A MOBILE COMMIT PROTOCOL BASED ON TIMEOUTS

Lavanya Sita Tekumalla, BE (CSE)
Osmania University College of Engineering
Hyderabad−500007, India.

ABSTRACT

In a mobile environment, the two phase commit protocol is not suitable for commit processing as it is a blocking protocol and involves a lot of messages. This project deals with implementing a commit protocol (a modification of the Transaction Commit on Timeout Protocol) which is more suitable for a mobile environment as it reduces the number of messages over the wireless channel and tries to achieve the maximum possible commit ratio. This project also examines the impact of introducing the concept of history in deciding upon timeouts. It shows that an estimation mechanism coupled with history can enhance the performance.

Introduction

During the past decade, the concept of mobility has undergone a paradigm shift. Earlier, mobility merely meant that the unit is pulled out from one LAN and plugged into another. Now, with the advances in the area of Wireless Technology, nomadic computing has become a reality. The users retain their network connections even while moving.

The Mobile Environment:

The entire geographic area is partitioned into cells and each cell has a base station. A mobile unit can reside in any of these cells, and it maintains contact with the wired network by communicating with the base station of the particular cell by passing messages over the wireless channel. The base station is a part of the wired network and it can communicate with other nodes in the wired network. The term disconnection is used when the mobile unit detaches itself from the wired network, and the term hand off is used when a mobile unit moves from one cell to another.

Typically, the wireless medium is characterized by Low Bandwidth, Frequent disconnections, High Bandwidth variability, Predictable disconnections, Communication monetarily expensive. Broadcast is physically supported in a cell.

The mobile Host is characterized by small size, small screen, limited battery life, and susceptibility to theft and accidents.

Mobile Transactions:

A mobile transaction is a distributed transaction where some parts of the computation are executed on mobile and some parts on non−mobile hosts. A large number of transactions in a mobile environment are expected to be read−only transactions where users will query large amount of data. Still some applications such as inventory control, banking applications and travel reservations will require updates. To facilitate the operation of the mobile host during disconnections, it is advantageous to execute a part of the transaction on the mobile host itself.
Commit Protocols

The commit phase of a transaction deals with the completion of the transaction and it indicates the fact that any changes made by the transaction are to be made permanent. When talking about commit protocols, the common protocols that come to our mind are the 2PC, and 2PC variants like the Presumed commit, Presumed abort and Early Prepare. The Transaction Commit On Timeout (TCOT) protocol is a commit protocol proposed for the mobile environment.

The Aim Of This Project

The aim of this project is

1. To implement the TCOT protocol. Mobility (hand−offs) is also considered as a part of the implementation and a change of coordinator is made.
2. To introduce a new concept, the concept of coordinator sending its guess of the timeout of cohorts with the help of history (This is not part of the original protocol). We show that this approach prevents some unnecessary overheads in terms of computation and messages. Particularly, the introduction of history decreases the number of estimates (which is explained in the next section). This is an improvement on the original protocol.

The modified protocol

The concept of estimates

This project is largely inspired by the Transaction Commit on Timeout Protocol (the TCOT). In this protocol, the coordinator of the transaction assumes a timeout value before which a vote has to be received from all the cohorts. If a cohort is unable to complete processing the transaction within this time, it sends an estimate message to the coordinator intimating the delay and the coordinator waits for the appropriate time. The TCOT also uses the concept of early prepare.

An Intelligent guess for timeouts using history

This project tries to make the concept of estimates more efficient. It tries to cut down on the overheads by reducing the number of estimates involved so that the protocol becomes more suitable for the mobile environment.

Here, the coordinator tries to guess the timeout value by considering the history. This timeout value is conveyed to the cohort along with the transaction fragment. In case the sender feels that the actual timeout should be greater than the value guessed by the cohort, it can explicitly send an estimate message to modify the timeout value. The receiver (the coordinator) can maintain a history of the previous timeout values to make an intelligent guess of the timeout and reduce the number of estimates.

This approach prevents unnecessary aborts due to a wrong assumption of the timeouts. All the same, it does not cause unnecessary overheads for transactions with uniform predictable timeouts.

A Description Of The Protocol

The Architecture

The following categories of processing elements play a role in this protocol.
Home Cell (HC): The cell where the Home Mobile Unit of the transaction is initially registered.

Home Base Station (HBS): It is the Base Station where the Mobile Unit initially registers.

All communication between Mobile Units and Database Servers is managed by Base Stations. Every Base Station maintains a database distribution directory through which it identifies the set of Database Servers where a portion of the transaction can be processed. If necessary, a Base Station can also fragment a transaction and distribute the fragments to a set of Database Servers for execution.

The transaction: The transaction originates at the Mobile Unit and is split into fragments. One of the fragments is executed at the Mobile Unit. The other fragments are distributed to the various database servers by the Base Station.

The Coordinator: In a scenario where the transaction is executed by a number of nodes, a coordinator is required to manage the commit of the transaction. The base station of the mobile unit is chosen as the coordinator of the transaction (as it alone can communicate directly with the mobile unit).

The commit set: The coordinator of a transaction must know where each of its fragments is processed. A commit set of the transaction is defined as the set of database servers and the home mobile unit that take part in the processing of the commit of the transaction. The commit set comes in use when there is a coordinator change during the transaction.

The protocol can be briefly described as follows.

The mobile unit initiates the transaction. The coordinator distributes this to various database servers. Along with this, it sends its guess of the cohorts timeout value based on history. The cohorts send the work-done indication in the form of a vote message. The mobile unit sends the work done message in the form of a shipment. The coordinator sends a decision in the form of a decision message only if all the cohorts vote positively and if the shipment is received. Estimates are sent by the cohorts as appropriate.

Timeouts:

The Execution timeout (e_t) is the time needed for a transaction fragment to be executed at a particular node.

The Data Shipping Timeout (s_t) is the time taken for the shipment (updates) from the mobile unit to reach the coordinator.

The sequence of steps taking place at each host is elaborated below.

- The Mobile Unit

- A transaction originates at Mobile Unit and the Base Station of this mobile unit is identified as the coordinator. The Mobile Unit takes its part of the transaction and computes its execution timeout. It sends the remaining transaction fragment to the coordinator along with the execution timeout and the shipping timeout.

- While processing the transaction fragment, the Mobile Unit updates its cache copy of the database, composes update shipment and appends it to a log.

- If it is determined that the fragment will execute longer than the execution timeout, then this value is
extended and the new value sent to the coordinator.

−After local commit of the transaction fragment the Mobile Unit sends the shipment of updates to the coordinator. The updates must reach the coordinator before $S_t$ expires.

In the case of the Mobile Unit, it should be noticed that since the mobile unit has the exclusive right over the cache, (there is no question of lending the data to another transaction and the cache does not contain the primary copy of the data in the first place), any changes made to the cache can be easily undone. Thus we can afford to commit the transaction by default at the mobile unit without a commit message in the absence of an abort message. This is not possible in the case of other cohorts (the Database Servers) because after commit, the data is made available to other transactions as the Database Server holds the primary copy of the data.

- Coordinator (the Base Station):

Upon receipt of the transaction from the Mobile Unit, the Base Station creates a token for the transaction fragment of Mobile Unit. A token is used to inform other coordinators the status of fragment processing by commit set members. The coordinator splits the transaction into various transaction fragments and sends them to the set of relevant Database Servers.

−Upon receipt of an estimate from a Database Server it extends the timeout for that fragment and updates the corresponding token.

−Upon receipt of updates (Shipments) from Mobile Unit, the coordinator logs the updates.

−If the coordinator has Mobile Unit shipment before $S_t$ expires and commit messages from other members of the commit set, then the transaction is committed and the result is force written onto the log. (At this time the updates from the Mobile Unit are sent to the Database Server for update to the primary copy of the databases if the primary copy is not kept at the home base station).

−If the total time found by summing all $E_t$ values and the $S_t$ value expires, then abort the transaction and send the global abort message to all participants. Otherwise, send the commit message to all members of the commit set except the Mobile Unit.

- Database Server

−Each Database Server, upon receiving its fragment, computes $E_t$ and sends it to the coordinator. Database Server begins processing its fragment using its own concurrency control technique.

−During processing, each Data Base Server updates its own database.

−If it is determined that the fragment will execute longer than $E_t$, then this value is extended and the new value sent to the coordinator.

−At the end of the transaction fragment it sends a commit message (the vote) to the coordinator and force writes the transaction onto the log.

−If the local fragment aborts (for any reason, including reasons of logic or because $E_t$ expires), then send an Abort message to the coordinator.

**HANDLING HANDOFFS**
The coordinator maintains a commit set, which is the set of all cohorts involved in the transaction. The commit set has tokens to store the status of every cohort that performs the transaction.

When there is a hand-off, the Mobile Unit contacts the base station of the new cell to register itself and at the same time give the address of the previous base station. The new base station acts as the new coordinator and therefore requests the old coordinator to send the commit set.

On receiving the commit set, the new coordinator contacts all the members of the commit set (all cohorts) to inform them of the coordinator change. But before the coordinator change is conveyed to the cohorts, some of the cohorts might have sent messages to the old coordinator. In such a case these messages are forwarded to the new coordinator. Since this forwarded vote may not reach the new coordinator in time, the timeout value of all pending cohorts is increased.

**DISCONNECTIONS**

Though handling disconnections has not been incorporated in the simulation of this protocol, they can be handled without sacrificing consistency by treating a disconnection as an abort.

**How does this protocol compare with the 2PC?**

- This protocol reduces the number of messages
- Reduces unnecessary aborts without too much overheads.

**Why is this protocol suitable for a mobile environment?**

- Without taking messages due to estimates into consideration, there is only one message exchanged between the mobile unit and the wireless network. Thus the number of wireless messages needed to commit the transaction is minimized.
- The protocol offers a method to deal with hand-offs in a very inexpensive way. No additional wireless messages are used as the new coordinator is notified as a part of registration.
- Since the mobile unit caches data it commonly needs for transactions, it can be used for disconnected processing.

**The modifications made to the TCOT protocol in this project are**

- In TCOT, upon receiving the transaction fragment from the coordinator, the cohort always sends an estimate in the form of an explicit message. Here a modification is introduced where the coordinator sends its guess of the estimate along with the transaction fragment (with the help of history) and the mobile unit sends an estimate only if the actual estimate is greater than that guessed by the coordinator.
- In TCOT, an explicit commit message is not sent by the coordinator. Here an explicit decision message is sent to preserve consistency.
The Typical Scenario during a Commit

Mobile Unit  Coordinator  Database Server

Initiation of the transaction

Transaction Fragment

Guess of success

Shipment

Vote

Commit" (decision)

Commit Phase
The first graph shows that the commit rate of the transaction increases by using the concept of estimates. (In this particular graph, the commit rate is 1. Under other circumstances, not considered here, the commit ratio is slightly less than 1. This figure by itself is not significant. However, what is significant is the improvement over the case when no estimates are used at all.)
The second graph shows that the number of estimates is reduced when this mechanism, of coordinator guessing the timeout value, is used.

**CONCLUSION**

**Discussion Of Results**

The concept of estimates need not necessarily be restricted to the mobile environment. But this concept incurs a cost. There are two overheads involved when a cohort sends an estimate.

- The cost of measuring the estimate
- The cost of sending the estimate

*My contribution*: The approach of coordinator making an intelligent guess of the timeout serves to reduce the second overhead. This reduction in the number of messages sent is specially meaningful in a mobile environment. This is because messages over the wireless channel are very costly.

In effect, this project says that an estimate mechanism is feasible in the mobile environment too provided it is backed by this method of coordinator making an intelligent guess of the timeout value.

This method of coordinator storing history and guessing the timeout has a rider. Storing history itself incurs a cost, which is insignificant when the coordinator knows
about only a small number of cohorts. But as the number of cohorts increases, demands on memory increase significantly. However, with decreasing memory cost, memory is not too difficult to sacrifice at the coordinator.

**Lines For Future Work:**

One feature that has not been incorporated here is the cohort’s mechanism for estimating the timeout. The TCOT makes an exhaustive analysis on the feasibility of having an estimate mechanism. The TCOT considers the effect of variations in the multiprogramming level, probability of cache hit and probability of conflicts on the execution timeout. In this project, the cohort estimates its timeout by considering the variation in level of multiprogramming. I hope to incorporate the other factors too into the project.

**REFERENCES**


