A FRAMEWORK FOR AN APPLICATION BASED MOBILE CACHE CONSISTENCY METHOD

[1]DOHA. ELSHARIEF, [2]HAMIDAH. IBRAHIM, [3]ALI. MAMAT, [4]MOHAMED OTHMAN

Faculty of Computer Science and Information Technology
University Putra Malaysia
Department of Computer Science, Faculty of Computer Science and Information Technology, University Putra Malaysia, 43400 Serdang, Selangor,
E-mail: [1]dohayagoub@yahoo.com, ([2]hamidah, [3]ali, [4]mothman)@fsktm.upm.edu.my

ABSTRACT

In a mobile environment, maintaining cache consistency is challenging. Applying one type of consistency levels either strict or weak is not suitable all the time, as the consistency requirements mainly depend on the mobile application system and differ from one to another. Also forcing the mobile client to use its cache data for the purpose of reading only limits the functionality of the caching. The stateful scheme Multi-level Mobile Cache Consistency Protocol that works in client-server architecture supports different levels of consistency. The Mobile client is able to issue updates transactions, and determine the consistency requirements upon its interest. Based on the Multi-Level Mobile Cache Consistency Protocol this paper presents a framework of stateful strategy; Application Based Multi-level Mobile Cache Consistency Method (ABMMCCM) that preserves the advantages of multi-level mobile cache consistency protocol and enhances its drawbacks. In ABMMCCM the consistency requirements are designed at the server side based on the application requirements, and each data item has a single consistency requirement entry. The proposed framework is initially compared to Multi-level Mobile Cache Consistency Protocol, and it appears that ABMMCCM reduces the number of messages transfer between the base server and the mobile client, which helps in better utilizing the wireless network, and reduces the overhead from the mobile client and the base server.

Keywords: *Mobile Cache Consistency, Stateful Approach, Multi-level Consistency Method*

1 INTRODUCTION

The advances in wireless networks and mobile devices led to a new paradigm mobile computing in which, user who carries portable device has access to information services through a shared infrastructure regardless of his physical location or movement behavior. The mobile computing environments are suffering from the narrow bandwidth, limited resources on mobile devices, limited battery power, disconnectivity, and user mobility (Safa et al., 2008). Caching frequently accessed data in a mobile client has been proposed to improve the performance of the various mobile computing systems by increasing the availability of data in the presence of disconnectivity, reducing the data retrieval from the original server, relieving the bandwidth consumption, and reducing the latency in data access (Huang et al., 2007). The multiple copies of the same data object distributed among different mobile clients need a consistency between them and with the original source of the data object on the server. Due to the inherited problems in mobile environments, maintaining mobile cache consistency is a complicated process, and the traditional invalidation strategies are not suitable for the mobile environment (Shao, and Shanglu, (2003)).

Mobile cache consistency level is one of the strategies in maintaining the consistency of the mobile cache. In (Cao et al., 2007; Huang et al., 2007) three levels of mobile cache consistency, are exist namely: (i) strict consistency in which the version of the cached data item at the mobile unit is always up to date with the source data at the server, (ii) delta consistency where every read of the cached data is never out of date by more than a specific time with the source data at the server, and (iii) weak consistency level in which a version of the data item exists at the mobile client cache is a old copy of the original data item at the server. Most of the schemes maintaining the cache consistency in the mobile environment support one level of consistency (Chan et al., 2005; Lee, 2006; Madhukar and Alhajj (2006); Yi et al., 2007; Chuang and Chiu (2008); Safa et al., 2008). Applying only one type of consistency level on the cached data items is not suitable all the time, since some applications can allow some degree of weak consistency and some critical cached data items have to be up to date with data in the source. Furthermore; the schemes assume that all updates are done in the server side, which limits the functionality of the caching system.

Multi-level mobile cache consistency protocol (Vora et al., 2002) that works in client-server architecture makes distinction between the strict and weak consistency levels, and when to apply each one depends on the mobile client interest. The client determines the consistency requirements on the cached data items and sends them to server. One of the important features of this protocol is that the mobile client is allowed to update its cached data. However, the consistency requirements determined by the client and upon its interest is not a good idea since in real systems the consistency requirements depend on the mobile application system, also it incur overhead to the clients and network consuming (uplink). Moreover, the consistency requirements on the same data item are differ from one client to another which is represent an overhead to the server in maintaining the consistency of the cached data item.

This paper, based on Multi-Level Mobile Cache Consistency Protocol we design a framework of a new multi-level scheme called Application Based Multi-level Mobile Cache Consistency Method (ABMMCCM) that preserves the advantages of Multi-Level Mobile Cache Consistency Protocol and enhances its drawbacks. The basic idea in the design is that: the consistency requirements are designed at the server based on the application requirements, and each data item has a single consistency requirements entry. The proposed framework is aimed to support different levels of mobile cache consistency, increases the mobile cache functionality, reducing the wireless network consumption, and reduces the overhead from the mobile client and the base server. Rest of the paper is organized as follows. Section 2 presents an overview of related works. In Section 3, we describe in more details the Multi-Level Mobile Cache Consistency Protocol. Section 4 presents our proposed framework and compares it with the Multi-Level Mobile Cache Consistency Protocol. Conclusion is offered in Section 5.

2 RELATED WORK

In the literature several schemes have been proposed to maintain the mobile cache consistency. These techniques are based on broadcasting of Invalidation Reports (IRs)/Update Invalidation Reports (UIRs) issued from the server to the mobile clients synchronously or asynchronously to validate their caches. Based on the underlying methodologies these schemes can be classified into two approaches namely: stateless approach (Barbara and Imielinski (1994)**;** Jing et al., 1997**;** Chun-Hung et al., 2000; Yuen et al., 2000; Hou et al., 2001;Cao, (2002); Shao, and Shanglu, (2003);Yi et al., 2004; Chan et al., 2005; Lee, 2006; Yi et al., 2007; Chuang and Chiu (2008); Safa et al., 2008) and stateful approach (Kahol, and Khurana (2001); Madhukar and Alhajj (2006)).

In the stateless approach the server broadcasts the updates over the network, and the client is responsible to validate its cache. In general this approach wastes the wireless network resources, increases the overhead on the mobile client with unnecessary processing load, and clients are unable to completely disconnect. However, it is suitable to be applied in a large group of users shared the same data item, without any increasing to the server resources**.** In contrast to this, the stateful approach in general saves the network bandwidth, supports disconnection of mobile clients. Also it is suitable for supporting users in a small group, but represents overhead to the base server when the number of mobile clients increases. Some schemes are mixed of the two schemes i.e. support some features from stateful-based approach and features from stateless-based; these schemes are classified under a hybrid approach (An et al., 2004; wang et al., 2004).

All the above mentioned schemes stateless/ stateful or hybrid support one level of cache consistency, i.e. either strict or weak. Furthermore they decrease the benefits of caching data by restricting the use of the cached data for read operations only. In the literature (Vora et al., 2002) proposed a stateful multi-level protocol in which the consistency requirements depends on the mobile client interest, and the mobile client is able to use its cached data for update purposes.

3 MULTILEVEL MOBILE CACHE CONSISTENCY PROTOCOL

In this section we discuss in more details Multi-Level Mobile Cache Consistency Protocol. The discussion includes the consistency requirements propagation, how the protocol, controls the access to the data items and maintains the consistency.

3.1 Consistency Propagation

Multi-Level Mobile Cache Consistency Protocol proposes a notification protocol to propagate and maintain the consistency as determined by the mobile client. The notification protocol consists of two types of messages: Notification Request (NR) and Update Notification (UN). NR message contains the consistency requirements on the cached data item issued from the mobile client to the base server after it cached a data item and granted a suitable lock on it. The NR message has the following form:

$(X: T, \{a1, a2, \ldots, an\}, P \wedge Q, £T)$

Where T is a type, X is an object of T, $\{a1, a2, \ldots, an\}$ are attributes of X, P and Q are predicate conditions (when violated the updated data item is propagated to the mobile client), and $£T$ is a time allowed to the server before broadcasting the updated data item to the mobile client. The second type of message is the Update Notification message, which is issued from the server to the affected mobile clients (depend on its consistency requirements) when receiving updates on a cached data item. The UN message is of the form:

$(X: T, \{a1, a2, \ldots, an\}, \{c1, c2, \ldots, cn\}, TS)$

Where T is a type, X is an object of type T, $\{a1, a2, \ldots, an\}$ is a set of updated attributes, $c1, c2, \ldots, cn$ are the new values, and TS is the time stamp of receiving the updates at the server.

3.2 Data Access Control

Accessing the data items in multi-level mobile cache consistency protocol is based on time lock. The lock type, and lock period are used by the server to allow a concurrent access to the data item in a consistent manner. The mobile client is forced to release lock on the cached data item after a reassigned time. Four types of time locks are proposed: (i) Strict Read Lock (SRL); many SRLs can be exist on the same data item; the mobile client is guaranteed that no other clients can modify the cached data, (ii) Strict Write Lock (SWL); one client can hold SWL at a time; the client can work offline for the period of lock; if the client upgrades SRL to SWL the base server sends messages to other mobile clients to revoke the SRL from them, (iii) Permissive Read Lock (PRL); this type of lock can be shared between multiple mobile clients to get concurrent read access to the cached item, and (iv) Ownership Server Lock (OSL); one client can hold OSL lock on the data item , with the existing of many PRLs locks. The a client holding an OSL on an item is forced to send the updates on that item immediately to the server, then the server broadcasts the updates to the mobile clients holding PRL on that item. PRL can be upgraded to SRL, SWL, or OSL during the lock period.

3.3 Maintaining Consistency

The mobile client can impose any level of consistency on the cached data item, starting from strict consistency level to weak levels. In the Strict Updates-Strict Notify (SUSN) level, the client desires its updates on the cached data to be final and receives notification message regarding any changes on the cached data item. In the Weak Updates–Strict Notify (WUSN), the client immediately notified about the updates on a cached data item when the predicate conditions on it are violated. In the Strict Update-Weak Notify (SUWN), the client would like to know all the updates on the cached data item, but in some delay depending on the allowed delay time on the Notification Request (NR). While in the Weak Updates–Weak Notify (WUWN), the client is interested to be notified in specific updates on the cached data item and allows the server a delay time before propagating the updates to it.

4 THE FRAMEWORK OF ABMMCCM

In this section we present the framework of our proposed scheme Application Based Multi-level Mobile Cache Consistency Method (ABMMCCM) that enhances the Multi-Level Mobile Cache Consistency Protocol presented by (Vora et al., 2002). In ABMMCCM, the consistency requirements on the cached data items are depend on the mobile application system requirements not on the mobile client user's interest. The developer of the mobile application system is responsible to specify the consistency requirements of the different data items and locate them on the server (consistency table); this will liberate the mobile client from the overhead of specifying the consistency requirements on its cached data items, and reduce the consuming of the wireless bandwidth channel especially in the uplink requests. Furthermore; each data item has one entity for its consistency requirements in the server, which is helps in reducing the overhead of maintaining many consistency requirement entities on the same data item supplied by different mobile clients, as in Multi-Level Mobile Cache Consistency Protocol. When the server receives an update on a data item, it verifies the update with the data item entity in the consistency table, then decides whether to propagate the updates to the affected clients or not, which saves the network bandwidth by reducing the messages exchange.

4.1 ABMMCCM System Model

The main components of ABMMCCM system model are the application developer, the base server, and the mobile client.

4.1.1 Mobile Application Developer

The mobile application system developer designs the consistency requirements of the data items depending to the nature of the application. The consistency requirement is stored on the server side as a consistency table. Each data item has a single entry in the table with the following form:

(*Did, P,* £*T*)

Where Did is the data item identifier, P is the predicate condition *(*when violated the updated data item is propagated to the mobile client), and £*T* is a time allowed to the server before broadcasting the updated data item to the mobile client.

4.1.2 Base Server Working Model

The role of the base server in ABMMCCM is to: (i) control the access to the data items and (ii) maintains the consistency and propagates updates to the mobile clients. As in Multi-Level Mobile Cache Consistency Protocol three tables are used by the server to perform its mission: *lock table* keeps the information about the current lock on the data items; *cache table* records the replicas that are cached by different clients; and *consistency table* keeps consistency requirement of the different data items. To control access to the data items, the server uses lock time subsystem, similar to the original Multi-level Mobile cache consistency protocol.
When the request of the mobile client to cache a data item is accepted by the server, the server sends a cache message in the form:

(*Did*, *Dval*, *LT*, *LP*, *TS*, *P* ^ *Q*, £*T*)

where *Did* is the cached data item identifier, *Dval* is the cached data item value, *LT* is the lock type grant (Read/Write), *LP* is the lock period grant, *TS* is the time mobile client caches the data item, *P* ^ *Q* are predicates conditions when violated the updates propagated to the affected mobile clients, and £*T* is a delay time allowed to the server before broadcasting the updated data item to the mobile client. If there is a conflict with the current lock, the server sends a reject message to the mobile client.

The propagation of updates to the mobile clients is accomplished in the following processes. When the server receives updates from a mobile client that holds a valid lock type on the updated data item, the server checks the consistency requirements entity of the updated data item in the consistency table. If the predicate conditions are violated or the allowed time is met, the server propagates the updated data item in the form of update notification message: Update notification (*Clientid*, *Did*, *Dvalold*, *Dvalnew*, *TS*) to the mobile clients have a copy of the modified data item. If any mobile client is in a disconnection state when the updates are propagated, an acknowledge message is not delivered to the server. The server queues the undelivered messages and immediately sends them to the mobile client when reconnect.

The base server revokes a lock given to a mobile client on a cached data item, if the mobile client upgrades its read lock on the data item to write lock within its valid lock period and the consistency requirement on that data is strict. The server sends a revoke message to the other mobile clients that hold a read lock on that item

4.1.3 The Mobile Client System Model

When the requested data is not available locally in a mobile client cache (cache misses), then the mobile client sends a request message to the base server to cache the item. The request message is in the form:

(*Did*, *Lt*, *LP*)

where *Did* is the identifier of the data item to be cached, *Lt* is the lock type required on the data item, and *LP* is the lock period required to hold a lock on the data item. If the request is accepted, the mobile client will receives a cache message from the base server, while in the case of rejection the mobile client will receives a rejection message and later sends a new request. A mobile client that is holds a write lock on a cached data item sends the update on that data item to the server during its valid lock period. The mobile client that is affected by that update, it will receive update notification message from the base server. In the case of weak consistency, the server may gather updates in one message that saves the bandwidth utilization. The mobile client sends acknowledgment message to the server and then validates its cache by committing or rolling back the conflicting operations with the given updates. To validate its cache, the mobile client first compares the time stamp when the updates on the cached data items received by the server to the time stamp of uncommitted operations on the cached data. If the time stamps of the uncommitted operations greater than the time stamps of the updates received on the server then these operations are rollback, otherwise the operations are committed. If the mobile is disconnected when the server sends the update notification message, after the client reconnect it receives all the missed Update Notification messages from the base server.

4.2 Flowchart of the System Model

The flowcharts of the processes in ABMMCCM system model are presented in Figures 1, 2 and 3.

4.2.1 Caching and Locking a Data Item

As we mentioned above the developer of the application designs the consistency requirements of the data items and stored them with the database in the database server. The base server caches these requirements by using any caching algorithm used in a wired network which is out of the scope of this paper. Here, we assume that the mobile client is connected to the base server to start a session. The flowchart of this process is illustrated in Figure 1.

4.2.2 Maintaining Updates

Assume that the mobile client holds a write lock type on a given cached data item in a valid lock period. The flowchart of the update process and its propagation to the other mobile clients holding a cached copy of the updated data item is illustrated in Figure 2.

4.2.3 Revoking and Releasing of Locks

Revoking the read lock from mobile client on its cached data item (with strict consistency requirements), is happen when one of the mobile clients with a read lock on that item would like to upgrade its lock to write lock or a new write lock requested by a client on that item. The server grants the write lock if no conflicting and revokes the locks from other clients hold a valid read lock on that data item.

The releasing of locks can be done voluntary by the mobile client holds a valid lock on the data item, or when the lock period granted to the mobile client on a data item expired. The detail of revoking and releasing locks are illustrated in Figure 3.

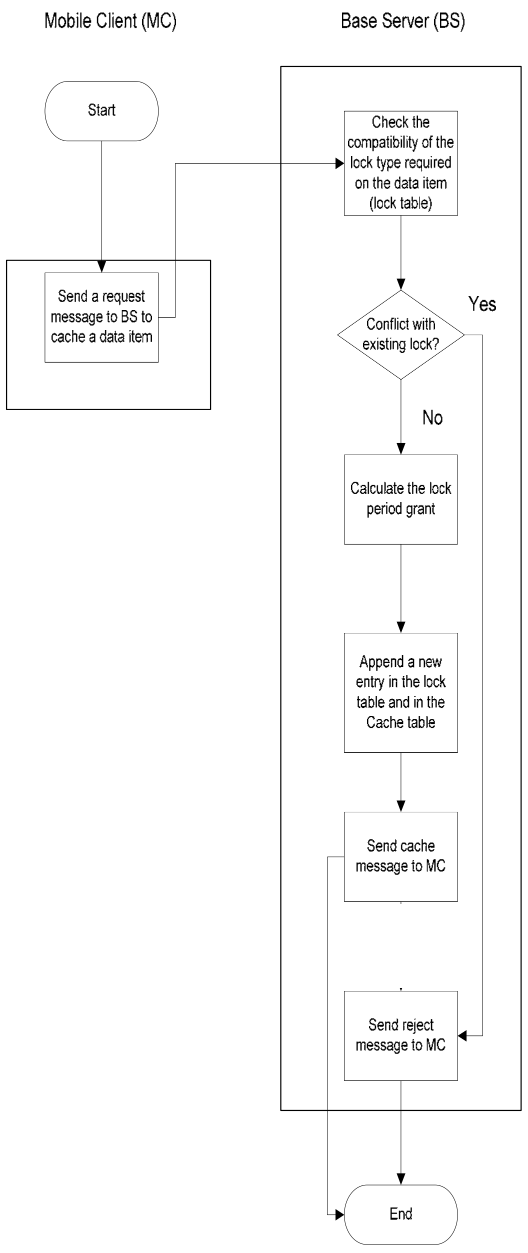
Mobile Client (MC)
Base Server (BS)
Start
Send a request message to BS to cache a data item
Check the compatibility of the lock type required on the data item (lock table)
Conflict with existing lock?
Yes
No
Calculate the lock period grant
Append a new entry in the lock table and in the Cache table
Send cache message to MC
Send reject message to MC
End

Figure 1. Caching and Locking Processes

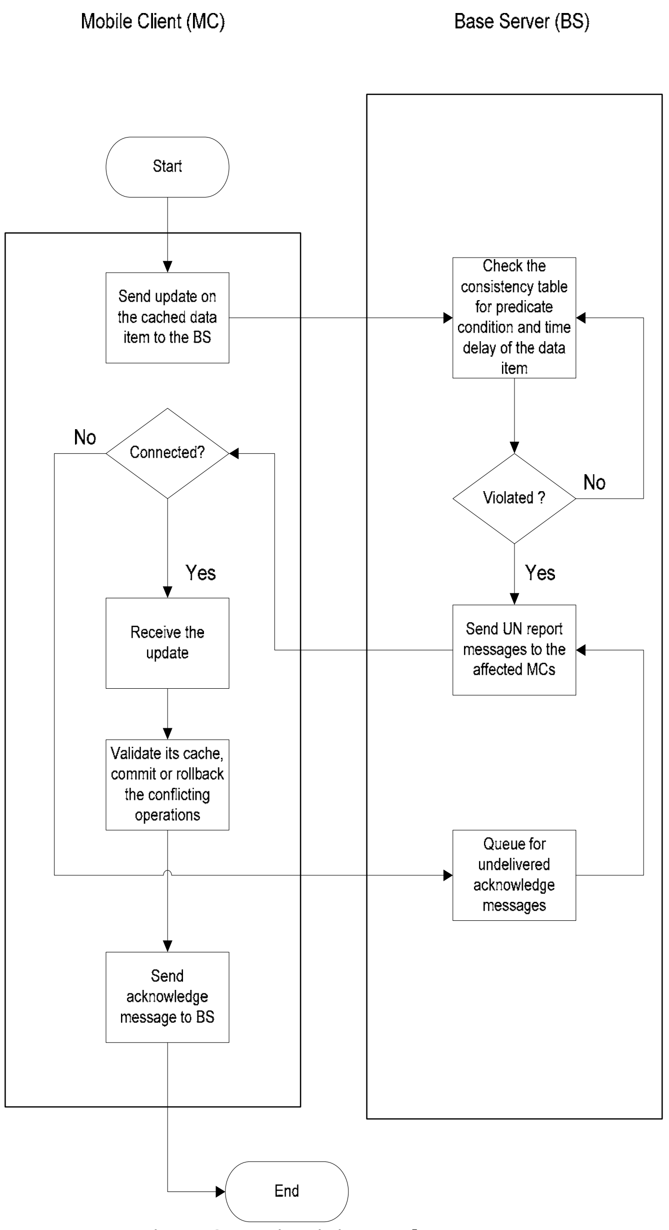
Mobile Client (MC)
Base Server (BS)
Start
Send update on the cached data item to the BS
Check the consistency table for predicate condition and time delay of the data item
No
Connected?
Violated ?
No
Yes
Yes
Receive the update
Send UN report messages to the affected MCs
Validate its cache, commit or rollback the conflicting operations
Queue for undelivered acknowledge messages
Send acknowledge message to BS
End

Figure 2. Maintaining Updates

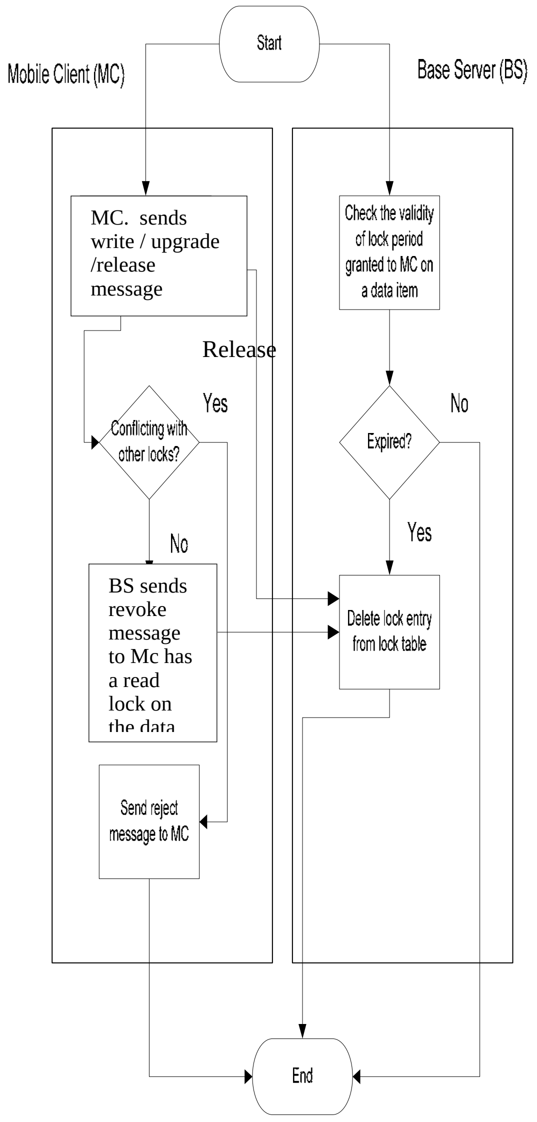
Start
Mobile Client (MC)
Base Server (BS)
MC. sends write / upgrade /release message
Check the validity of lock period granted to MC on a data item
Release
Yes
No
Conflicting with other locks?
Expired?
No
Yes
BS sends revoke message to Mc has a read lock on the data
Delete lock entry from lock table
Send reject message to MC
End

Figure 3. Revoking and Releasing Locks

4.2.4 Messages Exchange

For each data item *Did*, Table 1 illustrates the possible messages that are exchanged between a base server and a mobile client (*Clientid*) in the different states. If we compare ABMMCCM protocol with the multi-level mobile cache consistency protocol, with regards to the number of messages transfer in the process of caching data item, from Figure 4, it is clear that ABMMCCM reduces the number of messages transfer between the mobile client and the base server. This represents one of the main features of ABMMCCM to reduce the overhead from the mobile client and base server.

Table 1: Messages Structure in ABMMCCM

Name	**Sender**	**Receiver**	**Comment**
Request (*Clientid, Did, LT, LP*)	MC	BS	Request to cache a data item *Did* with lock type *LT* and for a lock period *LP*
Cache (*Clientid, Did, Dval, LT, LP, TS*)	BS	MC	Grant a *Clientid* a lock *LP* on *Did* at the time *TS* to cache
Reject (*Did, Clientid*)	BS	MC	*Did*
Update (*Clientid, Did, Dvalnew, Tu*)	MC	BS	*Clientid* sends update *Dvalnew* on *Did* to BS at time *Tu*
Update-notify (*Clientid, Did, Dvalold, Dvalnew, TS*)	BS	MC	Send the updates *Dvalnew* to *Clientid* on *Did* with the value *Dvalold* at time stamp *TS*
Rec(*Did, Clientid,Tr*)	MC	BS	Acknowledge message that *Clientid* has received the updates from BS at time *Tr*
Release (*Clientid, Did*)	MC	BS	*Clientid* releases its lock on item *Did*
Revoke (*Clientid, Did*)	BS	MC	Revoke the lock from *Clientid* on *Did*
Upgrade (*Clientid, Did, LT, LP*)	BS	MC	*Clientid* upgrades its lock on *Did*

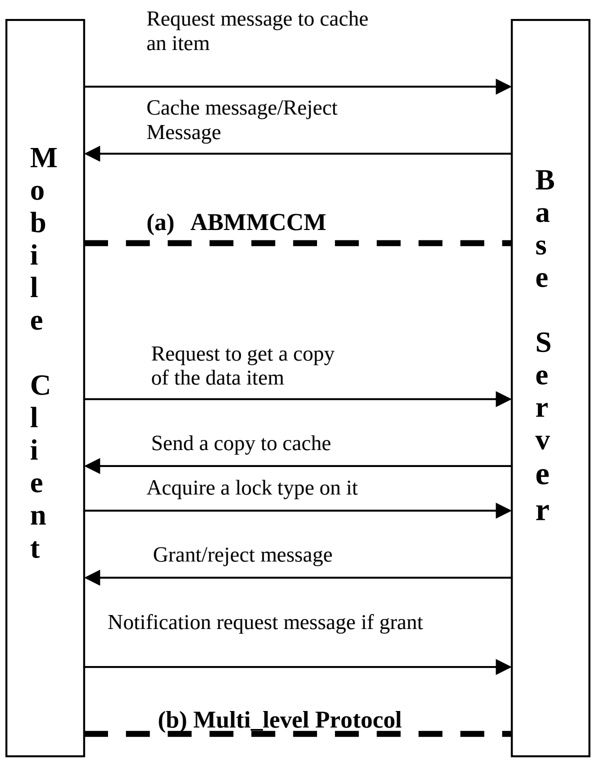
Mobile Client
Base Server
Request message to cache an item
Cache message/Reject Message
(a) ABMMCCM
Request to get a copy of the data item
Send a copy to cache
Acquire a lock type on it
Grant/reject message
Notification request message if grant
(b) Multi_level Protocol

Figure 4: Caching Process - Messages Exchanges
(a) ABMMCCM
(b) Multi-level mobile cache consistency protocol

5 CONCLUSION

In this paper we proposed a framework of a new stateful scheme called Application Based Multi-level Mobile Cache Consistency Method (ABMMCCM) for maintaining cache consistency in the client/server mobile environment. The proposed method is supports multiple levels of consistency depends on the application system requirements, and the mobile client is able to issue update transactions. In the literature most of the schemes support only one level of consistency, and the cached data used for reading purpose only. ABMMCCM is based on Multi-level Mobile Cache Consistency Protocol presented by (Vora et al., 2002), and preserves its advantages and enhances the drawbacks. When comparing the framework of our method with the Multi-Level Mobile Cache Consistency Protocol in terms of number of messages exchanges between the mobile client and the base server, it is clear that the number of messages transfers between the base server and the mobile client when using ABMMCCM is lower than in Multi-Level Cache Consistency Protocol which helps in better utilizing the wireless network, and reduces the overhead from the mobile client and the base server. More tests to evaluate the performance of ABMMCCM are to be considered in the future research.

REFERENCES

[1]. An, K. Jun, B. Cha, J. Hong, B. 2004. A Log-Based Cache Consistency Control of Spatial Databases in Mobile Computing Environments. DASFAA 2004. 2973/2004: 241-248. ISBN: 978-3-540-21047-4. DOI: 10.1007/b95600

[2]. Barbara, D. and T. Imielinski 1995. Sleepers and Workaholics: Caching Strategies for Mobile Environments.The VLDB Journal, 4: 567-602. ISSN: 1066-8888 (print) 0949-877X (Online). DOI: 10.1007/BF01354876.

[3]. Cao,G. 2002. On Improving the Performance of Cache Invalidation in Mobile Environments. Mobile Networks and Applications. *7*: 291–303. DOI: 10.1023/A:1015463328335. URL: http://www.cse.psu.edu/~gcao/mcn/paper/gcao/MONET02.pdf

[4]. Cao, J. Zhang,Y. Xie, L. and Cao, G. April 2007. Data Consistency for Cooperative Caching in Mobile Environments. *IEEE Computer Society*. 40: 60-66. ISSN: 0018-9162. DOI: 10.1109/MC.2007.123.

[5]. Chan, E. Woo, L. and Yuen, J. 2005. IAVI-UIR: An Efficient Caching Scheme for Mobile Computing Systems. 27th. International Conference Information Technology Interfaces ITI, Cavtat, Croatia. 4-51. ISBN: 953-7138-02-X. DOI: 10.1109/ITI.2005.1491095.

[6]. Chuang, P and Y. Chiu, Y. 2008**.** Constructing Efficient Cache Invalidation Schemes in Mobile Environments. Third International IEEE Conference on Signal-Image Technologies and Internet-Based System (SITIS 2007). ISBN: 978-0-7695-3122-9. DOI: 10.1109/SITIS.2007.133.

[7]. Hou, W. Su, M. H. Zhang, and Wang, H. 2001. An Optimal Construction of Invalidation Reports for Mobile Databases," *ClKM'01*, Atlanta, USA. 458 – 465. ISBN: 1-58113-436-3.

[8]. Huang,Y., Cao, J., Wang, Z., Jin, B. Feng,Y., 2007. Achieving Flexible Cache Consistency for Pervasive Internet Access. Proceedings of the 5th Annual IEEE International Conference Pervasive Computing and Communications (PerCom'07). ISBN: 0-7695-2787-6. DOI: 10.1109/PERCOM.2007.6.URL: http://cs.nju.edu.cn/yuhuang/huangyufiles/paper2006/PerCom07.pdf

[9]. Jing, J. Elmagarmid, A. Helal, A. and Alonso, R. 1997 Bit-Sequences: An Adaptive Cache Invalidation Method in Mobile Client/Server Environments. ACM/Baltzer *Mobile* Networks and Applications. 2: 115–127. ISSN: 383-469X (Print) 1572-8153 (Online). DOI: 10.1023/A:1013616213333.

[10]. Kahol, A. Khurana, S. Gupta, S. Srimani, P. July 2001. A Strategy to Manage Cache Consistency in a Disconnected Distributed Environment. IEEE Transaction on Parallel and Distributed Systems. 12.7: 686-700. DOI: 10.1109/71.940744. URL: http://people.cs.vt.edu/~irchen/6204/pdf/KAH01-cache-proxy.pdf

[11]. Lee, S. 2006. System and Method for Maintaining Cache Consistency in a Wireless Communication System. US Patent 7136968, Issued on November 14, 2006

[12].
Madhukar, A. Alhajj, R. 2006. An Adaptive Energy Efficient Cache Invalidation Scheme for Mobile Databases. Proceedings of the 2006 ACM symposium on Applied computing SAC 2006, Dijon France. 1122 – 1126. ISBN: 1-59593-108-2.

[13]. Safa, H. Artail, H. Nahhas, M. 2008. Enhancing Cache Invalidation in Mobile Environments. The international conference on mobile technology ,Applications and systems 2008 (mobility conference) . llan,Taiwan . DOI: ACM 978-1-60558-089-0.

[14]. Shao, X. and Shanglu, Y. 2003. Maintain Cache Consistency of Mobile Database using Dynamical Periodical Broadcast Strategy. Proceedings of the Second International

[15]. Conference on Machine Learning and Cybernetics,Xi'an. 4: 2389- 2393. ISBN: 0-7803-7865-2. DOI: 10.1109/ICMLC.2003.1259910. URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=01259910.

[16]. Vora, A. Tari, Z. Bertok, P. Lai, K. 2002. A Mobile Cache Consistency Protocol Using Shareable Read/Write Time Locks. Proceedings

[17]. of the Ninth International Conference on Parallel and distributed Systems (ICPADS'02). Taipei, Taiwan. IEEE. ISBN: 0-7695-1760-9. DOI: 10.1109/ICPADS.2002.1183413. URL:http://www.abhinavvora.com/research/icpads2002.pdf

[18]. Wang, Z. Das, Z. Che, H. Kumar, M. November 2004. A Scalable Asynchronous Cache Consistency Scheme (SACCS) for Mobile Environments. IEEE Transactions on Parallel and Distributed Systems., 15,11. ISSN: 1045-9219. DOI: 10.1109/TPDS.2004.60.

[19]. Yi, S. Jung, S. and Suh, J. 2007. Better Mobile Client's Cache Reusability and Data Access Time in a Wireless Broadcast Environment. ScienceDirect, Data & Knowledge Engineering. 63(2): 293–314**.** ISSN: 0169-023X.

[20]. Yi, S. Shin, H. Jung, S**.** 2004**.** Enhanced Cost Effective Cache Invalidation for Mobile Clients in Stateless Server. Springer Berlin / Heidelberg. Lecture Notes in Computer Science 3207/2004: 387-397. ISBN: 978-3-540-22906-3. DOI: 10.1007/b100039.

[21]. Yuen, J. Chan, E. Lam, K. and Leung, H. December 2000. Cache Invalidation Scheme for Mobile Computing Systems with Real-time Data. SIGMOD Record. 29(4): 34-39. ISSN: 0163-5808. URL : http://www.cs.cityu.edu.hk/~rtmm/iavi_sigmod.pdf