GEORGIA INSTITUTE OF TECHNOLOGY
OFFICE OF CONTRACT ADMINISTRATION
SPONSORED PROJECT INITIATION

Date: October 17, 1979

Project Title: Research on Fully Distributed Data Processing Systems

Project No: G-36-643 Cmd cont.

Project Director: Dr. Philip H. Enslow, Jr.

Sponsor: Office of Naval Research; Arlington, VA 22217

Agreement Period: From 9/1/79 Until 8/31/~~82~~ 83 (Perf. Period)

Type Agreement: Contract No. N00014-79-C-0873

Amount:

$1,114,620	ONR (G-36-643)	Partially funded through 9/30/80 as follows:	
250,000	GIT (G-36-336)	$289,683	ONR (G-36-643)
$1,364,620	TOTAL	125,000	GIT (G-36-336)
		$414,683	TOTAL

Reports Required: Progress Report (as req.); Final Report

Sponsor Contact Person (s):

Technical Matters

Scientific Officer

Program Director Information Systems
Mathematical and Information Sciences Division
Office of Naval Research
800 N. Quincy Street
Arlington, VA 22217

Contractual Matters
(thru OCA)

Office of Naval Research
Resident Representative
325 Hinman Research Building
Georgia Institute of Technology
Atlanta, GA 30332

Defense Priority Rating: DO-C9 under DMS Reg. 1

Assigned to: Information and Computer Science (School/Laboratory)

COPIES TO:

- Project Director
- Division Chief (EES)
- School/Laboratory Director
- Dean/Director–EES
- Accounting Office
- Procurement Office
- Security Coordinator (OCA)
- Reports Coordinator (OCA)
- Library, Technical Reports Section
- EES Information Office
- EES Reports & Procedures
- Project File (OCA)
- Project Code (GTRI)
- Other C. E. Smith

GEORGIA INSTITUTE OF TECHNOLOGY — OFFICE OF CONTRACT ADMINISTRATION

SPONSORED PROJECT TERMINATION/CLOSEOUT SHEET

Date January 14, 1986

Project No. G-36-643 R5365-0A0 — School/~~XXX~~ ICS

Includes Subproject No.(s) N/A

Project Director(s) Dr. Philip H. Enslow, Jr. — GTRC / ~~XXX~~

Sponsor Office of Naval Research Arlington, VA 22217

Title Research on Fully Distributed Data Processing Systems

Effective Completion Date: 8/31/83 (Performance) 10/31/83 (Reports)

Grant/Contract Closeout Actions Remaining:

- [ ] None
- [X] Final Invoice or Final Fiscal Report
- [X] Closing Documents
- [ ] Final Report of Inventions
- [X] Govt. Property Inventory & Related Certificate
- [ ] Classified Material Certificate
- [ ] Other

Continues Project No. : Continued by Project No.

COPIES TO:

Project Director
Research Administrative Network
Research Property Management
Accounting
Procurement/GTRI Supply Services
Research Security Services
Reports Coordinator (OCA)
Legal Services

Library
GTRC
Research Communications (2)
Project File
Other Heyser, Jones, Embry

FORM OCA 69.285

Note:

The following GIT-ICS Technical Reports were submitted as deliverables for this project:

79/11
81/01
81/03
81/04
81/11
81/12
81/15
81/16
81/18
82/01
82/12

They are listed in the Library's catalog under "GIT-ICS ; [Report Series]." See the catalog for complete bibliographical citation, call number, and location.

THE GEORGIA INSTITUTE OF TECHNOLOGY

RESEARCH PROGRAM IN

FULLY DISTRIBUTED PROCESSING SYSTEMS

Quarterly Progress Report Number 1
1 September, 1979 - 30 November, 1979

January, 1980

Supported by

Office of Naval Research
Contract N00014-79-C-0873

School of Information and Computer Science
Georgia Institute of Technology
Atlanta, Georgia 30332

1. INTRODUCTION

This is the first quarterly progress report prepared on the Georgia Tech Research Program in Fully Distributed Processing Systems (FDPS).

On 1 September, 1979, the School of Information and Computer Science, the Georgia Institute of Technology, received a Selected Research Opportunity (SRO) contract from the Office of Naval Research to establish a "Center of Excellence for Research in Highly Distributed Systems." This contract covers a three-year period of operation with a total funding of $1,114,620 plus an additional $250,000 cost sharing from Georgia Tech for equipment. Final negotiations are under way with three other organizations for additional research contracts in this area. It is anticipated that those contracts will be awarded in December, 1979. In addition, another research contract for a three-year period has already been awarded to an individual faculty member in the School of Information and Computer Science---U.S. Army Research Office, "Theory of Systems of Asynchronous Parallel Processors," Nancy Lynch, 3 years, approximately $150,000.

Although distributed processing systems, especially highly distributed and fully distributed ones, have been a subject of interest and research (unfunded for the most part) within the School for the last three years, this is the first major funding received in this area, and these first three months of the project have been devoted primarily to organizational matters and the initiation of specific research projects and tasks.

Since it is strongly felt that the maximum accomplishments will be achieved by the synergistic effects of a large number of individuals working together and interacting, it is planned that the Research Program will be run as a single program with specific directions for the research areas being taken from individual sponsors as appropriate.

2. ORGANIZATION AND STAFFING

Faculty

The following members of the ICS Faculty have been identified as participants in the FDPS Research Program.

Crews, Phillip--Assistant Professor
Demillo, Richard A.--Associate Professor
Enslow, Philip H. Jr.--Professor
Griffeth, Nancy--Assistant Professor

LeBlanc, Richard--Assistant Professor
Livesey, Jon--Assistant Professor (effective April, 1980)
Lynch, Nancy--Associate Professor
Underwood, William--Assistant Professor

Most of these individuals are presently working on specific projects in the program, while others are completing other work already in progress.

Staff

Jensen, Alton P.--Princ. Res. Eng.
McDonell, Sharon--Sr. Secy.
Myers, Jeanette--Res. Scientist
Peckham, Gary (EES)--Chief, Software Applications Div.
Pinion, Nancy--Part-time Secy.

Students

There are 27 students working on various projects in the FDPS Research Program. Of these, 12 are in the Ph.D. program and 4 are preparing MS Thesis on topics in FDPS.

3. RESEARCH PROJECTS INITIATED

The specific research projects have been organized into the major areas identified in the Program Proposal.

A. Theoretical and Formal Studies

A.1 Studies of the Theory of Asynchronous Processors

B. Physical Interconnection and Networking

C. Distributed Operating Systems

C.1 Decentralized Control
C.2 Resource Allocation and Work Distribution in an FDPS
C.3 Distributed Operating System - Initial Considerations
C.4 TBA (Distributed Operating Systems)
C.5 TBA (Distributed Operating Systems)
C.6 TBA (Distributed and Parallel Operating Systems)

D. Distributed Data Bases

D.1 TBA (Distributed Data Bases)

E. **Fault-Tolerance**

F. **Special Hardware to Support FDPS**

G. **Application of Distributed Processing**

G.1 TBA (FDPS Effects on Management Organization)

H. **System Design Methodologies**

H.1 FDPS Requirements Engineering Techniques
H.2 Coordinating Large Programming Projects

I. **System Utilization**

I.1 A Language for Distributed Programming
I.2 System Implementation Language Development

J. **Security**

J.1 Process Structures

K. **System Management**

L. **Evaluation and Comparison**

M. **FDPS Testbed**

M.1 Establishment of FDPS Testbed Facility
M.2 Remote Load Emulator

4. STATUS OF RESEARCH PROJECTS

With the exception of project A.1, "Studies of the Theory of Asynchronous Processors," all of the projects listed above were initiated in the first quarter of the Program and no immediate results are anticipated with one exception, Project H.1, "FDPS Requirements Engineering Techniques", was only a 3-month study and has been completed except for the final report.

Summary of Project A.1 to Date

A very general and tractable automaton-style model has been designed for describing distributed systems. Details of the formalism, as well as some prototypical arbiter examples, appear in reference [b]. The model separates the machinery used to describe the requirements for and the implementation of a system. Complexity measures for implementations have been proposed in [b], most importantly including a natural

measure of "running time" for asychronous systems. Analysis using this measure has been carried out for the examples in [b]. Some techniques derived from those used in ordinary sequential complexity theory have been carried over to the new distributed setting. Suggestions have been made for the types of system parameters whose measurement is of greatest interest. For the specific arbiter examples appearing in [b], our analysis has suggested a particular system design consistig of a tree of polling processes, which has been implemented at the University of Washington for actual use in their computer system. Preliminary observations suggest that the design does in fact perform efficiently.

Work in [a] involves measurement of the space complexity of algorithms for mutual exclusion. There are many results of technical interest, providing upper and lower bounds on the amount of storage required for the attainment of certain fairness properties. A major contribution of [a] has been the development of a concise mathematical notation for describing the execution of systems. We are able to describe precisely many alternative execution possibilities for a system, a task which is extremely difficult to do in English. It is this precision which allows us to prove meaningful lower bound results. Specific technical results of [a] demonstrate ways in which communication space (bandwidth) can be used simultaneously for several purposes, without danger of confusion.

In [c], the problem of designing algorithms which are immune to a limited amount of (non-malicious) process failure is considered. The problem area studied is a generalization of the problem in [a], involving allocation of multiple copies of identical resources. One key contribution of [c] is the careful statement of conditions of "graceful degradation of performance" -- exactly what the system is still required to do when a small number of processors fail. Another contribution is the design of some specific ways for introducing redundancy and thereby achieving the needed reliability. Another is an interesting method for describing some very complicated algorithms by means of layers of virtual systems. Finally, another contribution is the futher use and development of the notation for describing system execution.

In [d], a simplified but realistic distributed system resource allocation problem is considered. The full power of our model is applied to the problem statement, description of a particularly fast solution, and the time analysis of this solution. The contributions of [d] are the development of time analysis techniques and also the particular fast solution proposed.

Current work in progress involves study of (1) ways to cope with malicious failure of process in distributed systems, (2) further development of time analysis techniques (3) building of a small Fortran test system for our algorithms, (4) attempts to prove finally the correctness of our algorithms, (5) study of applications to resource allocation, data base and graph theory problems, (6) development of convenient high-level languages for describing systems within our model, (7) study of further possible improvements on the arbiter design of [b], and (8) generalizing the resource-allocation strategy of [d] for possible use in a real distributed system.

References

[a] Burns, J.E., M.J. Fischer, P. Jackson, N.A. Lynch, and G.L. Peterson, "Data Requirements for Implementation of N-Process Mutual Exclusion Using a Single Shared Variable," GIT-ICS-79/02.

See also - [BFJLP] Burns, J.E., M.J. Fischer, P. Jackson, N.A. Lynch, and G.L. Peterson, "Shared Data Requirements for Implementation of Mutual Exclusion Using a Test-and-Set Primitive," International Conference on Parallel Processing, Bellaire, Michigan, Aug. 1978.

Status: In process of revision for publication in Journal of the ACM.

[b] Lynch, N.A. and M.J. Fischer, "On Describing the Behavior and Implementation of Distributed Systems," GIT-ICS-79-03

See also - Lecture Notes in Computer Science, Semantics of Concurrent Computation, Proceedings Evian, France 1979, pp. 147-171.

Status: After presentation at International Symposium on Semantics of Concurrent Computation, Evian, France, July 1979, this paper was invited to be submitted to a special conference issue of Theoretical Computer Science. It is currently being revised for submission.

[c] Fischer, M.J., N.A. Lynch, J.E. Burns, and A. Borodin, "Resource Allocation with Immunity to Limited Process Failure," GIT-ICS-79-10.

See also - 20th Annnual Symposium on Foundations of Computer Science, Puerto, Rico, 1979, pp. 234-254.

Status: Will probably be submitted at a later date for journal publication.

[d] "Nearby Resource Allocation in a Distributed System," Proceedings of 1980 Symposium on Theory of Com-

puting, ACM.

5. PLANS FOR THE IMMEDIATE FUTURE

Of most significance is the series of visits planned for project A.1, "Studies of the Theory of Asynchronous Processors." Professor Michael Fischer, University of Washington, will be at GIT as a visitor for the months of January, February and March, 1980. Short visits during the next quarter are also planned for Arnold Schoenhage, Eshrat Arjomandi, Leslie Lamport, Armin Cremers, and Edward Lazowska.

6. TRAVEL

Dates of Trip: June, 1979
Individuals Travelling: Nancy Lynch
Purpose: Research work with Michael Fischer

Dates of Trip: 5-9 June, 1979
Individuals Travelling: Richard DeMillo
Purpose: Participation in AFOSR Summer School on Security, Draper Labs, Cambridge, MASS.

Dates of Trip: July, 1979
Individuals Travelling: Nancy Lynch
Purpose: Present Paper at Evian France

Dates of Trip: 14-21 September, 1979
Individuals Travelling Richard DeMillo
Purpose: Research on Stochastic Synchronization

Dates of Trip: 1-5 October, 1979
Individuals Travelling: Jack Corley, Phillip Crews, Philip Enslow, Richard LeBlanc, Tim Saponas, Don Sharp
Purpose: Attend the First International Conference on Distributed Processing, Huntsville, Alabama

Dates of Trip: 10 October, 1979
Individuals Travelling: Philip Enslow
Purpose: Presentation-"Research Issues ir Fully Distributed Systems", 1979 AICA National Conference, Bari, Italy.

Dates of Trip: November, 1979
Individuals Travelling: Nancy Lynch
Purpose: Present paper at Foundations of Computer Science

Conference IEEE, Puerto Rico

Dates of Trip: 5 November, 1979
Individuals Travelling: Philip Enslow
Purpose: Presentation "Research Issues in Fully Distributed Systems" IEEE Computer Society Chapter, Atlanta, GA.

Dates of Trip: 9 November, 1979
Individuals Travelling: Philip Enslow
Purpose: Presentation "Research Issues in Fully Distributed Processing Systems", 7th Annual CDC on Principles of Software Development, Minneapolis, MINN.

7. VISITORS

Dates of Visit: December, 1978
Visitor: Allan Borodin
Purpose: Discussion of common research interests
Individual Contacted: Nancy Lynch

Dates of Visit: January, 1979
Visitor: Michael Fischer
Purpose: Research Collaboration
Individuals Contacted: Nancy Lynch

Dates of Visit: 11 Jan, 1980
Visitor: R.J. Lipton
Purpose: Research
Individuals Contacted: Richard DeMillo

Dates of Visit: May, 1979
Visitor: Michael Fischer
Purpose: Research Collaboration
Individual Contacted: Nancy Lynch

Dates of Visit: May, 1979
Visitor: Damel Rozenkrantz, Richard Stearns, Philip Lewis
Purpose Discussion of common research interests
Individual Contacted: Nancy Lynch

Dates of Visit: November, 1979
Visitor: Michael Fischer
Purpose: Research collaboration
Individual Contacted: Nancy Lynch

Dates of Visit: 19 November, 1979
Visitor: Dave F. Palmer, General Research Corp.
Purpose: General orientation on the GIT FDPS Research Program. Discuss GRC work on distibuted processing funded by U.S. Army BMDATC-P, Huntsville, Alabama.
Individuals Contacted: Philip Enslow, Richard LePlanc, Jack

Corley

Date of Visit: 27 November, 1979
Visitor: Aaron H. Coleman, Joseph Schweyer, Computer Sciences Corporation
Purpose: Orientation on the GIT FDPS Research Program. Discussion of CSC research plans in DDP. Discussion of possible cooperation with GIT on proposed Navy research project.
Individuals Contacted: Philip Enslow, Richard LeBlanc, Richard DeMillo

Date of Visit: 28-29 November, 1979
Visitor: Dr. Robert Grafton, Office of Naval Research
Purpose: Review of progress on ONR Contract. Discussions with individual faculty members.
Individuals Contacted: Philip Enslow, Richard LeBlanc, Phillip Crews, Nancy Griffeth, Richard DeMillo, Nancy Lynch

Date of Visit: 29-30 November, 1979
Visitor: Dr. Virgil Wallentine, Kansas State University
Purpose: Present In-progress Review to AIRMICS. (Wallentine has a research contract with AIRMICS for the development of a network operating system for distributed processing.)
Individuals Contacted: Philip Enslow, Richard LeBlanc

8. PUBLICATIONS

Author(s) Philip Enslow, Robert Gordon
Title IPC Workshop Report
Number GIT-ICS-79/11
Type Final Technical Report
Date December, 1979

Author(s) Philip Enslow
Title Quarterly Progress Report Number 1
Date December, 1979

Author(s) Nancy Lynch
Title Fast Allocation of Nearby Resources in a Distributed System
Type Conference Paper
Date May, 1980
Comments Submitted for publication in Conference of ACM Annual Symposium on the Theory of Computing

Author(s) Michael Fischer, Nancy Lynch, Alan Borodin, James Burns
Title Resource Allocation with Immunity to Limited Process Failure
Number GIT ICS TR 79-10
Type Conference Paper
Date October 1979

Comments Also in Proceedings of 29th Annual IEEE Symposium on Foundations of Computer Science.

Author(s) Nancy Lynch, Michael Fischer
Title On Describing the Behavior and Implementation of Distributed Systems
Number GIT ICS TR 79-03-01
Type Conference Paper
Date July 1979
Comments Also in Proceedings of International Symposium on Semantics of Concurrent Computation, Evian, France, 1979. Springer-Verlag.
Also to appear in special conference issue of Theoretical Computer Science.
Author(s) James Burns, Michael Fischer, Paul Jackson, Nancy Lynch
Title Data Requirements for Implementation of N-Process Mutual Exclusion Using a Single Shared Variable
Number GIT ICS TR 79-02
Type Conference Paper
Date August 1978 (Journal of the ACM)
Comments Also published in Proceedings of 1979 Conference on Parallel Processing, Bellaire, Mich.

Author(s) George I. Davida, Richard DeMillo, Richard Lipton
Title A System Architecture to Support a Verifiably Secure Multilevel Security System
Number
Type Conference Paper
Date April 1980
Comments To be presented at 1980 IEEE Symposium on Privacy and Security

THE GEORGIA INSTITUTE OF TECHNOLOGY

RESEARCH PROGRAM IN

FULLY DISTRIBUTED PROCESSING SYSTEMS

Quarterly Progress Report Number 2
1 December, 1979 - 29 February, 1980

April, 1980

Supported by

Office of Naval Research (ONR)
Contract: N00014-79-C-0873
GIT Project: G36-643

U.S. Air Force Rome Air Development Center (RADC)
Contract: F30602-78-C-0120
GIT Project: G36-649

International Business Machines,
General Systems Division (IBM)
Agreement: GSD-210189
GIT Project: G36-648

National Science Foundation (NSF)
Contract: MCS77-28305
Subcontract from Univ. of Wisc.: 144-L729
GIT Project: G36-630

U.S. Army Research Office (ARO)
Contract: DAAG29-79-C-0155
GIT Project: G36-638

U.S. Army Institute for Research in
Management Information and Computer Science (AIRMICS)
Contract: DAAK70-79-D-0087
GIT Project: G36-647

School of Information and Computer Science
Georgia Institute of Technology
Atlanta, Georgia 30332

1. INTRODUCTION

This is the second quarterly progress report prepared on the Georgia Tech Research Program in Fully Distributed Processing Systems (FDPS).

a. Program Description.

The Georgia Tech Research Program in Fully Distributed Processing Systems is a comprehensive investigation of data processing systems in which both the physical and logical components are extremely loosely coupled while operating with a high degree of control autonomy at the component level. The definition of the specific class of multiple computer systems being investigated, and the operational characteristics and features of those systems is motivated by the desire to advance the state-of-the-art for that class of systems that will deliver a high proportion of the benefits currently being claimed for distributed processing systems. The scope of individual topics being investigated under this program ranges from formal modeling and theoretical studies to empirical examinations of prototype systems and simulation models. Also included within the scope of the program are areas such as the utilization of FDPS's and their interaction with management operations and structure.

b. Program Support.

The principle support for the program is a Selected Research Opportunity contract from the Office of Naval Research; however, there are a number of other sources of funding which also support the program. A complete list of these is given below.

Title: "Research on Fully Distributed Data Processing Systems"
Funding Agency: Office of Naval Research (ONR)
Contract Number: N00014-79-C-0873
GIT Project No.: G36-643
Principle Investigator: Philip H. Enslow, Jr.

Title: "Research on Distributed Control"
Funding Agency: U.S. Air Force Rome Air Development Center (RADC)
Contract Number: F30602-78-C-0120
GIT Project No.: G36-649
Principle Investigator: Philip H. Enslow, Jr.

Title: "Agreement By and Between IBM and GTRI"
Funding Agency: International Business Machines, General Systems Division (IBM)
Contract Number: GSD Agreement Number 21019
GIT Project Number: G36-648
Principle Investigator: Philip H. Enslow, Jr.

Title: "Foundations of Deterministic Scheduling of Processes for Parallel Execution"
Funding Agency: National Science Foundation (NSF)
Contract Number: MCS77-28308
(Univ. of Wisc. subcontract number: 144-L729)
GIT Project Number: G36-630
Principle Investigator: Richard A. DeMillo

Title: "Theory of Systems of Asynchronous Parallel Processors"
Funding Agency: U.S. Army Research Office (ARO)
Contract Number: DAAG29-78-G-0155
GIT Project Number: G36-638
Principle Investigator: Nancy Lynch

Title: "Support of MILPERCEN Data Storage Concept"
Funding Agency: U.S. Army Institute for Research in Management Information and Computer Science (AIRMICS)
Contract Number: DAAK70-78-D-0087
GIT Project Number: G36-647
Principle Investigator: A.P. Jensen

c. Administrative Changes

During this quarter, additional research contracts supporting the FDPS program have been awarded. Specifically, these are the RADC, IBM, NSF, ARO, and AIRMICS contracts described above. Also during this quarter the final report on one contract was published, completing that project.

Title: "Interprocess Communication in Highly Distributed Systems" (A Workshop)
Funding Agency: U.S. Army Research Office (ARO)
Contract Number: DAAG29-78-G-0010
GIT Project No.: G36-632
Principle Investigator: Philip H. Enslow, Jr.

2. ORGANIZATION AND STAFFING

Faculty

The following members of the ICS Faculty have been identified as participants in the FDPS Research Program.

Crews, Phillip--Assistant Professor
Demillo, Richard A.--Associate Professor
Enslow, Philip H. Jr.--Professor
Griffeth, Nancy--Assistant Professor
LeBlanc, Richard--Assistant Professor
Livesey, Jon--Assistant Professor
(effective September, 1980)
Lynch, Nancy--Associate Professor

Most of these individuals are presently working on specific projects in the program, while others are completing other work already in progress.

Staff

Jensen, Alton P.--Princ. Res. Eng.
McDonell, Sharon--Sr. Secy.
Myers, Jeanette--Res. Scientist
Pinion, Nancy--Part-time Secy.

Students

There are 30 students working on various projects in the FDPS Research Program. Of these, 12 are in the Ph.D. program and 5 are preparing their MS Thesis on topics in FDPS.

3. CURRENT RESEARCH PROJECTS

The specific research projects have been organized into the major areas identified in the basic program proposal.

A. Theoretical and Formal Studies

A.1 Studies of the Theory of Asynchronous Processors
A.2 Decomposition of Parallel Systems
A.3 Reliable Systems
A.4 Time Performance of Distributed Systems

A.3 Audit Algorithms
A.4 Ticket Systems
A.7 Synchronous Simulation
A.8 Distributed Resource Allocation

B. Physical Interconnection and Networking

B.1 Heterogeneous Networking
B.2 Local Networking in Fully Distributed Processing Systems

C. Distributed Operating Systems

C.1 Decentralized and Distributed Control
C.2 Resource Allocation and Work Distribution in an FDPS
C.3 Fully Distributed Operating System - Initial Considerations
C.4 TDA (Distributed Operating Systems)
C.5 Process Support in Distributed Systems
C.6 Non-Homogeneous Operating Systems
C.7 FDOS - Preliminary Implementation Studies

D. Distributed Data Bases

D.1 Implementation of Distributed Database Systems
D.2 Support of MILPERCEN Data Storage Concept

E. Fault-Tolerance

F. Special Hardware to Support FDPS

G. Application of Distributed Processing

H. System Design Methodologies

H.1 FDPS Requirements Engineering Techniques
H.2 Coordinating Large Programming Projects

I. System Utilization

I.1 A Language for Distributed Programming
I.2 System Implementation Language Development

J. Security

J.1 Process Structures

K. System Management

L. Evaluation and Comparison

M. FDPS Testbed

M.1 Establishment of FDPS Testbed Facility
M.2 Remote Load Emulator
M.3 Fully Distributed Operating System Simulation Testbed

4. SUMMARY OF PROGRESS

A.1 Studies of the Theory of Asynchronous Processors (Lynch, Fischer, Lamport, Lazowska, Schönhage, Arjomandi)

Work continues in the development of models, decomposition techniques, and complexity analysis techniques for distributed systems. Visitors this quarter have included Fischer, Lamport, Lazowska, Schönhage, and Arjomandi. Recent work has inspired projects A.2, A.3, A.4, A.5, A.6, A.7, and A.8 described below.

A.2 Decomposition of Parallel Systems (Lynch, Fischer)

Synchronization algorithms are decomposed using two stage models, with simulation used to eliminate initial simplifications, such as centralized control and multiple shared variables.

A.3 Reliable Systems (Lynch, Fischer, Lamport)

Redundancy is used to alleviate the effects of "shutdown", "death", and "malicious failure" of processes. Agreement with faulty inputs is difficult and slow, requiring $k+1$ "rounds" of information exchange to protect against up to k faults. Special cases are being considered.

A.4 Time Performance of Distributed Systems (Lynch, Fischer, Lazowska, Schönhage)

Application of complexity theory in developing tools to measure worst-case and expected performance of distributed systems under specified operating conditions. Analysis of arbiter problems includes work on lower (time) bounds, with restricted access to communication variables.

A.5 Audit Algorithms (Griffeth, Fischer, Lynch)

Development of algorithms for auditing distributed assets without delaying transactions, or contradicting information propagated through the system by obtaining a balance that could not have existed at any point in time.

A.6 Ticket Systems (Lynch, Fischer, Griffeth)

Design and analysis of algorithms for ticket distribution, including careful statement of correctness and performance requirements.

A.7 Synchronous Simulation (Lynch, Fischer, Arjomandi)

Development and analysis of techniques for converting synchronous, parallel algorithms into equivalent asynchronous algorithms, by devising protocols which insure progress of each process without actually stopping computation to achieve synchronization.

A.8 Distributed Resource Allocation (Lynch)

Development of a simple but realistic model of the resource allocation problem, a fast solution, and time analysis of the solution.

B.1 Heterogeneous Networking (Crews, Bray, Greene, Tuberville)

The IBM Series/1 has been connected to the FDPS test-bed (PRIME P-400's) through a unidirectional communication path. This path has facilitated file transfer from the testbed to the Series/1, and consequently has provided necessary software to initiate work on the common command language facility and software tools. At the same time, a link has been established between the Series/1 and CYBER system, and work is underway to establish an interface at the physical code, operating system, and programming language level.

B.2 Local Networking in FDPSs (Enslow)

A survey of the state-of-the-art in local network technology and equipment has been initiated. It appears that the primary weaknesses of systems currently available or proposed are in the areas of host interaction with the local network and host-to-host interaction.

C.1 Decentralized and Distributed Control (Enslow, LeBlanc, Crews, Saponas, Wice)

The first goal of this project is to characterize and analyze models of distributed and decentralized control applicable to highly distributed systems. A major problem facing the research team is the development of a technique or framework by which various control models can be described and catalogued. The principle effort thus far in this project has been focused on identifying various possible models and comparing and analyzing these models with the goal of developing a basis for a complete taxonomy. Major progress has been made in this first step, and a paper is being prepared for presentation at the IEEE Computer Society COMPCON in September, 1980.

C.2 Resource Allocation and Work Distribution in an FDPS (Enslow, Sharp)

Activity during this period has focused on developing a descriptive framework suitable for preparing a taxonomy of allocation and distribution models. The approach taken has been to prepare descriptions of as many models as possible and then to work backwards to develop the framework. An initial framework has been prepared and is being refined.

C.3 FDOS - Initial Considerations (Enslow, LeBlanc, Crews, Akin, Flinn, Forsyth, Fukuoka, Myers, Pitts, Saponas, Skowbo, Spafford, Wice)

The organization and outline for the complete specification of an FDOS is being prepared.

C.4 TBA - Distributed Operating Systems (Livesey)

No activity this quarter.

C.5 Process Support in Distributed Systems (Enslow, Skowbo)

A survey and analysis of communication protocols has been initiated to isolate essential features for support of distributed processes. Implications of transport protocols for the IPC interface are being studied.

C.6 Non-Homogeneous Operating Systems (Ratzel)

No significant activity this quarter. Project is still in preliminary stages.

C.7 FDOS - Preliminary Implementation Studies (Myers, Enslow, Gaither, L. Newell, S. Newell, Wice)

The design and implementation of the FDOS has been initiated. The approach being taken is to first address those areas on which there is general agreement as to the functionality required/desired. The first area being designed is that of "message transport".

D.1 Implementation of Distributed Database Systems (Griffeth)

This project is planned to commence in June, 1980.

D.2 Support of MILPERCEN Data Storage Concept (Jensen, Doyle, Gehl, Bingham)

Applicable literature and reference documentation has been collected. A site visit to the U.S. Army Military Personnel Center, Alexandria, Virginia, has been made, during which briefings were presented and interviews held.

H.1 FDPS Requirements Engineering Techniques (Underwood, Corley)

No significant activity this quarter.

H.2 Coordinating Large Programming Projects (Enslow, Smith)

A questionnaire to be utilized to gather historical information and manager's perception of the problems and possible solutions has been prepared. This draft questionnaire has been circulated to a number of individuals for comment and recommendations on a wider population to survey.

I.1 A Language for Distributed Programming (LeBlanc, Maccabe, Forsyth)

Existing languages with features related to our goals are currently being studied. This includes the implementation of multi-process programs in MODULA,

in order to gain experience with interprocess communication problems. In preparation for our language design work, design goals have been identified and a computational model on which the language will be based has been established. A paper entitled "A Language Model for Fully Distributed Systems" has been submitted to COMPCON '80 Fall. Another paper is being prepared for submission to the ACM Pacific '80 Conference, which has distributed processing as its theme.

I.2 System Implementation Language Development (LeBlanc, Akin, Strickland)

UW-Pascal is being transported to the PRIME-400 with extensions to support the FDPS development. No significant progress this quarter.

J.1 Process Structures (DeMillo, Lipton, Miller, Davida)

Investigation of several aspects of parallel and distributed system design, including multilevel security, models of synchronization, and efficiency of interprocess communication.

M.1 Establishment of FDPS Testbed Facility (Myers, Elshoff, Gaither, Howe, Flinn, L. Newell, S. Newell, Wice)

The subroutines comprising the Primenet interprocess communication facility are being used and tested by student programmers, and will be used in the near future to implement message transport and message handling.

M.2 Remote Load Emulator (Myers, Enslow, Forsyth, Howe)

Programmable "scripts" have been devised to describe a variety of loads. Student programmers have completed a lexical analyzer and parser to translate scripts for fast interpretation.

M.3 FDOS Simulation Testbed (LeBlanc, Gaither, Maccabe, Myers, S. Newell, Wice)

The FDOS simulation testbed will provide an environment for the initial testing and analysis of operating system algorithms currently being developed. The overall design is complete; a more detailed design is being written, utilizing SIMULA constructs, and is in its final stages.

5. TRAVEL RELATED TO THE FDPS PROGRAM

Dates of Trip: 1 February, 1980
Individuals Travelling: James Skowbo, Steve Newell
Itinerary: Atlanta, Georgia
Purpose: Attend ACM sponsored Professional Development Seminar - Distributed Processing Systems

Dates of Trip: 14 February, 1980
Individuals Travelling: Philip Enslow
Itinerary: Atlanta, Georgia
Purpose: Present briefing on FDPS program and other ICS research projects.

Dates of Trip: 25-29 February, 1980
Individuals Travelling: Phillip Crews
Itinerary: San Francisco, California
Purpose: Attend COMPCON '80 Spring

Dates of Trip: 27-28 February, 1980
Individuals Travelling: A.P. Jensen, John Gehl, Jim Doyle
Itinerary: Alexandria, Virginia
Purpose: Site survey of MILPERCEN Data Facilities.

6. VISITORS

Dates of Visit: 1 January - 31 March, 1980
Visitor: Michael Fischer
Purpose: Research Collaboration
Individual Contacted: Nancy Lynch

Dates of Visit: 11 January, 1980
Visitor: R.J. Linton
Purpose: Research Collaboration
Individual Contacted: Richard DeMillo

Dates of Visit: 14-16 January, 1980
Visitor: Robert Cook, U. Visc., Madison
Purpose: To discuss programming language design and operating system simulation work being done at Wisconsin.
Individual Contacted: Richard LeBlanc, Philip Enslow, Phillip Crews

Dates of Visit: 21-25 January, 1980
Visitor: Leslie Lamport, Stanford Research Institute
Purpose: Research collaboration on problems involving programming in an environment including faulty processors, and choice of primitive operations for models of asynchronous systems.
Individual Contacted: Nancy Lynch, Michael Fischer

Dates of Visit: 4-6 February, 1980
Visitor: Edward Lazowska, Univ. of Washington
Purpose: Research collaboration on problems involving performance evaluations of distributed systems, and design of arbitration protocols.
Individual Contacted: Nancy Lynch, Michael Fischer

Dates of Visit: 18-21 February, 1980
Visitor: Eshrat Arjomandi, York Univ., Toronto, Canada
Purpose: Research collaboration on problems involving relationships between synchronous and asynchronous models for parallel computation, and design of distributed graph algorithms.
Individual Contacted: Nancy Lynch, Michael Fischer

Dates of Visit: 21 February, 1980
Visitor: Tadaaki Bandoh, Hitachi Research Laboratory, Japan.
Purpose: Discuss Bandoh's work in dataflow machines and our work in FDPS.
Individual Contacted: P. Enslow, J. Myers, H. Fukuoka, D. Fitts, T. Saponas, D. Sharp, J. Skowbo, P. Wice.

Dates of Visit: 25 February - 14 March, 1980
Visitor: Arnold Schönhage, Univ. of Tubingen, Germany
Purpose: Research collaboration on problems involving models for parallel computation, stochastic analysis of distributed systems, design of arbiter systems, and techniques for proving lower bounds for arbitration problems.
Individual Contacted: Nancy Lynch, Michael Fischer

7. PUBLICATIONS

Author(s): P.H. Enslow, P. Gordon
Title: IPC Workshop Report
Number: GIT-ICS-79/11
Date: December, 1979

Author(s): R.A. DeMillo, R.J. Lipton, R.E. Miller
Title: Stochastic Synchronization
Date: June, 1980

Author(s): R.A. DeMillo, G.I. Davida, R.J. Lipton
Title: Secure Key Distribution
Type: Conference Paper
Date: April, 1980
Comments: To be presented at 1980 IEEE Symposium on Security and Privacy.

Author(s): P.H. Enslow
Title: Quarterly Progress Report - Number 2
Type: Quarterly Progress Report
Date: April, 1980

THE GEORGIA INSTITUTE OF TECHNOLOGY

RESEARCH PROGRAM IN

FULLY DISTRIBUTED PROCESSING SYSTEMS

Quarterly Progress Report Number 3
1 March, 1980 - 31 May, 1980

July, 1980

Supported by

Office of Naval Research (ONR)
Contract: N00014-79-C-0873
GIT Project: G36-643

U.S. Air Force Rome Air Development Center (RADC)
Contract: F30602-78-C-0120
GIT Project: G36-649

International Business Machines,
General Systems Division (IBM)
Agreement: GSD-210189
GIT Project: G36-648

National Science Foundation (NSF)
Contract: MCS77-28305
Subcontract from Univ. of Wisc.: 144-L729
GIT Project: G36-630

U.S. Army Research Office (ARO)
Contract: DAAG29-79-C-0155
GIT Project: G36-638

U.S. Army Institute for Research in
Management Information and Computer Science (AIRMICS)
Contract: DAAK70-79-D-0087
GIT Project: G36-647

School of Information and Computer Science
Georgia Institute of Technology
Atlanta, Georgia 30332

1. INTRODUCTION

This is the Third Quarterly Progress Report prepared on the Georgia Tech Research Program in Fully Distributed Processing Systems (FDPS).

a. Program Description.

The Georgia Tech Research Program in Fully Distributed Processing Systems is a comprehensive investigation of data processing systems in which both the physical and logical components are extremely loosely coupled while operating with a high degree of control autonomy at the component level. The definition of the specific class of multiple computer systems being investigated, and the operational characteristics and features of those systems is motivated by the desire to advance the state-of-the-art for that class of systems that will deliver a high proportion of the benefits currently being claimed for distributed processing systems. The scope of individual topics being investigated under this program ranges from formal modeling and theoretical studies to empirical examinations of prototype systems and simulation models. Also included within the scope of the program are areas such as the utilization of FDPS's and their interaction with management operations and structure.

b. Program Support.

The principle support for the program is a Selected Research Opportunity contract from the Office of Naval Research; however, there are a number of other sources of funding which also support the program. A complete list of these is given below.

Title: "Research on Fully Distributed Data Processing Systems"
Funding Agency: Office of Naval Research (ONR)
Contract Number: N00014-79-C-0873
GIT Project No.: G36-643
Principle Investigator: Philip H. Enslow, Jr.

Title: "Research on Distributed Control"
Funding Agency: U.S. Air Force Rome Air Development Center (RADC)
Contract Number: F30602-78-C-0120
GIT Project No.: G36-649
Principle Investigator: Philip H. Enslow, Jr.

Title: "Agreement By and Between IBM and GTRI"
Funding Agency: International Business Machines, General Systems Division (IBM)
Contract Number: GSD Agreement Number 210189
GIT Project Number: G36-648
Principle Investigator: Philip H. Enslow, Jr.

Title: "Foundations of Deterministic Scheduling of Processes for Parallel Execution"
Funding Agency: National Science Foundation (NSF)
Contract Number: MCS77-28305
(Univ. of Wisc. subcontract number: 144-L729)
GIT Project Number: G36-630
Principle Investigator: Richard A. DeMillo

Title: "Theory of Systems of Asynchronous Parall Processors"
Funding Agency: U.S. Army Research Office (ARO)
Contract Number: DAAG29-79-C-0155
GIT Project Number: G36-638
Principle Investigator: Nancy Lynch

Title: "Support of MILPERCEN Data Storage Concept"
Funding Agency: U.S. Army Institute for Research in Management Information and Computer Science (AIRMICS)
Contract Number: DAAK70-79-D-0087
GIT Project Number: G36-647
Principle Investigator: A.P. Jensen

2. ORGANIZATION AND STAFFING

Faculty

The following members of the ICS Faculty have been identified as participants in the FDPS Research Program.

Crews, Phillip--Assistant Professor
DeMillo, Richard A.--Associate Professor
Enslow, Philip H. Jr.--Professor
Griffeth, Nancy--Assistant Professor
LeBlanc, Richard--Assistant Professor
Livesey, Jon--Assistant Professor
(effective September, 1980)
Lynch, Nancy--Associate Professor

Most of these individuals are presently working on specific projects in the program, while others are completing other work already in progress.

Staff

Jensen, Alton P.--Princ. Res. Eng.
McDonell, Sharon--Sr. Secy.
Myers, Jeanette--Res. Scientist
Pinion, Nancy--Part-time Secy.

Students

There are 30 students working on various projects in the FDPS Research Program. Of these, 12 are in the Ph.D. program and 5 are preparing their MS Thesis on topics in FDPS.

3. CURRENT RESEARCH PROJECTS

The specific research projects have been organized into the major areas identified in the basic program proposal.

Theoretical and Formal Studies

A.1 Studies of the Theory of Asynchronous Processors
A.2 Decomposition of Parallel Systems
A.3 Reliable Systems
A.4 Time Performance of Distributed Systems
A.5 Audit Algorithms
A.6 Ticket Systems
A.7 Synchronous Simulation
A.8 Distributed Resource Allocation
A.9 Theory of Distributed Databases
A.10 Arbiter Design
A.11 Shared Memory Bounds for Synchronization Problems

B. Physical Interconnection and Networking

B.1 Heterogeneous Networking
B.2 Local Networking in Fully Distributed Processing Systems

C. Distributed Operating Systems

C.1 Decentralized and Distributed Control
C.2 Resource Allocation and Work Distribution in an FDPS
C.3 Fully Distributed Operating System - Initial Considerations
C.4 TBA (Distributed Operating Systems)

C.5 Process Support in Distributed Systems
C.6 Non-Homogeneous Operating Systems
C.7 FDOS - Preliminary Implementation Studies

D. **Distributed Data Bases**

D.1 Implementation of Distributed Database Systems
D.2 Support of MILPERCEN Data Storage Concept

E. **Fault-Tolerance**

F. **Special Hardware to Support FDPS**

G. **Application of Distributed Processing**

H. **System Design Methodologies**

H.1 FDPS Requirements Engineering Techniques
H.2 Coordinating Large Programming Projects

I. **System Utilization**

I.1 A Language for Distributed Programming
I.2 System Implementation Language Development

J. **Security**

J.1 Process Structures

K. **System Management**

L. **Evaluation and Comparison**

M. **FDPS Testbed**

M.1 Establishment of FDPS Testbed Facility
M.2 Remote Load Emulator
M.3 Fully Distributed Operating System Simulation Testbed

4. SUMMARY OF PROGRESS

A.1 Studies of the Theory of Asynchronous Processors (Lynch, Fischer, Lamport, Lazowska, Schönhage, Arjomandi)

Much of the quarter was spent writing up and exploring ideas originating during the previous quarter. Progress is as described in the Summaries for Projects A.2 through A.11.

A.2 Decomposition of Parallel Systems (Lynch, Fischer)

Further progress was made in organizing ideas for the decomposition.

A.3 Reliable Systems (Lynch, Fischer, Lamport)

No significant progress to report.

A.4 Time Performance of Distributed Systems (Lynch, Fischer, Lazowska, Schönhage)

A simulation program has been developed which calculates running times for distributed algorithms with a tree network topology. This system is being used to obtain performance results where analysis is difficult, and also to check analytic results. A first draft has been written of a paper which demonstrates techniques for proving lower bounds on time performance for algorithms solving distributed problems. The two results for the paper are a lower bound for a simple arbitration problem, and a lower bound for a synchronized simulation problem. Corresponding upper bounds are also described. This paper by Arjomandi, Schönhage, Fischer, and Lynch will be submitted to a conference in the near future.

A.5 Audit Algorithms (Griffeth, Fischer, Lynch)

No significant progress to report.

A.6 Ticket Systems (Griffeth, Lynch, Fischer)

Work this quarter includes the development, proof, and simulation study of an algorithm determining optimal static placement of resources in a distributed system. The first draft of a conference paper is being prepared. The static results are being incorporated into a dynamic distributed

algorithm. An apparently efficient distributed algorithm has been designed, based on the optimal static placement. Since we have so far been unable to prove formal bounds on the overhead introduced by the necessary dynamic decisions, we have programmed the algorithm in our simulation system (described above). We are currently running tests to see how well the algorithm performs under various input loads and with various network sizes. Performance results obtained so far have been uniformly excellent, but are still preliminary. We also have an outline of a formal correctness proof for our algorithm.

A.7 Synchronous Simulation (Lynch, Fischer, Arjomandi)

See Progress Summary for Project A.4.

A.8 Distributed Resource Allocation (Lynch)

The paper, "Fast Allocation of Nearby Resouces in a Distributed System", was presented by N. Lynch at the 1980 ACM Symposium on Theory of Computing in Los Angeles. Subsequently, the paper was invited for inclusion in a special issue of the UCSS to be devoted to papers of that conference.

A.9 Theory of Distributed Databases (Ghoudjehbaklou, Lynch)

Preliminary studies of problems arising in the distributed database area suitable for our types of analysis have begun. Ghoudjehbaklou is conducting studies of adaptive distributed resource allocation algorithms which use distributed data in their implementation.

A.10 Arbiter Design (Lynch, Griffeth, Schönhage, Fischer)

A design of an arbiter which provably works as fast as possible, both in the worst case and in the case of light input load, has been outlined.

A.11 Shared Memory Bounds for Synchronization Problems (Burns, Lynch)

Revision of a paper, "Shared Data Requirements for Implementation of Mutual Exclusion Using a Test-and-Set Primitive", was carried out for eventual publication in the Journal of the ACM.

B.1 Heterogeneous Networking (Crews, Greene)

Software Tools has been copied from the PR1ME testbed and is currently being implemented on the Series/1. Software Tools will establish a common user interface to the heterogeneous systems, and will be the first step in providing an efficient universal interface across the disparate systems.

B.2 Local Networking in FDPSs (Enslow)

A survey was made of the commercial equipment that could be made available during the time period of interest. As a result of this survey, a commercial system was selected for installation as soon possible to provide the initial testbed for work in this area.

C.1 Decentralized and Distributed Control (Enslow, LeBlanc, Crews, Saponas, Wice)

A presentation on the status of the work on this project was made to the RADC Distributed Processing Technical Exchange Seminar. Motivated by the need to present our findings to a large group of individuals who are not totally conversant with the concepts of fully distributed systems, major progress was made in the techniques and methodology for describing the various models of distributed processing systems. Both the physical and logical models of the components involved in distributed and decentralized control were defined.

In categorizing the various models of distributed control, it was found that the following characteristics are significant: (1) Form of the work request, (2) Work request processing, (3) Information gathering for resources required and available, (4) Sources of information, (5) Task-graph building, and (6) Mode of execution monitoring.

Four major types of work requests were identified based on combinations of the following attributes: (1) Are imbedded (invisible) external references allowed, (2) What is the form of the complete request (a single executable file or multiple executable files with connectivity present), and (3) Is process interaction present. Similarly, three major characteristics of the control model were identified: (1) When are external references resolved and resources located and allocated, (2) When is interprocess communication established, and (3) When is the task graph built. The two options possible in each case are: (a) prior to task initiation, and (b) as requested.

Based on the characterizations developed above, a number of examples were prepared clearly explicating the operation of decentralized control.

A paper, "A Model for Decentralized Control in a Fully Distributed Processing System", will be presented at COMPCON Spring '80. The model presented in the paper is being successively refined, and initial plans to simulate the model are being made.

C.2 Resource Allocation and Work Distribution in an FDPS (Enslow, Sharp)

The first draft of a working paper on this subject has been prepared.

C.3 FDOS - Initial Considerations (Enslow, LeBlanc, Crews, Akin, Flinn, Forsyth, Fukuoka, Myers, Pitts, Saponas, Skowbo, Spafford, Wice)

Recent work in this area has focused primarily on the development of a local operating system which will support network operating system operations.

C.4 TBA - Distributed Operating Systems (Livesey)

No activity this quarter.

C.5 Communications Support in Distributed Systems (Enslow, Skowbo)

A study of existing communications systems and proposed architectures suggests the need in fully distributed systems for specialized services at many levels of the protocol hierarchy. These services are being identified and defined in terms of existing or proposed standards for circuit- and packet-switched communication.

C.6 Non-Homogeneous Operating Systems (Ratzel)

No significant progress to report.

C.7 FDOS - Preliminary Implementation Studies (Myers, Enslow, Gaither, Newell, Wice)

No significant progress to report.

D.1 Implementation of Distributed Database Systems (Griffeth)

This project will build on the progress reported for Project A.6.

D.2 Support of MILPERCEN Data Storage Concept (Jensen, Doyle, Gehl, Bingham)

Work to date confirms that the MILPERCEN system for managing the records of enlisted and commissioned personnel is virtually unique with respect to dimensions, complexity, and scope. A comparative analysis of military, federal government, civilian, and private sector human-resource systems has been initiated in order to assess the potential and technical problems of integrated manpower management systems.

H.1 FDPS Requirements Engineering Techniques (Underwood, Corley)

No significant progress to report.

H.2 Coordinating Large Programming Projects (Enslow, Smith)

The principle activity during this period has involved the refinement of the questionnaire, as well as the explanatory material that will accompany it.

I.1 A Language for Distributed Programming (LeBlanc, Maccabe, Forsyth)

The paper entitled "A Language Model for Fully Distributed Systems has been accepted for presentation at COMPCON '80, Fall. Work is proceeding toward designing language features based on the model described in that paper. One aspect of this work is described in a paper entitled "Communication Facilities in Programming Languages for Fully Distributed Systems" which has been submitted to the ACM Pacific '80 Conference.

I.2 System Implementation Language Development (LeBlanc, Akin, Strickland)

Due to the development of a PASCAL compiler by PR1ME, we have determined that further work on transporting UW-PASCAL would not be worthwhile. Our attention has turned to the development of a MODULA compiler, to give us a language which supports multiple processes.

J.1 Process Structures (DeMillo, Lipton, Miller, Davida)

The principle result of our research this quarter has been the application of distributed computing technology to the traditional problems of operating system security. The key insight seems to be that it is desirable to separate those system functions which provide user services from those which mediate system security. By using encryption based protocols in a distributed system, many system functions can be supported with high security. Research is currently under way to increase the functionality of these designs. A byproduct of the effort is a unification of the theories of operating system and data security.

M.1 Establishment of FDPS Testbed Facility (Myers, Elshoff, Gaither, Flinn, Newell, Wice)

No significant progress to report.

M.2 Remote Load Emulator (Myers, Enslow, Forsyth, Howe)

The interpreter of script object code is near completion. Final enhancements and testing are now being undertaken.

M.3 FDOS Simulation Testbed (LeBlanc, Gaither, Maccabe, Myers, Newell, Wice)

A preliminary implementation of the simulation testbed has been completed, written in "C" on the PDP-11/45. This version is currently being extended and rewritten in RATFOR to run on the PR1ME P400's.

5. TRAVEL RELATED TO THE FDPS PROGRAM

Dates of Trip: 3-5 March, 1980
Individuals Traveling: Philip Enslow
Itinerary: Atlanta, Georgia
Purpose: Survey state-of-the-art in distributed processing equipment at AFIPS Office Automation Conference.

Dates of Trip: 5 March, 1980
Individuals Traveling: Nancy Lynch
Itinerary: Seattle, Washington
Purpose: Present talk at University of Washington about time analysis of distributed systems.

Dates of Trip: 14 March, 1980
Individuals Traveling: Philip Enslow
Itinerary: Baltimore, Maryland
Purpose: Discussion of decentralized distributed control.

Dates of Trip: 17 March, 1980
Individuals Traveling: Philip Enslow
Itinerary: Miami Beach, Florida
Purpose: Survey state-of-the-art in local network equipment and systems at INTERFACE '80 Exhibition.

Dates of Trip: 24-26 March, 1980
Individuals Traveling: Richard LeBlanc
Itinerary: Tallahassee, Florida
Purpose: Present paper at ACM Southeast Regional Conference entitled, "Research Issues in Fully Distributed Systems".

Dates of Trip: 2 April, 1980
Individuals Traveling: Philip Enslow
Itinerary: Baltimore, Maryland
Purpose: Discussions of decentralized distributed control.

Dates of Trip: 3-4 April, 1980
Individuals Traveling: Phillip Crews
Itinerary: Washington, D.C.
Purpose: Participate in organization and program selection for Fall COMPCON '80 Conference in Distributed Processing.

Dates of Trip: 13-14 April, 1980
Individuals Traveling: Nancy Lynch
Itinerary: Rochester, N.Y.
Purpose: Present talk at University of Rochester about time analysis of distributed systems.

Dates of Trip: 14-16 April, 1980
Individuals Traveling: Phillip Crews, Philip Enslow, Carolyn Greene
Itinerary: Ft. Lauderdale, Florida
Purpose: Present Georgia Tech FDPS Research Program, and participate in Third IBM University Research Seminar.

Dates of Trip: 14-18 April, 1980
Individuals Traveling: Richard DeMillo
Itinerary: Berkeley, California
Purpose: Attend IEEE Symposium on Security and Privacy.

Dates of Trip: 16-19 April, 1980
Individuals Traveling: Nancy Griffeth
Itinerary: Seattle, Washington
Purpose: Work with M. Fischer on paper, "Optimal Resource Placement in a Distributed System".

Dates of Trip: 27-30 April, 1980
Individuals Traveling: Nancy Lynch
Itinerary: Los Angeles, California
Purpose: Present the paper "Fast Allocation of Nearby Resources in a Distributed System" at the 1980 ACM Symposium on Theory of Computing.

Dates of Trip: 1-2 May, 1980
Individuals Traveling: Nancy Lynch
Itinerary: Palo Alto, California
Purpose: Consult with researchers in distributed computing at Xerox Palo Alto Research Center.

Dates of Trip: 13-15 May, 1980
Individuals Traveling: Philip Enslow
Itinerary: Rome, N.Y.
Purpose: Participate in the RADC Distributed Processing Technology Exchange Seminar, and presentations on the Georgia Tech FDPS Research Program and the project on Distributed Decentralized Control.

Dates of Trip: 14-16 May, 1980
Individuals Traveling: Nancy Griffeth
Itinerary: New York, N.Y.
Purpose: Present talk at New York University Symposium on Distributed Database Systems.

Dates of Trip: 19-22 May, 1980
Individuals Traveling: Richard DeMillo
Itinerary: Anaheim, California
Purpose: Attend NCC '80.

Dates of Trip: 28 May, 1980
Individuals Traveling: A.P. Jensen, John Gehl
Itinerary: Washington, D.C.
Purpose: Meet with Deputy Chief of Staff of Personnel, Department of the Army

6. VISITORS

Dates of Visit: 1 January - 31 March, 1980
Visitor: Michael Fischer, University of Washington
Purpose: Research Collaboration.
Individual Contacted: N. Lynch

Dates of Visit: 25 February - 14 March, 1980
Visitor: Arnold Schönhage, University of Tubingen, Germany
Purpose: Research collaboration on problems involving models for parallel computation, stochastic analysis of distributed systems, design of arbiter systems, and techniques for proving lower bounds for arbitration problems.
Individual Contacted: N. Lynch

Dates of Visit: 1-5 April, 1980
Visitor: Richard Lipton, University of California
Purpose: Research collaboration on distributed system security and synchronization.
Individual Contacted: R. DeMillo

Dates of Visit: 17 April, 1980
Visitor: William Boile, James Glymph, MILPERCEN
Purpose: Discuss OMF/EMP file structures.
Individual Contacted: A.P. Jensen

Dates of Visit: 22 April, 1980
Visitor: Admiral Albert J. Bacioco, Office of Naval Research
Purpose: Receive briefing on the Georgia Tech FDPS Research Program.
Individual Contacted: P. Enslow, R. DeMillo, N. Lynch, N. Griffeth, R. LeBlanc

Dates of Visit: 29 April, 1980
Visitor: Ray Spitz, Darrell Knaus, Gene Head, IBM General Systems Division
Purpose: Discuss the Georgia Tech FDPS research program.
Individual Contacted: P. Enslow

Dates of Visit: 27 May, 1980
Visitor: Charles Bass, Ungermann & Bass
Purpose: Discuss NETONE, a local communication network being developed by Ungermann & Bass.
Individual Contacted: P. Enslow

7. PUBLICATIONS

Author(s): P. Enslow, P. Crews
Title: IBM Series/1 UT200 RJE Workstation - Interim Progress Report Number 1
Type: Interim Progress Report [IBM Internal Use Only]
GIT Number: (unnumbered)
Publ. Date: 31 March, 1980

Author(s): A. Maccabe, R. LeBlanc
Title: A Language Model for Fully Distributed Systems
Type: Conference Paper
GIT Number: (unnumbered)
Publ. Date: (submitted March, 1980; accepted)

Author(s): N. Lynch
Title: Fast Allocation of Nearby Resources in a Distributed System
Type: Conference Paper
GIT Number: GIT-ICS-80/04
Publ. Date: April, 1980

Author(s): G. Davida, R. DeMillo, R. Lipton
Title: A System Architecture to Support a Verifiably Secure Multilevel Security System
Type: Conference Paper
GIT Number: GIT-ICS-80/05
Publ. Date: April, 1980.

Author(s): G. Davida, R. DeMillo, R. Lipton
Title: Secure Key Distribution
Type: Conference Paper
GIT Number: TBA
Publ. Date: April, 1980

Author(s): P. Enslow
Title: Quarterly Progress Report - Number 2
Type: Quarterly Progress Report
GIT Number: (unnumbered)
Publ. Date: April, 1980

Author(s): R. LeBlanc, J. Myers, S. Newell
Title: A Simulator for the Evaluation of Operating System Algorithms for Fully Distributed Systems
Type: Conference Paper
GIT Number: (unnumbered)
Publ. Date: (submitted April, 1980)

Author(s): A. Maccabe, R. LeBlanc
Title: Communication Facilities in Programming Languages for Fully Distributed Systems
Type: Conference Paper
GIT Number: (unnumbered)
Publ. Date: (submitted April, 1980)

THE GEORGIA INSTITUTE OF TECHNOLOGY

RESEARCH PROGRAM IN

FULLY DISTRIBUTED PROCESSING SYSTEMS

Quarterly Progress Report Number 4
1 June, 1980 - 31 August, 1980

November, 1980

Supported by

Office of Naval Research (ONR)
Contract: N00014-79-C-0873
GIT Project: G36-643

U.S. Air Force Rome Air Development Center (RADC)
Contract: F30602-78-C-0120
GIT Project: G36-649
GIT Project: G36-654

International Business Machines,
General Systems Division (IBM)
Agreement: GSD-210189
GIT Project: G36-648

National Science Foundation (NSF)
Contract: MCS77-28305
Subcontract from Univ. of Wisc.: 144-L729
GIT Project: G36-630

U.S. Army Research Office (ARO)
Contract: DAAG29-79-C-0155
GIT Project: G36-638

U.S. Army Institute for Research in
Management Information and Computer Science (AIRMICS)
Contract: DAAK70-79-D-0087
GIT Project: G36-647

School of Information and Computer Science
Georgia Institute of Technology
Atlanta, Georgia 30332

1. INTRODUCTION

This is the Fourth Quarterly Progress Report prepared on the Georgia Tech Research Program in Fully Distributed Processing Systems (FDPS).

a. Program Description.

The Georgia Tech Research Program in Fully Distributed Processing Systems is a comprehensive investigation of data processing systems in which both the physical and logical components are extremely loosely coupled while operating with a high degree of control autonomy at the component level. The definition of the specific class of multiple computer systems being investigated, and the operational characteristics and features of those systems is motivated by the desire to advance the state-of-the-art for that class of systems that will deliver a high proportion of the benefits currently being claimed for distributed processing systems. The scope of individual topics being investigated under this program ranges from formal modeling and theoretical studies to empirical examinations of prototype systems and simulation models. Also included within the scope of the program are areas such as the utilization of FDPS's and their interaction with management operations and structure.

b. Program Support.

The principle support for the program is a Selected Research Opportunity contract from the Office of Naval Research; however, there are a number of other sources of funding which also support the program. A complete list of these is given below. During this reporting period an additional project was assigned under the RADC contract.

Title: "Research on Fully Distributed Data Processing Systems"
Funding Agency: Office of Naval Research (ONR)
Contract Number: N00014-79-C-0873
GIT Project No.: G36-643
Principle Investigator: Philip H. Enslow, Jr.

Title: "Research on Distributed Control"
Funding Agency: U.S. Air Force Rome Air Development Center (RADC)
Contract Number: F30602-78-C-0120
GIT Project No.: G36-649
Principle Investigator: Philip H. Enslow, Jr.

Title: "Evaluation of Distributed Control Models"
Funding Agency: U.S. Air Force Rome Air Development Center (RADC)
Contract Number: F30602-78-C-0120
GIT Project No.: G36-654
Principle Investigator: Philip H. Enslow, Jr.

Title: "Agreement By and Between IBM and GTRI"
Funding Agency: International Business Machines, General Systems Division (IBM)
Contract Number: GSD Agreement Number 210189
GIT Project Number: G36-648
Principle Investigator: Philip H. Enslow, Jr.

Title: "Foundations of Deterministic Scheduling of Processes for Parallel Execution"
Funding Agency: National Science Foundation (NSF)
Contract Number: MCS77-28305
(Univ. of Wisc. subcontract number: 144-L729)
GIT Project Number: G36-630
Principle Investigator: Richard A. DeMillo

Title: "Theory of Systems of Asynchronous Parallel Processors"
Funding Agency: U.S. Army Research Office (ARO)
Contract Number: DAAG29-79-C-0155
GIT Project Number: G36-638
Principle Investigator: Nancy Lynch

Title: "Support of MILPERCEN Data Storage Concept"
Funding Agency: U.S. Army Institute for Research in Management Information and Computer Science (AIRMICS)
Contract Number: DAAK70-79-D-0087
GIT Project Number: G36-647
Principle Investigator: A.P. Jensen

2. ORGANIZATION AND STAFFING

Faculty

Crews, Phillip--Assistant Professor
DeMillo, Richard A.--Associate Professor
Enslow, Philip H. Jr.--Professor
Griffeth, Nancy--Assistant Professor
Jensen, Alton P.--Professor
LeBlanc, Richard--Assistant Professor
Livesey, Jon--Assistant Professor
(effective September, 1980)
Lynch, Nancy--Associate Professor

Staff

McDonell, Sharon--Sr. Secy.
Myers, Jeanette--Res. Scientist
Pinion, Nancy--Part-time Secy.

Students

There are 30 students working on various projects in the FDPS Research Program. Of these, 12 are in the Ph.D. program and 5 are preparing their MS Thesis on topics in FDPS.

3. CURRENT RESEARCH PROJECTS

The specific research projects have been organized into the major areas identified in the basic program proposal.

A. Theoretical and Formal Studies

A.1 Models of Asynchronous Processors
A.2 Decomposition of Parallel Systems
A.3 Reliable Systems
A.4 Time Performance of Distributed Systems
A.5 Audit Algorithms
A.6 Ticket Systems
A.7 Synchronous Simulation
A.8 Distributed Resource Allocation
A.9 Theory of Distributed Databases
A.10 Arbiter Design
A.11 Shared Memory Bounds for Synchronization Problems
A.12 Mutual Exclusion
A.13 Adaptive Distributed Resource Allocation Algorithms

B. Physical Interconnection and Networking

B.1 Heterogeneous Networking
B.2 Local Networking in Fully Distributed Processing Systems

C. Distributed Operating Systems

C.1 Decentralized and Distributed Control
C.2 Resource Allocation and Work Distribution in an FDPS
C.3 Fully Distributed Operating System - Initial Considerations
C.4 TBA (Distributed Operating Systems)
C.5 Process Support in Distributed Systems
C.6 Non-Homogeneous Operating Systems
C.7 FDOS - Preliminary Implementation Studies

D. Distributed Data Bases

D.1 Implementation of Distributed Database Systems
D.2 Support of MILPERCEN Data Storage Concept

E. Fault-Tolerance

F. Special Hardware to Support FDPS

G. Application of Distributed Processing

H. System Design Methodologies

H.1 FDPS Requirements Engineering Techniques
H.2 Coordinating Large Programming Projects

I. System Utilization

I.1 A Language for Distributed Programming
I.2 System Implementation Language Development
I.3 Experiments with a Distributed Compiler

J. Security

J.1 Process Structures

K. System Management

L. Evaluation and Comparison

M. FDPS Testbed

M.1 Establishment of FDPS Testbed Facility
M.2 Remote Load Emulator
M.3 Fully Distributed Operating System Simulation Testbed

4. SUMMARY OF PROGRESS

A.1 Models of Asynchronous Processors (Lynch, Fischer)

Final revisions were made to a paper entitled, "On Describing the Behavior and Implementation of Distributed Systems", scheduled to appear shortly in Theoretical Computer Science. Discussions were carried out aimed at extending time analysis techniques to communicating sequential process models for asynchronous processors.

A.2 Decomposition of Parallel Systems (Lynch, Fischer)

A manuscript entitled, "A Technique for Decomposing Algorithms which Use a Single Shared Variable", was completed and submitted in August for publication in the Journal of the ACM.

A.3 Reliable Systems (Lynch, Fischer, Lamport)

No significant progress to report

A.4 Time Performance of Distributed Systems (Lynch, Fischer, Lazowska, Schönhage)

Discussions were carried out for strengthening and clarifying bounds for a simple arbitration problem.

A.5 Audit Algorithms (Griffeth, Fischer, Lynch)

No significant progress to report

A.6 Ticket Systems (Fischer, Griffeth, Guibas, Lynch)

A simulation was designed and implemented for simulating the ticket system. It was implemented on the CYBER 74 using FORTRAN in order to take maximum advantage of the CYBER's speed and to avoid introducing overhead. The processor step times are normally distributed, with parameters mean and variance as input to the program; request arrival rate is exponentially distributed, with input parameter interarrival time. The results of the simulation suggested the following hypothesis about the behavior of the ticket system:

> As the expected interarrival time of requests for tickets increases, the expected response time also increases.

At first glance this seems counterintuitive, since larger interarrival times will mean less interference between processing of requests (in the extreme case, one ticket is processed before the next arrives). However, a different intuition explains why it is probably true. The intuition is that the sooner the requests arrive in the history of the system, the more information the system has at any given point in time about how the tickets should be allocated to requestors. Thus the hypothesis says that tickets are being allocated "intelligently" by the system, on the basis of whatever information it has.

A manuscript entitled, "Optimal Placement of Identical Resources in a Distributed Network", was submitted in August to next year's conference on distributed computation in Paris. This paper describes the optimal placement of resources in a distributed system. Simulation results suggest a very small expected running time for the corresponding dynamic case. A new theorem giving an analytic proof of this upper bound is currently being written up.

A.7 Synchronous Simulation (Lynch, Fischer, Arjomandi)

A paper is being prepared for conference submission.

A.8 Distributed Resource Allocation (Lynch)

An invited paper entitled, "Fast Allocation of Nearby Resources in a Distributed System", was submitted for publication in the special issue of the Journal of Computer and System Sciences, based on papers of the 1980 ACM Symposium on Theory of Computation.

A.9 Theory of Distributed Databases (Ghoudjehbaklou, Lynch)

More preliminary studies of problems arising in the distributed data-base area, suitable for time analysis, were carried out.

A.10 Arbiter Design (Lynch, Griffeth, Schönhage, Fischer)

No significant progress to report

A.11 Shared Memory Bounds for Synchronization Problems (Burns, Lynch)

No significant progress to report

A.12 Mutual Exclusion (Burns, Lynch)

A paper entitled, "Mutual Exclusion Using Indivisible Reads and Writes" was submitted to the Allerton Conference.

A.13 Adaptive Distributed Resource Allocation Algorithms (Ghoudjehbaklou, Lynch)

Several distributed resource-allocation problems and algorithms are being described and analyzed.

B.1 Heterogeneous Networking (Crews, Efruss, Greene, Ma, Ramirez)

The primary work accomplished during this period has been refinement of the programs already developed and analysis of the options available for the operational environment.

B.2 Local Networking in FDPSs (Enslow)

The operational capabilities desired in the local network testbed were established. The equipment selected was NETONE produced by Ungermann and Bass. NETONE is a baseboard-contention system (similar to ETHERNET). The user interfaces to NETONE are programmable, making it extremely useful as a research vehicle. Initial funding for the system was obtained. Delivery anticipated in October.

C.1 Decentralized and Distributed Control (Enslow, LeBlanc, Crews, Saponas)

The draft final report on the first phase of this study, a survey of decentralized control models, was prepared. The second phase, evaluation of these models, was initiated.

C.2 Resource Allocation and Work Distribution in an FDPS (Enslow, Sharp)

No progress to report due to the absence of Mr. Sharp during this period. He will return and the project will resume in September.

C.3 FDOS - Initial Considerations (Enslow, LeBlanc, Crews, Akin, Flinn, Forsyth, Fukuoka, Myers, Pitts, Saponas, Skowbo, Spafford, Wice)

Work proceeds on defining the desired and required capabilities for a Fully Distributed Operating System. Implementation experiments have been initiated.

C.4 TBA - Distributed Operating Systems (Livesey)

No activity to report. Dr. Livesey will join the Georgia Institute of Technology and the Project in September.

C.5 Communications Support for Distributed Systems (Enslow, Skowbo, Wice)

Further study of communications systems and network architectures is in progress. A list of services which may be provided at one or more levels in a hierarchy of protocols has been developed. Methods for assessing the effects of these services on system performance are being considered. The existing simulator for interprocess communication is being evaluated as a possible basis for further development in this project. Specific proposals for additional work are being evaluated and drafts are being written.

C.6 Non-Homogeneous Operating Systems (Ratzel)

No significant progress to report.

C.7 FDOS - Preliminary Implementation Studies (Myers, Enslow, Gaither, Newell, Wice)

Work continues on this project in support of Project C.3 and in preparation for actual implementation of a prototype FDOS.

D.1 Implementation of Distributed Database Systems (Griffeth)

A literature review and a bibliography of distributed database systems were begun. A plethora of algorithms for concurrency control were noted. Also, a variety of measures of "goodness" of such algorithms have been proposed. The area seems ripe for performance comparisons of the algorithms along a variety of axes. To this end, a detailed study of the concurrency algorithms has been begun. A second area in which some work (less than it warrants) was noted is the area of the allocation of data on a distributed system. Neglected areas are query procesing on distributed systems, recovery and reliability issues, and distributed directories.

The simulation program developed for the ticket system (A.6) was generalized to allow simulation of any distributed algorithm having the following properties:

(1) Each node uses the same algorithm;
(2) The number of ports at a node is bounded as the system size grows;
(3) Communication between each pair of nodes is via a mailbox which must be explicitly accessed by a node to find out if anything new has been sent.

A generalization of the ticket system simulation can thus be used to evaluate the concurrency control algorithms.

D.2 Support of MILPERCEN Data Storage Concept (Jensen, Doyle, Gehl, Bingham)

The work in this quarter consisted of four major activities. First, Pete Jensen visited MILPERCEN July 16 to present the results of the research done to date. Second, Captain Steve Ratzel and Jim Doyle visited USAREC in St. Louis to investigate the reserves personnel data base. Third, the phase-one report was written and sent to selected industry, academic, and military personnel in preparation for a planned October conference on human resource management. Fourth, a data element list of approximately 4800 data elements dealing with personnel data was compiled from 23 files and 65 transactions involving MILPERCEN, USAFAC, USAREC, and SIDPERS.

H.1 FDPS Requirements Engineering Techniques (Underwood, Corley)

No activity to report. Mr. Corley is expected to return to Georgia Tech in January, 1981, to complete the work on his Ph.D.

H.2 Coordinating Large Programming Projects (Enslow, Smith)

Questionnaire refinement was completed. The focus during this period was on completing the explanatory material. Progress was also made in identifying possible subjects.

I.1 A Language for Distributed Programming (LeBlanc, Maccabe, Hardin)

The development of features for our language for distributed programming has begun. Preparation for our presentation at COMPCON Fall '80 is in progress. A conference paper concerning the use of abstraction in our language has been submitted to L'Exposition de l'Informatique en Louisiana (see publications list). Another paper is being prepared for submission to the Second International Conference on Distributed Computing.

I.2 System Implementation Language Development (LeBlanc, Akin)

An implementation of an extended version of MODULA, a language which supports multiple processes, is now in progress.

I.3 Experiments with a Distributed Compiler (LeBlanc, Moore)

In order to test the feasibility of distributed programming, experiments have been designed involving several versions of a compiler. A compiler was chosen because there is considerable potential for taking advantage of parallelism in the compilation process. The experimental work is currently in progress.

J.1 Process Structures (DeMillo, Lipton, Miller, Davida)

Work continues on the application of cryptographic protocols and distributed computing to computer system security. A major output of this quarter's research is a survey of applicable cryptographic and operating system security theory. Work is beginning on the design of a prototype system. In addition, an extensive study of distributed system survivability, using statistical designs, was begun.

M.1 Establishment of FDPS Testbed Facility (Myers, Gaither, Flinn, Newell, Wice)

Work continues on simulations as well as general-purpose support software.

M.2 Remote Load Emulator (Myers, Enslow, Forsyth)

During this quarter, code generation routines were added to the script preprocessor and large parts of the script interpreter were implemented. Status of the various modules of the project is as follows: The script preprocesor is fully implemented. It does not yet recover from all syntax and semantic errors, although it does diagnose them and it generates correct object code for error-free scripts. The interpreter is implemented and partially tested. It can now access test systems through either asynchronous lines or through X.25 virtual circuits. The emulation session analysis programs are still being designed.

M.3 FDOS Simulation Testbed (LeBlanc, Gaither, Maccabe, Myers, Newell, Wice)

The simulator has been rewritten in RATFOR and is now running on our PR1ME computers (the initial version was written in C).

5. TRAVEL RELATED TO THE FDPS PROGRAM

Date of Trip: 8 July, 1980
Individual(s) Traveling: P.H. Enslow
Itinerary: Baltimore, Maryland
Purpose: Present one-day executive-level seminar on distributed data processing - research and market.

Date of Trip: August, 1980
Individual(s) Traveling: N. Lynch
Itinerary: Seattle, Washington
Contact: M. Fischer
Purpose: Worked with Mike Fischer on several of the listed projects. Presented talk on lower bounds for synchronization problems.

Date of Trip: 19-21 August, 1980
Individual(s) Traveling: R.A. DeMillo
Itinerary: Princeton, N.J.
Contact: Richard J. Lipton
Purpose: Consult in various topics relating to joint work in distributed systems.

6. VISITORS

Dates of Visit: 15-18 July, 1980
Visitor: George I. Davida
Contact: R.A. DeMillo
Purpose: Discuss joint work on cryptographic protocols

Dates of Visit: 4-8 August, 1980
Visitor: Peter Lauer
Contact: P.H. Enslow, N. Lynch, & other FDPS team members
Purpose: Discuss common interests and work in distributed processing

Dates of Visit: 21 August, 1981
Visitor: Mont Bernstein, System Development Corp.
Contact: P.H. Enslow
Purpose: Discuss SDC work in distributed processing and other common interests.

7. PUBLICATIONS

Author(s): T. Saponas & P. Crews
Title: A Model for Distributed and Decentralized Control in a Fully Distributed Processing System
Type: conference paper
Status: presented at Fall COMPCON '80, September 23-25, 1980

Author(s): R. LeBlanc
Title: Control and Communication Abstraction in a Programming Language for Distributed Systems.
Type: conference paper
Status: accepted for presentation

Author(s): N. Lynch
Title: Fast Allocation of Nearby Resources in a Distributed System
Type: conference paper
Status: invited & submitted

Author(s): N. Lynch & M. Fischer
Title: A Technique for Decomposing Algorithms which Use a Single Shared Variable
Type: journal article
Status: submitted

Author(s): N. Lynch & M. Fischer
Title: On Describing the Behavior & Implementation of Distributed Systems
Type: journal article
Status: final revisions made

Author(s): M. Fischer, N. Griffeth, L. Guibas, & N. Lynch
Title: Optimal Placement of Identical Resources in a Distributed Network
Type: conference paper
Status: submitted

Author(s): J. Burns & N. Lynch
Title: Mutual Exclusion Using Indivisible Reads and Writes
Type: conference paper
Status: submitted

Author(s): R.A. DeMillo
Title: A Brief Overview of Computer System Security
Type: report prepared for U.S. Army CORADCOM, Ft. Monmouth, N.J.
Status: delivered

Author(s): G.I. Davida, R.A. DeMillo, R.J. Lipton
Title: Protecting Shared Cryptographic Keys
Type: conference paper
Status: presented at 1980 IEEE Symposium on Security and Privacy

Author(s): G.I. Davida, R.A. DeMillo, R.J. Lipton
Title: A System Architecture to Support a Verifiably Secure Multilevel Security System
Type: conference paper
Status: published

Author(s): R.A. DeMillo
Title: Cryptographic Protocols
Type: conference paper
Status: accepted

Author(s): P.H. Enslow
Title: Quarterly Progress Report - Number 3
Type: Quarterly Progress Report
Status: printed & distributed
Publ. Date: July, 1980

APPENDIX

A report of work conducted by Professor Michael Fischer in cooperation with the Georgia Tech Research Program in Fully Distributed Processing Systems.

Design and Analysis of Distributed Algorithms
Second Quarterly Progress Report (March 1 - May 31, 1980)
Professor Michael J. Fischer - Principal Investigator

1. INTRODUCTION

This report covers the progress of the project, "Design and Analysis of Distributed Algorithms," directed by Professor Michael J. Fischer, for the period March 1 - May 31, 1980. This project is principally supported by ONR Contract N00014-80-C-0221.

From the beginning of the reporting period through April 2, part of the work was carried out at Georgia Institute of Technology where Professor Fischer was visiting. During that time, additional support for the project came from ONR Contract N00014-79-C-0873 through a subcontract from the Georgia Institute of Technology to the University of Washington.

2. TECHNICAL COLLABORATORS

The work reported on in Sections 4.1 and 4.2 was carried out in close collaboration with Professors Nancy A. Lynch and Nancy Griffeth of the Georgia Institute of Technology. Professor Griffeth also spent three days in Seattle in mid-April working with Professor Fischer after his return from Atlanta.

Professor Arnold Schönhage, University of Tübingen, Germany, visited the Georgia Institute of Technology for the period February 25 - March 14, 1980, and contributed greatly to the research.

3. PUBLICATIONS

No work on this contract has yet reached publication. References [2] and [3] were submitted to conferences.

4. TECHNICAL PROGRESS

The first quarterly progress report listed six areas in which investigations had been begun:

1. Decomposition of Parallel Systems
2. Reliable Systems
3. Time Analysis
4. Audit Algorithms
5. Ticket Systems
6. Synchronous Simulation

During the second quarter, work focused on areas (3) and (5). In addition, work begun on another project continued on developing a logic of concurrent processes.

4.1 TIME ANALYSIS

Working with Professors Schönhage, Lynch, and Griffeth, we continued to use the simple arbiter problem of [1] as a paradigm for the time analysis of distributed algorithms. We have taken three different approaches.

Worst-case analysis. The tree arbiter of [1] is an algorithm for allocating a single reusable resource among n competing users. The "lost" time of such a system is the total amount of time during which at least one user is requesting the resource but the resource is free. A simple worst-case analysis of the tree arbiter shows that the total lost time in processing r requests is at most $O(r \log n)$. This bound is independent of how the individual processes are scheduled, as long as each process takes at least one step during each unit interval of time. It is also independent of the arrival rate of the requests.

On the other hand, a simple polling arbiter (also described in [1]) has much better performance under a heavy load and much worse under a light load. These results led to a reexamination of the tree arbiter and the discovery of a new, slightly different, tree arbiter which combines the best features of both algorithms: under light loads the lost time per request is $O(\log n)$ and under heavy loads the lost time drops to a constant. The resulting algorithm is easy to implement and might be quite attractive in certain practical applications.

Lower bounds. The previous analysis leads naturally to the question of whether a still better tree arbiter exists, say one with constant lost time per request, independent of the load. Under a suitably restricted model of distributed computation, we are able to answer that question in the negative by proving that every distributed arbiter algorithm in that model must take time $\Omega(\log n)$ between receiving a request at some leaf and granting that request, even if there are no other requests present in the system. The proof of this fact follows from a (nontrivial) formalization of the notion that it takes $\Omega(\log n)$ steps for information to be broadcast throughout a network in which there are natural bounds on the number of neighbors of a node. The model unfortunately contains some technical restrictions that are somewhat unnatural but we believe also inessential. We believe this result can be extended to more natural models as well.

Expected-time analysis. Worst-case analysis can sometimes lead to unrealistically large time bounds which in fact occur with only very low probability. As an alternative, one can assume certain probability distributions on the parameters affecting the behavior of the system and then analyze the expected time of the system under those assumptions.

The resulting Markov models are generally too complicated to analyze exactly. Two approaches around that problem are to construct approximate models which can be analyzed exactly and to perform simulations. We began work on approximate models with Professor Lazowska during the previous quarter and explored it a little further with Professor Schönhage. We also sketched the outline of a computer program to carry out a simulation of arbiter systems in a reasonably efficient way, which we plan to implement.

4.2 TICKET SYSTEMS

We continued the work on algorithms designed to distribute large numbers of "tickets" at widely distributed "ticket windows". The ticket problem can be thought of as a generalization of the simple arbiter problem to more than one resource, and one has a choice of whether or not to allow tickets to be returned and later resold.

The first phase of our work involved thinking of various clever ways of matching up requests to tickets in a distributed network. After generating several different algorithms for this problem, it became apparent that we had little basis for evaluating the relative goodness of the proposed methods, and it was unclear what advantages, if any, the more esoteric solutions had.

We then turned to the problem of analyzing the performance of a very simple abstract ticket algorithm -- one in which the tickets are placed initially on nodes of a tree and not moved thereafter except to fulfill particular requests. Taking the expected distance between a request and the ticket which fulfills it as a performance measure, we were able to prove a constant upper bound. Thus, even without being clever about dynamically rearranging tickets during the processing in response to the requests that have already been processed, one still gets a "best possible" kind of performance bound.

The expected distance is not the same as expected waiting time, however, because we have not specified a mechanism whereby a request finds the optimal ticket to match with. This led to the development of a complete algorithm in which each node uses only local information in its operation, yet every request is guaranteed to eventually find a ticket. We hope this algorithm will be a close enough approximation to the simple abstract algorithm to still achieve a constant expected waiting time.

4.3 A LOGIC OF CONCURRENT PROCESSES

Karl Abrahamson is doing dissertation work under Professor Fischer on logics of concurrent processes. Programming logics are systems of mathematical logic to be used in making assertions and reasoning about computer programs. Previous work in this area by others has focused on termination properties of programs, that is, under what conditions will a program terminate and what will be true when it does? He extends that work by permitting assertions about a program in the midst of execution. This useful for understanding the behavior of any program but essential in analyzing systems of concurrent processes.

He is taking three approaches to process logic, developing several specific logics based on those approaches, and analyzing and comparing them with each other and with recent work by other researchers. The first approach is to extend program logics by adding Boolean variables. The second is to build on predicate calculus by using explicit time quantifiers. The third, based on modal logic, eliminates time variables in favor of a collection of carefully chosen modalities.

5. REFERENCES

1. N.A. Lynch and M.J. Fischer, "On Describing the Behavior and Implementation of Distributed Systems," University of Washington Technical Report 79-06-01 (Revised March 1980) and Georgia Institute of Technology Technical Report GIT-ICS-79/03. See also Lecture Notes in Computer Science, Semantics of Concurrent Computation, Springer-Verlag (1979), 147-171. Also submitted for publication in Theoretical Computer Science.

2. M.J. Fischer, N.D. Griffeth and N.A. Lynch, "Optimal Resource Placement in a Distributed System (extended abstract)," Manuscript (April 25, 1980).

3. M.J. Fischer, "On Developing a Theory of Distributed Computing: Summary of Current Research," Manuscript (May 1980).

THE GEORGIA INSTITUTE OF TECHNOLOGY

RESEARCH PROGRAM IN

FULLY DISTRIBUTED PROCESSING SYSTEMS

Quarterly Progress Report Number 5
1 September, 1980 - 30 November, 1980

December, 1980

Supported by

Office of Naval Research (ONR)
Contract: N00014-79-C-0873
GIT Project: G36-643

U.S. Air Force Rome Air Development Center (RADC)
Contract: F30602-78-C-0120
GIT Project: G36-649
GIT Project: G36-654

International Business Machines,
General Systems Division (IBM)
Agreement: GSD-210189
GIT Project: G36-648

National Science Foundation (NSF)
Contract: MCS77-28305
Subcontract from Univ. of Wisc.: 144-L729
GIT Project: G36-630

U.S. Army Research Office (ARO)
Contract: DAAG29-79-C-0155
GIT Project: G36-638

U.S. Army Institute for Research in
Management Information and Computer Science (AIRMICS)
Contract: DAAK70-79-D-0087
GIT Project: G36-647

School of Information and Computer Science
Georgia Institute of Technology
Atlanta, Georgia 30332

1. INTRODUCTION

This is the Fifth Quarterly Progress Report prepared on the Georgia Tech Research Program in Fully Distributed Processing Systems (FDPS).

a. Program Description.

The Georgia Tech Research Program in Fully Distributed Processing Systems is a comprehensive investigation of data processing systems in which both the physical and logical components are extremely loosely coupled while operating with a high degree of control autonomy at the component level. The definition of the specific class of multiple computer systems being investigated, and the operational characteristics and features of those systems is motivated by the desire to advance the state-of-the-art for that class of systems that will deliver a high proportion of the benefits currently being claimed for distributed processing systems. The scope of individual topics being investigated under this program ranges from formal modeling and theoretical studies to empirical examinations of prototype systems and simulation models. Also included within the scope of the program are areas such as the utilization of FDPS's and their interaction with management operations and structure.

b. Program Support.

The principle support for the program is a Selected Research Opportunity contract from the Office of Naval Research; however, there are a number of other sources of funding which also support the program. A complete list of these is given below. During this reporting period RADC Project G36-649 was completed.

Title: "Research on Fully Distributed Data Processing Systems"
Funding Agency: Office of Naval Research (ONR)
Contract Number: N00014-79-C-0873
GIT Project No.: G36-643
Principle Investigator: Philip H. Enslow, Jr.

Title: "Research on Distributed Control"
Funding Agency: U.S. Air Force Rome Air Development Center (RADC)
Contract Number: F30602-78-C-0120
GIT Project No.: G36-649
Principle Investigator: Philip H. Enslow, Jr.

Title: "Evaluation of Distributed Control Models"
Funding Agency: U.S. Air Force Rome Air Development Center (RADC)
Contract Number: F30602-78-C-0120
GIT Project No.: G36-654
Principle Investigator: Philip H. Enslow, Jr.

Title: "Agreement By and Between IBM and GTRI"
Funding Agency: International Business Machines, General Systems Division (IBM)
Contract Number: GSD Agreement Number 210189
GIT Project Number: G36-648
Principle Investigator: Philip H. Enslow, Jr.

Title: "Foundations of Deterministic Scheduling of Processes for Parallel Execution"
Funding Agency: National Science Foundation (NSF)
Contract Number: MCS77-28305
(Univ. of Wisc. subcontract number: 144-L729)
GIT Project Number: G36-630
Principle Investigator: Richard A. DeMillo

Title: "Theory of Systems of Asynchronous Parallel Processors"
Funding Agency: U.S. Army Research Office (ARO)
Contract Number: DAAG29-79-C-0155
GIT Project Number: G36-638
Principle Investigator: Nancy Lynch

Title: "Support of MILPERCEN Data Storage Concept"
Funding Agency: U.S. Army Institute for Research in Management Information and Computer Science (AIRMICS)
Contract Number: DAAK70-79-D-0087
GIT Project Number: G36-647
Principle Investigator: A.P. Jensen

2. ORGANIZATION AND STAFFING

Faculty

Crews, Phillip--Assistant Professor
Davida, George--Professor
DeMillo, Richard A.--Associate Professor
Enslow, Philip H. Jr.--Professor
Griffeth, Nancy--Assistant Professor
Jensen, Alton P.--Professor
LeBlanc, Richard--Assistant Professor
Livesey, Jon--Assistant Professor
Lynch, Nancy--Associate Professor
Miller, Raymond--Professor

Staff

McDonell, Sharon--Sr. Secy.
Myers, Jeanette--Res. Scientist
Pinion, Nancy--Part-time Secy.

Students

There are 30 students working on various projects in the FDPS Research Program. Of these, 12 are in the Ph.D. program and 5 are preparing their MS Thesis on topics in FDPS.

3. CURRENT RESEARCH PROJECTS

The specific research projects have been organized into the major areas identified in the basic program proposal.

A. Theoretical and Formal Studies

A.1 Models of Asynchronous Processors
A.2 Decomposition of Parallel Systems
A.3 Reliable Systems
A.4 Time Performance of Distributed Systems
A.5 Audit Algorithms
A.6 Ticket Systems
A.7 Synchronous Simulation
A.8 Distributed Resource Allocation
A.9 Theory of Distributed Databases
A.10 Arbiter Design
A.11 Shared Memory Bounds for Synchronization Problems
A.12 Mutual Exclusion
A.13 Adaptive Distributed Resource Allocation Algorithms
A.14 Using Complementary Distributed System Models
A.15 Probabilistic Algorithms in Distributed Systems

B. Physical Interconnection and Networking

B.1 Heterogeneous Networking
B.2 Local Networking in Fully Distributed Processing Systems

C. Distributed Operating Systems

C.1 Decentralized and Distributed Control
C.2 Resource Allocation and Work Distribution in an FDPS
C.3 Fully Distributed Operating System - Initial Considerations
C.4 Local Operating System
C.5 Communications Support for Distributed Systems
C.6 Non-Homogeneous Operating Systems
C.7 FDOS - Preliminary Implementation Studies

D. Distributed Data Bases

D.1 Implementation of Distributed Database Systems
D.2 Support of MILPERCEN Data Storage Concept
D.3 Implementation of the Audit Algorithm

E. Fault-Tolerance

F. Special Hardware to Support FDPS

G. Application of Distributed Processing

H. System Design Methodologies

H.1 FDPS Requirements Engineering Techniques
H.2 Coordinating Large Programming Projects

I. System Utilization

I.1 A Language for Distributed Programming
I.2 System Implementation Language Development
I.3 Experiments with a Distributed Compiler

J. Security

J.1 Process Structures

K. System Management

L. Evaluation and Comparison

L.1 Simulation of Distributed Algorithms (Griffeth, Lynch)

M. FDPS Testbed

M.1 Establishment of FDPS Testbed Facility
M.2 Remote Load Emulator
M.3 Fully Distributed Operating System Simulation Testbed

4. SUMMARY OF PROGRESS

A.1 Models of Asynchronous Processors (Lynch, Fischer)

No significant progress to report

A.2 Decomposition of Parallel Systems (Lynch, Fischer)

No significant progress to report

A.3 Reliable Systems (Lynch, Fischer, Lamport)

No significant progress to report

A.4 Time Performance of Distributed Systems (Lynch, Fischer, Lazowska, Schönhage)

No significant progress to report

A.5 Audit Algorithms (Griffeth, Fischer, Lynch)

Existing methods of reading a consistent state of a distributed database require either (a) stopping all transactions or (b) requesting an out-of-date version of the database. Neither of these is necessary. We have developed conditions for a set of artificial events in the system history to "represent" a consistent state of the database, in the following sense:

> Each node can compute a node state, using knowledge of the artificial events occurring at the node (i.e., without additional communication), in such a way that the set of all node states thus computed is a consistent system state.

The usual definition of consistency in a database is "serializability". Our notion of consistency is a generalized version of serializability: each transaction occurs either entirely before or entirely after the read of the DDB state, although the transactions need not be serialized with respect to each other. If the transactions happen to be serialized, then slightly more restrictive conditions on the set of artificial events gives a system state corresponding to the first N transactions in the serialization.

A paper is being prepared for submission to a journal. Also, implementation of the algorithm on the PR1ME is being examined.

A.6 Ticket Systems (Fischer, Griffeth, Guibas, Lynch)

More experiments on the simulation program verify the hypothesis that average response time increases with interarrival time. Perturbations in the results were found to be due to the polling cycle of the nodes.

A graphical display of the behavior of the ticket system is being prepared, using the simulation program. This type of display is suggested as a tool for developing intuition about distributed systems and finding counterexamples to hypotheses.

A.7 Synchronous Simulation (Lynch, Fischer, Arjomandi)

A conference version of a paper entitled, "A Difference in Efficiency Between Synchronous and Asynchronous Systems", was completed and submitted for possible presentation at next year's ACM Symposium on Theory of Computing. Results have been clarified and improved, and the main proof has been distilled into an interesting and clean graph-theoretic argument. The paper describes a synchronization problem for which there is a provable multiplicative factor of log n (n = the number of "ports" in the distributed system) distinguishing the achievable time complexity of this problem when implemented on a synchronous or on an asynchronous system. This is the first such result we know of, and seems to be quite fundamental to understanding the differences in capability between these two types of parallelism.

A.8 Distributed Resource Allocation (Lynch)

No significant progress to report

A.9 Theory of Distributed Databases (Lynch, Griffeth)

A seminar for interested faculty and graduate students was conducted, carrying out a careful study of recent work on distributed data base concurrency control. Plans for a continuation next quarter were made; the next quarter will involve simulation and other empirical evaluation of some of the algorithms.

A.10 Arbiter Design (Lynch, Griffeth, Schönhage, Fischer)

No significant progress to report

A.11 Shared Memory Bounds for Synchronization Problems (Burns, Lynch)

Revisions of Jim Burns's doctoral dissertation have been carried out; in particular, results on number-of-message bounds for distinguishing processes in a ring network have been clarified. The dissertation appears now to be virtually completed.

A.12 Mutual Exclusion (Burns, Lynch)

A paper entitled, "Mutual Exclusion Using Indivisible Reads and Writes", was presented at the Allerton Conference.

A.13 Adaptive Distributed Resource Allocation Algorithms (Ghoudjehbaklou, Lynch)

Several distributed resource-allocation problems and algorithms are being described and analyzed.

A.14 Using Complementary Distributed System Models (Lynch, Rounds, Miller)

Preliminary discussions were conducted for a project aimed at using a combination of several distributed system models to describe and prove correctness and performance properties of distributed algorithms. The intention is to describe a single algorithm at several levels (related by simulation mappings), proving safety properties at one (high) level, and fairness and performance properties at another (lower) level. The models to be used are Hoare's Communicating Sequential Processes model, Ladner's automata-theoretic communication model, Rounds's and Brooke's communication event model, and the Lynch-Fischer process-variable model.

A.15 Probabilistic Algorithms in Distributed Systems (Lynch, Arjomandi, Fischer)

Preliminary discussions were conducted for a project aimed at evaluating the performance of distributed algorithms which use controlled randomness to avoid conflicts.

B.1 Heterogeneous Networking (Crews, Efruss, Greene, Ma, Ramirez)

The Series/1 has been used in the following projects under the EDX operating system:

1. Training of students and demonstrations of the facilities of the system to staff and faculty. The training of students has involved support for the ICS 3422 class to run programs written in PL/I using the Session Manager facility of EDX, familiarization sessions with students interested in the system, and formal training for Susanna Ma so that she can help support EDX in the future.
2. Installation of EDX, Version 3.
3. Installation of the Software Tools package according to the "Cookbook" specification obtained through Dan Forsyth.

Since September, work on the Series/1, operating under the RPS operating system, has concentrated on developing a UT200 task to enable communications between the Series/1 and the Cyber 74. Although the design and the majority of the coding for this design had been previously completed, problems were encountered with the design due to inconsistencies in the UT200 protocol that could not be resolved without including a large amount of unreliability and complexity in the resulting system. A new design was then considered and accepted which bypassed the problems previously encountered. The code implementing this design is written, with the exception of portions of a user command-handler, and the remaining routines are being tested.

B.2 Local Networking in FDPSs (Enslow)

The local network hardware was ordered and delivered. An 11-node NETONE system produced by Ungermann and Bass is being installed (this is a baseboard contention system somewhat similar to Ethernet). This network will be used to provide access to all of the systems in the FDPS testbed as well as to experiment with computer-to-computer communication utilizing a front-end local network (there is already a high-speed coaxial cable ring network (8 Mbps) providing a back-end local network to interconnect the five PR1ME computers in the testbed). The software for normal access function is now being checked out. Perhaps the most important characteristic of the NETONE system is that it is totally "soft" except for the coax cable interface and can be reprogrammed by the operator. Cable installation has begun while the system is being checked out.

C.1 Decentralized and Distributed Control (Enslow, LeBlanc, Crews, Saponas, Hopkins)

The draft of the final report on the first phase of this study, "A Survey

of Distributed and Decentralized Control Models", was completed and submitted to the sponsors for review (publication anticipated in January, 1981). The work on the second phase of the project, an evaluation of these models, has focused on the development and implementation of an appropriately instrumented control model simulator.

C.2 Resource Allocation and Work Distribution in an FDPS (Enslow, Sharp)

An expanded draft version of a working paper on this subject has been completed and is in the process of being edited.

C.3 FDOS - Initial Considerations (Enslow, Livesey, LeBlanc, Crews, Akin, Flinn, Forsyth, Fukuoka, Maccabe, Myers, Pitts, Saponas, Skowbo, Spafford, Wice)

No significant progress to report

C.4 Local Operating System (Livesey, LeBlanc, Spafford)

A study has been begun into the design requirements for a Local Operating System (LOS) to support Fully Distributed Processing. The host hardware (PR1ME P400's & P500's) has been studied and a design document for the local operating system is underway. Additional documentation on the instruction set of the host hardware has been written to aid in implementation of the LOS.

C.5 Communications Support for Distributed Systems (Enslow, Skowbo, Wice)

A highly-distributed feedback mechanism for adaptive routing in a packet-switched network of very loosely-coupled systems has been outlined. Procedures for a highly-selective acknowledgement protocol have been developed as a possible basis for this feedback, and as a more general error-control mechanism. Further development and testing awaits implementation of proposed enhancements to the simulator for interprocess communication, which was found to have a number of limitations for this research. Specific enhancements will provide for the simulation of:

1. Acknowledgement Protocols as outlined above
2. Flow control Protocols
3. Pseudo-random message generation of varying size and frequency
4. Priority transmission
5. Delays due to propagation and bandwidth characteristics
6. Errors due to message loss or corruption, and to link failure

C.6 Non-Homogeneous Operating Systems (Ratzel)

This project has been cancelled

C.7 FDOS - Preliminary Implementation Studies (Myers, Enslow, Wice)

Work continues on this project in support of Project C.3 and in preparation for actual implementation of a prototype FDOS.

D.1 Implementation of Distributed Database Systems (Griffeth, Livesey, Lynch)

A study is in progress of the communication system and operating system facilities required for simulation and implementation of concurrency control algorithms.

D.2 Support of MILPERCEN Data Storage Concept (Jensen, Doyle, Gehl, Bingham)

A workshop, "Implications of Data Base Technology for Human Resource Information Management", was sponsored by AIRMICS and Georgia Tech as partial fulfillment of Phase Two of the MILPERCEN project. The workshop brought together people representing the military, corporate/industry, and academic communities who are interested in personnel and human resource management.

An overview of the Army manpower management systems was presented along with the data management problems concerning military personnel. The corporate/industry representatives gave presentations of how they are managing and developing manpower planning systems for use in operations and strategic planning. The academic representatives presented current research in date base architecture, data base administration, and distributed computing relating these topics to the problem of military personnel management. Presentations focused on issues of data base technology, extremely large files (greater than one million records), and other areas of human resource management systems including promotion, design, implementation, economics, and management of such systems.

After this exchange of ideas and views among the academic, corporate/industry, and military representatives, the workshop as a whole discussed the issues contained in the Phase I report. Smaller groups were formed to work out more specific solutions to the issues discussed. These solutions were then presented to the group as a whole for approval and more discussion.

D.3 Implementation of the Audit Algorithm (Griffeth, Livesey, Lynch)

Preliminary discussions have begun.

H.1 FDPS Requirements Engineering Techniques (Underwood, Corley)

No activity to report. Mr. Corley is expected to return to Georgia Tech in January, 1981, to complete the work on his Ph.D.

H.2 Coordinating Large Programming Projects (Enslow, Smith)

The initial form of the questionnaire has been tested by presenting it to several local data processing managers. A revision of the questionnaire is being developed based on the results of those interviews.

I.1 A Language for Distributed Programming (LeBlanc, Maccabe, Hardin)

Initial design of language features for support of distributed programming has been completed. Example programs are being written in order to test the usefulness of the design. The design of a prototype implementation has recently been started. LeBlanc and Maccabe traveled to Washington to present a paper at COMPCON Fall '80. A paper was submitted to the Second International Conference on Distributed Computing. Final revisions were made in a paper to be presented at L'Exposition de l'Informatique en Louisiana.

I.2 System Implementation Language Development (LeBlanc, Akin)

Work on the implementation of our extended version of MODULA continued during this quarter. The front-end scanner and parser for an optimizing compiler has been completed, and a general-purpose back-end code generator is under development.

I.3 Experiments with a Distributed Compiler (LeBlanc, Moore)

The experimental phase of this project has been completed. During this quarter, Moore has been analysing the data collected, and writing his M.S. thesis based on this work.

J.1 Process Structures (DeMillo, Lipton, Miller, Davida, Livesey)

Work on the design of secure operating systems continues. The degree of distribution of the system and the level of security achievable is being investigated. Encryption and appropriate protocols are being investigated for incorporation into the design of the local operating system. In particular, we are investigating the possibility of multiplexing processors between mutually secure processes.

L.1 Simulation of Distributed Algorithms (Griffeth, Lynch)

Preliminary discussions have begun

M.1 Establishment of FDPS Testbed Facility (Myers, Flinn, Wice)

No significant progress to report

M.2 Remote Load Emulator (Myers, Enslow, Forsyth)

No significant progress to report

M.3 FDOS Simulation Testbed (LeBlanc, Hopkins, Maccabe, Myers, Wice)

Work continues in support of Projects C.1, C.2, & C.5.

5. TRAVEL RELATED TO THE FDPS PROGRAM

Date of Trip: September, 1980
Individual(s) Traveling: N. Lynch
Itinerary: New York, N.Y.
Contact: presentation attendees at Columbia University
Purpose: Give invited presentation on Projects A.6 & A.8.

Date of Trip: 15-17 September, 1980
Individual(s) Traveling: P.H. Enslow
Itinerary: Technical University, Loughborough, England
Contact: course attendees
Purpose: Present invited talk on "Parallel Control in Distributed Systems" at Advanced Course on Parallel Processing

Date of Trip: 22-25 September, 1980
Individual(s) Traveling: R.J. LeBlanc & A.B. Maccabe
Itinerary: Washington, D.C.
Purpose: Attend COMPCON Fall '80 and present a paper based on work in Project I.1.

Date of Trip: 10 October, 1980
Individual(s) Traveling: P.H. Enslow
Itinerary: Hampton, VA
Contact: Aircraft Electronic Systems Branch, Flight Electronics Division, NASA, Langley Research Center
Purpose: Discuss common interests in distributed processing

Date of Trip: October, 1980
Individual(s) Traveling: N. Lynch
Itinerary: Syracuse, New York
Contact: conference attendees
Purpose: Attend conference on Foundations of Computer Science (IEEE)

Date of Trip: October, 1980
Individual(s) Traveling: N. Lynch
Itinerary: Toronto, Canada
Contact: Eshrat Arjomandi, York University
Purpose: Work on projects A.7 & A.15. Give presentation describing projects A.6 & A.8.

Date of Trip: October, 1980
Individual(s) Traveling: N. Lynch
Itinerary: Raleigh, North Carolina
Contact: presentation attendees at North Carolina State University
Purpose: Give invited presentation on Projects A.6 & A.8.

Date of Trip: 27-30 October, 1980
Individual(s) Traveling: P.H. Enslow, J. Skowbo, R. Wice
Itinerary: Atlanta, Georgia
Contact: conference attendees
Purpose: Attend ICCC '80 - the Fifth International Conference on Computer Communications

Date of Trip: 10-14 November, 1980
Individual(s) Traveling: P.H. Enslow
Itinerary: San Diego, California
Contact: meeting attendees
Purpose: Participate in a working meeting of IFIP Working Group 10.7, "Operating System Interfaces".

Date of Trip: 12 November 1980
Individual(s) Traveling: P.H. Enslow
Itinerary: Naval Ocean Systems Center, San Diego, CA
Contact: A.G. Diloreto (Code 1605), R.L. Goodbody (8105), D.W. Gage (8141), D.O. Christy (8121), G.R. Allagier (8242), R.D. Cook (164), and B.F. White (8314)
Purpose: Present briefing on GIT FDPS Research Program. Discuss further interactions.

6. VISITORS

Dates of Visit: 6 October, 1980
Visitor: Billy L. Dove, Head, Aircraft Electronic Systems Branch, Flight Electronics Division, NASA, Langley Research Center, Hampton, Virginia
Contact: P.H. Enslow
Purpose: To be briefed on the GIT FDPS Research Program

Dates of Visit: November, 1980
Visitor: William C. Rounds, University of Michigan, Ann Arbor
Contact: N. Lynch & R. Miller
Purpose: Work on Project A.14 with Prof's Lynch & Miller

7. PUBLICATIONS

Author(s): A.B. Maccabe & R.J. LeBlanc
Title: A Language Model for Fully Distributed Systems
Type: conference paper
Status: presented & published in conference proceedings

Author(s): R.J. LeBlanc & A.B. Maccabe
Title: P+D: Language Features for Distributed Programming
Type: conference paper
Status: submitted

Author(s): N. Lynch
Title: Fast Allocation of Nearby Resources in a Distributed System
Type: journal paper
Status: submitted

Author(s): E. Arjomandi, M. Fischer, & N. Lynch
Title: A Difference in Efficiency Between Synchronous and Asynchronous Systems
Type: conference paper
Status: submitted

Author(s): N. Lynch & M. Fischer
Title: On Describing the Behavior & Implementation of Distributed Systems
Type: journal paper
Status: final revisions made

Author(s): M. Fischer, N. Griffeth, L. Guibas, & N. Lynch
Title: Optimal Placement of Identical Resources in a Distributed Network
Type: conference paper
Status: submitted

Author(s): J. Burns & N. Lynch
Title: Mutual Exclusion Using Indivisible Reads and Writes
Type: conference paper
Status: presented

Author(s): P.H. Enslow
Title: Quarterly Progress Report - Number 4
Type: Quarterly Progress Report
Status: printed & distributed
Publ. Date: November, 1980

THE GEORGIA INSTITUTE OF TECHNOLOGY

RESEARCH PROGRAM IN

FULLY DISTRIBUTED PROCESSING SYSTEMS

Quarterly Progress Report Number 6
1 December, 1980 - 28 February, 1981

March, 1981

Supported by

Office of Naval Research (ONR)
Contract: N00014-79-C-0873
GIT Project: G36-643

U.S. Air Force Rome Air Development Center (RADC)
Contract: F30602-78-C-0120
GIT Project: G36-654

National Science Foundation (NSF)
Contract: MCS77-28305
Subcontract from Univ. of Wisc.: 144-L729
GIT Project: G36-630

U.S. Army Research Office (ARO)
Contract: DAAG29-79-C-0155
GIT Project: G36-638

U.S. Army Institute for Research in
Management Information and Computer Science (AIRMICS)
Contract: DAAK70-79-D-0087
GIT Project: G36-647

School of Information and Computer Science
Georgia Institute of Technology
Atlanta, Georgia 30332

1. INTRODUCTION

This is the Sixth Quarterly Progress Report prepared on the Georgia Tech Research Program in Fully Distributed Processing Systems (FDPS).

a. Program Description.

The Georgia Tech Research Program in Fully Distributed Processing Systems is a comprehensive investigation of data processing systems in which both the physical and logical components are extremely loosely coupled while operating with a high degree of control autonomy at the component level. The definition of the specific class of multiple computer systems being investigated, and the operational characteristics and features of those systems is motivated by the desire to advance the state-of-the-art for that class of systems that will deliver a high proportion of the benefits currently being claimed for distributed processing systems. The scope of individual topics being investigated under this program ranges from formal modeling and theoretical studies to empirical examinations of prototype systems and simulation models. Also included within the scope of the program are areas such as the utilization of FDPS's and their interaction with management operations and structure.

b. Program Support.

The principle support for the program is a Selected Research Opportunity contract from the Office of Naval Research; however, there are a number of other sources of funding which also support the program. A list of the currently active contracts and grants is given below.

Title: "Research on Fully Distributed Data Processing Systems"
Funding Agency: Office of Naval Research (ONR)
Contract Number: N00014-79-C-0873
GIT Project No.: G36-643
Principle Investigator: Philip H. Enslow, Jr.

Title: "Evaluation of Distributed Control Models"
Funding Agency: U.S. Air Force Rome Air Development Center (RADC)
Contract Number: F30602-78-C-0120
GIT Project No.: G36-654
Principle Investigator: Philip H. Enslow, Jr.

Title: "Foundations of Deterministic Scheduling of Processes for Parallel Execution"
Funding Agency: National Science Foundation (NSF)
Contract Number: MCS77-28305
(Univ. of Wisc. subcontract number: 144-L729)
GIT Project Number: G36-630
Principle Investigator: Richard A. DeMillo

Title: "Theory of Systems of Asynchronous Parallel Processors"
Funding Agency: U.S. Army Research Office (ARO)
Contract Number: DAAG29-79-C-0155
GIT Project Number: G36-638
Principle Investigator: Nancy Lynch

Title: "Support of MILPERCEN Data Storage Concept"
Funding Agency: U.S. Army Institute for Research in Management Information and Computer Science (AIRMICS)
Contract Number: DAAK70-79-D-0087
GIT Project Number: G36-647
Principle Investigator: A.P. Jensen

2. ORGANIZATION AND STAFFING

Faculty

Davida, George--Professor
DeMillo, Richard A.--Associate Professor
Enslow, Philip H. Jr.--Professor
Griffeth, Nancy--Assistant Professor
Jensen, Alton P.--Professor
LeBlanc, Richard--Assistant Professor
Livesey, Jon--Assistant Professor
Lynch, Nancy--Associate Professor
Miller, Raymond--Professor

Staff

McDonell, Sharon--Senior Secretary
Myers, Jeanette--Research Scientist
Pinion, Nancy--Part-time Secretary
Mongiovi, Roy--Research Technologist I

Students

There are approximately 30 students working on various projects in the FDPS Research Program. Of these, 12 are in the Ph.D. program and 5 are preparing their MS Thesis on topics in FDPS.

3. CURRENT RESEARCH PROJECTS

The specific research projects have been organized into the major areas identified in the basic program proposal.

A. Theoretical and Formal Studies

A.1 Models of Asynchronous Processors
A.2 Decomposition of Parallel Systems
A.3 Reliable Systems
A.4 Time Performance of Distributed Systems
A.5 Audit Algorithms
A.6 Ticket Systems

A.7 Synchronous Simulation
A.8 Distributed Resource Allocation
A.9 Theory of Distributed Databases
A.10 Arbiter Design
A.11 Shared Memory Bounds for Synchronization Problems
A.12 Mutual Exclusion
A.13 Adaptive Distributed Resource Allocation Algorithms
A.14 Using Complementary Distributed System Models
A.15 Probabilistic Algorithms in Distributed Systems
A.16 Stochastic Synchronization
A.17 Research Allocation in a Failure-Prone Environment

B. Physical Interconnection and Networking

B.1 Heterogeneous Networking
B.2 Local Networking in Fully Distributed Processing Systems

C. Distributed Operating Systems

C.1 Decentralized and Distributed Control
C.2 Resource Allocation and Work Distribution in an FDPS
C.3 Fully Distributed Operating System - Initial Considerations
C.4 Local Operating System
C.5 Communications Support for Distributed Systems
C.7 FDOS - Preliminary Implementation Studies

D. Distributed Data Bases

D.1 Concurrency Control in Distributed Database Systems
D.2 Support of MILPERCEN Data Storage Concept
D.3 Implementation of the Audit Algorithm

E. Fault-Tolerance

F. Special Hardware to Support FDPS

G. Application of Distributed Processing

H. System Design Methodologies

H.1 FDPS Requirements Engineering Techniques
H.2 Coordinating Large Programming Projects

I. System Utilization

I.1 A Language for Distributed Programming
I.2 System Implementation Language Development
I.3 Experiments with a Distributed Compiler

J. Security

J.1 Process Structures
J.2 System Security

K. System Management

L. Evaluation and Comparison

L.1 Simulation of Distributed Algorithms (Griffeth, Lynch)
L.2 Survivability

M. FDPS Testbed

M.1 Establishment of FDPS Testbed Facility
M.2 Remote Load Emulator
M.3 Fully Distributed Operating System Simulation Testbed

4. SUMMARY OF PROGRESS

A.1 Models of Asynchronous Processors (Lynch, Fischer)

No further progress anticipated. Project terminated.

A.2 Decomposition of Parallel Systems (Lynch, Fischer)

No significant progress to report.

A.3 Reliable Systems (Lynch, Fischer, Lamport)

No significant progress to report.

A.4 Time Performance of Distributed Systems (Lynch, Fischer, Lazowska, Schönhage)

No significant progress to report.

A.5 Audit Algorithms (Griffeth, Fischer, Lynch)

The earlier bank audit algorithm has been generalized considerably to an algorithm that returns a global state of a very general distributed system (e.g. a distributed data base), without halting concurrent operations in progress. The new general algorithm can be used for failure detection and recovery in distributed systems , and consistency checking in data base systems. It appears to be quite fast. The first draft of an invited paper, "Global States of a Distributed System", was prepared for presentation at the 1981 IEEE Conference on Distributed Software and Data Bases.

A.6 Ticket Systems (Fischer, Griffeth, Guibas, Lynch)

The paper, "Optimal Placement of Identical Resources in a Distributed Network", was rewritten for the Paris Conference on Distributed Systems.

A.7 Synchronous Simulation (Lynch, Fischer, Arjomandi)

A paper, "A Difference in Efficiency Between Synchronous and Asynchronous Systems", was written and submitted for presentation at the 1981 SIGACT Conference.

A.8 Distributed Resource Allocation (Lynch)

The paper, "Fast Allocation of Nearby Resources in a Distributed System", was revised for invited publication under the title, "Upper Bounds on Static Resource Allocation in a Distributed System, in a special issue of the Journal of Computer and System Sciences based on the 1980 SIGACT Conference.

A.9 Theory of Distributed Databases (Lynch, Griffeth)

The Audit Algorithm (Project A.5) has been generalized to apply to very general distributed data base systems. Discussions have been carried out attempting to generalize the usual notion of "serializability" used for correctness in data bases.

A.10 Arbiter Design (Lynch, Griffeth, Schönhage, Fischer)

No significant progress to report.

A.11 Shared Memory Bounds for Synchronization Problems (Burns, Lynch)

No further progress anticipated. Project terminated.

A.12 Mutual Exclusion (Burns, Lynch)

This Project was completed with the publication of a Ph.D. Thesis by J. Burns.

A.13 Adaptive Distributed Resource Allocation Algorithms (Ghoudjehbaklou, Lynch)

No further progress anticipated. Project terminated.

A.14 Using Complementary Distributed System Models (Lynch, Rounds, Miller)

No significant progress to report.

A.15 Probabilistic Algorithms in Distributed Systems (Lynch, Arjomandi, Fischer)

No significant progress to report.

A.16 Stochastic Synchronization (DeMillo, Miller, Lipton)

The work in Stochastic Synchronization has been brought to publication with an article, "Stochastic Synchronization". This paper reports results of simulations to support analytic results.

A.17 Resource Allocation in a Failure-Prone Environment (Fischer, Lynch, Burns, Borodin)

Revision of a paper, "Resource Allocation with Immunity to Limited Process Failure", is being carried out for journal publication.

B.1 Heterogeneous Networking

Initial task completed. No further work in this area during this quarter. Project terminated.

B.2 Local Networking in FDPSs (Enslow)

Work during this period has been focused on the control and software problems of the local network that has been installed. Emphasis is being placed on software and system reliability as well as the incorporation of important features such as a network name server.

C.1 Decentralized and Distributed Control (Enslow, LeBlanc, Saponas)

The first task under this project, identification and definition of models of distributed control, was completed with the publication of the final report. Work on the second task, evaluation of these models, is proceeding.

C.2 Resource Allocation and Work Distribution in an FDPS (Enslow, Sharp)

A draft of a paper entitled, "An Initial Examination of Resource Management and Work Distribution in a Fully Distributed Processing System", was completed and is expected to be published as a technical report in the near future. A draft version of a proposal for a simulation experiment to measure the performance of work distribution algorithms in fully distributed systems was completed. The proposal will also be published as a technical report.

C.3 FDOS - Initial Considerations (Enslow, Livesey, LeBlanc, Akin, Flinn, Forsyth, Fukuoka, Maccabe, Myers, Pitts, Saponas, Skowbo, Spafford)

Many of the initial considerations for the design and implementation of a Fully Distributed Operating System have appeared under other projects, primarily C.1, C.2, and C.4. The local operating system, Project C.4, has been designed to support alternative implementations of distributed operating system concepts. Project C.3, as well as Project C.7, will be terminated this quarter with the conclusion that our efforts are best directed towards the implementation of a local operating system that will provide a real testing environment for distributed operating system concepts.

C.4 Local Operating System (Livesey, LeBlanc, Spafford, Myers, Flinn, Forsyth, Fox, Fukuoka, Greene, Hopkins, Mongiovi, Pitts)

A preliminary design is complete and we are in the process of expanding this design, top-down, to the coding level. In anticipation of implementing an LOS on the PR1MEs, a special class, ICS 8113-F, "The Organization, Architecture, and Programming of PR1ME Computers", was offered this quarter by a member of the Local Operating System team. Also, in conjunction with this project, ICS 8113-L, a seminar on distributed operating system concepts, was offered.

C.5 Communications Support for Distributed Systems (Enslow, Skowbo, Wice)

At the commencement of the Sixth Quarter, three subtasks for this project had been identified:

1. Development and evaluation of a highly distributed routing algorithm for a message-switching network.
2. Completion of an enhanced Interprocess Communication Simulator to support communication research, particularly the evaluation phase of the third subtask.
3. Development and evaluation of a highly selective acknowledgement protocol for error control in fully distributed systems and other computer networks requiring improved throughput on high-speed, long-delay satellite links.

The first subtask has been abandoned for several reasons:

1. The physically tight coupling of the current hardware configuration in the Computer Lab would not permit a direct evaluation of routing methods, since communication between processors is constrained to follow a single fixed path, and there are no plans to change this configuration in the near future.
2. The development of software for a completely operational communication subsystem or its simulation is well beyond the limitations of this project.
3. Details of the proposed implementation reveal unforeseen operational difficulties which would largely negate the expected benefits of this algorithm, particularly in comparison with currently available alternatives which are more centralized and tightly-coupled, but highly adaptive and refined by years of operational experience.

The second subtask is very near completion, however, its exact status cannot be determined, pending a demonstration, evaluation, and final report in the form of a Masters Thesis by Mr. Wice. (Mr. Wice has left Georgia Tech for full-time employment. He anticipates completion of his thesis during the next quarter.)

The highly selective acknowledgement protocol has been described in some detail with a plan for undertaking its evaluation; however, further progress on this third subtask has been deferred, pending a more thorough study of the communications support required by fully distributed processing systems.

C.7 FDOS - Preliminary Implementation Studies (Myers, Enslow)

This project has been terminated as described above in the progress report on Project C.3.

D.1 Concurrency Control in Distributed Database Systems (Griffeth, Livesey, Lynch)

A model of a distributed database system has been developed for use in simulating distributed concurrency control algorithms. Design of the simulation is in progress.

D.2 Support of MILPERCEN Data Storage Concept (Jensen, Doyle, Gehl, Bingham)

This quarter's work included typing of the transcript of the workshop, "Implications of Data Base Technology for Human Resource Information Management", held in October, 1980. The transcript was sent to the workshop attendees for review and for approval concerning the basic ideas put forth at the workshop. Secondly, initial work was begun in reviewing the literature of data base technology from the perspectives of design, performance, data communications, and evaluation of data bases. Third, the project staff was invited on January 16, 1981 by AIRMICS to an IBM General Systems presentation on data base standards, IMS products and systems, and the data base environment. Fourth, the writing of the final project report was begun and is continuing.

D.3 Implementation of the Audit Algorithm (Griffeth, Livesey, Lynch)

An investigation of block allocation by the Series/1 file management system is in progress, for a test implementation of the audit algorithm.

H.1 FDPS Requirements Engineering Techniques (Underwood, Corley)

No further progress anticipated. Project terminated.

H.2 Coordinating Large Programming Projects (Enslow, Smith)

The initial formulation of the model of the communcation process in the development of large software systems has been completed. The model is now undergoing refinement and amplification.

I.1 A Language for Distributed Programming (LeBlanc, Maccabe, Hardin)

Design work has continued during this quarter and some implementation work has started. A paper was presented at the Louisiana Computer Exposition, providing some useful interaction with other researchers in this area.

I.2 System Implementation Language Development (LeBlanc, Akin)

The code generator has been completely designed. Implementation will start in the near future.

I.3 Experiments with a Distributed Compiler (LeBlanc, Moore)

Moore is currently writing his M.S. thesis describing this work.

J.1 Process Structures (DeMillo, Lipton, Miller)

We have been preparing a book-length treatment of cryptographic protocols for publication in late 1981. As part of this project a large number of protocols suited to distributed systems have been identified and potential lines of compromise explored. We have also identified several new algorithms for examination. Theoretical research has centered on quantitative measures of system security .

J.2 System Security (Livesey, Davida, DeMillo)

Operating systems security is a relatively new area. Harrison, Ruzzo, and Ullman have shown that the safety question for operating systems is undecidable. However, practical design issues require that new approaches to secure operating systems be developed. Davida, DeMillo, and Lipton have introduced a new architecture that implements the "star" property for multilevel secure operating systems. The approach differs from that of other designs which rely on verification techniques to implement a secure kernel.

A paper is being written in which we present a new architecture that achieves security in a timeshared operating system. This is a very important class of operating systems since they are widely used.

L.1 Simulation of Distributed Algorithms (Griffeth, Lynch)

The ticket simulation program has been transferred to the PR1ME and is being tested there. A graphical display of the simulated allocation of tickets has been developed.

L.2 Survivability (DeMillo, Martin)

This Project was completed with the publication of a Ph.D. Thesis by E. Martin on experimental aspects of survivability in distributed systems. A central result of this thesis was a factor analysis of approximately 300,000 data points to identify key parameters which influence system survivability.

M.1 Establishment of FDPS Testbed Facility (Myers, Fox)

A high-level language interface to PR1MENET's X.25 subroutines has been implemented. This interface allows asynchronously running programs to communicate using send and receive primitives.

M.2 Remote Load Emulator (Myers, Enslow, Forsyth)

No significant progress to report.

M.3 FDOS Simulation Testbed (LeBlanc, Hopkins, Myers)

The simulator is near completion. Some tailoring is being done to accommodate Project C.2, which will soon require this simulator.

5. TRAVEL RELATED TO THE FDPS PROGRAM

Date of Trip: 15-17 December, 1980
Individual(s) Traveling: Richard LeBlanc & Nancy Lynch
Itinerary: Fallbrook, California
Purpose: Attend a workshop on fundamental issues in distributed computing

Date of Trip: 3-5 January, 1981
Individual(s) Traveling: Richard DeMillo
Itinerary: San Francisco, California
Purpose: Present invited talk at annual meeting of American Mathematical Society

Date of Trip: 9-10 February, 1981
Individual(s) Traveling: Philip H. Enslow, Jr.
Itinerary: Virginia Polytechnic Institute and State University, Blacksburg, Virginia
Contact: Roger Ehrich
Purpose: Present talk on FDPS Program to faculty and students

Date of Trip: 26-27 February, 1981
Individual(s) Traveling: Richard LeBlanc
Itinerary: Lafayette, Louisiana
Purpose: Present a paper at the Louisiana Computer Exposition

6. VISITORS

Dates of Visit: 10-13 January, 1981
Visitor: Mike Fischer
Contact: Nancy Lynch & Nancy Griffeth
Purpose: Work on Project A.5 and participate in the Ph.D. Dissertation Defense of J. Burns.

7. PUBLICATIONS

Author(s): N. Lynch
Title: Fast Allocation of Nearby Resources in a Distributed System
Type: Conference paper
Status: Published
Publ. Date: May, 1980

Author(s): P.H. Enslow, Jr.
Title: Quarterly Progress Report - Number 5
Type: Quarterly Progress Report
Status: Published
Publ. Date: December, 1980

Author(s): E.W. Martin
Title: Survivability in Gracefully Degrading Computer Systems.
Type: Ph.D. Thesis
Status: Published
Publ. Date: January, 1981

Author(s): R.A. DeMillo
Title: Cryptographic Protocols
Type: Conference paper
Status: Published
Publ. Date: January, 1981

Author(s): J. Burns
Title: Complexity of Communication among Asynchronous Parallel Processes
Type: Ph.D. Thesis
Status: Published
Publ. Date: 12 January, 1981

Author(s): P.H. Enslow, Jr. & T.G. Saponas
Title: Distributed and Decentralized Control in Fully Distributed Processing Systems - A Survey of Applicable Models
Type: Final Technical Report
GIT Number: GIT-ICS-81/02
Status: Published
Publ. Date: February, 1981

Author(s): R.J. LeBlanc
Title: Communication and Control Abstractions in a Programming Language for Fully Distributed Systems
Type: Conference paper
Status: Published in Proceedings of the Third Annual Louisiana Computer Exposition
Publ. Date: February, 1981

Author(s): R.J. LeBlanc & A.B. Maccabe
Title: P+D: Language Features for Distributed Programming
Type: Technical Report
Status: In preparation
Publ. Date: March, 1981

Author(s): R.A. DeMillo, R. Lipton, & R. Miller
Title: Stochastic Synchronization
Type: Conference paper
Status: Submitted for publication
Publ. Date: March, 1981

Author(s): G. Davida, R. DeMillo, & R. Lipton
Title: Achieving Multilevel Security Through Distributed Systems
Type: Conference paper
Status: Submitted for publication
Publ. Date: April, 1981

Author(s): M. Fischer, N. Griffeth, L. Guibas, & N. Lynch
Title: Optimal Placement of Identical Resources in a Distributed Network
Type: Conference paper
Status: Accepted by Paris Conference on Distributed Systems
Publ. Date: April, 1981

Author(s): E. Arjomandi, M. Fischer, & N. Lynch
Title: A Difference in Efficiency Between Synchronous and Asynchronous Systems
Type: Conference paper
Status: Accepted by 1981 SIGACT Conference
Publ. Date: May, 1981

Author(s): M. Fischer, N. Griffeth, & N. Lynch
Title: Global States of a Distributed System
Type: Conference paper
Status: Invited by 1981 IEEE Conference on Distributed Software and Data Bases
Publ. Date: July, 1981

Author(s): N. Lynch
Title: Upper Bounds on Static Resource Allocation in a Distributed System
Type: Journal paper
Status: Invited, revised, and submitted to Journal of Computer and System Sciences

Author(s): M. Fischer, N. Lynch, J. Burns, & A. Borodin
Title: Resource Allocation with Immunity to Limited Process Failure
Type: Conference paper
Status: Revision in preparation

THE GEORGIA INSTITUTE OF TECHNOLOGY

RESEARCH PROGRAM IN

FULLY DISTRIBUTED PROCESSING SYSTEMS

Quarterly Progress Report Number 7
1 March, 1981 - 31 May, 1981

June, 1981

Supported by

Office of Naval Research (ONR)
Contract: N00014-79-C-0873
GIT Project: G36-643

U.S. Air Force Rome Air Development Center (RADC)
Contract: F30602-78-C-0120
GIT Project: G36-654

U.S. Army Research Office (ARO)
Contract: DAAG29-79-C-0155
GIT Project: G36-638

U.S. Army Institute for Research in
Management Information and Computer Science (AIRMICS)
Contract: DAAK70-79-D-0087
GIT Project: G36-647

National Science Foundation (NSF)
Contract: MCS-7924370
GIT Project: G36-652

School of Information and Computer Science
Georgia Institute of Technology
Atlanta, Georgia 30332

1. INTRODUCTION

This is the Seventh Quarterly Progress Report prepared covering the Georgia Tech Research Program in Fully Distributed Processing Systems (FDPS).

a. Program Description.

The Georgia Tech Research Program in Fully Distributed Processing Systems is a comprehensive investigation of data processing systems in which both the physical and logical components are extremely loosely coupled while operating with a high degree of control autonomy at the component level. The definition of the specific class of multiple computer systems being investigated, and the operational characteristics and features of those systems is motivated by the desire to advance the state-of-the-art for that class of systems that will deliver a high proportion of the benefits currently being claimed for distributed processing systems. The scope of individual topics being investigated under this program ranges from formal modeling and theoretical studies to empirical examinations of prototype systems and simulation models. Also included within the scope of the program are areas such as the utilization of FDPS's and their interaction with management operations and structure.

b. Program Support.

The principle support for the program is a Selected Research Opportunity contract from the Office of Naval Research; however, there are a number of other sources of funding which also support the program. A list of the currently active contracts and grants is given below.

Title: "Research on Fully Distributed Data Processing Systems"
Funding Agency: Office of Naval Research (ONR)
Contract Number: N00014-79-C-0873
GIT Project No.: G36-643/336
Principle Investigator: Philip H. Enslow, Jr.

Title: "Evaluation of Distributed Control Models"
Funding Agency: U.S. Air Force Rome Air Development Center (RADC)
Contract Number: F30602-78-C-0120
GIT Project No.: G36-654
Principle Investigator: Philip H. Enslow, Jr.

Title: "Theory of Systems of Asynchronous Parallel Processors"
Funding Agency: U.S. Army Research Office (ARO)
Contract Number: DAAG29-79-C-0155
GIT Project Number: G36-638/332
Principle Investigator: Nancy A. Lynch

Title: "Support of MILPERCEN Data Storage Concept"
Funding Agency: U.S. Army Institute for Research in Management Information and Computer Science (AIRMICS)
Contract Number: DAAK70-79-D-0087
GIT Project Number: G36-647
Principle Investigator: Alton P. Jensen

Title: "Complexity and Computability for Distributed Data Bases"
Funding Agency: National Science Foundation (NSF)
Contract Number: MCS-7924370
GIT Project Number: G36-652/340
Principle Investigator: Nancy A. Lynch

2. ORGANIZATION AND STAFFING

Faculty

Davida, George--Professor
DeMillo, Richard A.--Associate Professor
Enslow, Philip H. Jr.--Professor
Griffeth, Nancy--Assistant Professor
Jensen, Alton P.--Professor
LeBlanc, Richard--Assistant Professor
Livesey, Jon--Assistant Professor
Lynch, Nancy A.--Associate Professor
Miller, Raymond--Professor
Underwood, William -- Assistant Professor

Staff

McDonell, Sharon--Senior Secretary
Myers, Jeanette--Research Scientist
Pinion, Nancy--Part-time Secretary
Mongiovi, Roy--Research Technologist I

Students

There are approximately 30 students working on various projects in the FDPS Research Program. Of these, 12 are in the Ph.D. program, and 5 are preparing their M.S. Thesis on topics in FDPS.

3. CURRENT RESEARCH PROJECTS

The specific research projects have been organized into the major areas identified in the basic program proposal.

A. Theoretical and Formal Studies

A.2 Decomposition of Parallel Systems
A.3 Reliable Systems
A.4 Time Performance of Distributed Systems
A.5 Audit Algorithms
A.6 Ticket Systems
A.7 Synchronous Simulation

A.8 Distributed Resource Allocation
A.9 Theory of Distributed Databases
A.10 Arbiter Design
A.14 Using Complementary Distributed System Models
A.15 Probabilistic Algorithms in Distributed Systems
A.16 Stochastic Synchronization
A.17 Research Allocation in a Failure-Prone Environment

B. Physical Interconnection and Networking

B.2 Local Networking in Fully Distributed Processing Systems

C. Distributed Operating Systems

C.1 Decentralized and Distributed Control
C.2 Resource Allocation and Work Distribution in an FDPS
C.4 Local Operating System
C.5 Communications Support for Distributed Systems

D. Distributed Data Bases

D.1 Concurrency Control in Distributed Database Systems
D.2 Support of MILPERCEN Data Storage Concept
D.3 Implementation of the Audit Algorithm

E. Fault-Tolerance

F. Special Hardware to Support FDPS

G. Application of Distributed Processing

H. System Design Methodologies

H.2 Coordinating Large Programming Projects

I. System Utilization

I.1 A Language for Distributed Programming
I.2 System Implementation Language Development
I.3 Experiments with a Distributed Compiler

J. Security

J.1 Process Structures
J.2 System Security

K. System Management

L. Evaluation and Comparison

L.1 Simulation of Distributed Algorithms (Griffeth, Lynch)

M. FDPS Testbed

M.1 Establishment of FDPS Testbed Facility
M.2 Remote Load Emulator
M.3 Fully Distributed Operating System Simulation Testbed

4. SUMMARY OF PROGRESS

A.2 Decomposition of Parallel Systems (Lynch, Fischer)

A paper, "On Decomposing Algorithms that use a Single Shared Variable", is in the process of being revised for journal publication.

A.3 Reliable Systems (Lynch, Fischer, Lamport)

Work has begun on writing up results proving lower bounds on the number of rounds of communication required to solve the Byzantine Generals problem (i.e. reaching agreement in the presence of faulty processors). The present focus is on determining appropriate models for the statement of this problem, especially for versions of the problem allowing encrypted messages. We would like to interpret the "number of rounds" lower bound as a lower bound on the time required for the solution.

A.4 Time Performance of Distributed Systems (Lynch, Fischer, Lazowska, Schönhage)

No significant progress to report.

A.5 Audit Algorithms (Griffeth, Fischer, Lynch)

The audit algorithm has been generalized still further. The final draft of a paper, "Global States of a Distributed System", has been submitted for presentation at the 1981 IEEE Conference on Distributed Systems and Software.

A.6 Ticket Systems (Fischer, Griffeth, Guibas, Lynch)

The paper, "Optimal Placement of Identical Resources in a Distributed System", was presented at the Second International Conference on Distributed Systems (Paris) on April 12, 1981.

Discussions were carried out for new designs of distributed ticket algorithms, as alternatives to those already studied. Several of these will be simulated using our general tree network simulator program, and compared with our basic algorithm.

Further tests were carried out on our basic ticket distribution algorithm: we tested the effect of allowing returns of tickets, and used a simple overestimate of step time to obtain upper bounds on response time, as expected interarrival time for requests was allowed to increase indefinitely. This upper bound was found to be a monotone increasing and bounded function of the expected interarrival time.

A graphics component for the simulator was completed and used in several demonstrations of the operation of our algorithm.

A.7 Synchronous Simulation (Lynch, Fischer, Arjomandi)

The paper, "A Difference in Efficiency Between Synchronous and Asynchronous Systems", was presented at the 1981 SIGACT Conference.

A.8 Distributed Resource Allocation (Lynch)

The paper, "Upper Bounds on Static Resource Allocation in a Distributed System", was accepted for invited publication in a special issue of the Journal of Computer and System Sciences based on the 1980 SIGACT Conference.

A.9 Theory of Distributed Databases (Lynch, Griffeth)

The paper, "Multilevel Atomicity - A New Correctness Criterion for Distributed Databases", was prepared for journal publication. It contains a definition for "multilevel atomicity", a new generalization of the usual notion of "serializability" used for correctness in distributed databases. This new condition seems to admit more efficient implementation than the usual definition, yet seems sufficiently general to allow expression of the conditions required for most real database applications.

Discussions were carried out to see how this generalization applies to the "Eden objects" used as the logical basis for the design of the local network at the University of Washington.

A.10 Arbiter Design (Lynch, Griffeth, Schönhage, Fischer)

No significant progress to report.

A.14 Using Complementary Distributed System Models (Lynch, Rounds, Miller)

No significant progress to report.

A.15 Probabilistic Algorithms in Distributed Systems (Lynch, Arjomandi, Fischer)

No significant progress to report.

A.16 Stochastic Synchronization (DeMillo, Miller, Lipton)

No significant progress to report.

A.17 Resource Allocation in a Failure-Prone Environment (Fischer, Lynch, Burns, Borodin)

Revision of a paper, "Resource Allocation with Immunity to Limited Process Failure", is being carried out for journal publication.

B.2 Local Networking in FDPSs (Enslow)

Since the last progress report, several new and some improved features have been incorporated into the local network. Many of the reliability problems have diminished due to our receipt of new software from Ungermann-Bass. We are continuing to work with Ungermann-Bass to isolate and solve the remaining performance problems.

Among the new features incorporated into the local network are a network-global name service, flow-control buffer thresholds, extended RS-232C control, and parallel data communications. Improved network throughput is the result of better internal buffer management and more efficient device drivers.

Hardware reliability has been greatly enhanced due to our receipt and incorporation of some minor hardware modifications.

The draft of a report detailing our experiences with NET/ONE and our future plans for local networking has been prepared.

C.1 Decentralized and Distributed Control (Enslow, LeBlanc, Saponas)

Evaluation of models of distributed control will be done through simulation. Work this quarter has been devoted to more thoroughly defining these models for the analysis stage and to the completion of the simulator.

C.2 Resource Allocation and Work Distribution in an FDPS (Enslow, Sharp)

A proposal to conduct a simulation experiment to evaluate the performance of several work distribution algorithms was finalized. This paper is entitled "Work Distribution in a Fully Distributed Processing System", and will be published shortly as a technical report.

C.4 Local Operating System (Livesey, LeBlanc, Spafford, Myers, Fukuoka, Pitts)

A program has been set up to explore methods of adapting the PR1MOS operating system to act as a LOS prototype. So far, areas covered have been the system kernel, device drivers, and high level system implementation languages. A secondary study has been carried out on file system structures for the LOS. An internal document, "The Local Operating System", has been completed and extensively discussed.

C.5 Communications Support for Distributed Systems (Enslow, Skowbo)

Progress this quarter has focused on the role of satellite communications in a FDPS. Multiple-access protocols have some interesting properties that seem ideally suited to the operation of such systems. These properites are being investigated, and methods for reducing or eliminating related problems, such as channel contention, are being studied. A long-range research plan has been established as a guide for further progress.

D.1 Concurrency Control in Distributed Database Systems (Griffeth, Livesey, Lynch)

The simulation framework for concurrency control algorithms in distributed database systems has been implemented. A technical report including a description of the simulation framework and instructions for its use is in preparation.

D.2 Support of MILPERCEN Data Storage Concept (Jensen, Doyle, Gehl, Bingham)

The final report is in preparation.

D.3 Implementation of the Audit Algorithm (Griffeth, Livesey, Lynch)

Three potential applications of the audit algorithm have been investigated for an initial implementation: the SERIES/1 file system, the ticket system (see Progress Summary for Project L.1), and the distributed database simulation of Project D.1.

The file system implementation has been postponed because the theory must be generalized still further to allow the number of nodes in the system to grow. The distributed database simulation will be the first implemented application of the audit algorithm.

H.2 Coordinating Large Programming Projects (Enslow, Underwood, Smith)

Research during this period focused on the development of a design model describing the process of large software development and an investigation of the relationship between problem solving and the software requirements and design specification process. Rittel's analysis of "wicked" problems in architectural design was applied to problems of the design of complex software systems.

I.1 A Language for Distributed Programming (LeBlanc, Maccabe)

Design and implementation work continued this quarter. A technical report was published and a presentation on our work was given during a visit to the Siemens Research Laboratories in Munich.

I.2 System Implementation Language Development (LeBlanc, Akin)

No progress during this quarter. Work by Akin will resume during the next quarter.

I.3 Experiments with a Distributed Compiler (LeBlanc, Moore)

A M.S. Thesis by Moore has been completed. Work on refining and extending these experiments may begin during the next quarter.

J.1 Process Structures (DeMillo, Lipton, Miller)

We have isolated a class of cryptographic problems and are working on a model of cryptographic protocols. The intent of this model is to allow precise formulation of problems of the following type:

> There is no protocol built on a cryptosystem of type x that satisfies the property D.

The protocols that are allowed can be both deterministic and probabilistic. To date, a number of protocol problems have been translated into the model.

J.2 System Security (Livesey, Davida, DeMillo)

A paper by Livesey and Davida, "An Architecture to Support Secure Operating Systems", was presented at the Second Symposium on Security and Privacy.

L.1 Simulation of Distributed Algorithms (Griffeth, Lynch)

The graphics for the ticket system simulation have been completed. Also, the simulation experiments have proved to be consistent with the "monotonicity" hypothesis for the ticket system algorithm. The hypothesis states that the expected response time increases monotonically with the interarrival time.

Work is in progress on the development of new algorithms for the ticket system. It is expected that any reasonable algorithm will obey the monotonicity property. The cost of two other properties, fairness and the absence of starvation, will also be studied.

M.1 Establishment of FDPS Testbed Facility (Myers, Mongiovi, Pitts, Fox)

A preliminary design for distributed software tools (DSWT) is complete. DSWT will consist of one or more software tools subsystems (SWT) which communicate to locate and utilize resources and make decisions. SWT is a subsystem developed at Georgia Tech which operates under the PR1MOS operating system on PR1ME P350 and larger computers. The subsystem is a complete interface between the user and the PR1ME computer system. It offers a powerful command language similar to the one offered in UNIX, a programming environment, and a hierarchical file system. Many of the concepts behind software tools stem from those found in MULTICS and UNIX and especially from the book Software Tools by Brian W. Kernighan and P.J. Plauger. The general objective of DSWT is to expand upon the capabilities provided by SWT (implying a minimum of implementation effort), to explore some fully distributed processing concepts. DSWT will provide the following:

1. A distributed file system and an associated naming server.

2. Work distribution. Deciding where a process is to be run can depend on the location and size of the data file the process accesses, the current load on each of the individual systems of the network, the resources required by the process, etc.

3. Network utilities like 'mail', 'news', 'to', 'status', and file transfer that operate on a network-wide basis. Network utilities are actually 'applications' and DSWT will provide the framework for applications programs to be written as a set of concurrently executing processes which communicate using the network communication routines implemented during the previous quarter. This will include the ability to 'invoke' remote proceses, establish connections for future communication, and provide error handling, network port servers, remote file access, etc.

M.2 Remote Load Emulator (Myers, Enslow, Forsyth)

The implementation of the Remote Load Emulator is complete. Some preliminary tests have been made, and the emulator has successfully run 20 simultaneous terminal sessions. The M.S. Thesis describing the emulator is being written.

M.3 FDOS Simulation Testbed (LeBlanc, Saponas, Myers)

Work on the simulator continues in support of projects C.1 and C.2.

5. TRAVEL RELATED TO THE FDPS PROGRAM

Date of Trip: 6-11 April, 1981
Individual(s) Traveling: Richard LeBlanc
Itinerary: Paris, France
Purpose: Attend Second International Conference on Distributed Computing Systems.

Date of Trip: 6-11 April, 1981
Individual(s) Traveling: Nancy Griffeth
Itinerary: Paris, France
Purpose: Present paper, "Optimal Placement of Identical Resources in a Distributed System", at the Second International Conference on Distributed Systems.

Date of Trip: 9-13 April, 1981
Individual(s) Traveling: Nancy Lynch
Itinerary: Milwaukee, Wisconsin
Purpose: Attend 1981 SIGACT Conference (paper presented by coauthor, Arjomandi).

Date of Trip: 2-15 April, 1981
Individual(s) Traveling: Richard LeBlanc
Itinerary: Munich, West Germany
Contact: Anton Sauer
Purpose: Visit Siemens Research Laboratories.

Date of Trip: 20 April, 1981
Individual(s) Traveling: Jon Livesey
Itinerary: Oakland, California
Purpose: Present paper, "An Architecture to Support Secure Operating Systems", at the Second Symposium on Security and Privacy

Date of Trip: 30 April - 3 May, 1981
Individual(s) Traveling: Philip Enslow
Itinerary: Las Vegas, Nevada
Purpose: Present talk on the status of distributed processing at Interface '81.

Date of Trip: 18-20 May, 1981
Individual(s) Traveling: Philip Enslow
Itinerary: San Diego, California
Purpose: Present talk on Distributed Computing Systems at Naval Ocean Systems Center.

Date of Trip: 21-22 May, 1981
Individual(s) Traveling: Philip Enslow
Itinerary: Wrightville Beach, North Carolina
Purpose: Participate in Army Workshop on Research Directions for Multi-Micro Computers.

Date of Trip: 27 May, 1981
Individual(s) Traveling: Philip Enslow and Jon Livesey
Itinerary: Hampton, Virginia
Purpose: Explore applications of FDPS in avionics at NASA, Langley AFB.

6. VISITORS

Dates of Visit: 15-17 April, 1981
Visitor: Andrew C. Yao, Stanford University
Contact: Richard DeMillo
Purpose: Research collaboration and presentation entitled, "The Security Problem for Public-Key Protocols".

Dates of Visit: 27 April, 1981
Visitor: Bob Grafton, Dave Mizell, ONR.
Contact: Albert Badre, Richard DeMillo, Philip Enslow, Nancy Griffeth, Richard LeBlanc, Jon Livesey, Nancy Lynch, Raymond Miller
Purpose: Discuss FDPS Research Program.

7. PUBLICATIONS

Author(s): M. Fischer, N. Griffeth, L. Guibas, & N. Lynch
Title: Optimal Placement of Identical Resources in a Distributed System
Type: Conference Paper
Status: Presented and published in Proceedings of the Second International Conference on Distributed Systems (Paris, France).
Publ. Date: April, 1981

Author(s): J. Livesey & G.I. Davida
Title: An Architecture to Support Secure Operating Systems
Type: Conference Paper
Status: Presented and published in Proceedings of the Second Symposium on Security and Privacy (Oakland, California).
Publ. Date: April, 1981

Author(s): E. Arjomandi, M. Fischer, & N. Lynch
Title: A Difference in Efficiency Between Synchronous and Asynchronous Systems
Type: Conference Paper
Status: Presented and published in Proceedings of the 1981 SIGACT Conference.
Publ. Date: May, 1981

Author(s): R.J. LeBlanc and A.B. Maccabe
Title: PRONET: Language Features for Distributed Programming
Type: Technical Report
Status: Published
GIT Number: GIT-ICS-81/03
Publ. Date: May, 1981

Author(s): Gregory L. Moore
Title: A Distributed Compiler
Type: M.S. Thesis
Status: Presented
Publ. Date: May, 1981

Author(s): N. Lynch
Title: Upper Bounds on Static Resource Allocation in a Distributed System
Type: Journal Paper
Status: Accepted by the Journal of Computer and System Sciences.
Publ. Date: To be determined

THE GEORGIA INSTITUTE OF TECHNOLOGY

RESEARCH PROGRAM IN

FULLY DISTRIBUTED PROCESSING SYSTEMS

Quarterly Progress Report Number 8
1 June, 1981 - 31 August, 1981

October, 1981

Supported by

Office of Naval Research (ONR)
Contract: N00014-79-C-0873
GIT Project: G36-643

U.S. Air Force Rome Air Development Center (RADC)
Contract: F30602-78-C-0120
GIT Project: G36-654

U.S. Air Force Rome Air Development Center (RADC)
Contract: F30602-81-C-0249
GIT Project: G36-659

U.S. Army Research Office (ARO)
Contract: DAAG29-79-C-0155
GIT Project: G36-638

U.S. Army Institute for Research in
Management Information and Computer Science (AIRMICS)
Contract: DAAK70-79-D-0087
GIT Project: G36-647

National Science Foundation (NSF)
Contract: MCS-7924370
GIT Project: G36-652

School of Information and Computer Science
Georgia Institute of Technology
Atlanta, Georgia 30332

1. INTRODUCTION

This is the Eighth Quarterly Progress Report prepared covering the Georgia Tech Research Program in Fully Distributed Processing Systems (FDPS).

a. Program Description.

The Georgia Tech Research Program in Fully Distributed Processing Systems is a comprehensive investigation of data processing systems in which both the physical and logical components are extremely loosely coupled while operating with a high degree of control autonomy at the component level. The definition of the specific class of multiple computer systems being investigated, and the operational characteristics and features of those systems is motivated by the desire to advance the state-of-the-art for that class of systems that will deliver a high proportion of the benefits currently being claimed for distributed processing systems. The scope of individual topics being investigated under this program ranges from formal modeling and theoretical studies to empirical examinations of prototype systems and simulation models. Also included within the scope of the program are areas such as the utilization of FDPS's and their interaction with management operations and structure.

b. Program Support.

The principle support for the program is a Selected Research Opportunity contract from the Office of Naval Research; however, there are a number of other sources of funding which also support the program. A list of the currently active contracts and grants is given below.

Title: "Research on Fully Distributed Data Processing Systems"
Funding Agency: Office of Naval Research (ONR)
Contract Number: N00014-79-C-0873
GIT Project No.: G36-643/336
Principle Investigator: Philip H. Enslow, Jr.

Title: "Evaluation of Distributed Control Models"
Funding Agency: U.S. Air Force Rome Air Development Center (RADC)
Contract Number: F30602-78-C-0120
GIT Project No.: G36-654
Principle Investigator: Philip H. Enslow, Jr.

Title: "System Support Capabilities for Fully-Distributed / Loosely-Coupled Processing Systems"
Funding Agency: U.S. Air Force Rome Air Development Center (RADC)
Contract Number: F30602-81-C-0249
GIT Project No.: G36-659
Principle Investigator: Philip H. Enslow, Jr.

Title: "Theory of Systems of Asynchronous Parallel Processors"
Funding Agency: U.S. Army Research Office (ARO)
Contract Number: DAAG29-79-C-0155
GIT Project Number: G36-638/332
Principle Investigator: Nancy A. Lynch

Title: "Support of MILPERCEN Data Storage Concept"
Funding Agency: U.S. Army Institute for Research in Management Information and Computer Science (AIRMICS)
Contract Number: DAAK70-79-D-0087
GIT Project Number: G36-647
Principle Investigator: Alton P. Jensen

Title: "Complexity and Computability for Distributed Data Bases"
Funding Agency: National Science Foundation (NSF)
Contract Number: MCS-7924370
GIT Project Number: G36-652/340
Principle Investigator: Nancy A. Lynch

2. ORGANIZATION AND STAFFING

Faculty

DeMillo, Richard A. - Professor
Enslow, Philip H. Jr. - Professor
Griffeth, Nancy - Assistant Professor
Jensen, Alton P. - Professor
LeBlanc, Richard - Assistant Professor
Livesey, Jon - Assistant Professor
Lynch, Nancy A. - Associate Professor
Miller, Raymond - Professor
Underwood, William - Assistant Professor

Staff

McDonell, Sharon - Senior Secretary
Myers, Jeanette - Research Scientist
Pinion, Nancy - Part-time Secretary
Mongiovi, Roy - Research Technologist I

Students

There are approximately 30 students working on various projects in the FDPS Research Program. Of these, 12 are in the Ph.D. program, and 5 are preparing their M.S. Thesis on topics in FDPS.

3. CURRENT RESEARCH PROJECTS

The specific research projects have been organized into the major areas identified in the basic program proposal.

A. Theoretical and Formal Studies

A.2 Decomposition of Parallel Systems
A.3 Reliable Systems
A.4 Time Performance of Distributed Systems
A.5 Audit Algorithms
A.6 Ticket Systems
A.7 Synchronous Simulation
A.8 Distributed Resource Allocation
A.9 Theory of Distributed Databases
A.10 Arbiter Design
A.14 Using Complementary Distributed System Models
A.15 Probabilistic Algorithms in Distributed Systems
A.16 Stochastic Synchronization
A.17 Research Allocation in a Failure-Prone Environment
A.18 Multilevel Atomicity

B. Physical Interconnection and Networking

B.2 Local Networking in Fully Distributed Processing Systems

C. Distributed Operating Systems

C.1 Decentralized and Distributed Control
C.2 Resource Allocation and Work Distribution in an FDPS
C.4 Local Operating System
C.5 Communications Support for Distributed Systems
C.8 Distributed Software Tools
C.9 Command Languages in an FDPS

D. Distributed Data Bases

D.1 Concurrency Control in Distributed Database Systems
D.2 Support of MILPERCEN Data Storage Concept
D.3 Implementation of the Audit Algorithm

E. Fault-Tolerance

F. Special Hardware to Support FDPS

G. Application of Distributed Processing

H. System Design Methodologies

H.2 Coordinating Large Programming Projects

I. System Utilization

I.1 A Language for Distributed Programming
I.2 System Implementation Language Development
I.3 Experiments with a Distributed Compiler

J. Security

J.1 Process Structures
J.2 System Security

K. System Management

L. Evaluation and Comparison

L.1 Simulation of Distributed Algorithms (Griffeth, Lynch)

M. FDPS Testbed

M.1 Establishment of FDPS Testbed Facility
M.2 Remote Load Emulator
M.3 Fully Distributed Operating System Simulation Testbed

4. SUMMARY OF PROGRESS

A.2 Decomposition of Parallel Systems (Lynch, Fischer)

No significant progress to report.

A.3 Reliable Systems (Lynch, Fischer, Lamport, Merritt)

A paper, "A Lower Bound on Time to Achieve Interactive Consistency", was written, describing lower bound results for number of rounds to solve the Byzantine Generals problem. Mike Merritt generalized the result to an environment allowing authenticated communication.

A.4 Time Performance of Distributed Systems (Lynch, Fischer, Lazowska, Schönhage)

No significant progress to report.

A.5 Audit Algorithms (Griffeth, Fischer, Lynch)

Conditions under which the audit can be significantly optimized were identified. These are: (1) transactions visit the same sites regardless of the interleaving of transactions, and (2) transactions access the same data-items regardless of the interleaving of transactions. Complexity analysis for the normal case and these two special cases was done.

A paper, "Global States of a Distributed System", was presented, describing the general checkpoint algorithm. The paper was invited for submission to the special issue of TOSE based on the conference, and has been submitted. It is also being recast in database terms for submission to TODS, including some new results on complexity analysis.

A.6 Ticket Systems (Fischer, Griffeth, Guibas, Lynch)

A new algorithm was devised for dynamic tree-balancing when ticket returns are allowed. The tickets are moved from one location to another in the tree using a heuristic which is similar to the heuristic for buyers. In the case of tickets, a ticket is sent out in some direction if that direction is "deficient" in tickets according to a local estimate.

The ticket system simulation was modified to use a regenerative approach to the choice of epochs. This eliminates dependencies of results on behavior preceeding or following the epoch. It also simplifies the statistical inference methods required to assess the accuracy.

New results have been obtained generalizing the previous analysis of sequential resource allocation and general (interfering) resource allocation. Results are currently being written up in a paper, "Expected Time Analysis of a Distributed Resource-Allocation Algorithm".

A.7 Synchronous Simulation (Lynch, Fischer, Arjomandi)

A paper, "A Difference in Efficiency between Synchronous and Asynchronous Systems", has been rewritten from the conference version for journal submission.

A.8 Distributed Resource Allocation (Lynch)

This project has been completed.

A.9 Theory of Distributed Databases (Lynch, Griffeth)

Discussions were begun attempting to integrate ideas about multilevel atomicity of transactions, multilevel data structuring, and multilevel consistency constraints in databases. Some ideas for concurrency control designs were also discussed.

A.10 Arbiter Design (Lynch, Griffeth, Schönhage, Fischer)

No significant progress to report.

A.14 Using Complementary Distributed System Models (Lynch, Rounds, R. Miller)

No significant progress to report.

A.15 Probabilistic Algorithms in Distributed Systems (Lynch, Arjomandi, Fischer)

No further progress anticipated. Project completed.

A.16 Stochastic Synchronization (DeMillo, R. Miller, Lipton)

No significant progress to report.

A.17 Resource Allocation in a Failure-Prone Environment (Fischer, Lynch, Burns, Borodin)

No significant progress to report.

A.18 Multilevel Atomicity (Lynch)

The paper, "Multilevel Atomicity: A new Correctness Criterion for Distributed Databases", was written and submitted for journal publication. It contains a preliminary proposal for a systematic way in which one might weaken the usual "serializability" definition for transactions, in order to obtain increased concurrency.

B.2 Local Networking in FDPSs (Enslow)

A report on initial work in this area has been prepared.

C.1 Decentralized and Distributed Control (Enslow, LeBlanc, Saponas)

The second phase of this project was completed on 30 June 1981. A final report entitled, "Performance of Distributed and Decentralized Control Models for Fully Distributed Processing Systems - Initial Simulation Studies", has been completed and submitted to the sponsors for review. Final publication of this report is pending the reception of sponsor's approval. Meanwhile, further simulation experiments which were suggested by the previous work are being conducted.

C.2 Resource Allocation and Work Distribution in an FDPS (Enslow, Sharp)

Evaluation of models of work distribution and resource allocation will be done by means of simulation. Work this quarter consisted of coding three

RAWD algorithms in PASCAL for inclusion in the simulator used in Project C.1, thoroughly testing each of the three algorithms, and preparing data to be used as input to the simulation experiment.

C.4 Local Operating System (Livesey, LeBlanc, Spafford, Myers, Fukuoka, Pitts)

Further work has taken place in investigations into appropriate Local Operating Systems structures. A report has been written on the meta-system (PRIMOS) approach.

The members of this project have been working on Distributed Software Tools, Project C.8. Previous work done for this project, especially the exploration of methods of adapting the PRIMOS operating system to act as a LOS prototype, is providing design and implementation support for the DSWT project.

C.5 Communications Support for Distributed Systems (Enslow, Skowbo)

The working draft of a project proposal has been prepared and is being expanded as a result of continuing efforts to identify and describe communications requirements for fully distributed processing systems. The fundamental importance of broadcast services to support distributed application processes has prompted a comprehensive investigation of transmission hardware, network topologies, and the access and control procedures to support these services. For long-haul networks, satellites continue to dominate the picture. Several local-area network technologies are being studied and compared in light of their unique advantages and limitations for locally distributed processing.

C.8 Distributed Software Tools (Myers, Livesey, Hopkins, Lee, McGraw, Fox)

Work continues on the Distributed Software Tools Project initiated last quarter. The first phase of design and implementation of DSWT involves creating a mechanism for remote process execution. This capability will serve as a tool that can be used to write many network applications as small command language programs. For example, a distributed mail facility could implement local mail facilities on each node of the network and direct that mail to the user's terminal with the command, "mail; mail@gt.b; mail@gt.c", assuming that the user is connected to gt.a and that gt.b and gt.c are the only other nodes in the network where mail can be stored for the user. As of now, DSWT consists of five components which are in various stages of completion: the DSWT command interpreter, server processes called hosts which are responsible for initiating remote process execution, remote I/O processes which serve as interfaces between the local user and a remote process, a network file system, and a message passing facility.

C.9 Command Languages in an FDPS (Badre, Myers, Greene)

This project, initiated this quarter, is divided into two subprojects. The first has as its objective the design of a "friendly" command language suitable for a distributed environment. Work this quarter has been

devoted to the collection and review of the available literature on command languages.

The second subproject, scheduled to begin next quarter, will attempt to determine if multiple command languages can and should be made available in an FDPS. It will also explore various ways in which multiple command languages can be integrated in a network to provide some (perhaps all) of the claimed benefits of an FDPS.

D.1 Concurrency Control in Distributed Database Systems (Griffeth, Livesey, Lynch)

Experiments were designed for studying concurrency control algorithms on the simulation tool developed for Project L.1. For this purpose, four parameters were identified: (1) method of guaranteeing consistency, (2) method of conflict resolution, (3) locus of control, and (4) management of deadlocks. Ten significant experiments were identified.

D.2 Support of MILPERCEN Data Storage Concept (Jensen, Doyle, Gehl, Bingham)

The final report has been written and is being prepared for publication.

D.3 Implementation of the Audit Algorithm (Griffeth, Livesey, Lynch)

No significant progress to report.

H.2 Coordinating Large Programming Projects (Enslow, Underwood, Smith)

Major research activities this quarter included investigation of large software development as problem-solving and as design activity. Papers were written on both topics. A major paper detailing proposed future research was completed by the end of the quarter.

I.1 A Language for Distributed Programming (LeBlanc, Maccabe)

Design work has been completed. An implementation using a number of compiler development tools, including the code generator developed under Project I.2, is currently in progress. A paper was prepared for submission to the Symposium on Principles of Programming Languages.

I.2 System Implementation Language Development (LeBlanc, Akin)

Allen Akin has completed his M.S. thesis work on the development of a reusable code generator for our PRIME Computers. This tool will enable us to build compilers which generate high-quality code without developing a customized code generator for each one. The code generator is currently being tested by about 20 students in a compiler class and extensions will be planned based on their experiences.

I.3 Experiments with a Distributed Compiler (LeBlanc, J. Miller)

John Miller is currently working on refinements of the experiments previously conducted as part of this project.

J.1 Process Structures (DeMillo, Lipton, R. Miller, Merritt, Thomas)

Michael Merritt and Barbara Smith Thomas are currently studying the application of cryptographic techniques to supply utilities in a distributed system. Recent results have centered around providing secure communications.

J.2 System Security (Livesey, Davida, DeMillo)

Further studies are being carried out into an architecture to support secure multiprogramming.

L.1 Simulation of Distributed Algorithms (Griffeth, Lynch)

The design and specification of a simulation tool for distributed database systems was completed. The tool was designed to be usable for general distributed algorithms also. Details can be found in the technical report.

M.1 Establishment of FDPS Testbed Facility (Myers, Mongiovi, Pitts, Fox)

The Distributed Software Tools (DSWT) Project initiated under this project number last quarter is now Project C.8. Although DSWT will enhance the testbed facility by providing users with a distributed subsystem and an extended capability for running concurrent programs, it was decided that it belonged in the Distributed Operating Systems effort since it is essentially a Network Operating System implemented on top of several local operating systems.

M.2 Remote Load Emulator (Myers, Enslow, Forsyth)

This project has been completed this quarter. The emulator and a user's guide are now available. Near term plans for the emulator involve performance testing for DSWT, Project C.8, and the local network, Project B.2.

M.3 FDOS Simulation Testbed (LeBlanc, Saponas, Myers)

The simulator is complete and has been used to validate the control models developed in Project C.1.

5. TRAVEL RELATED TO THE FDPS PROGRAM

Date of Trip: July, 1981 - August, 1982
Individual(s) Traveling: N. Lynch
Itinerary: Cambridge, Massachusetts
Purpose: On leave at MIT.

Date of Trip: 20-23 July, 1981
Individual(s) Traveling: N. Griffeth, N. Lynch
Itinerary: Pittsburg, Pennsylvania
Purpose: Attend IEEE Conference on Reliability in Distributed Software and Databases, and present paper, "Global States of a Distributed System".

Date of Trip: 3-7 August, 1981
Individual(s) Traveling: R. LeBlanc
Itinerary: Santa Cruz, California
Purpose: Attend a course in Functional Programming at the Institute in Computer Science at the University of California at Santa Cruz.

Date of Trip: 26-28 August, 1981
Individual(s) Traveling: R. DeMillo, M. Merritt
Itinerary: Santa Barbara, California
Purpose: Attend CRYPTO '81 Symposium.

Date of Trip: 30 August - 4 September, 1981
Individual(s) Traveling: N. Griffeth
Itinerary: Boston, Massachusetts (MIT)
Contact: N. Lynch
Purpose: Develop distributed algorithms for ticket systems and discuss further work on performance studies of ticket systems.

6. VISITORS

(no visitors to report for this quarter)

7. PUBLICATIONS

Author(s): A. Akin
Title: V-mode Code Generator User's Guide
Type: internal document
Status: printed
Publ. Date: June, 1981

Author(s): M. Fischer, N. Griffeth, and N. Lynch
Title: Global States of a Distributed System
Type: conference paper
Status: Presented at 1981 IEEE Conference on Distributed Software and Databases. Invited and submitted for publication in special issue of TOSE based on this conference. Database version in preparation for submission to TODS.
Publ. Date: July 21, 1981

Author(s): N. Griffeth
Title: A Simulation Tool for Distributed Database Systems
Type: technical report
Status: published
GIT Number: GIT-ICS-81/15
Publ. Date: August, 1981

Author(s): A. Akin
Title: A Reusable Code Generator for PR1ME 50-series Computers
Type: M.S. Thesis
Status: being printed
GIT Number: GIT-ICS-81/16
Publ. Date: August, 1981

Author(s): D. Forsyth
Title: A Remote Terminal Emulator for PR1ME Computers
Type: technical report
Status: being printed
GIT Number: GIT-ICS-81/12
Publ. Date: August, 1981

Author(s): D. Forsyth
Title: User's Guide for the PR1ME Remote Terminal Emulator
Type: internal document
Status: printed
Publ. Date: August, 1981

Author(s): A.B. Maccabe and R.J. LeBlanc
Title: Communication Features for Distributed Computing Environments
Type: conference paper
Status: submitted to the Symposium on Principles of Programming Languages

Author(s): N. Lynch
Title: Multilevel Atomicity: A New Correctness Criterion for Distributed Databases.
Type: journal paper
Status: submitted for publication

THE GEORGIA INSTITUTE OF TECHNOLOGY

RESEARCH PROGRAM IN

FULLY DISTRIBUTED PROCESSING SYSTEMS

Quarterly Progress Report Number 9
1 September, 1981 - 30 November, 1981

January, 1982

Supported by

Office of Naval Research (ONR)
Contract: N00014-79-C-0873
GIT Project: G36-643

U.S. Air Force Rome Air Development Center (RADC)
Contract: F30602-78-C-0120
GIT Project: G36-654

U.S. Air Force Rome Air Development Center (RADC)
Contract: F30602-81-C-0249
GIT Project: G36-659

U.S. Army Research Office (ARO)
Contract: DAAG29-79-C-0155
GIT Project: G36-638

U.S. Army Institute for Research in
Management Information and Computer Science (AIRMICS)
Contract: DAAK70-79-D-0087
GIT Project: G36-647

National Science Foundation (NSF)
Contract: MCS-7924370
GIT Project: G36-652

School of Information and Computer Science
Georgia Institute of Technology
Atlanta, Georgia 30332

1. INTRODUCTION

This is the Ninth Quarterly Progress Report prepared covering the Georgia Tech Research Program in Fully Distributed Processing Systems (FDPS).

a. Program Description.

The Georgia Tech Research Program in Fully Distributed Processing Systems is a comprehensive investigation of data processing systems in which both the physical and logical components are extremely loosely coupled while operating with a high degree of control autonomy at the component level. The definition of the specific class of multiple computer systems being investigated, and the operational characteristics and features of those systems is motivated by the desire to advance the state-of-the-art for that class of systems that will deliver a high proportion of the benefits currently being claimed for distributed processing systems. The scope of individual topics being investigated under this program ranges from formal modeling and theoretical studies to empirical examinations of prototype systems and simulation models. Also included within the scope of the program are areas such as the utilization of FDPS's and their interaction with management operations and structure.

b. Program Support.

The principle support for the program is a Selected Research Opportunity contract from the Office of Naval Research; however, there are a number of other sources of funding which also support the program. A list of the currently active contracts and grants is given below.

Title: "Research on Fully Distributed Data Processing Systems"
Funding Agency: Office of Naval Research (ONR)
Contract Number: N00014-79-C-0873
GIT Project No.: G36-643/336
Principle Investigator: Philip H. Enslow, Jr.

Title: "Evaluation of Distributed Control Models"
Funding Agency: U.S. Air Force Rome Air Development Center (RADC)
Contract Number: F30602-78-C-0120
GIT Project No.: G36-654
Principle Investigator: Philip H. Enslow, Jr.

Title: "System Support Capabilities for Fully-Distributed / Loosely-Coupled Processing Systems"
Funding Agency: U.S. Air Force Rome Air Development Center (RADC)
Contract Number: F30602-81-C-0249
GIT Project No.: G36-659
Principle Investigator: Philip H. Enslow, Jr.

Title: "Theory of Systems of Asynchronous Parallel Processors"
Funding Agency: U.S. Army Research Office (ARO)
Contract Number: DAAG29-79-C-0155
GIT Project Number: G36-638/332
Principle Investigator: Nancy A. Lynch

Title: "Support of MILPERCEN Data Storage Concept"
Funding Agency: U.S. Army Institute for Research in Management Information and Computer Science (AIRMICS)
Contract Number: DAAK70-79-D-0087
GIT Project Number: G36-647
Principle Investigator: Alton P. Jensen

Title: "Complexity and Computability for Distributed Data Bases"
Funding Agency: National Science Foundation (NSF)
Contract Number: MCS-7924370
GIT Project Number: G36-652/340
Principle Investigator: Nancy A. Lynch

2. ORGANIZATION AND STAFFING

Faculty

DeMillo, Richard A. - Professor
Enslow, Philip H. Jr. - Professor
Griffeth, Nancy - Assistant Professor
Jensen, Alton P. - Professor
LeBlanc, Richard - Assistant Professor
Livesey, Jon - Assistant Professor
Lynch, Nancy A. - Associate Professor
Miller, Raymond - Professor
Underwood, William - Assistant Professor

Staff

McDonell, Sharon - Administrative Secretary
Myers, Jeanette - Research Scientist
Pinion, Nancy - Part-time Secretary
Mongiovi, Roy - Research Technologist I

Students

There are approximately 30 students working on various projects in the FDPS Research Program. Of these, 12 are in the Ph.D. program, and 5 are preparing their M.S. Thesis on topics in FDPS.

3. CURRENT RESEARCH PROJECTS

The specific research projects have been organized into the major areas identified in the basic program proposal.

A. Theoretical and Formal Studies

A.2 Decomposition of Parallel Systems
A.3 Reliable Systems
A.4 Time Performance of Distributed Systems
A.5 Audit Algorithms
A.6 Ticket Systems
A.7 Synchronous Simulation
A.9 Theory of Distributed Databases
A.10 Arbiter Design
A.14 Using Complementary Distributed System Models
A.16 Stochastic Synchronization
A.17 Research Allocation in a Failure-Prone Environment
A.18 Multilevel Atomicity
A.19 Formal Semantics and Specification of Distributed Systems
A.20 Nested Transactions with Aborts

B. Physical Interconnection and Networking

B.2 Local Networking in Fully Distributed Processing Systems

C. Distributed Operating Systems

C.1 Decentralized and Distributed Control
C.2 Resource Allocation and Work Distribution in an FDPS
C.4 Local Operating System
C.5 Communications Support for Distributed Systems
C.8 Distributed Software Tools
C.9 Command Languages in an FDPS

D. Distributed Data Bases

D.1 Concurrency Control in Distributed Database Systems
D.2 Support of MILPERCEN Data Storage Concept
D.3 Implementation of the Audit Algorithm
D.4 User Interfaces to Database Systems

E. Fault-Tolerance

F. Special Hardware to Support FDPS

G. Application of Distributed Processing

H. System Design Methodologies

H.2 Coordinating Large Programming Projects

I. System Utilization

I.1 A Language for Distributed Programming
I.2 System Implementation Language Development
I.3 Experiments with a Distributed Compiler

J. Security

J.1 Process Structures
J.2 System Security

K. System Management

L. Evaluation and Comparison

L.1 Simulation of Distributed Algorithms (Griffeth, Lynch)

M. FDPS Testbed

M.1 Establishment of FDPS Testbed Facility
M.3 Fully Distributed Operating System Simulation Testbed

4. SUMMARY OF PROGRESS

A.2 Decomposition of Parallel Systems (Lynch, Fischer)

This project was completed with the submission of a report for journal publication.

A.3 Reliable Systems (Lynch, Fischer, DeMillo, Lamport, Merritt)

A lower bound on time requirements for solution to the Byzantine Generals problem, in an environment allowing authentication, has been undergoing revision for presentation at SIGACT 1982. Work is also proceeding on development of better Byzantine Generals algorithms, requiring less communication than currently known algorithms.

A.4 Time Performance of Distributed Systems (Lynch, Fischer, Lazowska, Schönhage)

Project Complete.

A.5 Audit Algorithms (Griffeth, Fischer, Lynch)

The paper "Global States of a Distributed System" is being rewritten for submission to TODS.

A.6 Ticket Systems (Fischer, Griffeth, Guibas, Lynch)

Algorithms for sequential resource allocation are being developed, to prove the upper bound on ticket allocation time when tickets are allocated one after the other. The algorithms have been written in such a way as to allow generalization to the case in which allocation of different tickets may proceed concurrently.

A.7 Synchronous Simulation (Lynch, Fischer, Arjomandi)

This project was completed with the submission of a report for journal publication.

A.9 Theory of Distributed Databases (Lynch, Griffeth)

No significant progress to report.

A.10 Arbiter Design (Lynch, Schönhage, Fischer)

Project Complete.

A.14 Using Complementary Distributed System Models (Lynch, Rounds, R. Miller)

Project Complete.

A.16 Stochastic Synchronization (DeMillo, R. Miller, Lipton)

We have begun work under the new NSF contract to extend our previous results in this area.

A.17 Resource Allocation in a Failure-Prone Environment (Fischer, Lynch, Burns, Borodin)

No significant progress to report.

A.18 Multilevel Atomicity (Lynch)

Connections between the multilevel atomicity model and the nested transaction model (as used by Leskov) are being studied.

A.19 Formal Semantics and Specification of Distributed Systems (Lynch, Stark)

Dr. Lynch is acting as major Ph.D. advisor to Eugene Stark, an MIT Ph.D. student. His thesis work involves formal models for describing distributed system behavior. The model has some similarities to previous work of Lynch and Fischer (there is no implicit synchronization, and finite delay is assumed). However, his model is at a much more abstract level than the previous work.

A.20 Nested Transactions with Aborts (Lynch, Leskov)

Dr. Lynch is studying the nested transaction structure used as a basis for Argus, Dr. Leskov's new distributed computing language. The language includes failure of subactions explicitly in its semantics. Dr. Lynch is formulating precise abstract semantics for the language, thereby attempting to prove correctness of some of the algorithms used in the implementation of Argus. A key difficulty is guaranteeing the preservation of consistency in the presence of "orphans": active processes with ancestors which have aborted.

B.2 Local Networking in FDPSs (Enslow, Myers, Manno, Brundette, Hutchins)

A technical report "Initial Experiences with a Local Network --- Net/One by Ungermann-Bass, Inc." was completed and distributed.

C.1 Decentralized and Distributed Control (Enslow, LeBlanc, Saponas)

Further simulation studies were carried out to evaluate the distributed control models. The report on the evaluation of these models has been published. A draft of Saponas's Ph.D. Dissertation was prepared for the final report. Saponas presented a defense of his thesis on this subject.

C.2 Resource Allocation and Work Distribution in an FDPS (Enslow, Sharp)

Algorithms developed for work distribution continue to be tested and evaluated using the simulator developed in project M.3.

C.4 Local Operating System (Livesey, LeBlanc, Saponas, Maccabe, Alchin, Fukuoka)

A project has been initiated to study high level inter-process communication based on a database description of a distributed system. A design of a domain-structured file system for a distributed system has been completed, and incorporated in the technical report being prepared for project C.8.

C.5 Communications Support for Distributed Systems (Enslow, Skowbo)

Work continues on identifying and describing the requirements for communications support in detail. Several network technologies are being considered to evaluate and compare their applicability to the fulfillment of these requirements. The formal project proposal is nearing completion and is currently under review.

C.8 Distributed Software Tools (Myers, Livesey, Hopkins, Lee, Fox)

Work continues on the implementation of Distributed Software Tools and preparation of a technical document describing our experiences.

C.9 Command Languages in an FDPS (Badre, Myers, Greene)

Work continues on the literature survey for command languages and the design of a "user-friendly" command language. Two abstracts have been submitted for consideration as papers for the IFIP Technical Comittee-2, Working Group 2.7 Conference on Operating Systems Interfaces to be held in September, 1982.

D.1 Concurrency Control in Distributed Database Systems (Griffeth, Livesey, Lynch)

A system for simulating concurrency control in distributed database systems has been proposed, as outlined in technical report GIT-ICS-81/15.

D.2 Support of MILPERCEN Data Storage Concept (Jensen, Doyle, Gehl, Bingham)

This project has been completed. The research conducted under this contract has identified and summarized many issues which affect the manner in which human resource information is managed for the Army. Several recommendations concerning how the Army should address itself in upgrading its Automated Manpower and Personnel Resources Management Informations Systems are made in the Final Report, "Automating the Exchange of Military Personnel Data Among Selected Army Organizations." A second report is a comprehensive set of appendices containing data collected during the course of study.

D.3 Implementation of the Audit Algorithm (Griffeth, Livesey, Lynch)

No significant progress to report.

D.4 User Interfaces to Database Systems (Griffeth)

An experimental system to test the usability and power of various database user interfaces is being developed. The actual storage structure will be based on the relational model, because it is easier to present a variety of interfaces when the underlying model is relational. However, the user data model may include such structures as CODASYL sets, repeating groups, vectors, semantic nets, or even higher level objects such as entities, relationships, aggregations, generalizations, etc. Furthermore, the user language may include navigational features as well as the ability to reference rows by value.

Currently, the underlying database is one operation short of implementation (this is the division operation). A pretest to determine the subject's background and database sophistication has been developed and is being refined. The language interpreters are in the design stage. A game for the subjects to play has been designed and is close to implementation.

H.2 Coordinating Large Programming Projects (Enslow, Underwood, Smith)

During this quarter several sources for data on large software development projets were investigated. Of the sources identified, the RADC Productivity Database was considered to be most suitable. A copy of the data was obtained and preliminary analysis begun. Noting deficiencies in the coverage of the database, a search for additional data sources will proceed in parallel with continued analysis of the data so far obtained.

I.1 A Language for Distributed Programming (LeBlanc, Maccabe)

The PRONET language features (described in technical report GIT/ICS-81/03) are currently being implemented. Most of the work done so far has been concerned with the required run-time support software.

I.2 System Implementation Language Development (LeBlanc, Wilkes)

A compiler for an extended version of Pascal is currently under development, using the code generation tool previously implemented as part of this project. This compiler will be extended to implement PRONET (project I.1) and will be used in future work in project I.3.

I.3 Experiments with a Distributed Compiler (LeBlanc, J. Miller)

No significant progress to report.

J.1 Process Structures (DeMillo, Lipton, R. Miller, Merritt, Thomas)

We have continued to develop new cryptographic protocols. This work will be summarized in a forthcoming paper.

J.2 System Security (Livesey, Davida, DeMillo)

A proposal for a secure operating system based on the paper "Secure Architectures - the Master/Slave Model" has been completed, and awaits submission to NSF.

L.1 Simulation of Distributed Algorithms (Griffeth, Lynch)

See D.1; otherwise, no significant progress due to loss of student working on simulation program.

M.1 Establishment of FDPS Testbed Facility (Myers, Mongiovi, Pitts, Fox)

No significant progress to report.

M.3 FDOS Simulation Testbed (LeBlanc, Saponas, Myers)

No significant progress to report.

5. TRAVEL RELATED TO THE FDPS PROGRAM

Date of Trip: 30 August - 4 September, 1981
Individual(s) Traveling: N. Griffeth
Itinerary: Boston, Massachusetts (MIT)
Contact: N. Lynch
Purpose: Develop distributed algorithms for ticket systems and discuss further work on performance studies of ticket systems.

Date of Trip: 14-16 October, 1981
Individual(s) Traveling: J. Livesey (for P. Enslow)
Itinerary: Rome, New York
Contact: Tom Lawrence
Purpose: RADC Technical Exchange Meeting

Date of Trip: 21 October, 1981
Individual(s) Traveling: P. Enslow (accompanied by Dr. Robert Grafton, ONR)
Itinerary: New London, Connecticut
Contact: Naval Underwater Systems Center
Purpose: Orientation on Georgia Tech FDPS Research Program

Date of Trip: 22 October, 1981
Individual(s) Traveling: P. Enslow (accompanied by Dr. Robert Grafton, ONR)
Itinerary: Newport, Rhode Island
Contact: Naval Underwater Systems Center
Purpose: Orientation on Georgia Tech FDPS Research Program

Date of Trip: October, 1981
Individual(s) Traveling: N. Lynch
Itinerary: Nashville, Tennessee
Purpose: Attend FOCS Conference

Date of Trip: November, 1981
Individual(s) Traveling: N. Lynch
Itinerary: New Haven, Connecticut
Contact: Mike Fischer
Purpose: Work on projects and give presentation on latest progress.

6. VISITORS

Dates of Visit: September, 1981
Visitor: M. Fischer
Contact: N. Lynch (at MIT)
Purpose: Work on various projects.

Dates of Visit: 23 October, 1981
Visitor: Herman J. Weegenaar (Centraal Beheer, The Netherlands)
Contact: P. Enslow
Purpose: Discuss distributed operating system functions

Dates of Visit: 29-30 October, 1981
Visitor: Tom Lawrence (RADC) and Rudy Nothdurft
Contact: P. Enslow
Purpose: Review and planning session on new RADC contract

7. PUBLICATIONS

Author(s): M. Fischer, N. Lynch
Title: A Lower Bound for the Time to Assure Interactive Consistency
Type: technical report
Status: published
GIT Number: GIT-ICS-81/13
Publ. Date: September, 1981

Author(s): P. Enslow, P. Manno, and J. Myers
Title: Initial Experience with a Local Network - NET/ONE by Ungermann-Bass
Type: technical report
Status: published
GIT Number: GIT-ICS-81/11
Publ. Date: October, 1981

Author(s): R. DeMillo, N. Lynch, and M. Merritt
Title: Cryptographic Protocols
Status: abstract submitted to conference
Publ. Date: Winter '82

Author(s): A. Jensen, J. Bingham, J. Doyle, and J. Gehl
Title: Automating the Exchange of Military Personnel Data Among Selected Army Organizations
Type: final report
Status: published
Publ. Date: June, 1981

Author(s): A. Jensen, J. Bingham, J. Doyle, and J. Gehl
Title: Automating the Exchange of Military Personnel Data Among Selected Army Organizations
Type: appendices
Status: published
Publ. Date: June, 1981

THE GEORGIA INSTITUTE OF TECHNOLOGY

RESEARCH PROGRAM IN

FULLY DISTRIBUTED PROCESSING SYSTEMS

Quarterly Progress Report Number 10
1 December, 1981 - 28 February, 1982

March, 1982

Supported by

Office of Naval Research (ONR)
Contract: N00014-79-C-0873
GIT Project: G36-643

U.S. Air Force Rome Air Development Center (RADC)
Contract: F30602-78-C-0120
GIT Project: G36-654

U.S. Air Force Rome Air Development Center (RADC)
Contract: F30602-81-C-0249
GIT Project: G36-659

U.S. Army Research Office (ARO)
Contract: DAAG29-79-C-0155
GIT Project: G36-638

National Science Foundation (NSF)
Contract: MCS-7924370
GIT Project: G36-652

School of Information and Computer Science
Georgia Institute of Technology
Atlanta, Georgia 30332

1. INTRODUCTION

This is the Tenth Quarterly Progress Report prepared covering the Georgia Tech Research Program in Fully Distributed Processing Systems (FDPS).

a. Program Description.

The Georgia Tech Research Program in Fully Distributed Processing Systems is a comprehensive investigation of data processing systems in which both the physical and logical components are extremely loosely coupled while operating with a high degree of control autonomy at the component level. The definition of the specific class of multiple computer systems being investigated, and the operational characteristics and features of those systems is motivated by the desire to advance the state-of-the-art for that class of systems that will deliver a high proportion of the benefits currently being claimed for distributed processing systems. The scope of individual topics being investigated under this program ranges from formal modeling and theoretical studies to empirical examinations of prototype systems and simulation models. Also included within the scope of the program are areas such as the utilization of FDPS's and their interaction with management operations and structure.

b. Program Support.

The principle support for the program is a Selected Research Opportunity contract from the Office of Naval Research; however, there are a number of other sources of funding which also support the program. A list of the currently active contracts and grants is given below.

Title: "Research on Fully Distributed Data Processing Systems"
Funding Agency: Office of Naval Research (ONR)
Contract Number: N00014-79-C-0873
GIT Project No.: G36-643/336
Principle Investigator: Philip H. Enslow, Jr.

Title: "Evaluation of Distributed Control Models"
Funding Agency: U.S. Air Force Rome Air Development Center (RADC)
Contract Number: F30602-78-C-0120
GIT Project No.: G36-654
Principle Investigator: Philip H. Enslow, Jr.

Title: "System Support Capabilities for Fully-Distributed / Loosely-Coupled Processing Systems"
Funding Agency: U.S. Air Force Rome Air Development Center (RADC)
Contract Number: F30602-81-C-0249
GIT Project No.: G36-659
Principle Investigator: Philip H. Enslow, Jr.

Title: "Theory of Systems of Asynchronous Parallel Processors"
Funding Agency: U.S. Army Research Office (ARO)
Contract Number: DAAG29-79-C-0155
GIT Project Number: G36-638/332
Principle Investigator: Nancy A. Lynch

Title: "Complexity and Computability for Distributed Data Bases"
Funding Agency: National Science Foundation (NSF)
Contract Number: MCS-7924370
GIT Project Number: G36-652/340
Principle Investigator: Nancy A. Lynch

2. ORGANIZATION AND STAFFING

Faculty

DeMillo, Richard A. - Professor
Enslow, Philip H. Jr. - Professor
Griffeth, Nancy A. - Assistant Professor
Jensen, Alton P. - Professor
LeBlanc, Richard J. - Assistant Professor
Livesey, Jon - Assistant Professor
Lynch, Nancy A. - Associate Professor
McKendry, Martin S. - Assistant Professor
Miller, Raymond - Professor
Underwood, William - Assistant Professor

Staff

McDonell, Sharon - Administrative Secretary
Myers, Jeanette - Research Scientist
Pinion, Nancy - Part-time Secretary
Mongiovi, Roy - Research Technologist I

Students

There are approximately 30 students working on various projects in the FDPS Research Program. Of these, 12 are in the Ph.D. program, and 5 are preparing their M.S. Thesis on topics in FDPS.

3. CURRENT RESEARCH PROJECTS

The specific research projects have been organized into the major areas identified in the basic program proposal.

A. Theoretical and Formal Studies

A.3 Reliable Systems
A.4 Time Performance of Distributed Systems
A.5 Audit Algorithms
A.6 Ticket Systems
A.9 Theory of Distributed Databases
A.16 Stochastic Synchronization

A.17 Research Allocation in a Failure-Prone Environment
A.18 Multilevel Atomicity
A.19 Formal Semantics and Specification of Distributed Systems
A.20 Nested Transactions with Aborts

B. Physical Interconnection and Networking

B.2 Local Networking in Fully Distributed Processing Systems

C. Distributed Operating Systems

C.1 Decentralized and Distributed Control
C.2 Resource Allocation and Work Distribution in an FDPS
C.4 Local Operating System
C.5 Communications Support for Distributed Systems
C.8 Distributed Software Tools
C.9 Command Languages in an FDPS
C.10 Distributed Operating System Implementation

D. Distributed Data Bases

D.1 Concurrency Control in Distributed Database Systems
D.3 Implementation of the Audit Algorithm
D.4 User Interfaces to Database Systems

E. Fault-Tolerance

F. Special Hardware to Support FDPS

G. Application of Distributed Processing

H. System Design Methodologies

H.2 Coordinating Large Programming Projects

I. System Utilization

I.1 A Language for Distributed Programming
I.2 System Implementation Language Development
I.3 Experiments with a Distributed Compiler

J. Security

J.1 Process Structures
J.2 System Security

K. System Management

L. Evaluation and Comparison

L.1 Simulation of Distributed Algorithms (Griffeth, Lynch)

M. FDPS Testbed

M.1 Establishment of FDPS Testbed Facility
M.3 Fully Distributed Operating System Simulation Testbed

4. SUMMARY OF PROGRESS

A.3 Reliable Systems (Lynch, Fischer, Fowler, Lamport, Merritt)

A new Byzantine Generals algorithm was devised, with better performance than previously known algorithms, in terms of amount of message traffic. The number of rounds is also considerably smaller than in previous algorithms with good message performance.

The lower bound proof for number of rounds in an environment allowing arbitrary authentication capabilities, has been considerably refined and clarified.

A.4 Time Performance of Distributed Systems (Lynch, Fischer, Lazowska, Schönhage)

No significant progress to report.

A.5 Audit Algorithms (Griffeth, Fischer, Lynch)

The paper, "Global States of a Distributed System", was accepted for a special issue of Transactions on Software Engineering (May, 1982). Also, work is in progress on analysis of some significant special-case systems.

A.6 Ticket Systems (Fischer, Griffeth, Guibas, Lynch)

The ticket system simulation has been modified to reduce the variance in results and generalized to allow returns and reallocation of tickets. Initial tests have been run. Further modifications are planned to test limiting cases.

A.9 Theory of Distributed Databases (Lynch, Griffeth)

No significant progress to report.

A.16 Stochastic Synchronization (DeMillo, R. Miller, Lipton)

No significant progress to report.

A.17 Resource Allocation in a Failure-Prone Environment (Fischer, Lynch, Burns, Borodin)

No significant progress to report.

A.18 Multilevel Atomicity (Lynch)

Some work is being done to integrate the multilevel atomicity concept with related work on nested transactions.

A.19 Formal Semantics and Specification of Distributed Systems (Lynch, Stark)

Equivalence has been proved for three natural definitions for the class of possible behaviors of distributed systems.

A.20 Nested Transactions with Aborts (Lynch, Leskov)

Formal correctness conditions for nested transaction systems, and correctness proofs for implementations using locking, are in process of development.

B.2 Local Networking in FDPSs (Enslow, Myers, Brundette, Hutchins, Arius)

A 12-1/2 hour video Net/One Programming class created by Ungermann-Bass was taken by members of this project. This course described in detail Net/One operation and the recently released software development support package.

We are waiting for the delivery of a new component for the local network, an NCF-2, which replaces the MCZ. (The MCZ is no longer supported by Ungermann-Bass). The NCF-2 includes the software development support package, support for the C programming language and, due to an IEEE-448/79 interface to the local network, reduces downloading time from approximately 90 seconds to 19 seconds per board.

C.1 Decentralized and Distributed Control (Enslow, LeBlanc, Saponas)

A technical report concluding Tim Saponas's research in this area is ready for distribution. This report analyzes in detail the factors contributing to the particular evaluations of each of the control models covered in a previously distributed technical report.

C.2 Resource Allocation and Work Distribution in an FDPS (Enslow, Sharp)

A technical report concluding Don Sharp's research in this area is ready for distribution. Results of simulation experiments indicated that the best criterion for work distribution is to minimize communication between processes represented as nodes of a task graph.

C.4 Local Operating System (Livesey, LeBlanc, McKendry, Myers, Allchin, Fukuoka, Maccabe, Pitts, Spafford)

Work continues on the identification and research of those design concepts related to distributed operating systems. This includes atomicity, consistency, recovery, fault tolerence, interprocess communication, file systems, etc. Of special importance is the area of distributed databases. Due to the amount and distribution of information required by a distributed operating system, we are looking into building a distributed database as an intrinsic part of the operating system.

Project C.10 has been created to attempt a "first try" implementation of this combination of data base and operating system.

C.5 Communications Support for Distributed Systems (Enslow, Skowbo)

The formal project proposal is essentially complete and is currently being reviewed, revised, and extended. The specification and design of tools for an experimental evaluation of proposed alternatives for communications support is also in progress.

C.8 Distributed Software Tools (Myers, Livesey, Hopkins, Fox)

Work continues on the implementation of Distributed Software Tools and preparation of a technical report describing our experiences.

C.9 Command Languages in an FDPS (Badre, Myers, Greene)

During this quarter, the main effort has been directed toward characterizing the user, his requirements, and his needs. To accomplish this in part, user activity has been monitored and these data processed to determine how people are using the current facilities. We are studying what commands are being used and in what general class of processing these commands are used (e.g. text manipulation, program development, etc.). Also, the commands available in the same system have been examined to determine what these commands functionally provide to a user. The information gathered from these approaches is being used as a strong basis for the friendly user interface that is the goal of this work.

C.10 Distributed Operating System Implementation (McKendry, Allchin, Thibault, Maccabe)

Realizing that there remain many unsolved problems and open areas of research related to distributed operating systems, we believe that it

would be beneficial to begin an implementation based on what we currently know that we can do. We have named our prototype operating system "CLOUDS", an acronym for "Coalescing Local Operating Systems Under Decentralized Supervision." The descriptor "Coalescing" is an important one. CLOUDS will be capable of stand-alone operation, but when networked with other CLOUDS, will naturally come together or "coalesce" into one distributed operating system.

Current efforts are centered around designing a scaled down version of such an operating system for the first implementation.

D.1 Concurrency Control in Distributed Database Systems (Griffeth, Livesey, Lynch)

No significant progress to report.

D.3 Implementation of the Audit Algorithm (Griffeth, Livesey, Lynch)

No significant progress to report.

D.4 User Interfaces to Database Systems (Griffeth)

Several interfaces have been completed for the PR1ME relational database. They are: (1) a relational algebra interface, (2) a relational calculus interface, and (3) a network interface. Preliminary tests on the effectiveness of these interfaces will begin next quarter.

A project, "Distributed Database Algorithms", will be funded by NSF for the period July 1982 - June 1984.

H.2 Coordinating Large Programming Projects (Enslow, Smith)

A number of additional data sources were identified. A tentative metric for the quality of communication activities during large software development was proposed and is currently under investigation.

I.1 A Language for Distributed Programming (LeBlanc, Maccabe)

Design work has been completed; implementation and evaluation are in progress. A paper was submitted to the Third International Conference on Distributed Computing Systems.

I.2 System Implementation Language Development (LeBlanc, McKendry, Wilkes)

Implementation of a Pascal compiler, using the code generator previously developed under this project, is nearly complete. Design of extensions for system implementation support is in progress. Implementation should begin during next quarter.

I.3 Experiments with a Distributed Compiler (LeBlanc, J. Miller)

Work by Miller has confirmed and considerably improved earlier results. Design for balanced message flows was found to be crucial for best performance. A paper was submitted to the Third International Conference on Distributed Computing Systems.

J.1 Process Structures (DeMillo, Lipton, R. Miller, Merritt, Thomas)

No significant progress to report.

J.2 System Security (Livesey, Davida, DeMillo)

No significant progress to report.

L.1 Simulation of Distributed Algorithms (Griffeth, Lynch)

No significant progress to report.

M.1 Establishment of FDPS Testbed Facility (Myers, Mongiovi, Fox)

No significant progress to report.

M.3 FDOS Simulation Testbed (LeBlanc, Saponas, Myers)

No significant progress to report.

5. TRAVEL RELATED TO THE FDPS PROGRAM

Date of Trip: 25-27 January, 1982
Individual(s) Traveling: Richard LeBlanc
Itinerary: Albuquerque, New Mexico
Purpose: Attend ACM Principles of Programming Languages Symposium

Date of Trip: February, 1982
Individual(s) Traveling: Nancy A. Lynch
Itinerary: Brandeis University, Northeastern University, Boston University
Purpose: Speak about new Byzantine Generals algorithm.

6. VISITORS

Dates of Visit: February, 1982
Visitor: Bharat Bhargava
Contact: Nancy A. Lynch (at MIT)
Purpose: Discussions about correctness proofs of concurrency control algorithms, and about reliability properties of distributed algorithms.

7. PUBLICATIONS

Author(s): John A. Miller and Richard J. LeBlanc
Title: Distributed Compilation: A Case Study
Type: conference paper
Status: submitted

Author(s): Arthur B. Maccabe and Richard J. LeBlanc
Title: The Design of a Programming Language Based on Communication Networks
Type: conference paper
Status: submitted

Author(s): Michael J. Fischer, Nancy D. Griffeth, and Nancy A. Lynch
Title: Global States of a Distributed System
Type: journal paper
Status: accepted for publication in Transactions on Software Engineering
Publ. Date: May, 1982

Author(s): Nancy A. Lynch
Title: Multilevel Atomicity
Type: conference paper
Status: accepted by Principles of Database Systems Conference
Publ. Date: March, 1982

Author(s): Nancy A. Lynch, Michael J. Fischer, and Robert Fowler
Title: A Simple and Efficient Byzantine Generals Algorithm
Type: conference paper
Status: submitted to IEEE Symposium on Reliability in Distributed Software and Database Systems.

Author(s): Richard A. DeMillo, Nancy A. Lynch, and Michael Merritt
Title: Cryptographic Protocols
Type: conference paper
Status: accepted by SIGACT

THE GEORGIA INSTITUTE OF TECHNOLOGY

RESEARCH PROGRAM IN

FULLY DISTRIBUTED PROCESSING SYSTEMS

Quarterly Progress Report Number 11
1 March, 1982 - 31 May, 1982

August, 1982

Supported by

Office of Naval Research (ONR)
Contract: N00014-79-C-0873
GIT Project: G36-643

U.S. Air Force Rome Air Development Center (RADC)
Contract: F30602-78-C-0120
GIT Project: G36-654

U.S. Air Force Rome Air Development Center (RADC)
Contract: F30602-81-C-0249
GIT Project: G36-659

U.S. Army Research Office (ARO)
Contract: DAAG29-79-C-0155
GIT Project: G36-638

National Science Foundation (NSF)
Contract: MCS-7924370
GIT Project: G36-652

School of Information and Computer Science
Georgia Institute of Technology
Atlanta, Georgia 30332

1. INTRODUCTION

This is the Eleventh Quarterly Progress Report prepared covering the Georgia Tech Research Program in Fully Distributed Processing Systems (FDPS).

a. Program Description.

The Georgia Tech Research Program in Fully Distributed Processing Systems is a comprehensive investigation of data processing systems in which both the physical and logical components are extremely loosely coupled while operating with a high degree of control autonomy at the component level. The definition of the specific class of multiple computer systems being investigated, and the operational characteristics and features of those systems is motivated by the desire to advance the state-of-the-art for that class of systems that will deliver a high proportion of the benefits currently being claimed for distributed processing systems. The scope of individual topics being investigated under this program ranges from formal modeling and theoretical studies to empirical examinations of prototype systems and simulation models. Also included within the scope of the program are areas such as the utilization of FDPS's and their interaction with management operations and structure.

b. Program Support.

The principle support for the program is a Selected Research Opportunity contract from the Office of Naval Research; however, there are a number of other sources of funding which also support the program. A list of the currently active contracts and grants is given below.

Title: "Research on Fully Distributed Data Processing Systems"
Funding Agency: Office of Naval Research (ONR)
Contract Number: N00014-79-C-0873
GIT Project No.: G36-643/336
Principle Investigator: Philip H. Enslow, Jr.

Title: "Evaluation of Distributed Control Models"
Funding Agency: U.S. Air Force Rome Air Development Center (RADC)
Contract Number: F30602-78-C-0120
GIT Project No.: G36-654
Principle Investigator: Philip H. Enslow, Jr.

Title: "System Support Capabilities for Fully-Distributed / Loosely-Coupled Processing Systems"
Funding Agency: U.S. Air Force Rome Air Development Center (RADC)
Contract Number: F30602-81-C-0249
GIT Project No.: G36-659
Principle Investigator: Philip H. Enslow, Jr.

Title: "Theory of Systems of Asynchronous Parallel Processors"
Funding Agency: U.S. Army Research Office (ARO)
Contract Number: DAAG29-79-C-0155
GIT Project Number: G36-638/332
Principle Investigator: Nancy A. Lynch

Title: "Complexity and Computability for Distributed Data Bases"
Funding Agency: National Science Foundation (NSF)
Contract Number: MCS-7924370
GIT Project Number: G36-652/340
Principle Investigator: Nancy A. Lynch

2. ORGANIZATION AND STAFFING

Faculty

DeMillo, Richard A. - Professor
Enslow, Philip H. Jr. - Professor
Griffeth, Nancy A. - Assistant Professor
Jensen, Alton P. - Professor
LeBlanc, Richard J. - Assistant Professor
Livesey, Jon - Assistant Professor
Lynch, Nancy A. - Associate Professor (currently visiting at MIT)
McKendry, Martin S. - Assistant Professor
Miller, Raymond - Professor
Underwood, William - Assistant Professor

Staff

McDonell, Sharon - Administrative Secretary
Myers, Jeanette - Research Scientist
Pinion, Nancy - Part-time Secretary
Mongiovi, Roy - Research Technologist I

Students

There are approximately 30 students working on various projects in the FDPS Research Program. Of these, 12 are in the Ph.D. program, and 5 are preparing their M.S. Thesis on topics in FDPS.

3. CURRENT RESEARCH PROJECTS

The specific research projects have been organized into the major areas identified in the basic program proposal.

A. Theoretical and Formal Studies

A.3 Reliable Systems
A.4 Time Performance of Distributed Systems
A.5 Audit Algorithms
A.6 Ticket Systems
A.9 Theory of Distributed Databases
A.16 Stochastic Synchronization

A.17 Research Allocation in a Failure-Prone Environment
A.18 Multilevel Atomicity
A.19 Formal Semantics and Specification of Distributed Systems
A.20 Nested Transactions with Aborts

B. Physical Interconnection and Networking

B.2 Local Networking in Fully Distributed Processing Systems

C. Distributed Operating Systems

C.4 Local Operating System
C.5 Communications Support for Distributed Systems
C.8 Distributed Software Tools
C.9 Command Languages in an FDPS
C.10 Distributed Operating System Implementation

D. Distributed Data Bases

D.1 Concurrency Control in Distributed Database Systems
D.3 Implementation of the Audit Algorithm
D.4 User Interfaces to Database Systems

E. Fault-Tolerance

F. Special Hardware to Support FDPS

G. Application of Distributed Processing

H. System Design Methodologies

H.2 Coordinating Large Programming Projects

I. System Utilization

I.1 A Language for Distributed Programming
I.2 System Implementation Language Development
I.3 Experiments with a Distributed Compiler

J. Security

J.1 Process Structures
J.2 System Security

K. System Management

L. Evaluation and Comparison

L.1 Simulation of Distributed Algorithms (Griffeth, Lynch)

M. FDPS Testbed

M.1 Establishment of FDPS Testbed Facility
M.3 Fully Distributed Operating System Simulation Testbed
M.4 Interactive Monitoring of Distributed Programs

4. SUMMARY OF PROGRESS

A.3 Reliable Systems (Lynch, Fischer, Fowler, Merritt)

A paper, "Cryptographic Protocols", was presented at SIGACT 82.

A.4 Time Performance of Distributed Systems (Lynch, Fischer, Lazowska, Schönhage)

No significant progress to report.

A.5 Audit Algorithms (Griffeth, Fischer, Lynch)

No significant progress to report.

A.6 Ticket Systems (Fischer, Griffeth, Guibas, Lynch)

A draft of a paper, "Analysis of a Network Resource Allocation Algorithm", has been prepared for presentation at the June ACM workshop on probabilistic complexity.

A.9 Theory of Distributed Databases (Lynch, Griffeth)

Initial work on a new formulation of concurrency control, providing a more basic definition of correctness than earlier work, was carried out.

A.16 Stochastic Synchronization (DeMillo, R. Miller, Lipton)

No significant progress to report.

A.17 Resource Allocation in a Failure-Prone Environment (Fischer, Lynch, Burns, Borodin)

No significant progress to report.

A.18 Multilevel Atomicity (Lynch)

No significant progress to report.

A.19 Formal Semantics and Specification of Distributed Systems (Lynch, Stark)

The model was used to specify and prove correct a simple arbiter algorithm. Both safety and fairness properties are easily expressed and proved.

A.20 Nested Transactions with Aborts (Lynch, Liskov)

A paper, "Concurrency Control for Resilient Nested Transactions", was written. It defines the semantics of resilient nested transactions, and uses the framework for proving correctness of a version of Moss's locking algorithm (the implementation of nested transactions used in the Argus system).

B.2 Local Networking in FDPSs (Enslow, Myers, Brundette, Hutchins, Arius)

The Net/One Network Configuration Facility (NCF-2) has been successfully installed and performs according to published specificaions. A Whitesmith C compiler has been received for the NCF-2 to be used for local software development.

C.4 Local Operating System (Livesey, Fukuoka)

A paper by Fukuoka presenting a comprehensive taxonomy of IPC facilities based on the semantic aspects associated with the IPC has been submitted to ACM Transactions on Programming Languages and Systems (TOPLAS). This paper has been forwarded by TOPLAS for review by ACM Computing Surveys. The paper includes a survey of IPC mechanisms in fifteen existing and proposed programming languages and systems for distributed processing.

C.5 Communications Support for Distributed Systems (Enslow, Skowbo)

The formal project proposal is essentially complete and is currently being reviewed, revised, and extended. The specification and design of tools for an experimental evaluation of proposed alternatives for communications support is also in progress.

C.8 Distributed Software Tools (Myers, Livesey, Hopkins, Lee, Fox)

Hosts, which initiate additional user processes to perform functions on local and remote hosts, have been implemented but require some modification as to the "initial" state of the processes. Since we have found the overhead of "logging in" another process to be high, we are proposing a special network-server process to be used instead of an additional user process when a specific user environment is not required.

The domain-structured file system has been implemented and awaits testing and incorporation into DSWT. A paper describing the file system was presented at the ACM Southeastern Regional Conference, April,1982, in Knoxville, Tennessee.

An IPC facility has been incorporated into the SWT I/O Subsystem. Currently, this facility requires that virtual circuit connections be made to specific ports on specific systems. This is being modified so that port numbers and system names will be completely transparent to the user program.

C.9 Command Languages in an FDPS (Badre, Myers, Greene)

During the past quarter, work on characterizing the user, his requirements, and his needs has continued. User activity was monitored throughout the quarter resulting in a data set that spans one year. This information has been gathered in an effort to understand the differences, if any, between "guru" or sophisticated users and "novice" or unsophisticated users. It is hoped that this will result in a definition of the extent of function a user will use in an interface. Specifically, the processing task has included calculating frequency counts of the commands used by both groups and sampling of the terminal sessions of members of both groups of users. Again, the purpose of the analysis of this data is to understand what is needed in an interface--to understand the difference in functional need and use of a computer by a user, whether a sophisticated or unsophisticated user.

C.10 Distributed Operating System Implementation (McKendry, Allchin, Thibault)

The CLOUDS project is constructing a distributed operating system for a group of workstations connected by a high-speed local-area network. The fundamental aims of the project are to provide a testbed for evaluation of algorithms developed within the FDPS program and to evaluate structural concepts for distributed operating systems.

The overall structure of the operating system has been defined, the interprocess communication mechanism has been designed and documented in an internal working paper, and considerable progress has been made in the study of data consistency requirements for the system.

Research is currently concentrating on data management and resource management. An implementation of the kernel for PERQ workstations is underway, and preparation of a conference paper is in progress.

D.1 Concurrency Control in Distributed Database Systems (Griffeth, Livesey, Lynch)

Correction to the March 1982 Quarterly Progress Report: The project, "Distributed Database Algorithms", reported under Project D.4, "User Interfaces", should have been reported under this heading.

Work is underway on analyzing preliminary hypotheses and developing the simulation. Currently, special attention is being paid to establishing results which will simplify the simulation (e.g., read/write mix has no effect, the same ratio of transaction load to database size has the same effect regardless of absolute database size, a database with a skewed distribution of frequencies of data-item requests behaves like a smaller database with a uniform distribution). Combinatoric and queuing-theoretic techniques are being used.

D.3 Implementation of the Audit Algorithm (Griffeth, Livesey, Lynch)

No significant progress to report.

D.4 User Interfaces to Database Systems (Griffeth)

Pilot studies have been run using a "cops-and-robbers" game and a registration problem to test the effectiveness of the relational calculus in a problem-solving situation. Preliminary indications are that each subject will required more than three hours.

H.2 Coordinating Large Programming Projects (Enslow, Smith)

The proposed metric for effectiveness of communication during large software development has been refined. A major focus has been the determination of the criteria that such a metric must meet. A series of experiments has been planned to evaluate the proposed metric.

I.1 A Language for Distributed Programming (LeBlanc, Maccabe, Mongiovi)

Intensive work on implementation and evaluation has continued this quarter. Work has also been done to further develop features for handling process and processor failures. Maccabe's Ph.D. thesis is in preparation.

I.2 System Implementation Language Development (LeBlanc, McKendry, Wilkes)

Work continues on the Pascal compiler which is intended to be the basis of our implementation.

I.3 Experiments with a Distributed Compiler (LeBlanc, J. Miller)

Further experiments have been conducted to study the effects of buffering messages. This concept appeared as a major factor in our earlier studies. A journal paper based on these experiments is in preparation.

J.1 Process Structures (DeMillo, Lipton, R. Miller, Merritt, Thomas)

No significant progress to report.

J.2 System Security (Livesey, Davida, DeMillo)

No significant progress to report.

L.1 Simulation of Distributed Algorithms (Griffeth, Lynch)

Hardware and software selection are underway. The initial simulations will be run on a PRIME 550, VAX 780, or CYBER 170/760. Available simulation languages include GPSS, SIMULA, and SIMSCRIPT. The ticket system simulation written in FORTRAN includes event list management and statistical routines.

M.1 Establishment of FDPS Testbed Facility (Myers, Mongiovi, Fox)

Test facilities are being developed in conjunction with development work in Projects C.8 and C.10.

M.3 FDOS Simulation Testbed (LeBlanc, Saponas, Myers)

No significant progress to report.

M.4 Interactive Monitoring of Distributed Programs (LeBlanc, Robbins)

With the development of distributed computing systems, it becomes necessary to provide programmers with appropriate tools to effectively utilize them. The first required tool is a programming language which supports the design and construction of distributed programs. One such language, called PRONET, has been developed as part of the FDPS Research Program (see Project I.1). The newly initiated project described here is concerned with the next required tool: a monitor which will allow programmers to examine the behavior of distributed programs, interactively. Monitoring a distributed program presents significant new challenges, since the "state" of such a program involves information about an arbitrary number of processes running on a number of machines. This problem is far more complex than monitoring a typical program on a single machine, in which case, all of the state information is in a single address space. The desired monitoring capability should be generalized, so that it can be used both for debugging and performance analysis. Such generality is a reasonable goal, since the most important aspect of monitoring distributed programs will concern the collection of data about the interactions among program parts, a task that is independent of the use intended for the data.

Initial work on the design of a monitoring system has begun and proposals have been prepared and submitted in order to obtain support for this work.

5. TRAVEL RELATED TO THE FDPS PROGRAM

Date of Trip: 18-20 May, 1982
Individual(s) Traveling: Philip Enslow
Itinerary: Rome Air Development Center
Contact: Tom Lawrence
Purpose: Participate in RADC Distributed Processing Technology Exchange Meeting

Date of Trip: May, 1982
Individual(s) Traveling: Nancy Lynch
Itinerary: Marina del Rey
Contact:
Purpose: Attended Symposium on Principles of Database Systems. Presented the paper, "Multilevel Atomicity".

6. VISITORS

No visitors to report.

7. PUBLICATIONS

Author(s): T. Allen Akin and Richard J. LeBlanc
Title: The Design and Implementation of a Code Generation Tool
Type: journal paper
Status: accepted for publication in Software - Practice and Experience

Author(s): Hirobumi Fukuoka
Title: Interprocess Communication Facilities for Distributed Systems: A Taxonomy and a Survey
Type: journal paper
Status: submitted to ACM Transactions on Programming Languages and Systems; subsequently forwarded to ACM Computing Surveys for review.
GIT Number: GIT-ICS-82/06

Author(s): N.J. Livesey
Title: Extending File Systems to Distributed Systems
Type: conference paper
Status: presented at the ACM April Southeastern Regional Conference
GIT Number: GIT-ICS-82/07
Publ. Date: April, 1982

Author(s): Richard DeMillo, Nancy Lynch, Michael Merritt
Title: Cryptographic Protocols
Type: conference paper
Status: presented
GIT Number: TBA
Publ. Date: May, 1982

DISTRIBUTED AND DECENTRALIZED CONTROL

IN

FULLY DISTRIBUTED PROCESSING SYSTEMS

A Survey of Applicable Models

FINAL TECHNICAL REPORT

GIT-ICS-81/02

15 January 1980 - 30 September 1980

Philip H. Enslow, Jr.
Timothy G. Saponas

February, 1981

Rome Air Development Center (ISCP)
Department of the Air Force
Griffiss Air Force Base, New York 13441

Contract Number F30602-78-C-0120
GIT Project Number G36-649

The Georgia Tech Research Program in
Fully Distributed Processing Systems
School of Information and Computer Science
Georgia Institute of Technology
Atlanta, Georgia 30332

THE VIEW, OPINIONS, AND/OR FINDINGS CONTAINED IN THIS REPORT ARE THOSE OF THE AUTHORS AND SHOULD NOT BE CONSTRUED AS AN OFFICIAL DEPARTMENT OF THE AIR FORCE POSITION, POLICY, OR DECISION, UNLESS SO DESIGNATED BY OTHER DOCUMENTATION.

unclassified

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER GIT-ICS-81/02	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* DISTRIBUTED AND DECENTRALIZED CONTROL IN FULLY DISTRIBUTED PROCESSING SYSTEMS --- A Survey of Applicable Models		5. TYPE OF REPORT & PERIOD COVERED Final Technical Report 15 Jan 80 - 30 Sept 80 6. PERFORMING ORG. REPORT NUMBER GIT-ICS-81/02
7. AUTHOR(s) Philip H. Enslow Jr. Timothy G. Saponas		8. CONTRACT OR GRANT NUMBER(s) F30602-78-C-0120
9. PERFORMING ORGANIZATION NAME AND ADDRESS School of Information and Computer Science Georgia Institute of Technology Atlanta, Georgia 30332		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Rome Air Development Center (ISCP) Department of the Air Force Griffiss Air Force Base, New York 13441		12. REPORT DATE February, 1981 13. NUMBER OF PAGES 101 + ix
14. MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)* same as item 11		15. SECURITY CLASS. *(of this report)* Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution limited.

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

same

18. SUPPLEMENTARY NOTES

RADC Project Engineer: Thomas F. Lawrence (ISCP)

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 Air Force position, policy, or decision, unless so designated by other documentation.

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

Control
Decentralized Control
Distributed Processing
Fully Distributed Processing Systems
Network
Network Operating System

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

Parallel processing has been a popular approach to improving system performance through several generations of computer systems design. Although it is not usually characterized as a "parallel" processing system, a distributed processing system has the inherent capability for highly parallel operation. In order to capitalize on the potential performance improvements achievable by a distributed system, major parallel control problems must be solved. Central to the issue of parallel control is the design and implementation of distributed and decentralized control. The study of distributed and decentralized control was

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

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

initiated with a survey of applicable control models. The results of this survey are presented along with an extensive discussion of the control problems applicable to distributed systems --- specifically "fully" distributed systems.

ABSTRACT

Parallel processing has been a popular approach to improving system performance through several generations of computer systems design. Although it is not usually characterized as a "parallel" processing system, a distributed processing system has the inherent capability for highly parallel operation. In order to capitalize on the potential performance improvements achievable by a distributed system, major parallel control problems must be solved. Central to the issue of parallel control is the design and implementation of distributed and decentralized control. The study of distributed and decentralized control was initiated with a survey of applicable control models. The results of this survey are presented along with an extensive discussion of the control problems applicable to distributed systems --- specifically "fully" distributed systems.

PREFACE

Comments from the Principal Investigator

Although this is the final report on only one of the approximately 30 research projects currently being performed in the Georgia Tech research program on Fully Distributed Processing Systems, it serves a much broader function than just reporting on the work done in this single project. Since this is the first major technical report published under the program, it has been necessary to document here much of the background applying to the program in general. Specifically, this report presents an extensive discussion of the general philosophies of fully distributed control and fully distributed processing as well as the notation that has been developed to describe the control actions supporting such processing activities.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

SECTION 1

BACKGROUND

1.1 GOALS OF COMPUTER SYSTEM DEVELOPMENT

Although the state of the art in digital computers has certainly been advancing faster than any other technological area in history, it is somewhat remarkable that the goals motivating most computer system development projects have remained basically unchanged since the earliest days. Perhaps the most important of these long sought-after improvements are the following:

1. Increased system productivity
   - Greater capacity
   - Shorter response time
   - Increased throughput
2. Improved reliability and availability
3. Ease of system expansion and enhancement
4. Graceful growth and degradation
5. Improved ability to share system resources

The "final or ultimate values" for these various goals cannot be expressed in absolute numbers, so it is not surprising that they continue to apply even though phenomenal advances have been made in many of them such as speed, capacity, and reliability. What is perhaps more noteworthy and important to the discussion being presented here is how little progress has been made in areas such as easy modular growth, availability, adaptability, etc.

It seems that each new major systems concept or development (e.g., multiprogramming, multiprocessing, networking, etc.) has been presented as "the answer" to achieving all of the goals listed above plus many others. "Distributed processing" is no exception to this rule. In fact, many salesmen have dusted off their old lists of benefits and are marketing today's distributed systems as the means to achieve all of them. Table 1 lists some of the benefits currently being claimed for distributed processing systems in current sales literature. Although some forms of distributed processing appear to offer great promise as a possible means to make significant advances in many of the areas listed, the state-of-the-art, particularly in system control software, is far from being able to deliver even a significant proportion of these benefits today.

Table 1. "Benefits" Provided by Distributed Processing Systems

A Representative List Assembled from Claims Made in Actual Sales Literature

High Availability and Reliability

Reduced Network Costs

High System Performance

Fast Response Time

High Throughput

Graceful Degradation, Fail-soft

Ease of Modular and Incremental Growth

Configuration Flexibility

Automatic Load and Resource Sharing

Easily Adaptable to Changes in Workload

Incremental Replacement and/or Upgrade

Easy Expansion in Capacity and/or Function

Good Response to Temporary Overloads

1.2 APPROACHES TO IMPROVING SYSTEM PERFORMANCE

Efforts to improve the performance of digital computer systems can address or be focused on a number of major levels or design issues within the overall computer structure. These levels are:

1. **Materials** - the basic materials used in the construction of operating devices such as transistors, integrated circuits, or other switching devices.
2. **Devices** - operating devices such as transistors, integrated circuits, junctions, etc.
3. **Switching circuits** - design of circuits that provide fast and reliable logic operations.
4. **Register-transfer** - assemblies such as registers, buses, shift registers, adders, etc.
5. **System architecture** - algorithms for executing the basic functions such as arithmetic and logic operations, interrupt mechanisms, control of processor and memory states, etc.
6. **System organization** - the interconnection of major functional units such as control, memory, I/O, arithmetic/logic units, etc., and the rules governing the flow of data and control signals between these units. This level also considers the implementation of multiple, parallel paths for simultaneous operations and transfers.
7. **Network organization** - the number, characteristics, and topology of the interconnection of "complete" systems and the rules governing the control and utilization of the resources those systems provide.
8. **System software** - control and support software for the effective management and utilization of the hardware capabilities provided.

From the very beginning of the computer era there has been activity at all of these levels and such work continues today. (To place it into proper perspective, it should be noted that the research work carried on under this project is focused primarily at the three highest levels, system organization, network organization, and system software, with some work at level 5, system architecture.)

1.3 PARALLEL PROCESSING SYSTEMS

An important theme of computer system development work at levels 5-8, "system architecture," "system organization," "network organization," and "system software," has been parallel processing. Parallel processing has been implemented utilizing approaches focused primarily on the system hardware or the software as well as integrated systems design.

Since the early days of computing, a direction of research that has offered high promise and attracted much attention is "parallel computing." Work in this area dates from the late 1950's which saw the development of the PILOT system [Lein58] at the National Bureau of Standards. The PILOT system consisted of "three independently operating computers that could work in cooperation."[Ensl74] (From the information available, it appears that PILOT would be classified as a "loosely-coupled system" today.) It is interesting to note that the evolution of parallel "hardware" systems lead primarily to the development of tightly-coupled systems such as the Burroughs B-825 and B-5000, the earliest examples of the classical multiprocessor. Other development paths saw the introduction of specialized hardware systems such as SOLOMON and the ILLIAC IV, examples of other forms of tightly-coupled processors.

1.3.1 System Coupling

System coupling refers to the means by which two or more computer systems exchange information. It refers to both the physical transfer of such data as well as the manner in which the recipient of the data responds to its contents. These two aspects of system interconnection are called "physical coupling" and "logical coupling," and they are present in all multiple component systems whether the components of interest are complete computers or some smaller assembly.

The terms, "tight" and "loose" have been utilized to describe the mode of operation of each type of coupling. (Some authors have utilized a third category "medium coupling" and related it to a range of data transfer speeds; however, history has clearly shown that basing any characterizations of digital computers on speed, size, or even cost is an incorrect approach.) The interconnection and interaction of two computer systems can then be described by specifying the nature of its physical coupling and the nature of its logical coupling. It is important to point out that all four combinations of these characteristics are possible and that they all have been observed in implemented systems.

1.3.1.1 Tightly-Coupled Computer Systems

During the 1960's and 1970's, activities in the development of parallel computing, specifically multiple computer systems, were focused primarily on the development of tightly-coupled systems. These tightly-coupled systems

took the form of classical multiprocessors (i.e., shared main memory) as well as specialized computation systems such as vector and array processors. This tight physical coupling resulted in a sharing of the directly executable address space common to both processors. There was no means by which the recipient of the data or information being transferred could refuse to physically accept it --- it was already there in his address space.

These early systems also usually implemented tight logical coupling. In this form of system interaction, the recipient of a message is required to perform whatever service is specified therein. With tight logical coupling, there is no independence of decision allowed regarding the performance of the service or activity "requested." The relationship between the sender and recipient is basically that of master-slave.

Although the concept of tightly-coupled multiprocessor systems appears to be a viable approach for achieving almost unlimited improvements in performance (i.e., increases in system throughput) with the addition of more processors, such has not been the results obtained with implemented systems. It is the very nature of tight-coupling that results in limitations on the improvements achievable. Some of the ways that these limitations have manifested themselves are listed below.

1. The direct sharing of resources (memory and input/output primarily) often results in access conflicts and delays in obtaining use of the shared resource.
2. User programming languages that support the effective utilization of tightly-coupled systems have not been adequately developed. The programmer must still be directly involved in job and task partitioning and the assignment of resources.
3. The development of "optimal" schedules for the utilization of the processors is very difficult except in trivial or static situations. Also, the inability to maintain perfect synchronization between all processors often invalidates an "optimal" schedule soon after it has been prepared.
4. Any inefficiencies present in the operating system appear to be greatly exaggerated in a tightly-coupled system.

There was also significant activity during these earlier periods in the development of multiple computer systems characterized as "attached support processors (ASP)." These systems were physically loosely-coupled; but, logically, they were tightly-coupled. The earliest examples of this type of system organization were the use of attached processors dedicated to

input/output operations in large-scale batch processing systems. In the latter part of the 1970's, specialized vector and array processors as well as other special-purpose units such as fast Fourier transform units were being connected to general computational systems and utilized as attached support processors. In any event, the specialized nature of the services provided by the attached processor excludes them from consideration as possible approaches to providing general-purpose computational support such as that available from tightly-coupled general-purpose processors functioning as multiprocessors.

Tightly-coupled systems certainly do have a role to play in the total spectrum of computer systems organization; however, their limitations should certainly be considered. It was the recognition of these limitations and the small amount of progress made in overcoming them despite the expenditure of very large research efforts that contributed to the decision to focus our current research program on loosely-coupled systems.

1.3.1.2 Loosely-Coupled Systems

Loosely-coupled systems are multiple computer systems in which the individual processors both communicate physically and interact logically with one another at the "input/output level." There is no direct sharing of primary memory, although, there may be sharing of an on-line storage device such as a disk in the interconnecting input/output communication path. The important characteristic of this type of system organization and operation is that all data transfer operations between the two component systems are performed as input/output operations. The unit of data transferred is whatever is permissible on the particular input/output path being utilized; and, in order to complete a transfer, the active cooperation of both processors is required (i.e., one might execute a READ operation in order to accommodate or accept another's WRITE).

Probably the most important characteristic of loose logical coupling is that one processor does not have the capability or authority to "force" another processor to do something. One processor can "deliver" data to another; however, even if that data is a request (or a "demand") for a service to be performed, the receiving processor, theoretically, has the full and autonomous rights to refuse to execute that request. The reaction of processors to such requests for service is established by the operating system rules of the receiving processor, not by the transmitter. This allows the recipient

of a request to take into consideration "local" conditions in making the decision as to what actions to take. It is important to note that it is possible for a system to be physically loosely-coupled but logically tightly-coupled due to the rules embodied in the component operating systems, e.g., a permanent master/slave relationship is defined. The other reverse condition, tight physical and loose logical coupling, is also possible.

1.3.2 Computer Networks

A computer network can be characterized as a physically loosely-coupled, multiple-computer system in which the interconnection paths have been extended by the inclusion of data communications links. Fundamentally there are no differences between the basic characteristics of computer network systems and other loosely-coupled systems other than the data transfer rates normally provided. The transfer of data between two nodes in the network still requires the active cooperation of both parties involved, but there is no inherently required cooperation between the operation of the processors other than that which they wish to provide.

1.3.3 Distributed Systems

Although there is a large amount of confusion, and often controversy, over exactly what is a "distributed system," it is generally accepted that a distributed system is a multiple computer network designed with some unity of purpose in mind. The processors, databases, terminals, operating systems, and other hardware and software components included in the system have been interconnected for the accomplishment of an identifiable, common goal. That goal may be the supplying of general-purpose computing support, a collection of integrated applications such as corporate management, or embedded computer support such as a real-time process control system.

This research program is concerned with a very specific subclass of all of the systems currently being designated "distributed." The environment of interest here has been given the title "Fully Distributed Processing System" or FDPS. Section 2 discusses the general characteristics of FDPS's.

SECTION 2

INTRODUCTION TO FULLY DISTRIBUTED PROCESSING SYSTEMS

2.1 MOTIVATION OF THE FDPS CONCEPT

A large number of claims have been made as to the benefits that will be achieved with distributed processing systems. As pointed out above, this list is very similar to the lists of "benefits to be achieved" with several earlier computer technologies. However, each of those earlier solutions failed to deliver its promises for various reasons. It was an examination of the "weaknesses" in the earlier concepts and the development of a set of principles to overcome these obstacles that led to the concept of **"Fully Distributed Processing Systems"** or as it is commonly referred to **"FDPS."**

The principle of parallel (i.e., simultaneous and/or concurrent) operation of a multiplicity of resources continues to be perhaps the most important goal. The unique feature of FDPS's is the means or environment in which this is attempted. A distributed system should exhibit a continual increase in performance as additional processing components are added. The users should observe shorter response times as well as an increase in total system throughput. In addition, the utilization of system resources should be higher as a result of the system's ability to perform automatic load balancing servicing a large quantity and variety of user work requests. A distributed system should also permit the sharing of data between cooperating users and the making available of specialized resources found only on certain processors. In general, a distributed system should provide more facilities and a wider variety of services than those that can be offered by any system composed of a single processor [Hopp79]. Another important and highly desirable feature of such a system is extensibility. Extensibility might be realized in several different ways. The system might support modular and incremental growth permitting flexibility in its configuration, or it might support expansion in capacity, adding new functions, or both. Finally, it might provide for incremental replacement and/or upgrading of system components, either hardware or software. The executive control of the system is obviously the key to attaining these goals, and it is in the area of executive control that some of the most significant deficiencies of earlier systems have been found.

The major weaknesses in the executive control of earlier forms of parallel systems appear to result from an excessive degree of centralization of control functions reflected in centralized decision making or centralized maintenance of system status information or both of these. The net effect of these aspects of control was to produce a rather tightly-coupled environment in which resources often were idle waiting for work assignments and the failure of one major component often resulted in catastrophic and total system failure. The solution to this problem is to force a condition of very loose coupling on both the logical/control decision making process as well as the physical linkages of components. This property of "universal" loose coupling results in an environment in which the various components are required to operate in an autonomous manner.

If a single design principle must be identified as the most important or central theme of FDPS design, it is component autonomy or "cooperative autonomy" as described below. All of the other features of the definition of Fully Distributed Processing Systems given below have resulted from determining what is required to support and utilize the autonomous operation of the very loosely-coupled physical and logical resources.

2.2 THE DEFINITION OF AN FDPS

Fully Distributed Processing Systems (FDPS) were first defined by Enslow in 1976 [Ensl78] although the designation "fully" was not added until 1978 when it became necessary to clearly distinguish this class of distributed processing from the many others being presented. An FDPS is distinguished by the following characteristics:

1. **Multiplicity of resources:** an FDPS is composed of a multiplicity of general-purpose resources (e.g., hardware and software processors that can be freely assigned on a short-term basis to various system tasks as required; shared data bases, etc.).
2. **Component interconnection:** the active components in the FDPS are physically interconnected by a communications network(s) that utilizes two-party, cooperative protocols to control the physical transfer of data (i.e., loose physical coupling).
3. **Unity of control:** the executive control of an FDPS must define and support a unified set of policies (i.e., rules) governing the operation and utilization or control of all physical and logical resources.

4. System transparency: users must be able to request services by generic names not being aware of their physical location or even the fact that there may be multiple copies of the resources present. (System transparency is designed to aid rather than inhibit and, therefore, can be overridden. A user who is concerned about the performance of a particular application can provide system specific information in order to aid in the formulation of management control decisions.)

5. Component autonomy: both the logical and physical components of an FDPS should interact in a manner described as "cooperative autonomy" [Clar80, Ensl78]. This means that the components operate in an autonomous fashion requiring cooperation among processes for the exchange of information as well as for the provision of services. In a cooperatively autonomous control environment, the components are afforded the ability to refuse requests for service, whether they be execution of a process or the use of a file. This could result in anarchy except for the fact that all components adhere to a common set of system utilization and management policies expressed by the philosophy of the executive control.

2.2.1 Discussion of the Definitional Criteria

In order for a system to qualify as being fully distributed it must possess all five of the criteria presented in this definition.

2.2.1.1 Multiple Resources and Their Utilization

The requirement for resource multiplicity concerns the assignable resources that a system provides. Therefore, the type of resources requiring replication depends on the purpose of a system. For example, a distributed system designed to perform real-time computing for air traffic control requires a multiplicity of special-purpose air traffic control processors and display terminals. It is not required that replicated resources be exactly homogenous, however, they must be capable of providing the same services.

In addition to this multiplicity, it is also required that the system resources be dynamically reconfigurable to respond to a component failure(s). This reconfiguration must occur within a "short" period of time so as to maintain the functional capabilities of the overall system without affecting the operation of components not directly involved. Under normal operation the system must be able to dynamically assign its tasks to components distributed throughout the system.

The extent to which resources are replicated can vary from those systems where none are replicated (not a fully distributed system) to systems where all assignable resources are replicated. In addition, the number of copies of

a particular resource can vary depending on the system and type of resource. In general, the greater the degree of replication, particularly of resources in high demand, the greater the potential for attaining benefits such as increased performance (response time and throughput), availability, reliability, and flexibility [Ensl78].

2.2.1.2 Component Interconnection and Communication

The extent of physical distribution of resources in distributed systems can vary from the length of connection between components on a single integrated chip to the distance between two computers connected through an international network. In addition, interconnection organizations can vary from a single bus to a complex mesh network. Since a component in a distributed system communicates with other components through its own logical process, all physical and logical resources can be thought of as processes, and interactions between resources can be referred to as interprocess communication [Davi79]. For example, an application program interacting with processors and data files is accomplished through communication between logical processes.

Both the physical and logical coupling of the system components are characterized as "extremely loose." "Gated" or "master-slave" control of physical transfer is not allowed. Communication, i.e., the physical transfer of messages, is accomplished by the active cooperation of both the sender and addressees. The primary requirement of the intercommunication subsystem is that it support a two-party cooperative protocol. This is essential to enable the system's resources to exist in cooperative autonomy at the physical level.

The advantages of using a message-based (loosely-coupled) communication system with a two-party cooperative protocol include reliability, availability, and extensibility. The disadvantage is the additional overhead of message processing incurred to support this method of communication. There are a variety of interconnection organizations and communication techniques that can be used to support a message-based system with a two-party cooperative protocol.

2.2.1.3 Unity of Control

In a fully distributed data processing system, individual processors will each have their own local operating systems, which may or may not be unique, that control local resources. As a result, control is distributed

throughout the system to components that operate autonomously of one another. However, to gain the benefits of distributed processing it is required that the autonomous components of the system cooperate with each other to achieve the overall objectives of the system. To insure this, the concept of a high-level operating system was created to integrate and unify, at least conceptually, the decentralized control of the system.

A high-level operating system is essential to successfully implementing a distributed processing system. This operating system is not a centralized block of code with strong hierarchical control over the system, but rather it is a well-defined set of policies governing the integrated operation of the system as a whole. To insure reliable and flexible operation of the system, these policies should be implemented with minimal binding to any of the system's components [Ensl78].

What policies are required and how they should be implemented depends greatly on the system. For example, if it is a general-purpose system supporting interactive users, then a command interpreter and a user control language will be required to make the system's components compatible and transparent to the user.

2.2.1.4 Transparency of System Control

The high-level operating system also provides the user with his interface to the distributed system. As a result, the user is accessing the system as a whole rather than just a host computer in the network.

In order to increase the effectiveness of the distributed system, the actual system is made transparent, and the user is presented with a virtual machine and a simplified command language to access it. The user uses this language to request services by name and does not have to specify the specific server to be used. Clearly, the same request might be assigned a different server depending on the state of the total system when the request is made. However, to make the system truly effective for all users, knowledgeable individuals must be able to interact with the system more intimately, requesting specific servers or developing service routines to increase the efficiency or effectiveness of the system [Ensl78].

2.2.1.5 Cooperative Autonomy

Cooperative autonomy has already been described at the physical interconnection level. It is also required that all resources be autonomous at the

logical control level. That is, a resource must have full control of itself in determining which requests it will service and what future operations it will perform. However, a resource must also cooperate with other resources by operating according to the policies of the high-level operating system. Cooperative autonomy is an essential prerequisite for systems to have fault tolerance and high degrees of extensibility [Ensl78]. It is perhaps the most important as well as the most distinguishing characteristic of a fully distributed processing system.

2.2.2 Effects on System Organization

Although the detailed design of the hardware and software required to implement an FDPS is still in progress, it has been possible for some time to identify certain characteristics that these components must have. One area in which certain criteria already appear reasonably well defined is the nature of the organization of the following system components:

- Hardware
- System control software
- Data bases

It should be noted that a number of definitions and descriptions of distributed systems in general are based on the principle that one or more of these components is physically distributed. (Some such discussions add to this list a fourth component --- "processing or function;" however, considering the distribution of processing independent from the distribution hardware is quite improper. Why distribute the hardware if it will not have some function to perform; similarly, how can the processing be distributed without a corresponding distribution of the hardware? That would be processing on a truly "virtual machine.")

An important characteristic of an FDPS is that, in order to meet the definitional criteria given above while also attempting to provide as many as possible of the benefits listed in Table 1, all of the three components listed above must be physically distributed and the degree of distribution must in each case exceed a reasonably well-defined threshold. A diagram illustrating this requirement is shown in Figure 1. The various organizations of each component identified and positioned along each axis is not meant to be an exhaustive list. These points are listed to better identify the relative location of the three thresholds defining the volume of space occupied by FDPS's. (It might also be noted that it seems quite proper to characterize

any system that is not in the "origin cube" as being "distributed" to some degree.)

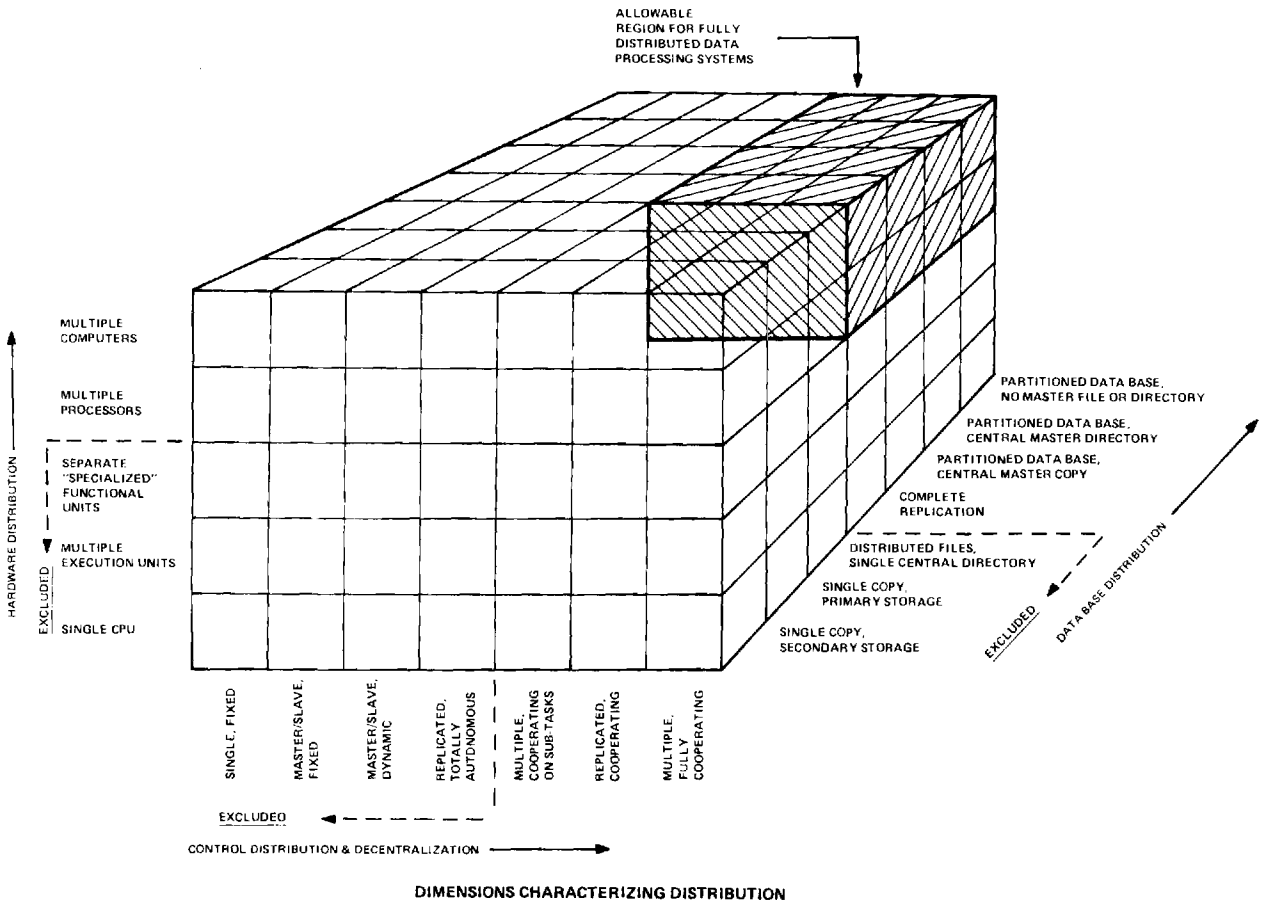


Figure 1. Axes of Distribution

2.3 IMPLICATIONS OF THE FDPS DEFINITION ON CONTROL

2.3.1 General Nature of FDPS Executive Control

Several of the characteristics of an FDPS are found to directly impact the design and implementation of the executive control for such a system. These include system transparency to the user, extremely loose physical and logical coupling, and cooperative autonomy as the basic mode of component interaction. System transparency means that the FDPS appears to a user as a large uniprocessor which has available a variety of services. It must be possible for the user to obtain these services by naming them without specifying any information concerning the details of their physical location. The result is that system control is left with the task of locating all appropriate instances (copies) of a particular resource and choosing the instance to be utilized.

"Cooperative autonomy" is another characteristic of an FDPS heavily impacting its executive control. The "lower-level" control functions of both the logical and physical resource components of an FDPS are designed to operate in a "cooperatively autonomous" fashion. Thus, an executive control must be designed such that any resource is able to refuse a request even though it may have physically accepted the message containing that request. Degeneration into total anarchy is prevented by the establishment of a common set of criteria to be followed by all resources in determining whether a request is accepted and serviced as originally presented, accepted only after bidding/negotiation, or rejected.

Another important FDPS characteristic that definitly affects the design of its executive control is the extremely loose coupling of both physical and logical resources. The components of an FDPS are connected by communication paths of relatively low bandwidth. The direct sharing of primary memory between processors is not acceptable. Even though the logical coupling could still be loose with this physical interconnection mechanism, the presence of a single critical hardware element, the shared memory would create fault-tolerance limitations. All communication takes place over "standard" input/output paths. The actual data rates that can be supported are primarily a function of the distance between processors and the design of their input/output paths. In any event, the transfer rates possible will probably be much less than memory transfer rates. This implies that the sharing of

information among components on different processors is greatly curtailed, and system control is forced to work with information that is usually out-of-date and, as a result, inaccurate.

The control of an FDPS requires the action and cooperation of components at all layers of the system. This means that there are elements of FDPS control present in the lowest levels of the hardware as well as software components. This paper is primarily interested in the software components of the FDPS control which are typically referred to as "the executive control."

The executive control is responsible for managing the physical and logical resources of a system. It accepts user requests and obtains and schedules the resources necessary to satisfy a user's needs. As mentioned earlier, these tasks are accomplished so as to unify the distributed components of the system into a whole and provide system transparency to the user.

2.3.2 Why Not Centralized Control?

Why then is a centralized method of control not appropriate? In systems utilizing a centralized executive control, all of the control processes share a single coherent and deterministic view of the entire system state. An FDPS, though, contains only loosely-coupled components, and the communication among these components is restricted and subject to variable time delays. This means that one cannot guarantee that all processes will have the same view of the system state [Jens78]. In fact, it is an important characteristic of an FDPS that they will not have a consistent view.

A centralized executive control weakens the fault-tolerance of the overall system due to the existence of a single critical element, the executive control itself. This obstacle, though, is not insurmountable for strategies do exist for providing fault-tolerance in centralized applications. Garcia-Molina [Garc79], for example, has described a scheme for providing fault-tolerance in a distributed data base management system with a centralized control. Approaches of this type typically assume that failures are extremely rare events and that the system can tolerate the dedication of a relatively long interval of time to reconfiguration. These restrictions are usually unacceptable in an FDPS environment where it is important to provide fault-tolerance with a minimum of disruption to the services being supported.

Also, the extremely important issue of overall system performance must be considered. A distributed processing system is expected to utilize a large quantity and a wide variety of resources. If a completely centralized executive control is implemented, there is a high probability that a bottleneck will be created in the node executing the control functions. A distributed and decentralized approach to control attempts to remove this bottleneck by dispersing the control decisions among multiple components on different nodes.

2.3.3 Distributed vs. Decentralized

This paper advocates utilizing an approach for the control of an FDPS that is both distributed and decentralized. There is a clear distinction between the terms "distributed" and "decentralized" as they are used in the context of this project. "Distributed control" is characterized by having its executing components physically located on different nodes. This means there are multiple loci of control activity. In "decentralized control," on the other hand, control decisions are made independently by separate components at different locations. In other words, there are multiple loci of control decision making. Thus, distributed and decentralized control has active components located on different nodes and those components are capable of making independent control decisions.

2.4 AN FDPS APPLICATION --- DATA FLOW PROCESSING

The operating characteristics specified for an FDPS appear to be especially suited to applications composed of cooperating processes that may be executed simultaneously. One class of such applications have been referred to as data flow networks [Denn78, Nels78]. They utilize the independence of the processors combined with the implicit potential for parallel operation of data flow networks to improve performance. In addition to potentially improving performance, the data flow approach often provides a more natural method for expressing a solution to a particular problem. Other systems, including ADAPT [Peeb80], Medusa [Oust80], and TRIX [Ward80], have been designed to service similar types of applications. An application of this type can be expressed either as a command level program [Akin78] or a program in a high level language [Feld79, Macc80]. The execution of individual processes may result from the invocation of files containing either executable code or commands. In such a system, calls to other processes (executable files or com-

mand files) can originate from any process, and the nesting of such calls is unlimited.

2.5 PROJECT SCOPE AND ORGANIZATION OF THIS REPORT

Following these two sections of introductory comments, this report discusses the results of an initial study of distributed and decentralized control including, where appropriate, material concerning the results of other projects in the Georgia Tech Research Program on Fully Distributed Processing Systems (FDPS). This initial study of FDPS control has been focused primarily on the qualitative aspects of various forms and implementations of control. The project description is as follows:

> "Define and refine existing models of distributed and decentralized control and develop new models as appropriate to provide a capability of fault tolerance, automatic reconfiguration, and dynamic control."

It is important to note that very few "existing models of distributed control" have been identified and those that have been located are so incompletely defined that this project has proceeded primarily by defining candidate models while attempting to develop a suitable taxonomy of other possible models. Since this project was undertaken fully cognizant that a quantitative study of the models would follow immediately, it is felt that the development of such a taxonomy will help to insure that no significant variations are overlooked.

2.5.1 Discussion of FDPS Models

Along with the development of the various models for distributed and decentralized control, the FDPS team is also developing total system models. These system models provide an essential part of the description of the total environment within which the executive control must operate. Although it is clear at this time that these system models are still evolving, descriptions of their present versions are presented in Section 3.

2.5.2 Issues in Decentralized Control

Although most readers probably have some understanding of the functions of the executive control in a centralized system, the overall effects of the distributed environment and the set of totally new requirements placed on a decentralized executive control are perhaps not so obvious. The purpose of Section 4 is to discuss the effects of the operating environment and to

explicitly identify as many as possible of the new control requirements and limitations as well as variations from centralized control models.

2.5.3 Work Requests

There is a strong relationship between the forms of work requests that the distributed system is expected to process and the capabilities required in the control model. Section 5 focuses on the variations possible in the work requests leaving the discussion of the resulting effects on the operation of the executive control until Section 7.

2.5.4 Characteristics of a Decentralized Control Model

Section 6 of this report presents and discusses those attributes that distinguish various models in the present catalog of decentralized control models. (Note that this is not presented as a complete "taxonomy.") The attributes are characterized in terms of the information that needs to be maintained and the decisions that must be made by an executive control. Also discussed in this section are some of the operational aspects of the models identified thus far.

2.5.5 Control Model Functions

It is during a detailed discussion of the functions performed by an executive control that many of the aspects of decentralized control are best highlighted. In Section 7 discussion of the individual operations are presented and then representative examples of functions such as task graph building are discussed. (A task graph is used to maintain information about the processes being utilized to satisfy a work request. See Paragraph 7.1 for a more complete definition of task graphs.) Experience has shown that many individuals do not fully grasp the significance of distributed and decentralized control until they study examples such as those presented in Section 7.

2.5.6 Example Control Models

A few specific control models that have been developed thus far are presented in Section 8. These include control models advanced by other research teams as well as several developed in the FDPS research program.

2.5.7 Control Model Evaluation

Immediately following this survey of control models the various models will be evaluated. Section 9 presents a preliminary discussion of some of the evaluation criteria to be applied.

SECTION 3

FDPS SYSTEM MODELS

3.1 INTRODUCTION

Models serve extremely important, if not essential, roles in the development of complex systems. This is especially true for systems in which the effects of complexity are further complicated by inconsistencies, ambiguities, and incompleteness in the use of the terms that are employed to describe the structure as well as the operation of the systems involved and the components thereof. Suitable models are valuable, if not essential, tools to support and clarify such discussions. When examining or using any model, it is equally important to recognize that it may have been prepared or developed for a specific purpose (e.g., logical or physical description, simulator design, implementation guide, etc.) and may not be totally suitable for other uses.

3.1.1 Why a "New" Model and "New" Terminology?

Since the concepts of "full distribution" were first conceived over four years ago, members of the FDPS project have been plagued by severe problems in explaining the significance of various aspects of the definition of an FDPS. Most of these problems have been caused by the difficulties in clearly communicating the extremely important differences between "fully" distributed systems and those that are merely "distributed." These problems in understanding appear often to result from the "listener" incorrectly equating certain aspects of FDPS operation with those of a similarly appearing distributed system. Such misunderstandings are not totally unreasonable, for some of the most significant differences are quite subtle. One highly desirable effect anticipated from "new" system models and "new" terminology is to prevent, or at least make less likely, these undesirable associations with existing system concepts.

3.1.2 Approaches to Modelling

There are a number of approaches that may be followed in the development of a system model. The selection of the approach to be taken is based on the intended use of the model and the nature of the system being modelled.

3.1.2.1 Scenario or Flow Chart Models

Certainly one of the most commonly encountered models is the simple flow chart. A flow chart depicts the thread or threads of processing that the system will perform in response to a given set of inputs. A flow chart is probably the best method to illustrate or model the sequence of processing activities involved in a transaction processing or similar type system.

3.1.2.2 Structure Models

Logical and physical structure models are focused more on the organization and modularization of the processing software and hardware than on the actual processing those modules perform. Perhaps the most important use of structure models is in the partitioning of functionality and code for implementation.

3.1.2.3 Interaction Models

Interaction models which focus on the relationships between software and hardware processing entities are becoming quite popular in the area of computer networks; however, they are certainly not limited to just those applications. The basic principle employed in the development of these models is layering with interactions between pairs of peer layers and sets of adjacent layers being specified. The operation and functionality provided by each layer is defined in terms of its protocols and interfaces.

The rules and procedures defining the interactions between peer layers are known as "protocols," whereas "interfaces" define the boundaries and procedures for interaction between adjacent layers. (See Figure 2) (This usage of the term "interface" is consistent with its definition as the boundary between dissimilar entities.) To complete the system description at this level of abstraction, the interfaces are defined in terms of the services provided by a lower layer and the services provided to a higher layer.

It should be noted that in the area of computer networking, the combination of a complete set of protocols and a complete set of interfaces is referred to as a "network architecture."

Preparing a layered model with defined interfaces and protocols is no guarantee that a "clean" layering structure will result. A classic example of this is the ARPANET layers of protocol shown in Figure 3. Although they all make use of the Host-to-IMP protocol, there are many instances in ARPANET in which layers are bypassed completely.

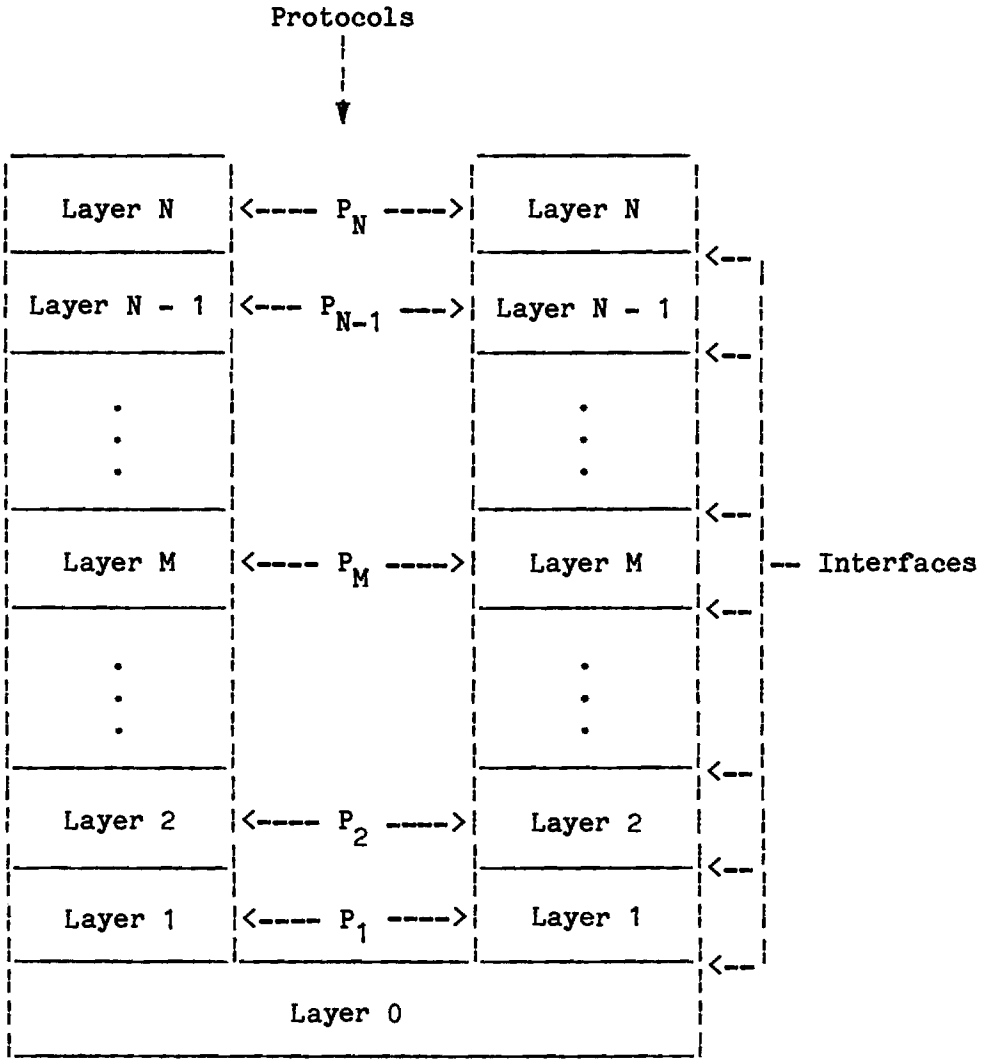


Figure 2. Protocols and Interfaces

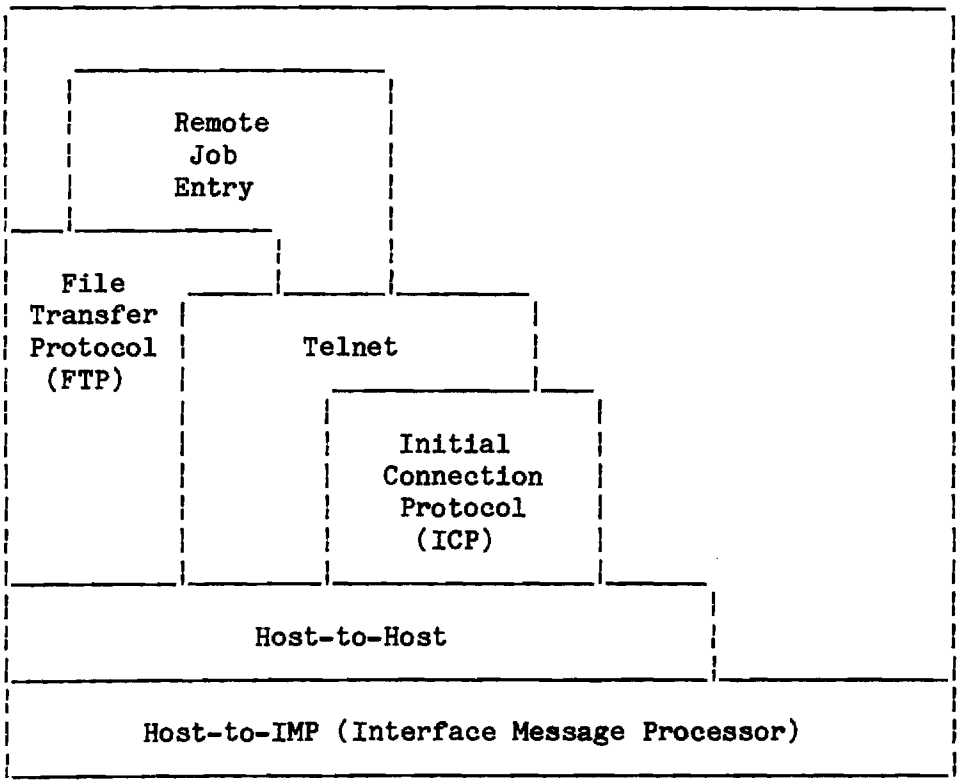


Figure 3. The ARPANET Protocol Layers

3.1.2.4 Performance and Mathematical Models

Obviously, the objective or purpose of this class of models is to provide tools to examine, and usually quantify, the performance of a system.

3.1.2.5 Summary of Model Types

The various types of models discussed above do not represent different ways to accomplish the same task. Although there is some common information found in or derivable from two or more of the various type of models, each is actually focused on quite different aspects of the system description.

- Physical structure model: Depicts the manner in which the various hardware and software components are partitioned and packaged.
- Logical structure model: Focuses on the functionality provided by the hardware and software components and how they may be logically organized into modules.
- Scenario or flow chart model: Depicts the sequence of processing actions taken on the data.
- Interaction model: Focuses on the interactions between processing entities --- services provided to or received from adjacent layer entities and the protocols governing the communication and negotiations that can occur between corresponding peer layers.
- Analytic model: Focuses on the performance of complete systems or subsystems. Often the external performance characteristics of the system being modelled are available.
- Simulation model: Depicts a system or subsystem by modelling as close as possible the operations that it performs. Provides more internal detail than an analytic model.

3.2 OTHER MODELS

Although the work on FDPS models has certainly been strongly influenced by the numerous existing "models" of multiprocessors, multiple computer systems, and computer networks, there has been very little influence from other "distributed system" models since few of these have been developed to the point that they can be closely analyzed. One model that has had a great deal of influence on the development of the FDPS models, at least in guiding the manner in which those models are presented, is the "Reference Model for Open System Interconnection" developed by Sub-Committee 16 of the International Standards Organization Technical Committee 97.

3.2.1 The ISO Reference Model for OSI

The ISO Reference Model, a layered-interaction model, is being prepared by Sub-Committee 16 to establish a framework for the development of standard

protocols and interfaces as appropriate for the interconnection of heterogeneous nodes in an "open" computer network and the intercommunication between the processes in these nodes. (This model is almost totally focused on the IPC process, i.e., interprocess communication.) The ISO model is a 7-layer structure as shown in Figure 4.

Although the ISO Reference model has been influential in providing ideas and concepts applicable to a layered model of an FDPS, there are two major factors limiting its direct applicability:

1. The ISO model is almost totally concerned with communication between the nodes of a network. Some references are made to higher level protocols in the applications layer, but these are not a part of the ISO model.
2. Although it is not explicitly stated, there appears to be a general assumption in the ISO model of a degree of coupling that is tighter than that anticipated for an FDPS. (This comment also applies to nearly all of the current network architectures --- even those that include application layer protocols.)

3.2.2 Protocol Hierarchies

As stated above, the ISO Reference Model addresses only a subset of the protocols and interfaces that will be found in a complete distributed system. A more complete picture is shown in Figure 5.

3.3 THE FDPS MODELS

3.3.1 The FDPS Logical Model

The current version of the FDPS logical model is organized into five layers above the "physical interconnection" layer. (Figure 6) The important or significant characteristics of this logical model are:

1. It is also a rudimentary layered-interaction model; however, to be useful, the interaction model must eventually delineate more layers.
2. The operating system has been divided into two parts based on a division of functionality and responsibilities:
    a. The Local Operating System (LOS) is responsible for the detailed control and management of the users and resources at a single node.
    b. The Network Operating System (NOS) is responsible for interactions between this node and all others.

3. The correlation of FDPS layers and ISO layers is the following:

FDPS Layers	ISO Layers
Users and Resources Local Operating System Network Operating System	Application
Message Handler	Presentation Session
Message Transporter	Transport Network Data Link Physical

3.3.2 An FDPS Physical Model

One of the possible physical models for an FDPS operating system is shown in Figure 7. This is a good example of how logical models and physical models may differ in their modularization. In Figure 7, the division between the LOS and NOS layers of the logical model runs horizontal through the MANAGERS in the physical model.

3.3.3 The FDPS Interaction Model

All of the individual layers of the FDPS interaction model have not yet been identified; however, a more detailed list of the protocols that may be loosely related to Figure 5 is given in Figure 8. This list of protocols is especially significant to the FDPS research project since it identifies those specific areas in which work must be done.

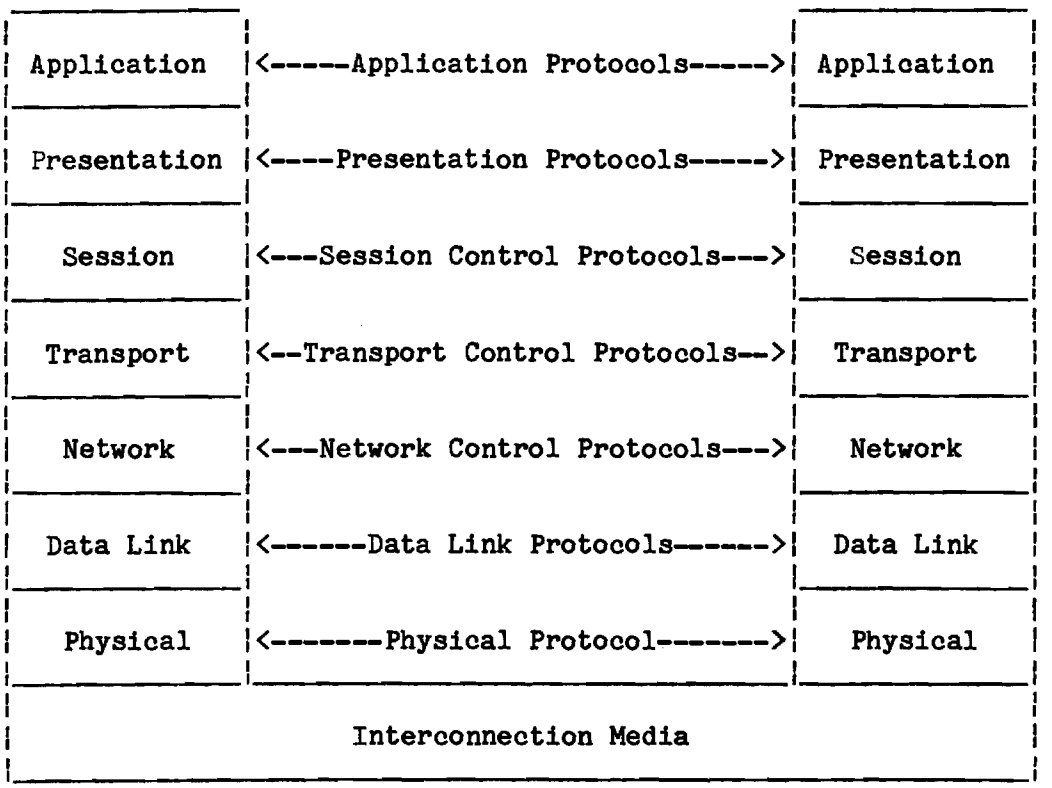


Figure 4. The ISO Reference Model for OSI

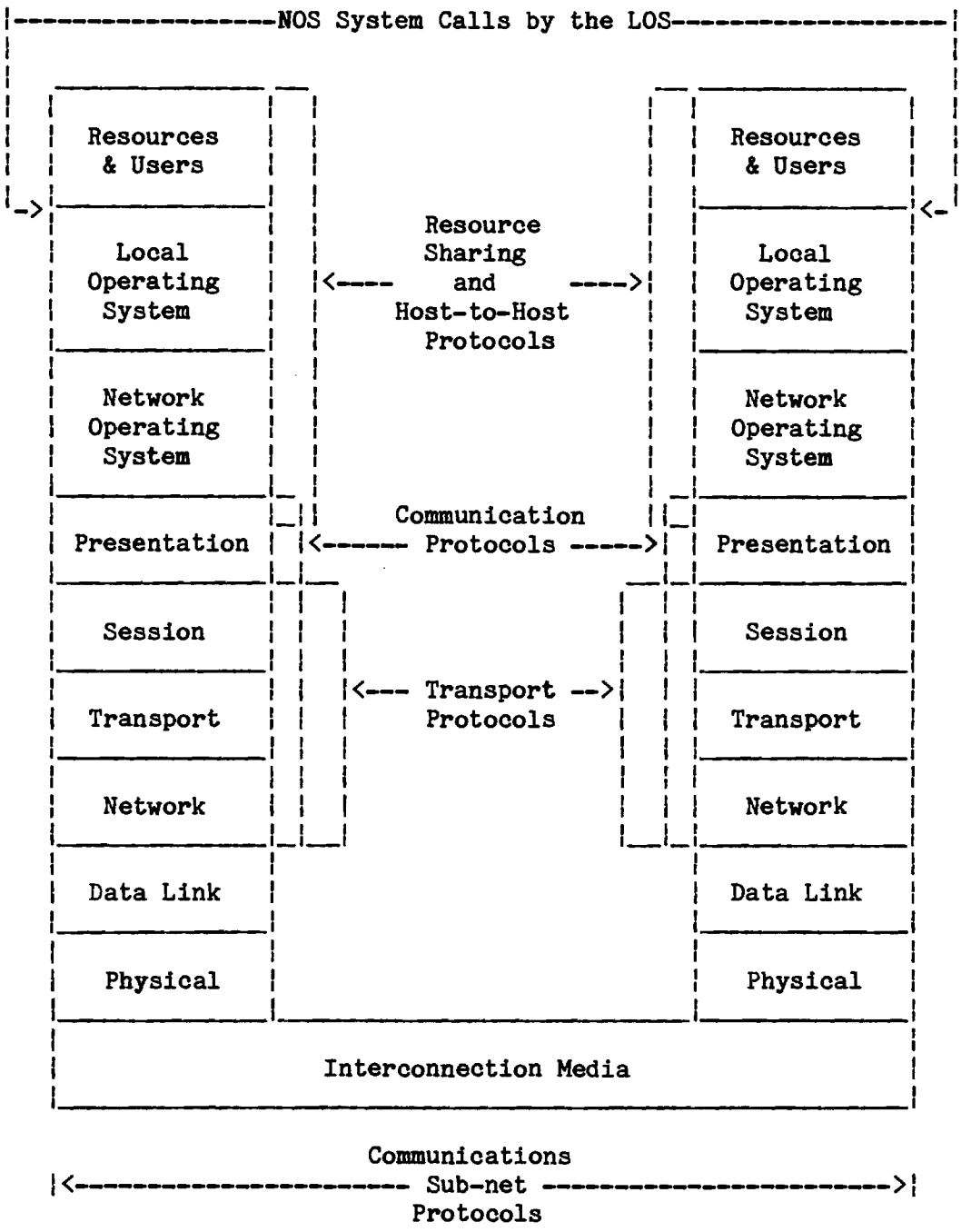


Figure 5. A "Complete" Protocol Hierarchy

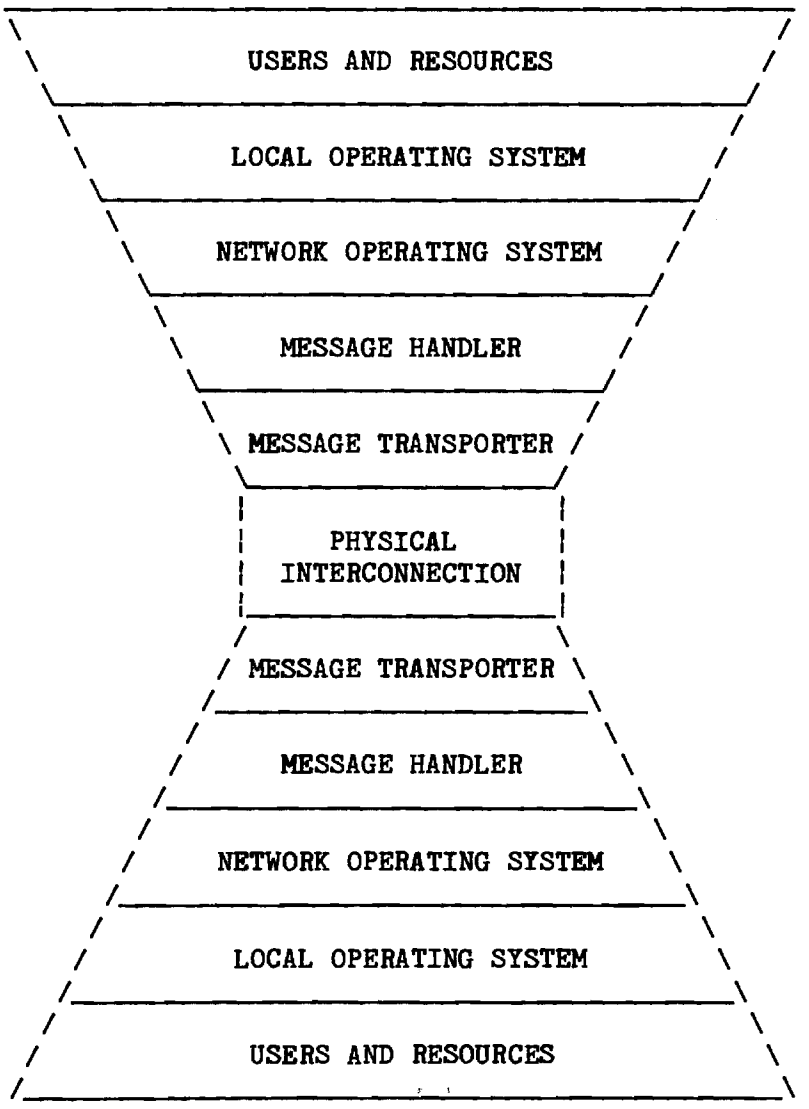


Figure 6. A Logical Model of an FDPS

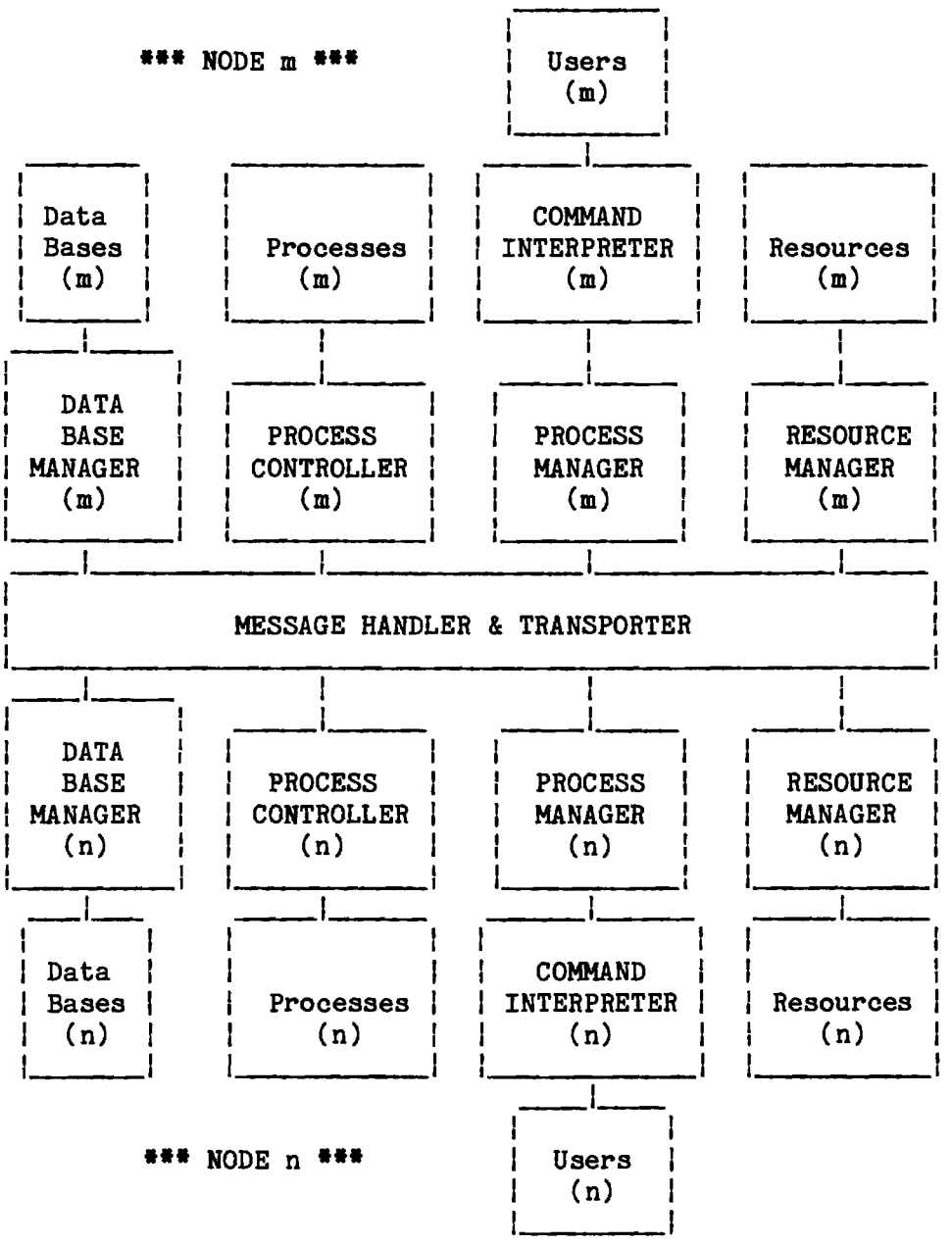
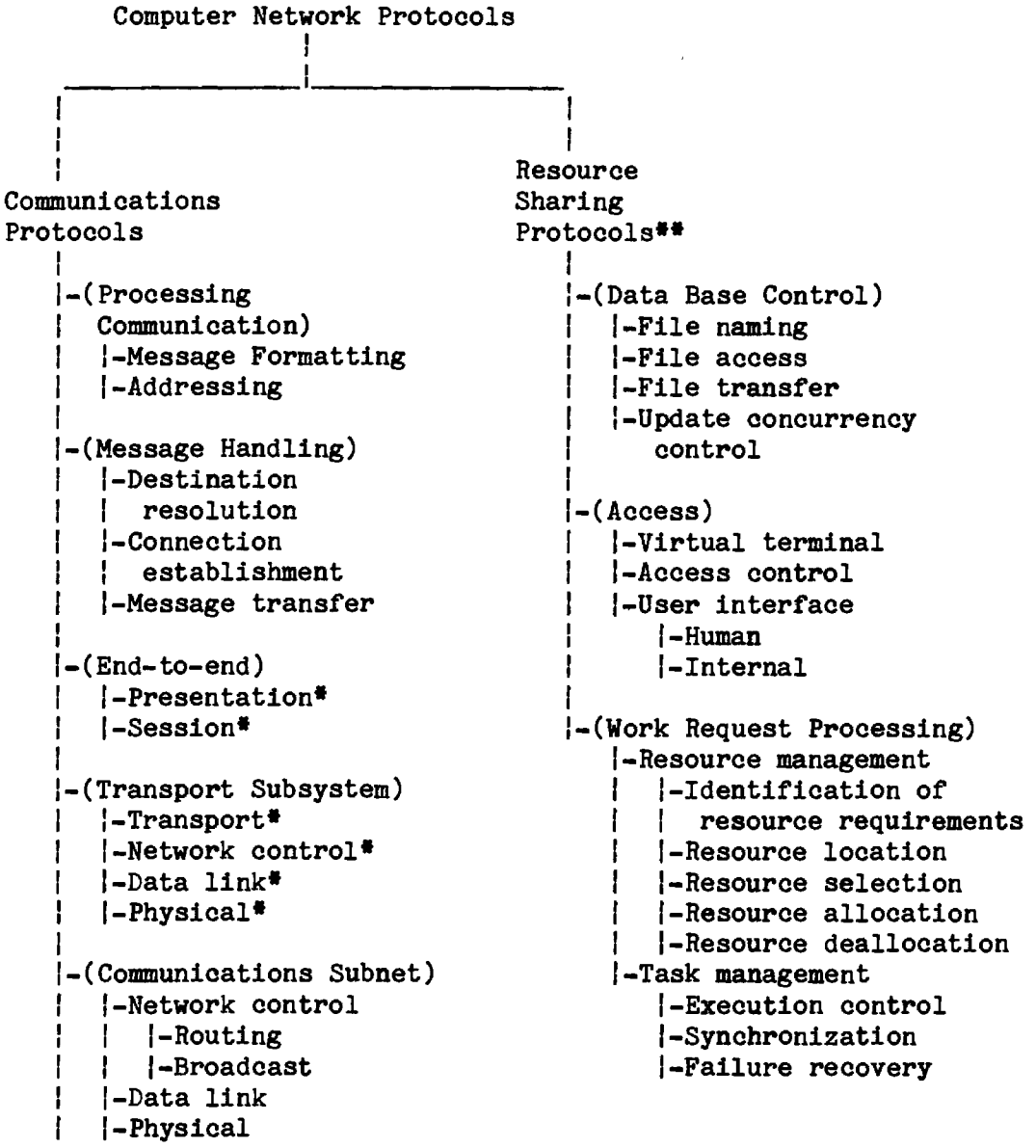


Figure 7. Physical Model of FDPS Control



* Classifications (layers) defined by the ISO and CCITT Network Architecture Models

** A preliminary list for FDPS's

Figure 8. Classifications of Computer Network Protocols

SECTION 4

ISSUES IN DISTRIBUTED CONTROL

Before examining specific aspects of executive control in an FDPS, a look at some of the various issues of distributed control is appropriate. There are three primary issues that require examination: 1) the effect of the dynamics of FDPS operation on an executive control, 2) the nature of the information an executive control must maintain, and 3) the principles to be utilized in the design of an executive control.

4.1 DYNAMICS

Dynamics is an inherent characteristic of the operation of an FDPS. Dynamics are found in the work load presented to the system, the availability of resources, and the individual work requests submitted. The dynamic nature of each of these provides the FDPS executive control with many unique problems.

4.1.1 Workload Presented to the System

In an FDPS, work requests can be generated either by users or active processes and can originate at any node. Such work requests potentially can require the use of resources on any processor. Thus, the collection of executive control procedures must be able to respond to requests arriving at a variety of locations from a variety of sources. Each request may require system resources located on one or more nodes, not necessarily including the originating node. One of the goals of an FDPS executive control is to respond to these requests in a manner such that the load on the entire system is balanced.

4.1.2 Availability of Resources

Another dynamic aspect of the FDPS environment concerns the availability of resources within the system. As mentioned above, a request for a service to be provided by a system resource may originate at any location in the system. In addition, there may be multiple copies of a resource or possibly multiple resources that provide the same functionality (e.g., there may be functionally equivalent FORTRAN compilers available on several different nodes). Since resources are not immune to failures, the possibility of losing existing resources or gaining both new and old resources exists. Therefore, an FDPS executive control must be able to manage system resources in a dynamic

environment in which the availability of a resource is unpredictable.

4.1.3 Individual Work Requests

Finally, the dynamic nature of the individual work requests must be considered. As mentioned above, these work requests define, either directly or indirectly, a set of cooperating processes which are to be invoked. An indirect definition of the work to be done occurs when the work request is itself the name of a command file or contains the name of a command file in addition to names of executable files or directly executable statements. A command file contains a collection of work requests formulated in command language statements (see Figure 10 for a description of the syntax for a suitable command language) that are interpreted and executed when the command file is invoked. The concept of a command file is similar to that of a procedure file which is available on several current systems.

Management of the processes for a work request thus includes the possibility that one or more of the processes are command files requiring command interpretation. The presence of command files will also result in the inclusion of additional information in the task graph or possibly additional task graphs. (See paragraph 7.5 for a discussion of the impact of command files on the task graph.)

An important objective of work request management is to control the set of processes and do so in such a manner that the inherent parallelism present in the operations to be performed is exploited to the maximum. In addition, situations in which one or more of the processes fail must also be handled.

4.2 INFORMATION

All types of executive control systems require information in order to function and perform their mission. The characteristics of the information available to the executive control is one aspect of fully distributed systems that result in the somewhat unique control problems that follow:

1. Because of the nature of the interconnection links and the delays inherent in any communication process, system information on hand is always out of date.
2. Because of the autonomous nature of operation of all components, each processor can make "its own decision" as how to reply to an inquiry; therefore, there is always the possibility that information received is incomplete and/or inaccurate.
3. Because of the inherent time delays experienced in exchanging

information among processes on different nodes, some information held by two processes may conflict during a particular time interval.

4.3 DESIGN PRINCIPLES

Designing the system control functions required for the extremely loosely-coupled environment of an FDPS and implementing those functions to operate in that environment will certainly require the application of some new design principles in addition to those commonly utilized in operating systems for centralized systems. These design principles must address at least the two distinguishing characteristics of FDPS's:

- System information available, and
- Nature of resource control

4.3.1 System Information

The various functions of an FDPS executive control must be designed recognizing that system information is:

- "Expensive" to obtain
- Never fully up-to-date
- Usually incomplete
- Often inaccurate

All of these characteristics of system information result from the fact that the components providing the information are interconnected by relatively narrow bandwidth communication paths (see paragraph 2.3.1) and that those components are operating somewhat autonomously with the possibility that their state may change immediately after a status report has been tansmitted. Further, it is important to note that the mere existence (or disappearance) of a resource is not of interest to a specific component of the FDPS executive control until that component needs that information.

The design principles applying to system information that have been identified thus far include the following:

1. Economy of communication: ask for only the information required.
2. Resiliency: be prepared to recover and continue in the absence of replies.
3. Flexibility: be prepared to recover and continue if the information provided proves to be inaccurate when it is utilized.

4.3.2 Resource Control

Since all of the resources are operating under local control under the policies of cooperative autonomy, all requests for service, or the utilization of any resource such as a file, must be effected through negotiations that culminate in positive acknowledgements by the server. In all instances, the control function requesting a service or a resource must be prepared for refusal.

SECTION 5

CHARACTERIZATION OF FDPS WORK REQUESTS

5.1 THE WORK REQUEST

One of the goals of an FDPS is the ability to provide a hospitable environment for solving problems that allows the user to utilize the natural distribution of data to obtain a solution which may take the form of an algorithm consisting of concurrent processes. The expression of the solution is in terms of a work request that describes a series of cooperating processes, the connectivity of these processes (how the processes communicate), and the data files utilized by these processes. This description involves only logical entities and does not contain any node specific information. A description of one command language capable of expressing requests for work in this fashion can be found in [Akin78] (see Figure 10).

5.2 IMPACT OF THE WORK REQUEST ON THE CONTROL

The nature of allowable work requests (not just the syntax but what can actually be accomplished via the work request) determines to a large extent the functionality of an executive control. Therefore, it is important to examine the characteristics of work requests and further to see how variations in these characteristics impact the strategies utilized by an FDPS executive control.

Five basic characteristics of work requests have been identified:

1. the external visibility of references to resources required by the task,
2. the presence of any interprocess communication (IPC) specifications,
3. the number of concurrent processes,
4. the nature of the connectivity of processes, and
5. the presence of command files.

5.2.1 Visibility of References to Resources

References to the resources required to satisfy a work request may either be visible prior to the execution of a process associated with the work request or embedded in such a manner that some part of the work request must be executed to reveal the reference to a particular resource. A resource is made "visible" either by the explicit statement of the reference in the work

request or through a declaration associated with one of the resources referenced in the work request. An example of the latter means of visibility is a file system in which external references made from a particular file are identified and stored in the "header" portion of the file. In this case, the identity of a reference can be obtained by simply accessing the header.

The greatest impact of the visibility characteristic of resource requirements occurs in the construction of task graphs and the distribution of work. The time at which resource requirements are detected and resolved determines when and how parts of the task graph can be constructed. Similarly, some work cannot be distributed until certain details are resolved. For example, consider a case where resource references cannot be resolved until execution time. Assume there exists two processes X and Y where process X has a hidden reference to process Y. An executive control cannot consider Y in the work distribution decision that is made in order to begin execution of X. The significance of this is that certain work distribution decisions may not be "globally optimal" because total information was not available at the time the decision was made.

5.2.2 The Number of Concurrent Processes

A work request can either specify the need to execute only a single process or the execution of multiple processes which may possibly be executed concurrently. Obviously with multiple processes, more resource availability information must be maintained; and there is a corresponding increase in the data to the work distribution and work allocation phases of control. In addition, the complexity of the work distribution decision algorithm increases with more resources needing to be allocated and multiple processes needing scheduling. The complexity of controlling the execution of the work request is also increased with the presence of multiple processes since the control must monitor multiple processes for each work request.

5.2.3 The Presence of Interprocess Communication

The problems described in the previous paragraph are amplified by the presence of communication connections between processes. When interprocess communication is described in a work request, the work distribution decision must consider the requirement for communication links. In addition, a compromise must be made in order to satisfy the conflicting goals of maximizing the inherent parallelism of the processes of the work request and minimizing

the cost of communication among these processes. The control activity required during execution is also impacted by the presence of interprocess communication. It must provide the means for passing messages, buffering messages, and providing synchronization to insure that a reader does not underflow and a writer does not overflow the message buffers.

5.2.4 The Nature of Process Connectivity

There are a variety of techniques available for expressing interprocess communication including pipes (see [Ritc78]) and ports (see [Balz71, Have78, Suns77, Zuck77]). There are a number of approaches to realizing these different forms of interprocess communication. The main impact on an executive control, though, is in those components controlling process execution.

5.2.5 The Presence of Command Files

A command file is composed of work requests. Execution of a work request that references a command file results in a new issue dealing with the construction of task graphs. This issue is concerned with whether a new task graph should be constructed to describe the new work request or should these new processes be included in the old task graph. The differences between these two approaches becomes important during work distribution. It is assumed that the work distribution decision will be made only with the information available in the task graph. Thus, with the first approach, only those tasks in the new work request are considered while the second approach provides the ability to take into consideration the assignment of tasks from previous work requests.

5.3 A CLASSIFICATION OF WORK REQUESTS

This examination of the characteristics of FDPS work requests has lead to the identification of five basic attributes which have significant impact on an executive control. In Figure 9, all possible types of work requests are enumerated resulting in 32 different forms of work requests. It should be noted, though, that 16 of these (those with an asterisk beside the task number) contain conflicting characteristics and thus are impossible.

No.	Resource References		IPC		Resources Distributed on Different Nodes		Multiple Copies Resources		Some Resources on Node Other Than Home Node	
	All Visible	Some Embedded	YES	NO	YES	NO	YES	NO	YES	NO
1		X		X		X		X		X
2		X		X		X		X	X	
3*		X		X		X	X			X
4*		X		X		X	X		X	
5*		X		X	X			X		X
6		X		X	X			X	X	
7*		X		X	X		X			X
8		X		X	X		X		X	
9		X	X			X		X		X
10		X	X			X		X	X	
11*		X	X			X	X			X
12*		X	X			X	X		X	
13*		X	X		X			X		X
14		X	X		X			X	X	
15*		X	X		X		X			X
16		X	X		X		X		X	
17	X			X		X		X		X
18	X			X		X		X	X	
19*	X			X		X	X			X
20*	X			X		X	X		X	
21*	X			X	X			X		X
22	X			X	X			X	X	
23*	X			X	X		X			X
24	X			X	X		X		X	
25	X		X			X		X		X
26	X		X			X		X	X	
27*	X		X			X	X			X
28*	X		X			X	X		X	
29*	X		X		X					X
30	X		X		X				X	
31*	X		X		X		X			X
32	X		X		X		X		X	

Figure 9. Classification of Work Requests

SECTION 6

CHARACTERISTICS OF FDPS CONTROL MODELS

6.1 APPROACHES TO IMPLEMENTING FDPS EXECUTIVE CONTROL

There are two basically different approaches available for implementing an operating system for a distributed processing system, the base-level approach and the meta-system approach [Thom78]. The base-level approach does not utilize any existing software and, therefore, requires the development of all new software. This includes software for all local control functions such as memory management and process management. In contrast, the meta-system approach utilizes the "existing" operating systems (called local operating systems (LOS)) from each of the nodes of the system. Each LOS is "interfaced" to the distributed system by a network operating system (NOS) which is designed to provide high level services available on a system-wide basis. The meta-system approach is usually preferred due to the availability of existing software to accomplish local management functions, thus, reducing development costs [Thom78].

Figure 6 depicts a logical model applicable to an FDPS executive control utilizing either approach. The LOS handles the low-level (processor-specific) operations required to directly interface with users and resources. In the meta-system approach, the LOS represents primarily the operating systems presently available for nodes configured in stand-alone environments. The LOS resulting from a base-level approach has similar functionality; however, it represents a new design, and certain features may be modified in order to allow the NOS to provide certain functions normally provided by the LOS. Any "network" operations are performed by the NOS. System unification is realized through the interaction of NOS components, possibly residing on different processors, acting in cooperation with appropriate LOS components. Communication among the components is provided by the message handler which utilizes the message transport services.

6.2 INFORMATION REQUIREMENTS

Two types of information are required by an executive control, information concerning the structure of the set of tasks required to satisfy the work request and information about system resources. This data is maintained in a variety of data structures by a number of different components.

6.2.1 Information Requirements for Work Requests

Each work request identifies a set of cooperating tasks, nodes in a logical network that cooperate in execution to satisfy a request and the connectivity of those nodes. Figure 10 illustrates the notation used in this project to express work requests. An example of a work request using this notation is presented in Figure 11. Work requests as linear textual forms can be easily accepted and manipulated by the computer system; however, task graphs, which are an internal control structure used to describe work requests, must be represented in a manner such that the linkage information is readily available. This can take the form of the explicit linking of node control blocks (Figure 12) or an interconnection matrix (Figure 13).

Information concerning a particular task, i.e., logical node, is maintained in a node control block (Figure 12). Associated with each logical node is an execution file, a series of input files, and a series of output files. The node control block contains information on each of these entities that includes the name of the resource, the locations of possible candidates that might provide the desired resource, and the location of the candidate resource chosen to be utilized in the satisfaction of the work request. In addition to this information, the node control block maintains a description of all interprocess communication (IPC) in which the node is a party. This consists of a list of input ports and output ports. (Interprocess communication is a term describing the exchange of messages between cooperating processes of a work request.) Typically, a message is "sent" when it is written to the output port of a process. The message is then available for consumption by any process possessing an input port that is connected to the previously mentioned output port. The message is actually consumed or accepted when the process owning the connected input port executes a READ on that port.

A global view of interprocess communication is provided by the node interconnection matrix (Figure 13). This structure indicates the presence or absence of an IPC link between an output port of one node and an input port of another node. Thus, links are assumed to carry data in only a single direction.

An example of a task graph resulting from the work request in Figure 11 utilizing the direct linking of node control blocks is presented in Figure 14. Figure 15 illustrates the utilization of an interconnection matrix.

```
<work request> ::= [ <logical net> { ; <logical net> } ]

<logical net> ::= <logical node> { <node separator>
                  { <node separator> } <logical node> }

<node separator> ::= , | <pipe connection>

<pipe connection> ::= [ <port> ] '|' [ <logical node number> ]
                      [ .<port> ]

<port> ::= <integer>

<logical node number> ::= <integer> | $ | <label>

<logical node> ::= [ :<label> ] [ <simple node> |
                   <compound node> ] |
                   ( <simple node> | <compound node> )

<simple node> ::= { <i/o redirector> } <command name>
                  { <i/o redirector> | <argument> }

<compound node> ::= { <i/o redirector> } '{' <logical net>
                    { <net separator> <logical net> } '}'
                    { <i/o redirecotr> }

<i/o redirector> ::= <file name> '>' [ <port> ]  |
                     [ <port> ] '>' <file name>  |
                     [ <port> ] '>>' <file name> |
                                '>>' [ <port> ]

<net separator> ::= ;

<command name> ::= <file name>

<label> ::= <identifier>
```

Figure 10. Work Request Syntax
(Taken from [AKIN78])

Work Request:

```
pgm1 | pgm2 1|a 2|b :a pgm3 | pgm4 |c.1 :b pgm5 | pgm6 |.2 :c pgm7
    (0)     (1) (2) (3)    (4)    (5)   (6)    (7)    (8)  (9)
```

(0) Output port 1 of pgm1 is connected to input port 1 of pgm2.
(1) Ouptut port 1 of pgm2 is connected to input port 1 of the logical node labeled "a," pgm3.
(2) Output port 2 of pgm2 is connected to input port 1 of the logical node labeled "b," pgm5.
(3) Label for the logical node containing pgm3 as its execution module.
(4) Output port 1 of pgm3 is connected to input port 1 of pgm4.
(5) Output port 1 of pgm4 is connected to input port 1 of the logical node labeled "c," pgm7.
(6) Label for the logical node containing pgm5 as its execution module.
(7) Output port 1 of pgm5 is connected to input port 1 of pgm6.
(8) Output port 1 of pgm6 is connected to input port 2 of pgm7.
(9) Label for the logical node containing pgm7 as its execution module.

Data Flow Graph of the Work Request:

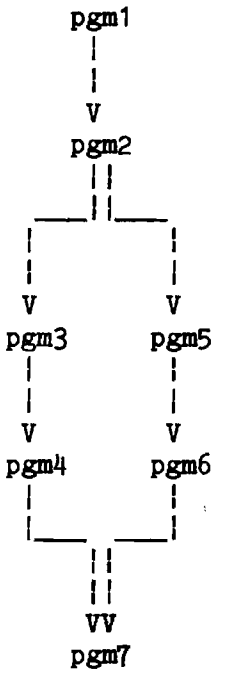


Figure 11. Example of a Work Request

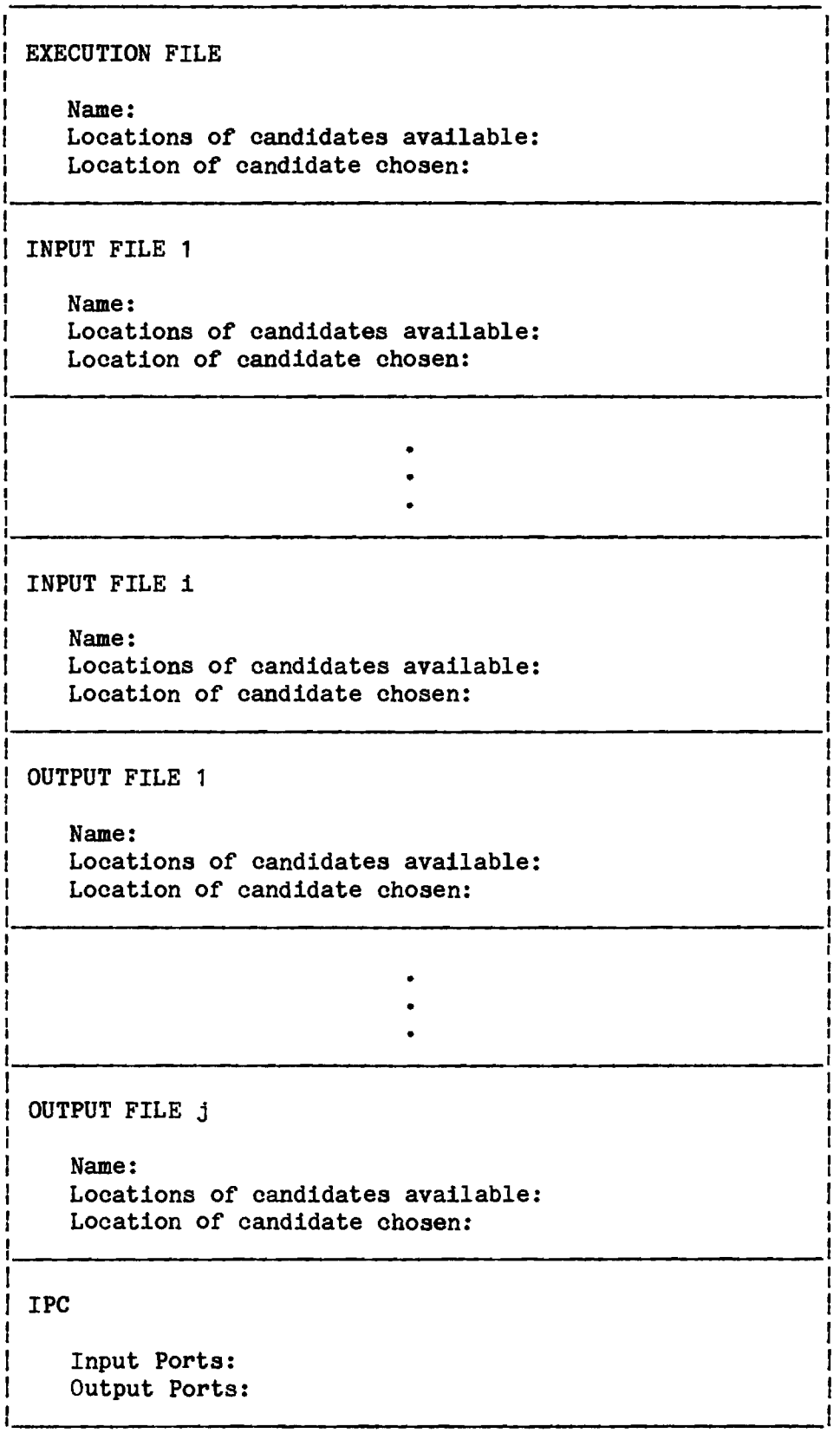


Figure 12. Node Control Block

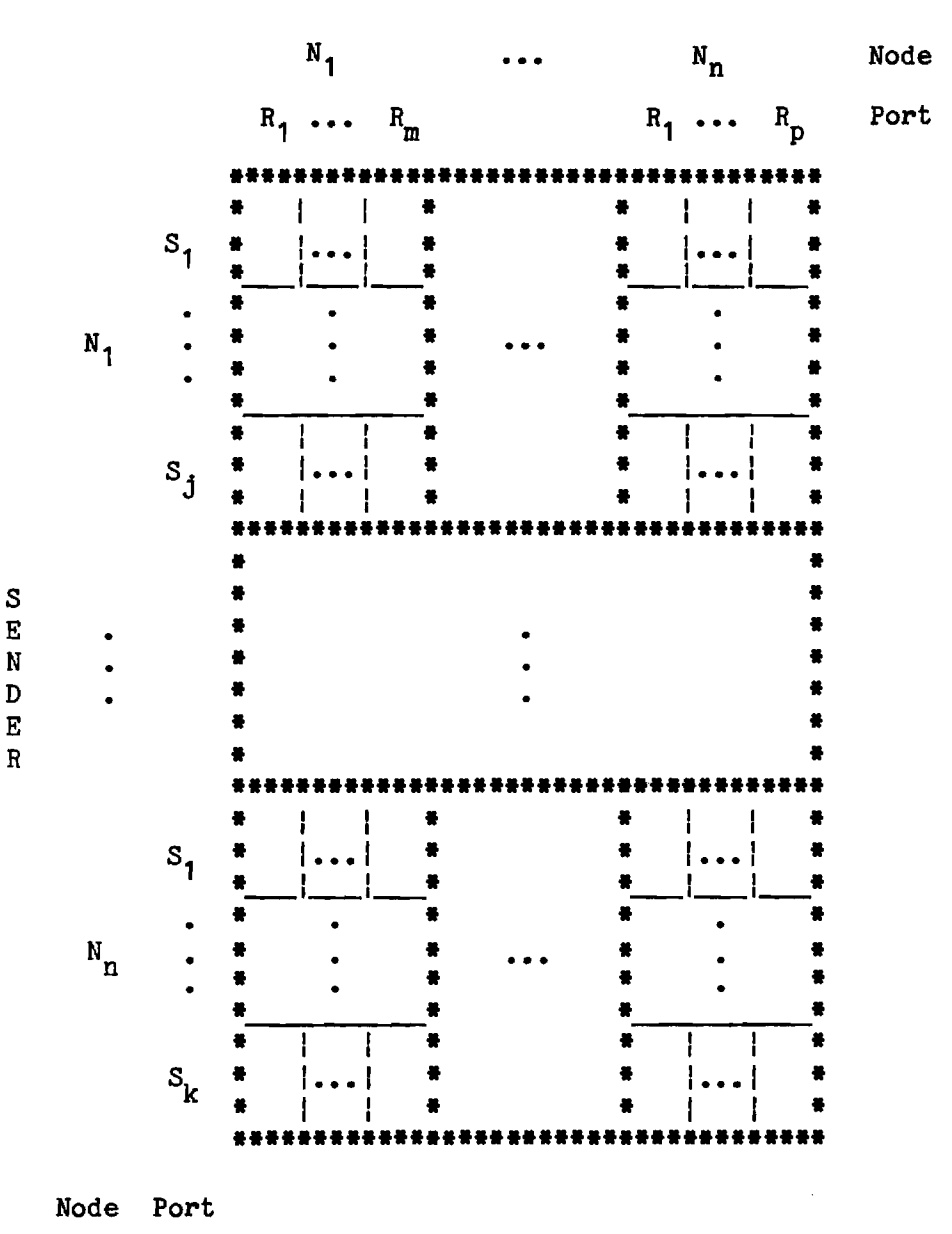


Figure 13. Node Interconnection Matrix

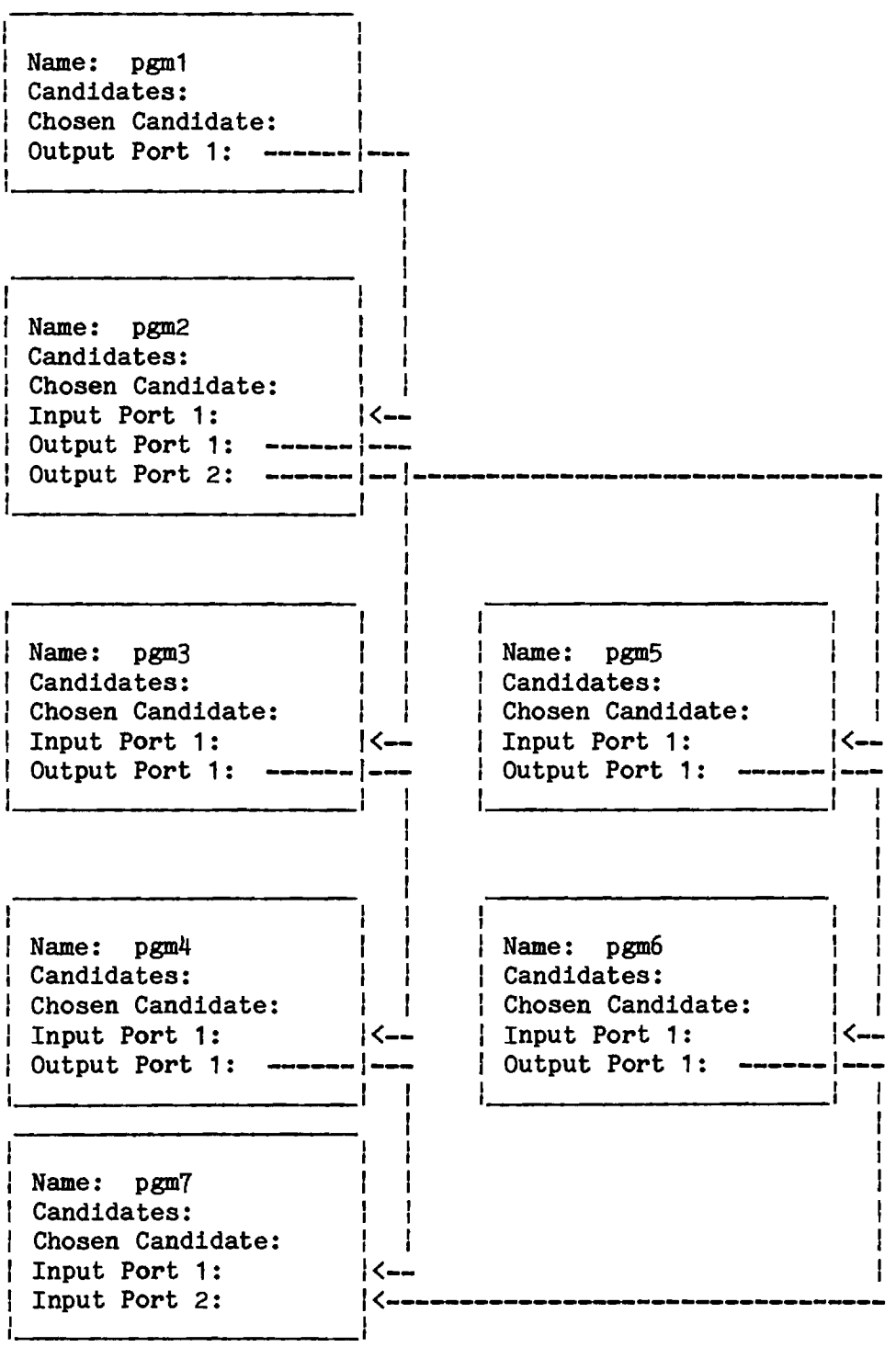


Figure 14. Example of a Task Graph Using Links within the Node Control Blocks

(Based on the Work Request Shown in Figure 11)

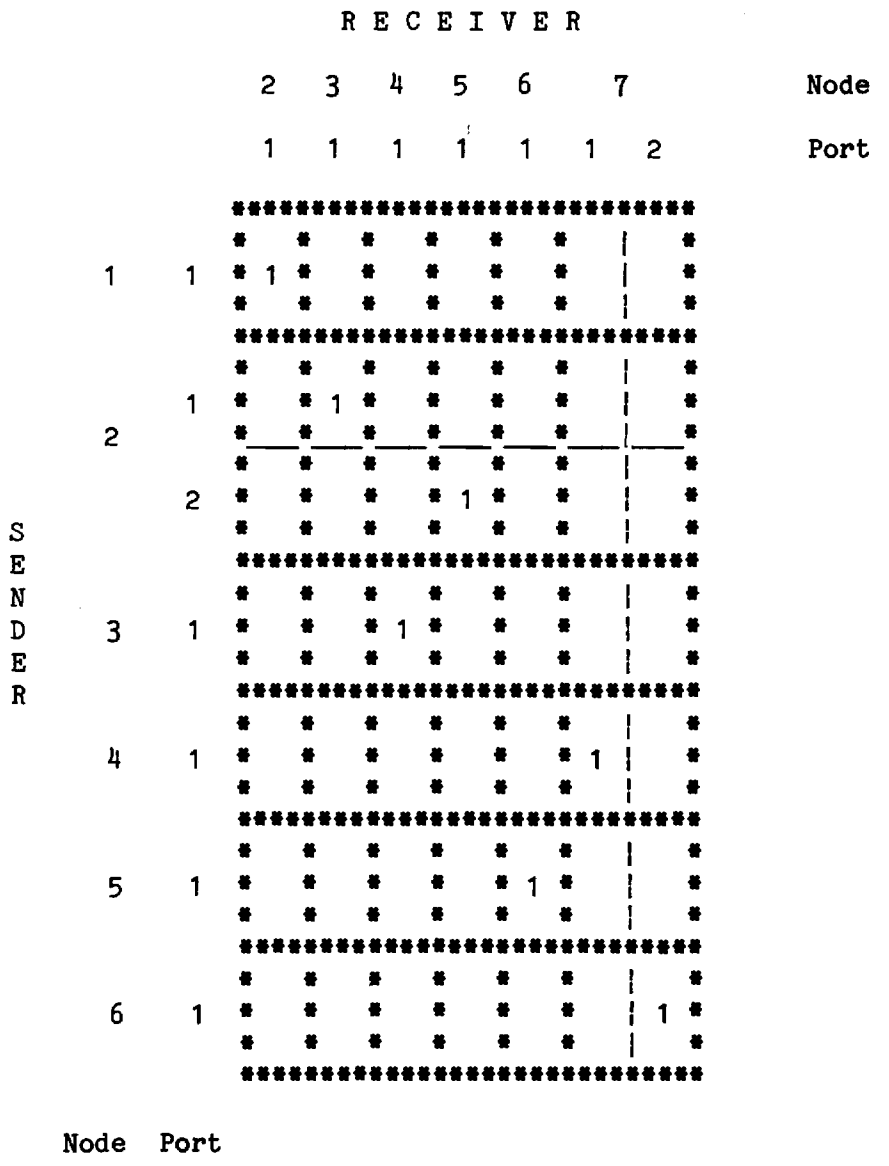


Figure 15. Example of a Node Interconnection Matrix

(Based on Work Request Shown in Figure 11)

6.2.2 Information Requirements for System Resources

Regardless of how the executive control is realized (i.e., how the components of the executive control are distributed and how the control decisions are decentralized), information concerning all system resources (processors, communication lines, files, and peripheral devices) must be maintained. This information includes at a minimum an indication of the availability of resources (available, reserved, or assigned). Preemptable resources (e.g., processors and communication lines) capable of accommodating more than one user at a time may also have associated with them utilization information designed to guide an executive control in its effort to perform load balancing.

As discussed below, there are a number of techniques that may be employed to gather and/or maintain the system resource information.

6.3 BASIC OPERATIONS OF FDPS CONTROL

The primary task of an executive control is to process work requests that can best be described as logical networks. A node of a logical network specifies an execution file that may either contain object code or commands (work requests), input files, and output files. These files may reside on one or more physical nodes of the system and there may be multiple copies of the same file available. Thus, to process a work request, an FDPS executive control must perform three basic operations: 1) gather information, 2) distribute the work and allocate resources, and 3) initiate and monitor task execution. These operations need not be executed in a purely serial fashion but may take a more complex form with executive control operations executed simultaneously or concurrently with task execution as the need arises.

Examination of the basic operations in further detail (Figure 16) reveals some of the variations possible in the handling of work requests. Two steps exist in information gathering --- 1) collecting information about task requirements for the work request and 2) identifying the resources available for satisfying the request requirements. Information gathering is followed by the task of distributing the work and allocating resources. If this operation is not successful, three alternatives are available. First, more information on resource availability can be gathered in an attempt to formulate a new work distribution. There may have been a change in the status of some resources since the original request for availability information. Second, more information can be gathered as above, but this time the requester will

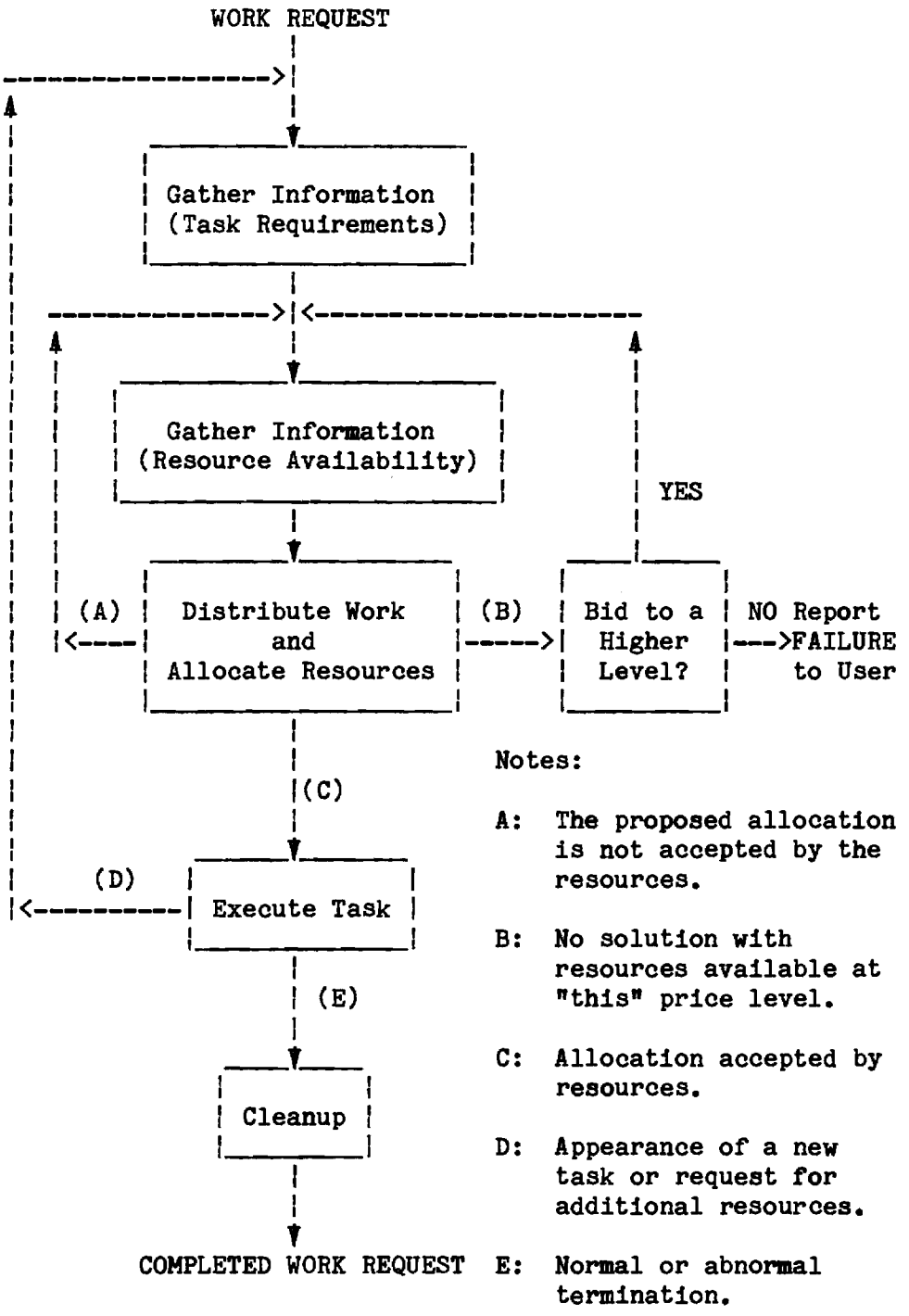


Figure 16. Work Request Processing (Detailed Steps)

indicate a willingness to "pay more" for the resources. This is referred to as bidding to a higher level. Finally, the user can simply be informed that it is impossible to satisfy his work request.

6.3.1 Information Gathering

Upon receiving a work request, the first task of the control is to discover what resources are needed to satisfy the work request (Figure 17) and which resources are available to fill these needs (Figure 18). Each work request includes a description of a series of tasks and the connectivity of those tasks. Associated with each task is a series of files. One is distinguished as the execution file and the rest are input/output files. The executive control must first determine which files are needed. It then must examine each of the execution files to determine the nature of its contents (executable code or commands). Each task will need a processor resource(s), and those tasks containing command files will also require a command interpreter.

An FDPS executive control must also determine which of the system resources are available. For nonpreemptable resources, the status of a resource can be either "available," "reserved," or "assigned." A reservation indicates that a resource may be used in the future and that it should not be given to another user. Typically, there is a time-out associated with a reservation that results in the automatic release of the reservation if an assignment is not made within a specified time interval. The idea here is to free resources that otherwise would have been left unavailable by a lost process. The process may be lost because it failed, its processor failed, or the communication link to the node housing the particular resource may have failed. An assignment, on the other hand, indicates that a resource is dedicated to a user until the user explicitly releases that assignment. Preemptable resources may be accessed by more than one concurrent user and thus can be treated in a different manner. For these resources, the status may be indicated by more continuous values (e.g., the utilization of the resource) rather than the discrete values described above.

6.3.2 Work Distribution and Resource Allocation

The FDPS executive control must determine the work distribution and the allocation of system resources (Figure 19 & 20). This process involves choosing from the available resources those that are to be utilized. This decision

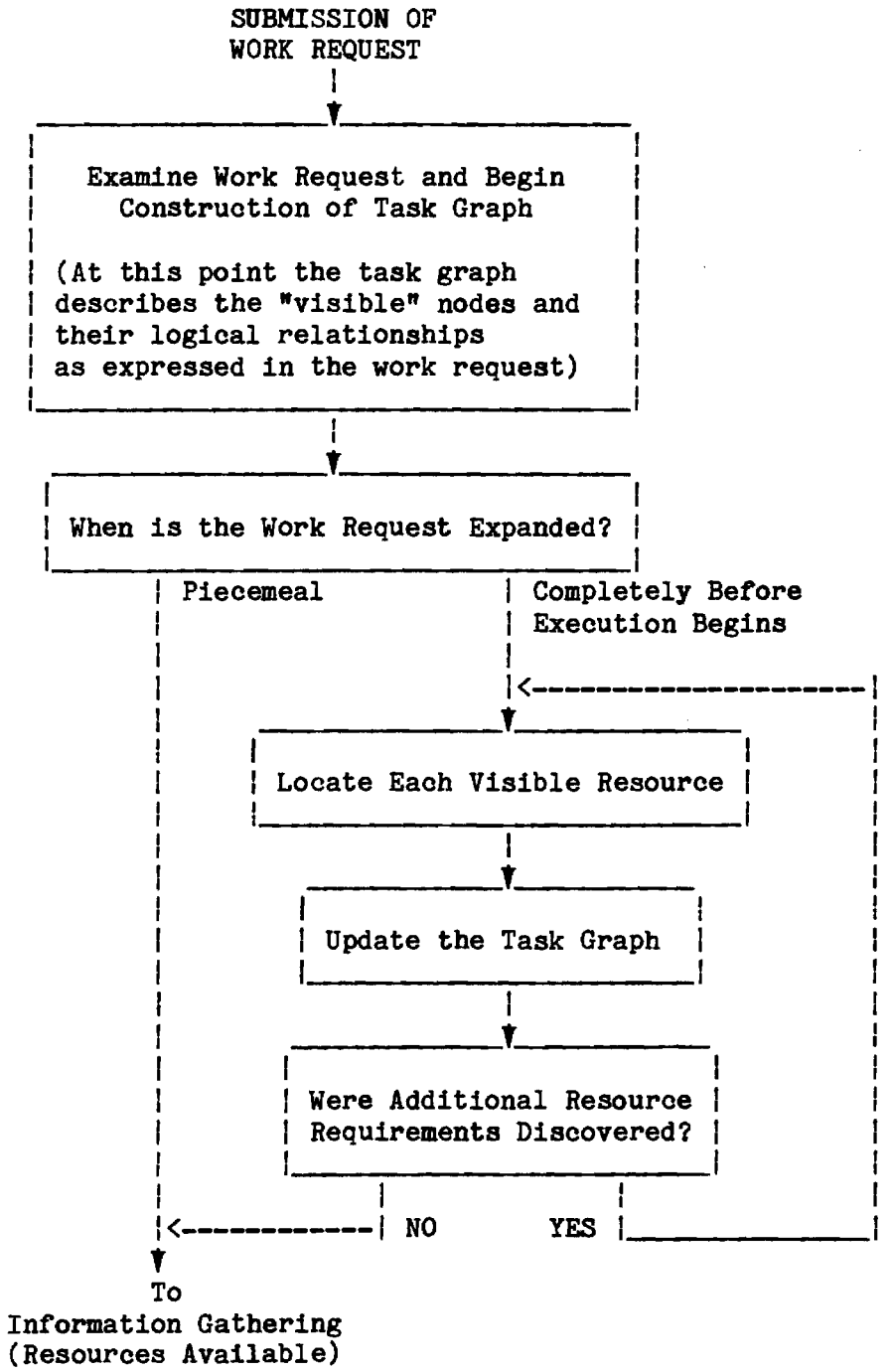
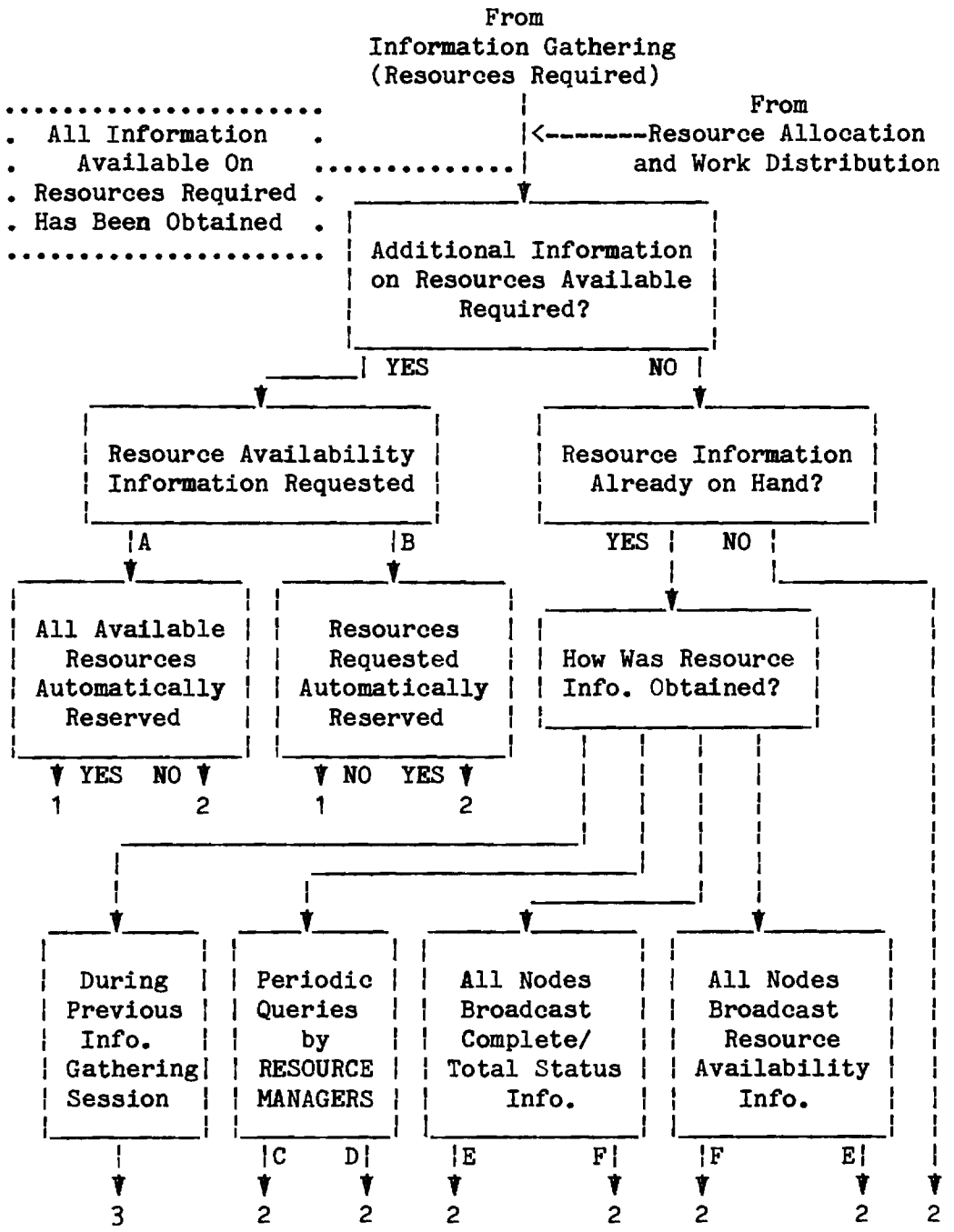


Figure 17. Information Gathering (Resources Required)



LEGEND AND NOTES

1: Resources Reserved During Information Gathering
2: No Resources Reserved
3: Some Resources May Be Reserved
A: General, for all resources
B: To meet specific task/job requirements
C: Replies cover information on resources available only
D: Replies cover information on the total status
E: Broadcast only significant changes
F: Periodic broadcasts at regular intervals

Figure 18. Information Gathering (Resources Available)

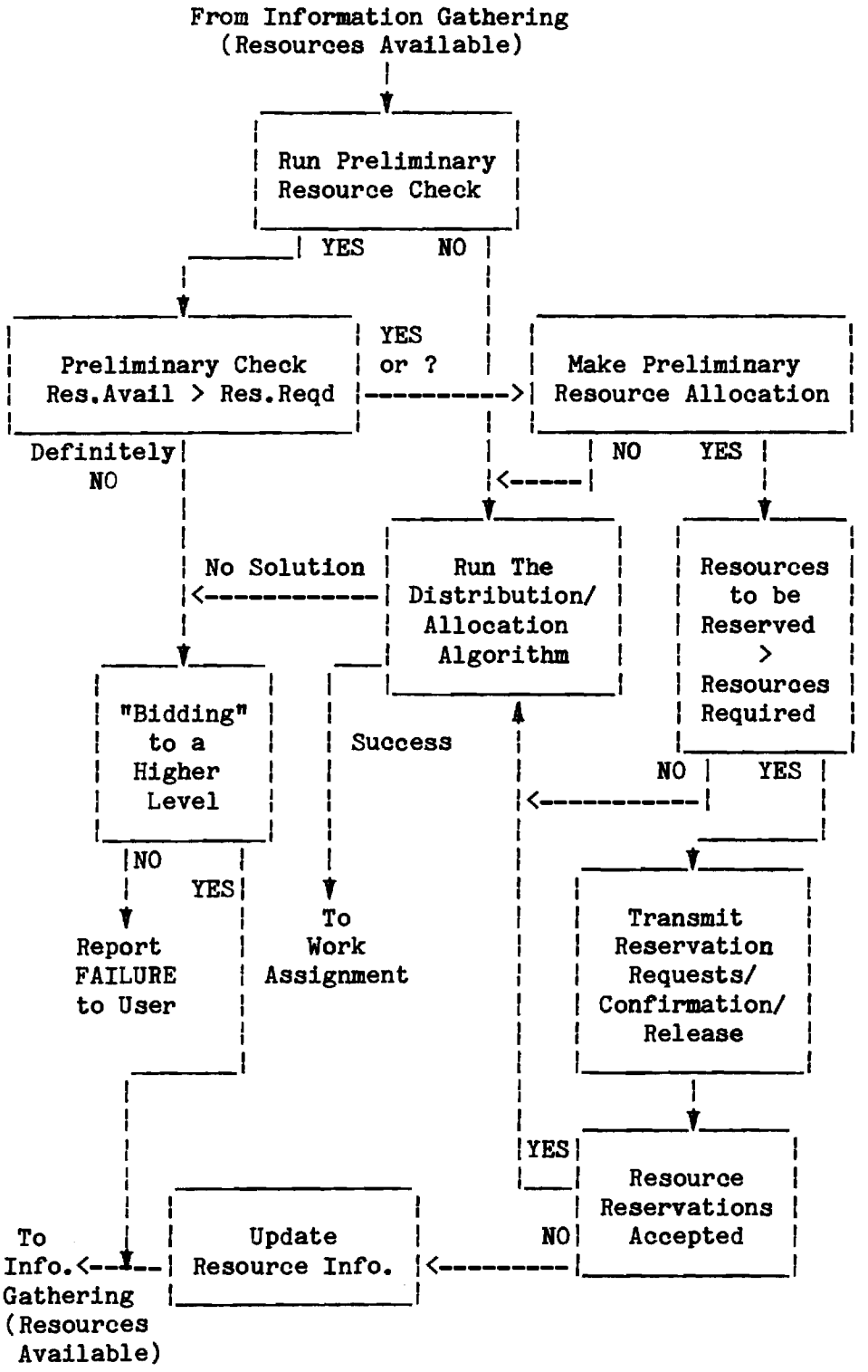


Figure 19. Resource Allocation and Work Distribution

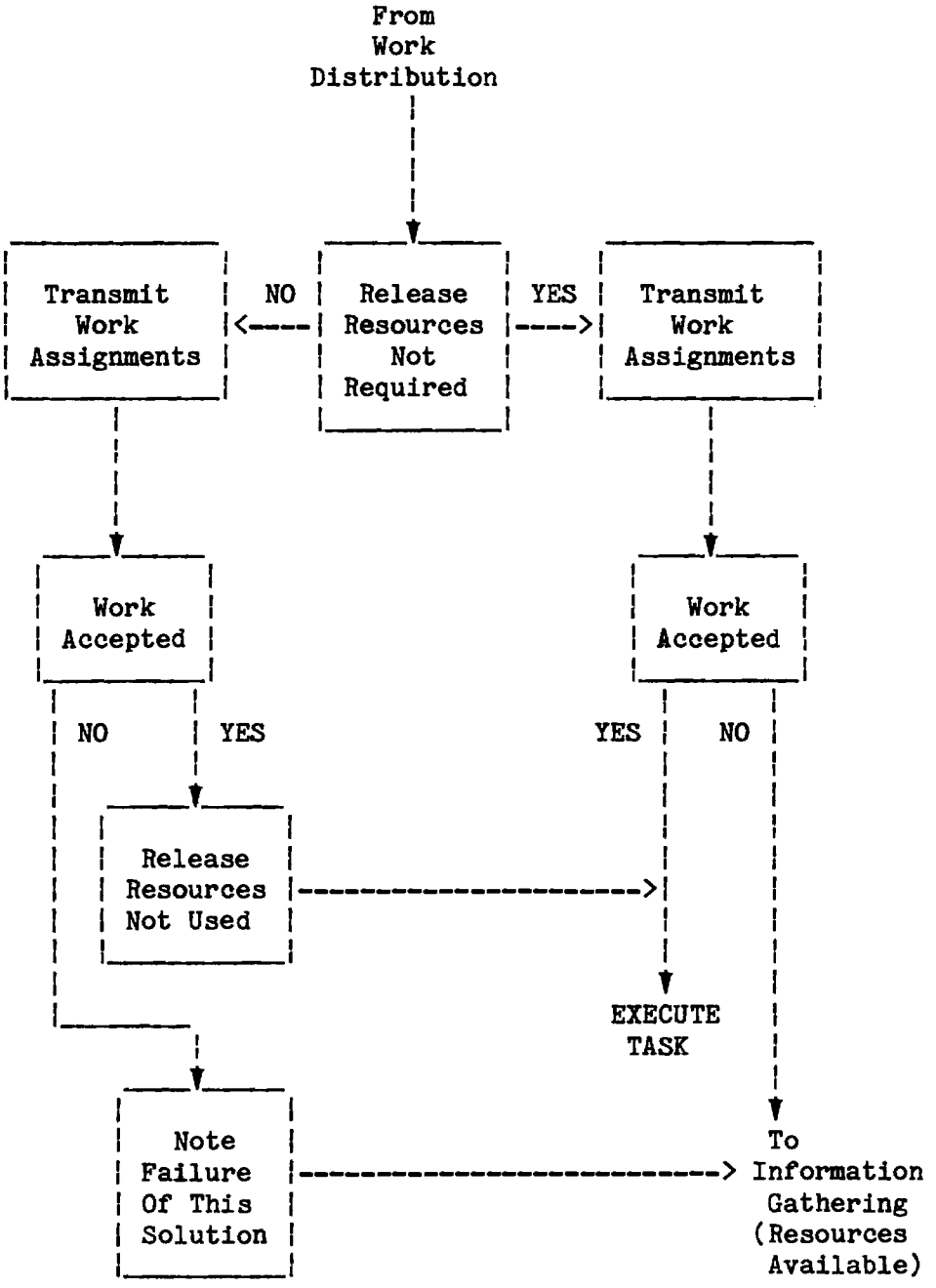


Figure 20. Work Assignment

is designed to achieve several goals such as load balancing, maximum throughput, and minimum response time. It can be viewed as an optimization problem similar in many respects to that discussed by Morgan [Morg77].

Once an allocation has been determined, the chosen resources are allocated and the processes comprising the task set are scheduled and initiated. If a process cannot be immediately scheduled, it may be queued and scheduled at a later time. When it is scheduled, a process control block and any other execution-time data structures must be created.

6.3.3 Information Recording

Information is recorded as a result of management actions as well as providing a means to maintain a historical record or audit trail of system activity. The information recording resulting from management actions maintains the system state and provides information for decision making. The historical information is useful in monitoring system security. It provides a means to examine past activity on a system in order to determine if a breach of security occurred or how a particular problem or breach of security may have occurred.

Management information is maintained in various structures, including the task graph. The task graph is used to maintain information about the structure of an individual work request, and, thus, its contents change as progress on the work request proceeds. A task graph is created when a work request is first discovered, and information is then constantly entered into the structure as work progresses through information gathering to work distribution and resource allocation and on to task execution. The task graph remains active until completion of the work request.

Much of the information contained in the task graph is applicable to historical records. In fact, the task graph can be used to house historical information as it is gathered during work request processing. Upon completion of the work request, the historical information is extracted and entered into the permanent historical file. Alternatively, the historical file can be created directly skipping the intermediate task graph structure.

6.3.4 Task Execution

Finally, an executive control must monitor the execution of active processes. This includes providing interprocess communication, handling requests from active processes, and supervising process termination. The

activities associated with interprocess communication include establishing communication paths, buffering messages, and synchronizing communicating processes. The latter activity is necessary to protect the system from processes that flood the system with messages before another process has time to absorb the messages. Active processes may also make requests to the executive control. These may take the form of additional work requests or requests for additional resources. Work requests may originate from either command files or files containing executable code.

An executive control must also detect the termination of processes. This includes both normal and abnormal termination. After detecting process termination, it must inform processes needing this information that termination has occurred, open files must be closed, and other loose ends must be cleaned up. Finally, when the last process of a work request has terminated, it must inform the originator of the request of the completion of the request.

6.3.5 Fault Recovery

If portions (tasks) of the work request are being performed on different processors, there is inherently a certain degree of fault recovery possible. The problem is in exploiting that capability. The ability to utilize "good" work remaining after the failure of one or more of the processors executing a work request depends on the recovery agent having knowledge of the location of that work and the ability of the recovery agent to reestablish the appropriate linkages to the new locations for the portions of the work that were being executed on the failed processor(s).

SECTION 7

VARIATIONS IN FDPS CONTROL MODELS

There is an extremely large number of features by which variations in distributed control models can be characterized. Of these, only a few basic attributes appear to deserve attention. These include the nature of how and when a task graph is constructed, the maintenance of resource availability information, the allocation of resources, process initiation, and process monitoring. In this section, these issues are examined; but again, since the number of variations possible in each issue are rather large, only those choices considered significant are discussed. Table 2 contains a summary of the problems that have been identified and possible solutions (significant and reasonable solutions) to these problems.

7.1 TASK GRAPH CONSTRUCTION

The task graph is a data structure used to maintain information about the applicable task set. The nodes of a task graph represent the tasks of the task set, and the arcs represent the connectivity or flow of information between tasks. There are basically four issues in task graph construction: 1) who builds a task graph, 2) what is the basic structure of a task graph, 3) where are the copies of a task graph stored, and 4) when is a task graph built.

The identity of the component or components constructing the task graph is an issue that presents three basic choices. First, a central node can be responsible for the construction of task graphs for all work requests. Another choice utilizes the control component on the node receiving the work request to construct the task graph. Finally, the job of building the task graph can be distributed among several components. In particular, the nodes involved in executing individual tasks of the work request can be responsible for constructing those parts of the task graph that they are processing.

The general nature of the task graph itself provides two alternatives for the design of an executive control. What is of concern here is not the content of a task graph but rather its basic structure. One alternative is to maintain a task graph in a single structure regardless of how execution is distributed. The other choice is to maintain the task graph as a collection of subgraphs with each subgraph representing a part of the work request. For

Table 2. Variations in Control Models

TASK GRAPH CONSTRUCTION:

Who builds the task graph?
1. A central node specializing in task graph building.
2. The node intially receiving and analyzing the work request.
3. All nodes involved in executing the work request.

What is the nature of the task graph?
1. A single complete structure.
2. Multiple structures each consisting of a subgraph.
3. Multiple structures each consisting of a subgraph with one copy of the complete task graph.

Where is the task graph stored?
1. A central node.
2. The node intially receiving and analyzing the work request.
3. A node determined to be in an optimal location.
4. All nodes involved in executing the work request.

When is the task graph built?
1. Completely prior to execution.
2. Piecemeal during execution.

RESOURCE AVAILABILITY INFORMATION:

Who maintains this information?
1. A single central nnde.
2. Each node maintains information about its own resources.
3. All nodes maintain common information.
4. A designated node for each type of resource.

Where is the information maintained?
1. At a central node.
2. Separate pieces of information concerning a particular resource type may be kept on different nodes.
3. In multiple redundant copies.
4. Information concerning a particular resource type is kept on a specially designated node.

ALLOCATION OF RESOURCES:

How is concurrency control provided?
1. None is provided.
2. Reservations are used prior to a work distribution decision and then allocated by a lock.
3. Allocated by a lock after the work distribution decision.
4. Resources are locked before the work distribution decision is made.

PROCESS INITIATION:

How is responsibility distributed?
1. A central component retains all responsibility.
2. A single component is in charge of a single work request.
3. There is a hierarchy of responsibility.
4. Responsibility is distributed among specialist components.

How is refusal of a request to execute a process by a node handled?
1. After repeated attempts, the request is abandoned.
2. After repeated attempts, a new work distribution is obtained.

PROCESS MONITORING:

What type of interprocess communication is provided?
1. Synchronized communication.
2. Unsynchronized communication.

How are task graphs resulting from additional work requests handled?
1. The new task graph is made part of the old one.
2. The new task graph is kept separate.

PROCESS TERMINATION:

Options selected here are determined by those selected for PROCESS INITIATION.

example, a subgraph can represent that portion of the work request that is to be executed on the particular node at which that subgraph is stored.

Another issue of task graph construction concerns where the various copies of the task graph are stored. If the control maintains a task graph as a unified structure representing the complete set of tasks for a work request, this structure may either be stored on a single node, or redundant copies can be stored on multiple nodes. The single node can either be a central node that is used to store all task graphs, the node at which the original work request arrived (the source node), or a node chosen for its ability to provide this work request with optimal service. If the task graph is divided into several subgraphs, these can be maintained on multiple nodes.

Finally, there is the issue concerning the timing of task graph construction in the sequence of steps that define work request processing. Two choices are available: 1) the task graph can be constructed completely, at least to the maximum extent possible, before execution is begun, or 2) the task graph can be constructed incrementally as execution progresses.

7.2 RESOURCE AVAILABILITY INFORMATION

Another possible source of variability for control models is the maintenance of resource availability information. What is of importance here is "Who maintains this information" and "Where is this information maintained." A particular model need not uniformly apply the same technique for maintaining resource availability information to all resources. Rather, the technique best suited to a particular resource class may be utilized.

The responsibility for maintaining resource availability information can be delegated in a variety of ways. The centralized approach involves assigning a single component this responsibility. In this situation, requests and releases for resources flow through the specialized component which maintains the complete resource availability information in one location.

A variation of this technique maintains complete copies of the resource availability information at several locations [Caba79a,b]. Components at each of these locations are responsible for updating their copy of the resource availability information in order to keep it consistent with the other copies. This requires a protocol to insure that consistency is maintained. For example, two components should not release a file for writing to different users

at the same time. To provide this control, messages containing updates for the information tables must be exchanged among the components. In addition, a strategy for synchronizing the release of resources is required. An example of such a strategy is found in [Caba79a,b] where a baton is passed around the network. The holder of the baton is permitted to release resources.

Another approach exhibiting more decentralization requires dividing the collection of resources into subsets or classes and assigning separate components to each subset. Each component is responsible for maintaining resource availability information on a particular subset. In this case, requests for resources can only be serviced by the control component responsible for that resource. Resources may be named in a manner such that the desired manager is readily identifiable. Alternatively, a search may be required in order to locate the appropriate manager. This search may involve passing the request from component to component until one is found that is capable of performing the desired operation.

Preemptable resources which can be shared by multiple concurrent users (e.g., processors and communication lines) do not necessarily require the maintenance of precise availability information. For these resources, it is reasonable to maintain only approximate availability information because such resources are rarely exhausted. The primary concern in this instance is degraded performance. Therefore, a good estimate of resource utilization is needed.

7.3 ALLOCATING RESOURCES

One of the major problems experienced in the allocation of resources is concurrency control. In a hospitable environment, it is possible to ignore concurrency control. The users are given the responsibility of insuring that access to a shared resource such as a file is handled in a consistent manner. In other environments, for example that presented by an FDPS, this is an important issue. In an FDPS, the problem is even more difficult than in a centralized system due to the loose coupling inherent in the system.

There are basically two approaches to solving the problem of concurrent requests for shared resources. The first utilizes the concept of a reservation. Prior to the allocation of resources (possibly when resource availability information is acquired), a resource may be reserved. The reser-

vation is effective for only a limited period (a period long enough to make a work distribution decision and allocate the resources determined by the decision) and prevents other users from acquiring the resource. The other solution to this problem is to make the work distribution decision without the aid of reservations. If resources cannot be allocated, the executive control will either wait until they can be allocated or attempt a new work distribution.

7.4 PROCESS INITIATION

Several issues arise concerning process initiation. Chief among these is the distribution of responsibility. There are a large number of organizations possible, but only a few are reasonable. The basic organizations utilize either a single manager, a hierarchy of managers, or a collection of autonomous managers. Two approaches result from the single manager concept. In the first organization, a central component is in charge of all work requests and the processes resulting from these work requests. All decisions concerning the fate of processes and work requests are made by this component. A variation on this organization assigns responsibility at the level of work requests. In other words, separate components are assigned to each work request. Each component makes all decisions concerning the fate of a particular work request and its processes.

Management can also be organized in a hierarchical manner. There are a variety of ways hierarchical management can be realized, but we will concentrate on only two, the two-level hierarchy and the n-level hierarchy. The two-level hierarchy has at the top level a component that is responsible for an entire work request. At the lower level are a series of components each responsible for an individual task of the work request. The lower level components take direction from the high level component and provide results to this component. The n-level hierarchy utilizes in its top and bottom levels the components described for the two-level hierarchy. The middle levels are occupied by components that are each responsible for a subgraph of the entire task graph. Therefore, a middle component takes direction from and reports to a higher level component which is in charge of a part of the task graph that includes the subgraph for which the middle component is responsible. The middle component also directs lower level components each of which are responsible for a particular task.

Another organizational approach utilizes a series of autonomous management components. Each component is in charge of some subset of the tasks of a work request. Cooperation of the components is required in order to realize the orderly completion of a work request.

Regardless of the organization, at some point, a request for the assumption of responsibility by a component will be made. Such a request may be reasonably denied for two reasons: 1) the component does not possess enough resources to satisfy the request (e.g., there may not be enough space to place a new process on an input queue), or 2) the component may not be functioning. The question that arises concerns how this denial is handled. One solution is to keep trying the request either until it is accepted or until a certain number of attempts have failed. In this case if the request is never accepted, the work request is abandoned, and the user is notified of the failure. Instead of abandoning the work request, it is possible that a new work distribution decision can be formulated utilizing the additional knowledge concerning the failure of a certain component to accept a previous request.

7.5 PROCESS MONITORING

The task of monitoring process execution presents the FDPS executive control with two major problems, providing interprocess communication and responding to additional work requests and requests for additional resources. With regard to the problem of interprocess communication, there is some question as to the nature of the communication primitives an FDPS executive control should provide. This question arises due to the variety of communication techniques being offered by current languages. There are two basic approaches found in current languages, synchronized communication and unsynchronized communication (buffered messages). Synchronized communication requires that the execution of both the sender and the receiver be interrupted until a message has been successfully transferred. Examples of languages utilizing this form of communication are Hoare's Communicating Sequential Processes [Hoar78] and Brinch Hansen's Distributed Processes [Brin78]. In contrast, buffered messages allow the asynchronous operation of both senders and receivers. Examples of languages using this form of communication are PLITS [Feld79] and STARMOD [Cook80].

The executive control is required to provide communication primitives that are suitable to one of the communication techniques discussed above. If

the basic communication system utilizes synchronized communication, both techniques can be easily handled. The problem with this approach is that there is extra overhead incurred when providing the message buffering technique. On the other hand if the basic communication system utilizes unsynchronized communication, there will be great difficulty in realizing a synchronized form of communication.

The task of monitoring processes also involves responding to requests generated by the executing tasks. These may be either requests for additional resources (e.g., an additional file) or new work requests. If the request is a work request, there is a question as to how a new set of tasks is to be associated with the existing set of tasks. The new set could either be included in the existing task graph, or a new task graph could be constructed for these new tasks. The former technique allows the component making the work distribution decision for the new work request to consider the utilization of other resources by the control. The latter technique does not allow such a situation to occur.

7.6 PROCESS TERMINATION

When a process terminates there is always some cleanup work that must be accomplished (e.g., closing files, returning memory space, and deleting records concerning that process from the executive control's work space). In addition, depending on the reason for termination (normal or abnormal), other control components may need to be informed of the termination. In the case of a failure, the task graph will contain the information needed to perform cleanup operations (e.g., the identities of the processes needing information concerning the failure). Both the nature of the cleanup and the identity of the control components that must be informed of the termination are determined from the design decisions resulting from the issues discussed in Section 7.5.

7.7 EXAMPLES

To gain a better appreciation of some of the basic issues of control in an FDPS, it is useful to examine several examples of work request processing on an FDPS. In each example, emphasis is placed on the operations involved in the construction of task graphs. The work distribution decision that is utilized is a simple one that assigns the execution of processes to the same nodes that house the files containing their code. The concern of the first

eight examples is the impact of variations in work requests on task graph construction. In these examples, the various parts of the overall task graph describing the complete work request are stored on the nodes utilized by each part. The last three examples, though, examine three different techniques for storing the task graphs. In the examples (Figures 21 to 31) the following symbols are utilized:

[ ]	visible external reference(s)
{ }	embedded external reference(s)
(n)A	responsibility for A delegated from node n
A(n)	responsibility for A delegated to node n
a-->b	IPC from process a to process b
A,B,...	uppercase letters indicate command files
a,b,...	lowercase letters indicate executable files
u,v,w,x,y,z	indicate data files

The first example (Figure 21) consists of a simple request in which all external references made are visible and all files required are present on the node where the original request arrived (referred to as the source node). Since the references are visible, the entire task graph can be completed in one step. The second example (Figure 22) is similar to the first except that there are more references that are chained. Again, since all references are visible, the entire task graph can be completed in one step. This work request can be processed in an alternate manner as shown by the third example (Figure 23) where references are located and linked in a piecemeal fashion. Finally, example 4 (Figure 24) adds a slight variation by introducing an explicit interprocess communication (IPC) definition. In this case, the task graph can still be constructed in one step because all references are visible.

The next series of examples consider the impact of locating resources on nodes other than the source node. In example 5 (Figure 25), all the referenced resources reside on a single node other than the source node with the exception of one resource that has redundant copies on two different nodes. Since the resources are not on the source node, negotiation is required to transfer responsibility for a piece of the task graph. In addition, since there is a resource with two redundant copies, a decision as to which to utilize must be made and a negotiation must occur to transfer responsibility. Example 6 (Figure 26) is similar to example 5 and demonstrates the impact of IPC across nodes.

The effect of embedded references is demonstrated in examples 7 and 8. In example 7 (Figure 27), all resources turn out to reside on the source node.

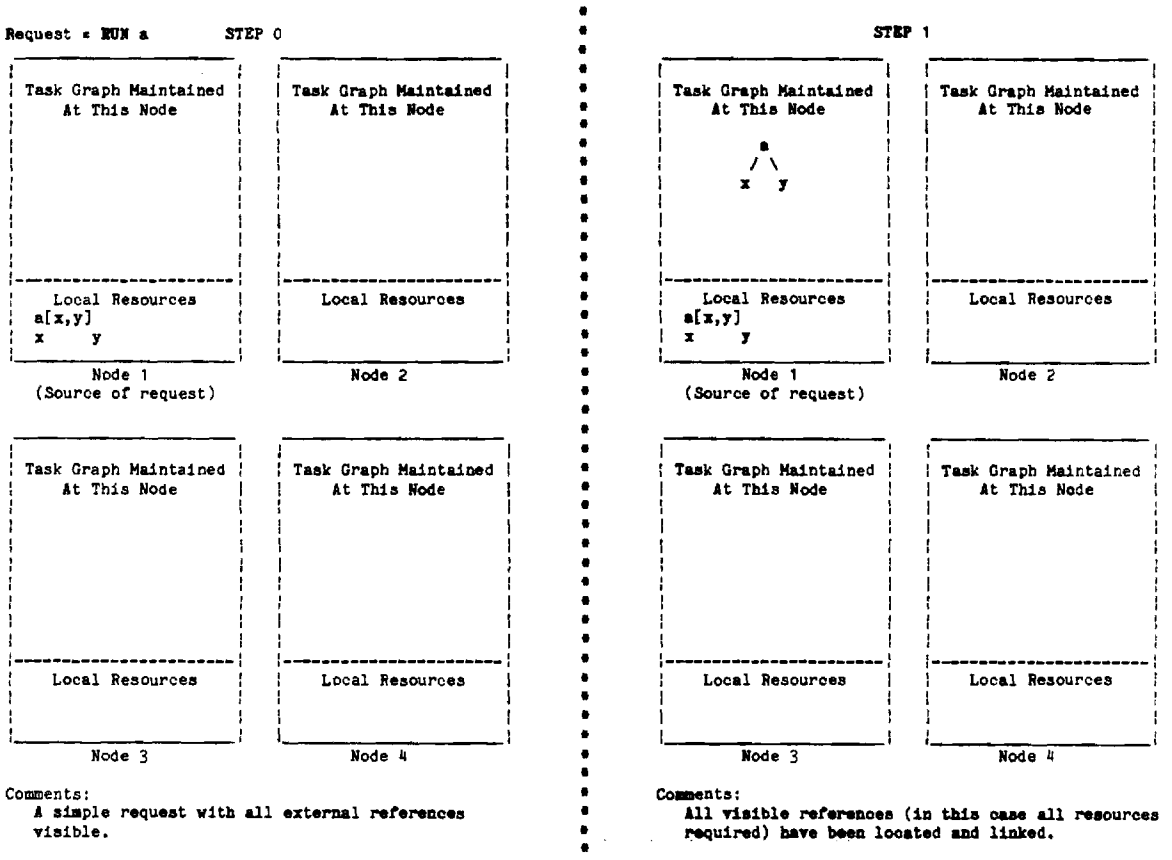


Figure 21. Example 1

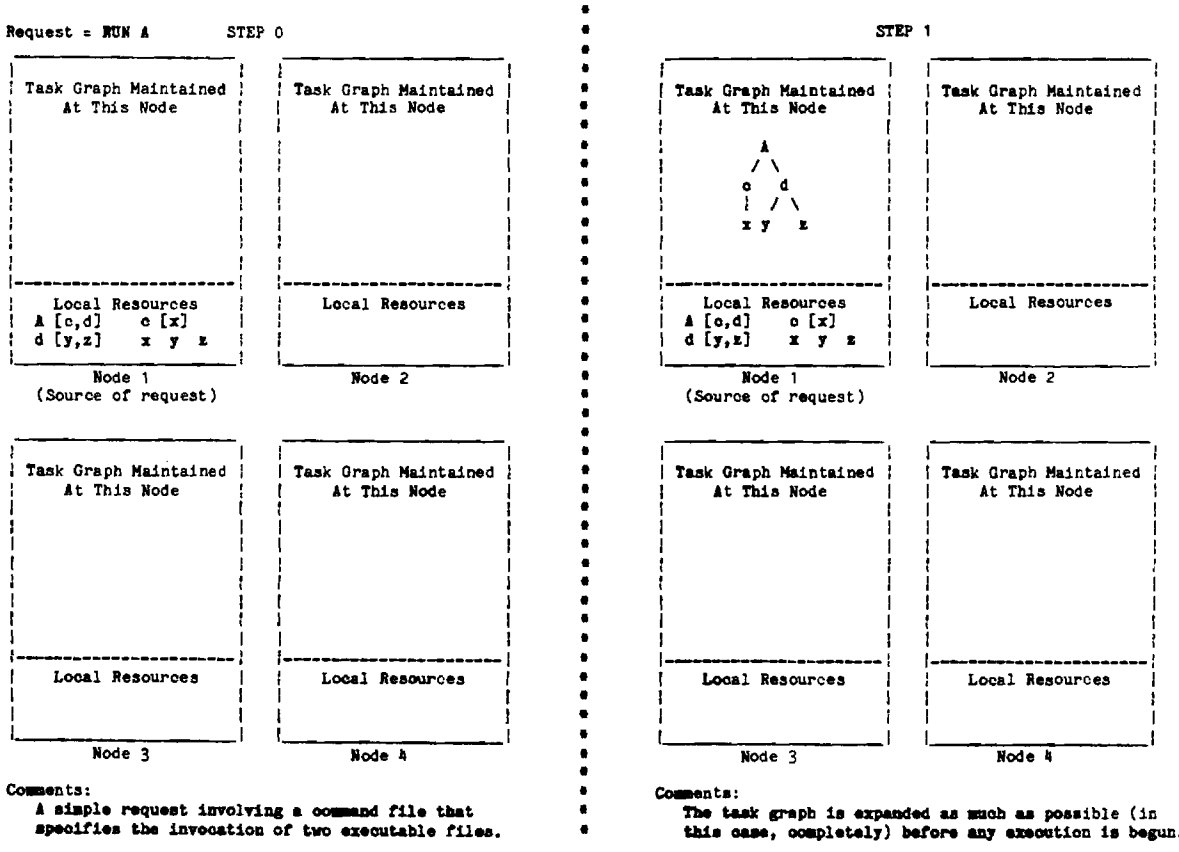


Figure 22. Example 2

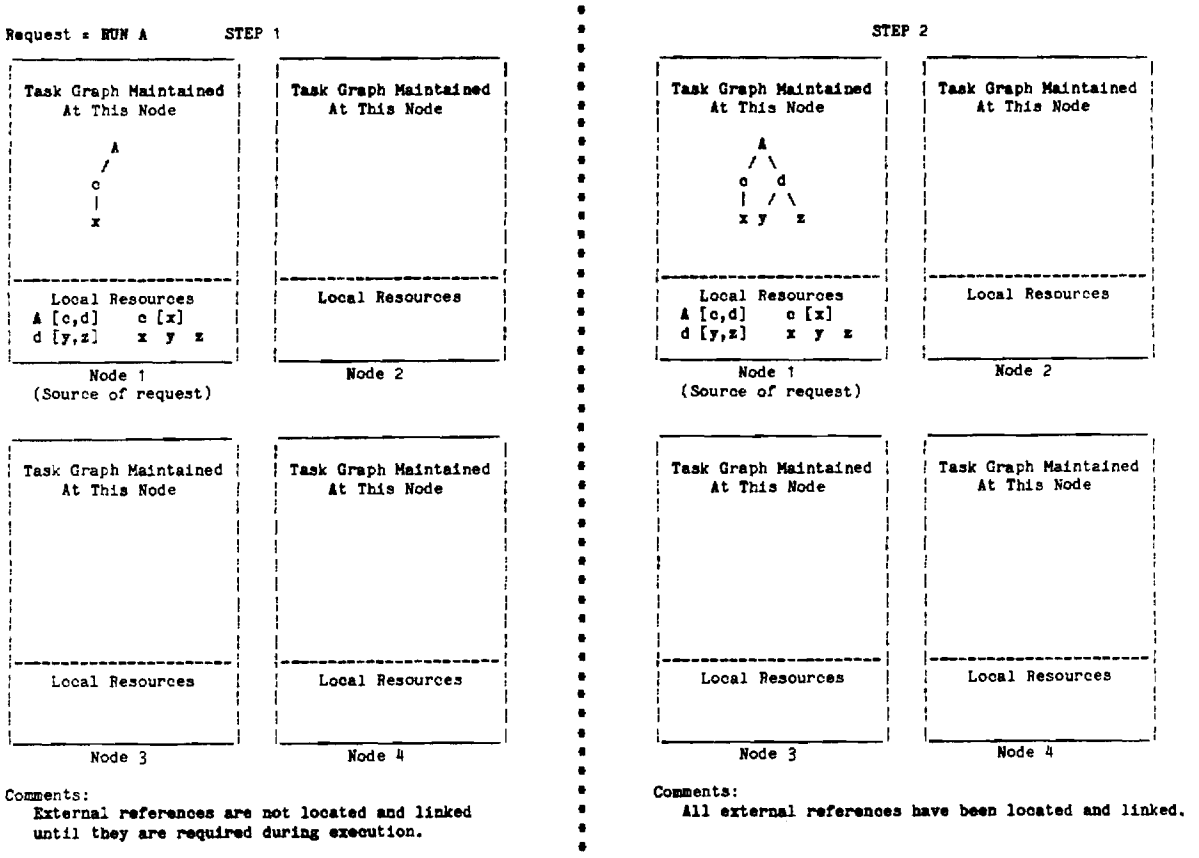


Figure 23. Example 3

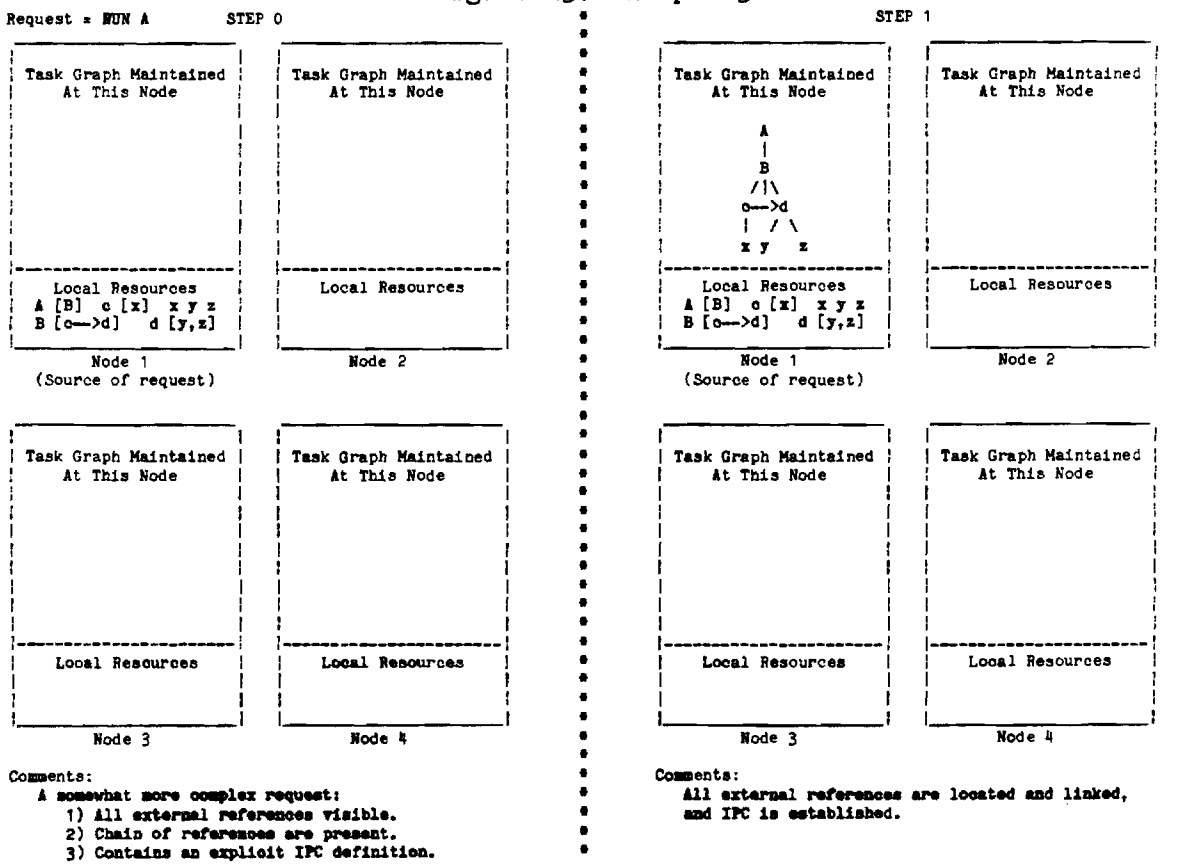


Figure 24. Example 4

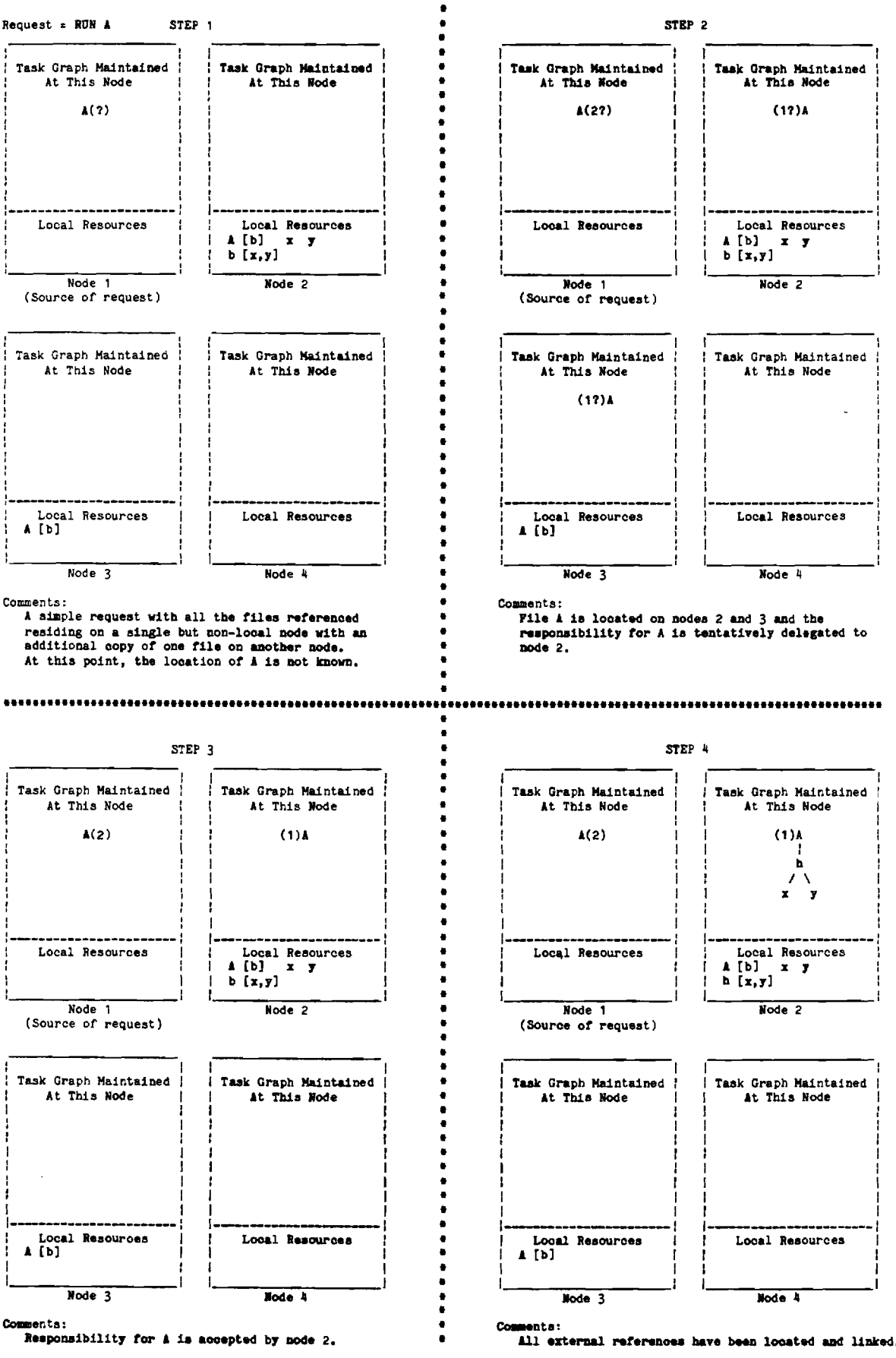


Figure 25. Example 5

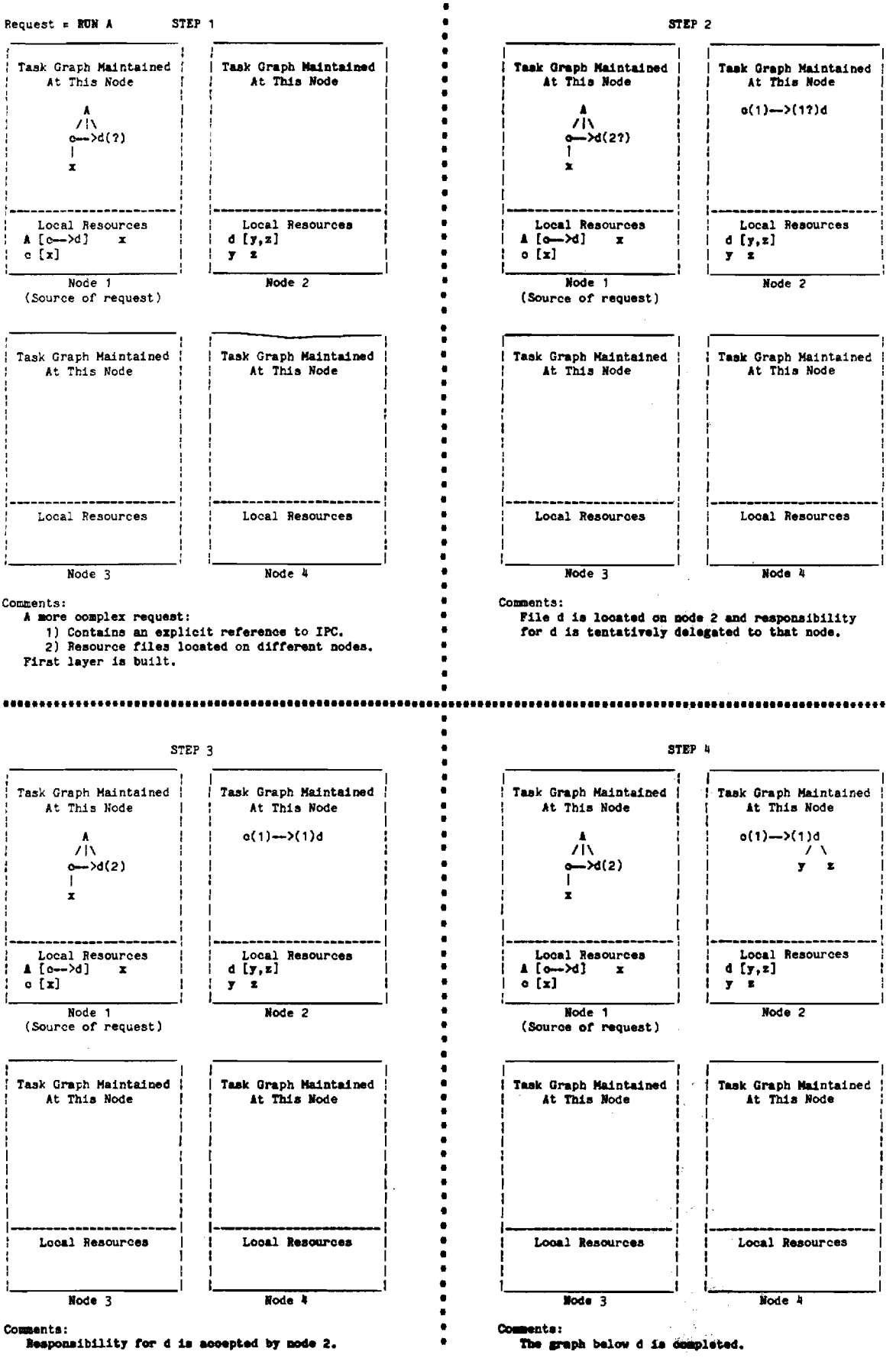


Figure 26. Example 6

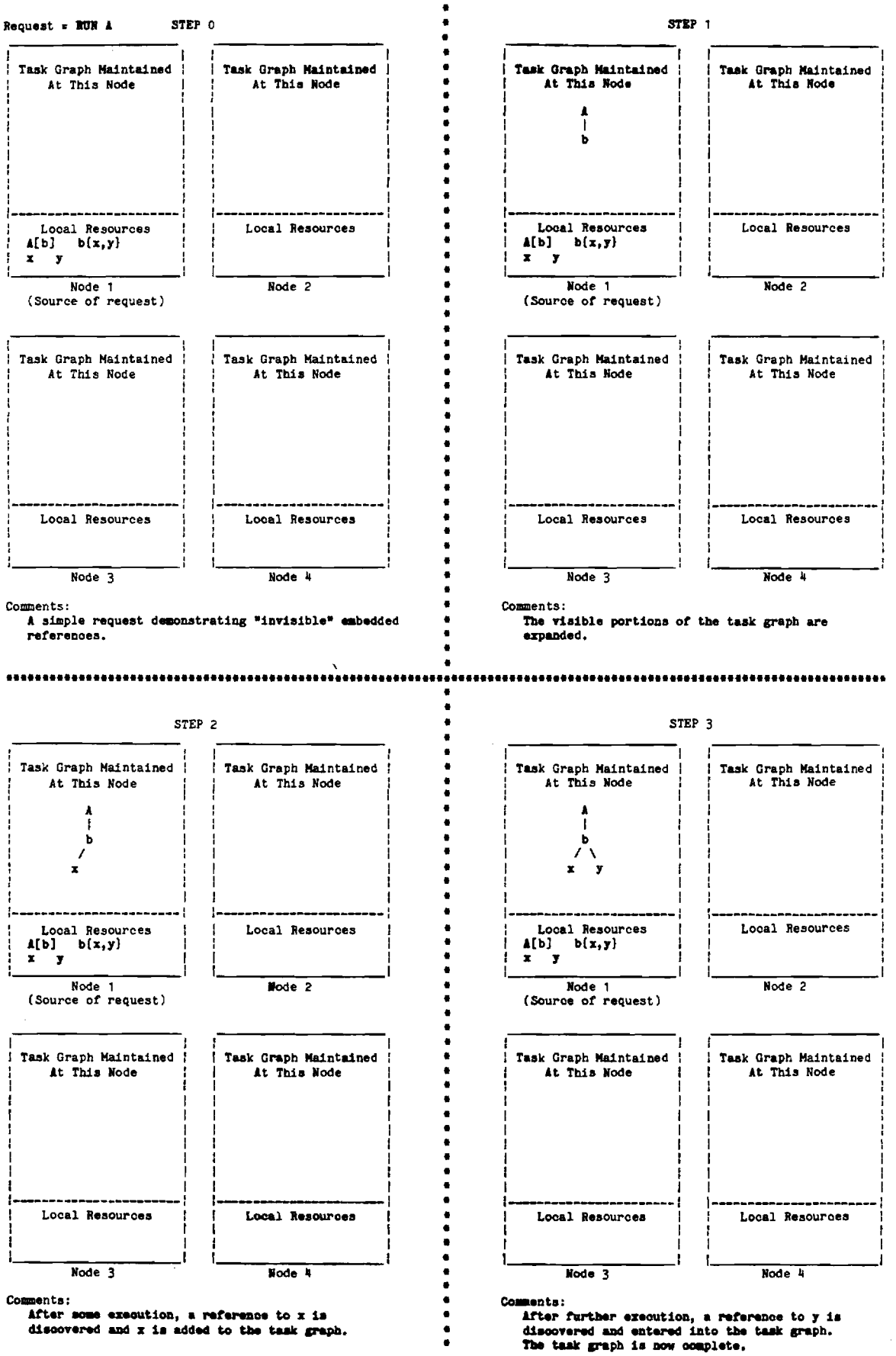


Figure 27. Example 7

Multiple steps, though, are required to construct the task graph because not all of the resources are visible and thus cannot be identified until after execution has progressed to the point where the reference is encountered. Example 8 (Figure 28) is slightly more complex with resources spread over multiple nodes. Again multiple steps are required since parts of the task graph cannot be constructed until their references are observed. In addition since resources are distributed on different nodes, negotiation must occur.

The last three examples demonstrate three different techniques for storing task graphs. In each example, the same work request is utilized. This request has all visible references to resources distributed over multiple nodes. In the first eight examples and example 9 (Figure 29), the parts of the overall task graph are stored on the nodes executing their processes. In addition, each subgraph contains a small portion of information linking it to the rest of the overall task graph. Example 10 (Figure 30) maintains these subgraphs and in addition retains a complete task graph at the source node. Finally, example 11 (Figure 31) maintains complete task graphs at all nodes where processing occurs. The motivation for the last two techniques in which a large amount of redundant information is maintained is to enhance the ability to recover from failures.

Now that we have taken a look at the construction of task graphs in a broad sense, let us examine the details of the task of processing a work request. This is illustrated in two figures. Figure 32 outlines the basic steps involved in work request processing. Finally, Figure 33 depicts the steps involved in processing a specific work request. In this case, the work request is the same as that from example 6 (c.f., Figure 26).

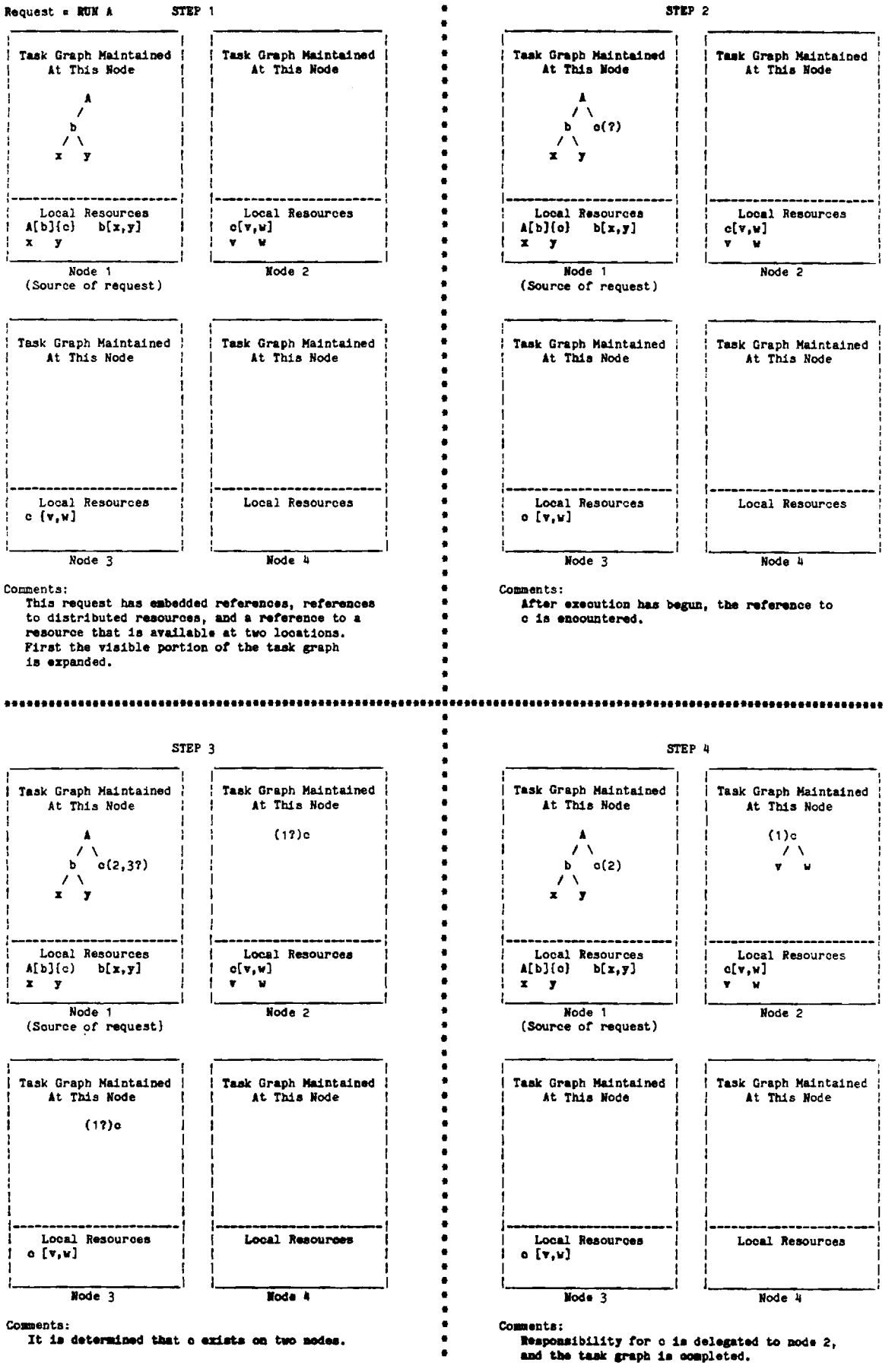


Figure 28. Example 8

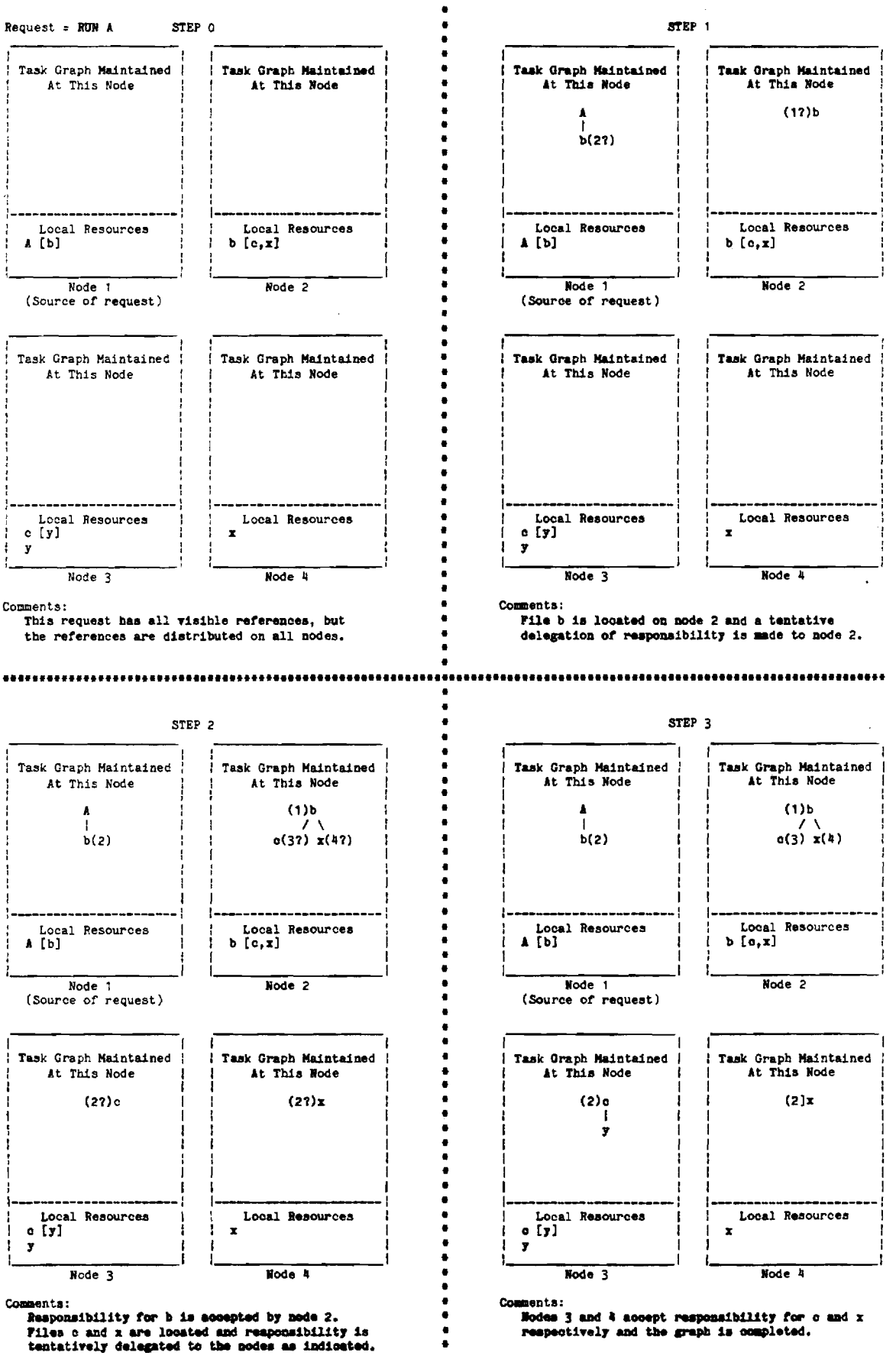


Figure 29. Example 9

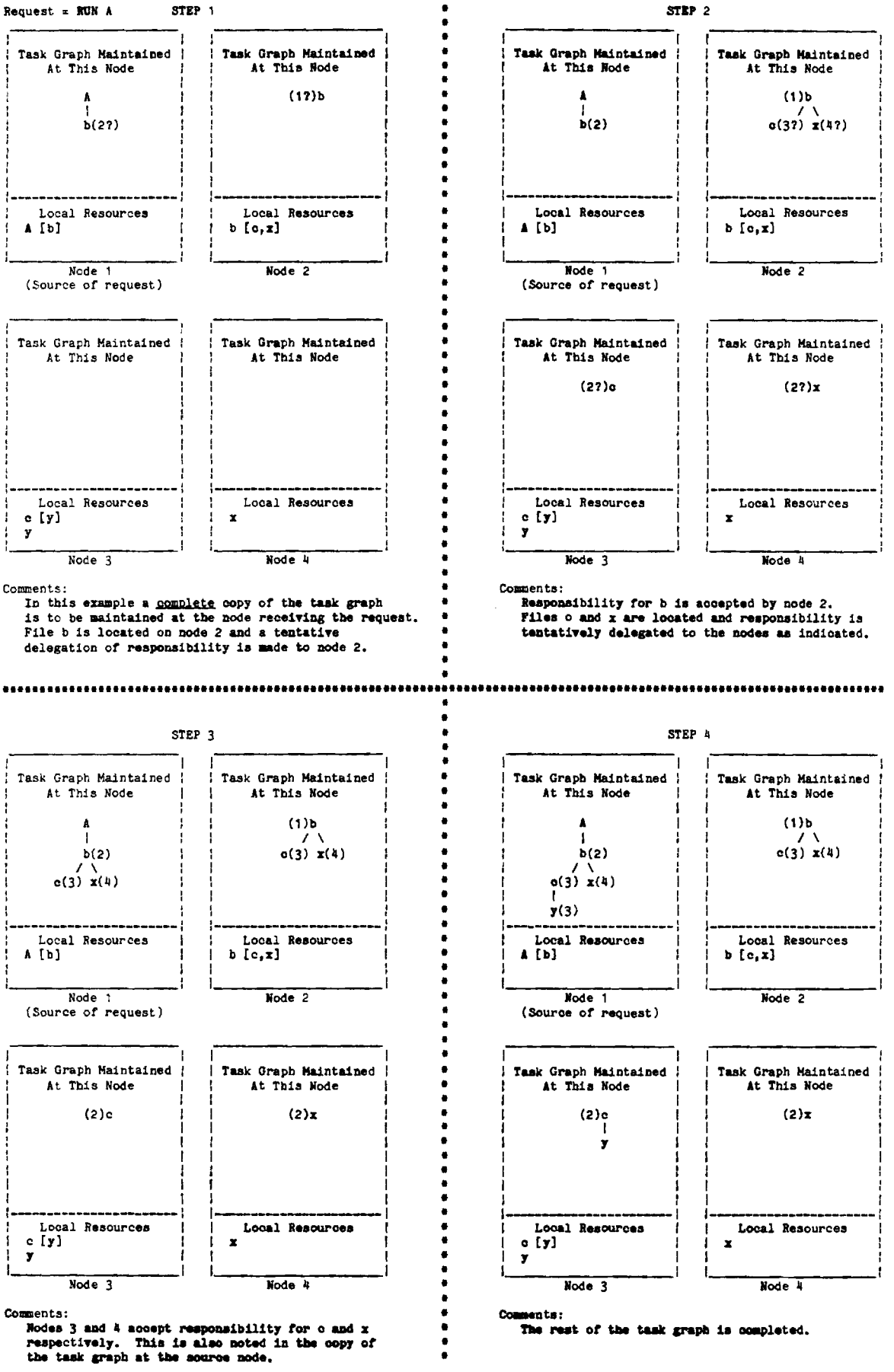
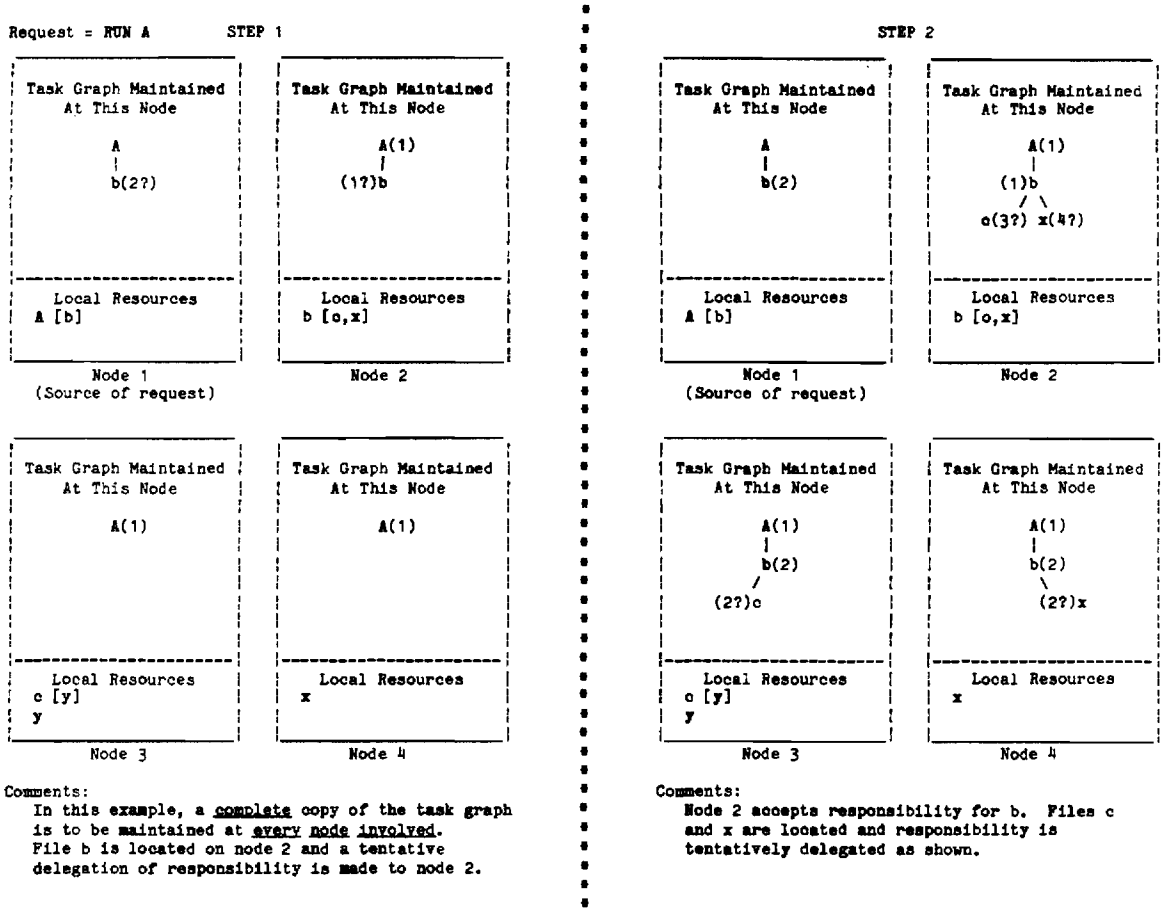


Figure 30. Example 10



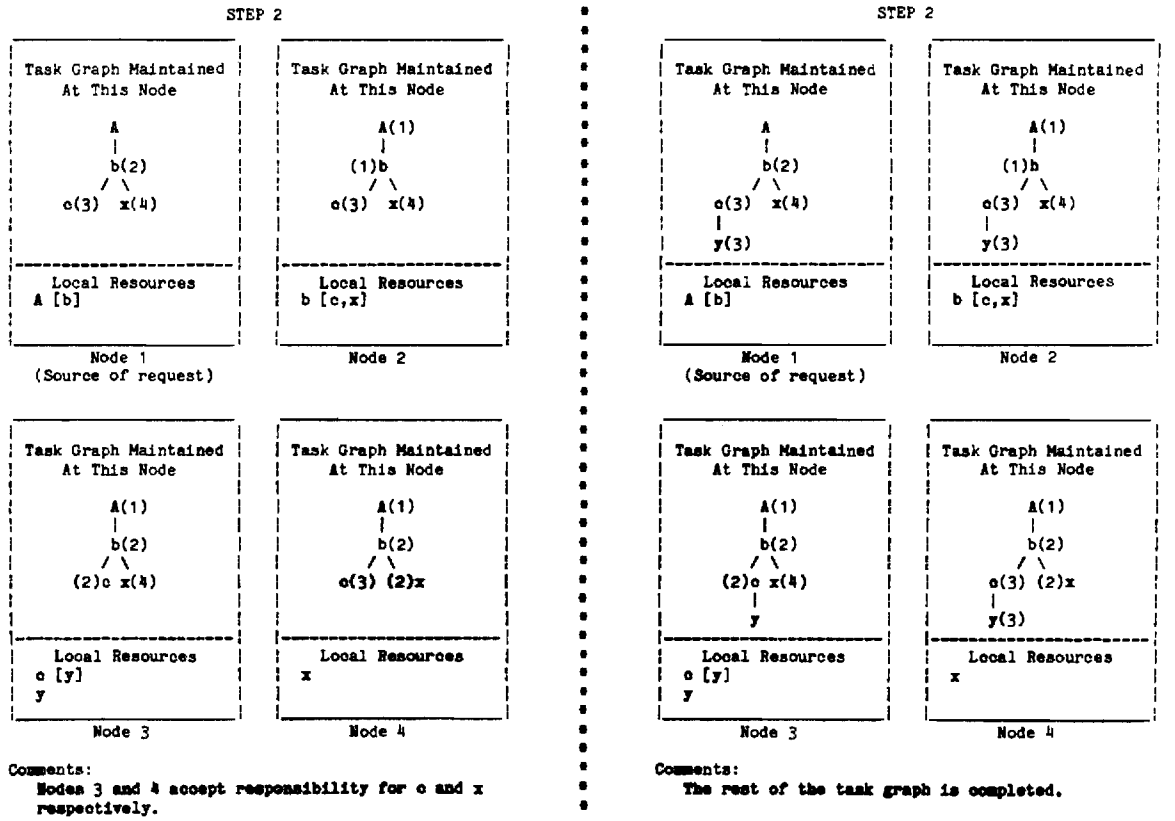


Figure 31. Example 11

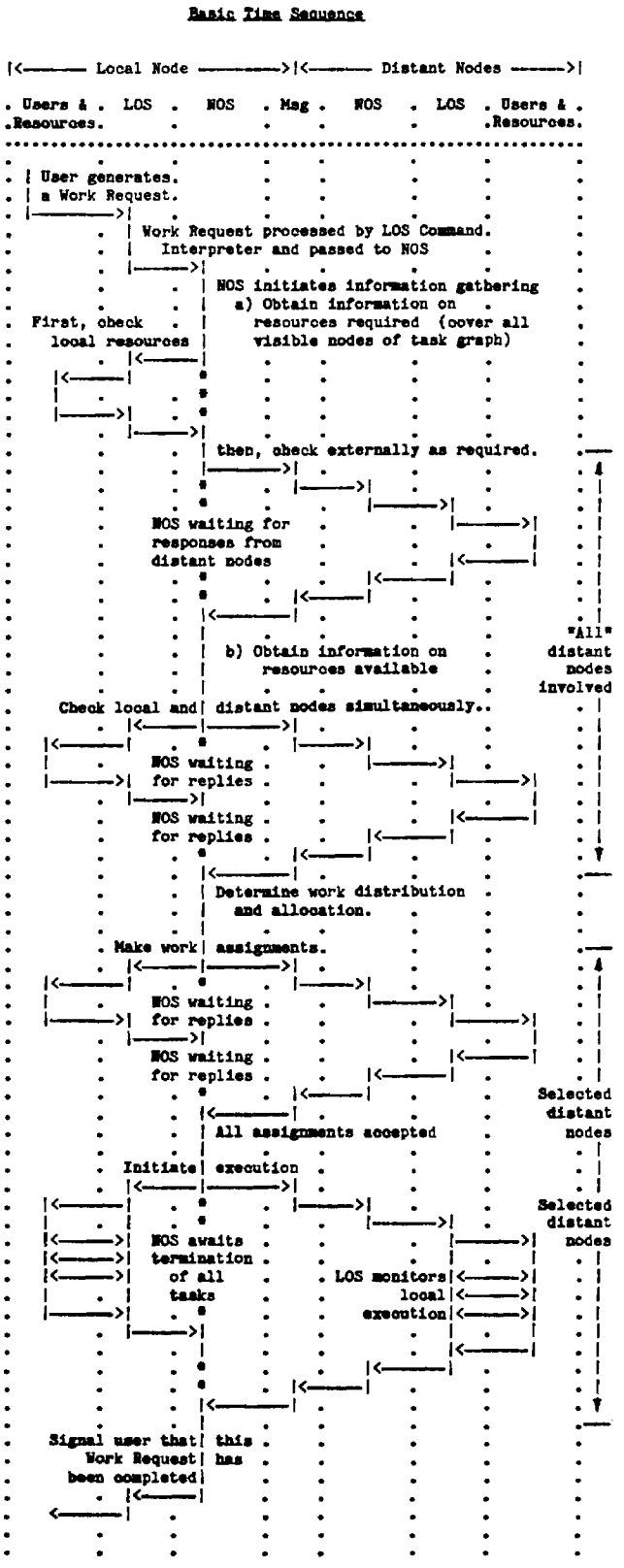


Figure 32. Basic Steps in Work Request Processing

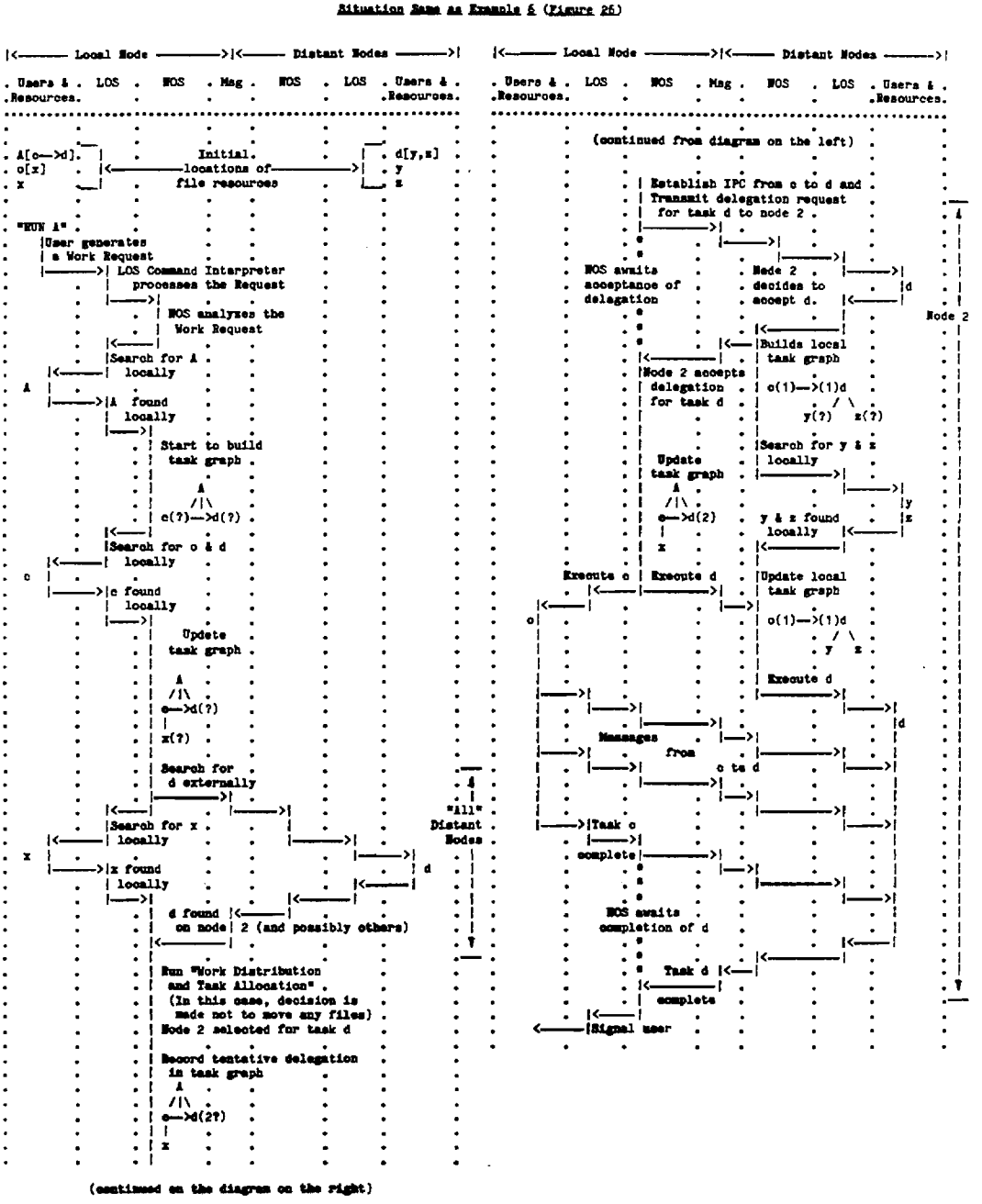


Figure 33. An Example of Work Request Processing

SECTION 8

MODELS OF CONTROL

In this section, we demonstrate how both existing and proposed models of control fit into the classification scheme described in Section 7. With the exception of the first model, these controls are designed to service work requests that specify multiple concurrent communicating processes. The first model considers work requests that involve only a single process.

8.1 ARAMIS

A decentralized operating system model for the ARAMIS Distributed Computer System is described in [Caba79a,b]. A brief outline of how this model fits into the classification scheme of Section 7 is provided by Table 3.

8.1.1 Architecture

The ARAMIS Distributed Computer System consists of two types of machines, hosts and managers. Users are connected to hosts which in turn are connected to managers. The managers are connected to each other in a virtual ring. Execution of work requests is provided by the hosts while control decisions are made by the managers.

8.1.2 Work Requests

This system is designed to handle a work request that is less sophisticated than those handled by the other systms described in this section. The work request must specify only a single process or task and the list of resources (sharable and nonsharable) required by that task.

8.1.3 The Control Model

Control of the system is accomplished through the managers. Each manager maintains a data structure called the resource state table (RST) which contains state information for every resource available on the system. To insure that these redundant copies remain consistent, two vectors are utilized. The control vector (CV) cycles around the virtual ring. Only the manager possessing the CV is permitted to allocate and deallocate resources. Upon completing this work, a manager can pass the CV along. In addition, modifications made to the RST (information describing the allocation and deallocation of files) are passed along to the other managers on the virtual ring in the form of an update vector (UPV).

Table 3. The Decentralized Control Model of the ARAMIS Distributed Computer System

TASK GRAPH CONSTRUCTION:

Who builds the task graph?
A manager on each node builds the task graph for the work requests arriving at that node.

What is the nature of the task graph?
A single structure.

Where is the task graph stored?
On the node initially receiving and analyzing the work request and the node where execution of the task occurs.

When is the task graph built?
Completely prior to execution.

RESOURCE AVAILABILITY INFORMATION:

Who maintains this information?
All nodes maintain common information.

Where is the information maintained?
In multiple redundant copies.

ALLOCATION OF RESOURCES:

How is concurrency control provided?
Resources are locked before the work distribution decision is made.

PROCESS INITIATION:

How is responsibility distributed?
Each node has a manager. The node initially receiving and analyzing the work request retains enough information to restart the task if the execution node dies.

How is refusal of a request to execute a process by a node handled?
This possibility is not discussed.

PROCESS MONITORING:

What type of interprocess communication is provided?
IPC is not supported.

How are task graphs resulting from additional work requests handled?
Additional requests cannot occur.

When a work request arrives at a host, it is passed along to the local manager to which the host is connected. This manager is in charge of resource allocation and task routing. It first identifies the resources that are needed and allocates sharable resources. After the CV has arrived and various algorithms insuring mutual exclusion and the prevention of deadlocks have been executed, the nonsharable resources are allocated. Next the optimal site for execution of the task is determined taking into account the burden various choices place on the communication system. Finally, the information concerning the allocation of resources is transmitted in the form of a UPV, and the information describing the task routing is sent to the hosts needing the information.

8.1.4 Conclusion

This model represents a simplified approach to the control problem. All nodes are provided with a complete global view of the system via their copy of the RST. Modifications to the state are carefully controlled by permitting only one manager at a time to change this information. The capability to perform modifications on the RST is passed around the virtual ring in the form of the CV.

8.2 MEDUSA

Medusa [Oust80a,b] is a distributed operating system for the Carnegie-Mellon Cm* multimicroprocessor. This system differs from an FDPS in that it allows multiple nodes to share primary memory. Table 4 describes how this control model fits into the classification scheme of Section 7.

8.2.1 Architecture

Cm* consists of a number of relatively independent processors or computer modules (Cm) and a number of communication controllers (Kmap). The Cm's are arranged in clusters with a Kmap presiding over each cluster. A switch, Slocal, connects a Cm with the interprocessor communication structure. Each Slocal contains tables that allow it to decide on each memory reference whether to access local memory or pass the reference along to the Kmap to locate the desired information in either the local cluster or a distant cluster. Thus, any processor can access the memory of any other processor. It must be kept in mind, though, that a substantial time delay results from accessing the memory of distant processors.

Table 4. The Medusa Control Model

TASK GRAPH CONSTRUCTION:

Who builds the task graph?
The node containing an activation of the task force manager.

What is the nature of the task graph?
Multiple structures (the task force control block is stored in the SDL and the activity control block is stored in the PDLs).

Where is the task graph stored?
Multiple nodes.

When is the task graph built?
Completely prior to execution.

RESOURCE AVAILABILITY INFORMATION:

Who maintains this information?
A number of utilities each realized as a task force.

Where is the information maintained?
In a shared data structure.

ALLOCATION OF RESOURCES:

How is concurrency control provided?
By means of locks.

PROCESS INITIATION:

How is responsibility distributed?
The task force manager keeps overall control, but other special managers are available to provide specific services.

How is refusal of a request to execute a process by a node handled?
This is not discussed in the literature.

PROCESS MONITORING:

What type of interprocess communication is provided?
Unsynchronized communication.

How are task graphs resulting from additional work requests handled?
It is not clear if additional work can be requested.

8.2.2 Work Requests

Work requests are used to describe task forces. A task force consists of a number of relatively independent communicating processes capable of concurrent execution that are working toward the solution of some task. Interprocess communication is accomplished via pipes which differ slightly from those found in UNIX [Ritc78]. There are two unique features found in these pipes: 1) they insure that only whole messages are read, and 2) they identify the sender of the message to the receiver.

In addition to processes and pipes, a task force contains a shared descriptor list (SDL) and a number of private descriptor lists (PDL). These structures contain descriptors which are basically capabilities for certain system objects. There is only one SDL per task force. This provides access to objects that are shared among all processes of a task force. For each process, there is a PDL which provides access to private objects. Thus, the significant feature of the task force concept is the capability to directly share objects by means of the SDL.

8.2.3 The Control Model

The distributed control is composed of a series of five utilities each of which is implemented as a task force. The five utilities are as follows:

1. Memory Manager: allocates primary memory and aids the Kmap in descriptor list manipulation.
2. File System: acts as a controller for all I/O devices of the system and implements a hierarchical file system.
3. Task Force Manager: creates, schedules, and deletes task forces and the processes that comprise task forces.
4. Exception Reporter: communicates information about unusual occurrences to those processes that need to know this information.
5. Debugger/Tracer: holds symbol table and performance measurement information for all utilities and provides facilities for on-line debugging of the system and gathering of performance data.

Communication between user processes and utilities is accomplished by means of pipes. There is one pipe for each utility. Access to these pipes is provided by the utility descriptor list (UDL) which is present on all nodes. A process utilizes this structure to locate the proper pipe into which a message for a particular utility is to be placed.

8.2.4 Conclusion

Medusa introduces two features that are pertinent to this discussion. These are the concept of a task force and the concept of sharing primary memory. A task force provides concurrent communicating processes to solve a common task. In addition to communicating by means of messages, processes are permitted to share data. The idea of shared memory is also seen in the hardware by the ability to directly reference memory on distant processors.

8.3 CNET

CNET [Smit79, Smit80] is a distributed problem solver consisting of a collection of loosely coupled knowledge sources located on a number of distinct processors. Table 5 depicts how this model fits into the classification scheme of Section 7.

8.3.1 Architecture

The system is intended for use on a network of loosely coupled asynchronous processors. Communication between nodes is realized through broadcast messages.

8.3.2 Work Requests

Applications for CNET can potentially take the form of cooperating processes. An individual work request specifies the work that must be accomplished. Depending upon decisions of the control, a task may be divided into subtasks, and the subtasks may be further divided.

8.3.3 The Control Model

CNET utilizes a hierarchical form of control for each task. At the top level is the manager for the task that is described in the original work request. This manager attempts to find a suitable contractor to execute the task. This is accomplished by means of a negotiation that begins with a message from the manager. This message can take the form of a general broadcast, a limited broadcast, or a point-to-point announcement. The contents of the message include an eligibility specification (a list of criteria required of a node to execute the task), a task abstraction (a brief description of the task), a bid specification (describes the expected form of a bid from a possible contractor), and an expiration time (describes the time period that the announcement is valid). A general broadcast is utilized when the manager has no knowledge concerning the nodes capable of executing the task. A limited

Table 5. The CNET Control Model

TASK GRAPH CONSTRUCTION:

Who builds the task graph?
Multiple nodes.

What is the nature of the task graph?
Multiple structures each consisting of a subgraph.

Where is the task graph stored?
Multiple nodes.

When is the task graph built?
Piecemeal.

RESOURCE AVAILABILITY INFORMATION:

Who maintains this information?
Each node maintains information about its own resources.

Where is the information maintained?
Separate pieces of information concerning a particular resource type may be kept on different nodes.

ALLOCATION OF RESOURCES:

How is concurrency control provided?
Resources are locked before the work distribution decision is made.

PROCESS INITIATION:

How is responsibility distributed?
There is a hierarchy of responsibility.

How is refusal of a request to execute a process by a node handled?
Once a contract is made it is binding.

PROCESS MONITORING:

What type of interprocess communication is provided?
Not specified.

How are task graphs resulting from additional work requests handled?
The new task graph is kept separate.

broadcast can be utilized when the manager knows a specific group of nodes is capable of executing the task. Finally, a point-to-point announcement is made when the manager knows about the availability of a single suitable node. This knowledge is obtained from idle nodes that broadcast messages indicating their availability.

The manager sends these messages and waits for the arrival of bids from possible contractors. When the bids arrive, they are examined in order to determine a choice for the task assignment. All bids are binding so the manager can make a choice with confidence that a chosen node will accept the task. Once a node is chosen, the contract is awarded and the chosen node becomes known as a contractor. The contractor may further divide the task and utilize other contractors for the various pieces. Thus, a node can act both as a manager and a contractor.

A contractor provides the manager with reports that contain information concerning partial execution (interim report) or completion (final report). A report contains a result description that specifies execution results. A manager has complete authority over a contractor and thus may terminate contracts at any time with a termination message. This terminates execution of a contract and all outstanding subcontracts.

8.3.4 Conclusion

CNET utilizes a hierarchical control scheme with a manager supervising the work of possibly multiple contractors working to solve a given task. A manager locates contractors by broadcasting an announcement for bids. It then waits for the bids from the contractors to arrive. After this negotiation phase, a bid is accepted, a contract is awarded, and execution of the task is begun. The manager can terminate execution of a task at any time and is the recipient of interim and final reports from the contractors.

8.4 THE XFDPS SERIES OF MODELS

In Section 7, a list of design alternatives for an FDPS executive control is presented (See Table 2). The rest of this section is devoted to the presentation of a series of control models designed by this research team by choosing among these alternatives. Each of the models is referred to as XFDPS.i where i is an identifying numeral. It is neither possible nor practical to present all possible models for an FDPS executive control.

Therefore, only a few models are investigated. The models were chosen by selecting a collection of design alternatives which were both logical and provided significant distinction among the various models.

The models are described in such a manner as to give the reader a feeling for the overall control strategy. A more complete comparison of the models can be obtained through tables 6 through 8 which contain a list of design alternatives for each model.

8.4.1 Architecture

An FDPS is composed of a multiplicity of independent processors physically connected by a network providing communication by means of a two-party protocol. There is no sharing of primary memory, and, thus, the processors are considered to be loosely coupled. The processors operate in an autonomous but cooperative manner. Therefore, it is the responsibility of the control to insure that there is a unification of operation in the system.

8.4.2 Work Requests

Work requests describe concurrent communicating processes and are assumed to provide the functionality available with the command language described in Figure 10.

8.4.3 XFDPS.1

The XFDPS.1 model [Sapo80] (see Table 6 for a characterization of this model and Figure 34 for a view of the model's components) is a distributed and decentralized control model that is designed to shield the user from the system. In other words, it provides the system transparency that is fundamental to the FDPS definition. It is designed to encapsulate each processor's local operating system as advocated by Kimbleton [Kimb76]. This is the meta-system approach to implementing distributed operating systems discussed above and has been practiced in several systems including ADAPT [Peeb80]. The XFDPS.1 model is composed of a set of cooperating processes called managers and is similar in this respect to Medusa [Oust80] and ADAPT [Peeb80]. Each manager is designed to control a subset of the system's resources (logical and physical).

Each manager requires reliable message communication with the other managers in order to perform its responsibilities. The XFDPS.1 model does not assume the presence of any particular interconnection of processors or for that matter any particular technique of message communication. This means

Table 6. The XFDPS.1 Control Model

TASK GRAPH CONSTRUCTION:

Who builds the task graph?
The source node.

What is the nature of the task graph?
Multiple structures each consisting of a subgraph with one copy of the complete task graph.

Where is the task graph stored?
Multiple nodes.

When is the task graph built?
Completely prior to execution.

RESOURCE AVAILABILITY INFORMATION:

Who maintains this information?
Each node maintains information about its own resources.

Where is the information maintained?
At the node which contains the resource.

ALLOCATION OF RESOURCES:

How is concurrency control provided?
Reservations are used prior to a work distribution decision and then allocated by a lock.

PROCESS INITIATION:

How is responsibility distributed?
There is a hierarchy of responsibility.

How is refusal of a request to execute a process by a node handled?
After repeated attempts, the request is abandoned.

PROCESS MONITORING:

What type of interprocess communication is provided?
Unsynchronized communication.

How are task graphs resulting from additional work requests handled?
The new task graph is kept separate.

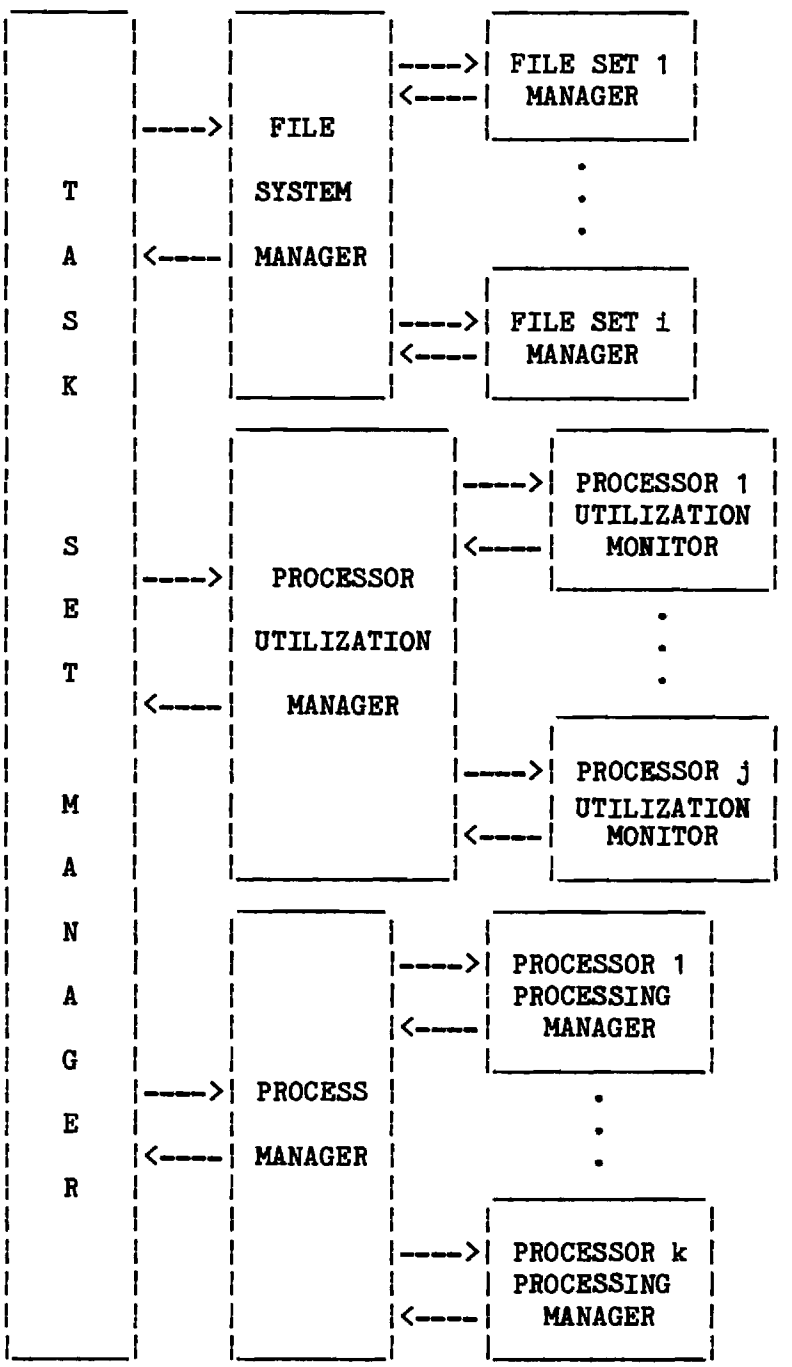


Figure 34. The XFDPS.1 Control Model

that the model is applicable to systems that are interconnected in a variety of ways including loops, stars, regular networks, irregular networks, or fully interconnected networks [Ande75] and utilizing various message communication techniques including the ISO model [Bach78, Desj78] and Ethernet [Metc76].

The XFDPS.1 model is composed of several types of processes called managers which are responsible for various aspects of the control problem. These managers include the Task Set Manager, the File System Manager, the Processor Utilization Manager, and the Process Manager.

8.4.3.1 Task Set Manager

The Task Set Manager is responsible for handling work requests arriving from either users or active processes. A Task Set Manager is assigned to every work request. It must first identify the tasks comprising the Task Set which are needed to satisfy the work request and then communicate with the File System Manager to obtain information concerning the availability of files. The Processor Utilization Manager is also consulted in order to determine the relative utilization of the processors. Using the information acquired in this manner, a work allocation decision is made that results in the assignment of tasks to processors. This decision involves an optimization problem similar in many respects to that discussed by Morgan [Morg77].

The second phase of the Task Set Manager's responsibility concerns carrying out the decision arrived at in the first phase. This again involves communication with the File System Manager to allocate needed files and to deallocate these files when they are no longer needed. In addition, communication is required with the Process Manager which activates the processes and observes when these processes have terminated. The last act of the Task Set Manager is to inform the original requester as to the completion status of the request. In doing so it will either indicate that it was successful in completing the request or provide a description concerning why the request could not be completed.

8.4.3.2 File System Manager

The File System Manager is responsible for maintaining the file system for the entire FDPS. Instances of the File System Manager are found on all processors. Management of the file system is achieved through communication among these instances of the File System Manager.

The implication of this design is that several requests to the file system can be acted upon simultaneously provided these requests arrive at different processors. These requests may either elicit availability information or ask that the file status information be updated (i.e., making a reservation, placing a lock, or releasing a lock on a file). This simultaneity is in marked contrast to the resource allocation found in the ARAMIS Distributed Computer System [Caba79a,b] in which all nodes possess a Resource State Table containing the state of all resources in the system. This system only permits resource allocation by at most one node at any one time.

In the XFDPS.1 model, the file system is divided into several disjoint sets. The design of the control does not restrict how this division is realized. For example, these sets can be defined by processor boundaries. For each set, there is a separate manager called a File Set Manager. In order to perform its management duties, the File System Manager must communicate with each File Set Manager.

The File System Manager handles three types of requests, all originating from the Task Set Manager. The first type of request is for availability information concerning a collection of files. The File System Manager converts this request into a series of requests concerning individual files and presents these requests to the File Set Managers. The File System Manager waits for responses from all File Set Managers before returning its response. A File Set Manager will return an indication of the file's availability. If a file is available, the File Set Manager will reserve the file for the Task Set from which the request originated. This reservation remains effective for a limited period of time, and it is the responsibility of the Task Set Manager to confirm the reservation before its effectiveness has expired.

The second request that can be made to the File System Manager concerns the allocation of a series of files. Again this request is converted into a number of requests concerning the reservations of individual files and is sent to specific File Set Managers which in turn perform the necessary locking of the files.

Finally, the File System Manager can receive requests for the deallocation of files. These requests are handled in a manner similar to allocation requests and result in the release of locks or reservations on specific files.

8.4.3.3 Processor Utilization Manager

Another type of process found in the control is the Process Utilization Manager. Instances of this manager are replicated on all processors. The main function of the Process Utilization Manager is the maintenance of a data base of processor utilization information for the processors comprising the FDPS. The information in this data base is not intended to be complete and accurate but rather is designed to provide the work assignment algorithm in the Task Manager with an estimate of the utilization of the processors in the system.

The Processor Utilization Manager obtains the information needed to update its data base from periodic messages directed to it from Processor Utilization Monitors located on each processor. These processes monitor the utilization of the processor in which they are located and issue periodic messages reporting their findings. If a Processor Utilization Manager does not receive a report from a Processor Utilization Monitor within a certain period of time, a message from the Manager is sent to the Monitor asking for an immediate response concerning the processor's state. If a response to this request is not received within a certain time period, it is assumed the processor is lost, and the Processor Utilization Manager updates its data base to reflect this. This will prevent the Task Set Manager from attempting to assign processes to a processor that has apparently been lost.

8.4.3.4 Process Manager

The last process type found in the control is the Process Manager. A Process Manager is activated for each Task Set Manager. This process accepts requests from the Task Set Manager for the activation of processes for the Task Set. The Process Manager identifies which processors are to receive processes. It then issues requests to Processing Managers on each processor. Each Processing Manager is responsible for controlling the processes assigned to its processor.

In addition to assigning processes and waiting for the notification of their termination, the Process Manager is responsible for providing interprocess communication between executing processes. In this model, interprocess communication is provided by means of ports [Balz71, Have78, Suns77, Zuck77]. A port provides a common location where communicating processes can either send or fetch messages without knowing about the other's

location. Buffer space is also required in order to allow the communicating processes to operate as independently as possible. This type of interprocess communication is similar to the stream communication utilized in TRIX [Ward80]. The Process Manager must therefore decide where a buffer for the port resides and then provide the necessary linkages within the communicating processes in order for them to address the port.

8.4.3.5 Conclusion

The fundamental philosophy of the XFDPS.1 model is that the control over logical and physical resources must be distributed among various processes or managers. The reason for taking this approach is to provide better utilization of system resources by making use of the inherent parallelism found in distributed processing systems.

8.4.4 XFDPS.2

XFDPS.2 is a variation of model XFDPS.1. The main difference between the two models exists in the technique used to construct the task graph. A complete outline of the characteristics of XFDPS.2 is found in Table 7.

The construction of task graphs in XFDPS.2 is performed by multiple nodes resulting in a task graph that consists of multiple structures each of which is a subgraph of the complete task graph. The overall strategy works as follows. After a work request arrives at a particular node, work on constructing a task graph is begun. When a node is chosen to perform part of a task graph, responsibility for that portion of the task graph is given to a control component on that node. This component will maintain that portion of the task graph and in so doing may also choose other nodes to perform part of the work that the subgraph represents.

Thus, there are two main differences between XFDPS.2 and XFDPS.1: 1) the task graph is not maintained in one location but rather on multiple nodes, and 2) this construction is performed in a piecemeal fashion in XFDPS.2. This means that the components of XFDPS.2 possess greater independence than those of XFDPS.1.

8.4.5 XFDPS.3

XFDPS.3 (see Table 8) is a variation on the XFDPS.2 model. In this case, the difference exists in the maintenance of resource availability information. In both XFDPS.1 and XFDPS.2, each physical node maintains information about its own resources. XFDPS.3, though, utilizes the approach

Table 7. The XFDPS.2 Control Model

TASK GRAPH CONSTRUCTION:

Who builds the task graph?
Multiple nodes.

What is the nature of the task graph?
Multiple structures each consisting of a subgraph.

Where is the task graph stored?
Multiple nodes.

When is the task graph built?
Piecemeal.

RESOURCE AVAILABILITY INFORMATION:

Who maintains this information?
Each node maintains information about its own resources.

Where is the information maintained?
Separate pieces of information concerning a particular resource type may be kept on differentt nodes.

ALLOCATION OF RESOURCES:

How is concurrency control provided?
Reservations are used prior to a work distribution decision and then allocated by a lock.

PROCESS INITIATION:

How is responsibility distributed?
There is a hierarchy of responsibility.

How is refusal of a request to execute a process by a node handled?
After repeated attempts, the request is abandoned.

PROCESS MONITORING:

What type of interprocess communication is provided?
Unsynchronized communication.

How are task graphs resulting from additional work requests handled?
The new task graph is kept separate.

Table 8. The XFDPS.3 Control Model

TASK GRAPH CONSTRUCTION:

Who builds the task graph?
Multiple nodes.

What is the nature of the task graph?
Multiple structures each consisting of a subgraph.

Where is the task graph stored?
Multiple nodes.

When is the task graph built?
Piecemeal.

RESOURCE AVAILABILITY INFORMATION:

Who maintains this information?
Components for each type of resource.

Where is the information maintained?
Information concerning a particular resource type is kept on a single node.

ALLOCATION OF RESOURCES:

How is concurrency control provided?
Reservations are used prior to a work distribution decision and then allocated by a lock.

PROCESS INITIATION:

How is responsibility distributed?
There is a hierarchy of responsibility.

How is refusal of a request to execute a process by a node handled?
After repeated attempts, the request is abandoned.

PROCESS MONITORING:

What type of interprocess communication is provided?
Unsynchronized communication.

How are task graphs resulting from additional work requests handled?
The new task graph is kept separate.

taken in Medusa which assigns a control component to each type of resource and maintains information concerning a particular type of resource in a single location.

Thus, when resource availability information is required, a resource needs allocation, or a resource needs deallocation, it is only necessary to determine the type of the resource in order to determine the proper control component to perform the desired operation. This is in contrast to XFDPS.1 and XFDPS.2 both of which require a search for the correct component.

SECTION 9

THE EVALUATION OF THE MODELS

9.1 EVALUATION PLAN

As stated earlier in this report, it was planned from the initiation of this survey of control models that it would be followed immediately by an evaluation study of the various models identified or developed. It was also anticipated that this evaluation would cover both the quantitative and qualitative aspects of the various models.

To support the quantitative evaluation of the various forms of system control, a distributed control model simulator is being developed.

9.2 EVALUATION CRITERIA

A number of evaluation criteria have already been identified. The tentative list is summarized in Table 9.

Table 9. Possible Evaluation Criteria for Distributed Control Models

RESOURCE UTILIZATION
- Memory Space Utilization
  - By the Control Algorithm
    - Complexity
    - Redundancy
  - By the Control Information
- Time
  - Local Processing Time
  - Communications Delays
  - Delays in Work Initiation
- Communication
  - Complexity
  - Quantity

PERFORMANCE
- Throughput
- Response Time
- Bottlenecks

SYSTEM FLEXIBILITY
- Reconfiguration Potential
- Modularity
  - Logical Complexity
  - Maintainability
  - Problem Partitioning and Algorithm Design

FAULT-TOLERANCE
- Detection
- Recovery
- Extent to Which Processed Work Can Be Recovered

PROTECTION
- Privacy
- Security

REFERENCES

Akin78 Akin, T. Allen, Flinn, Perry B., Forsyth, Daniel H., "A Prototype for an Advanced Command Language," Proceedings of the 16th Annual Southeastern Regional ACM Conference (April, 1978): 96-102.

Ande75 Anderson, George A., and Jensen, E. Douglas., "Computer Interconnection Structures: Taxonomy, Characteristics, and Examples," Computing Surveys 4 (December, 1975): 197-213.

Bach78 Bachman, Charles, and Canepa, Mike, "The Session Control Layer of an Open System Interconnection," COMPCON Fall 78 (September, 1978): 150-156.

Balz71 Balzer, R. M., "PORTS - A Method for Dynamic Interprogram Communication and Job Control," AFIPS Conference Proceedings 38 (1971 Spring Joint Computer Conference): 485-489.

Brin78 Brinch Hansen, Per, "Distributed Processes: A Concurrent Programming Concept," Communications of the ACM 21 (November, 1978): 934-941.

Caba79a Cabanel, J. P., Marouane, M. N., Besbes, R., Sazbon, R. D., and Diarra, A. K., "A Decentralized OS Model for ARAMIS Distributed Computer System," Proceedings of the First International Conference on Distributed Computing Systems (October, 1979): 529-535.

Caba79b Cabanel, J. P., Sazbon, R. D., Diarra, A. K., Marouane, M. N., and Besbes, R., "A Decentralized Control Method in a Distributed System," Proceedings of the First International Conference on Distributed Computing Systems (October, 1979): 651-659.

Clar80 Clark, David D., and Svobodova, Liba, "Design of Distributed Systems Supporting Local Autonomy," COMPCON Spring 80 (February, 1980): 438-444.

Cook80 Cook, Robert P., "The STARMOD Distributed Programming System," COMPCON Fall 80 (September, 1980): 729-735.

Davi79 Davies, D. W., Barber, D. L. A., Price, W. L., and Solomonides, C. M., Computer Networks and Their Protocols, John Wiley and Sons, 1979.

Denn78 Denning, Peter J., "Operating Systems Principles for Data Flow Networks," Computer (July, 1978): 86-96.

Desj78 desJardins, Richard, and White, George, "ANSI Reference Model for Distributed Systems," COMPCON Fall 78 (September, 1978): 144-149.

Ensl74 Enslow, Philip H., Jr. (ed.), Multiprocessors and Parallel Processing, New York: John Wiley and Sons, 1974.

Ensl78 Enslow, Philip H., Jr., "What is a 'Distributed' Data Processing System?" Computer (January, 1978): 13-21.

Farb73 Farber, D. J., Feldman, J., Heinrich, F. R., Hopwood, M. D., Larson, K. C., Loomis, D. C., and Rowe, L. A., "The Distributed Computing System," COMPCON Spring 73 (February, 1973): 31-34.

Feld79 Feldman, J. A., "High Level Programming for Distributed Computing," Communications of the ACM 22 (June, 1979): 353-368.

Garc79 Garcia-Molina, H., "Performance Comparison of Update Algorithms for Distributed Databases, Crash Recovery in the Centralized Locking Algorithm," Progress Report No. 7, Stanford University, 1979.

Have78 Haverty, J. F., and Rettberg, R. D., "Inter-process Communications for a Server in UNIX," COMPCON Fall 78 (September, 1978): 312-315.

Hoar78 Hoare, C. A. R., "Communicating Sequential Processes," Communications of the ACM 21 (August, 1978): 666-677.

Hopp79 Hopper, K., Kugler, H. J., and Unger, C., "Abstract Machines Modelling Network Control Systems," Operating Systems Review 13 (January, 1979): 10-24.

Jens78 Jensen, E. Douglas., "The Honeywell Experimental Distributed Processor - An Overview," Computer (January, 1978): 28-38.

Kimb76 Kimbleton, Stephen R., and Mandell, Richard L., "A Perspective on Network Operating Systems," AFIPS Conference Proceedings 45 (1976 National Computer Conference): 551-559.

Lein58 Leiner, A. L., and Weinberger, A., "PILOT, the NBS Multicomputer System," Proceedings of the Eastern Joint Computer Conference (1958): 71-75.

Macc80 Maccabe, Aurthur B., and Leblanc, Richard J., "A Language Model for Fully Distributed Systems," COMPCON Fall 80 (September, 1980): 723-728.

Metc76 Metcalfe, R. M., and Boggs, D. R., "Ethernet - Distributed Packet Switching for Local Computer Networks," Communications of the ACM 19 (July, 1976): 395-404.

Morg77 Morgan, Howard L., and Levin, K. Dan, "Optimal Program and Data Locations in Computer Networks," Communications of the ACM 20 (May, 1977): 315-322.

Nels78 Nelson, David L., and Gordon, Robert L., "Computer Cells - A Network Architecture for Data Flow Computing," COMPCON Fall 78 (September, 1978): 296-301.

Oust80 Ousterhout, John K., "Partitioning and Cooperation in a Distributed Multiprocessor Operating System: Medusa," Ph.D. Thesis, Carnegie-Mellon University, April, 1980.

Oust80 Ousterhout, John K., Scelza, Donald A., and Sindhu, Pradeep S., "Medusa: An Experiment in Distributed Operating System Structure," Communications of the ACM 23 (February, 1980): 92-105.

Peeb80 Peebles, Richard, and Dopirak, Thomas, "ADAPT: A Guest System," COMPCON Spring 80 (February, 1980): 445-454.

Ritc78 Ritchie, D. M., and Thompson, K., "The UNIX Time-Sharing System," The Bell System Technical Journal 57 (July-August, 1978): 1905-1929.

Sapo80 Saponas, Timothy G., and Crews, Phillip L., "A Model for Decentralized Control in a Fully Distributed Processing System," COMPCON Fall 80 (September, 1980): 307-312.

Smit79 Smith, Reid G., "The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver," Proceedings of the 1st International Conference on Distributed Computing (October, 1979): 185-192.

Smit80 Smith, Reid G., "The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver," IEEE Transactions on Computers C-29 (December, 1980): 1104-1113.

Suns77 Sunshine, Carl, "Interprocess Communication Extensions for the UNIX Operating System: I. Design Considerations," Rand Technical Report R-2064/1-AF, June 1977.

Thom78 Thomas, Robert H., Schantz, Richard E., and Forsdick, Harry C., "Network Operating Systems," Bolt Beranek and Newman Report No. 3796 (March, 1978).

Ward80 Ward, Stephen A., "TRIX: A Network-Oriented Operating System," COMPCON Spring 80 (February, 1980): 344-349.

Zuck77 Zucker, Steven, "Interprocess Communication Extensions for the UNIX Operating System: II. Implementation," Rand Technical Report R-2064/2-AF, June, 1977.