PETRANET: a Power Efficient Transaction management technique for Real-time mobile Ad-hoc NETwork databases

Le Gruenwald Percy Bernedo Prasanna Padmanabhan
University of Oklahoma, Norman, OK 73019, USA
{ggruenwald, percy_bernedo, prasannap}@ou.edu

1. Introduction

A Mobile Ad-Hoc Network (MANET) is a collection of wireless autonomous mobile nodes with no fixed infrastructure. Since no fixed infrastructure is required, MANET fits well in military operations, emergency disaster rescue, and mobile ad-hoc voting. There are many issues that have to be addressed while designing a technique for managing real-time database transactions in MANET: 1) energy limitations; 2) client and server mobility; 3) real-time constraints imposed on transactions; and 4) frequent disconnection and network partitioning. We have designed PETRANET[1]: a Power-Efficient Transaction management technique for Real-time mobile Ad-hoc NETwork databases that addresses the above specified issues. In this paper, we present a system prototype that we have developed to implement PETRANET for a military database application.

2. PETRANET design and implementation

2.1. Real Time Transaction Types

There are two types of real time transactions in PETRANET: firm and soft. Firm transactions have one deadline and are aborted if they have missed their deadlines. Soft transactions have two deadlines; they continue execution with a lower priority when their first deadlines have expired and are finally aborted when their second deadlines have passed.

2.2. Power consumption and modes of operations

We have considered three power modes of operation of a mobile node. It is important to note that these modes of operation are hardware specific; nevertheless they have some common characteristics: 1) Active mode - Wireless Card can transmit and receive packets; CPU is in a completely functional state; 2) Doze mode - Wireless Card can transmit and receive packets; CPU is working at a lower computational rate; 3) Standby mode - Wireless Card can neither transmit nor receive packets; all the circuits of the CPU except timer and RAM are switched OFF.

2.3. Client Side

The work flow of transactions at the client side is as follows: the user on a client mobile node can set deadline(s) and select a transaction from a set of predefined transactions. If the transaction is firm, the client will switch to active mode to reduce the chance that the transaction will miss its deadline; otherwise it will switch to doze mode. The client then needs to send the transaction to an appropriate server (called the coordinator) for processing. To choose this server, it considers the location, remaining energy and transaction workload of each server [3]. Firm transactions are sent to the nearest server that has the least workload, while soft transactions are forwarded to the server with the highest remaining energy. The goal is to reduce the number of transaction aborts while maintaining a balance of energy consumption distribution among mobile hosts.

2.4. Server Side

2.4.1. Transaction Distribution. When a transaction is submitted to a server (coordinator), its operations are parsed to determine which database items are needed. Then a global schema stored on each server is used to determine whether the operation can be processed locally or not and in which servers reside the needed database items. At this point, if we are dealing with a distributed transaction, we intercept the operation and then we store the results in temporary tables and replace the distributed operation by a local one that points to the just created local table(s).

2.4.2. Power State Switching. The server examines its transaction's queue every time an operation is ready to be

1 This material is based upon work supported by (while serving at) the National Science Foundation (NSF) and the NSF Grant No. IIS-0312746. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF

executed. If all the transactions are soft the server will switch to doze mode to save energy, otherwise if there is at least one firm transaction, the server will switch to active mode (if its remaining energy allows it [2]).

2.4.3. Real Time Scheduler. Currently PETRANET does not modify the operating system scheduler, still it sets different priorities for the incoming transactions [5]. The real time scheduler sets priorities based on the transactions' slack time. The transaction with the smallest slack time will have the highest priority while all the others will have a waiting priority[3]. Each client's connection is served by one thread with a round robin scheduling policy. Each thread can hold only one transaction at a time; this is why setting the thread's priority would affect directly the scheduling of the transaction.

2.4.4. Commit Protocol. We use the MySql XA interface (X OPEN Distributed Transaction Processing (DTP)) which provides a mechanism to put a transaction into a PREPARE state before its actual commit [4]. Currently, we assume immediate constraint check; so we PREPARE for committing the transaction at the end of each of its operations. Therefore at commit time we only need to perform a One Phase Commit(1PC) protocol [6],[7].

2.4.5. Result Submission. After a transaction's coordinator database server gets back the acknowledgments or partial results from the participating servers, it sends the results to the client. Due to disconnections and network partitioning, the results might not always be delivered in the first attempt. The server, instead of re-executing the whole transaction, will use the temporary table(s), where the partial results were stored, to retrieve the results. The temporary table(s) is/are kept as long as the transaction has slack time remaining. The coordinator names these tables using a combination of the transaction ids and the client's IP addresses; so it can recognize when it needs a resubmission. After a successful result delivery, the temporary tables are dismissed.

3. Testing Application

We have worked with The University of Oklahoma's Military Reserve Officer Training Corps (OU Military ROTC) to gather the requirements for a military database application. Its database tables have been created and populated with one million rows which are stored with partial replication in each server.

2 The operating system will not allow to switch to the highest power consuming mode when the remaining energy is less than 5 percent.

3 The priority is so low that the operating system scheduler will not give them quanta

4. Demonstration of the PETRANET

During the demonstration, a set of 10 PDAs and 5 laptops, equipped with a GPS and wireless card, will be provided. The conference participants will be able to interact with the servers through PDAs. They will be able to walk into different rooms and stay connected due to the multi-hop wireless communication; disconnection will occur when a node is out of range of all the other nodes. The clients' front end will allow the conference participants to choose a transaction from a set of predefined transactions, set (manually) a deadline and transaction type (firm or soft) for it unless it has a predefined deadline/type, submit it for execution and wait for the results. The conference participants will observe how the PDAs change their power modes to save energy after submitting the transaction and while waiting for the results.

5. Conclusions and future research

We have successfully implemented and tested PETRANET using the OU military ROTC database. We are currently working on energy-efficient and real-time data caching, commit and concurrency control protocols for MANET databases. We will also work with the Norman Fire Department to gather its application requirements and will implement them in our prototype. We will also investigate additional MANET database system architectures.

References

[1] D. Barbara, Mobile Computing and Databases - A Survey, IEEE Transactions on Knowledge and Data Engineering, pp.108-117, 1999.

[2] [Hong] X. Hong, M. Gerla, G. Pei, and C. Chiang, A group mobility model for ad hoc wireless networks, in Proc. ACM/IEEE MSWiM, 1999.

[3] Chuo Ning Lau, Handling mobile host disconnection, data caching, replication in managing real-time transactions for mobile ad-hoc network databases, Master's Thesis, The University of Oklahoma, 2002.

[4] M. Matthews, Distributed Transaction Processing with MySQL XA, MySQL Users Conference, 2005.

[5] B. Adelberg, H. Garcia-Molina, B. Gao, Emulating Soft Real-Time Scheduling Using Traditional Operating System Schedulers, in Proc. IEEE RT Systems Symposium, pp 292-298, 1994.

[6] C. Bobineau and P. Pucheral and M. Abdallah, A Unilateral Commit Protocol for Mobile and Disconnected Computing, Technical Report, PRiSM Laboratory, University of Versailles, France, March 2000.

[7] M, Abdallah, R. Guerraoui, P. Pucheral ,One-Phase Commit: Does it make Sense?, ICPADS, pp. 182-192, 1998.