ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 1.852

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

Electing the Query Directory Using PSO in DCIM

S.Mahalakshmi*

* ME, Networking and Internet Engineering, Ratnavel Subramaniam College Of Engineering & Technology, Dindigul, India

mahamkce@gmail.com

Abstract

In MANET environments, data caching is essential because it increases the ability of mobile devices to access desired data, and to improve overall system performance. We propose a distributed cache invalidation mechanism (DCIM), which is client based cache consistency scheme that's enforced on prime of an antecedently proposed design for caching information things in mobile ad hoc networks (MANETs), specifically COACS, wherever special nodes cache the queries and also the addresses of the nodes that store the responses to those queries. We've additionally antecedently proposed a server-based consistency theme, whereas in this paper, we tend to introduce DCIM that's entirely client-based. DCIM could be a pull-based algorithm that implements adaptive time to live (TTL), piggybacking, and prefetching, and provides close to robust consistency capabilities. Cached information things are assigned adaptive TTL values that correspond to their update rates at the info source, wherever things with expired TTL values are classified in validation requests to the info source to refresh them, whereas valid ones however with high request rates are prefetched from the server. During this paper, DCIM is analyzed to assess the delay and information measure gains (or costs) compared to polling on every occasion and push-based schemes. To reduce the delay time we propose a selection based PSO algorithm for elect the Query Director (QD). DCIM was additionally enforced using ns2, and compared against client-based and server-based schemes to assess its performance through an experiment. The consistency quantitative relation, delay, and overhead traffic are according versus many variables, where DCIM showed better result by using PSO algorithm.

Keywords: MANET, data caching, pull-based, TTL, PSO.

Introduction

As Mobile Ad Hoc Networks (MANETs) are becoming increasingly widespread, the need for developing methods to improve their performance and reliability increases. One of the biggest challenges in MANETs lies in the creation of efficient routing techniques, but to be useful for applications that demand collaboration; effective algorithms are needed to handle the acquisition and management of data in the highly dynamic environments of MANETs.

The major issue that faces client cache management concerns the maintenance of data consistency between the cache client and the data source [1]. All cache consistency algorithms seek to increase the probability of serving from the cache data items that are identical to those on the server. 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 in. 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 [2] probabilistic consistency and probabilistic delta consistency. With weak consistency, client queries might get served with inconsistent (stale) data items, while in delta consistency, cached data items are stale for up to a period of time denoted as delta. In probabilistic consistency, a data item is consistent with the source with a certain probability denoted as p. Finally, in probabilistic delta consistency, a certain cached item is at most delta units of time stale with a probability not less than p.

In order to fetch the data and to maintain delay free communication, three types of basic algorithms were used:

(1) Push- or Server-based
(2) Pull- or client based
(3) Hybrid based.

1) In First approach, Push- or server based, content owners (server) keep track of locations and send invalidation reports (messages) or updated contents whenever the contents are modified. It informs client

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 1.852

about its updates and its cache current state. Because of server have to maintain all update records, when request arises frequently it is so hardly get the queries. It is a dangerous disadvantage in this approach.

2) In Second approach, Pull-or Client-based methods are client-based mechanism, the client checks the updates, considered outdated, are validated before serving new requests. In this client asks server to update or validate its cached data. In this every node maintains it update history, when request arises frequently, it is easy to fetch data from the nodes.

3) In Third approach, where both caching node and server cooperate to keep the data to update. Server pushes the updates or client pulls them is Hybrid based.

The pull-based policy that is most commonly used in practice is Time-to-Live (TTL), which estimates an expiration time for an object as a function of the time it was last modified. This policy assumes that objects that were recently updated in the past are more likely to be updated in the near future, and does not consider earlier updates to an object. The TTL policy works well for objects that experience bursts of updates that occur close together. However, this policy may not work well for objects with more regular and predictable behavior

In this paper, we propose a pull-based algorithm that implements adaptive TTL, piggybacking, prefetching, and provides near strong consistency guarantees. Cached data items are assigned adaptive TTL values that correspond to their query directory 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 storage capacity of query directory. This approach, which we call distributed cache invalidation mechanism (DCIM), works on top of the COACS cooperative caching architecture we introduced in [3]. Here we propose by using PSO algorithm which is selection based algorithm for electing the query directory in client side approach employing adaptive TTL and achieving superior availability, severe delays and network traffic, it needs entry list to maintain the caching data table to avoid unnecessary delays. It selects the query directory based on storage capacity by using the PSO algorithm

In the rest of this paper, Section 2 discusses related work and reveals the contributions of the proposed system, which we elaborate in Section 3. Section 4 provides an analytical analysis of the system, whereas Section 5 presents the experimental results and discusses their significance. Section 6 finishes the paper with concluding remarks and suggestions for future works.

Literature overview

Caching has its capability to improve the performance and available limitations of weakly connected and disconnected operation and thus it plays a vital role in the mobile computing. However calculating the alternative caching approach for mobile computing creates problem we introduced the in [4]. In general, the cache management in mobile environment involves the following issues to be addressed.

1. The cache discovery algorithm that is used to efficiently discover, select, and deliver the requested data item(s) from neighboring nodes.
2. Cache admission control - this is to decide on what data items can be cached to improve the performance of the caching system.
3. The cache consistency algorithm which ensures that updates are propagated to the copies elsewhere, and no stale data items are present.
4. The design of cache replacement algorithm - when the cache space is sufficient for storing one new item, the client places the item in the cache. Otherwise, the possibility of replacing other cached item(s) with the new item is considered.

Caching frequently used data objects at the local buffer of a Mobile Node is a better way to decrease query delay, save bandwidth and ameliorate system performance. However, in wireless mobile computing environments, difficulty in cache consistency arises with the frequent disconnection and roaming of an Mobile Node. A strategy that is successful must be able to efficiently handle both disconnectedness and mobility. The advantage with the broadcast is that it is able to serve an arbitrary number of Mobile Nodes with minimum bandwidth consumption. So, efficient mobile data transmission architecture should prudently design its broadcast and cache management schemes to maximize and minimize delay. Efficient mobile data transmission architecture should also be scalable, such that it works efficiently for large database systems and also upholds a large number of Mobile Nodes.

Cache Invalidation techniques, earliest scheme, used in mobile ad hoc networks to maintain consistency of data among cache and to reduce long query latency. Server-based mechanism normally implements invalidation reports (IRs), are broadcasted periodically.

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 1.852

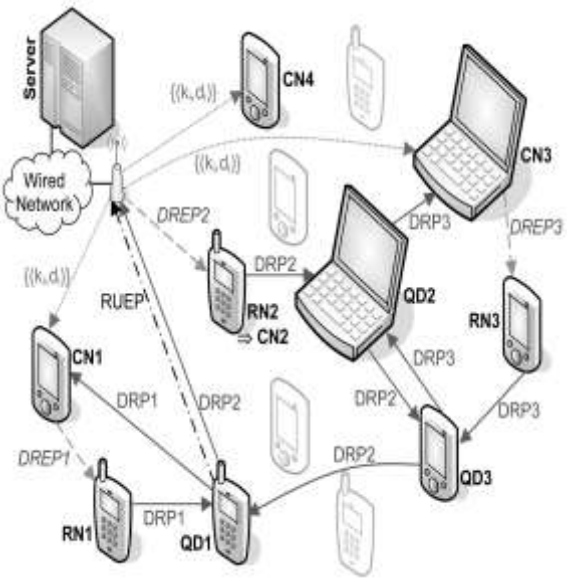

Fig 1: DCIM Design

In cache invalidation the push-based mechanism mainly uses invalidation reports (IRs). The original IR approach was proposed, but since then several algorithms have been proposed. They include stateless schemes where the server stores no information about the client caches and state-full approaches where the server maintains state information, as in the case of the AS scheme. Many optimizations and hybrid approaches were proposed to reduce traffic and latency, like SSUM [5], and the SACCS scheme where the server has partial knowledge about the mobile node caches, and flag bits are used both at the server and the mobile nodes to indicate data updates. Such mechanisms necessitate server side modifications and overhead processing. More crucially, they require the server to maintain some state information about the MANET, which is costly in terms of bandwidth consumption especially in highly dynamic environments. DCIM, on the other hand, belongs to a different class of approaches, as it is a completely pull-based scheme. Hence, we will focus our survey of previous work on pull-based schemes, although we will compare the performance of DCIM with that of our recently proposed push-based approach, namely SSUM [5], in Section 5.

Push-based strategies are more suitable to a stable network. They can provide good consistency guarantees for users who are always online and reachable from the source. However, such strategies have low query latency and cannot solve the disconnection problem. If the cache nodes are disconnected from the network, they cannot receive the invalidation messages and will share the stale data upon reconnection.

The client processes generated queries one by one. If the referenced data items are not cached on the client side, the data *id*s are sent to the server for fetching the data items. Once the requested data items arrive on the channel, the client brings them into its cache. Client cache management follows the LRU replacement policy, but there are some differences between the TS (or BS) algorithm and our algorithm. In the TS algorithm, since the clients will not use the invalid cache items, the invalidated cache items are first replaced. If there is no invalid cache item, LRU is used to replace the oldest valid cache item. In our algorithm, if there are invalid data items, the client replaces the oldest invalid item. If there is no invalid cache item, the client replaces the oldest valid cache item. The difference is due to the fact that the clients in our algorithm can download data from the broadcast channel. The IP TTL is used, somewhat schizophrenically, as both a hop count limit and a time limit. Its hop count function is critical to ensuring that routing problems can't melt down the network by causing packets to loop infinitely in the network. The time limit function is used by transport protocols such as TCP to ensure reliable data transfer. Many current implementations treat TTL as a pure hop count, and in parts of the Internet community there is a strong sentiment that the time limit function should instead be performed by the transport protocols that need it.

An example of pull approaches is the time to live (TTL)-based algorithms, where a TTL value is stored alongside each data item d in the cache, and d is considered valid until T time units go by since the last update. Such algorithms are popular due to their simplicity, sufficiently good performance, and flexibility to assign TTL values to individual data items [6]. 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.

In this specification, we have reluctantly decided to follow the strong belief among the router vendors that the time limit function should be optional. They argued that implementation of the time limit function is difficult enough that it is currently not generally done. They further pointed to the lack of documented cases where this shortcut has caused TCP to corrupt data (of course, we would expect the problems created to be rare and difficult to reproduce, so the lack of documented cases provides little reassurance that there haven't been a number of undocumented cases). IP multicast notions such as the

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 1.852

expanding ring search may not work as expected unless the TTL is treated as a pure hop count [7]. The same thing is somewhat true of trace route.

Proposed work

Cache Invalidation

As the "Cache Invalidation Techniques" researchers have proposed a wide range of strategies for maintaining cache consistency. However, each strategy has its own design goals and application scenarios. No uniform or structured methods exist for current work, making it difficult to evaluate their relative effectiveness and performance. Cache invalidation techniques have been widely used in distributed systems to maintain data consistency among caches [8].

However, a critical design issue, *cache invalidation*, needs to be addressed for applications requiring data consistency with the server. When a data item in a server is updated, it is necessary to make sure that the cached copies of this data item are validated before they can be used.

COACS

The proposed DCIM system builds on top of COACS, which we introduced in [7] 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 [9]. 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. The operations of the QDs and CNs are described

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 elsewhere in the network), and thus is more suitable to dynamic MANETs than a push-based 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[10 , 11]. It also estimates the inter-request interval for each data item to predict its next request time, and then prefetches items that it expects to be requested soon.

Time-to-live

Caching mechanisms in Internet systems are designed to scale to large numbers of caches. A common way of achieving scalable caching is to use *time-to-live (TTL)-based caches*, which work as follows: for any data item *D*, the site that maintains the current, authoritative version of *D* is called the *origin*. If the origin receives a request for *D* at time *t* it returns the current version along with a TTL period, *T*. The *requestor*, which is a cache used by one or more clients or client caches, is allowed to cache *D*. Any subsequent requests at the request or in the time interval (*t*, *t* + *T*) can instead be served from the requestor's cache *without* contacting the origin site. However, the first request after time *t+T must* go to the origin site, since the TTL has expired for *D* in the requestor's cache. A time-to-live (TTL) interval *t* is defined for data entries stored in the wireless handheld device. The TTL for a data entry is determined based on whether the data entry is modified.

TTL-based caching scales well because origin sites neither have to maintain any per-requestor (*i.e.*, per-cache) state, nor even have to know of the existence of caches. This also enables "opportunistic caching" by caches across the Internet.

Pull-based freshness has also been addressed in the context of synchronizing a large collection of objects, e.g., improving the performance of web crawlers [4, 8, 14]. Updates are detected by periodically prefetching objects from remote sources to maximize the freshness of cached objects. These pull-based policies are not based on complete update histories and therefore may be less accurate.

Query Directory (QD)

In distinction to the CNS that becomes caching nodes after they initial request non-cached information, QDs square measure non-appointive supported their resource capabilities, as delineated. A procedure is enclosed in [11] that explain however the quantity of QDs within the system is delimited by 2 limits. The boundary corresponds to having enough QDs, such a further (elected) QD won't yield A considerable reduction in average QD load. The edge, on the opposite hand corresponds to a delay threshold, since traversing a bigger range of QDs can cause

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 1.852

higher response times. Between these limits, the quantity of QDs will modification dynamically betting on what quantity of the QD storage capability is employed. Within the simulations performed during this work, the quantity of QDs averaged seven at steady state once the quantity of nodes was one hundred. Regarding load, it had been shown in [6] that the common load on a QD node is one.5 times the common load on a CN node. Since in DCIM the QDs don't seem to be assigned further roles as compared, their average load mustn't modification. In general, the QD system yields a high hit rate that causes the request to traverse less QDs on the average, and consequently keeps the load per QD fairly low. Moreover, aside from the occasional COACS updates concerning replaced things (due to capability constraints), the CNS ne'er update the QDs concerning the invalid information things. Consequently, the QD perpetually forwards the DRP to the CN just in case of a success. This makes the system easier and saves traffic, however would possibly incur further delay on condition that the item is invalid at the CN, as this can cause the CN to contact the server to update the info item. However, this extra internal delay is little in comparison to the server delay, and it's remunerated by reduced consistency updates from the CNS to the QDs. Also, we have a tendency to show later within the experimental analysis section that the system maintains a suitable hit rate, which means that the chance an information item are invalid at the CN is low. By victimization the PSO algorithmic rule we are able to elect the question directory for store the info in QDs.

(i) PSO algorithm

When the search space is too large to search exhaustively, population based searches may be a good alternative, however, population based search techniques cannot guarantee you the optimal (best) solution.

Particle Swarm has two primary operators: Velocity update and Position update. During each generation each particle is accelerated toward the particles previous best position and the global best position. Each iteration new velocity value for each particle is calculated based on its current velocity in [8], the distance from its previous best position, and the distance from the global best position. The new velocity value is then used to calculate the next position of the particle in the search space. This process is then iterated a set number of times or until a minimum error is achieved.

Experimental results

DCIM was implemented using ns2, and a replacement into class was developed that mimics the server method storing in Query directory and process the validation requests. Timers were utilised to implement the monitoring thread: the timer sleeps for the polling interval duration and so wakes up to run the innerloop operate, i.e., when Npoll runs of the innerloop, (the piggybacking interval) the outer-loop is invoked. Ns2 may be a single rib simulator, however it's all the same capable of dominant the operations of the timers autonomously, so acting similar to a multithreaded application [12, 13, 14].

Two approaches were implemented for reduce the traffic in MANET DCIM architecture: the Poll based approach which is the poll-every-time mechanism, where each time data is requested, it is validated and check expire interval time. Then another is calculating the TTL value for adding to the current time to the expiry interval, when TTL value was expired the item is flagged and it is fetched from server in [4]. We can select the query directory by using the PSO algorithm for allotting the space for the data that we fetch already.

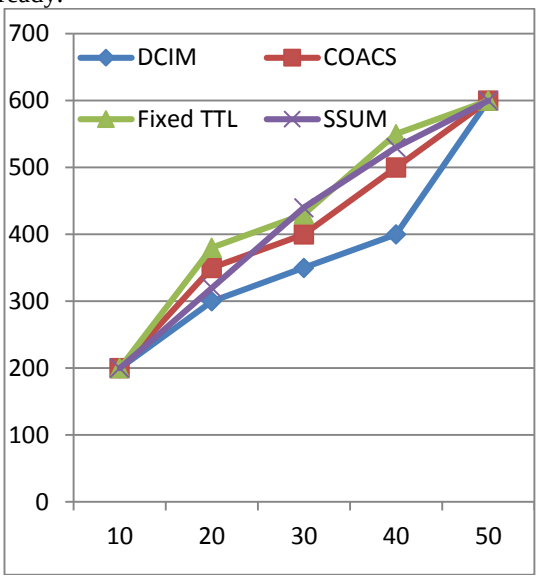

Fig 2: Performance evaluation of DCIM mechanisms

In Fig 2: it explains the performance evaluation of DCIM method. How the method is perform well when compared to other technique like Fixed TTL, SSUM, and COACS. It improves the consistency ratio when compare to other technique.

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 1.852

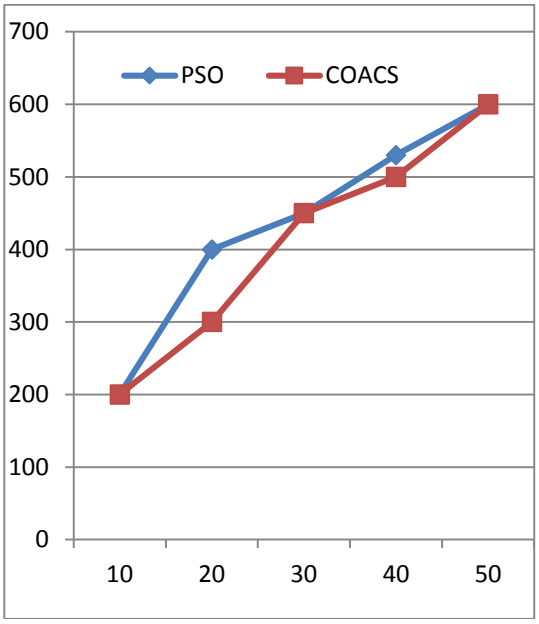

Fig 3: Delay traffic caching

In Fig: 3 it reduce the delay of traffic, so that the data can send easily from server to client by using the TTL value by PSO algorithm it select the storage device in Query Directory so it can reduce the delay of traffic when compare to existing one.

Comparing

Comparing of this entire algorithm, PSO algorithm is more efficient. The general architecture of mobile ad hoc networks consists of caching node used to store the wandering data in order to provide requesting query data in query directory to any other prescribed request nodes in [7]. By using the query directory it elect the storage space by using PSO algorithm. It is one of the best one to select the storage space in query directory. Here Base Station transfer queries to all the entire nodes and each of the Access Point (AP) acts as a gateway in order to transmit the data to every Nominal Node (Requesting Node).Through which by using PSO client pulling may solve the traffic delay.

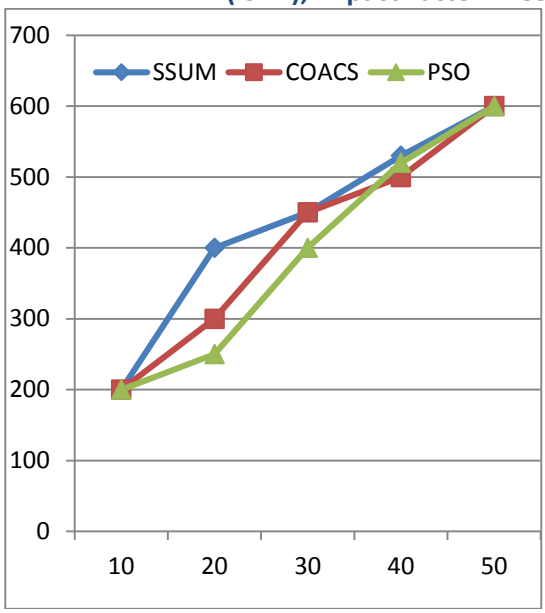

Fig 4: Performance comparison of SSUM, COACS, and PSO.

COACS achieves consistency with a delta equal to the communication time between the server and caching nodes illustrated in Fig.3. DCIM also provides consistency guarantees if the polling interval is not high, with a comparable generated traffic.

For traffic, in SSUM it's because of maintaining the server state, and pushing knowledge things proactively, whereas in DCIM it's as a result of validation requests and proactive taking of things. UIR provides powerful consistency however at the expense of additional traffic and deteriorated knowledge handiness, whereas DCIM on the opposite hand provides close to sturdy consistency as was shown earlier, however with significantly lower traffic and better knowledge availability. Moreover, Fig. three shows the effectiveness of PSO as a caching system in providing knowledge handiness while keeping traffic within the network low when put next to other caching systems. One will sit down with [2] for elaborate analysis of COACS in PSO and comparisons to existing systems. We conclude the experimental results with Table one, which summarizes key properties of DCIM and compares them to the given pull-based approaches and SSUM.

TABLE 1: Comparison between DCIM and Other Approaches

Property	SSUM	FIXED TTL
Type	Push(server side)	Pull(Client side)
Query Delay	Low	Based on TTL Value
Consistency	High	Depending on TTL
Server	Medium	Low to

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 1.852

traffic		Medium
Property	COACS	PSO
Type	Pull based	Storage
Query Delay	Low	Low
Consistency	medium	Depending on storage
Server traffic	Low	Very low

Conclusion

We presented a Pull based cache consistency scheme for MANETs that relies on estimating the inter update intervals of data items to set their expiry time. The use of PSO algorithm is to increase the accuracy of its consistency ratio, traffic, and query delay. We compare this approach to TTL and server based approach (SSUM and UIR). This shows a better result when compare to existing one. 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 [1] 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 using PSO algorithm, QD traversal, QD and CN information integrity, and TTL monitoring and calculation. The first three can be mitigated through encryption and trust schemes. The last issue was not tackled before, except in the case of [10].

References

1. *Kassem Fawaz, "DCIM: Distributed Cache Invalidation Method for Maintaining Cache Consistency in Wireless Mobile Networks", 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. *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.*
4. *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.*
5. *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.*
6. *Q. Hu and D. Lee, "Cache Algorithms Based on Adaptive Invalidation Reports for Mobile Environments," Cluster Computing, vol. 1, pp. 39-50, 1998.*
7. *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.*
8. *Y. Fang, Z. Haas, B. Liang, and Y.B. Lin, "TTL Prediction Schemes and the Effects of Inter-Update Time Distribution on Wireless Data Access," Wireless Networks, vol. 10, pp. 607-619, 2004.*
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. *T. Hara and S. Madria, "Dynamic Data Replication using Aperiodic Updates in Mobile Ad Hoc Networks," Proc. Database Systems for Advanced Applications, pp. 111-136, 2004.*
11. *W. Zhang and G. Cao, "Defending Against Cache Consistency Attacks in Wireless Ad Hoc Networks," Ad Hoc Networks, vol. 6, pp. 363-379, 2008.*
12. *K. Fawaz and H. Artail, "A Two-Layer Cache Replication Scheme for Dense Mobile Ad Hoc Networks," Proc. IEEE Global Comm. Conf. (GlobeCom), Dec. 2012.*
13. *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.*
14. *G. Cao, L. Yin, and C. Das, "Cooperative Cache-Based Data Access in Ad Hoc Networks," Computer, vol. 37, no. 2, pp. 32-39, 2004.*
15. *P. Papadimitratos and Z. Haas, "Secure Data Transmission in Mobile Ad Hoc Networks," Proc. ACM Workshop* Wireless Security, pp. 41-50, 2003.