Disconnection Tolerance in Soft Real-Time Mobile Databases

Laurent Amanton, Bruno Sadeg and Samia Saad-Bouzefrane
LIH, Faculté des Sciences et Techniques
25, Rue Philippe Lebon
76058 Le Havre Cedex, France
{amanton, sadeg, bouzefrane}@univ-lehavre.fr

Abstract

Many current applications widely use wireless technologies where mobile hosts communicate with fixed base stations thanks to wireless communications. These applications have often real-time constraints. The main problem is then to efficiently manage transactions despite the network disconnections. In this paper, we study the problem of insuring computing continuity of query real-time transactions in spite of the disconnection problems. To this purpose, we develop a protocol, called DT-protocol, where are taken into account the frequent disconnections of the wireless network. We show, through simulations results, that DT-protocol brings a value added in the management of query transactions in wireless environment.

Key-words: mobile databases, soft real-time transactions, wireless environment, network disconnection

1 Introduction

Rapid advances in wireless communications technologies have led many current applications to widely use these technologies. For example, in an electronic commerce application, there is a main distributed database (located in a fixed site, say S) and there are actors of the commerce equipped with small mobile devices (mobile hosts, MHs). The MHs communicate with S via wireless networks. In this system, transactions done by the mobile devices, notably query transactions, must receive results from the server as timely as possible in order to avoid economical consequences. In this application, each mobile device contains a view of the database and it must be possible for each vendor/customer to reach the fixed site in order to execute a transaction. The results must be provided by the server as timely and possible despite the network disconnections. The main problem is then to insure transactions computing continuity. In this paper, we address this problem and we focus particularly on soft real-time query transactions. The paper is organized as follows. In section 2, the model we propose is described. In section 3, we recall the epsilon-delta and delta-deadline concepts. Section 4 presents the main contribution of this paper, that is a protocol to manage soft real-time query transactions in wireless environment. In section 5, are presented some simulation results. In section 6, an overview of the researches done on issues met in a mobile environment is given. In section 7, we have summarized this paper and some extensions to this work are presented.

2 Modelization

We consider a mobile system which consists of a set of fixed base stations and mobile hosts. Without loss of generality, we use one fixed site and N mobile nodes, called mobile hosts (`MH`) linked by wireless communications. We consider a main database located in the fixed site and distributed over all mobile nodes so as each `MH` contains a small replica of the main database. Each `MH` may query or update the database server. We also assume that the applications may tolerate approximate results and transactions deadline overruns. Generally, disconnections durations are frequent but of little duration so as a period of disconnection is often (in probability) less than the extended transactions deadlines [6]. Therefore, when the connection is reestablished, the server either sends the result to the `MH` or becomes ready to receive queries or updates from `MH`s. The main objective is then to insure the availability and freshness of information in the database, in spite of disconnections.

3 ε-data and Δ-deadline concepts

ε-data and Δ-deadline notions have been proposed by Saad et al. [6] in order to deal with controlled imprecision when manipulating data items, and to deal with controlled transactions deadline overruns. The traditional compatibility matrix used in transactions concurrency control has been modified in order to take into

account approximate data values. This allows read and write transactions to access simultaneously the same data item under certain conditions. These conditions may be summarized by saying that the imprecision introduced by this concurrent access to a data item must be bounded by the imprecision tolerated by this data.

In this paper, these notions are applied to deal with the network disconnections, allowing then soft real-time transactions to meet their deadlines.

4 DT-Protocol description

We propose a protocol to ensure computing continuity of soft real-time transactions. Each transaction have an initial deadline and an extended deadline (Δ-deadline). Let T be a query transaction sent by a MH to the server S. In order for the MH to receive an exploitable result, S must send a timely result. However, due to the disconnections, S must determine at which time it will send the result. If S estimates that a disconnection will occur before it sends the whole result, then S sends a partial result before the disconnection. Sometimes, the server may estimate that it can be better to wait in order to send the better result. The problem is then to determine either the last moment for sending a result before a possible disconnection (called *response time limit*) or to determine if it is better to wait the re-connection instant.

Main notations

We now give the main notations and data structure used.
MH : mobile host;
S : server;
$\mathcal{D}$: transaction deadline;
$\mathcal{D}_\Delta$: extended deadline;
WQ : server ready transactions queue;
$\mathcal{L}_{rt}$: $\mathcal{L}$imit for the response time; it represents the farthest date after which the server must send an answer; after, a disconnection between S and MH may occur;
δ : maximum duration of a message transmission between S and a MH, without a disconnection;
$MinC$: minimum duration of a reliable communication before a probable disconnection;
$MaxD$: maximum duration of a disconnection period before a new communication establishment.

Computing of the response time limit

Now, we'll check whether or not an interrupted communication may be resumed after the disconnection duration. When a new connection is established at time t_c, the server makes an estimation whether or not the communication will be terminated at least before the end of the response time limit $\mathcal{D}(T_Q) + \Delta(T_Q)$ (we denote by $\mathcal{D}_\Delta$ this instant). Figure 1 illustrates the three possible cases:

1. $MinC < \mathcal{D}_\Delta - t_c < MinC + MaxD$: if a disconnection occurs before the transaction reaches its extended deadline $\mathcal{D}_\Delta$, then the transaction has less chances to meet its deadline $\mathcal{D}_\Delta$. Then the server must restart the transaction if it is not too late. Then, the response-time limit must be fixed to $\mathcal{L}_{rt} = t_c + MinC$.
2. $0 < \mathcal{D}_\Delta - t_c < MinC$: a disconnection will not occur before time $\mathcal{D}_\Delta$. Therefore, the farthest instant to send the answer must be equal to $\mathcal{D}(T_Q) + \Delta(T_Q)$; $\mathcal{L}_{rt} = \mathcal{D}_\Delta$.
3. Let t be the current time. If $MinC + MaxD < \mathcal{D}_\Delta - t_c$ and $\mathcal{D}_\Delta + t_c - t < MinC + MaxD$ then we have to consider that the communication may continue until the limit $\mathcal{D}_\Delta - MaxD$. So, we have $\mathcal{L}_{rt} = \mathcal{D}_\Delta - MaxD$. If the communication is interrupted and reestablished again, then the response time limit will become as we have seen in one of the previous cases.

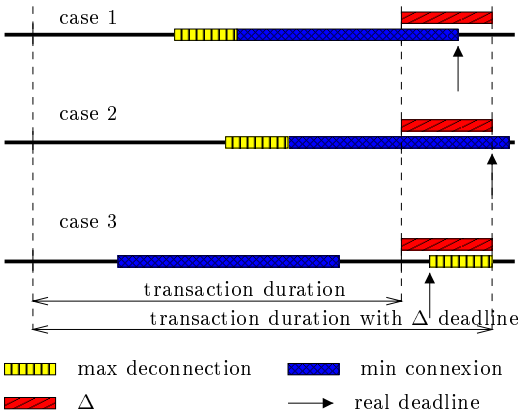

Figure 1: How to find the response time limit

Algorithm

The following algorithm is composed of two parts: one part is executed on the mobile host and the other part is executed on the server.

Details of the algorithm

On MH_i

```
◇ A query transaction is sent to S
Connection to the server S and initialization:
      AccurateValueObtained (T_Q_ij) = false
      Send Request(T_Q_ij, Δ_ij)
◇ Reception of the transaction reply (d, precision) from S
if ¬precision then TemporaryValue(d) = d
else /* Use d as a final value */
AccurateValueObtained (T_Q_ij) = true
◇ Periodically at date t (the period is fixed by MH_i)
if ((t > D_Δ(T_Q_ij) + 2δ)) and
      (AccurateValueObtained(T_Q_ij) = false)
      then use TemporaryValue(d) as the final value
```

On S:

Transaction scheduler

◇ *Reception of the transaction request $T_{Q_{ij}}$ sent by MH_i*
/* Initializes the response-time limit of $T_{Q_{ij}}$ */
$\mathcal{L}_{rt}(T_{Q_{ij}}) = \mathcal{D}_{\Delta}(T_{Q_{ij}})$
/* and insertion of $T_{Q_{ij}}$ in the ready-transactions queue wq*/
inserts $(T_{Q_{ij}},\ \mathcal{D}_{\Delta}(T_{Q_{ij}}),\ \mathcal{L}_{rt}(T_{Q_{ij}}))$
◇ *On each activation (date t)*
if ($T_{Q_{ij}}$), an active transaction has terminated **then**
call Commit_Proc ($T_{Q_{ij}}$)
else inserts it in wq
foreach $T_{Q_{ij}}$ in wq do
if $\mathcal{D}_{\Delta}(T_{Q_{ij}}) < t$ **then** call Abort_Proc ($T_{Q_{ij}}$)
/* because ($T_{Q_{ij}}$) has missed its extended deadline */
else /* computes the response-time limit for $T_{Q_{ij}}$ */
/* by using the function $RespTimeLimit$: */
$\mathcal{L}_{rt\,ij}$ =RespTimeLimit ($T_{Q_{ij}}$,$\mathcal{D}_{\Delta}(T_{Q_{ij}})$, $MinC_i$, $MaxD_i$)
Sort (wq) /*according to the response-time limits*/

Transaction manager

◇ *Reception of the transaction request $T_{Q_{ij}}$ sent by MH_i*:
/* Because $T_{Q_{ij}}$ has not started yet, we have: */
$Commit(T_{Q_{ij}}) = false$
◇ *Commit_Proc ($T_{Q_{ij}}$): - used by TM to commit -*
/* an accurate value of d is sent to MH_i, with conditions: */
$Commit(T_{Q_{ij}}) = true$

Wait (MH_i connected or $t \geq \mathcal{D}_{\Delta}(T_{Q_{ij}})$)
if $t \leq \mathcal{D}_{\Delta}(T_{Q_{ij}})$
AccurateValueObtained= $true$ /* precise value is sent */
send Reply(d, AccurateValueObtained) to MH_i
else Abort_Proc ($T_{Q_{ij}}$)
◇ *Abort_Proc ($T_{Q_{ij}}$):* Rollback $T_{Q_{ij}}$
◇ *On each activation (date t)*
if $t = ResponseTimeLimit(T_{Q_{ij}}) - 2\delta$ **then**
if $Commit(T_{Q_{ij}}) = false$ **then** */
AccurateValueObtained($T_{Q_{ij}}$) = false
send Reply(d, AccurateValueObtained) to MH_i

Function called by the scheduler

Function RespTimeLimit ($T_{Q_{ij}}$,$\mathcal{D}_{\Delta}(T_{Q_{ij}})$, $MinC_i$, $MaxDi$)
Computing $\mathcal{L}_{rt}(T_{Q_{ij}})$ ***(at time t)***

Begin
Let $T_{Q_{ij}}$ be the active query transaction
$t_c = GetConnectionDate(T_{Q_{ij}})$
Case $\mathcal{D}_{\Delta}(T_{Q_{ij}}) \in [t_c + MinC, t_c + MinC + MaxD]$:
$\mathcal{L}_{rt}(T_{Q_{ij}}) = t + MinC$
Case $\mathcal{D}_{\Delta}(T_{Q_{ij}}) \in [t_c, t_c + MinC]$:
$\mathcal{L}_{rt}(T_{Q_{ij}}) = \mathcal{D}_{\Delta}(T_{Q_{ij}})$
Case $\mathcal{D}_{\Delta}(T_{Q_{ij}}) > t_c + MinC + MaxD$:
$\mathcal{L}_{rt}(T_{Q_{ij}}) = \mathcal{D}_{\Delta}(T_{Q_{ij}}) - MaxD$
Return $\mathcal{L}_{rt}(T_{Q_{ij}})$
End

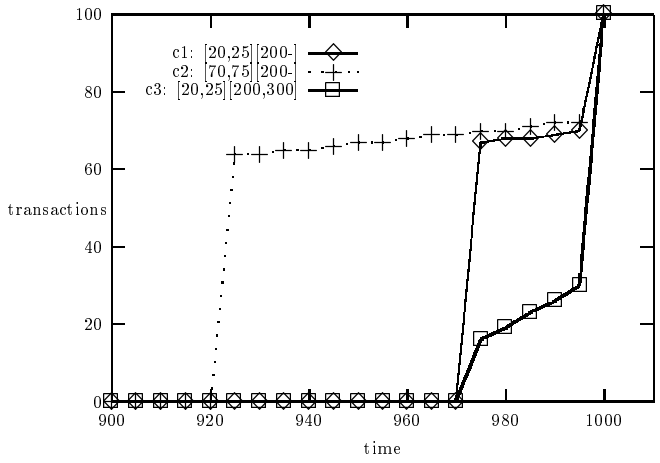

Figure 2: Cumulative diagram where connections duration are larger than disconnections duration

5 Simulation results

5.1 Response time limit adjustment

Because the transaction duration is variable, the response time limit must be adjusted. This section presents the impact of disconnections on the response time limits, and a method to adjust them is presented, according to the situations we have explained in section 4. The diagram 2 shows specific configurations for connection and disconnection durations. Each point of a curve represents the rate of transactions that must be terminated at a given deadline.

For the simulation, we have considered one million of successive transactions whose execution-time is equal to 1000 units. A $\Delta-$deadline is assigned to each transaction. Figure 2 depicts three cases where the disconnection duration is relatively short in comparison with the connection duration. The intervals for each curve c_i, represent respectively the disconnection duration and the connection duration. For example, The first curve c_1 has a disconnection duration that fluctuates between *20* and *25* units of time and a connection period that has only a minimum value of 200 units.

For all the configurations c_i, the first limit for the server to send a reply corresponds to *the extended deadline* minus *the maximum disconnection duration*. When the connection duration is very large, the third case becomes preponderant (c_1 and c_2). We note that most of the answers are to be returned to the mobile host before the initial deadline, then the progression is relatively constant up to the deadline. As soon as the connection duration is bounded and relatively deterministic (short interval) as for c_3, most of transactions can meet their

extended deadlines.

As a conclusion, we state that even if the communication is frequently interrupted, the response time limit depends only on the disconnection duration. The main problem consists then of determining this instant.

5.2 The $\varepsilon - data$ usefulness

The $\varepsilon - data$ notion is useful when the server has not yet finished its transaction processing and the response time limit is close be reached. In this case, it's possible for the server to send an approximate value ($\varepsilon - data$) before the response time limit $\mathcal{L}_{rt}(T_Q)$. After this instant, the system evolves in three cases:

- the disconnection lasts until $\mathcal{D}(T_Q) + \Delta(T_Q)$ then the only exploitable value by MH is the $\varepsilon - data$ already received.
- the communication is still established until $\mathcal{D}(T_Q) + \Delta(T_Q)$, then the server have the possibility to send the exact value during the added time (if the transaction extended deadline is still met).
- the communication is cut and reestablished before the deadline $\mathcal{D}(T_Q) + \Delta(T_Q)$ expires, then the server may send the exact (or complete) value if it has already computed it before this deadline.

6 Related work

For more than a decade, several works have been done on mobile databases. Walborn et al. in [10] deal with consistency in mobile databases in PRO-MOTION. In their paper, the authors designed a method which allows to transform global constraints into local constraints in the database in order to achieve consistency. Sang Keun et al. in [7] deal with a new protocol to support transactional cache consistency in a wireless computing environment. Dunham et al. [2] used analytical performance to study the impact of mobility on three different options to the management of mobile transactions. Shek et al. [8] developed information dissemination services to support the dissemination and maintenance of situations awareness over a satellite wireless network infrastructure. In [1], Badrinath et al. gave a method to implement deferred transmissions in a mobile host. Keller et al. [3] studied the management of the intermittent connectivity in independent devices. In [5], Pitoura et al. exploit versions for handling updates. However, few work has been done on real-time management in wireless environment. Among these works, the paper presented by Lam et al. [9] proposed Distributed Hight Priority-Two-Phase Locking protocol (DHP-2PL) which deals with transaction concurrency control in mobile distributed environment. They notably used data similarity notion [4] in order to deal with network disconnections.

7 Conclusion and future work

Many current applications use large databases located in fixed sites and mobile hosts communicate with these databases via wireless links. The main problem is then to efficiently manage frequent network disconnections inherent to this kind of systems. Furthermore, more of these applications are real-time in nature. In this paper, we presented DT-Protocol, a protocol that deals with the soft real-time transactions sent to and received from mobile databases in order to achieve the continuity of computing and the database consistency. This will be a new challenge in mobile computing where disconnections between user devices and the main sites occur frequently. An other issue that we are dealing with is the reconciliation issue in mobile databases. The reconciliation process consists of updating locally database replica and of homogenizing these local updates according to the global database consistency.

References

[1] B.R. Badrinath, P. Sudame, " To Send or not to Send: Implementing Deferred Transmissions in a Mobile Host", *Int. Conf. on Dist. Computing Systems*, 1996.

[2] M.H. Dunham, V. Kumar, "Impact of Mobility on Transaction Management", *proc. of the MobiDE'99*, 1999.

[3] A.M. Keller, O. Densmore, "Zippering: Managing Intermittent Connectivity in DIANA", *ACM NOMAD (Mobile Networks and Applications)*, **Vol. 2**, 1997.

[4] T.W. Kuo and A.K. Mok, "SSP: a Semantics-Based Protocol for Real-Time Data Access", *the proc. of IEEE Real-Time Systems Symposium (RTSS'93)*, 1993.

[5] E. Pitoura, P.K. Chrysanthis, "Exploiting Versions for Handling Updates in Broadcast Disks", *proc. of VLDB*, 1999.

[6] S. Saad-Bouzefrane, B. Sadeg, "Relaxing the Correctness criteria in Real-Time DBMSs", *IJCA journal*, Vol. 7 N°4, 2000.

[7] Sang Keun Lee, Chong-Sun Hwang, HeoChang Yu, "Supporting Transactional Cache Consistency in Mobile Database Systems", *proc. of the MobiDE'99*, 1999.

[8] E.C. Shek, S.K. Dao, Y. Zhang, D.V. Buer, " Dynamic Multicast Information Dissemination in Hybrid Satellite-Wireless Networks", *proc. of the MobiDE'99*, 1999.

[9] O. Ulusoy, "Real-Time Data Management for Mobile Computing", *Proc. of Int. Workshop on Issues and Applications of Database Technology (IADT-98)*, 1998.

[10] G.D. Walborn, P.K. Chrysanthis, " PRO-MOTION: Support for Mobile Database Access", *Personal Technologies*, Vol. 1, n 3, pp. 171-181, 1997.