, in turn, updates its cache and answers all pending requests. Finally, and as described in [25], QD disconnections and reconnections do not alter the cache of the CNs, and hence, the pointers that the server holds to the CNs remain valid.

2.3 Adapting to the Ratio of Update rate and Request rate

ESIM suspends server updates when it deems that they are unnecessary. The mechanism requires the server to monitor the rate of local updates, Ru , and the rate of RN requests, Rr , for each data item di . Each CN also monitors these values for each data item that it caches. Whenever a CN receives an update from the server, it calculates Ru =Rr and compares it to a threshold À. If this ratio is greater than or equal to À, the CN will delete di and the associated information from its cache and will send an Entry Deletion Packet (EDP) to the QD (say, QDd) that caches query qi . The CN includes in the header of EDP a value for Ru , which tells QDd that di is being removed due to its high update-to-request ratio. Normally, when a QD gets an EDP, it removes the cached query from its cache, but here, the nonzero value of Ru in the EDP causes QDd to keep the query cached, but with no reference to a CN. Next, QDd will ask the server to stop sending updates for di . Afterward, when QDd receives a request from an RN node that includes qi , it forwards it to the server along with a DONT_CACHE flag in the header to be later passed in the reply, which includes the results, to the RN. Under normal circumstances in COACS, when an RN receives a data item from the server in response to a query it had submitted, it aESIMes the role of a CN for this item and will ask the nearest QD to cache the query. The DONT_CACHE flag instructs the RN to treat the result as if it were coming from the cache and not become a CN for it. Now, at the server, each time an update for qi occurs and a new Ru=Rr is computed, if this ratio falls below a second threshold, (< À), the server will reply to the RN with a DREP that includes the CACHE_NEW flag in the header. Upon receiving the DREP, the RN sends a QCRP with the CACHE_NEW flag to its nearest QD. If this QD caches the query of this item (with À1 as its CN address), it sets its address to its new CN, else it forwards the request to its own nearest QD. If the QCRP traverses all QDs without being processed (implying that the QD caching this item has gone offline), the last QD at which the QCRP arrives will cache the query with the CN address. By appropriately selecting the values of   and À, the system can reduce unnecessary network traffic. The processing time of qi will suffer though when Ru =Rr is above after it had passed À since QDd will be sending qi to the server each time it receives it. However, the two thresholds allow for favoring bandwidth consumption over response time, or vice versa. This makes ESIM suitable for a variety of mobile computing applications: a large À may be used when disconnections are frequent and data availability is important, while a low À could be used in congested environments where requests for data are infrequent or getting fresh data is not critical.

2.4 Accounting for Latency in Receiving Server updates

Given the different processes running at the server and since it sends the updates to the CNs via unicasts, there may be a time gap between when an update occurs and when the CN actually receives the updated data item d. Hence, if the CN gets a request for d during this time, it will deliver a stale copy of d to the RN. Our design uses the time stamp that the server sends with each update in an attempt to mitigate this issue. To explain this, suppose that the time stamp sent with d is ts and the time of receiving d by the CN is tc . Upon getting an update, the CN checks if it had served any RN a copy of d from its cache in the past ts-tc milliseconds. If it is the case, the CN sends a new DREP to the RN, but now it includes the fresh copy of d. The above solution assumes that the clocks of the nodes in the MANET and that of the server are synchronized. This assumption is realistic given that node clock synchronization is part of the MAC layer protocol, as specified by the IEEE 802.11 standards [4]. In particular, IEEE 802.11 specifies a Timing Synchronization Function (TSF) through which nodes synchronize their clocks by broadcasting their timing information using periodic beacons. Since the Access Point (AP) is considered a node in the MANET and it can synchronize its clock with that of the server asynchronously with respect to the MANET through the wired network, it will be possible to synchronize the clocks of the mobile nodes with that of the server at almost a zero cost to them (no protocol besides the MAC layer's TSF is needed). The suitability of TSF for ESIM depends on its effectiveness. It was shown in [22] that using TSF, the maximum clock offset in the case of 500 nodes is 700 s. Several approaches were proposed to reduce this error. The Automatic Self-time Correcting Procedure (ASP) [16] reportedly reduced the maximum offset to 300 s, while the Multi hop Adaptive TSF (MATSF) method [22] cut it down to 50 s, but at the expense of adding 8 bits to the IEEE 802.11 frame. In Section 2.6, we show that ESIM is best characterized by the delta consistency model [9], where the upper bound for the delta between the time of the server's update and the time the RN gets a fresh copy is in tens of milliseconds. It follows that TSF will virtually not increase this delta, especially in small to moderately sized networks.

2.5 Overhead Cost

When a node joins the network, the server will know about it when it first gets a query from it in a DRP that is forwarded by one of the QD nodes. Each data item at the server that is cached in the network is associated with a query id, a request rate, an update rate, and the address of the CN caching it. This additional information could cost the server about 16 bytes of extra storage per record. Hence, from a storage perspective, this cost may be deemed insignificant when considering the capabilities of modern servers. In terms of communication cost, the server communicates with the CNs information about the rates uses control packets (RUEP, SCUP, CICP, and CIRP) to manage the updating process. Here, it suffices to state that the simulation results (Section 4.10) indicate that the overall overhead traffic (including other packets that do not concern the server) is a small portion of the data traffic. Finally, from a processing load point of view, the server is only required to manipulate the update rate when appropriate. In conclusion, the server will not incur any notably additional load due to its role in this architecture. Hence, the system should be able to scale to a large number of cached items. A similar argument can be made for the CNs, although the main concern here is the impact on the cache space and replacement frequency. Using the same value of 5 KB for the average data item size (as in the simulations of Section 4), with caching capacity of 200 KB, a CN can cache about 40 data items. The additional overhead required for storing the request and update rates of one single data item is 8 bytes, and therefore, the overhead for storing the request and update rates at the CN is less than 0.16 percent of the available space. It follows that the space for caching at the CNs is minimally impacted by this additional information. Also, the frequency of cache replacements will not increase in a major way because of this.

2.6 Consistency Model

From the characteristics of ESIM described above, we may relate it to the delta consistency model [21]. Assuming the RN always requires a fresh copy of item d, a CN may end up sending two copies of d to an RN in case the CN serves a copy of d from its cache shortly before receiving a server update for it. First, we emphasize that from the perspective of the RN, d will be considered fresh only if it is a duplicate of the server's version at the time it is received (i.e., the server has not updated it just yet). We now define two boundary situations in which the RN gets a stale copy of d. In the first one, immediately after the server updates d at ts and sends it into the network, the RN receives a copy of it from the CN based on a request it had sent. In the second one, right after the CN serves the RN a copy of d, it receives an updated version from the server. Assuming that when the server updates d, it immediately sends it out, and that it
is connected to the MANET via an access point in the corner of the network, we can now compute an interval for the maximum "delta" between ts and the time the RN gets a fresh copy of it. To do this, we borrow some definitions from our work in [25] in relation to the average number of hops. The top two definitions below are used here while the other two are used in Section 3, but all are detailed in

- HC is the average number of hops between the corner of the topology and a random node in the MANET. It applies when a packet is sent between the server and the random node.
- HR is the expected number of hops between any two randomly selected nodes.
- HA is the expected number of hops to traverse all the QDs in the system, which usually occurs in the case of a cache miss.
- HD is the expected number of hops to reach the QD which holds the reference to the requested data, in the case of a hit

Hence, if Tin is the transmission delay between two neighboring nodes in the wireless network and Tout is the delay in the wired network to/from the server, then the delay until the RN gets a fresh copy of d in the first situation is Tout þ HC Â Tin þ HR Â Tin, while in the second situation, it is simply HR Â Tin. Using the values of HC and HR in [2], we can deduce that the maximum delta will be expressed as Ámax 2 ½0:52Tina=r0; Tout þ 1:29Tin a=r0Š, where a is the side length of the square topography and r0 is the wireless transmission range. Taking a as 1,000 m, r0 as 100 m, Tin as 5 ms, and Tout as 40 ms, Ámax will be in the range of 25-105 ms.

3. PERFORMANCE EVALUATION

We used the ns2 software to implement the SSUM system, and to also simulate the Updated Invalidation Report (UIR) mechanism [10] in addition to a version of it that is implemented on top of COACS so that a presumably fairer comparison with ESIM is done.

3.1 Network and Cache Simulation Parameters

A single database server is connected to the wireless network through a fixed access point, while the mobile nodes are randomly distributed. The client cache size was fixed to 200 Kb, meaning that a CN can cache between 20 and 200 items, while the QD cache size was set to 300 Kb, and therefore, a QD can cache about 600 queries. We used the least recently used (LRU) cache replacement policy when the cache is full and a data item needs to be cached. Each scenario started with electing one QD, but more were elected when space was needed. Each scenario lasted for 2,000 seconds and repeated 10 times with the seed of the simulation set to a new value each time, and the final result
was taken as the average of the 10 runs.
The SSUM system was implemented as a new C++ agent in ns2 that gets attached to the node class in the tcl code at simulation runtime. This implementation includes a cache class that defines and sets the needed data items as well as the operations of the caching methods that were described. Also, the routing protocols in ns2 were modified to process the SSUM packets and to implement the functions of the MDPF algorithm used for traversing the QD system [3]. Other changes to the ns2 C++ code included modifying the packet header information which is used to control the cache update process. After implementing the changes in the C++ code, tcl scripts were written to run the various described scenarios

3.2 The Query Model Parameters

The client query model was chosen such that each node in the network generates a new request every Tq seconds. When the simulation starts, each node generates a new request, and after Tq seconds, it checks if it has not received a response for the request it generated in which case, it discards it and generates a new request. We chose a default value for Tq equal to 20 seconds, but in order to examine the effect of the request rate on the system performance, we simulated several scenarios with various request rates. The process of generating a new request followed a Zipf-like access pattern, which has been used frequently to model nonuniform distributions [32]. In Zipf law, an item ranked where ranges between 0 (uniform distribution) and 1 (strict Zipf distribution). The default value of the zipf parameter was set to 0.5. Every second, the server updates a number of randomly chosen data items, equal to a default value of 20. The default values of À and were set to 1.25 and 0.75, respectively, while the default number of node disconnections is 1 every two minutes with a period of 10 seconds, after which the node returns to the network.

3.3 Varying the Number of Nodes

This section presents the effects of varying the node density in the fixed network area. Fig. 5a shows that the query delay of UIR is much greater than that of ESIM and C_UIR. The reason for this is that an issued query in ESIM does not have to wait for any report from the server, as it is always served directly after it is issued, whereas in UIR, it must wait for the next UIR to arrive from the server before getting processed. Moreover, in the event of a local cache miss in UIR, the data item must be fetched from the server, but in COACS and C_UIR, the data are next searched for in the QD system for possible network cache hit. Fig. 5b shows that the update delays of UIR and C_UIR are less than that of ESIM when there are less than 100 nodes, simply because updates in UIR are broadcasted.

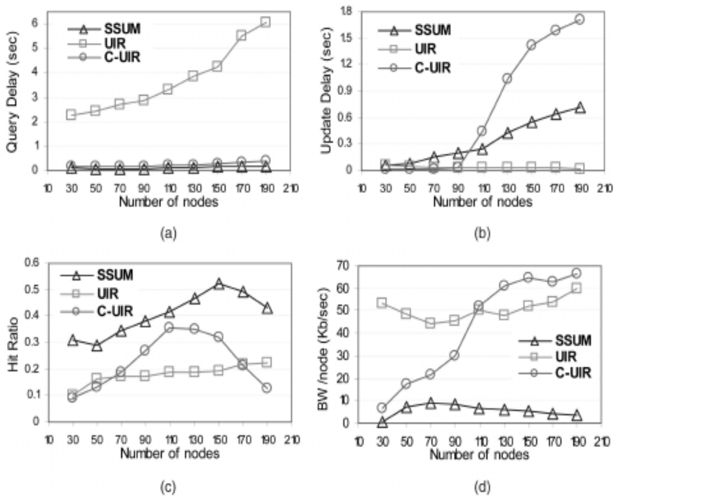

Fig. 5. Query and update delay, hit ratio, and bandwidth usage versus number of nodes.

But, when the number of nodes increases, congestion starts to buildup, which why the hit ratio of this scheme drops rapidly and its traffic increases as shown in Figs. 5c and 5d. Finally, we reiterate the fact that the presented update delay for UIR and C_UIR is not the total delay since it measures the time until the item's ID reaches the RN/CN, whereas for ESIM, it is the entire delay since it is the time until the data item itself reaches the CN.

3.4 Varying the Query Request Rate

When the request rate is increased, the query delay of ESIM rises as shown in Fig. 6a due to queuing more packets in the nodes, while the update delay decreases initially and then settles down because as more items are cached, new CNs are set up.

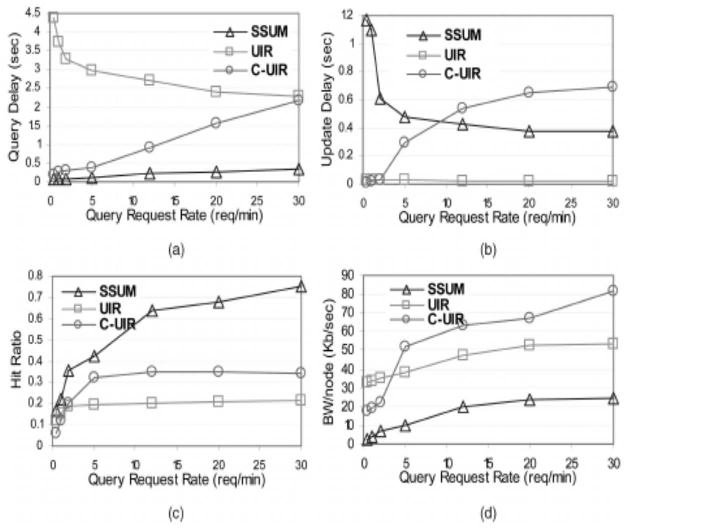

Fig. 6. Query and update delay, hit ratio, and bandwidth usage versus query request rate

This increases the probability of having more CNs closer to the access point, which, in turn, results in smaller number of hops, on average, for the update packets to reach their destinations. Unlike ESIM, the update delay of C_UIR increases due to the increase in network traffic as seen in Fig. 6d. This large traffic, which is the cumulative result of requests, update reports, and control packets, can result in congestion that delays the query and update packets of C_UIR. It can also lead to more unsuccessful requests, which, in turn, decreases the hit ratio. In contrast, the hit ratio of SSUM keeps increasing as the request rate increases for two reasons: first, the number of cached queries increases, and second, the network is not overcome by congestion as in the other two schemes. The query delay of UIR drops as its hit ratio increases while its update delay is generally unaffected due to the broadcasting of updates.

REFERENCES

[1] Y. Chung and C. Hwang, "Transactional Cache Management with Aperiodic Invalidation Scheme in Mobile Environments," Advances in Computing Science, pp. 50-61, Springer, 1999.

[2] Elmagarmid, J. Jing, A. Helal, and C. Lee, "Scalable Cache Invalidation Algorithms for Mobile Data Access," IEEE Trans.Knowledge and Data Eng., vol. 15, no. 6, pp. 1498-1511, Nov. 2003.

[3] H. Jin, J. Cao, and S. Feng, "A Selective Push Algorithm for Cooperative Cache Consistency Maintenance over MANETs,"Proc. Third IFIP Int'l Conf. Embedded and Ubiquitous Computing, Dec.2007.

[4] IEEE Standard 802.11, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification, IEEE, 1999.

[5] J. Jing, A. Elmagarmid, A. Helal, and R. Alonso, "Bit-Sequences: An Adaptive Cache Invalidation Method in Mobile Client/ServerEnvironments," Mobile Networks and Applications, vol. 15, no. 2,pp. 115-127, 1997.

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

[7] X. Kai and Y. Lu, "Maintain Cache Consistency in Mobile Database Using Dynamical Periodical Broadcasting Strategy," Proc. Second Int'l Conf. Machine Learning and Cybernetics, pp. 2389-2393, 2003.

[8] B. Krishnamurthy and C.E. Wills, "Piggyback Server Invalidation for Proxy Cache Coherency," Proc. Seventh World Wide Web (WWW) Conf., Apr. 1998.

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

[10] D. Li, P. Cao, and M. Dahlin, "WCIP: Web Cache Invalidation Protocol," IETF Internet Draft, http://tools.ietf.org/html/draft-danli-wrec-wcip-01, Mar. 2001.

[11] W. Li, E. Chan, Y. Wang, and D. Chen, "Cache Invalidation Strategies for Mobile Ad Hoc Networks," Proc. Int'l Conf. Parallel Processing, Sept. 2007.

[12] S. Lim, W.-C. Lee, G. Cao, and C.R. Das, "Performance Comparison of Cache Invalidation Strategies for Internet-Based Mobile-Ad Hoc Networks," Proc. IEEE Int'l Conf. Mobile Ad-Hoc and Sensor Systems, pp. 104-113, Oct. 2004.

[13] M.N. Lima, A.L. dos Santos, and G. Pujolle, “A Survey of Survivability in Mobile Ad Hoc Networks,” IEEE Comm. Surveys and Tutorials, vol. 11, no. 1, pp. 66-77, First Quarter 2009.

[14] H. Maalouf and M. Gurcan, “Minimisation of the Update Response Time in a Distributed Database System,” Performance Evaluation, vol. 50, no. 4, pp. 245-266, 2002.

[15] P. Papadimitratos and Z.J. Haas, “Secure Data Transmission in Mobile Ad Hoc Networks,” Proc. ACM Workshop Wireless Security (WiSe ’03), pp. 41-50, 2003.

[16] J.P. Sheu, C.M. Chao, and C.W. Sun, “A Clock Synchronization Algorithm for Multi-Hop Wireless Ad Hoc Networks,” Proc. 24th Int’l Conf. Distributed Computing Systems, pp. 574-581, 2004.

[17] W. Stallings, Cryptography and Network Security, fourth ed. Prentice Hall, 2006.

[18] D. Wessels, “Squid Internet Object Cache,” http://www. squid-cache.org, Aug. 1998.

[19] J. Xu, X. Tang, and D. Lee, “Performance Analysis of Location-Dependent Cache Invalidation Schemes for Mobile Environments,” IEEE Trans. Knowledge and Data Eng., vol. 15, no. 2, pp. 474-488, Feb. 2003.

[20] L. Yin, G. Cao, and Y. Cai, “A Generalized Target Driven Cache Replacement Policy for Mobile Environments,” Proc. Int’l Symp. Applications and the Internet (SAINT ’03), Jan. 2003.

[21] J. Yuen, E. Chan, K. Lain, and H. Leung, “Cache Invalidation Scheme for Mobile Computing Systems with Real-Time Data,”SIGMOD Record, vol. 29, no. 4, pp. 34-39, Dec. 2000.

[22] D. Zhou and T.H. Lai, “An Accurate and Scalable Clock Synchronization Protocol for IEEE 802.11-Based Multihop Ad Hoc Networks,” IEEE Trans. Parallel and Distributed Systems, vol. 18, no. 12, pp. 1797-1808, Dec. 2007.

[23] G. Zipf, Human Behavior and the Principle of Least Effort. Addison-Wesley, 1949.

[24] S. Acharya, R. Alonso, M. Franklin, and S. Zdonik, “Broadcast Disks: Data Management for Asymmetric Communications Environments,” Proc. ACM SIGMOD, pp. 199-210, May 1995.

[25] 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.

[26] H. Artail and K. Mershad, “MDPF: Minimum Distance Packet Forwarding for Search Applications in Mobile Ad Hoc Networks,” IEEE Trans. Mobile Computing, vol. 8, no. 10, pp. 1412-1426, Oct. 2009.

[27] O. Bahat and A. Makowski, “Measuring Consistency in TTL-Based Caches,” Performance Evaluation, vol. 62, pp. 439-455, 2005.

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

[29] C. Bettstetter and J. Eberspacher, “Hop Distances in Homogeneous Ad Hoc Networks,” IEEE Proc. 57th IEEE Semiann. Vehicular Technology Conf., vol. 4, pp. 2286-2290, Apr. 2003.

[30] N.A. Boudriga and M.S. Obaidat, “Fault and Intrusion Tolerance in Wireless Ad Hoc Networks,” Proc. IEEE Wireless Comm. And Networking Conf. (WCNC), vol. 4, pp. 2281-2286, 2005.

[31] J. Cai and K. Tan, “Energy-Efficient Selective Cache Invalidation,” Wireless Networks J., vol. 5, no. 6, pp. 489-502, Dec. 1999.

[32] J. Cao, and C. Liu, “Maintaining strong Cache Consistency of Cooperative Caching in the world wide web,” IEEE Trans. Computers, vol. 47, no. 4, pp. 445-457, Apr. 1998.