An Update Algorithm for Replicated Signaling Databases in Wireless and Advanced Intelligent Networks

Kin K. Leung
AT&T Laboratories
Phone: 908-888-7041
Fax: 908-949-1436
Email: kkleung@att.com

Abstract In the Personal Communication Networks (PCN), and the Universal Personal Telecommunications (UPT) services and possibly other services offered by the Advanced Intelligent Networks (AIN), customer records for call routing and other signaling functions can be distributed and replicated at multiple sites to improve access delay and system availability. We observe that these applications can tolerate data inconsistency among replicated records for a short period of time because the main consequence of accessing obsolete data is a small probability of call misroute. By exploiting this observation, we propose the *Primary-Writer Protocol (PWP)* as the concurrency control and commitment protocol for updating the replicated databases. The fraction of calls misrouted under the PWP is analyzed. Our results reveal that the fraction of calls misrouted under the PWP is very small for the expected customer behavior in the PCN (wireless networks), and the UPT and other AIN services.

Index terms Advanced intelligent networks, concurrency control, distributed database, mobility management, replicated database, signaling database, signaling networks, update protocols, wireless networks.

1. INTRODUCTION

The *Personal Communication Network (PCN)* consists of a fixed network and a wireless network. It allows mobile customers to initiate and receive calls, while moving from one location to another. The PCN supports this terminal mobility by maintaining customer location information in network databases. The current approach to supporting such mobility requires a *home database* and a *visitor database* [8]. The routing and other signaling functions of each call for a mobile customer access the information stored in the databases. In the Advanced Intelligent Networks (AIN), the *Universal Personal Telecommunications (UPT)* services will enable customers to access other telecommunication services while allowing personal mobility [7]. In order for the AIN to support this personal mobility, it also has to rely on the extensive use of databases for call routing and other signaling functions.

As the PCN and UPT traffic load grows, the number of queries and updates to the databases will also increase commensurately. Depending on the specific needs of the services, it will be advantageous to distribute and replicate customer records at multiple places of the signaling network for easy access (for example, see [4]). Thus, it is essential to design such databases to achieve a satisfactory level of efficiency, robustness, reliability, and availability.

Much research effort has been dedicated to the study of distributed database systems; see e.g., [2], [3] and [5]. The main drawback of distributing and replicating data items at physically separated sites is the overhead incurred by the use of concurrency control protocols to maintain the correctness and consistency of records stored at various sites. To allow recovery after system failures at different sites, a set of commitment protocols becomes necessary. The commitment protocols not only complicate the system design, but also add more overhead to the system operations.

It is interesting to observe that the concurrency control and commitment protocols proposed in the literature are devised for general use. They do not account for the specific characteristics of the applications at hand. To illustrate the unique characteristics of the telecommunication services, let us first consider banking applications. In a distributed database of a banking system, a customer record, including the account balance, can be replicated at multiple sites. The concurrency control protocol in use maintains the consistency of the replicated records at all times. Otherwise, banks will suffer significant monetary loss as customers can easily overdraw their accounts from different sites simultaneously. Thus, expensive concurrency control mechanisms must be used to prevent such fraudulent acts from happening.

In contrast, if the replicated copies of customer records are inconsistent in a communication network, some calls may be routed according to the obsolete information. The major consequence will be "*call misroutes*" and loss of revenue. As long as the occurrence of misroutes is kept under a certain level, neither the network operating companies nor the customers will recognize the effects. Therefore, we can exploit this unique attribute to simplify the design of concurrency control and commitment protocols for the distributed database of customer records. Even if a sophisticated protocol is used to guarantee data consistency at all times, it will either cause call blocking if records are locked, or delay the availability of new customer information for call processing due to control overheads. In this spirit, we propose a new algorithm, the *primary-writer protocol (PWP)*, for updating replicated databases for the PCN and UPT services.

The organization of the rest of this paper is as follows. We propose and examine the operations of the PWP in Section 2. We analyze the performance of the PWP in Section 3. Finally, Section 4 presents our concluding remarks.

2. AN EFFICIENT CONCURRENCY CONTROL & COMMITMENT PROTOCOL

Since the distributed database issues are common to the PCN and the UPT services, the general database terminology is adopted here. Specifically, a computer system where parts of the database are located is referred to as a *site*. Customer records for call routing, features, service profiles, and so on are simply called *records*. Reads and writes to the database are also referred to as *queries* and *updates*, respectively. For failure recovery, each site keeps a *log* (also known as *journal*) of updates made to the local database in a stable memory, which can survive the failures.

Queries associated with a call and updates for a customer access a single record. That is, the *readset* and *writeset* of a query and an update are one record. Thus, if queries and updates are processed *atomically*, unnecessary data inconsistency can be avoided. Furthermore, several queries may access the associated record during the call setup time, which typically lasts a few seconds. Although the record may have been updated in the meantime, it is advantageous not to immediately remove an obsolete copy of the record so that subsequent queries of the calls in progress can be processed in a consistent way according to the previous record. Thus, this eliminates the need of locking records for query processing. Additionally, call processing will not be disrupted while the customer’s record is being updated.

Since the readset and writeset consist of one record, it is natural to treat records as the unit for distribution and replication at multiple sites. A concurrency control protocol is required to maintain the internal and mutual consistency, while the commitment protocol is needed for failure recovery. Because of the nature of the applications, we devise a single protocol to perform the concurrency control and the commitment functions.

Let us first make several assumptions about the system:

a. All sites are connected by a signaling network and messages sent from one site to another may be lost. Even if messages reach their destination sites, they may not necessarily arrive in the order in which they are sent.

b. Records of the database are distributed and replicated at the sites. Suppose that a record (say record A) is replicated at N sites, indexed by $i = 1, 2, \ldots, N$. To take advantage of load balancing, the initial queries of calls requiring access to record A are sent to these N sites according to some call distribution algorithm (e.g., [1] and [4]). To avoid unnecessary data inconsistency if a call setup involves multiple queries to the same record, subsequent queries of a call are routed to the same site where the initial query of the call is processed.

c. Each site can keep multiple versions of a record [3] so that queries for a call can access the same version of the associated record for consistent signaling information. An obsolete version of the record is deleted after the processing of all calls querying it has been completed.

d. For recovery purposes, each site records the update activities in a log in stable storage.

2.1 The Primary-Writer Protocol (PWP)

For each record A, one site with a copy of the record is selected to be its *primary site (PS)* and all other sites having a copy of the same record are referred to as the *secondary sites (SS's)*.

Queries for record A are routed to the PS or any of the SS's according to the call distribution algorithm. However, all updates for record A must be first submitted to and processed by the PS. If the processing of an update is completed without causing any data inconsistency, the update is committed at the PS. Then, the PS sends the update to all SS's to update their replicated copies. This concurrency control and commitment protocol is referred to as the *Primary-Writer Protocol (PWP)*.

A PS can serve as the PS for many records stored at the same site. In addition, customer data in *Home Location Register (HLR)* can be replicated in a cellular network where the customer is roaming, and assign the replicated HLR to be the PS. This shortens the update delay because updates are likely to be generated by the roaming customer.

To use the PWP, two fields are added to each record: a *version number (VN)* and a *call counter (CC)*. The VN is used to identify the version of a record, while the CC is to indicate the number of ongoing calls, which have previously accessed that version of the record. If the CC is zero and the record has been updated, the older version of the record can be deleted. Let $R_A(n)$ be the version of record A with the VN being n for $n = 0, 1, 2, \cdots$, and $CC(n)$ denote the CC of $R_A(n)$. Assume that $R_A(0)$ initially exists in the system. Further, let *vn_late* be the VN of the latest version of record A existing at a site. We use U_n to denote an update to record A, which successfully creates $R_A(n)$. Using this notation, the PWP is presented in Figure 1.

2.2 Properties of the PWP

Recall that all updates for record A are first processed by the PS. Thus, if an update causes any data inconsistency, it will be rejected (aborted) by the PS; otherwise, the update is committed. In fact, the PWP also preserves the internal and mutual consistency for all replicated copies of record A, as stated in the following.

Proposition 1: If the copies of $R_A(0)$ replicated at the PS and all SS's are identical and internally consistent, the PWP maintains the internal consistency for all replicated copies of record A.

Proof: Consider a set of updates $\{U_n; n = 1,2,3, \cdots\}$ committed at the PS where U_n maps $R_A(n-1)$ to $R_A(n)$. The mapping is denoted by $R_A(n) = U_n(R_A(n-1))$. Since $R_A(0)$ is consistent and since U_n's are committed by the PS only if they cause no inconsistency, $R_A(n) = U_n(U_{n-1}(\cdots U_1(R_A(0)) \cdots))$ for all $n = 1,2,3,...$ are always internally consistent at the PS. By use of the VN's, the PWP forces the U_n's to be posted at each SS in the same sequence in which they are committed by the PS. Thus, the internal consistency at the PS implies internal consistency at all SS's. □

The PWP maintains data consistency by a mechanism similar to that for the exclusive-writer protocol [6], although the latter does not consider failure recovery and the characteristics of telecommunication services. We also note that once an update is committed by the PS, it can always be committed at the SS's. Thus, database rollback is not needed.

Proposition 2: If the copies of $R_A(0)$ replicated at the PS and all SS's are identical, the PWP maintains the mutual consistency for all replicated copies of record A.

Proof: Suppose that updates $\{U_n; n = 1,2,...,N\}$ have been committed by the PS and no further updates will be submitted. The use of VN's by the PWP forces the updates U_n's to be posted at all SS's in the sequence in which they are committed at the PS. Since $R_A(0)$ is identical at all sites, copies of $R_A(N)$, which equals $U_N(U_{N-1}(\cdots U_1(R_A(0)) \cdots))$, at all sites are identical after these updates U_n's are processed by all sites (although some of them may have failed for a finite amount of time). □

It is noteworthy that the PWP allows non-identical copies of records replicated at different sites to be available for call processing simultaneously. Thus, queries may access the obsolete information at some SS's while updates are pending for processing, thus causing call misroutes. As discussed earlier, if the probability of such occurrence is satisfactorily small, there is no need for exclusive access to the records during updates. Without exclusive access, the PWP is thus deadlock free. Furthermore, the PWP does not require extensive message exchanges among sites, which would otherwise be needed for concurrency control protocols such as locking to provide the exclusive access. Given that a record is replicated at N sites, the total number of messages exchanged between the PS and the SS's for each update is $2(N-1)$. This is the minimum number of message exchanges to support the degree of mutual and internal consistency needed for call routing and signaling. Of course, the PWP can be used in other applications as long as they can tolerate the record inconsistency and the probability of accessing obsolete data is at an acceptable level.

As for the recovery ability for the PWP, let us consider the following possible scenarios.

a. *Loss of Messages and Link Failures*: Any loss of update and acknowledgement messages will cause the timeout to expire on the PS. When this happens, the PS resends the update message to the associated sites.

b. *Site Failures*: If the PS fails before an update is committed in the log, the update is simply lost. The source discovers this if it does not receive a completion message from the PS after a certain time period. In case the PS fails after a committed update has been written in the log, the PS can resume its operations after the failure as if the update was just written in the log. As a result, the update is sent to all SS's again. For those SS's which have already posted the update successfully, they simply discard the duplicated update and send an acknowledgement back to the PS. If a SS fails, the PS resends the update, if any, to the failed site when the timeout associated with the update expires. When the failed SS recovers, it can also resume its operations as if the last update was just written in the log. (Several failure recovery strategies for cellular networks can be found in [11].)

3. **PERFORMANCE ANALYSIS OF THE PWP**

In this section, we analyze the fraction of calls misrouted under the PWP. Readers are referred to [10] for comparison of the PWP with the *Primary-Site Locking* [5] and the *Basic Time-Stamp* [5, 2] protocols.

3.1 Analysis of Call Misroute

A key performance measure for the PWP is the fraction of calls misrouted (denoted by P_m) due to inconsistent copies of records at different sites. Recall that a record is first updated at its PS under the PWP. Let us define a *vulnerable period* for the record at a SS as the time interval from the completion of an update to the record at the PS until the time when the update is posted at the SS. As the most updated record has been available at the PS at the start of the vulnerable period, queries accessing the record at the SS during the time period reference obsolete data, thus causing call misroutes. Certainly, such misroutes will not occur, should there be no record replication. Figure 2 presents a timing diagram to the PWP. For example, the vulnerable period for update U_n at the SS *i* is from time A to B.

The length of a vulnerable period depends on: a) the network delay incurred by an update message sent from the PS to the SS, and b) the update response time at the SS. These two factors in turn depend on the characteristics of the signaling network, the system design of each site, the traffic load, and the service scheduling discipline for query and update processing. To gain a general understanding of the performance of the PWP, we assume the following:

1. Each site has a single computer responsible for query and update processing. Queries and updates are processed on a first-come-first-served (FCFS) basis.

2. Let record A be replicated at N sites, indexed by $0, 1, \ldots, N-1$, where site 0 is the PS and other sites are SS's. Assume that each call generates only one query and queries for the record are sent to these N sites according to some call distribution algorithm. For the algorithm in use, let queries for record A and queries for all other records arrive at each site i according to two independent Poisson processes with rate α_i^A and α_i^O, respectively.

3. Updates for all records other than record A arrive at each site i according to a Poisson process with rate β_i^O.

4. Updates for record A are submitted to site 0 at a rate β^A. The time interval between the processing completion epochs of two successive updates for record A at site 0 has an exponential distribution.

5. Message delays in the signaling network (which connects all sites together) are constant. The message delay from site 0 to each other site i is D_i and its Laplace transform (L.T.) is denoted by $D_i^*(s)$.

6. The processing (service) time for acknowledgements is negligibly small.

Denote the L.T.'s for the processing times of a query and an update at any site by $X_q^*(s)$ and $X_u^*(s)$, respectively. Let their averages be $\bar{x}_q$ and $\bar{x}_u$. We define the *waiting time* as the time interval from the arrival of a query or update until its processing is started at a site.

Lemma 1: Let $W_i^*(s)$ be the L.T. for the waiting time at site *i*. We have

$$W_i^*(s) = \frac{s(1 - \rho_i)}{s - \lambda_i + \lambda_i B_i^*(s)} \tag{1}$$

where $\rho_i = (\alpha_i^A + \alpha_i^O)\bar{x}_q + (\beta^A + \beta_i^O)\bar{x}_u$, $\lambda_i = \alpha_i^A + \alpha_i^O + \beta^A + \beta_i^O$ and $B_i^*(s) = [X_q^*(s)(\alpha_i^A + \alpha_i^O) + X_u^*(s)(\beta^A + \beta_i^O)]/\lambda_i$.

Proof: Due to Assumptions 4 and 5, updates for record A arrive at any site i $(=1,2,...,N)$ according to a Poisson process. Combining this with Assumptions 2 and 3 of the arrival processes for queries and other updates, each site can thus be modeled as an M/G/1 queue. The proof is completed by use of existing results for the M/G/1 queue [9]. □

A call accessing record A is successfully routed by site *i* for $i = 1$ to $N - 1$, if it accesses the current version of the record. That is, site *i* has the most updated version of record A available as site 0 does when the call is processed.

Lemma 2: Let P_i^s be the probability of successful routing of an arbitrary call for record A at site *i*. Then,

$$P_i^s = D_i^*(\beta^A)\ W_i^*(\beta^A) \tag{2}$$

where $D_i^*(.)$ is the L.T. for the network delay and $W_i^*(.)$ is given by (1).

Proof: Assume that the waiting time (a random variable) for an arbitrary call accessing record A at site *i* is denoted by W_i. Recall that the network delay from site 0 (i.e., the PS) to site *i* is D_i. If the last update to record A is completed at site 0 either less than D_i time units prior to the arrival of the call or during the waiting time W_i, site *i* will not have processed the update before the call due to the FCFS service. Then, the call accesses the obsolete record at site *i*, and is thus misrouted. On the other hand, if the last update to record A is completed at site 0 at least D_i time units prior to the arrival of the call, then the call can access the most updated record at site *i*, and is thus routed successfully. Hence, P_i^s is equal to the probability that no update to record A is completed by site 0 in a combined time interval of D_i and W_i. We thus obtain (2) by Assumption 4 that the time interval between the completion epochs of two successive updates for record A at site 0 is exponentially distributed. □

Proposition 3: The total throughput of successful calls (to be referred to as "goodput") for record A at all sites is

$$\gamma = \alpha_0^A + \sum_{i=1}^{N-1} \alpha_i^A \, D_i^*(\beta^A) \, W_i^*(\beta^A) \tag{3}$$

and the fraction of calls misrouted due to inconsistent records at the SS's is given by

$$P_m = \sum_{i=1}^{N-1} \alpha_i^A \left[1 - D_i^*(\beta^A) \, W_i^*(\beta^A) \right] / \sum_{i=0}^{N-1} \alpha_i^A . \tag{4}$$

Proof: Since calls can always access the latest version record A at site 0 (i.e., the PS), its goodput is α_0^A. By Lemma 2, the goodput associated with record A at site *i* is $\alpha_i \, D_i^*(\beta^A) \, W_i^*(\beta^A)$. Summing the goodput at all sites yields (3). Then, (4) is simply one minus the ratio of goodput to the total access rate to record A. □

3.2 Numerical Study for Call Misroute Under the PWP

We assume that the *N* sites where record A are replicated have identical traffic load. That is, $\alpha_i^A = \alpha^A$, $\alpha_i^O = \alpha^O$, and $\beta_i^O = \beta^O$ for $i = 0, 1, ..., N-1$. Due to this balanced traffic, the total call throughput for record A is given by $N\alpha^A$. Further, let the query-update ratio be denoted by R_q, which is assumed to be identical for all records. Thus, $R_q = \alpha^A / \beta^A = \alpha^O / \beta^O$. Assume that the query and update processing times are exponentially distributed with averages, $\bar{x}_q = 5$ msec and $\bar{x}_u = 10$ msec, respectively. The computer occupancy of each site is denoted by ρ and

$\rho = (\alpha^A + \alpha^O)(\bar{x}_q + \bar{x}_u / R_q)$. The traffic load at each site is adjusted so that ρ is set to 90%. The network delay between site 0 and every other site i is $D_i = 10$ msec.

Using these parameters, the fraction of calls misrouted is depicted in Figure 3 for the 2-site replication, and a wide range of R_q and call throughput for record A. Usually, the call throughput of a mobile customer in the PCN or cellular networks will be at most tens of calls per hour. Assuming that R_q associated with these mobile customers can reach as low as 1, the probability of call misroute is less than 0.1%, which is lower than the typical call blocking probability in the wireless networks. Hence, such low probability of misroute will not affect the quality of service as perceived by customers. This same remark also applies to individual customers of the UPT services.

On the other hand, Figure 3 reveals that the probability of call misroute is unacceptable when R_q is small and call throughput is very high. Fortunately, to our knowledge, such a situation does not occur in practice. Consider big business customers with a call throughput of ten thousand calls per hour. It is rare for them to update their records more than once every minute. As a result, R_q is larger than 1,667. Correspondingly, the probability of call misroute for these customers drops to a satisfactory level of well below 0.1% for $R_q \geq 1{,}667$. For customers with moderate call throughput of of 1,000 to 10,000 calls/hour, their misroute probability is also satisfactorily low, as long as R_q is larger than a few tens, which is likely to be the case for medium sized business customers using the AIN services.

The fraction of misroutes is slightly increased as the record is replicated at additional sites. This can be seen by comparing Figures 3 and 4, where the record is replicated at five sites in the latter figure. Although it has not been shown, the misroute probability actually decreases as the traffic load is lower than our settings of 90% occupancy. This is so because the waiting time, thus the vulnerable period, is shortened when the load is decreased.

If certain customers have a high call throughput and a small R_q due to some unexpected reasons, the high probability of misroute indicates that it may not be desirable to replicate their records at multiple sites. However, it may still be worthwhile to do so to improve the availability of records in case of partial system failures. This can be achieved by replicating these records at the PS and at a very few SS's with most or all of queries processsed by the PS at normal conditions. As shown in Figure 5 where record A is replicated on the PS and one single SS, the misroute probability under the PWP can be significantly reduced by sending only a small fraction of the queries to the SS. In this case, the use of PWP for the record replication eliminates much of the overhead incurred in other concurrency control protocols.

Although these numerical results are based on a particular set of parameters, the trend of the results and their implications are expected to remain valid for other parameter settings as well as other service scheduling disciplines.

4. CONCLUSIONS

To improve access delay and system availability, customer data for call routing and other signaling functions can be distributed and replicated at multiple sites. Since telecommunication services can tolerate calls to access obsolete information at various sites for a short period of time, we have proposed the PWP as an efficient concurrency control and commitment protocol for updating the replicated databases. The probability of call misroute for the PWP has been analyzed. Our numerical results reveal that the misroute probability is small for the expected customer behavior in the PCN, wireless networks and AIN.

ACKNOWLEDGMENTS

The author thanks Y. Levy for his suggestion of using the fraction of calls misrouted as a performance measure for evaluating the PWP and for his comments. Thanks are also due to A.B. Bondi for his discussion.

REFERENCES

[1] Y. Arian and Y. Levy, "Algorithms for Generalized Round Robin Routing," *Oper. Res. Letters*, 12, 1992, 313-319.

[2] P.A. Bernstein and N. Goodman, "Concurrency Control in Distributed Database Systems," *ACM Computing Survey*, 13, June 1981, 185-222.

[3] P.A. Bernstein, V. Hadzilacos, and N. Goodman, *Concurrency Control and Recovery in Database Systems*, Addison-Wesley Publishing Co., Massachusetts (1987).

[4] A.B. Bondi and V.Y. Jin, "A Performance Model of a Design for a Minimally Replicated Distributed Database for Database-Driven Telecommunications Services," *Distributed and Parallel Databases*, 4, 1996, 295-397.

[5] S. Ceri and G. Pelagatti, *Distributed Databases Principles and Systems,* McGraw-Hill Inc., New York (1984).

[6] W.W. Chu and J. Hellerstein, "The Exclusive-Writer Approach to Updating Replicated Files in Distributed Processing Systems," *IEEE Trans. on Computers,* C-34, June 1985, 489-500.

[7] R.L. Franks and P.E. Wirth, "UPT Traffic Issues -- An Agenda for the 90's," *Proc. of 8th ITC Specialist Seminar on Universal Personal Telecommunication,* Genova, Italy, Oct. 1992, 107-115.

[8] D.J. Goodman, G.P. Pollini, and K.S. Meier-Hellstern, "Network Control for Wireless Communications," *IEEE Communication Magazine*, Dec. 1992, 116-124.

[9] L. Kleinrock, *Queueing Systems, Vol.I: Theory,* Wiley, New York (1975).

[10] K.K. Leung, "An Efficient Approach to Updating Replicated Databases in Wireless and Advanced Intelligent Networks," *Fifth WINLAB Workshop on Third Generation Wireless Information Networks*, NJ, April 1995, 421-449; also in *Wireless Information Networks: Architecture, Resource Management, and Mobile Data*, J.M. Holtzman (Ed.), Kluwer Academic Publishers, Boston (1996), 379-393.

[11] Y.-B. Lin, "Failure Restoration of Mobility Databases for Personal Communication Networks," *ACM-Baltzer J. Wireless Networks,* 1995, 365-372.

Figure 1a. Query Processing of the PWP

```
    *Query Processing for Record A at the PS or a SS:*
    If the query for record A is the first query of a call, then
        begin
            Access $R_A(vn\_late)$;  (* Access to the latest version of record A *)
            Record the VN, $vn\_late$, for the call;
            $CC(vn\_late) \leftarrow CC(vn\_late)+1$
        end
else begin
          Access $R_A(n)$ that has been accessed by previous queries of the call;
          If the query is the last one of the call, then
              begin
                  $CC(n) \leftarrow CC(n)-1$;
                  If $CC(n)=0$ and $vn\_late > n$, then $R_A(n)$ is deleted
              end
      end
```

Figure 1b. Update Processing of the PWP

```
Update Processing for Record A at the PS:
Process an update;
If the update does not cause data inconsistency (e.g., incomplete routing data), then
    begin   (* Update committed *)
        Create $R_A(vn\_late+1)$ and set $vn\_late \leftarrow vn\_late+1$;
        Record the committed update, $U_{vn\_late}$, in the log;
        If $CC(vn\_late-1)=0$, delete $R_A(vn\_late-1)$;
        Send an update message, $(U_{vn\_late}, vn\_late)$, to all SS's;
        Start a timeout period for the update $U_{vn\_late}$
    end
else  Abort the update and inform the source (* Update aborted *)

Update Processing for Record A at a Secondary Site i :
For an update message $(U_n, n)$, compare $n$ with $vn\_late$ at the site;
If $n > vn\_late+1$ then
    begin     (* Updates arrive out of sequence *)
        Add the out-of-sequence update $(U_n, n)$ to a update list for record A, sorted in
        the ascending order of VN's for future processing
    end
else if $n = vn\_late+1$ then
    begin     (* Update with the correct VN *)
        Process $U_n$, create $R_A(n)$, and set $vn\_late \leftarrow vn\_late+1$;
        Record the committed update, $U_{vn\_late}$, in the log;
        If $CC(vn\_late-1)=0$, delete $R_A(vn\_late-1)$;
        Send an acknowledgement with the VN $vn\_late$ and the site number $i$ to the PS;
        Repeat this procedure to process the first update in the update list for record A if its
        VN equals $vn\_late+1$
    end
else if $n \le vn\_late$ then
    begin     (* Duplicated update which has been previously processed *)
        Send an acknowledgement with the VN $n$ and site number $i$ to the PS
    end
```

Figure 1c. Acknowledgement and Timeout of the PWP

Acknowledgement Processing for Record A at the PS:
If the acknowledgement has been received before, it is discarded;
Record the acknowledgement for update U_n from site i in the log;
If all acknowledgements associated with U_n have been received from all SS's, de-activate the timeout and send a "completion" message to inform the source that all copies of record A have been updated successfully

Timeout Expiration for Record A at the PS:
Resend the update U_n to those sites from which the associated acknowledgements have not been received

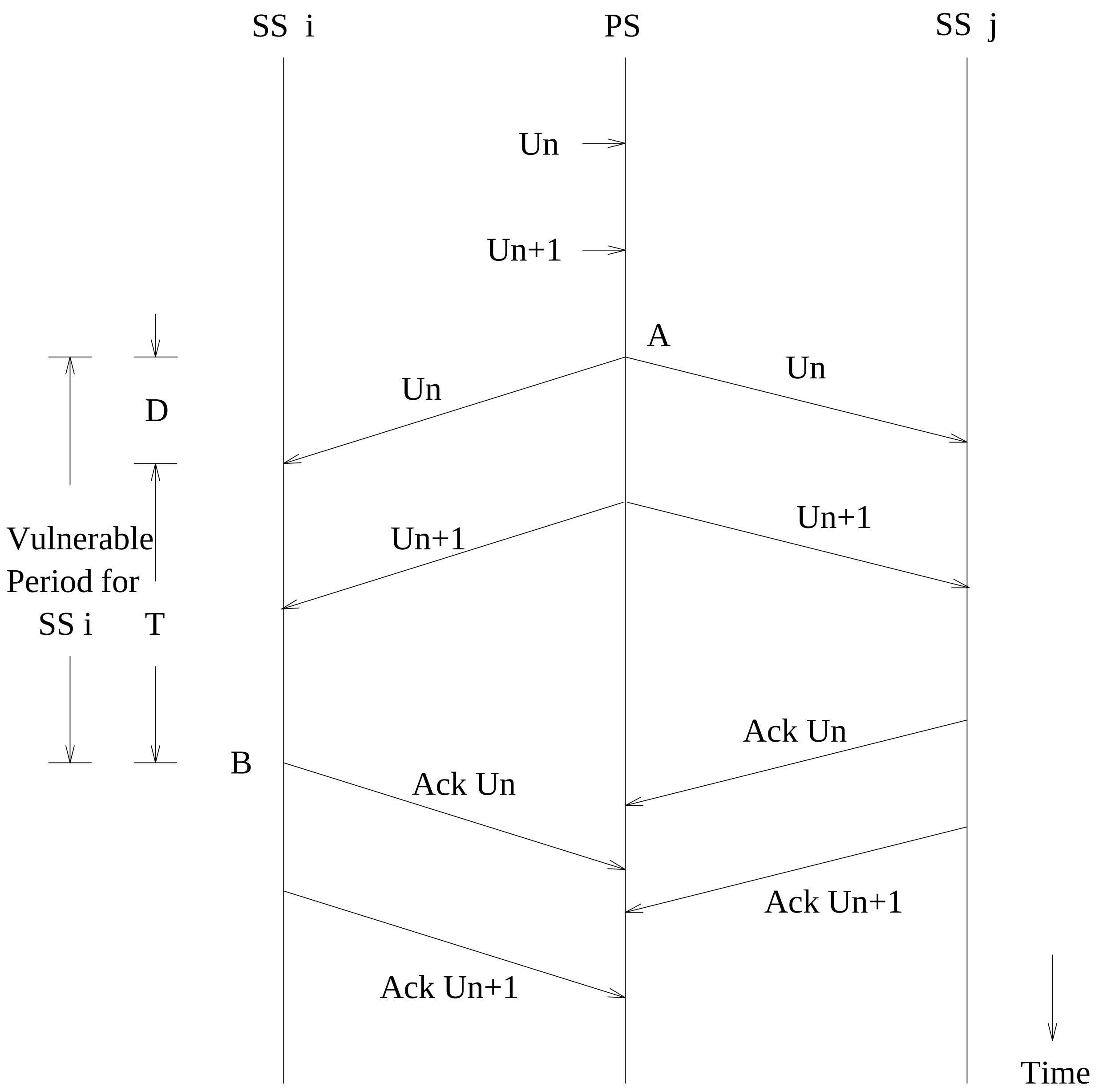


Fig.2. A Timing Diagram for the PWP

Fig.3. Call Misroute for 2-Site Replication

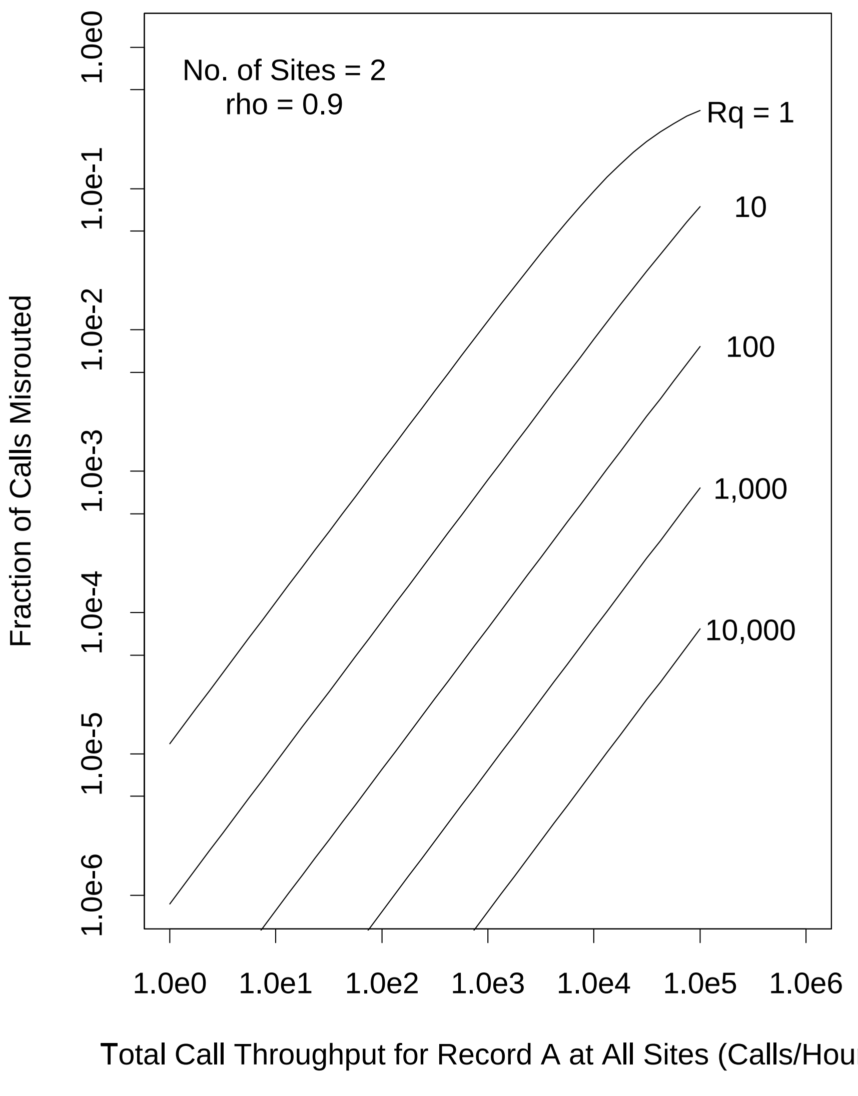


Fig.4. Call Misroute for 5-Site Replication

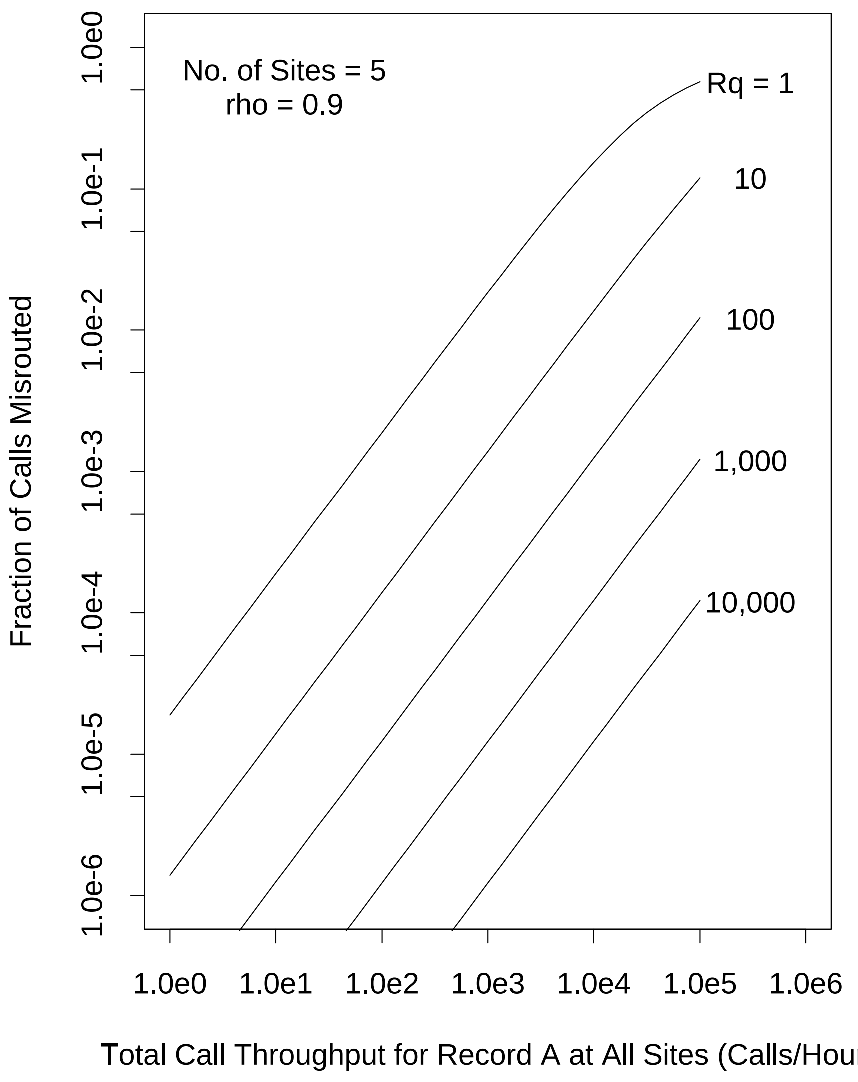
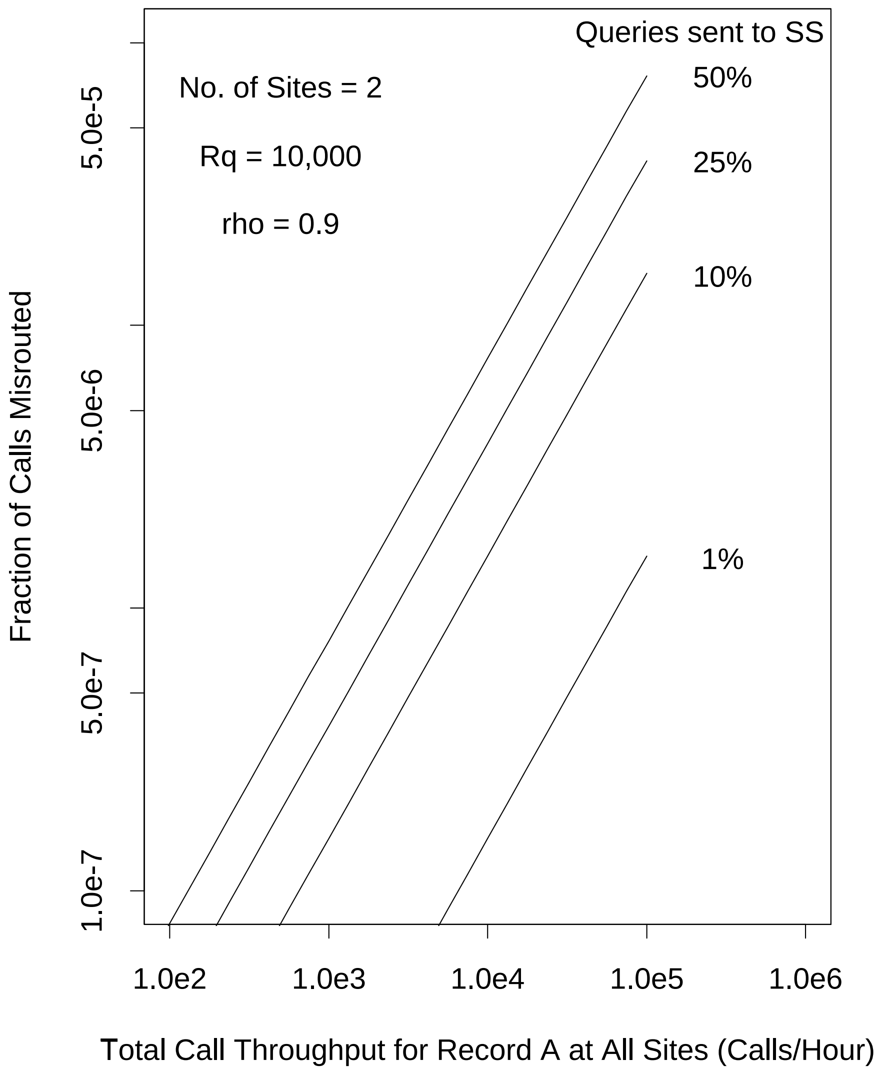


Fig.5. Call Misroute for Non-Uniform Traffic Load


Queries sent to SS
No. of Sites = 2
Rq = 10,000
rho = 0.9
Fraction of Calls Misrouted
5.0e-5
5.0e-6
5.0e-7
1.0e-7
1.0e2
1.0e3
1.0e4
1.0e5
1.0e6
Total Call Throughput for Record A at All Sites (Calls/Hour)