Agent Based Mechanism for maintaining Cache Consistency in Mobile Environment

G.Shanmugarathinam[1] N. Junath[2] Dr.K.ViveKanandan[3]
[1] **Research scholar(external mode) in Bharathiyar University-India**
[2] **Lecturer ,IT Dept, Ibri College of Technology - Sultanate of Oman**
[3] **Professor-Bharathiyar university - India**
Email: metshanmugam@yahoo.co.in, profjunath@gmail.com

Abstract

Caching at the mobile client is prospective technique that can reduce the number of uplink requests, lighten the server load .However, Variable data size, data updates ,limited client resource and frequent client disconnection make cache management a challenge .object caching is often used to improve the performance of mobile application .Consistency approach to maintaining cache consistency with the use of Invalidation (or) update reports. The server periodically broadcasts updates (or) invalidation report to clients .update report reflect the changes in the state database.

A drawback of this method is that invalidation report impose a high processing load on clients .clients have to listen to all report ,even through there may be no changes in the data caching .With the aim of reducing the cache consistency maintenance work .I have proposed agent based mechanism to save wireless bandwidth ,reduce network traffic and reduce the workload in server .Based on the mechanism derived queuing model for the simulation .Moreover Ns2 simulation performance were analyze the result proposed technique over existing system

***Keywords*: mobile database, wireless networks, database cache, threads Agent.

1.Introduction

Mobile computing environments are characterized by slow wireless links and relatively underprivileged hosts with limited battery powers, predisposed to frequent disconnections. Caching data at the Mobile Hosts (MHs) in a wireless network helps alleviate problems associated with slow, limited bandwidth wireless links, by reducing latency and conserving bandwidth. Battery power is conserved by reducing the number of up-link requests. A mobile computing

environment is a distributed system, thus when data at the server changes, the client hosts must be made aware of this fact in order to invalidate their cache, otherwise the host would continue to answer queries with the cached values returning incorrect data.

Recent advances in wireless and mobile networks have led to the exponential growth of mobile applications. Unlike conventional computing, mobile computing has stringent constraints in network resources, such as bandwidth and connectivity. As such, data in mobile applications are often cached at clients to increase performance, data availability and reliability. Most fault-tolerant schemes for wireless sensor networks focus on power failures or crash faults. Little attention has been paid to the data inconsistency failures.

Although a number of studies have been made in this subject, few researchers focused on mobile data access. In this paper, we design a node the master client cache. It is between the server and client. Whenever server data was updated immediately synchronization starts with master client cache and the client. Some of the clients wake up from sleep mode immediately request the master client cache for the updated data and need not request the server. So it reduces the work load in the server database.

2. Related work

2.1 Updated Invalidation Report (UIR)

In this approach[1] the server periodically broadcasts an IR (Invalidation Report) in which the changed data items are predicted. Since IR (Invalidation Report) arrive periodically, client can go to sleep most of time and only wake up when the IR (Invalidation Report) comes. It brings long query latency and low hit ratio.

2.2 Prefetch to Cache Hit Ratio

In most previous IR (Invalidation Report) based scheme, even though many clients cache the same updated data object, all of them have to query the server and fetch the data object from the server separately. This approach may not be suitable to hot (or) dynamic data objects.

This problem[2] is solved by making clients to Prefetch their data object which are needed for future use. Even though this mechanism is good, but insufficient because each time when server broad casts data objects the client has to make a request in order to update its cache.

2.3 Cache Invalidation scheme for mobile database

This approach improve mobile caching by reducing the communication bandwidth for query processing object consistent.[3]

3. Agent Based mechanism

In this paper we proposed new technique Agent based mechanism called log and thread synchronization model for wireless network .In our design does not required to produce an Invalidation report, thread agent maintain a log and thread synchronization in client and server, maintain the cache consistency . The following subsection describe the proposed algorithm in detail

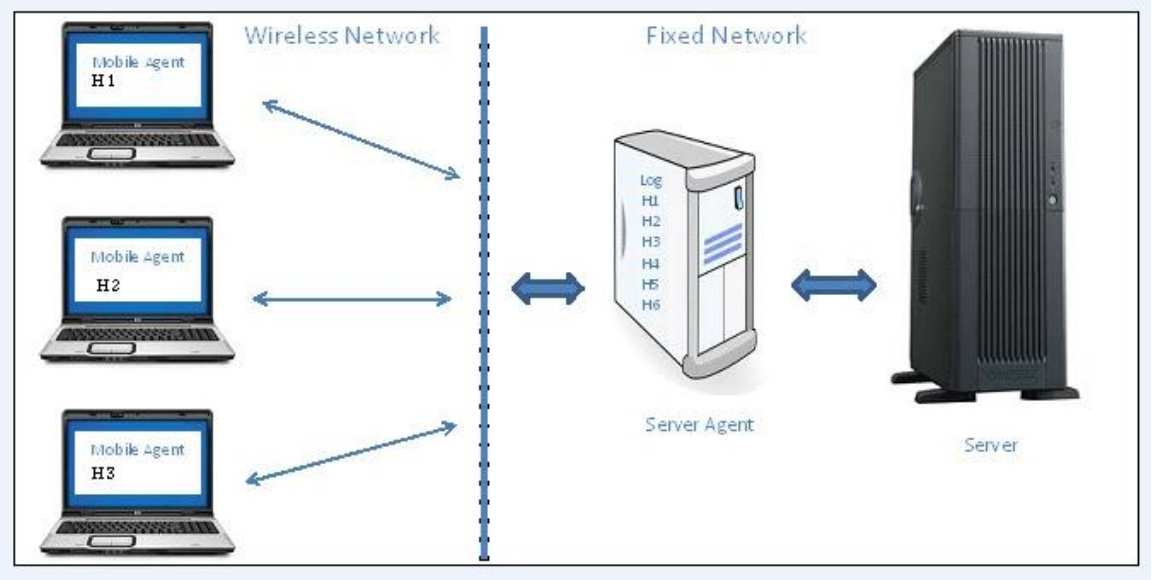


Fig1 : Agent Based mechanism Architecture

3.2 Thread Agent at Server :

The Thread Agent (**TA**$_s$) 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**$_s$) maintains a Thread log which holds information about broadcast values, information of mobile client who needs updated value

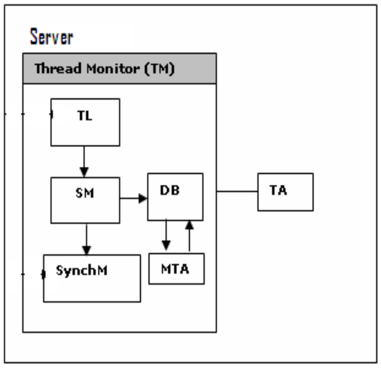


Fig3: server Agent

3.3 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 cache client ,

3.4 Thread Agent at Client

The Thread Agent (**TA**$_C$) at client maintains as well as keeps on monitoring the frequent broadcast values and Thread Agent (**TA**$_s$) at server and Mater cache client. 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.

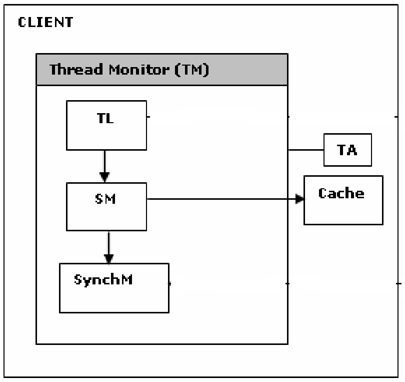


Fig 2: client Agent

The Thread Agent (**TA**c) at client maintains a Thread log which holds information such as broadcast values, broad cast time, threadIDc, threadID$_s$, logIDc, and logIDs

Agent based Cache data Updation

Let us explain how Agent maintains consistency between the Server data and Mobile caches. For each cached data object Agent uses log to maintain consistency between

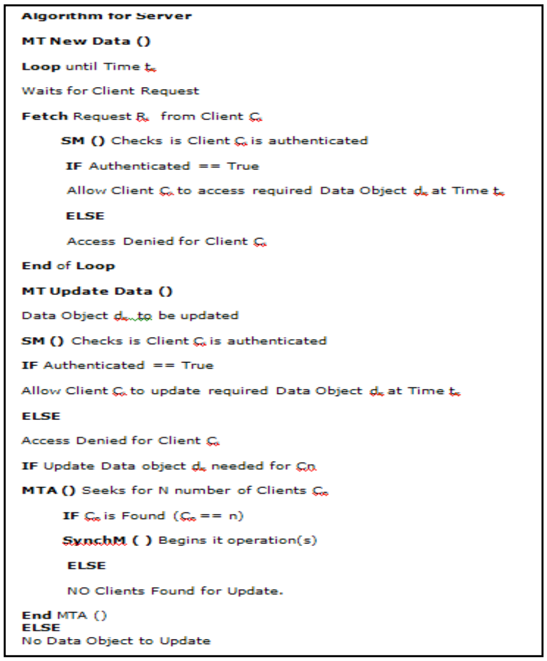
Server and Mobile client. When a data dx retrieved by a mobile client log is created to indicate data is valid or not. If and when the Server receives an updated data object dx it broadcasts and synchronizes with Thread Listener (TL) of client to make cache data object reliable. During this process a 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.

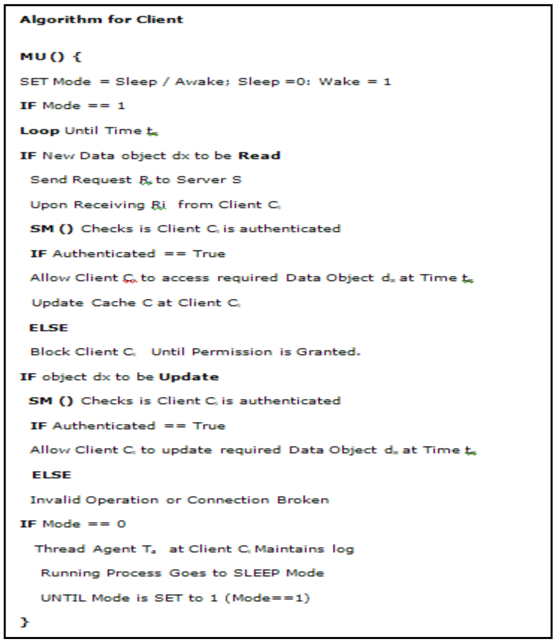
Client Agent analysis the cached date based on Frequently update data (FUD), Non frequently update data (NFUD) for example client side cached the stock prices data are called as FUD are required the update with short time . NFUD example are weather forecast information does not change in short time . Client and server agent communicate together update the cache data

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 Psudocodes MT New Data ( ) and MT Update Data ( ) and MU ( ) are shown below.



```
Algorithm for Server
MT New Data ()
Loop until Time $t_x$
Waits for Client Request
Fetch Request $R_x$  from Client $C_x$
      SM () Checks is Client $C_x$ is authenticated
       IF Authenticated == True
       Allow Client $C_x$ to access required Data Object $d_x$ at Time $t_x$
       ELSE
       Access Denied for Client $C_x$
End of Loop
MT Update Data ()
Data Object $d_x$ to be updated
SM () Checks is Client $C_x$ is authenticated
IF Authenticated == True
Allow Client $C_x$ to update required Data Object $d_x$ at Time $t_x$
ELSE
Access Denied for Client $C_x$
IF Update Data object $d_x$ needed for Cn
MTA () Seeks for N number of Clients $C_n$
      IF $C_n$ is Found ($C_n$ == n)
      SynchM ( ) Begins it 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 $t_x$
IF New Data object dx to be Read
  Send Request $R_x$ to Server S
  Upon Receiving Ri from Client $C_i$
  SM () Checks is Client $C_i$ is authenticated
  IF Authenticated == True
  Allow Client $C_x$ to access required Data Object $d_x$ at Time $t_x$
  Update Cache C at Client $C_i$
  ELSE
  Block Client $C_i$ Until Permission is Granted.
IF object dx to be Update
  SM () Checks is Client $C_i$ is authenticated
  IF Authenticated == True
  Allow Client $C_i$ to update required Data Object $d_x$ at Time $t_x$
  ELSE
  Invalid Operation or Connection Broken
IF Mode == 0
   Thread Agent $T_a$ at Client $C_i$ Maintains log
    Running Process Goes to SLEEP Mode
    UNTIL Mode is SET to 1 (Mode==1)
}
```

5. Server based Queuing model

1. The arrivals follows poison distribution with a mean arrival rate λ
2. The service time has exponential distribution with a mean service rate μ
3. Arrivals are infinite population $\propto$
4. Clients are served on a first in , first out basis (FIFO)
5. There is only single central server

Traffic intensity

$$\rho = \frac{\lambda}{\mu}$$

Expected number of client

$$L_s = \frac{\rho}{1-\rho}$$

Expected number of client in queue

$$L_q = \frac{\rho^2}{1-\rho}$$

Latency

$$Latency = \frac{\lambda}{\mu(\mu-\lambda)}$$

State

$$\lambda < \frac{\mu^2}{1+\mu}$$

(Servicing rate) - (Arrival rate) $> \rho$

$\mu - \lambda > \rho$

5.1Simulation

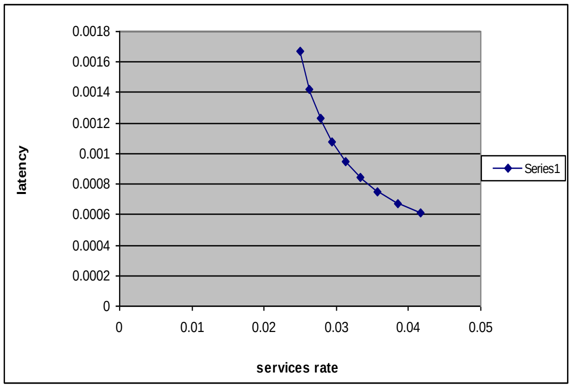


Fig 4 :Service rate with latency

Show that services rate increase

and latency decrease in the graph

5.2 Ns2 simulation

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 700×700 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 Agent 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 Agent based cache consistency model the mobile host processing time is very less

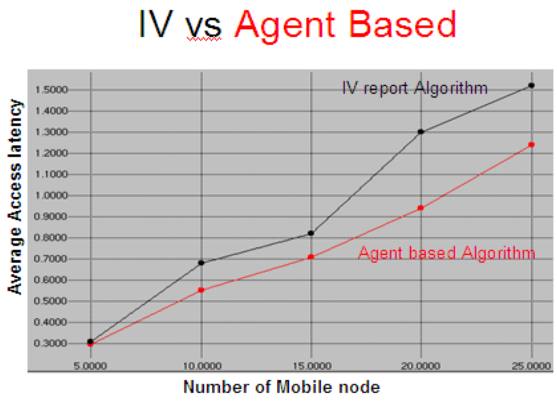


Fig 5: IV algorithm Vs agent Based

Conclusion

In this paper, we proposed agent based mechanism for cache consistency maintenance for mobile environments. 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. Agent does not require broadcasting of Invalidation Report. Client side Agent analysis cache data and update with server agent. Server based queuing model is used for Simulation and show the result that the service rate increase the latency decrease .Implementation used Ns2 for and results shows that proposed algorithm has significantly better performance than earlier approaches .

REFERENCE

[1] "Mark Kai Ho Yeung, Yu-Kwong Kwok, "Wireless Cache Invalidation Schemes with Link Adaptation and Downlink Traffic", IEEE Transaction On Mobile Computing, 4, No 1, January/February 2010.

[2] Huaping Shen, Mohan Kumar, Sajal K.Das, "Energy Efficient Caching and Prefetching with Data Consistency in Mobile Distributed Systems", 0-7695-2132-0/04/2004, IEEE.

[3] Niraj Tolia and Adam Wolbach, "Improving Mobile a Database Access Over Wide Area Networks without Degrading Consistency", ACM, 2006.

[4] C.-Y. Chang and M.-S. Chen, "Exploring Aggregate Effect with Weighted Transcoding Graphs for Efficient Cache Replacement in Transcoding Proxies", Proc. 18th Int'l Conf. Data Eng., Feb. 2002.
[5] Elsen, F. Hartung, U. Horn, M. Kampmann, and L. Peters, "Streaming Technology in 3G Mobile Communication Systems", Computer, 34, No. 9, pp. 46-52, 2001.
[6] S. Hosseini-Khayat, "On Optimal Replacement of Nonuniform Cache Objects", IEEE Trans. Computers, 47, No. 4, pp. 445-457, 2000.
[7] T. Imie linski and B.R. Badrinath, "Wireless Graffiti—Data, Data Everywhere Matters", Proc. 28th Int'l Conf. Very Large Data Bases, 2002.
[8] A. Kahol, S. Khurana, S.K.S. Gupta, and P.K. Srimani, "A Strategy to Manage Cache Consistency in a Distributed Disconnected Wireless Environment", IEEE Trans. Parallel and Distributed System, 12, No. 7, pp. 686-700, 2001.