ISSN:0975-9646
J.Arumai Ruban et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 3 (3) , 2012,3910-3916

Using Propagation Techniques to Enhance Cache Consistency in Wireless Ad-hoc Networks

J.Arumai Ruban , D.Selvam , L.Dinesh

Department of Computer Science and Engineering, Periyar Maniammai University

Thanjavur, Tamil Nadu, India

Abstract- Caching is an important technique to support pervasive Internet access. To reduce data access cost and delay, caching has been widely used as an important technique. In this paper, we propose a general consistency model called Probabilistic Delta Consistency (PDC), which integrates the flexibility granted by existing consistency models, covering them as special cases. We also propose the Flexible Combination of Push and Pull (FCPP) algorithm which satisfies user-specified consistency requirements under the PDC model. The analytical model of FCPP is used to derive the balance of minimizing the consistency maintenance cost and ensuring the specified consistency requirement.The cached nodes communicate their version numbers to their peers. The cached nodes can be made to replicate the messages sent to them from the access point to all other nodes. When a cached system receives an invalidation message, it can be made to pass this value to all its peers. The communication between cached nodes can be further optimized by periodically communicating the values using a timer. All the previously mentioned process can be optimized by passing the data to the nearest neighbours instead of all the nodes.

Index Terms- Cache Consistency, PDC Model, Virus Propagation.

I. INTRODUCTION

On wireless computer networks, ad-hoc mode is a method for wireless devices to directly communicate with each other. Operating in ad-hoc mode allows all wireless devices within range of each other to discover and communicate in peer-to-peer fashion without involving central access points (including those built in to broadband wireless routers to set up an ad-hoc wireless network, each wireless adapter must be configured for ad-hoc mode versus the alternative infrastructure mode. In addition, all wireless adapters on the ad-hoc network must use the same SSID and the same channel number. When a cached system receives an invalidation message, it can be made to pass this value to all its peers and hence the timer is updated for all the cached nodes. This helps in maintaining consistent and up dated data among all the nodes. This helps in maintaining consistent and up dated data among all the nodes.

Example of Ad Hoc Network

Example 1: In a battlefield, an ad hoc network may consist of several commanding officers and a group of soldiers around the officers. Each officer has a relatively powerful data center, and the soldiers need to access the data centers to get various data such as the detailed geographic information, enemy information, and new commands. The neighboring soldiers tend to have similar missions and thus share common interests. If one soldier accessed a data item from the data center, it is quite possible that nearby soldiers access the same data some time later. It saves a large amount of battery power, bandwidth, and time if later accesses to the same data are served by the nearby soldier who has the data instead of the faraway data center.

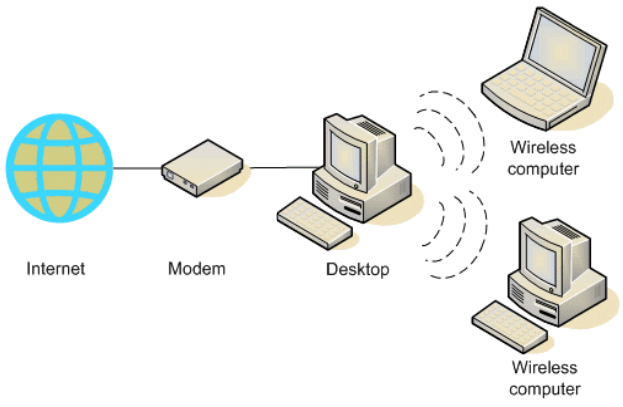


Fig 1.1 Ad Hoc Network

Example 2:

Recently, many mobile info station systems have been deployed to provide information for mobile users. This info stations deployed by tourist information center may provide maps, pictures, history of attractive sites. Info station deployed by a restaurant may provide menus. Due to limited radio range, an info station can only cover a limited geographical area. If a mobile user, says Jane, moves out of the info station range, she will not be able to access the data provided by the info station. However, if mobile users are able to form an ad hoc network, they can still access the information. In such an environment, when Jane's request is forwarded to the info station by other mobile users, it is very likely that one of the nodes along the path has already cached the requested data. Then, this node can send the data back to Jane to save time and bandwidth.

II. LITERATURE SURVEY

A) Achieving Flexible Cache Consistency for Pervasive Internet Access

They consider a commonly used system model where each data object is associated with a single data source node. Only the data source node can update the source. Each data object is cached by a collection of caching nodes. The data copies held by the caching nodes are called cache copies. There are two basic mechanisms for cache consistency maintenance: push and pull. Using push, the data source node proactively informs the caching nodes of cache information. Using pull, a caching node fetches cache information from the data source node. We also assume that the data source node and the caching nodes have synchronized clocks.

B) Supporting Cooperative Caching in Ad Hoc Networks

Most researches in ad hoc networks focus on routing and not much work has been done on data access. A common technique used to improve the performance of data access is caching. Cooperative caching, which allows the sharing and coordination of cached data among multiple nodes, can further explore the potential of the caching techniques. Due to mobility and resource constraints of ad hoc networks, cooperative caching techniques designed for wired networks may not be applicable to ad hoc networks. In this paper, they design and evaluate cooperative caching techniques to efficiently support data access in ad hoc networks. They first propose two schemes: CacheData, which caches the data, and CachePath, which caches the data path. After analyzing the performance of those two schemes, they propose a hybrid approach (HybridCache), which can further improve the performance by taking advantage of CacheData and CachePath while avoiding their weaknesses. Cache replacement policies are also studied to further improve the performance. Simulation results show that the proposed schemes can significantly reduce the query delay and message complexity when compared to other caching schemes. On the other hand, our evaluation results show that FCPP effectively satisfies arbitrarily specified consistency requirements. The results also show that FCPP can save up to 50 percent of the traffic overhead and reduce the query delay by up to 40 percent, compared with the widely used Pull with TTR algorithm.

C) Maintaining Strong Cache Consistency in the World-Wide Web

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,* using prototype implementation and trace replay in a simulated environment. Our results show that *invalidation* generates less or a comparable amount of network traffic and server workload than *adaptive TTL* and has a slightly lower average client response time, while *polling-every-time* generates more network traffic 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 invalidation based protocol.

III. IMPLEMENTATION

A. Maintain Cache Consistency:

There is a cache consistency issue in both Cache Data and Cache Path. We have done some work [9], [10] on maintaining strong cache consistency in single-hop based wireless environment. However, due to bandwidth and power constraints in ad hoc networks, it is too expensive to maintain strong cache consistency, and the weak consistency model is more attractive. A simple weak consistency model can be based on the Time-To-Live (TTL) mechanism, in which a node considers a cached copy up-to-date. If it's TTL has not expired, It removes the map from its routing table (or removes the cached data) if the TTL expires. As a result, future requests for this data will be forwarded to the data center. Due to TTL expiration, some cached data may be invalidated. Usually, invalid data are removed from the cache. Sometimes, invalid data may be useful. As these data have been cached by the node, it indicates that the node is interested in these data. When a node is forwarding a data item and it finds there is an invalid copy of that data in the cache, it caches the data for future use. To save space, when a

cached data item expires, it is removed from the cache while its id is kept in "invalid" state as an indication of the node's interest. Certainly, the interest of the node may change, and the expired data should not be kept in the cache forever. In our design, if an expired data item has not been refreshed for the duration of its original TTL time (set by the data center), it is removed from the cache.

B. Probabilistic Delta Consistency

PDC, users can specify their consistency requirements in two orthogonal dimensions. The dimension along the x-axis specifies the value, which denotes the maximum acceptable deviation (in time, value, etc.) between the source data and the cache copies; the dimension along the y-axis specifies the probability p, which represents the minimum ratio of queries served by consistent cache copies.

B.1) Applying PDC

Users can flexibly specify their consistency requirements under the PDC model in different scenarios. For example, in the scenario discussed in Section 1, the user accessing stock prices can assure that the cached prices will not be too stale by specifying a small (e.g., 1 minute). Since he frequently checks the prices, he might be able to tolerate that a small portion of accesses not satisfying the requirement on. He can specify a p value slightly less than 100 percent (e.g., 95 percent). Meanwhile, the user accessing the weather forecast information can specify a large _ such as 2 hours, since the weather forecast information is relatively stable. The user may not check the weather information frequently. So, he may set a high p value such as 90 percent. In the other scenario, since the traffic information cannot change dramatically in a short time and the taxi driver may frequently access the traffic information, the driver may specify less stringent requirement on both the deviation (e.g., 10 minutes) and the ratio p (e.g., 70 percent).

C. The Flexible Combination Of Push And Pull Algorithm:

Details of the proposed Flexible Combination of Push and Pull (FCPP) algorithm are presented in this section. In FCPP, we consider a commonly used system model, where each data object is associated with a single node which can update the source data. This node is referred to as the data source. Each data object can be cached by a group of nodes called caching nodes. The data copies held by the caching nodes are called cache copies. There are mainly two basic mechanisms for achieving cache consistency: push and pull. Using push, the data source informs every caching node of the data update. Using pull, for each data access, a caching node sends a request to the data source to check if the cache copy is up-to-date.

C.1) Design Aspects

In designing FCPP, we focus on three design aspects, as discussed in detail below:

Consistency Level.

To enable users to flexibly trade cache consistency for reduced We need to provide them multiple consistency levels with fine granularity. We have addressed this issue in design of the PDC model. PDC enables the users to continuously tune their consistency requirements in two orthogonal dimensions.

Update Delay.

Existing schemes mainly focus on how cache consistency should be maintained after the data source node has updated the source data. In such schemes, the data source node can directly update the data without considering consistency maintenance. However, in many cases, the data source node can wait for certain time before updating the source data, as in the Two Phase Commit (2PC) protocol and the Lease protocol. The update delay can be utilized to further decrease the consistency maintenance cost, as in Lease. However, Lease does not provide any bound on how long the data source node needs to wait before updating the source data. Between the two extremes of no update delay an unbounded update delay, FCPP supports declarative update delay (denoted by a system parameter D).

Consistency Control

A simple but widely used approach to consistency control is to associate time-out values with cache copies. When the time-out values expire, the caching nodes renew the time-out values from the data source node. Upon a data update on the data source node, it first needs to push invalidations to the caching nodes. Since the invalidations may be lost, especially in dynamic wireless ad hoc networks, the caching nodes are required to send back the acknowledgments.

Algorithm 1 FCPP on a caching node

Upon receiving a query

(1) IF (l > 0) serve the query with the cache copy;
(2) ELSE // l has decreased to zero
  (2.1) Send a RENEW message to renew the
    Timeout from the data source and update the
    Cache copy;

(2.2) after l is renewed, serve the query with the Updated cache copy;

Algorithm 2 FCPP on the data source

When the source data is ready to be updated
(1) Send an INV message to each caching node With positive l;
(2) IF (receive an INV_ACK message for each INV message OR has waited for D seconds)
(2.1) update the source data;
Upon receiving a RENEW message from a Caching node
(3) Grant timeout with duration l and the source Data to the caching node;-

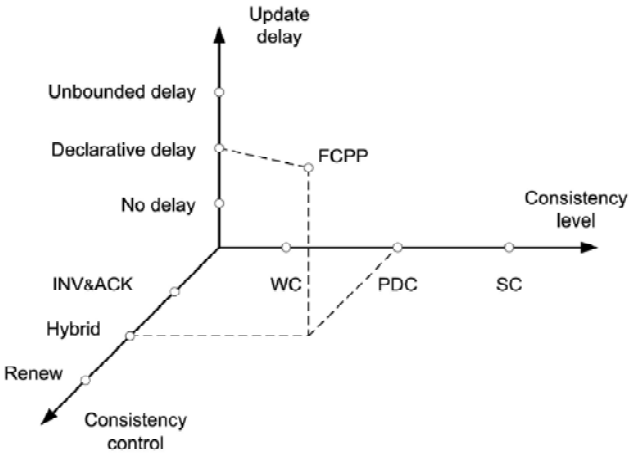


Fig 5.1 The 3D design framework

C.2) FCPP as a Generic Scheme

FCPP is a generic scheme, covering many existing schemes as its special cases:

- When the data source node can always delay the update until it receives all the ACKs, or the time-out values on all unresponding caching nodes expire, FCPP provides Strong Consistency and covers the Lease protocol as its special case.
- When the time-out value l is set to zero, FCPP transforms to the Pull each read scheme.
- When the data source node lets l be sufficiently large, FCPP transforms to the Push with ACK scheme.

The key issue in tuning FCPP is to quantify the trade-off between consistency requirements PDC (δ, p) and the consistency maintenance cost.

D. The Flexible Combination of Push And Pull Algorithm:

D.1) Communication between Caching Nodes

The cached nodes are made to communicate such that their version numbers can be kept updated. This helps in maintaining consistency even if there is a missed acknowledgement.

D.1.1) Shortest Path Algorithm

Let the node at which we are starting be called the initial node. Let the distance of node Y be the distance from the initial node to Y. Dijkstra's algorithm will assign some initial distance values and will try to improve them step by step.

1. Assign to every node a tentative distance value: set it to zero for our initial node and to infinity for all other nodes.
2. Mark all nodes except the initial node as unvisited. Set the initial node as current. Create a set of the unvisited nodes called the unvisited set consisting of all the nodes except the initial node.
3. For the current node, consider all of its unvisited neighbors and calculate their tentative distances. For example, if the current node A is marked with a distance of 6, and the edge connecting it with a neighbor B has length 2, then the distance to B (through A) will be 6+2=8. If this distance is less than the previously recorded distance, then overwrite that distance. Even though a neighbor has been examined, it is not marked as visited at this time, and it remains in the unvisited set.
4. When we are done considering all of the neighbors of the current node, mark it as visited and remove it from the unvisited set. A visited node will never be checked again; its distance recorded now is final and minimal.
5. The next current node will be the node marked with the lowest (tentative) distance in the unvisited set.
6. If the unvisited set is empty, then stop. The algorithm has finished. Otherwise, set the unvisited node marked with the smallest tentative distance as the next "current node" and go back to step 3.

D.2) Using Replicating Cached Nodes

This module helps in providing the cached with the possibility to communicate with their peers the data that they have received

from the access point. This can be further made to communicate the invalidation messages and the timer settings.

D.3)Optimization of Communication between Cached Nodes Using a Timer

This communication can be further optimized by using a timer to maintain the communication time between the access point and the cached nodes, and between the cached nodes.

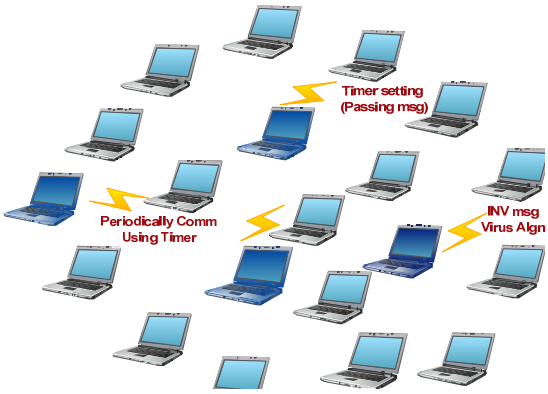


Fig 6.1 Propagation in Ad Hoc Model

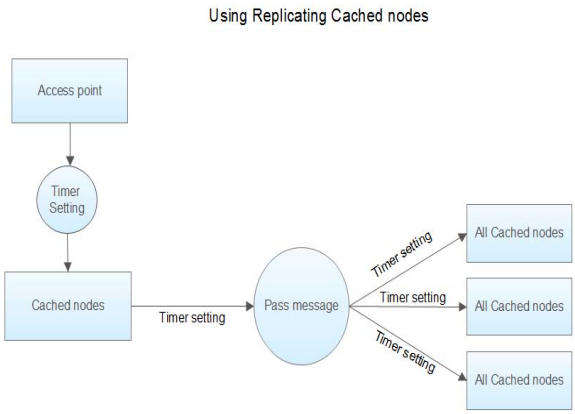


Fig 6.2 Using Replicating Cached Nodes

D.4. INV Message

When a cached system receives an invalidation message, it can be made to pass this value to all its peers and hence the timer is updated for all the cached nodes. This helps in maintaining consistent and updated data among all the nodes.

D.4.1) Virus Propagation Algorithm[10]

- Receive data from Source node
- All the other nodes that are present in the current network are listed
- Every node is initially checked if it contains the updated data, if so the propagation is halted and the remaining nodes are checked for the updated data
- If a node does not contain the updated data, then it is updated and the nodes that are connected to the currently updated node are analyzed.
- This process of analyzing a node before attacking the node is called Pre scanning, which is used for Improved Target Selection performed by the Blaster algorithm[19].
- These processes are carried out by using the UDP connection instead of a TCP connection.
- In addition to not having to go through a TCP handshake and not having to keep a state, sessions and sockets allocated, this also allowed for a faster updation of targets.
- This process results in higher infection speed carried out by the Sapphire/Slammer worm [19].
- This process can be further optimized by analyzing only the nodes connected to the currently connected nodes.
- This can effectively reduce the time taken for analyzing the node that is not updated. This process is called Subdividing.
- This process can be used for transmission of information through the network.
- For every node receiving the message,
  If the node does not have the updated copy,
  go back to step 2
  Else makes it a dead end and
  stops the data propagation.

D.5. Analysis of the Optimized Timeout Duration of FCPP:

In this section, we derive an analytic model to study the relationship between the timeout duration l and the consistency requirement PDC (δ, p), as well as the relationship between l and the traffic overhead. Then we calculate the optimized timeout duration which satisfies the user-specified PDC with minimum traffic overhead. In the analysis, we assume that the data update and the query follow Poisson Process. The number of hops counted in data transmission is used to measure the traffic overhead.

Cost for Time-out Renewal.

When a query comes after the time-out value expires, the caching node first renews the time-out value to l, and then, serves the next l .r (on average) queries. Thus, for every l .r +1 queries, there will be onetime-out renewal. So, we average the cost over the queries and obtain the expected renewal cost per unit time:

2. $\bar{h}$ .N. r / (l.r+1)

Cost for the INV & ACK Process.

Concerning the data updates, for every l .r +1 query on a cache copy, l.r of them occur when the time-out value is valid. The probability of having a valid time-out value is l .r / (l. r + 1). As long as the time-out value is valid, the data source node needs the INV & ACK process upon a data update. Thus, the expected INV & ACK cost per unit time is:

2. $\bar{h}$ ..N. ω. l. r / (l.r+1) According to the discussions above, we obtain the total consistency maintenance cost per unit time:

$$C=2N.\ \bar{h}\ \frac{r+\omega.l.r}{l.r+1}=2r.N.\ \bar{h}\frac{1+\omega.l.}{l.r+1}$$

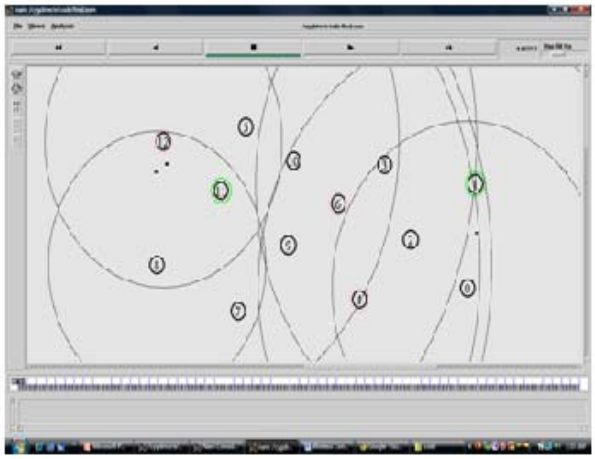
The notations used in the analytical model are listed in Table 1.

Table 1 Notations used in the analytical model

$\bar{h}$	Average path length between data source node and caching node
N	No of caching nodes.
ω	Average data update rate.
l	Time out duration.
r	Average cache query rate

IV .SCREENSHOTS

PROPAGATION IN AD-HOC NETWORK



UDP PACKET LOSS

IV. CONCLUSION

In this paper, we have addressed the problem of how to provide the users with flexibility in specifying their consistency requirements, and how to satisfy the flexible user-specified consistency requirements with minimum overhead. Toward this objective, our contributions can be described as follows: (1) we have proposed a general consistency model PDC, allowing users to flexibly specify their consistency requirements in two

orthogonal dimensions; (2) we have developed the FCPP algorithm to maintain cache consistency under the PDC consistency model. FCPP flexibly and efficiently combines push and pull based on timeouts; (3) we have derived an analytical model for FCPP to calculate the optimized timeout duration, so as to provide user-specified PDC with minimum traffic overhead in IMANETs. In our future work, we will study how to enable efficient cooperation among the caching nodes, in order to further reduce the cache maintenance cost. We also plan to study how to satisfy heterogeneous consistency requirements of the users.

REFERENCES

[1]. Flexible Cache Consistency Maintenance over Wireless Ad Hoc Networks- IEEE transactions on parallel and distributed systems, vol. 21, no 8, august 2010.

[2]. L. Yin and G. Cao, "Supporting Cooperative Caching in Ad Hoc Networks," IEEE Trans. Mobile Computing, vol. 5, no. 1, pp. 77-89,Jan. 2006.

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

[4]. 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.

5] J. Cao, Y. Zhang, L. Xie and G. Cao, Data Consistency for Cooperative Caching in Mobile Environments, to appear in IEEE Computer.

[6] V. Duvvuri, P. Shenoy and R. Tewari, Adaptive Leases: A Strong Consistency Mechanism for the World Wide Web, IEEE Trans. on Knowledge and Data Engineering, Vol. 15, No. 4, 2003.

[7] J.Lan, X. Liu, P. Shenoy and K. Ramamritham,Consistency Maintenance in Peer-to-Peer File Sharing Networks, the 3rd IEEE Workshop on Internet Applications, 2003.

[8] J. Cao, Y. Zhang, L. Xie, and G. Cao, Consistency of Cooperative Caching in Mobile Peer-to-Peer Systems Over MANET, Intl. J. of Parallel, Emergent, and Distributed Systems, Vol. 21, No. 3, June 2006.

[9] Y. Huang, J. Cao and B. Jin, A Predictive Approach to Achieving Consistency in Cooperative Caching in MANET, in Proc. of the 1st Intl. Conf. on Scalable Information Systems, P2PIM workshop session, ACM Press, New York, USA, 2006.

[10] Simulating and optimising worm propagation algorithms Tom Vogt <tom@lemuria.org>29th September 2003 (updated 16th February 2004)

[11] M. Corson, J. Macker and G. Cirincione, Internet-based Mobile Ad Hoc Networking, in IEEE Internet Computing, pp.63-70, July-August, 1999.

First Author – Mr.J.Arumai Ruban, He has Completed M.C.A from St.Joseph's College Thiruchirappali on 2007-2010, TamilNadu, India .He is currently pursuing his M.E (Computer Science and Enginering) in Periyar Maniammai University, Thanjavur, TamilNadu, India. He has presented several papers in national and international conferences.
rubsjoe@gmail.com

Second Author – **Mr. D.Selvam,** Working as Assistant Professor in Periyar Maniammai University. Completed BE (CSE) from Sri Krishna College of Engineering and Technology, Coimbatore on 1998-2002 and ME (CSE) through Gate 2006 from PSG TECH, Coimbatore on 2006-2008, He has published an international Journal in IJCA 2010 and presented several papers in national and international conferences.
seldurai999@gmail.com

Third Author – **Mr. L. Dinesh** He received his M.Sc Degree [5 ears Integrated] in Software Engineering from Anna University, TamilNadu, India and also Completed MBA in Human Resource and Finance. He is currently pursuing his M.E (Software Engineering) in Periyar Maniammai University, Thanjavur, TamilNadu, India. He has published an international Journal in IJCSI, November 2011 and presented several papers in international and national conferences.
l_dinuma@yahoo.co.in

y