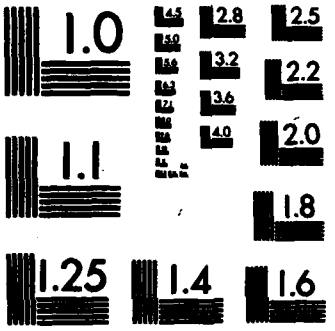
AD-A135 320 RESEARCH IN CONTINUITY OF OPERATIONS IN DISTRIBUTED TACTICAL INFORMATION SYSTEMS(U) PEGASUS SYSTEMS SUMMIT NJ D E PERRY 25 OCT 83 ARO-18622.1-EL-S 1/1

UNCLASSIFIED DAAG29-82-C-0014 F/G 15/7 NL

END

FILMED

DTIC



(12)

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

ARO 18622.1-EL-S

A135320

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. A135320 / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Research in Continuity of Operations in Distributed Tactical Information Systems. Final Report	5. TYPE OF REPORT & PERIOD COVERED Final 1 June 82 - 31 May 83
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Dewayne E. Perry	8. CONTRACT OR GRANT NUMBER(s) DAAG 29-82-C-0018 14
9. PERFORMING ORGANIZATION NAME AND ADDRESS Pegasus Systems Summit, NJ 07901	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS U. S. Army Research Office Post Office Box 12211 Research Triangle Park, NC 27709	12. REPORT DATE 25 October 1983
	13. NUMBER OF PAGES 15
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)	15. SECURITY CLASS. (of this report) Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

The view, opinions, and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Distributed Systems, Recovery, Reconfiguration

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report is the final report for the first year of a projected 3 year effort on the continuity of operation in distributed information systems. We define a system model abstracted from a "real" battlefield information system, define simplifying assumptions to limit and focus the level of research, and present some preliminary results with respect to global system state recovery by means of an example common to recovery and reconfiguration.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Research in the Continuity of Operations In Distributed Tactical Information Systems

FinalReport

Dewayne E. Perry

25 October 1983

U. S. Army Research Office

Contract No. DAAG29-82-C-0014

Pegasus Systems

Table of Contents

1. Introduction

This final report represents the first year effort in a projected three year research effort into the continuation of operation in distributed tactical information systems. The principle investigator is Dr. Dewayne E. Perry of Pegasus Systems and the work is sponsored by the Army Research Office under contract DAAG29-82-C-0014 for the period 1 June, 1982 through 31 May 1983.

2. Research Goals

In this research project, we are investigating the techniques and tools required to build reliable systems that provide continuity of operations despite component failure. Specifically, we are interested in software solutions to the problems on reliability, recovery and reconfiguration.

The basic goals of this research, over the three year effort are summarized as follows:

- To identify and refine the fundamental issues and the relationships between these issues in the construction or survivable distributed systems;
- To survey existing approaches to these various issues, to determine which are suitable for survivable systems, and to explore their interaction with each other;
- To determine in levels of performance by there various design and implementation strategies; and
- To establish and design approach applicable to survivable distributed tactical information systems.

The basic underlying issue is the extent to which the goal of survivability places special requirements upon the various aspects of system design and construction. Are there existing techniques that are satisfactory within the constraints of acceptable system performance or must new techniques be developed to solve the particular needs of continious operation?

The goals of this research is explored though the use of one or more models of actual tactical information systems. These models are designed in various stages in order to fully understand the various solution strategies, their implications and their cost. While these models represent "real" systems the emphases is upon computer science issues rather than on application related issues.

The Ada language will be used to specify, design and construct these various system models in sufficient depth to expose the relevant probelms of the design and, where they occur, the inabilities of Ada to handle the problems of distributed system design and construction.

The goal for this first year of effort are as follows:

- define a model system abstract from a "real" field situation;

- within the context of this model, identify the simplifying assumption are to be made in order to restrict the complexity of the problem and control the interaction between various issues; and
- address the issues of redundancy, locus on control and reconfiguration.

The effort with respect to the third goal has centered primarily on issues of recovery.

3. Summary of Results

First, the system model has been defined as an abstraction or a battlefield information system: a set of battalions with a set of forward observers. This model has all the requirements needed for a framework for considering the survivability of a distributed information system:

- distributed processors (the battalions of the model);
- distributed resources (the forward observers); and
- distributed information (the battalion database).

A discription of the model is to be found below.

Second, the simpifying assumptions have been delinated that restrict the focus of the investigation to the appropriate level of detail and that support the issues under investigation. These assumptions have been made in order to concentrate our attention on the problems of recovery and reconfiguration, in response to component failure. These assumptions are presented with the system model in the discussion below.

Finally, we have focused our attention specifically in the problem of global system recovery and present a methodology for determining, and recovering the system to, the most recent consistent global state. An example common to recovery and reconfiguration, that of transferring databases for backup and repartitioning purposes, is given to illustrate the methodolgy.

4. Research Results

The results of this research are discussed in the next two subsections. First, we present the sysem model and accompanying simplifying assumptions. Second, we discuss the problems of global recovery and present a methodology for determining global recovery states in distributed, cooperating processes.

4.1. The System Model

The system model described below is abstracted from a "real" military system: a set of battalions together with a set of forward observers which serve as the input sources; the output of the systems can be ignored with loss of generality. The battalions are independent components, dispersed geographically, that cooperate

with each other in gathering tactical information and that pursue individual as well as global goals. The forward observers serve as the source of the tactical information.

By abstracting away the application dependent aspects of the system, we define the following system model for our research. The model consists of

- a set of autonomous, homogeneous nodes,
- a set of remote input sources,
- a database of information, and
- a fully connected network.

We will discuss the simplifying assumptions with results to these model concerning the following issues:

- node behavior
- remote input behavior
- database characteristics
- communications
- local storage
- redundancy
- recovery

The nodes in the system may enter and leave the system. It may be use useful to distinguish two kinds of node removal from the system: planned and unplanned. In the first case, the system can a priori provide the temporary coverage within the system for that node. (ie, a basic minimun reconfiguration that allows for node reentry later). In the second case, the worst case is assumed and the system recovers by reconfiguring around the missing node. Whether reentry into the system causes reconfiguration is application dependent - we assume it does cause reconfiguration.

Remote input sources behave in a manner similar to that of the nodes - they enter and leave the system. The system provides a mapping of remote input sources into nodes that is reconfigurable dependent upon the behavior of the nodes and of the remote input sources. Remove input sources report to the system and the system provides the mapping dynamically.

We assume that there neither malicious nodes nor malicious remote input sources. Further, we assume that

each node detects its own failure and recovers to a consistent state. Local recovery - that is, without reconfiguration - maybe possible for this state of affairs.

The basic goal of the system with regard to the data base is no loss of data or information. For the sake of simplicity, we assume that the data base is partitioned. Each node is responsible for its own data base. Further, the mapping of remote input is not done to nodes per se, but to the partitioning of the data base - and set of remote input sources is mapped to a particular partition of the database. Replication of the data base is considered from the perspective of backup purpose only. The rationale for this approach is that we are not concerned with accessing the data but only with maintaining the completness of the data. Each partition then is replicated on some other node.

It is in the area of communications that we make the stringent assumptions because we wish to concentrate on the higher level issues rather than on the mechanics of the connection between components. First, we assume that the communication is fault free - either the communication completes correctly, or it does not complete at all (and this latter condition occurs only if the destination node is unreachable). Second, the network is adaptive in such a way that all nodes agree as to the faults in the system and as to the nodes connected to the system. Third, notificationoccurs when a node or set of nodes is not reachable. Finally, we assume that packet radio is used in order to provide the basis for performance determinations.

Local storage for each node is assumed to be stable in the sense of [Lampson 76]. As a result of stable storage and fault-free communication we can assume there are no consistency problems with respect to the data. The only problems are those due to latency of data between nodes.

As previously mentioned, the data base is fully replicated by means of a single copy backup. In order to focus our attention on the most basic interaction, we make the (extremely simplifying) assumption that the partitioning of the data base is such that the source and backup copies will not both be lost in a single recovery/reconfiguration cycle.

For recovery, we assume that only one node will disappear at a time and that there is sufficient time between disappearances for the recovery and reconfiguration to take place. Later, we will relax this assumption. Further, we assume that a director of recovery and reconfiguration is selected to direct the repartitioning and reconfiguration of the system.

With these assumptions, we have the basic framework to investigate the issues of recovery at a relatively high level in order to examine the interactions between nodes.

4.2. Recovery

Some of the basic problems that occur in a distributed system that result in changes to the system, or global state, are as follows:

- a node fails,
- a node enters the system,
- the interconnection between nodes fails
- the system is partitioned

The traditional notion of recovery - returning to a previously consistent state - is not entirely applicable here. It may be impossible to return the system state to a previous condition. The error conditions or the causes of the error conditions may preclude such a return. It is the very disintegration of the global state that may be the cause of the problem in a distributed system. Particularly since the connections between components and the components themselves that comprise the global state may fail.

It is for this reason - the disintegration of the global state - that backward recovery may not be a viable recovery mechanism and that we turn our attention to forward recovery considerations. Recovery in a distributed system under these conditions is more a matter of reconfiguration and the establishing of a new system state. The basic goal, however, is still to maintain a consistent global state with regard to the data base.

A basic part of the recovery by reconfiguration is concerned with repartitioning the database. When a node fails, the data maintained by the failed node needs to be integrated into another node and the remote input sources remapped to that node. We will assume for the time being that the partition is kept intact, primarily because repartitioning multiplies the amount of work for reconfiguration and we want to look at the simplest case first.

One of the major concerns in this reconfiguration is the movement of a new backup copy of the database to the new backup node. This involves the participation of three nodes: the initiator node, i.e., the director of the recovery; the source node of the backup file; and the destination node where the partition is now to reside, either as a backup or as integrated into the nodes database. The following represents the interaction between the nodes.

Initiator	connect to	Source	
Initiator	-->	Source	: send database to dest
Source	connect to	Destination	

Source	-->	Destination	: request to send database
Destination	-->	Source	: ready to receive database
Source	-->	Initiator	: ready to send database
Source	-->	Destination	: database
Destination	-->	Source	: database installed
Source	disconnect from	Destination	
Source	-->	Initiator	: database installed
Initiator	disconnect from	Source	

The interaction of the three nodes can be seen in the following state description of the three nodes simultaneously. Where the states proceed concurrently, the nodes can proceed concurrently. Where there is no processing indicated, the node is waiting for synchronization on the next non-empty global state. The global states are separated by lines; global state transitions that represent global recovery points are indicated by asterisks.

state	Initiator	Source	Destination
1	decide recovery plan		
2	connect to S wait for accept	wait for connect accept connect	
3	send request to S	receive request	
4		connect to D wait for accept	wait for connect accept connect

5		send request to D	receive request
6		prepare database	
7		send database	receive database
8			install database
9		receive installed	send installed
10		disconnect from D	
11	receive db installed	send db installed	
12	disconnect from S		

In the normal case, the global state transition occurs in parallel with interacting processes proceeding concurrently on the basis of the message transmission and reception. A send does not complete until the message has been received. Non-interaction processes proceed directly to their next non-empty global state.

If a connection is broken and then resumed, the two interacting processes must determine the the most recent state to reinstate. The usual case is to recover back to the earliest of the two processes state and proceed from there. In this case, however, one of the processes may be able to establish a more recent state as the effective state because it represents the system's global state where it is most important - in this case, with respect to the database.

When connections are broken, and connection is restablished, there are several possible courses of action.

First, the recovery of the global state may only affect the two processes involved. In this case, no further recovery is needed but that needed to resolve the appropriate state of the system. Second, the recovery may affect other processes involved in the operation and thus a more global recovery action is needed.

For example, once state 4 has been reached, there is no need to recover the cooperating processes to a point earlier that that if the connection between the Source and The Destination is broken unless the Source node must recover to a state prior to state 4. If the Source node fails and recovers to global state 8 but the destination process is state 10, there is no need for the both the Source and the Destination nodea to recover back to state 8 when state 10 will maintain the consistency of the recovery plan - it already has installed the database.

The net result of this kind of analysis is that recovery can move forward to a later state (that is the most recent global recover state), rather than back to an earlier state, under certain conditions.

In summary, the following considerations result from this methodological approach.

- There are identifiable global states that represent either the completion of interaction between two cooperating processes (or nodes) or that represent important events in one of the processes that can be considered the point event (for example, the reception of the data base, or the installation of the database).
- An important question is how to represent and distinguish local and global states. For example, a group of local states might be grouped together to represent an important global state with respect to the other cooperating processes.
- Synchronization points are expressed by message exchanges and represent important transitions in the global state.
- The dichotomy between local and global state changes can be expressed in parallel for the cooperating processes. Even they are distinct events, they do occur in parallel and can be represented that way.

4.3. Summary

The preceding discussion represents a preliminary investigation into the problems of recovery in distributed systems. A system model has been constructed and restrictions placed on it to narrow the focus of the research. The approach to determining forward recovery points is an initial look at ways of reconstructing the global state of a distributed system when the global state disintegrates because of node failure.

References

[Anderson 76] T. Anderson and R. Kerr.
Recovery Blocks in Action.
In *Proceedings of the International Conference on Software Engineering*, pages 444-457. 1976.

[Anderson 78] T. Anderson, P. A. Lee and S. K. Shrivastava.
A Model of Recoverability in Multilevel Systems.
IEEE Transactions on Software Engineering SE-4(8):486-494, November, 1978.

[Anderson 79a] T. Anderson, P. A. Lee and S. K. Shrivastava.
System Fault Tolerance.
In *Computing Systems Reliability*, pages 153-210. Cambridge University Press, 1979.

[Anderson 79b] T. Anderson and P. A. Lee.
The Provision of Recoverable Interfaces.
In *The Ninth Annual International Symposioum on Fault-Tolerant Computing*, pages 87-94. Madison, Wisconsin, June, 1979.

[Andler 81] S. Andler, et al.
System D: a Distributed System for Availability.
Technical Report RJ3313, IBM Research Laboratory, San Jose, CA, November, 1981.

[Andrews 81] G. R. Andrews.
SR: A Language for Distributed Programming.
Technical Report TR 81-14, The University of Arizona, October, 1981.
Department of Computer Science.

[Bhargava 81] B. Bhargava.
Software Reliability in Real-Time Systems.
In *AFIPS 1981 National Computer Conference*, pages 297-309. AFIPS, May, 1981.

[Bjork 73] L. A. Bjork.
Recover Scenario for a DB/DC system.
In *1973 Proceedings of the ACM*, pages 142-146. ACM, 1973.

[Blasgen 79] M. W. Blasgen, et al.
System R: an Architectural Update.
Technical Report RJ2581, IBM Research Laboratory, San Jose, CA, July, 1979.

[Carter 79] W. C. Carter.
Fault Detection and Recovery Algorithms for Fault Tolerant Systems.
In *Euro IFIP 79*, pages 725-734. IFIPS, 1979.

[Clarke 82] E. C. Clarke and C. N. Nikolaou.
Distributed Reconfiguration Strategies for Fault-Tolerant Multiprocessor Systems.
IEEE Transactions on Computers C-31(8):771-784, August, 1982.

[Coffman 81] E. G. Coffman Jr, E. Gelenbe and B. Plateau.
Optimization of the Number of Copies in a Distributed Data Base.
IEEE Transactions on Software Engineering SE-7(1):78-84, January, 1981.

[Davies 73] C. T. Davies.
Recovery Semantics for a DB/DC System.
In *1973 Proceedings of the ACM*, pages 136-141. ACM, 1973.

[Denning 76] P. J. Denning.
Fault-Tolerant Operating Systems.
Computing Surveys 8(4):359-389, December, 1976.

[Durham 81] Ivor Durham.
The Design and Construction of Reliable Software for Distributed Computing.
March 1981.
Thesis Proposal. Carnegie-Mellon University.

[Garcia-Molina 82]
H. Garcia-Molina.
Using Semantic Knowledge for Transaction Processing in a Distributed Database.
June, 1982.
Department of Electrical Engineering and Computer Science.

[Gracia-Molina 80]
H. Garcia-Molina.
Reliability Issues for Completely Replicated Distributed Databases.
In *CompCon 1980*, pages 442-449. IEEE, Fall, 1980.

[Gray 78] J. Gray.
Notes on Data Base Operating Systems.
Technical Report RJ2188, IBM Research Laboratory, San Jose, CA, February, 1978.

[Gray 79] J. Gray, et al.
The Recovery Manager of a Data Management System.
Technical Report RJ2623, IBM Research Laboratory, San Jose, CA, August, 1979.

[Hecht 76] H. Hecht.
Fault-Tolerant Software for Real-Time Applications.
Computing Surveys 8(4):391-407, December, 1976.

[Jensen 80] E. Douglas Jensen.
Decentralized Control.
In *Distributed Systems: An Advanced Course.* Springer-Verlag, New York., 1980.

[Kim 80] K. H. Kim.
An Approach to Secure Design of Distributed and Reconfigurable Real-Time Computer Systems.
In *Distributed Data Acquisition, Computing, and Control Symposium*, pages 69-75. December, 1980.

[Krishnarao 78] T. Krishnarao.
A Systematic Design and Analysis of Reconfigurable Distributed Conputer Systems.
PhD thesis, University of California, Berkely, August, 1978.

[Lamport 78] L. Lamport.
The Implementation of Reliable Distributed Multiprocess Systems.
Computer Networks 2:95-114, 1978.

[Lampson 76] B. Lampson and H . Sturgis.
Crash Recovery in a Distributed Data Storage System.
Technical Report, Xerox PARC, 1976.

[Le Lann 79] G. Le Lann.
An Analysis of Different Approaches to Distributed Computing.
In *First International Conference on Distributed Computing Systems*, pages 222-232. Huntsville, Alabama, October, 1979.

[Le Lann 81] G. Le Lann.
A Distributed System for Real-Time Transaction Processing.
In *14th Hawaii International Conference onn System Science.* Honolulu, Hawaii, January, 1981.

[Levin 77] R. Levin.
Program Structures for Exceptional Conditional Handling.
PhD thesis, Carnegie-Mellon University, 1977.

[Linden 76] T. A. Linden.
Operating System Structures to Support Security and Reliable Software.
Computing Surveys 8(4):409-445, December, 1976.

[Lindsay 79] B. G. Lindsay, et al.
Notes on Distributed Databases.
Technical Report RJ2571, IBM Research Laboratory, San Jose, CA, July, 1979.

[liskov 79] B. Liskov.
Primitives for Distributed Computing.
In *Seventh Symposiou on Operating System Principles*, pages 33-42. ACM SIGOPS, December, 1979.

[McDermid 81] J. A. McDermid.
Checkpointing and Error Recovery in Distributed Systems.
In *The 2nd International Conference on Distrib ed Computing Systems*, pages 271-282. April, 1981.

[Ousterhout 82] J. K. Ousterhout.
Medusa. A Distributed Operating System.
UMI Research Press, 1982.

[Randell 75] B. Randell.
System Structure for Software Fault Tolerance.
IEEE Transactions on Software Engineering SE-1(2):220-232, June, 1975.

[Randell 78] B. Randell, P. A. Lee, P. C. Treleaven.
Reliability Issues in Computing Systems Design.
Computing Surveys 10(2), June, 1978.

[Randell 79a] B. Randell.
System Reliability and Structuring.
In *Computing Systems Reliability*, pages 1-18. Cambridge University Press, 1979.

[Randell 79b] B. Randell.
Software Fault Tolerance.
In *Euro IFIP 79*, pages 721-724. IFIPS, 1979.

[Ryan 80] T. M. Ryan.
Backup and Recovery for Distributed Interactive Computer Systems.
In *Proceedings: Distributed Computing. CompCon80*, pages 101-107. IEEE Computer Society, September, 1980.

[Svobodova 79a] L. Svobodova, B. Liskov, D. Clark.
Distributed Computer Systems: Structure and Semantics.
Technical Report MIT/LCS/TR-215, Laboratory for Computer Science, MIT, March, 1979.

[Svobodova 79b] L. Svobodova.
Reliability Issues in Distributed Information Processing Systems.
In *The Ninth Annual International Symposioum on Fault – Tolerant Computing*, pages 9-16. Madison Wisconsin, June, 1979.

[Williams 81] R. Williams, et al.
R: An Overview of the Architecture.*
Technical Report RJ 3325, IBM Research Laboratory, San Jose, CA, December, 1981.

END

FILMED

1-84

DTIC