Modelling the Performance of Distributed Database Protocols for Real Time Environments

Alan Allwright[1] & Chris Steketee[2]

[1]Information Technology Division, DSTO (Salisbury), South Australia
[2]Advanced Computing Research Centre, University of South Australia, South Australia

Abstract. Distributed database management protocols are usually based on the assumption that data consistency is paramount. However, this assumption is invalid for distributed real-time applications such as ship combat systems and air traffic control, where data availability is crucial and may need to be achieved at the expense of guarantees of data consistency. Two protocols designed for real-time combat systems which achieve availability by relaxing consistency restraints are ADDAM (ARE (Admiralty Research Establishment) Distributed Database Manager) and ALDM (ADDAM Like Distributed Database Manager). In this paper, the performance of these protocols is compared with the well-known Two-Phase Commit (2PC).

1 Introduction

The protocols described in this report were designed to manage distributed and replicated data. The management of replicated data requires a substantial amount of coordination between the participant processes in addition to the transaction management processing required for non-distributed databases. The data consistency and conflict resolution requirements of many systems are not absolute. In cases where atomicity, durability, serializability and isolation [1] are mandatory a 2- or 3-phase commit may be necessary. In other cases, depending upon distribution reliability, consistency and performance, a range of message passing strategies ranging from Two-Phase Commit (2PC) to reliable message passing may be more appropriate. In order to judge the relative merits of different protocols, the message passing strategy and performance tradeoffs must be considered.

This report provides an overview of the authors' work in assessing message passing and performance tradeoffs for a specific class of real time monitoring and control system, namely combat or air traffic control systems. The remainder of this paper is organised as follows. In Section 2 we present the three distributed database management protocols being studied. The simulation model is described in Section 3, and the performance results are provided in Section 4. Section 5 presents our conclusions.

2 Distributed Database Management Protocols

In this paper the performance of three distributed database protocols, 2PC [1], ADDAM [2], and ALDM [3], is compared.

We assume a database which is horizontally fragmented into pages; pages are replicated for resilience. Each page has one owner process to which updates are directed, and a number of subscriber processes which hold copies of the page. The process responsible for the creation and distribution of a transaction is called the originator. This, in turn is invoked by a higher level application request.

The protocols use timestamping for concurrency control to avoid the possibility of deadlocks associated with locking. Loss of access to data caused by deadlock could have catastrophic consequences for high availability systems such as combat and air traffic control systems. Each page is provided with a timestamp which is essentially a count of the number of updates completed for the page.

In a combat system two types of data must be managed, namely reliable and performance data. Reliable data is generated by operators, the loss of which could compromise operation of the system and requires a distributed database management protocol to ensure its successful distribution. The protocols designed to manage reliable data are described in sections 2.1, 2.2, 2.3. Performance data is generated by sensors such as radar, and is replaced at a high rate so that inconsistencies are short lived. The management of performance data uses a low-overhead protocol with no consistency guarantees. This is described in section 2.4.

2.1 Two-Phase Commit Protocol

Two-Phase Commit [1] is used as a baseline because it is similar to the ALDM and ADDAM protocols, is well researched and has current popularity. In this version of 2PC (Fig. 1) updates are distributed with the originator request at the beginning of the transaction. The message may be either local (between processes on the same node), or remote (between processes on different nodes). Each of the owners checks the pages to be updated for conflicting requests. A conflicting request is detected by comparing the timestamp for the originator's page with the timestamp in the owner's copy of the page. If these are not the same, this indicates there has been an intervening update since the record was read, and the update is rejected. Otherwise, the update is accepted and an accepted response returned to the originator. In the second phase owners either commit or roll back as requested by the originator.

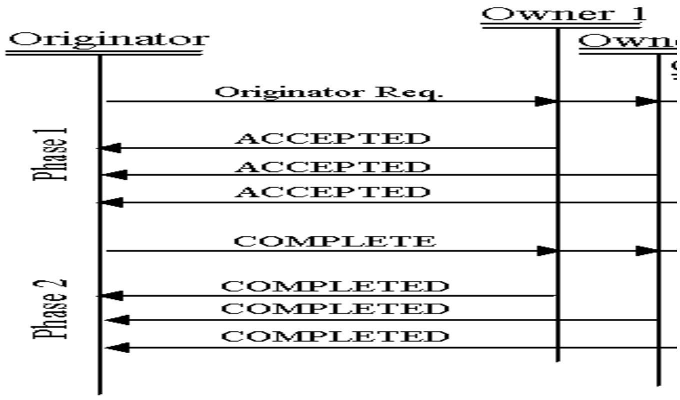

Fig. 1. 2PC Message Passing

2.2 ADDAM Protocol

A feature of 2PC is that transactions can be blocked indefinitely if an owner fails to respond to the originator or a subscriber fails to respond to an owner, whether this is caused by the loss of a message or the failure of a participating node. The ADDAM protocol [2] was devised for combat systems, in which indefinite blocking of transactions cannot be tolerated. It avoids blocking by relaxing consistency constraints - the owner does not wait for subscriber acknowledgments (Fig. 2(a)). Potential inconsistencies arising from this are dealt with by means of an additional background protocol, Subscriber and Owner Monitoring (SOM), which works in conjunction with the ADDAM reliable update protocol to detect and rectify database inconsistencies. This protocol, shown in Figure 2(b), checks for the presence of owners and subscribers, and for consistency between the pages held by subscribers and the pages held by owners. Each subscriber periodically sends a subscriber notification message to each page owner. This message contains the page number and a Page Last Update Number (PLUN). The owner uses the PLUN in the notification to check for inconsistencies. If an inconsistency is detected, a new page is distributed to the subscriber.

2.3 ALDM Protocol

The SOM protocol of ADDAM constitutes an additional overhead which may impair performance. To address this, the ALDM protocol [3] was developed by removing SOM from ADDAM and re-introducing subscriber acknowledgments as in 2PC. However, unlike 2PC, acknowledgments only occur in phase 1, and the owner only waits until it has received some of the acknowledgments, given by a parameter

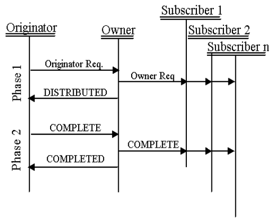

Fig. 2(a). ADDAM Message Passing

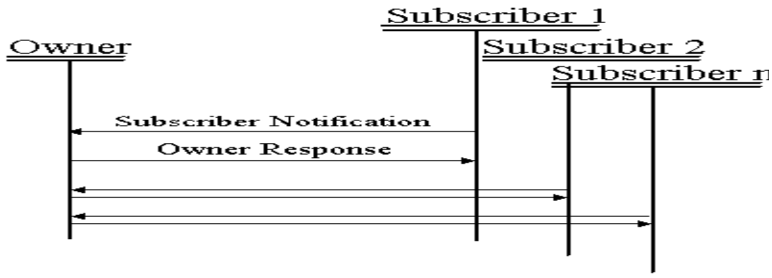

Fig. 2(b). SOM Message Passing

Minimum Number of Subscribers (MNS). This avoids blocking when a subscriber is lost. Originator and owner request messages are used to test for inconsistencies between originators, owners and subscribers. The loss of an owner is detected at the initiation of a transaction by checking for owner responses. The loss of subscribers is detected only when the number of subscriber responses falls below MNS.

2.4 Performance Data Management Protocol

The performance protocol is designed to manage data created by sensors. This data is expected to be periodic with a high data rate. The performance data is used in application programs to construct tracks and trajectory estimates for the objects, such as planes and ships, being tracked. Regardless of the reliable update protocol (ALDM, ADDAM, 2PC), a performance data management protocol is also required. Performance updates are broadcast, do not require acknowledgments (unreliable distribution), do not share data with reliable updates, and are handled independently by each destination node.

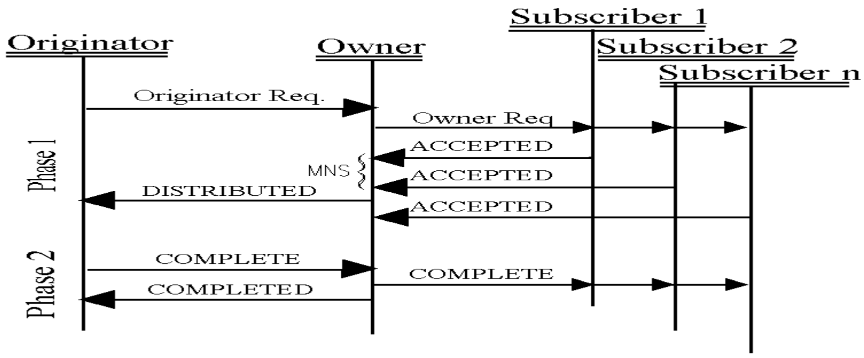

Fig. 3. ALDM Message Passing

3 Simulation Model

The queuing model for the ALDM implementation is shown in Figure 4. In this figure, the originator, owner and subscriber processes are as described in Section 2. The receiver, message handler, sender and transmitter processes are responsible for the distribution and management of internal and network messages.

The ALDM simulation model is based on a testbed implementation of the ALDM protocol. The testbed was used to obtain transaction processing times to calibrate and validate the simulation model. The testbed comprises two or more IBM compatible 386 Personal Computers (PC) and a Fibre Distributed Data Interface (FDDI) [5] Local Area Network (LAN). The ADDAM and 2PC models were built without a testbed implementation, using measurements from the ALDM implementation. The ADDAM, ALDM and 2PC simulation models were developed on IBM compatible PCs using the GPSS/H simulation language.

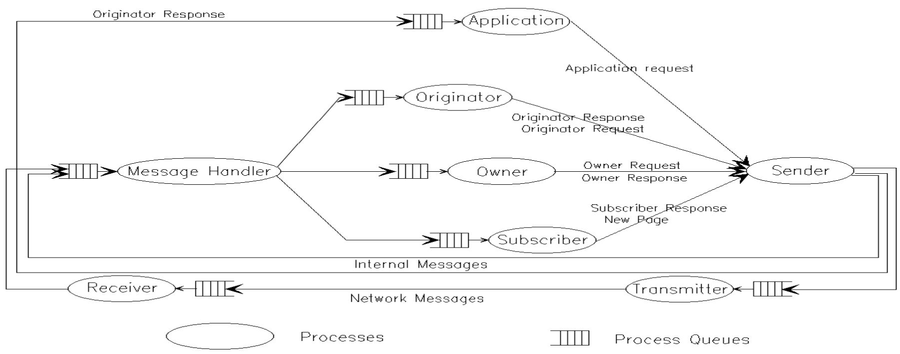

Fig. 4. ALDM Queuing Model

Table 1. Typical Simulated Process Times

Process	Process Time (microseconds)
Application	300
Originator	410
Owner	470
Subscriber	200
Sender	248
Transmit	600
Message Handler	140
Receiver	1200

A detailed description of the simulation model and its validation can be found in Allwright [4].

The database in these applications (combat systems and air traffic control systems) is small (relative to many commercial databases i.e. less than 100 MByte) and is kept in main memory rather than disk. The effects of disk storage could be simulated by introducing delays for data storage and retrieval into the models. Each update is distributed with a separate message. The typical message size is 120 Bytes. Table 1 provides a list of typical process times (in microseconds) for the processes shown in Figure 4.

In the experiments described in this report the network utilisation is always less than 1 Mbps and little network contention was observed. Analysis of the FDDI protocol and measured latency on the testbed suggested network delays would always be less than 10 microseconds.

4 Experiments

The measures of interest in this study were the transaction time and the transaction throughput. The transaction time was measured from the time when a transaction is generated to the time when the transaction response was sent to the application. The time to process a transaction was logged at the completion of each transaction. The transaction throughput is the number of transactions that can be successfully processed per second. A batch oriented sequential procedure [6] was used to ensure results are within the 95% confidence interval used in all experiments.

The experimental parameters in this study were the transaction size, the transaction rate, page replication, and the performance data rate. Ranges for transaction size and replication are loosely based upon experiments conducted on the ADDAM database [7]. These parameters are generally application dependant. For example, to create a record for a new track may require 4 updates (track id, vehicle type, location, hostility), whereas updates to a specific field may require only one update. In these experiments a range of one to ten update transactions was used for sensitivity analysis. Factors such as transaction rate, performance data rate were varied from zero to the limit of the capacity of the simulation models.

The database in each experiment was divided into pages, each page holding ten records. The size of a transaction was the number of updates that are made by a transaction. The transaction rate was the rate at which transactions were generated and submitted to the database for processing. The performance rate was the rate at which performance messages were generated and submitted to the database for processing. The number of replicates is the number of copies of each page (one owner and replicates-1 subscribers).

4.1 Experiment 1 - Page Replication

The effect on transaction time of changing the number of copies of each page was assessed in this experiment. The trends for the protocols for 2 ("2R") and 4 ("4R") page replicates are shown in Figure 5. The database consisted of either 50 (2 Replicates) or 200 (4 Replicates) pages distributed over 5 nodes. Each node owned 10 pages. Transaction sizes were varied between one and ten updates. One node generated transactions. A delay was introduced between transactions to guarantee the transaction was complete before a new transaction was created.

The 2 replicate results show the ALDM transaction times increase faster than the ADDAM or 2PC times. The 2 replicate ADDAM and 2PC have similar times until the number of updates is 4, when 2PC times exceed the ADDAM times. These results are largely a consequence of the message passing requirements discussed earlier. The processing of responses for page replicates is the reason for the differences between the ADDAM and 2PC protocols. For the 2PC protocol, when the number of replicates and updates is small, the processing of replicate responses does not incur a significant penalty. This is because the local updates can be processed completely before any remote responses are received. Consequently 2PC has to wait for and process the same number of remote responses as the ADDAM protocol. The ALDM times increase substantially faster than 2PC and ADDAM times because the ALDM has an extra wait for subscriber responses. The length of the wait depends on the distribution of page owners and subscribers.

Transaction throughput was measured for the same set of transaction sizes and page replication. The throughput in all cases had an inverse relationship to the transaction time. The following function (1) was fitted to the measured throughput for all the data in Figure 5:

$$\text{Throughput} = 1/(\text{Xact\_time} + 0.001185\ \text{Size} + 0.00405) \qquad (1)$$

In all cases the transaction throughput was estimated to within 1% of the measured value. The transaction time (Xact_time) is expressed in seconds and the throughput

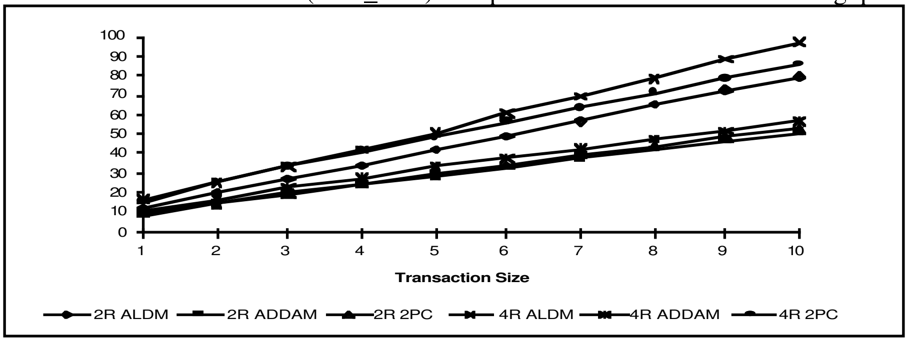

Fig. 5. 2 and 4 Replicate Composite Transaction Time Comparison

is expressed in transactions per second (Xact/sec). The fitted throughput is consistent with the way in which transactions are generated and processed. Each transaction requires a set up time and a process time. The set up time is the time required for the application to communicate the details of the transaction to the originator. The 0.00405 second delay represents the wait time between transactions. The 0.001185 second delay represents the communication time between the application and the originator.

4.2 Experiment 2 - Subscriber and Owner Monitoring

The effect of the Subscriber and Owner Monitoring (SOM) protocol on the ADDAM transaction processing time was investigated in this experiment. The SOM protocol would always be used in conjunction with the ADDAM protocol. Because there is a range of effects that the SOM processing can have on the ADDAM transaction processing performance, it was investigated as a separate experiment. The effects of the SOM processing must always be taken into account when assessing the ADDAM transaction processing performance in the other experiments.

The database in this experiment consisted of 100 pages distributed over 10 nodes. Each node owned 10 pages. Transactions were generated one at a time by one node only. The notification period was varied from 2 seconds to 200 ms in decrements of 200 ms. The process times in this case were 345, 185 and 470 microseconds for the subscriber notification, owner response and the subscriber processing respectively. All other process times, i.e. message send and receive are shown in table 1.

The transaction processing times are affected by the scheduling of subscriber notifications. Three scheduling schemes are assessed; Bursty, Local and Global. The bursty schedule allows all subscribers to distribute subscriber notifications at much the same time. The number of messages and the resulting load on transaction processing is directly proportional to the number of subscribers. A burst of subscriber notifications results in a large increase in contention for owner processes over a short period of time. For the local schedule the subscribers are provided with a specific time to broadcast the notification for each page on each node. The global schedule extends the local scheduling mechanism by only allowing a single notification at a time.

The results for the mean transaction times (mean) and scaled (10 times) standard deviations (10 X SD) are shown in Figure 6. The results are shown for the 3 subscriber notification scheduling techniques (Bursty, Local and Global) for 5 update transactions and 4 page replicates. The overall trend is of a substantial increase in transaction times and standard deviations as the subscriber notification period is decreased. The reason for the trend is the processing load for subscriber notification messages. The trend is similar to the trends observed for processing performance messages described in experiment 4. The results suggest the scheduling does not effect the mean transaction time but it does effect the transaction standard deviations, with the largest standard deviation occurring for bursty scheduling.

4.3 Experiment 3 - Transaction Loading

In this experiment the effects of varying transaction throughput on transaction times were investigated. The database consisted of 10 nodes each with a maximum of 8000 pages. Pages had either 2 (2R) or 4 (4R) replicates per page. Each transaction had 4 updates. The transaction Inter-Arrival Times (IAT) were Poisson distributed, with mean IAT varying from 360 ms to 4520 ms per node. Figure 7 shows the transaction times for the 3 protocols against the change in transaction IAT.

The trends show a dramatic increase in transaction times as the IAT falls below one second for all protocols. It can be seen the 2 replicate ADDAM and 2PC protocols provide the smallest transaction times, the 4 replicate ALDM provides the highest. This is the same order of transaction times observed in the previous experiments. In each case the level of variation in transaction times increases quickly with the decrease in IAT. The maximum transaction times were obtained during periods of peak contention resulting from a large number of transactions in a short period of time.

As the transaction IATs were reduced the length and intensity of these periods of peak contention increased thus resulting in increasing maximum transaction times. The minimum transaction times at long transaction IATs are expected to converge to those determined from experiment 1. The minimum transaction times were always constant across the range of IATs. It was determined that the processes were essentially regenerative, that is conditions representative of the initial start up appeared periodically. The minimum transactions times occurred at random intervals. The length of the interval appeared to be related to the mean transaction

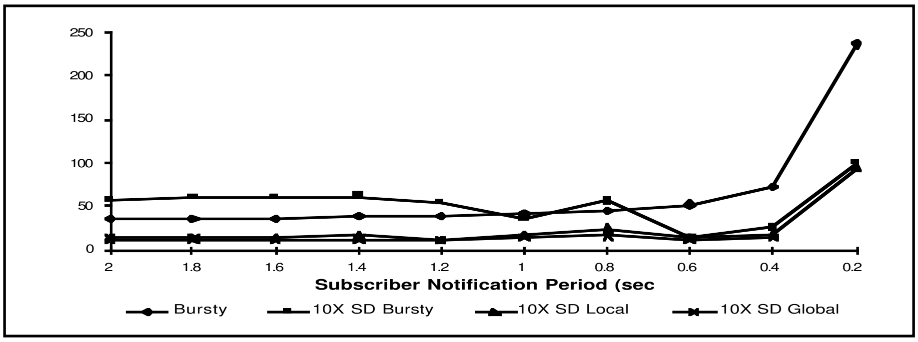

Fig. 6. SOM Scheduling Comparison - Transaction Time

IAT, the transaction size and the number of page replicates. As the transaction time increased, the frequency of regeneration decreased.

4.4 Experiment 4 - Performance Loading

The effects of performance updates on reliable update performance were investigated in this experiment. The experimental setup was the same as that used in Experiment 3 (Transaction Loading). Figure 8 shows the transaction processing time for each protocol using 3-update transactions as the performance data rate is increased. Transaction processing time estimates were calculated using Equation (2). These trends at higher performance message rates can be explained in terms of the performance messages starving the transactions of processor time. Because performance messages are broadcast to all nodes, transaction processing is essentially halted whilst performance data are being processed. To account for this effect, a Performance Utilisation factor (PU) can be calculated. This has 3 components - a fixed overhead to process the message, a message copy time to copy the message from the hardware buffer on the network card to node processor main memory and a variable process time for the message. In this case the process time is approximately 1340 microseconds per performance message which is composed of the receive and message handler simulated process times shown in table 1.

The PU is calculated as the total time per second that processing of performance messages utilises the processor (Performance Message Rate * 1340 microseconds). The processor free factor is 1 minus the PU. The scaling factor is the inverse of the processor free factor. This scaling factor is multiplied by the transaction accept time for zero performance data rate (Tap0) to give the transaction accept time for other performance data rates (Tapi). The estimated transaction time (est) using this formula is shown on Figure 8, and agrees closely with the results of the simulation experiments.

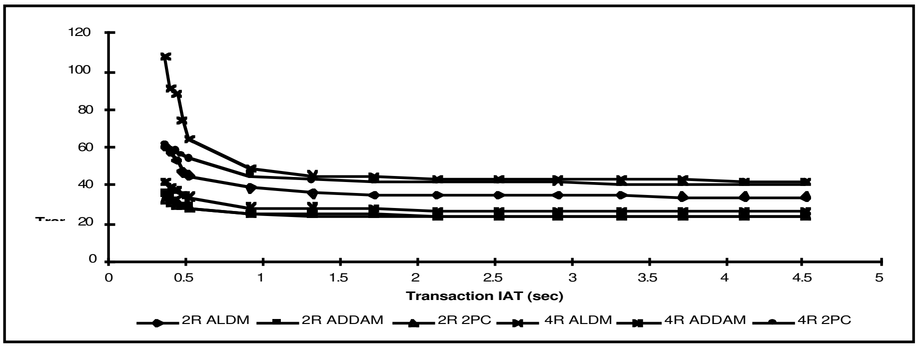

Fig. 7. ALDM, ADDAM, 2PC Comparative Transaction Times

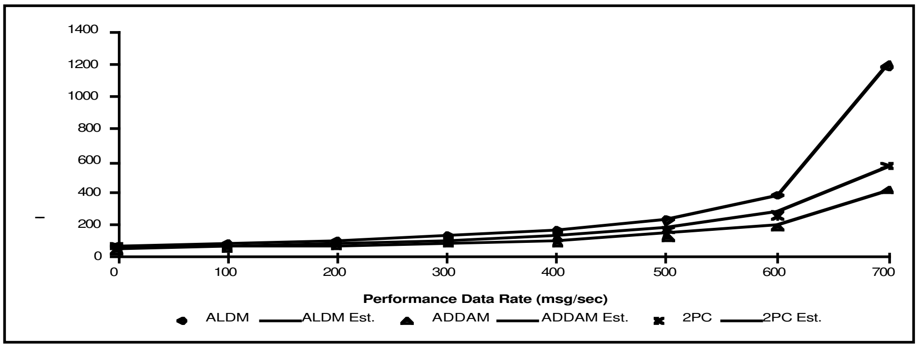

Fig. 8. Effect of Performance Data

$$T_{api} = T_{ap0} * 1 / (1 - PU) \quad (2)$$

5 Conclusion

This work has investigated the steady state performance of a number of real-time distributed database protocols. Two of the protocols, ALDM and ADDAM, were designed specifically for combat systems. These protocols also have application in other systems where data availability and high transaction throughput are principal requirements. The 2PC protocol was also investigated as a baseline for performance comparisons. It has been shown that the policy used in the design of the ADDAM protocol can be implemented to provide better transaction performance than conventional protocols. The relatively poor performance of the ALDM protocol and the effects of subscriber and owner monitoring on the ADDAM protocol result largely from the message passing requirements of the protocols. Alternative message passing strategies that should improve the transaction processing performance of the ALDM and ADDAM protocols are discussed in Allwright [4].

Central to the theme of this paper is that failures will occur in computer systems. Database inconsistencies that result from these failures must be resolved in an efficient and effective manner. In this paper we have described two protocols (ADDAM, ALDM) that can improve the availability of the database system by allowing the database to operate with transient inconsistencies. In current and highly reliable environments network and node failures are much less frequent than when 2PC was first proposed. The ADDAM and ALDM protocols more than any conventional protocols take advantage of the increased reliability.

In summary this work has shown that by reducing the stringent consistency criteria supported by conventional protocols significant performance gains can be achieved. The reduction of consistency testing in highly reliable environments does not lead to significant increases in the frequency of database inconsistencies.

6 Acknowledgments

The authors wish to acknowledge the assistance provided by Dr. David Panton at the University of South Australia, and Mr Jeff Schapel at the Defence Science and Technology Organisation (DSTO), Salisbury South Australia.

7 References

1. Ceri, S., (1984), Pelagatti, G., Distributed Databases : Principles and Systems, Computer Science Series, McGraw-Hill.
2. Reilly, M. (1985),Tillman, P.R., Maintaining Consistency and Accommodating Network Repair in a Self Regenerative Distributed Database Management System, AGARD-CP-360, (Conference Proceedings, Cost Effective and Affordable Guidance and Control Systems).
3. Miller, S.J. (1991), A Distributed Database Manager based on ADDAM, WSRL-TM-58/91 (Weapon Systems Research Laboratory, Technical Memo).
4. Allwright, A.M. (1995), Performance Analysis of Distributed Database Protocols for Combat Systems. Masters Thesis, School of Computer and Information Science, University of South Australia.
5. Ross, F.E. (1989), An Overview of FDDI: The Fiber Distributed Interface, IEEE Journal on Selected Areas in Communications, Vol. 7, No. 7, pp 1043-1051.
6. Law, A.M. (1979), Carson J.S., A Sequential Procedure for Determining the Length of Steady-State Simulations, Operations Research, Vol. 27, No. 5, pp 1011-1025.
7. Murrell, J.G. (1985), ADDAM Performance Measurements Final Report, DIAS D473T/TR14, (Distributed Information Architecture for Ships, Defence Research Agency, UK)