In this paper, we propose a cache consistency maintenance scheme called Thread Based Cache Consistency Model (TBCCM), for mobile environment. It lies on the following four key features. (i) Use of log at both server and mobile user’s cache maintains data consistency. (ii) Use of Thread Agent (TA) at both client & server. (iii) Use of log database at server. (iv) Use of Migration Thread Agent at server. Since log is maintained in both server and client it is easy for server as well as client to make their data object reliable. The data object is reliable due to the usage of log comparison technique. Thread Agents at server and client make an internal request response operation(s) to access or update data object. This makes the medium to be liable. Log database at server maintains log of both client and server. Whenever there is need of update or access, log of client is checked along with log database at server. If there is need of access or update it precedes further process such as Security Level checking, synchronization Process etc., Migration Agent at server is activated upon updation of data object at server by certain client. It also seeks for other client(s) who are in need of the same data object that are updated currently.

Keywords: Mobile Database, Wireless Networks Database Cache, Threads Agent, Low Bandwidth Networks.

1. INTRODUCTION

In mobile client/server computing environments, mobile clients make access to their server to get interested data and then are disconnected because of high cost of wireless communication or low bandwidth or low power consumption. Mobile clients usually keep their own local copies in order to reduce the overhead of communication with the server. The updates of the server database sometimes are subject to leading to invalidation of the cached map in mobile clients.[1],[2]. However it is not efficient to resend the entirely copied map from the server to mobile clients for solving invalidation. This paper proposes a Thread based update propagation method to propagate the server’s update into its corresponding mobile clients by sending only update logs. Our approach overrides Invalidation Report Scheme in previous algorithm(s). Instead of it we use various Agents in server and client. Upon request/response an internal report is transferred between Thread Agent (TA) at client and Thread Agent (TA) at server. Migration Thread keeps track of frequently accessed data objects & users. Log based approach solves inconsistent data object. Because whenever energy level or bandwidth is down, a log is maintained in client cache. After certain period when client fetches adequate bandwidth or energy, Thread Agent allows to run a new Thread along with last accessed data log. The log maintained in the server database is compared with recent client access log. If there is an Updation at server, synchronization module begins to function in order to update new data object at Mobile client. Upon Updation to client cache a log is maintained in both server and client. Our process deals with disconnected approaches. Agent provides optimization on the data object and services between client and server.

2. RELATED WORK

An evolution of data consistency becomes most required because of two major reasons. One is the limited bandwidth of wireless channels and the other one is short battery span. To conquer these two major problems many have approached their techniques. But we are in situation to choose upon which would be better in sense of performance, reliability and cost. In order to make data reliable earlier algorithms have adapted various schemes. Some of them are Greedy dual-LU, Bit Sequences (BS), IR-Based Cache Invalidation Modal [2], [3]. All the above specified approaches make an invalidation report for inconsistent data object. Based on the report the client has to update their data object in their cache. Here even though the process is far better, it spends time in generating Invalidation report and checking the report generated.

3. THREAD BASED CACHE CONSISTENCY MODEL (TBCCM)

In this paper, we propose a cache consistency maintenance scheme called Thread Based Cache Consistency Model
(TBCCM), for mobile environment. Strictly speaking, TBCCM is a hybrid of stateless and stateful approaches that it maintains log state information. On the other hand, unlike existing synchronous stateless approaches, TBCCM does not require to produce an Invalidation Report (IR). Since Thread Agent (TA) at client keeps track with Thread Agent (TA) at server. Each make a mutual understanding between them, upon it data object is made to be valid in both client and server. Moreover maintaining log at server and client makes data object to be reliable. The following subsections describe the proposed algorithm in detail.

3.1 System Architecture

![Thread Based Approach System Architecture](image)

**Key Terms**
- TL = Thread Listener
- SM = Security Module
- SynchM = Synchronization Module
- TM = Thread Monitor or Thread Manager
- MTA = Migrating Thread Agent

**Table 1**

<table>
<thead>
<tr>
<th>Components</th>
<th>Client</th>
<th>Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thread Listener</td>
<td>To listen for response, to synchronize and prompt with server.</td>
<td>To listen for requests, to synchronize and prompt users for login authentication.</td>
</tr>
<tr>
<td>Security Module</td>
<td>To authenticate user identity and to control user access based on granted privileges.</td>
<td>To verify userID and to keep track of who can which database tables at the server for synchronization.</td>
</tr>
<tr>
<td>Synch Module</td>
<td>To work together with the synchronization fixed server agent to propagate changes to and from the client table cache.</td>
<td>Works together with the synchronization module at the mobile client’s agent to propagate changes to and from the server.</td>
</tr>
<tr>
<td>Thread Module</td>
<td>This module keeps track upon Thread creation, flow and maintenance.</td>
<td>This module keeps track upon Thread creation, flow and maintenance.</td>
</tr>
<tr>
<td>DBListener</td>
<td>To listen for database connection requests from server.</td>
<td>To listen for database connection requests from client.</td>
</tr>
<tr>
<td>Table Cache</td>
<td>To store Data Base objects</td>
<td>To store Data Base objects</td>
</tr>
</tbody>
</table>
3.1.1 Thread Agent at Server
The Thread Agent (TA) at server maintains as well as keeps on monitoring the frequent broadcast values and frequent client cache access as shown in Figure 1. Whenever a value is read / written to server, it has to be updated and to be broadcasted. During this process Thread Agent (TA) maintains a Thread log which holds information about broadcast values, information of mobile client who needs updated value.

3.1.2 Migration Thread Agent
Whenever write operation is performed by mobile client to the server, a special Thread called “Migration Thread Agent” will be activated upon write operation by client to the server, which will be keep monitoring which client is performing the write operation to the server. It maintains a write log of both client & server. It also seeks whether the updated data object is required by any other mobile clients. If so it finds certain client(s) and it begins to synchronize in order to keep data object at cache as more reliable.

3.1.3 Thread Agent at Client
The Thread Agent (TA) at client maintains as well as keeps on monitoring the frequent broadcast values and Thread Agent (TA) at server. Whenever a value is read or written to server, it will be updated to server. Now the updated value will be broadcasted to the requested mobile client. The Thread Agent (TA) at client maintains a Thread log which holds information such as broadcast values, broadcast time, threadIDc, threadIDu, logIDc, and logIDs.

3.1.4 Data Sequence
For every data object with unique identifier x, the data sequence for Mobile server and mobile user’s cache are as follows:

\[(d_i, t_i, tid, u_i, l_{id})\] where \(d_i\) is the data object, \(t_i\) is thread id, \(u_i\) is update time & \(l_{id}\) log id

\[(d_j, ts_j, ts_{id}, us_j, ts_{id})\] where \(d_j\) is the data object, \(t_j\) is thread id, \(u_j\) is update time & \(l_{id}\) log id

<table>
<thead>
<tr>
<th>Name</th>
<th>Sender</th>
<th>Receiver</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Update</td>
<td>MSS</td>
<td>MU</td>
<td>Indicating (d_i) has been updated to (d_j) at time (t_j)</td>
</tr>
<tr>
<td>Vdata</td>
<td>MSS</td>
<td>MU</td>
<td>Broadcast valid data object (d_j) with update time at (u_j)</td>
</tr>
<tr>
<td>Log</td>
<td>MSS</td>
<td>MU</td>
<td>Indicating (d_j) is valid (ts_j = t_s)</td>
</tr>
<tr>
<td>Query</td>
<td>MSS</td>
<td>MU</td>
<td>Query for data object (d_x)</td>
</tr>
</tbody>
</table>

Table 2: Communication Message in TBCCM

4. ALGORITHM
Algorithm presented below in figure 2 and 3 shows typical approach of managing data consistency in mobile computing. We present two procedures MT New Data () and MT Update Data () at server and each MU continuously executes the MT New Data () or MT Update Data (). The pseudocodes MT New Data () and MT Update Data () and MU () are shown below.

---

Algorithm for Server

MT New Data ()
Loop until Time 1
Waits for Client Request
Fetch Request \(R\) from Client \(C\)
SM () Checks is Client \(C\) is authenticated
IF Authenticated == True
Allow Client \(C\) to access required Data Object \(d\) at Time 1
ELSE
Access Denied for Client \(C\)
End of Loop
MT Update Data ()
Data Object \(d\) to be updated
SM () Checks is Client \(C\) is authenticated
IF Authenticated == True
Allow Client \(C\) to update required Data Object \(d\) at Time 1
ELSE
Access Denied for Client \(C\)
End of Loop
MTA ()
Seeks for N number of Clients \(C\)
IF \(C\) is Found \((C = n)\)
SyncM () Begins operation(s)
ELSE
NO Clients Found for Update.
End MTA ()
ELSE
No Data Object to Update

Algorithm for Client

MU ()
SET Mode = Sleep / Awake; Sleep = 0; Wake = 1
IF Mode == 1
Loop Until Time 1
IF New Data object \(d_x\) to be Read
Send Request \(R\) to Server \(S\)
Upon Receiving \(R\) from Client \(C\),
SM () Checks is Client \(C\) is authenticated
IF Authenticated == True
Allow Client \(C\) to access required Data Object \(d_x\) at Time 1
Update Cache \(C\) at Client \(C\),
ELSE
Block Client \(C\), Until Permission is Granted.
IF object \(d_x\) to be Update
SM () Checks is Client \(C\) is authenticated
IF Authenticated == True
Allow Client \(C\) to update required Data Object \(d_x\) at Time 1
ELSE
Invalid Operation or Connection Broken
IF Mode == 0
Thread Agent \(T\) at Client \(C\), Maintains log
Running Process Goes to SLEEP Mode
UNTIL Mode is SET to 1 (Mode == 1)

---

Figure 2: Algorithm for Server

Figure 3: Algorithm for Client
4.1 Cache Consistency Maintenance

Let us explain how TBCCM maintains consistency between the Server data and Mobile caches. For each cached data object TBCCM uses log to maintain consistency between Server and Mobile client. When a data $d_x$ retrieved by a mobile client log is created to indicate data is valid or not. If and when the Server receives an updated data object $d_x$ it broadcasts and synchronizes with Thread Listener (TL) of client to make cache data object reliable. During this process a log maintained in server is compared with recent log of client, If so there in a need of Updation, it processes to perform update function(s). In mobile environments a Mobile Cache is one of two states. (i) Awake or (ii) Sleep. If a Mobile Client is awake an internal request is shared between Thread Agent at server and Thread Agent at client to ensure that data object is updated.

If there is an Updation the SynchM of server synchronizes with SynchM of client in order to make mobile client cache as valid data object. The data objects of a Mobile Client in the sleep state are unaffected until it wakes up. When a mobile client wakes up a new Thread upon is created which holds last accessed log, this log passes to the server, On receiving upon the log it compares with previous log maintained by it.

5. PERFORMANCE EVALUATION BY SIMULATION

The Experimental performance evaluation of Thread Based Cache Consistency Model (TBCCM) algorithm using NS2 simulation software. The Ns2 is used to simulate the mobile computing concept. The channel capacity of each mobile host has 3 Mbps. The MAC protocol is used 802.11. The Mobile hosts moves in 800×800 m rectangular region. We take of number nodes 25, number cell 5, number of client 5 for each cell. slot duration 2 ms, speed of mobile 5ms. Using thread based cache consistency the average of latency time less compare to the invalidation report algorithm, In invalidation algorithm each updating, the report will send to each mobile host, so the mobile host take more time to process, but using the thread based cache consistency model the mobile host processing time is very less.

6. CONCLUSION

In this paper, we proposed a cache consistency maintenance scheme called Thread Based Cache Consistency Model (TBCCM), for mobile environments. Four key features as stated earlier (1). Use of log at both Server and Mobile Users cache maintains data consistency. (2). Use of Thread Agent (TA) at both client & server (3). Use of log database at server (4). Use of Migration Thread Agent at server makes the data object to be consistent.

TBCCM does not require broadcasting of Invalidati on Report. TBCCM inherits the positive features from both stateful and stateless algorithms. Simulation results show that the proposed algorithm has significantly better performance than earlier approaches.

REFERENCE


