1

4

OF/DE

24226

National Library
of CanadaBibliothèque nationale
du CanadaCANADIAN THESES
ON MICROFICHETHÈSES CANADIENNES
SUR MICROFICHE

NAME OF AUTHOR/NOM DE L'AUTEUR Samy Abd-Elhamid Mahmoud

TITLE OF THESIS/TITRE DE LA THÈSE Resource Allocation and File Access Control
in Distributed Information Networks.

UNIVERSITY/UNIVERSITÉ Carleton University

DEGREE FOR WHICH THESIS WAS PRESENTED/
GRADE POUR LEQUEL CETTE THÈSE FUT PRÉSENTÉE Ph.D.

YEAR THIS DEGREE CONFERRED/ANNÉE D'OBTENTION DE CE DEGRÉ 1975

NAME OF SUPERVISOR/NOM DU DIRECTEUR DE THÈSE Dr. J.S. Riordon & Dr. J. de Mercado (Co-Supervisors)

Permission is hereby granted to the NATIONAL LIBRARY OF
CANADA to microfilm this thesis and to lend or sell copies
of the film.

*L'autorisation est, par la présente, accordée à la BIBLIOTHÈ-
QUE NATIONALE DU CANADA de microfilmer cette thèse et
de prêter ou de vendre des exemplaires du film.*

The author reserves other publication rights, and neither the
thesis nor extensive extracts from it may be printed or other-
wise reproduced without the author's written permission.

*L'auteur se réserve les autres droits de publication; ni la
thèse ni de longs extraits de celle-ci ne doivent être imprimés
ou autrement reproduits sans l'autorisation écrite de l'auteur.*

DATED/DATE

29/1/75

SIGNED/SIGNÉ

S. O. Mahmoud

PERMANENT ADDRESS/RÉSIDENCE FIXE

784 Aylmer Avenue,
Ottawa, Ont.,
K1S 2Y5

RESOURCE ALLOCATION AND FILE ACCESS CONTROL
IN DISTRIBUTED INFORMATION NETWORKS

by

Samy A. Mahmoud

A thesis submitted to the Faculty of Graduate Studies
in partial fulfilment of the requirements for the
degree of Doctor of Philosophy
in
Electrical Engineering

Faculty of Engineering
Carleton University
Ottawa, Ontario
January 1975

The undersigned hereby recommend to the Faculty of Graduate Studies acceptance of this thesis submitted by Samy A. Mahmoud in partial fulfilment of the requirements for the degree of Doctor of Philosophy.

.....
Supervisor

.....
Chairman of the Division

.....
External Examiner

ABSTRACT

This research is concerned with resource allocation and file access control problems in information networks which offer distributed computing services.

A comprehensive network design approach is presented in which the problems of file allocation and link capacity assignment are examined simultaneously to achieve minimum cost, subject to network delay and file availability constraints. The resulting optimization problem falls into a class of nonlinear integer programming problems which can be solved by a nonlinear branch-and-bound integer programming technique. The solutions, while theoretically convergent, are computationally cumbersome even for small sized problems. A new heuristic algorithm is developed, based on a decomposition technique which greatly reduces the computational complexity of the problem. Numerical results for a variety of network configurations indicate that the heuristic algorithm, while not theoretically convergent, yields practicable low cost solutions with substantial savings in computer processing time and storage requirements. Moreover, it is shown that this algorithm is capable of solving realistic network problems whose solution using current integer programming techniques is beyond the capacity of present computers.

The second problem addressed in this research is that of file access control. It is proposed that the operating system of each information processor will provide its local users with an automatic access service to file data at other locations, on an individual request basis. Such an approach, usually referred to as software controlled

access (as opposed to user controlled access) has several attractive features; the most obvious one is its simplicity from the users point of view, since in this mode the user need not be concerned with file location or separation. On the other hand, software controlled access has two serious problems, viz., job interference (deadlock situations), and the task of controlling access to files.

The application of three general abstract models for deadlock detection and recovery or deadlock prevention to the file access control problem is investigated. Three algorithms are derived based on these models. Any of the three algorithms can be used to construct the logical functions of file access control systems.

Two approaches to the design and implementation of a software controlled access system are presented in this study. In the first approach, information concerning file access status and active user processes is exchanged between processing nodes so that the control is distributed. In the second approach control resides in a designated node, which is referred to by all active user processes when requests for file access are initiated; control is thus centralized. Simulation shows that when perfect reliability is assumed, the centralized scheme is more efficient except when traffic is primarily local in nature at each node. In practice, however, a centralized controller is often undesirable, since failure of the central node causes system paralysis. A modified control scheme in which the centre of control can shift its location to adapt to failure of network elements is proposed. It is surmised that such an adaptive controller will prove superior in performance to either of the previous alternatives.

ACKNOWLEDGEMENT

The field of Distributed Information Networks was originally suggested to the author, as a source of practically useful and challenging problems, by Dr. J.B. Demercado, Department of Communications, Government of Canada, Ottawa. The author is grateful to Dr. Demercado for this suggestion, and especially for his advice and support throughout the thesis work.

The author would like to place on record his gratitude to his thesis supervisor, Dr. J.S. Riordon of the Division of Systems Engineering for his supervision, constant encouragement and many long and fruitful discussions during the course of this work.

The author is indebted to Mr. James Martin of the IBM Systems Research Institute, New York, for his valuable review and comments as an External Examiner of this thesis.

The author would like also to express his sincere thanks to Mrs. L.H. Ward, of Ottawa. Her kind and generous motherly treatment has provided the author with a wonderful environment for study and reflection throughout his residence in Ottawa.

The research described in this thesis was made possible by the National Research Council of Canada through the award of a Graduate Fellowship.

TABLE OF CONTENTS

	PAGE
CHAPTER 1 - INTRODUCTION	1
1.1 Distributed Data Bases and Information Networks	1
1.1.1 Computer-to-Computer Communication Networks	5
1.1.2 File Systems	9
1.1.3 Mode of Information Network Usage	11
1.2 Uses and Applications of Distributed Data Bases	13
1.3 Technical Problems Associated with Future Developments ..	17
1.4 Review of the Existing Literature	20
1.5 Objectives and Contributions of this Study	23
1.6 Thesis Organization	26
CHAPTER 2 - FILE SYSTEM AVAILABILITY	
2.1 Introduction	28
2.2 Node Reliability	29
2.3 Terminal Reliability	33
2.4 Time Dependent Terminal Reliability	40
CHAPTER 3 - TRAFFIC ROUTING AND MESSAGE DELAYS	49
3.1 General Network Concepts	49
3.2 Traffic Routing Strategies	51
3.3 Message Delays	56
3.4 Priority Service Queue	59
CHAPTER 4 - THE RESOURCE ALLOCATION MODEL	69
4.1 Introduction	69
4.2 The Cost Function	69
4.3 The Delay Constraint	72
4.4 The Reliability Constraint	76
4.5 Related Models	86
4.5.1 The Unconstrained File Allocation Model	86
4.5.2 File Allocations in Fully Connected Topologies with Delay Constraints	88
4.5.3 Capacity and Flow Assignment Models	89
4.6 File Security and Privacy Considerations	90

TABLE OF CONTENTS (continued)

CHAPTER 5 - BIVALENT PROGRAMMING TECHNIQUES	92
5.1 Introduction	92
5.2 Linear Integer Programming Techniques	95
5.3 Nonlinear Integer Programming Techniques	96
CHAPTER 6 - HEURISTIC SOLUTION	101
6.1 Introduction	101
6.2 Prologue to the Heuristic Algorithm	102
6.3 The Heuristic Algorithm	111
6.4 Example	122
CHAPTER 7 - COMPUTATIONAL EXPERIENCE	140
7.1 Computational Experience	141
7.2 Sensitivity of Optimal File Assignment to Changes in File Availability Requirements	161
7.3 Effect of File Demand Variations on Resource Allocation ..	167
7.4 Summary	170
CHAPTER 8 - OPERATIONAL ASPECTS OF DISTRIBUTED DATA BASES ...	176
8.1 Methods of Creating a Distributed Data Base	176
8.2 Associating Interactive Executions with the Distributed File Systems	179
8.3 The Distributed File Access Methods	181
8.3.1 Transmission of File Copies	181
8.3.2 Chaining of Jobs with Multi-Access Applications ...	182
8.3.3 Software Controlled Access Methods	185
8.3.4 Delayed and Synchronized Updating of Files with Multiple Copies	186
8.4 Management of Tasks and Resources in Distributed Information Networks	191
CHAPTER 9 - SOFTWARE CONTROLLED ACCESS TECHNIQUES	193
9.1 Distributed File Access Modes	194
9.2 The Maximum Limit (Claim) Model	194
9.3 The Demand Graph Model	204
9.3.1 Definitions and Notations	207
9.3.2 Feasibility and Safety in the Demand Graph Model ...	210
9.3.3 Application of the Demand Graph Model to the Dis- tributed File Problem	217

TABLE OF CONTENTS (continued)

9.4	Deadlock Detection and Recovery: Preemptive Scheme	224
9.5	Implementation of the Distributed Data Base Management Facility	238
9.5.1	The Centralized Control Approach	243
9.5.2	The Distributed Control Approach	246
9.6	The Protocol Problem	247
9.6.1	The Centralized Control Case	248
9.6.2	The Distributed Control Case	253
9.7	Simulation and Results	264
9.7.1	Performance Criteria	264
9.7.2	Simulation Model Parameters	265
9.7.3	The GPDS Model	268
9.7.4	Simulation Values	275
9.7.5	Simulation Results and Analysis	278
9.7.6	Reliability Consideration	292
CHAPTER 10 - CONCLUSIONS AND SUGGESTIONS FOR FURTHER RESEARCH		293
10.1	Thesis Contributions	293
10.2	Suggestions for Further Research	297
REFERENCES		299
APPENDIX A - DERIVATION OF TERMINAL FAILURE RATE FUNCTION ..		A-1
APPENDIX B - JOINT PROBABILITY DISTRIBUTION OF NUMBER OF MESSAGES IN QUEUE		B-1
APPENDIX C - RELATION OF FILE AVAILABILITY MEASURE a_{ij} TO ACTUAL FILE AVAILABILITY		C-1

LIST OF ILLUSTRATIONS

FIGURE		PAGE
1.1	Distributed Computing Network Configuration.....	3
2.1	Components of a Computer System.....	30
2.2	Reliability of Network Components.....	34
2.3	Example of a 4-node Network.....	45
2.4	Two-State Markov Chain.....	48
3.1	Line Utilization Factor vs. Ratio of Total Average Queuing Time With Priorities to Average Queuing Time Without Priorities.....	67
4.1	Traffic Flow Components.....	74
4.2	Star Topology.....	79
4.3	A 5-Node Network.....	81
6.1	Cost-Delay Function.....	108
6.2	A 4-Node Network.....	123
7.1	A 5-Node Network.....	142
7.2	A Proposed Topology For CANUNET.....	145
7.3	A 19-Node Network.....	149
7.4	Total Cost as a Function of Elapsed CPU Time.....	157
7.5	File Assignment Based on an Intuitive (ad hoc) Solution (Example 5).....	158
7.6	File Assignment Based on the Application of the Heuristic Algorithm.....	159
7.7	Sets of File Assignments Corresponding to Different Values of Availability Requirements.....	164
7.8	The Optimal Cost as a Function of the Availability Requirements.....	165
7.9	Effect of Changing the Availability Requirement on the Optimal Cost.....	166
7.10	A Procedure For Re-allocation of Resources to Meet Traffic Demand Variations.....	171
7.11	Automatic Loading of Traffic Routing Schemes.....	172
8.1	File Transmission vs. Traffic Volume.....	183
8.2	Job Chaining in Distributed Information Networks.	184
8.3	Deadlock State.....	187

LIST OF ILLUSTRATIONS (continued)

FIGURE		PAGE
8.4	Delayed Update.....	189
8.5	Asynchronous Updating of a 2-Copy File in A 3-Node Network.....	190
9.1	Flow Chart of Algorithm 1.....	203
9.2	Demand Graph Model.....	206
9.3	A Demand Graph Representing a System with One Resource.....	209
9.4	A Demand Graph Representing a System with Two Resources.....	215
9.5	An Example to Test the Adequacy of the Safeness Algorithm.....	216
9.6	Flow Chart of Algorithm 2.....	218
9.7	Demand Graph of System S_1	223
9.8	Demand Graph of System S_2	225
9.9	Deadlock State.....	239
9.10	Flow Chart of Algorithm 3.....	240
9.11	Details of A Network Node.....	244
9.12	Control Command Message Format.....	250
9.13	Example to Illustrate the File Access Process in the Central Control Approach.....	252
9.14(a)	A Deadlock Situation (Remote) Caused By Incompl- ete Information.....	257
9.14(b)	A Deadlock Situation (Local) Caused By Incomplete Information.....	258
9.14(c)	A Deadlock Situation Caused By Network Introduced Delays.....	259
9.15	File Status Messages.....	260
9.16	Network Messages Generated By the Centralized Management Facility.....	273
9.17	Network Messages Generated By the Distributed Management Facility.....	274
9.18	Components of the Simulation Model.....	276
9.19	Processing Time vs. R_S	284
9.20	Average Storage Requirements vs. R_S	286

LIST OF ILLUSTRATIONS (continued)

FIGURE		PAGE
9.21	Overhead Traffic vs. R_s	286
9.22	Network Traffic Delay Characteristics.....	287
9.23	Average Number of Requests in File Queue vs. the Ratio MCF/SCF for Different Update/Retrieval Ratios.....	288
9.24	Average Waiting Time (in minutes) of Requests in File Queue vs. the Ratio MCF/SCF for Different Update/Retrieval Ratios.....	289
9.25	Average Number of Requests in File Queue vs. the Average Number of Requests Originated in Time Δt , for Different Update and Retrieval Durations $(T_r + T_u)/2$	290
9.26	Average Waiting Time (in minutes) of Requests vs. the Retrieval/Update Duration Ratio for Different MCF/SCF Values.....	291

LIST OF TABLES

TABLE		PAGE
3.1	Arrival Statistics (GPDS Model)	65
3.2	Output Statistics (GPDS Model)	65
4.1	File Availability to Node 1	83
4.2	File Availability to Node 2	83
4.3	File Availability to Node 5	83
6.1	File Access Traffic in Units of Traffic/unit of time ($u_{ij} = u'_{ij}$, $v_{ij} = v'_{ij}$)	124
6.2	Storage Cost in \$/month	124
6.3	Values of A and B in Dollar/month	125
6.4	Communication Lines Leasing Cost in Dollar/month	125
7.1	File Access Requirements u_{ij} and v_{ij} (in messages/ sec) and File Storage Cost in (\$/month) for the Problem of Example 2	144
7.2	Values of Constants A and B for Different Line Capacities	144
7.3	Link Reliabilities and Link Lengths	146
7.4	File Access Requirements u_{ij} and v_{ij} (in messages/ sec) and File Storage Cost (in \$/month) for the Problem of Example 3	146
7.5	File Access Requirements u and v (in messages/sec) and File Storage Cost (in \$/month) for the Problem of Example 4	147
7.6	Link Reliabilities and Link Lengths	150
7.7	Traffic Matrices	151
7.8	Storage Costs in \$/bit/month (Example 5)	152
7.9	Storage Requirements (1 byte = 8 bits)	160

LIST OF TABLES (continued)

TABLE		PAGE
9.1	File Queue Table	228
9.2	File Queue Table	231
9.3	Deadlock Condition	232
9.4	Pre-Emptying Process P_1	234
9.5	File Access Completion	235
9.6	File Access Completion	236
9.7	Control Commands for the Centralized Access Control Protocol	249
9.8	File Request Modes	277

CHAPTER 1

INTRODUCTION

1.1 Distributed Data Bases and Information Networks

Over the last decade, the interaction between computer and communications has steadily developed. While many government agencies, academic institutions and business firms continue to make use of their own computers, an increasing number of these organizations are using communication facilities to access commercial time-sharing and batch processing services. As the use of such remote computer services increases, the demand for reliable and low-cost means of communicating over wide geographic areas also increases.

For many years, a number of networks of interconnected computers have been developed, while many others are currently under study. The main objective of these networking efforts has been to demonstrate that substantial economic savings and/or increased capabilities are possible by the sharing of computing and communications resources.

An important example of resource sharing in a computer-to-computer network is the shared access to specialized data bases in which a user program running in one computer can access file systems that are on another, possibly remote, computer. Computer communication networks whose main shared resources are data bases or file systems are referred to in this study as *information networks*.

In addition to its computers, an *information network* can include any or all of the following:

- . Telecommunication facilities
- . Communication processors
- . Terminals
- . Multiplexers, concentrators, etc.
- . Other interface devices.

This study is concerned primarily with information networks which offer a distributed as distinct from a centralized computing or data processing service to its users. Such networks therefore include two or more data processing computers (information processors), each of which has permanent files attached to its auxiliary storage devices, as shown in Fig. 1.1. Files attached to a given information processor are accessible to users who are directly connected to it, as well as to all other remote users in the network. Furthermore, all users can read/update/execute records in all files attached to all remote information processors with the same ease they can read/update/execute records of files attached to these information processors to which they are directly connected.

The total set of permanent files stored on all the information processors of the network form a *distributed data base*. Information networks supporting a distributed data base can be characterized by the following basic parameters [B 1]:

- . *Geography*
- . *Volume of information*
- . *Information processing*
- . *Response time requirements*
- . *Availability of Network*

FIGURE 1.1
DISTRIBUTED COMPUTING NETWORK CONFIGURATION

The term *Geography* above refers to the location of the terminals requiring access to the network and the location of the computers that process the information for these terminals.

The *Volume of Information* to be transmitted between terminals and computers will be a function of the particular application and the type of terminal involved as well as the frequency of usage.

Information processing involves the manipulation of information and the data base requirements.

Response time requirements in a given network may range from fractions of a second to minutes, or even hours, depending on the priorities involved. Furthermore, the *response time to a request* is the sum of two distinct times:

- the time required by the information processor (i.e., the search, retrieval or processing time)
- the time required to transmit the request through the communication network and receive a reply.

The *Availability* of the network is a measure of the probability that the communication network, as well as the information files, are available when demanded by a given user.

It follows from the above definitions that the information network can be considered to be composed of

- a computer communication network supporting a distributed data base that consists of
- a group of shared information files.

In the following discussion, classes of computer communication networks and file systems considered in this study are presented.

1.1.1 Computer-to-Computer Communication Networks

A computer network is an interconnected set of dependent or independent computer systems (information processors) which communicate with each other in order to share resources (such as hardware resources, software resources and data bases) - and/or for load sharing, and reliability reasons.

A very important component of the computer network is the *communication network* (often referred to as *communication subnetwork*), which includes all the hardware and software (modems, channels, concentrators, etc., specifically dedicated to the transfer of data from a computer centre to another. In some networks, the burden of the data transmission (buffering, routing, etc.) is shared between computers and communication network; in some others, it is completely assigned the communication network. Many different communication schemes can be implemented: dial-up lines, dedicated lines, radio broadcasting and satellite communications. The computer systems of the network are referred to as network nodes and the communication channels connecting these nodes are referred to as network links. The structure of connection of the communication channels (network links) is usually referred to as network *topology*.

Three classes of network configurations exist. These are the distributed, centralized (star) and loop (ring) structures. Most networks constructed so far fall into one of these classes. The structure and operation of each of these configurations will be illustrated through the use of examples.

As an example of distributed computer communication networks we consider the ARPA (the United States Advanced Research Project Agency) network [C3, C4, F1, R2]. The ARPA network is a nationwide system designed both to explore network topology and to interconnect a number of research centres. The key aim of the system is to allow the accessing of programs, services and data from any place on the network.

The ARPA network is a distributed network; sites (nodes) on the network are connected to each other either directly or indirectly through intermediate sites. The network can be broken into two parts. One part consists of the computers which provide the computational services of the network - the *Hosts*; the other part deals with the function of servicing the communication needs of the network - the *Communications Subnetwork*.

The participant computer systems (*Hosts*) in the ARPA network are in general *heterogeneous*. This means that most of these computers differ in make, word length, speed and software operating system, among other features.

The Communications Subnetwork consists of communication processors (mini-computers) connected via 50-kilobit leased telephone lines. In the ARPA network, these communication processors are called IMPS (Interface Message Processors). Each Host computer is connected to the Communication Subnetwork through an IMP. The IMPS thus act as *Front End Processors* to the host computers. The Communication Subnetwork operates in a message-oriented store-and-forward fashion: a message is stored at intermediate points as it makes its way toward the destination. Each time the message is handed forward correctly, the handing node is freed from any further

responsibility for the message. Since it is often necessary to send messages of substantial size, the network breaks long messages into smaller submessages called *packets*. The packets are independently forwarded through the Communications Subnetwork (this is known as packet switching) and reassembled into the original message at the IMP of the destination node before delivery to the destination host. The IMPS of the Communications Subnetwork govern routing of packets through the network in order to minimize the transit time of the message and to increase the utilization of the transmission facility.

Other examples of distributed computer communication networks are:

- CYBERNET Control data corporation's homogeneous commercial computer network interconnects about 40 computers or nodes, around the U.S., using both voice grade lines and lines operating at 40.8 kilobits per second.
- INFONET Computer Science Corporation's commercial network handles mainly remote batch and remote job-entry applications for a variety of subscribers.
- TYMNET Tymshare Inc.'s network, another commercial system, interconnects several Xerox Data Systems host processors and offer time sharing services to a variety of users.

Centralized Computer Networks

In centralized computer networks, all sites are connected together via one central site. All communication functions (message routing, code conversion, etc.) are performed at the central site. Similarly all network control functions relating to network job scheduling and the allocation of host resources to users are monitored from one location. Examples of centralized computer networks are:

TUCC Network

The triangle Universities Computation Center network is a joint undertaking of the Duke, North Carolina State and North Carolina Universities. This is a centralized network of homogeneous machines. At each of the three nodes of the network there are IBM 360/40S or 50S. These 360S do local batch jobs in addition to handling the telecommunications necessitated by the network. The nodes of the network are connected to the central facility by means of leased 40,800 bps half duplex lines.

IBM Network/440

Using a star topology, this research network at IBM's Yorktown Heights, N.Y., facility interconnects heterogeneous processors at various universities and IBM locations throughout the United States [K1].

Loop (ring) Computer Networks

The loop configuration is suitable for a network in which the remote resources are relatively close to each other, for example, where the nodes are clustered within the limits of one large city. In this case, communication interfaces become cheaper. But the loop circuits carry the maximum traffic from all nodes. Therefore a loop network faces the problems of high cost and lack of ready availability of ultra-high-speed lines (operating at mega-bits per second), especially when the resources connected are geographically dispersed. An example of a loop network is the DCS (The Distributed Computing Systems), which is an experimental network [F2], structured to interconnect small and medium size processors in a loop (ring) network. The processors are located within a relatively small geographic area: the University of California (Irvine) Campus.

Efforts to construct computer communication networks are also under way in other countries. Examples include CYCLADES in France, the

NPL project in England [54], and COST11 multi-national network in Europe [B12].

The selection of the best network configuration for a given application is certainly a very difficult problem and depends on many factors (degree of decentralization of the network, traffic characteristics, delay requirements, hardware costs, etc.). The degree of decentralization of the network is determined by the type of participation expected by the nodes (host computers). Any node can be a provider of resources exclusively, a user of resources exclusively, or some combination of resource provider and resource user.

Centralized network structures are appealing when a relatively large number of sites need only reach one or few sites which participate merely as resource providers. This means that each remote host computer needs only to address another host that is at a higher level in the network hierarchy. Fully distributed network structures are most appropriate when all sites participate both as resource users and resource providers.

The present study considers a distributed computer network configuration, with store-and-forward communication scheme and general distributed topological structure.

1.1.2 File Systems

In this dissertation, an information file is defined as a collection of logically related data records. A file organization is defined as an interrelated set of structures and algorithms used to store, update, maintain and retrieve the file data records [51]. A group of files, together with information about their organization form a data base.

An example of a file system is an airline reservation system data base. In such a system, the collection of reservation records for a single route may constitute a file. Each record contains a number of data items and corresponds to one flight on the route. Each data item contains status information for a single seat on the flight. The data base is the collection of all the flight files. The records of the data base are retrieved, displayed and even altered by the ticket agents before making reservations and receiving confirmations.

Records of a given information file are characterized by the following basic parameters:

- . Rate at which the record is requested
- . Probabilities of origin of record enquiries
- . Distribution of lengths of request and reply messages for the record
- . The storage size of the record.

In the case of airline reservation system, the probabilities of origin of record requests would be high for cities along principal routes and low for others.

For this study, it is assumed that all records of a given file are *homogeneous*. This means that the request rate, the probabilities of origin of record enquiries and the distribution of lengths of request and reply messages are identical for all the records of a given file.

A formal definition of a file system can be given as follows

[W1]:

A file system F is a finite collection of files,

$$F = \{F_j | j = 1, 2, \dots, M\}$$

where F_j denotes file j . A file j is a collection of records with the following common set of parameters:

$$F_j = \{L_j, R_j, K_j, K'_j, P_{1j}, \dots, P_{nj}\}$$

L_j is the number of bytes of storage required by file j

R_j is the rate at which file j is requested

K_j is the message length of requests of records of file j

K'_j is the message length of replies of records of file j

P_{ij} is the probability that a request for a record of file j is originated at network node i , ($i = 1, 2, \dots, n$). 5

The request and reply message lengths may be chosen from specified probability distributions. In many cases, the request and reply lengths were found to have an exponential distribution or a uniform distribution [F3]. Either of these distributions can be completely specified by its mean value.

The set of probabilities $\{P_{1j}, P_{2j}, \dots, P_{nj}\}$ defines a probability mass function over the set of all nodes, i.e.,

$$\sum_{i=1}^n P_{ij} = 1$$

1.1.3 Mode of Information Network Usage

In information networks supporting a distributed data base, the following five major types of applications are identified [D1]:

- Remote job entry
- Dynamic file access/transfer
- Remote batch processing
- Interactive
- Load sharing.

In *remote job entry* applications, one computer acts as a remote job entry station. Jobs in the form of programs and data are sent to this computer, which processes the jobs and sends the results back to the originating host computer.

In *dynamic file access/transfer* applications, a user at a terminal or a host computer executing a batch process can automatically request a data file to be transferred to it from another location in real time and on demand.

In *remote batch processing* applications, the local host computer operates in its batch mode but has access (through the network) to data files at a remote node. The network monitor programs and communications interfaces maintain necessary file directories and perform all required data conversions between any two computers at a given time.

In *interactive* applications, a user terminal connected to the network through his local host computer may control one or more batch processes at remote systems.

In *load sharing* applications, the user submits a complete job to the network via his local computer. The user does not normally know which site will get the job since it is the responsibility of the network control

program to find all the necessary data, construct the proper job-control sequence and assign the job to be processed in the best way by the network's aggregate resources. This mode of operation offers the lowest cost utilization of computer and communications resources, and hence represents the economic objective of multi-computer networking.

The distributed data base model derived in this thesis describes those information networks that employ as *principal applications* the last three types of operation, namely; remote batch, interactive and load sharing. The model, however, does not preclude the incorporation of the first two types of applications, i.e., remote job entry and dynamic file access/transfer. It merely places an emphasis on the latter three applications.

1.2 Uses and Applications of Distributed Data Bases

The uses and applications of distributed data bases can be demonstrated by several examples. A major one is the proposed use of CANUNET (Canadian Universities Computer Network) to support a segmented data base (distributed information files).

The shared or common information files which will be stored in some computer centres and accessed by the others in the network may comprise for example, the data banks for the following information systems [C5]:

I Library Information Systems

While nearly every university in Canada has some form of automated library system, the approach to automation has varied. Examples are:

University of Toronto

The basic concept has been to build a common catalogue data base which can be used for many applications. There are about 700,000 tiles now in the machine readable form which is equivalent to about 1.8 million volumes.

University of Guelph

An off-line batch system catalogue processing, serials, government documents, acquisitions and circulation is almost complete.

Laval University

A system in operation at Laval for the past two (2) years provides on-line access to a file of 25,000 technical publications giving information on the state of collection and its location on the campus. The system could be expanded to include books, maps, records and other periodicals without major modification and its services could be extended to other universities via a computer communications network.

A computer network could substantially improve access to holdings on a national basis by making all library catalogue files available on line from any location. A network could also allow for immediate access to specialized national data bases, for example, government documents, from any university location.

II Legal Information Systems

Queen's University QUIC/law system was started in 1969 and its function is to provide legal researchers with terminal access to the text of Federal Supreme Court Decisions 1923-70 and has been extended to pro-

vide NRC with access to 6000 Pollution Documents. In total, Queen's has about 15 million words of text in machine readable form including the following files on the information retrieval system:

- Federal Supreme Court Decisions 1923-70
- Ontario Court Reports 1962-70
- Federal Statutes (consolidated)
- Federal Regulation (consolidated)
- Canadian criminal cases
- B.C. Statutes (in process)
- Treaties of the Commonwealth (in process).

Examples of the systems users:

- Law offices
- Provincial and district courts
- Offices of the federal government
- Individual legal researchers.

Users sharing the legal information files are obviously scattered in different geographic locations and a file allocation scheme is therefore needed.

III Environmental Information Systems

There are a number of environmental applications in various stages of development at Canadian universities. These tend to be based on the need to access large data files either to retrieve specific information or to analyze quantities of data in the following fields:

- Survey, mapping and land titles
- Geodetic surveys
- Geological data
- Soil data
- Urban and Regional Planning.

The need to access these information files by different communities of users in the different regions of the country makes the concept of segmented or distributed data bases more feasible and the application more useful.

IV Social Science Data Banks

Information on data banks and the needs of the social sciences do not seem to be readily available at the present time. This may be owing to the fact that most of the data in universities appears to be in small private collections of individual research workers, rather than being accumulated in a large data bank. Before examining the applicability of the segmented data base concept to this area, study must be conducted to determine:

- if presently compiled social science data are interesting to a wider community than the individual research worker and whether some of this data might be accessible over a network;
- how more general information from such agencies as Statistics Canada, might be made available to social scientists and if a university computer network might be a possible vehicle for its dissemination.

The concept of distributed data bases is also feasible for companies operating over a wide area. In this case, the company may choose

to distribute its data base and link the smaller data bases using the network. For example, a manufacturing firm with a large customer file can place records for its west coast customers in a western regional computer centre and records for eastern customers in a compatible system on the east coast. Such a network linkage probably will reduce communications costs below those of a completely centralized data base, since access to regionally located records will not require transcontinental message transfers.

A similar application is also feasible in the case of large payrolls system where it may be possible to decentralize the payroll data base and store its different segments in the information processors located near the regional offices. Such a distributed system, besides offering a higher reliability, may further reduce communications costs.

1.3 Technical Problems Associated with Future Developments

The development and operation of distributed information networks involve a number of technical problems. Two of the most important ones are:

1. The design (resource allocation) problem associated with early phases of development of information networks.
2. The distributed data base access control problem associated with the operation of the information network.

The Resource Allocation (Design) Problem

In general, the resource allocation (design) problem involves the optimization of a cost (objective) function over a number of design variables subject to a predetermined set of constraints. The objective function

includes combined costs of storage, communications (line capacities and front-end-storage) and data processing. Design variables include: node locations, front end storage allocation, network topology, line capacity assignment, file assignment, traffic routing scheme and message priority discipline. Design constraints include average message delay, throughput, reliability (file availability) and topological constraints.

The general resource allocation (design) problem is extremely difficult to solve in its entirety. This is because the design variables are interdependent and general design techniques are unknown to date. In order to make progress towards a solution, the optimization problem is restricted to a subset of design variables and constraints while the remaining design variables are assumed given. This way, a variety of different problems can be obtained from the above formulation. A literature review of the approaches that have been developed to solve some of these problems is presented in Section 1.4. In the present study, a new and more comprehensive approach to the resource allocation (design) problem is developed. An introduction to the approach is presented in Section 1.5.

The Distributed Data Base Access Control Problem

In the operation of distributed information networks, means must be provided to associate jobs or interactive executions with file(s) needed. This can be done either manually, i.e., under user control or handled automatically by the information processor (computer system) software.

User Controlled Access

All file placement and job addressing for file access is manually controlled by the user. Each user addresses all of his input to a specific information processor, usually the one closest to his terminal. All files created by the user are therefore attached to that information processor. If a user wishes to access another user's file, he must find out where it is located, and addresses his job or interaction to the information processor of that location.

Software Controlled Access

The user control described above is ideal for networks in which most file use is local. However, it may become unworkable if there is heavy use of files remote from the user's local information processor and/or heavy use of files in multiple information processors accessed by a single application. In cases like these, software knowledge of file location is necessary. Jobs can then be directed automatically (under the supervision of the information processor's operating system) to the correct location for execution.

Software controlled access has several attractive features; the most obvious one is its simplicity from the user's point of view, since in this mode the user need not be concerned about file location or separation. On the other hand, software controlled access has two serious problems, viz., job interference (deadlock situations and critical rate conditions), and the task of controlling access to files.

Deadlock situations could be caused between jobs; each of two jobs could have files locked at a different location, and the two would be mutually hung waiting for access to the other's files.

Critical rate conditions are those situations where the outcome of a sequence of (supposedly independent) operations is dependent upon the order in which the operations occur. Critical rate conditions are a form of job interference introduced by network (message) delays and affect the synchronized updating of multiple copy files.

It is apparent that to avoid job interference, the operating system providing the software access service must be capable of performing the following functions:

- recognizing dependencies between jobs, detecting and resolving deadlock situations;
- controlling recovery of an abnormally terminated job which was updating files at multiple locations;
- synchronizing the update processes of multiple copy files within the information network.

The task of controlling access to the distributed file system requires interaction between the operating systems of the host computers. Each operating system will be equipped with the necessary logic to perform the functions mentioned above. The interaction between the operating systems will be based on a predefined set of conventions, i.e., *Protocol*.

1.4 Review of the Existing Literature

In the following discussion, related work that has been published in the areas of resource allocation (design) and software controlled access of distributed data bases is reviewed.

Resources Allocation (design) in Information Networks

A number of authors have tackled the resource allocation problem in information networks using different models. Chu [C2] was the first to

approach the problem. He investigated the optimal allocation of files in multiple computer system using a 0-1 integer programming model. In this model, fully connected computer communication networks and a pre-fixed number of copies of each file are considered.

Whitney [W1] investigated the problem of assigning file copies to nodes of a linear graph in such a way that minimizes the traffic induced along the edges (communication channels) of the graph. In this model, it was shown that the communication system cost is also minimized as a result of minimizing the traffic.

Casey [C6] approached the file assignment problem from a different point of view; essentially his model distinguished between two types of traffic: update traffic and query traffic. Using this model, an upper bound on the number of copies of the file in the system was derived as a function of the update/query traffic ratio. The same model was also used to develop a search for the minimum cost allocation of file copies to nodes of the network. The use of the model, however, is limited to the case of fully connected networks or when it is required to allocate the resources prior to the topological design since it does not take into consideration the delay and reliability aspects.

The problem of optimal file allocation in multi-level storage systems was considered by Chen [C1] and Rammamorthy [R1]. Models were derived whose main objective is to allocate files within a hierarchy storage system so that the total storage cost is minimized and at the same time a given system response time requirements are satisfied.

The problem of assigning capacities to communications links when the flow matrix is given was examined by Gerla [G1] who applied both Dynamic Programming and Lagrangian Decomposition to solve the discrete capacity case. Demercado [D2] approached the same problem using a heuristic algorithm which in some cases sacrifices optimality in order to arrive at "good" solutions quickly.

Software Controlled Access of Distributed Data Bases

As will be explained in later chapters of this dissertation, the problem of software controlled access of distributed data bases is closely related to the problems of Host to Host protocol design and interprocess communication in resource sharing computer networks. Because of this relationship, a brief survey of the work published concerning the protocol and interprocess communication problems is presented in the following discussion.

Experimentation with the ARPA network has yielded the most sophisticated hands on experience in the field of computer to computer communication protocols. The concept underlying the protocol design in the ARPA network is the use of a "layered" approach in the specifications of the communications protocol [C3]. The inner layer specifies the Host to Front-End-Processor protocol. The next layer specifies methods of establishing communications path, managing buffer space at each end of a communications path, and providing a method of "interrupting" a communications path. This protocol is known as the second level or Host/Host protocol and is used by all higher level protocols. Examples of further layers of protocol include a data transfer protocol, a file transfer protocol, a graphics protocol and a remote job service protocol.

Postel [P1] examined the problem of communications protocol design using graph modelling techniques. The communication process between a pair of sending and receiving processes in two different computer systems is represented by graphs with nodes and edges. The nodes represent the the states of the sender or the receiver with respect to the phases of the communication process and the edges represent the transitions to the different states. Using graph properties, the model can be applied successfully to a given set of protocols to test if deadlock and critical rate conditions are existing.

Walden [W2] proposed a system for interprocess communication in a resource sharing network. In this system a set of operations enabling interprocess communication within a single time-sharing system is first described. This set of operations differs from many of the interprocess communication techniques currently in use within time sharing systems, such as communications through shared memory. Instead, it relies on techniques that can be easily generalized to permit communications between remote processors. The application of the proposed system is, however, still hypothetical and its performance cannot be assessed before actual implementation. Moreover, its scope is conceptually limited to networks in which the operating systems of the different host computers are identical (homogeneous).

1.5 Objectives and Contributions of this Study

The main objectives of this dissertation are:

1. to develop a more comprehensive approach to the resource allocation (design) problem in distributed information networks.
2. to resolve some of the problems associated with software controlled access methods in distributed data bases. Specifically, job interference and file access control problems are considered.

In the *design area*, a model for the allocation of resources in distributed information networks is formulated (Chapters 2, 3 and 4). The model allows for the simultaneous allocation of files (to nodes) and communication capacity (to links) and results in an optimization problem in which a combined storage and communication cost function is minimized subject to an average message delay and file availability constraints. It will be shown (Chapter 4) that the model is in fact a generalization of previously developed models and differs from them in several aspects. The main differences are:

- The introduction of file availability constraint (this was not considered in previous models).
- The ability to handle arbitrary and general network topologies.
- The representation of more realistic communication and storage cost structures.

The optimization problem resulting from the resource allocation model falls into the class of nonlinear bivalent programming problems. The use of classical integer programming techniques to solve such an optimization problem yields solutions which, while theoretically convergent, are computationally cumbersome, even for small size problems. These techniques also require excessive computer storage space which prohibits their use for solving large size problems (Chapter 7).

A new algorithm is developed to solve the resource allocation model (Chapter 6). The algorithm yields relatively low (but not minimum) cost solutions quite rapidly. In comparison with an integer programming approach, the new algorithm is shown to allow substantial savings in computer time and storage requirements (Chapter 7).

In the area of *software controlled access to distributed data bases*, three general abstract models for deadlock detection and recovery or deadlock prevention are introduced in Chapter 9. Based on these models, three algorithms will be derived; any of these algorithms can be used to construct the logical functions of file access control systems.

Two approaches to the design and implementation of a software controlled access system are presented in Chapter 9. In the first approach, information concerning file access status and active user processes is exchanged between processing nodes so that the control is *distributed*. In the second approach control resides in a designated node, which is referred to by all active user processes when requests for file access are initiated; control is thus *centralized*. Protocol specifications are derived for each of the distributed and centralized control approaches.

The two approaches are simulated for a network of information processors supporting a group of data files shared by a distributed user group. The main purpose of the simulation is to evaluate the two approaches in the light of several performance criteria and provide guidelines for selecting the software controlled access approach which is most suitable for a given information network.

1.6 Thesis Organization

The remaining nine (9) chapters of this thesis are organized as follows:

Chapter 2 presents a reliability model for the distributed information network supporting shared information files. The model takes into account the reliability of the information processing and communication processing devices in every node as well as the reliability of the communication links. This chapter also includes a summary of techniques used to compute terminal reliability. The results of this chapter will be used in the derivation of the file availability constraint.

Chapter 3 is concerned primarily with the traffic routing and message delay aspects in the communication subnetwork. The results of this chapter are used to compute the traffic flows over the communication links which are necessary in the estimation of the average message delay.

Chapter 4 is concerned with the derivation of the resource allocation model. The model consists of a cost (objective function) and a set of delay and reliability constraints. It is shown that the model falls into the class of nonlinear bivalent programming problems.

Chapter 5 presents a discussion of linear and nonlinear bivalent programming techniques used to obtain solutions to the class of programming problems represented by the resource allocation model.

Chapter 6 examines the properties of the resource allocation model and the limitations of the bivalent programming techniques. A heuristic algorithm is then presented which permits computationally tractable solu-

tions to the resource allocation model. The heuristic algorithm is illustrated by means of an example.

Chapter 7 presents the results of numerical computations carried on a number of study cases. Sensitivity of resource allocation to variations in the values of some of the parameters of the resource allocation model is examined.

Chapter 8 defines the problems associated with the implementation and operation of distributed file access systems. Specifically the software controlled access methods are discussed.

Chapter 9 is devoted to the development of techniques to resolve the job interference and asynchronous multiple-copy file updating problems associated with software controlled access methods. Two approaches are proposed and their performance is evaluated. Results of computer simulation carried on a hypothetical information network are reported.

Chapter 10 contains some concluding remarks and recommendations for further study.

CHAPTER 2

FILE SYSTEM AVAILABILITY

2.1 Introduction

Availability considerations are important in the assignment of copies of each file to the network nodes. The availability of a given file is defined as the probability that the file is accessible when requested by users, remote or local with respect to the storage location of the file. Since the file system under consideration is supported by a computer communication network, the availability of each file will depend on the reliability of the nodes and links comprising the network.

As mentioned earlier, each network node consists of a computer system serving a group of local users and supporting some of the shared network files. The computer system interfaces with the communication network through a front end processor which handles all the communication functions (message switching, error checking, line control, etc.). The set of front end processors and communication lines form the communication subnetwork. In such a configuration it is safe to assume that the computer system reliability is totally independent of the reliability of the communication subnetwork. In order to arrive at an expression for the file availability, the reliability of the information network main components, viz., the computer systems and the communication subnetwork, must be defined first.

The node reliability is a function of the reliability of the computer system components (CPU, storage devices, users interface, etc.) and is examined in Section 2.2.

The terminal reliability, defined as the probability of maintaining communication between any two nodes in the network, will be used as a measure of the communication subnetwork reliability. The terminal reliability for any given pair of nodes depends on the reliability of the links and the front end processors (acting as message switching centres) that lie on the paths between these two nodes. Techniques for computing the terminal reliability are introduced in Section 2.3. A note on the time dependent terminal reliability is also included in Section 2.4.

The results of this chapter will be used in the derivation of the file availability constraints (Chapter 4).

2.2 Node Reliability

Without loss of generality, the computer system at each node is considered to be composed of the following four units (see Fig. 2.1):

Unit 1 consists of the main memory (the high speed memory), the arithmetic and central processor (CPU), other supporting devices (buffer storage, bus control unit, etc.), depending on the structure of the computer system and a high speed secondary storage device. The reliability of this unit is denoted by a_{ci} for computer i .

Figure 2.1

Components of a Computer System

Unit 2 consists of the auxiliary storage devices such as discs, drums, tapes, etc. Part of the data base (shared information files) is stored in one or more of these devices. The reliability of this unit is denoted by a_{si} for computer i .

Unit 3 consists of the communication channels and communications support devices that allow unit 1 to interfaced with the front end processor which in turn connects the computer system into the communication network. The reliability of this unit is denoted by a_{li} for computer i .

Unit 4 consists of the communication controller, teletypes and other interfaces necessary for a user to interact with the computer system. The reliability of this unit is denoted by a_{ui} for computer i .

The reliability of any unit is defined as the fraction of time the unit is operating. Figure 2.1 shows these basic four units and represents a computer system which allows a number of local users to access files stored in the auxiliary storage devices of this system. The computer system also allows the users to access files stored in other network remote computers. For a user sitting at the location of computer i with the desire to access file j , the following two possibilities exist:

- 1) If file j is stored in unit 2 of computer i , then units 1, 2 and 4 must be operating during the period of accessing file j
- 2) If file j is stored in unit 2 of some other computer k in the network, then the following units must be operating during the period of accessing the file:

- units 1, 2 and 3 of computer k
- units 1, 3 and 4 of computer i
- At least one communication path between computers i and k. The probability of maintaining communication between nodes i and k (the terminal reliability) will be denoted by r_{ik} .

To obtain an expression for the availability of file j to a user local to node i, in terms of the reliabilities of the information network components, consider the following cases:

Case 1 a single copy of file j is stored in the computer system of node i. Under the assumption that the computer system components fail independently, the file availability will be equal to,

$$a_{ui} \times a_{ci} \times a_{si}$$

as represented by Fig. 2.2(a).

Case 2 a single copy of file j is stored in the computer system of node k. The file availability will be equal to,

$$a_{ui} \times a_{ci} \times a_{li} \times r_{ik} \times a_{lk} \times a_{ck} \times a_{sk}$$

as represented by Fig. 2.2(b).

Case 3 two identical copies of file j exist in the network with one copy stored in node i and the other copy stored in node k.

The file availability will be equal to,

$$a_{ui} \times a_{ci} [1 - (1 - a_{si})(1 - a_{li} \times r_{ik} \times a_{lk} \times a_{ck} \times a_{sk})]$$

as represented by Fig. 2.2(c).

Case 4 two identical copies of file j exist in the network with one copy stored in node k_1 and the other copy stored in node k_2 . In order to calculate the availability of file j to a user at node i , the reliability of communication between node i and both or either one of the two nodes k_1 and k_2 holding copies of file j , must be computed first. Unfortunately for general network topologies, no feasible technique exists for computing such reliability. The problem will be even more complicated if an arbitrary number of copies of the file exist in several network nodes and an expression for the file availability to any network node as a function of the file locations is required. This is because the paths from node i to the other network nodes holding copies of the file may share in common a number of elements (nodes and links). The problem of expressing the file availability in an analytical form in terms of the file allocations is examined in Chapter 4.

In the following two sections, a summary of techniques used to compute the terminal reliability, r_{ik} , for any pair of nodes i and k , and a given network topology is presented.

2.3 Terminal Reliability

The calculation of the terminal reliability is based on a linear graph model representation of the network. The graph consists of nodes

(a)

(b)

(c)

Figure 2.2
RELIABILITY OF NETWORK COMPONENTS

and links connected in a given topology. In message switching computer communication networks, nodes refer to front end processors while links refer to the communication channels connecting the front end processors to form the communication subnetwork. In general, the network can be assumed to have neither parallel lines in a link nor links in series, since any given network may be reduced to a simple network of this form using parallel or series reductions. Thus parallel lines or series links can always be replaced by a single link with equivalent reliability.

Each network element i may be either operating or failing with probabilities p_i and $q_i = (1-p_i)$, respectively. The probability p_i is called the reliability of element i . A binary variable x_i can then be utilized to characterize the failure of element i . When x_i assumes the value 0, this indicates the failure of element i , and when x_i assumes a value of 1, this indicates the operation of element i . Thus the binary vector

$$x \triangleq (x_1, x_2, \dots, x_n)$$

characterizes the state of the network.

Under the assumption that the network elements fail independently, the probability measure for state x_i is simply $p(x_i)$. That is, $p(x_i)$ is equal to either p_i or to q_i depending on whether the binary variable x_i assumes the value 1 or 0. Thus $p(x_i)$ can be expressed as

$$p(x_i) = x_i p_i + (1 - x_i) q_i.$$

In calculating the terminal reliability, r_{ik} , for any pair of nodes i and k , it is necessary to enumerate the set S of these states x of the network, each of which allows communication between nodes i and k . The sum of probabilities associated with each of these states is equal to the terminal reliability r_{ij} .

In order to determine the set S , we first define a function ψ on the set of all possible states of the network, as follows

$$\psi(x) = \begin{cases} 1, & \text{if the state } x \text{ allows communication between} \\ & \text{the pair of nodes under consideration} \\ 0, & \text{otherwise} \end{cases}$$

Also,

$$\psi((1,1,\dots,1)) = 1$$

$$\psi((0,0,\dots,0)) = 0$$

A general expression for r_{ik} can be written as,

$$r_{ij} = \sum_{x \in S} \prod_{i=1}^l [x_i p_i + (1-x_i) q_i] \quad (2.1)$$

where the set S is defined as,

$$S = \{x | \psi(x) = 1\}$$

Several techniques have been developed to calculate the terminal reliability using expressions of the type shown in eqn. (2.1). In the

following, a brief description of the main approaches underlying the derivation of these techniques is presented. A complete survey of these techniques is given in [D3].

Path Enumeration Method

For two nodes in the network to communicate there must exist at least one path between them. Furthermore, all the network elements in this path must be operating. Thus it is important to know the alternate possible paths between the nodes under consideration. Once these are known, it is possible to determine the corresponding set S of states. A path finding algorithm for general network topologies was developed by Fratta [F4] and employed successfully to find all the possible communication paths between any pair of nodes.

Cut Enumeration Method

A cut between two nodes in a network is defined as any minimal set of elements whose removal from the network would disrupt all paths between the two nodes in question. Therefore, once all cuts between the nodes under consideration have been identified, the set of failed states can be determined readily. A procedure to identify these cuts has been developed [W3] and a method applying cut enumeration to calculate the terminal reliability, r_{ij} , has recently been given [H1].

The question whether the *path* or *cut* enumeration method should be used for reliability calculations can be determined as follows. In a network with b links and n nodes the order of the number of cuts is

2^{n-2} whereas the order of the number of paths between any pair of nodes is 2^{b-n+1} . Therefore for networks with nodes of degree (number of incident links) greater than 4, $b > 2n$ and $2^{b-n+1} > 2^{n-2}$. Consequently, such networks have more paths than cuts. This implies that computation should be reduced in such cases by calculating network reliability using cut enumeration instead of path enumeration methods [H1].

The Parametric Method

A new approach to calculate an approximate value of terminal reliability has recently been proposed [B2]. This approach is based on simple transformation of reliabilities of the various network elements. After this transformation is carried out, the transformed network will be equivalent to an electric network with resistances replacing the original links. The value of each resistor is equal to the transformed value of the corresponding link reliability. Using series, parallel and delta-to-star conversion formulae, the network between the two terminal nodes is reduced to an equivalent single resistance. The value of the equivalent resistance is then transformed back to the equivalent terminal reliability value.

The transformation approach, known as the parametric approach, greatly simplifies reliability calculations in complex networks. However, its validity is limited to cases where the values of the elements reliabilities are relatively high. Otherwise it yields results significantly different from the exact terminal reliability values.

Iterative Techniques

A number of iterative techniques has recently been developed for reducing reliability calculations in complex networks to calculations in simpler networks that are derived from the original network.

The first one of these methods is based on the use of eqn. (2.1) rewritten in the form

$$r_{ik} = p_j L + M \quad (2.2)$$

where p_j is the reliability of element j , $j = 1, 2, \dots, l$, $p_j \cdot L$ contains those terms in the sum that have p_j as a factor and M is the remainder whose terms do not contain p_j as a factor. Equation (2.2) may further be rewritten as

$$\begin{aligned} r_{ik} &= p_j(L+M) + q_j \cdot M \\ &= p_j(r_{ik}|p_j = 1) + q_j(r_{ik}|p_j = 0) \end{aligned} \quad (2.3)$$

The term $(r_{ik}|p_j = 1)$ represents the terminal reliability r_{ik} when element j is assumed to be perfectly reliable. Similarly the term $(r_{ik}|p_j = 0)$ represents the terminal reliability r_{ik} when element j is removed. Equation (2.3) suggests the following iterative procedure to calculate r_{ik} :

- Step 1: choose any element j of the network
- Step 2: put $p_j = 1$ and calculate r_{ik}
- Step 3: remove element j from the network and calculate r_{ik}
- Step 4: apply eqn. (2.3) to calculate r_{ik} .

The same four steps may be applied to the networks generated in steps 2 and 3 as many times as necessary to simplify the computational process, i.e., to reduce the original network to much simpler forms for which the terminal reliability can be computed easily. This explains the iterative nature of the technique.

A second iterative technique, developed by Hansler [H2], partitions the network into smaller subnetworks in order to calculate an exact value for the disconnection probability, $(1 - r'_{ik})$, between any pair of nodes i and k . The concept of graph partitioning was also used by Demercado, et. al. [D4] to develop an iterative procedure for calculating the disconnection probability. Essentially, their technique consists of partitioning the network graph into two subgraphs by means of a cut separating nodes i and k . The states of the first subgraph are then used to partition the set S of all possible states of the graph into a number of pair-wise disjoint subsets. Within each of these subsets the failed states of the network can be identified using the states of the second subgraph.

2.4 Time Dependent Terminal Reliability

In this section a network with ℓ elements (nodes and links) is considered; each element i at any time t can be in one of two possible states; namely an operating state with probability $p_i(t)$, or, a failure state with probability $q_i(t) = 1 - p_i(t)$.

All network elements are assumed to fail independently, and the density and cumulative distribution functions $f_i(t)$ and $F_i(t)$ of the first time to failure of each element i are assumed to be given.

If $f(t)$ and $F(t)$ are the density and distribution functions of the first time to failure (or the duration of the failure free operation) between a given pair of nodes, then an overall failure rate function $O(t)$ can be defined for this pair of nodes as:

$$O(t) = \frac{f(t)}{[1 - F(t)]}$$

Since the notion of time dependence is being introduced, some of the definitions introduced in Section 2.3 will be restated.

The binary variable $x_i(t)$ is used to indicate the state of element i at time t . That is, if $x_i(t) = 1$, at time t , then element i is considered operable and if $x_i(t) = 0$, then it is considered failed. The binary vector $x(t)$, namely

$$x(t) \triangleq [x_1(t), x_2(t), \dots, x_n(t)]$$

characterizes the state of the network at time t . The vector $x(t_2)$ will be said to be greater than or equal to vector $x(t_1)$ if $x_i(t_1) \leq x_i(t_2)$ for $i = 1, 2, \dots, n$.

In a similar way a structural function, $\psi[x(t)]$, can be introduced. If the value of the vector $x(t)$ corresponds to a state of possible communication between the pair of nodes under consideration, then $\psi[x(t)] = 1$, otherwise $\psi[x(t)] = 0$. In most practical computer-communication networks, the structural function $\psi[x(t)]$ will be monotonic (see next section); because if the network is in some failure state with no communication between a given pair of its nodes, failure of additional elements

will produce further degradation in the reliability for the two nodes under consideration.

The probability of communication, $r_{ik}(t)$, between a given pair of nodes, i and k , can be expressed as

$$r_{ik}(t) = \sum_i \prod [x_i(t)p_i(t) + (1 - x_i(t))q_i(t)] \quad (2.4)$$

where the summation in eqn. (2.4) is over all those states assumed by the vector $x(t)$ subject to the condition that $\psi[x(t)] = 1$.

In what follows, terminal reliability functions for networks operating in both repair and non-repair environments will be developed.

Non-Repair Environments

A network that operates in a non-repair environment will experience progressive failures among its elements. This condition is reflected by the monotonicity of the structural function $\psi[x(t)]$. That is, $\psi[x(t)]$ satisfies the condition that whenever

$$x(t) \leq x(t_1)$$

then

$$\psi[x(t)] \leq \psi[x(t_1)].$$

for all $t > t_1$.

The probability of communication, $r_{ik}(t)$, between a pair of network nodes can be written as

$$r_{ik}(t) = 1 - F(t)$$

or from eqn. (2.4)

$$1-F(t) = \sum_i \prod_j [x_j(t)p_j(t) + (1-x_j(t))q_j(t)] \quad (2.5)$$

where the summation in eqn. (2.5) is over all these states assumed by the vector $x(t)$, subject to the condition that $\psi[x(t)] = 1$.

In particular for a given network element i ,

$$p_i(t) = 1 - F_i(t)$$

In actual practice it is difficult, if not impossible to evaluate eqn. (2.5), and even the problem of obtaining a computable expression for the failure rate function $\Theta(t)$, is still unsolved for the case where the failure rates of the network elements are general time dependent functions. However, if the failure rate functions, $\Theta_i(t)$, of the individual elements are time dependent and of the Weibull form, that is

$$\Theta_i(t) = k\mu_i t^{k-1} \quad ; \quad k \geq 1$$

and, μ_i is very small, then it is shown in Appendix A that the overall failure rate, $\Theta(t)$, between a given pair of nodes is once again (approximately) Weibull. Specifically, it is shown that the failure rate, $\Theta(t)$, is in fact approximately,

$$\Theta(t) \approx \sum_{\zeta} \left\{ mkt^{mk-1} \prod_{S(\zeta_j)} \mu_i \right\} \quad (2.6)$$

where ζ is the set of all minimum cut sets of the elements separating these nodes and $S(\zeta_i)$ is the set of elements in a given minimum cut set $\zeta_i \in \zeta$. Here, m is the number of elements in set ζ_i and this number is the same for all the minimum cut sets between a given pair of nodes.

Example

The following example illustrates these notions. The network shown in Fig. 2.3 has 5 links, the failure rate, $\theta_i(t)$, of link i , $i = 1, 2, \dots, 5$, is given by

$$\theta_i(t) = k\mu_i t^{k-1}$$

Assuming that all network nodes are perfectly reliable, and μ_i for $(i = 1, 2, \dots, 5)$ are all small.

Equation (2.6) will now be used to find an expression for the failure rate between nodes 1 and 4.

The set ζ contains all the minimum cut sets (the cut sets containing the fewest number of links) separating nodes 1 and 4, and in fact there are two such sets, namely ζ_1 and ζ_2 . That is

$$\zeta = \{\zeta_1, \zeta_2\}$$

and

$$S(\zeta_1) = \{l_1, l_2\}, \quad S(\zeta_2) = \{l_4, l_5\}$$

here $m = 2$, is the number of elements in the sets ζ_1 and ζ_2 .

From eqn. (2.6), the failure rate function, $\theta(t)$, for nodes 1 and 4 is

Figure 2.3

Example of a 4-node Network

$$\begin{aligned}\Theta(t) &\sim mkt^{mk-1} \prod_{S(\zeta_1)} \mu_i + mkt^{mk-1} \prod_{S(\zeta_2)} \mu_i \\ &= 2kt^{2k-1} \mu_1 \mu_2 + 2kt^{2k-1} \mu_4 \mu_5\end{aligned}$$

That is

$$\Theta(t) \approx 2kt^{2k-1} [\mu_1 \mu_2 + \mu_4 \mu_5]..$$

This result indicates that the failure rate $\Theta(t)$ between nodes 1 and 4 depends on the number of elements, m , in the minimum cut sets separating these two nodes. Furthermore, $\Theta(t)$ is also a function of time even if the failure rates of the elements are constants with respect to time (the case where $k = 1$), i.e., the exponential distribution case. This result is a new and important generalization of the result obtained by Burfen and Pittel [B3].

Repair Environments

In a repair environment a completely analogous form to eqn. (2.4) holds for the probability of communication, r_{ik} , at time t between a pair of nodes, i and k . That is

$$r_{ik}(t) = \sum_{\{x(t) | \psi[x(t)] = 1\}} \prod_{i=1}^n [x_i(t)p_i(t) + (1-x_i(t))q_i(t)] \quad (2.7)$$

However, since there is repair possible to each of the elements in the network, two distributions must be considered for each element i . In the case of constant failure, μ_i , and repair, λ_i , rates, these distri-

butions are:

$$F_i(t) = 1 - e^{-\mu_i t} \quad (\text{failure})$$

$$G_i(t) = 1 - e^{-\lambda_i t} \quad (\text{repair})$$

In such a case the state space for each element can be modelled as a two state Markov chain [D5] as shown in Fig. 2.4. The probability of operation, $p_i(t)$, for element i , at time t , given operation at time $t = 0$, is:

$$p_i(t) = \frac{\lambda_i + \mu_i e^{-(\lambda_i + \mu_i)t}}{\lambda_i + \mu_i} \quad (2.8)$$

and

$$q_i(t) = 1 - p_i(t) \quad (2.9)$$

It is now immediately apparent that substituting eqns. (2.8) and (2.9) in eqn. (2.7) leads to an expression which cannot be generally evaluated. This difficulty in obtaining an analytical expression for $r_{ik}(t)$ exists because of the difficulty in enumerating the summation, that is, in enumerating the states of the set

$$\{x(t) | \psi[x(t)] = 1\}$$

Figure 2.4.

Two-State Markov Chain

CHAPTER 3

TRAFFIC ROUTING AND MESSAGE DELAYS

This chapter presents a review of the traffic routing procedures and the associated message delays. The primary focus of the discussion is on the calculation of the average message delay for a given routing algorithm. This will ultimately serve in the derivation of the delay constraint of the resource allocation model. The review is confined to communication networks employing store and forward or message switching (as apposed to circuit switching) routing strategies since the information network model used in the resource allocation problem is based on this type of network.

3.1 General Network Concepts

The communication subnetwork consists of a number of front end processors connected by communication lines (channels) according to a given topology. Each front end processor (FEP) is connected to its local host and represents a node in the communication subnetwork. Host computers communicate with each other via physical records called "messages" which may vary in length up to a certain predetermined maximum. The message is transferred from the host computer to its local FEP via a serial interface. The address of the destination host computer and other control information is contained in the leader of the message which is

followed by the text. The messages received by the FEP from its own host computer is segmented into smaller blocks of data called "packets", for transmission through the remainder of the network. Each packet has a fixed length and contains a number of control bits (headers) in addition to the main text. A packet is stored at intermediate nodes as it makes its way toward the destination node. Each time the packet is forwarded correctly, the previous node is freed from any further responsibility for the packet upon the receipt of a positive acknowledgement. The queueing and routing of the packets or messages at various node (relay points) between the sender and recipient of the message are the factors characterizing message switching techniques. It is obvious that a routing strategy is required at each node to select the next line a message will transverse upon exiting the node. Thus each transmission line thereby multiplexes messages from a large number of source-destination pairs of nodes. In this manner, the bandwidth of the communication channels is more efficiently utilized.

Once the packets arrive at their destination FEP, the headers are stripped off and then reassembled into a physical record (a message). The text contained in the reassembled message is exactly the same as that which entered the network at the source node. The message is then transferred to the destination host computer. Based on the above discussion, the main functions of the front end processor in store and forward computer communication networks are to:

- Segment host messages into packets
- Build packet headers
- Receive, route and transmit store-and-forward packets
- Error detection
- Retransmit unacknowledged packets
- Reassemble received packets into messages
- Monitor network status and gather network statistics
- Generate different acknowledgement messages
- Allocate storage space for message (packets)
- Packet priority assignment (explained in Section 3.3).

The other main component of the communication subnetwork consists of the set of communication channels (links) connecting the front end processors. A communication channel can take on many forms in a given network in terms of microwave links, waveguides, buried cables, telephone lines, satellite links, or underwater links. Associated with each type of communication channel is a bandwidth limitation which constrains the analog transmission bandwidth or the digital data rate (channel capacity).

3.2 Traffic Routing Strategies

Message routing techniques are generally classified into three distinct categories:

- Deterministic routing techniques;
- Stochastic routing techniques; and
- Flow control routing techniques.

Deterministic routing techniques compute routes based upon a given deterministic decision rule and produce a loop-free routing procedure. For example, one can compute routes based upon the shortest distance between any given pair of nodes in terms of the mileage (distance in miles) or the number of hops (number of nodes transversed by a packet enroute to the destination node). Stochastic techniques, on the other hand, operate as probabilistic decision rules. Routes are selected utilizing network topology and estimates of the present state of the network. For example, the time delay associated with each link can be estimated from observations on messages transmitted over it and then packets can be routed along the shortest estimated delay paths. Finally, flow control routing techniques involve a close coordination between the network message routing algorithm and the activities of source and destination FEPs as to the sequencing of messages, the network connectivity, the response of the host computers, and FEP buffer allocation, among other factors. The selection of the traffic routing technique that is most suitable for a given network situation is usually done based on a number of performance measures. These include:

- The average message delay corresponding to different load values
- Sensitivity of the routing algorithm to node and link failures
- Adaptivity to varying source-destination traffic loads
- The overhead traffic generated in the process of exchanging updated routing information (usually limited to stochastic and flow control routing techniques)

- The node storage requirements and processing requirements.

The first three performance measures represent the main characteristics of a routing technique while the last two measures are related more to the implementation (realization) of the technique in a given network.

In the development of the resource allocation model of Chapter 4, deterministic message routing is assumed exclusively. There are two principal reasons for this:

1. In the resource allocation problem (Chapter 4), delay is a function of line capacity assignments as well as link traffic flows; these, in turn, depend on the file node assignment. In stochastic routing or flow control routing techniques, the flow over each link cannot be expressed analytically as a function of the unknown capacity and file assignment. As a result, the resource allocation problem will become analytically intractable if any of these two techniques is employed.
2. Solution of the resource allocation problem is based on specification of the average rate of traffic transactions that are expected to be generated as a result of file updating and retrieval processes.

Four basic deterministic techniques have been developed to date.

These are:

- Ideal scheduling routing
- Flooding
- Fixed routing
- Bifurcated routing.

In order to facilitate understanding the concepts behind these techniques, some notation is defined. For a given message generated into the network, define N_s to be the source node, N_d to be the destination node.

Also defined $\pi(N_s, N_d)$ to be the path (route) followed by the message from N_s to N_d . Let N_e be the current node on $\pi(N_s, N_d)$ at which the message is located in the network. The four deterministic routing techniques are now briefly explained.

Ideal Scheduling Routing

This technique depends on knowledge of the complete present information about messages already in the network and their known routes. Based upon this knowledge, each time a new message enters a node from its host computer, its route is computed to minimize the travel time to its destination node. The technique is obviously impractical for a real network situation. However it provides a theoretical minimum average message delay against which all other routing techniques can be compared.

Flooding

In this technique, each node receiving or originating a message transmits a copy of it over all output links (or over a selected subset of these links). This transmission occurs after the node has checked to determine that the same message has not been transmitted previously, or that the node itself is not the destination of the message. The technique is obviously inefficient for handling a large volume of communications traffic because of the transmission redundancies.

Fixed Routing

Fixed routing algorithms specify a unique path $\pi(N_e, N_d)$ to be followed by a message. The path depends only on the current node, N_e ,

and the destination node, N_d . This is accomplished by storing a routing table in each node whose entries point the next node that corresponds to the destination node of the message. Kleinrock [K3] and Prosser [P2] have examined several of these techniques. The results indicate their efficiency in handling high volumes of traffic flow. However, they require completely reliable nodes and links which is impossible to attain in operational networks. One method of overcoming this deficiency is to designate one node in the network as the Network Routing Control Centre. This centre then collects performance information about the network operation, computes routing tables and then transmits the appropriate updated table to each node in the network.

Bifurcated Routing

These techniques, unlike fixed routing techniques, allow traffic to flow on more than one path between a given source-destination node pair. As an example, assume that the two paths $\pi_1(N_s, N_d)$ and $\pi_2(N_s, N_d)$ exist between nodes N_s and N_d . Then a message at node N_s may be routed on path $\pi_1(N_s, N_d)$ with probability p and on path $\pi_2(N_s, N_d)$ with probability $1-p$. This splitting of traffic leads to a better balance of flow throughout the network. Thus a smaller average message delays can be achieved.

Kleinrock [K3] and Prosser [P2] have investigated numerous random stochastic routing techniques and have demonstrated that they are highly inefficient in terms of message delay, but are extremely stable against small changes in the network structure. A complete and extensive study of adaptive routing techniques has also been conducted by Fultz [F5] and Boehm [B5].

Flow control routing techniques have been investigated by Davies [D6] and Kahn [K4]. The main effort in these techniques is directed towards limiting the amount of traffic generated into the network and thus prevent network congestion which leads in some instances to excessive message delay. However, hands-on experience with these techniques, i.e., their application to operational networks, is still limited.

3.3 Message Delays

As a measure of performance of store-and-forward computer communication networks, the average time, T , spent by messages in the network is usually employed. It is usually defined as follows [K3]:

$$T \triangleq \frac{1}{\gamma} \sum_{i,k} \gamma_{ik} Z_{ik} \quad (3.1)$$

where γ_{ik} is the average number of messages entering network per second from origin node i to destination node k , Z_{ik} is the average delay for these messages and, γ is the total arrival rate of messages from all external sources, given by

$$\gamma = \sum_{i,k} \gamma_{ik} \quad (3.2)$$

Using Little's result [K3], a straightforward computation of T gives

$$\gamma T = \bar{n} = \sum_i \bar{n}_i \quad (3.3)$$

where $\bar{n}$ is the average number of messages in the network and $\bar{n}_i$ is the average number of messages (in server and queue), of link i . Again from Little's result, $\bar{n}_i$ can be expressed as

$$\bar{n}_i = \lambda_i T_i \quad (3.4)$$

where λ_i is the average arrival rate at channel i and T_i is the average message delay (queueing + service) on channel i . Using eqns. (3.3) and (3.4), the average time delay T can be expressed as

$$T = \frac{1}{Y} \sum_i \lambda_i T_i \quad (3.5)$$

In order to obtain an estimate the average delay T_i on channel i , Kleinrock [K3] introduced the following assumptions:

- external poisson arrivals
- exponential message length distributions
- infinite nodal storage
- error free channels
- no nodal processing delay (other than queueing delay)
- messages at each intermediate node, are assigned a new length, at random, from an exponential distribution (the independence assumption).

With such assumptions the store-and-forward network becomes a network of queues of the Jackson type (a network of independent M/M/1 queues). Therefore the average delay T_i on channel i is given by

$$T_i = \frac{1}{\mu C_i - \lambda_i} \quad (3.6)$$

where $1/\mu$ is the average message length in bits and C_i is the capacity of channel i in bits/sec. The average arrival rate of messages, λ_i , at channel i can be computed easily for a given input traffic matrix and deterministic routing procedure.

Substituting for T_i in eqn. (3.5) from eqn. (3.6), the expression for the average time delay T will be

$$T = \frac{1}{Y} \sum_i \frac{\lambda_i / \mu}{C_i - \lambda_i / \mu} \quad (3.7)$$

In order to account for the propagation delay, p_i , in channel i and the nodal processing time, τ_i , in the node after channel i , eqn. (3.5) is modified to the form

$$T = \frac{1}{Y} \sum_i (\lambda_i T_i + \lambda_i (\tau_i + p_i)) \quad (3.8)$$

where τ_i and p_i are expressed in (sec/message) units and T_i is given by eqn. (3.6).

An expression for the average delay T_i on channel i for the case of general distribution for the length of the messages was derived by Kleinrock [K3]. The average delay associated with multi-packet messages were studied by Fultz [F5]. For the purposes of this study, the average message delay given by eqns. (3.7) and (3.8) will be used.

It is important to note that the transformation of the store-and-forward network to a Jackson type network under the previously mentioned

assumptions is valid only for the case where messages have the same priority class. The case where messages have multiple classes of priorities requires careful verification. This case is examined in the following discussion.

3.4 Priority Service Queue

In the transmission of messages (packets) in store-and-forward computer communication networks, the sending FEP keeps a copy of a transmitted message (packet) until it receives a positive acknowledgement indicating that the packet was received error-free. If no acknowledgement is received within a given period of time due to a transmission error or lack of buffer space at the receiving FEP, the message is re-transmitted. It is clear that prompt acknowledgement is desirable since buffer space in the sending node is freed rapidly, and congestion is thus avoided. Acknowledgements, therefore should be considered as high priority messages, with other messages having secondary priority.

A queueing model with infinite waiting room, poisson arrivals and general distribution of message lengths (M/G/1 queue model) will now be studied under a head-of-the-line priority service discipline with two levels. The main purposes of investigating this model are:

- to show that the use of a head-of-the-line priority service discipline based on the packet's length, when implemented in a store-and-forward network, reduces the average packet delay, particularly when the communication lines are highly utilized.
- to examine the departure process of this queue and its effect on the transformation of store-and-forward network to a Jackson type network.

In Appendix B, an expression is obtained for the stationary joint probability distribution of the number of high and low priority packets left behind in the system by a departing packet, in a M/G/1 queueing system. We will now derive an expression for the distribution of the interpacket departure times assuming for the sake of simplicity that the packet lengths are exponentially distributed. Let the mean arrival rates of class 1 and class 2 packets be λ_1 and λ_2 and let the mean length of class 1 and 2 packets be respectively equal to $1/\mu_1$ and $1/\mu_2$. Let $D(t)$, the interpacket departure time distribution be given by

$$D(t) = P_r [\text{time between departures} \leq t]$$

and its Laplace transform by

$$D^*(s) = \int_0^{\infty} e^{-st} d D(t)$$

In the same way we can define $A(t)$ and $A^*(s)$, respectively the interpacket arrival time distribution and its Laplace transform

$$A(t) = 1 - e^{-\lambda t}$$

and

$$A^*(s) = \frac{\lambda}{s + \lambda}$$

where λ represents the mean arrival rate of packets to the queue. We can also define the overall service time distribution $B(t)$ and the

service time distribution for class 1 and 2 packets $B_1(t)$ and $B_2(t)$.

Their Laplace transforms will be

$$B^*(s) = \frac{\mu}{\mu + s}$$

$$B_1^*(s) = \frac{\mu_1}{\mu_1 + s}$$

$$B_2^*(s) = \frac{\mu_2}{\mu_2 + s}$$

The Laplace transform of $D(t)$ conditioned on the fact that the system is empty will be:

$$D^*(s/\text{system is empty}) = B^*(s) \cdot A^*(s)$$

that is:

$$D^*(s/\text{system is empty}) = \frac{\lambda}{\lambda + s} \cdot \frac{\mu}{\mu + s}$$

If the system is empty of type 1 packets, which are high priority packets, then

$$\begin{aligned} D^*(s/\text{system has no type 1 packets}) &= B_2^*(s) \\ &= \frac{\mu_2}{\mu_2 + s} \end{aligned}$$

If the system has packets of both types, then

$$\begin{aligned} D^*(s/\text{system has packets of both types}) &= B_1^*(s) \\ &= \frac{\mu_1}{\mu_1 + s} \end{aligned}$$

Using the joint probability distribution* obtained in Appendix B,

$$\lim_{n \rightarrow \infty} P_r \left\{ x_n^{(1)} > 0, x_n^{(2)} \geq 0 \right\} = \frac{\lambda_1}{\mu}$$

$$\lim_{n \rightarrow \infty} P_r \left\{ x_n^{(1)} = 0, x_n^{(2)} > 0 \right\} = \frac{\lambda_2}{\mu}$$

$$\lim_{n \rightarrow \infty} P_r \left\{ x_n^{(1)} = 0, x_n^{(2)} = 0 \right\} = 1 - \rho, \quad \rho = \frac{\lambda_1}{\mu} + \frac{\lambda_2}{\mu}$$

The Laplace transform of the interpacket departure time distribution will be given by

$$D^*(s) = (1-\rho) A^*(s) \cdot B^*(s) + \frac{\lambda_1}{\mu} B_1^*(s) + \frac{\lambda_2}{\mu} B_2^*(s)$$

that is:

$$D^*(s) = (1-\rho) \cdot \frac{\lambda}{\lambda+s} \cdot \frac{\mu}{\mu+s} + \frac{\lambda_1}{\mu} \cdot \frac{\mu_1}{\mu_1+s} + \frac{\lambda_2}{\mu} \cdot \frac{\mu_2}{\mu_2+s}$$

The last equation can be written in terms of the density function $d(t)$, after taking the inverse of $D^*(s)$,

$$d(t) = \lambda \left(e^{-\lambda t} - e^{-\mu t} \right) + \frac{\lambda_2 \mu_2}{\mu} e^{-\mu_2 t} + \frac{\lambda_1 \mu_1}{\mu} e^{-\mu_1 t}$$

Knowing that the service time density $b(t)$ can be written in terms of $b_1(t)$ and $b_2(t)$ as:-

* In Appendix B, $x_n^{(1)}$ and $x_n^{(2)}$ denote the number of high and low priority messages left behind in the system by the n^{th} departing message after $t=0$ and ρ is the line utilization factor.

$$b(t) = \frac{\lambda_1}{\lambda} b_1(t) + \frac{\lambda_2}{\lambda} b_2(t)$$

it is immediately apparent that:

$$d(t) = \lambda e^{-\lambda t} \quad (3.9)$$

which means that the interdeparture times are exponentially distributed with mean equal to the mean of the input Poisson process. This result also shows that the interdeparture time distribution will always be the same regardless of the traffic intensity or the way packets are divided into two priority classes. It has been shown by Burke [B6] that the output of an M/M/1 queue with a FIFO service discipline is Poisson, i.e., the interdeparture times are independent and exponentially distributed. In the case being considered here, the independence of the interdeparture times has not been proven. Since the output of a channel in store-and-forward networks is the input of another channel, the property of Poisson arrivals which makes the analysis tractable could be destroyed.

In the analysis of networks of queues, the emphasis is placed on the total arrival process at each node from the neighbouring nodes. Because of the multiplicity of paths in and out of each node, the dependence between the interarrival times of packets at a node and their lengths is considerably reduced. This in fact represents the factor upon which the independence assumption is based. For the same reason, the independence of interdeparture time intervals for a non-preemptive priority service discipline is assumed. A simulation carried by Baskett and Muntz [B7] further upheld this assumption.

In order to verify the result given by eqn. (3.9), a GPDS model was built [M1]. In this model, the packet arrival process and the packet lengths were represented by a Poisson process and an exponential distribution respectively. Comparing the input and the output of the simulation model, the following observations were drawn:

For a period of time corresponding to the generation of 30,000 packets and for an average utilization factor of 0.643, approximately 76.27% of the total number of packets were high priority packets for which the mean delay time (average time/trans) was 70.9 units of time. The low priority packets which constitute 23.73% of the total traffic waited on the average 195.5 units of time. Of the total number of high priority packets 35.2% entered the service without queueing at all. The mean rate of arrival of packets to the queue as given by Table 3.1 is 123.3 and the standard deviation which should be equal to the mean in a Poisson process is equal to 124.3. This slight discrepancy is within the statistical limits expected of the simulation model. From the contents of Table 3.1 it is possible to plot the probability distribution function as well as the density function of the arrival process. If the same is done to the contents of Table 3.2, which contain all the information relating to the output departure process, a close fit of the two probability distributions will be observed. Notice that the mean and standard deviation of the input process (123.32 and 124.31) is to all purposes identical to the mean and standard deviation of the output pro-

QUEUE	MAXIMUM CONTENTS	AVERAGE CONTENTS	TOTAL ENTRIES	TOTAL ZERO ENTRIES	p ZERO ENTRIES	AVERAGE TIME/TRANS	AV EX-ZERO TIME/TRANS	QUEUE NUMBER
1	8	0.377	7125	2555	35.8	195.988	305.561	0
2	9	0.438	22875	8072	35.2	70.943	109.628	0

TABLE
ENTRIES IN TABLE
29999

3.1

MEAN ARGUMENT
123.329STD. DEVIATION
124.312SUM OF ARGUMENTS
3699774.000

NON-WEIGHTED

UPPER LIMIT	OBSERVED FREQUENCY	PER CENT OF TOTAL	CUMULATIVE PERCENTAGE	CUMULATIVE REMAINDER	MULTIPLE OF MEAN	DEVIATION FROM MEAN
0	235	0.78	0.7	99.2	0.000	-0.992
10	2319	7.73	8.5	91.4	0.081	-0.911
20	2168	7.22	15.7	84.2	0.162	-0.831
30	2015	6.71	22.4	77.5	0.243	-0.750
40	1795	5.98	28.4	71.5	0.324	-0.670
50	1653	5.51	33.9	66.0	0.405	-0.589
60	1577	5.25	39.2	60.7	0.486	-0.509
70	1379	4.59	43.8	56.1	0.567	-0.428
80	1268	4.22	48.0	51.9	0.648	-0.348
90	1249	4.16	52.1	47.8	0.729	-0.268
100	1090	3.63	55.8	44.1	0.810	-0.187
110	1050	3.43	59.2	40.7	0.891	-0.107
120	938	3.12	62.3	37.6	0.972	-0.026
130	878	2.92	65.3	34.6	1.054	0.053
140	811	2.70	68.0	31.9	1.135	0.134
150	701	2.33	70.3	29.6	1.216	0.214
160	654	2.18	72.5	27.4	1.297	0.294
170	667	2.22	74.7	25.2	1.378	0.375
180	627	2.07	76.8	23.1	1.459	0.455
190	557	1.79	78.6	21.3	1.540	0.536
200	505	1.68	80.3	19.6	1.621	0.616
210	460	1.53	81.8	18.1	1.702	0.697
220	432	1.44	83.2	16.7	1.783	0.777
230	411	1.37	84.6	15.3	1.864	0.858
240	330	1.10	85.7	14.2	1.945	0.938
250	345	1.15	86.9	13.0	2.027	1.018
260	289	0.96	87.8	12.1	2.108	1.099
270	250	0.83	88.7	11.2	2.189	1.179
280	259	0.86	89.5	10.4	2.270	1.260
OVERFLOW	3126	10.42	100.0	0.0	2.270	1.260
AVERAGE VALUE OF OVERFLOW		404.86				
LARGEST VALUE IN OVERFLOW		1365				

TABLE
ENTRIES IN TABLE
29999

3.2

MEAN ARGUMENT
123.326STD. DEVIATION
123.771SUM OF ARGUMENTS
3699678.000

NON-WEIGHTED

UPPER LIMIT	OBSERVED FREQUENCY	PER CENT OF TOTAL	CUMULATIVE PERCENTAGE	CUMULATIVE REMAINDER	MULTIPLE OF MEAN	DEVIATION FROM MEAN
0	249	0.83	0.8	99.1	0.000	-0.996
10	2353	7.84	8.6	91.3	0.081	-0.915
20	2081	6.93	15.6	84.3	0.162	-0.834
30	1945	6.48	22.0	77.9	0.243	-0.754
40	1878	6.26	28.3	71.6	0.324	-0.673
50	1628	5.42	33.7	66.2	0.405	-0.592
60	1508	5.02	38.8	61.1	0.486	-0.511
70	1425	4.75	43.5	56.4	0.567	-0.430
80	1316	4.38	47.9	52.0	0.648	-0.350
90	1216	4.05	51.9	48.0	0.729	-0.269
100	1084	3.61	55.6	44.3	0.810	-0.188
110	983	3.27	58.8	41.1	0.891	-0.107
120	994	3.31	62.2	37.7	0.972	-0.026
130	797	2.65	64.8	35.1	1.054	0.053
140	891	2.97	67.8	32.1	1.135	0.134
150	845	2.82	70.6	29.3	1.216	0.215
160	678	2.26	72.9	27.0	1.297	0.296
170	683	2.27	75.1	24.8	1.378	0.377
180	541	1.80	76.9	23.0	1.459	0.457
190	527	1.75	78.7	21.2	1.540	0.538
200	571	1.90	80.6	19.3	1.621	0.619
210	397	1.32	81.9	18.0	1.702	0.700
220	374	1.24	83.2	16.7	1.783	0.781
230	411	1.37	84.5	15.4	1.864	0.861
240	328	1.09	85.6	14.3	1.945	0.942
250	317	1.05	86.7	13.2	2.027	1.023
260	363	1.21	87.9	12.0	2.108	1.104
270	280	0.93	88.8	11.1	2.189	1.185
280	259	0.86	89.7	10.2	2.270	1.265
OVERFLOW	3076	10.25	100.0	0.0	2.270	1.265
AVERAGE VALUE OF OVERFLOW		405.73				
LARGEST VALUE IN OVERFLOW		1000				

cess (123.32 and 123.77). A close fit of the two probability distributions was also observed for other values of the utilization factor.

Another simulation experiment was carried out to study the effect of using dynamic assignment of message priorities on the average message delay, T , where T is given by eqn. (3.5) as:

$$T = \frac{1}{Y} \sum \lambda_i T_i$$

and the summation is carried over all network links. A three node fully connected network with line capacities of 50 kbits/sec throughout and average packet length of 400 bits was simulated. For simplicity the traffic matrix is assumed to be uniform and symmetric. This allows the same line utilization factors to be used for the three channels. Two nodal service disciplines were examined: first-in-first-out (FIFO) service discipline and non-preemptive priority service discipline with two levels. The assignment of the two priority levels is carried dynamically by choosing an optimal threshold level based on the rate of input traffic to each channel [D7].

The ratio, $(TAQ)_p / (TAQ)_{np}$, of the total average queueing time under a priority service discipline over the total average queueing time under a FIFO service discipline was calculated as a function of the line utilization factor, ρ , and the result is shown in Fig. 3.1.

For low line utilizations the difference between the total average queueing time with the without priorities is negligible. As the

Figure 3.1

line utilization increases, however, it becomes more advantageous to consider a priority scheme. That is, for high line utilization the reduction in queueing time obtained using a priority scheme is dramatic.

CHAPTER 4

THE RESOURCE ALLOCATION MODEL

4.1 Introduction

This chapter is devoted to the formulation of the resource allocation model. The primary design objective is the allocation of file and capacity resources to minimize the combined storage and communication cost, and at the same time satisfy predetermined delay and reliability constraints.

The cost function is formulated in Section 4.2. It includes file storage cost and communication line capacity cost. The results of Chapters 2 and 3 are used respectively in Sections 4.3 and 4.4 to formulate the delay and availability constraints of the resource allocation model. In Section 4.5, it is shown that the resource allocation model is, in fact, a generalization of previously developed file allocation and capacity assignment models. Section 4.6 discusses file security and privacy considerations.

4.2 The Cost Function

As a function of the channel capacities, the communications cost is usually expressed in one of the following forms:

- Linear cost-capacity
- Concave cost-capacity
- Discrete cost-capacity.

In practice, line capacities are available in a number of discrete values (for example, 2.4, 4.8 ..., k bit/sec). For this reason the communications cost will be represented by a discrete cost-capacity function in the resource allocation model.

Assume that for each of the $2m$ unidirectional (half-duplex) links of the network, an N discrete values $h_1, h_2, \dots, h_N$ of capacities are available. From commercial data a cost matrix D may be developed such that d_{ij} is the cost (dollars/month) of link i if it is assigned capacity j , where $i = 1, 2, \dots, 2m$, $j = 1, 2, \dots, N$. Choice of link capacities is equivalent to the specification of a binary $2m \times N$ capacity allocation matrix Y , where

$$Y = \{y_{ij}\}$$

such that $y_{ij} = 1$ if link i has capacity h_j , i.e., if $C_i = h_j$
 $= 0$, otherwise

and

$$\sum_{j=1}^N y_{ij} = 1$$

It is easily seen that the monthly communications cost C_c is given by

$$C_c = \text{tr } D^T Y \quad (4.1)$$

Storage Cost

The formula used to calculate the storage cost varies from one computer system to another depending on the type and volume of service, the data processing applications as well as the accounting procedures employed by the system administration. To provide an example of a storage cost formula, the following cost figures representing the way the storage cost is estimated in a commercial computer utility:

The utility offers APL (scientific programming and shared file access) service in both time-sharing and batch processing modes of access to a number of users. An IBM 360/70 computer system is used to provide the service. The file control programs are stored in work spaces; the size of each work space is equal to 77k bytes while the disc data files are available for on-line access. Storage cost for work spaces is equal to 50 cents/work space-day. Disc data files are counted in 100K bytes units. The total number of units-day are added at the end of the month to form the total units-day for the month and the storage cost is calculated as following:

the first 10 units are free of charge

next 90 units cost 80 cents/unit

next 150 units cost 65 cents/unit

extra units cost 50 cents/unit

In the resource allocation model, the storage cost will be assumed to be given by an average monthly cost (in dollars/bit-month). The storage cost in this case will depend on the average monthly length of the file.

Suppose that M distinct files are to be used. Each must be placed at one node at least, but multiple copies may also be assigned. Let ℓ_j be the average length in bits of file j , and b_i be the average monthly cost of storing one bit at node i . Choice of network file locations is equivalent to the specification of a binary $n \times M$ file allocation matrix X where

$$X = \{x_{ij}\}$$

and $x_{ij} = 1$ if node i contains file j .

$= 0$, otherwise

$$i = 1, 2, \dots, n, \quad j = 1, 2, \dots, M$$

File storage cost is given by

$$C_s = \underline{b}^T X \underline{\ell} \quad (4.2)$$

where

$$\underline{b}^T = (b_1, b_2, \dots, b_n)$$

$$\underline{\ell} = (\ell_1, \ell_2, \dots, \ell_M)$$

The sum of eqns. (4.1) and (4.2) gives the total cost function.

4.3 The Delay Constraint

Consider a pair of nodes i and k in the network. It is desired to express the total average number of messages, γ_{ik} , entering the network per second with origin i and destination k as a function of the file

allocations x_{ij} 's. As a matter of convenience, we define first the following quantities:

u_{ij} = total average query traffic in messages per second generated from node i as a result of accessing file j

u'_{ij} = return traffic of u_{ij}

v_{ij} = total average update traffic in messages per second generated from node i as a result of accessing file j

v'_{ij} = return traffic of v_{ij}

If a copy of file j is stored in node k and no other copy of the file exists in the network, then the traffic, γ_{ik} , will be given by

$$\gamma_{ik} = u_{ij} + v_{ij} + u'_{kj} + v'_{kj}$$

As a function of the file allocation variables, x_{ij} and x_{ik} , the traffic γ_{ik} can be expressed as

$$\gamma_{ik} = \sum_{j=1}^M [(u_{ij}(1-x_{ij}) + v_{ij})x_{kj} + (u'_{kj}(1-x_{kj}) + v'_{kj})x_{ij}] \quad (4.3)$$

Figure 4.1 illustrates the components of the query and update traffic in both directions.

Equation (4.3) does not take into consideration the possibility that identical copies of file j may exist in other network nodes. In this case a probability of access should be defined for each copy with respect to each node. This probability of access may depend on factors

FIGURE 4.1

TRAFFIC FLOW COMPONENTS

not accounted for in the resource allocation model. Examples are the administrative and accounting factors associated with the computer centres providing or receiving the file access service. It is assumed that the probability of access is equal for all identical file copies and all network nodes. Therefore, the query traffic in eqn. (4.3) will be divided by the number of copies, q_j , of file j where q_j is given by

$$q_j = \sum_{i=1}^n x_{ij}$$

and eqn. (4.3) will be modified to the form

$$\gamma_{ik} = \sum_{j=1}^M \left[\left(\frac{u_{ij}}{q_j} (1-x_{ij}) + v_{ij} \right) + \left(\frac{u'_{kj}}{q_j} (1-x_{kj}) + v'_{kj} \right) x_{ij} \right] \quad (4.4)$$

The total arrival rate of messages, γ , from external sources is the sum of γ_{ik} 's over all node pairs (i,k) in the network:

$$\begin{aligned} \gamma &= \sum_{(i,k)} \gamma_{ik} \\ &= \sum_{i=1}^n \sum_{j=1}^M [(u_{ij} + u'_{ij})(1-x_{ij}) + (v_{ij} + v'_{ij})(q_j - x_{ij})] \end{aligned} \quad (4.5)$$

In eqn. (4.5) the query traffic given by u_{ij} and u'_{ij} is not considered as network traffic if file j is stored in node i since the file query access process in this case is limited to the local file.

Let S_ℓ be the set of source nodes whose originated traffic flows over link ℓ enroute to a (disjoint) set of destination nodes S'_ℓ . For a given deterministic routing procedure, S_ℓ and S'_ℓ are known for all links. Thus the total traffic λ_ℓ in messages per second over unidirectional link ℓ is given by

$$\lambda_\ell = \sum_{(i \in S_\ell, k \in S'_\ell)} \gamma_{ik}$$

and γ_{ik} is given by eqn. (4.4). The average delay, T , expressed by eqns. (3.7) and (3.8) is thus formulated as a function of capacity and file allocations. The delay constraint is given by the following inequality:

$$T \leq \bar{T} \quad (4.6)$$

where $\bar{T}$ is the maximum allowed average delay per message.

4.4 The Reliability Constraint

In this section, expressions are derived for the file availability with respect to each of the network nodes as a function of the file allocation variables x_{ij} 's. Since network elements (nodes and links) are not perfectly reliable, it is expected that an increase in the number of identical copies of a given file would lead to a higher availability of the file to network nodes. Thus by specifying a minimum file availability requirement, a lower bound on the number of copies of each file in the network is determined.

Let a_{ij} represent the availability of file j with respect to node i . Obviously a_{ij} depends on both the number of identical copies of file j , as well as their locations in the network. Assume that file j has two identical copies stored in nodes k_1 and k_2 in the network and that both copies are accessible to users of node i . Recall from Section 2.2 that the availability of a single-copy file stored in any network node k with respect to remote node i depends on the terminal reliability r_{ik} . It follows that for the case of two copies of a file stored in two different nodes (k_1 and k_2) the availability a_{ij} is a function of the reliability $r_{ik_1k_2}$ which represents the probability of maintaining communication between node i and any of the two nodes k_1 and k_2 . In networks of general topologies, it is extremely difficult, if not impossible, to express $r_{ik_1k_2}$ as a function of r_{ik_1} and r_{ik_2} in an analytical closed form. The same remark applies to files with multi-copies in the network.

The above discussion illustrates the difficulties encountered in obtaining analytical expressions for the file availability a_{ij} as a function of the file allocation variables x_{ij} 's.

To obtain approximations of the file availability, the following two measures are proposed:

Measure I

The availability of file j to node i is given by the approximate value a'_{ij} , where a'_{ij} is a function of file locations, node component reliability and link-reliability as follows:

$$a'_{ij} = a_{ui} a_{ci} \left[1 - \left\{ (1 - a_{li}) + a_{li} \prod_{\substack{k=1 \\ k \neq i}}^n (1 - r'_{ik} x_{kj}) \right\} (1 - a_{si} x_{ij}) \right] \quad (4.7)$$

$$r'_{ik} = r_{ik} a_{lk} a_{ck} a_{sk} \quad (4.8)$$

The terms of eqns. (4.7) and (4.8) are as defined in Section 2.2. Under the assumption that a_{li} is equal to 1, eqn. (4.7) can be simplified to the form:

$$a'_{ij} = a_{ui} a_{ci} \left[1 - \prod_{k=1}^n (1 - r'_{ik} x_{kj}) \right], \quad r'_{ii} = a_{si} \quad (4.9)$$

The expression for a'_{ij} given by eqn. (4.7) or its simplified form of eqn. (4.9) represents the first approximate measure of the exact file availability, a_{ij} .

It is important to note that the file availability, a'_{ij} given by eqn. (4.7) or eqn. (4.9) is equal to the actual file availability a_{ij} in the following two cases:

- 1) A single-copy assignment of file j in the network.
- 2) A star network topology in which node i is the central node (see Fig. 4.2).

In general, measure I is characterized by the following property.

For any network with general topology, the availability of file j with respect to node i , given by measure I as a'_{ij} , specifies an upper bound to the actual file availability, a_{ij} , for any given node-file assignment, i.e.,

FIGURE 4.2
STAR TOPOLOGY

$$a'_{ij} \geq a_{ij} \quad \text{for } q_j = 1, 2, \dots, n.$$

Proof of the above property is given in Appendix C.

Measure II

The availability of file j to node i is given by the approximate value a''_{ij} , where a''_{ij} is a function of file locations, node components reliability and link reliability as follows:

$$a''_{ij} = a'_{ij} [1 - F_i(1 - a_{si}x_{ij})] \quad (4.10)$$

F_i represents the probability that node i is completely disconnected (isolated) from the network and a'_{ij} is given by eqn. (4.7).

The expression for a''_{ij} given by eqn. (4.10) represents the second approximate measure of the exact file availability, a_{ij} .

In order to test the validity of the proposed approximate measures for file availability, numerical simulations were conducted. A typical sample of the results is given in the following example. Consider the 15-node network shown in Fig. 4.3. The value associated with each link represents the reliability of the link while the node components are assumed to be perfectly reliable. The terminal reliability, r_{ik} , for any pair of nodes i and k can be computed using one of the techniques mentioned in Section 2.3 and is given below Fig. 3.2 for all pairs of nodes.

Assume now that a file with one or more copies is stored in one or more network nodes and is accessible to all network nodes. The avail-

Figure 4.3

5-node Network

$$r_{12} = 0.97956$$

$$r_{13} = 0.962010$$

$$r_{14} = 0.98496$$

$$r_{15} = 0.673407$$

$$r_{23} = 0.96445$$

$$r_{24} = 0.97636$$

$$r_{25} = 0.675115$$

$$r_{34} = 0.96436$$

$$r_{35} = 0.7$$

$$r_{45} = 0.675052$$

ability of the file to node 1 calculated using the approximate measures as well as the exact availability is reported in Table 4.1 for different node-file assignments. It is obvious that the difference between the values of the exact availability and those obtained using the second approximate measure, a''_{1j} , is for most practical considerations, negligible. For a single file copy assignment the first measure, a'_{1j} , is equal to the exact file availability. As the number of copies increases, the first measure tends to overestimate the file availability.

The availability of the file with respect to node 2 calculated using the approximate measures and the exact method is given in Table 4.2. For a single copy of the file in the network, the availability, a'_{2j} is equal to a_{2j} while a''_{2j} appears to underestimate the file availability. As the number of copies of the file increases, the first measure a'_{2j} tends to overestimate the file availability while the second measure, a''_{2j} , asymptotically approaches the exact value of file availability. It should be noted that the connectivity of node 2, while less than that of node 1, is still relatively strong.

The observations made about the file availability to node 2 are emphasized by examination of the availability of the file to node 5 as shown in Table 4.3. For a single copy of the file in the network, the availability a'_{5j} is equal to a_{5j} while a''_{5j} clearly underestimates the file availability. Again as the number of copies increases, a''_{5j} asymptotically approaches the exact value a_{5j} while a'_{5j} tends to be deceiving as it estimates the file availability at much higher values than the

TABLE 4.1
File availability to node 1

FILE COPIES	MEASURE I a'_{ij}	MEASURE II a''_{ij}	EXACT AVAILABILITY a_{ij}
1 copy in node 2	0.97956	0.97564	0.97956
2 copies in nodes 2 and 4	0.99969	0.995693	0.9936
3 copies in nodes 2, 3 and 4	0.999988	0.995988	0.996
4 copies in nodes 2, 3, 4, and 5	0.999997	0.995997	0.996

TABLE 4.2
File availability to node 2

FILE COPIES	MEASURE I a'_{2j}	MEASURE II a''_{2j}	EXACT AVAILABILITY a_{2j}
1 copy in node 1	0.97956	0.96486	0.97956
2 copies in nodes 1 and 4	0.999516	0.98452	0.985
3 copies in nodes 1, 3 and 4	0.999982	0.984983	0.985
4 copies in nodes 1, 3, 4 and 5	0.9999944	0.984994	0.985

TABLE 4.3
File availability to node 5

FILE COPIES	MEASURE I a'_{5j}	MEASURE II a''_{5j}	EXACT AVAILABILITY a_{5j}
1 copy in node 2	0.675115	0.47258	0.675115
2 copies in nodes 1 and 2	0.89394	0.62573	0.6755
3 copies in nodes 1, 2 and 4	0.96552	0.675865	0.679
4 copies in nodes 1, 2, 3 and 4	0.98966	0.69276	0.7

2

OF/DE

4

actual availability values. It is noted however that the connectivity of node 5 is smaller than that of node 1 or node 2. Since the file availability to any network node never exceeds its probability of connection to the network (assuming that the file is not stored in the node), measure I tends to overestimate the file availability for nodes with low connectivity. In order to correct this overestimation, measure I is multiplied by the node connectivity which represents an upper bound to the file availability. Multiplying measure I by the node connectivity results in measure II as indicated by eqn. (4.10).

The results shown in Tables 4.1, 4.2 and 4.3 for the 5-node network of Fig. 4.3 indicate the validity of using measure I to represent the file availability for nodes that have strong connectivity. For nodes with low probability of connection to the network, measure I can still be used if the file being assigned is expected to have a single copy in the network. Otherwise measure II should be employed as it asymptotically approaches the actual value of the file availability with the increase in the number of copies of the file.

The average value of the availability of file j to network nodes, denoted by α_j is given by:

$$\alpha_j = \frac{\sum_{i=1}^n w_{ij} a_{ij}}{\sum_{i=1}^n w_{ij}} \quad (4.11)$$

where w_{ij} is the total traffic in messages/sec generated by node i in accessing file j , i.e.,

$$w_{ij} = u_{ij} + u'_{ij} + v_{ij} + v'_{ij}$$

Thus α_j is equal to the average value of the weighed file availabilities, a_{ij} ($i = 1, 2, \dots, n$).

The file availability constraint states that α_j , $j = 1, 2, \dots, M$ should be greater than or equal to a specified value α'_j , i.e.,

$$\alpha_j \geq \alpha'_j, \quad j = 1, 2, \dots, M$$

Problem Summary

The following data are given: *

n :	number of network nodes	
$2m$:	number of (unidirectional) links	
N :	number of link capacity alternatives	
M :	number of files	
D :	capacity cost matrix	($2m \times N$)
U, U' :	query traffic matrices	($n \times M$)
V, V' :	update traffic matrices	($n \times M$)
$\underline{b}$:	storage cost vector	(n)
$\underline{l}$:	file length vector	(M)
$\underline{\xi}$:	link reliability vector	($2m$)
$\underline{n}$:	node reliability vector	(n)
μ^{-1} :	average message length	scalar
$\bar{T}$:	maximum acceptable average delay	scalar
$\underline{\alpha}'$:	vector of minimum acceptable file availabilities $\alpha'_1, \dots, \alpha'_M$	(M)

The design problem is to choose the matrices

X file allocation matrix ($n \times M$)

Y capacity allocation matrix ($2m \times N$)

to achieve

$$(C_c + C_s)_{\min} = \min_{\{X, Y\}} [\text{tr } D^T Y + \underline{b}^T X \underline{e}]$$

subject to constraints of:

(a) Delay:

$$T \leq \bar{T} \quad (T \text{ is given by eqn. (3.7)})$$

(b) Availability:

$$\alpha_j \geq \alpha_j^l, \quad j = 1, 2, \dots, M \quad (\alpha_j \text{ is given by eqn. (4.11)})$$

4.5 Related Models

In this section, it is shown that file allocation models as well as capacity-flow assignment models previously developed ([C2], [C6], [G1]) can be derived as special cases from the general resource allocation model discussed in this chapter.

4.5.1 The Unconstrained File Allocation Model

In this model, a fully connected network topology is considered. Thus a link exists between any two nodes i and k in the network. A linear communication cost function is considered while no delay and reliability

constraints are imposed. It is desired to assign copies of different files to the network nodes so that the total communication and storage cost is minimized.

If we assume that the traffic over link (i,k) in the direction i to k is given by λ_{ik} (messages/sec), then for a fully connected topology, the equality:

$$\gamma_{ik} = \lambda_{ik} \quad (4.12)$$

holds. From eqn. (4.4), λ_{ik} can be written as

$$\lambda_{ik} = \sum_{j=1}^M \left[\left(\frac{u_{ij}}{q_i} (1-x_{ij}) + v_{ij} \right) x_{kj} + \frac{u'_{kj}}{q_i} (1-x_{kj}) + v'_{kj} \right] x_{ij} \quad (4.13)$$

and the corresponding communication cost, C_c , will be given by:

$$C_c = \sum_{\substack{(i,k) \\ i \neq k}} \frac{1}{\mu} d_{ik} \lambda_{ik} \quad \$/\text{month}$$

where d_{ik} is the cost in \$/month of transmitting 1 bit over link (i,k) and $1/\mu$ is the average message length in bits. The storage cost C_s is given by eqn. (4.2). Thus the total cost is equal to

$$C_c + C_s = \sum_{\substack{(i,k) \\ (i \neq k)}} \left(\frac{1}{\mu} d_{ik} \lambda_{ik} \right) + \underline{b}^T \times \underline{x} \quad (4.14)$$

Because of the linearity of the cost function and the absence of the delay constraint, files may be allocated separately, i.e., the optimization problem can be decomposed into M subproblems, where M is the number of files.

The above model was adopted by Casey [C6] who developed an algorithm that facilitates the search for optimal solutions to eqn. (4.14).

4.5.2 File Allocations in Fully Connected Topologies With Delay Constraints

In this model, a fully connected topology and a linear communication cost function are assumed [C2]. Thus eqns. (4.12) and (4.13) can be applied to describe the traffic over each network link (i,k) . Under the assumption that links are assigned fixed capacities and c_{ik} represents the capacity of link (i,k) in bits/sec, then the average message delay, T_{ik} , over link (i,k) will be given by

$$T_{ik} = \frac{1}{\lambda_{ik}} \frac{\lambda_{ik}/\mu}{c_{ik} - \lambda_{ik}/\mu}$$

under the assumptions stated in Section 3.3. It is required that this average delay does not exceed a given required delay $\bar{T}_{ik}$,

$$T_{ik} \leq \bar{T}_{ik} \quad \text{for all links } (i,k) \quad (4.15)$$

The total cost function in the model is the same as that given by eqn. (4.14). It should be noted that while the links are assigned fixed capacities, the communication cost is estimated based on the volume of data transmitted over the different links.

4.5.3 Capacity and Flow Assignment Models

Corresponding to each file assignment, X , a traffic flow matrix (defined as the matrix whose entry (i,j) is equal to the traffic γ_{ij}) can be computed using the set of node-file update and retrieval demands. For a given network topology and traffic flow matrix, the following three problems are encountered in the design of message switching computer communication networks.

1. The capacity assignment problem

This is the problem of assigning link capacities for a given routing scheme to minimize the communication cost and at the same time satisfy the delay constraint.

2. The flow assignment problem

This is the problem of assigning link flows (routing schemes) for a given link capacity assignment to minimize the average message delay.

3. The capacity-flow assignment problem

This is the problem of simultaneously assigning link capacities and link flows (routing scheme) in order to minimize the communication cost and at the same time satisfy the delay constraint.

The three problems mentioned above have been examined by Gerla [G1] who developed computationally tractable and efficient techniques for solving them. Application of some of these techniques to the resource allocation model will be examined in Chapters 6 and 7. It should

be noted that the flow assignment problem and the capacity-flow assignment problem are not special cases of the model developed in this study since the model assumes a predetermined traffic routing scheme.

4.6 File Security and Privacy Considerations

In a distributed information network environment, file security and privacy considerations may prohibit the storing of a given file in certain nodes. This is because the security measures which are imbedded in the file system can be overridden, in most cases, by the local operating system of the node in which the file is stored. The owner of a data file with sensitive information may thus hesitate to store the file in nodes under the control of a competitor or in nodes where strict security measures are not guaranteed.

The security constraints can be imposed in the resource allocation model developed in this study by simply restricting the allocation of file copies to a "permissible" set of nodes. This can be expressed formally as follows:

Define the set Ω_j to contain the "non-permissible" nodes with respect to file j (nodes which cannot be assigned a copy of file j because of security and/or privacy reasons). In the resource allocation model, a security constraint is introduced if $\Omega_j \neq \phi$. An expression for such a constraint will be given by

$$\{x_{ij} = 0/\text{for all } i \in \Omega_j\}, \quad j = 1, 2, \dots, M$$

If the set of non-permissible nodes contains k nodes with respect to a given file ($k < n$), then the size of the file assignment problem will be reduced to the set of 2^{n-k} combinations which is smaller than the set of 2^n combinations of the original problem.

CHAPTER 5

BIVALENT PROGRAMMING TECHNIQUES

5.1 Introduction

The resource allocation model derived in the previous chapter describes a constrained optimization problem. The optimization variables are the binary elements of the two allocation matrices X and Y , representing the assignment of file copies to network nodes and capacities to network links. The objective (cost) function is linear in terms of the elements of X and Y while the constraint inequalities are nonlinear (polynomial with constant coefficients). The resource allocation model thus falls into the class of nonlinear bivalent programming models.

A number of integer programming techniques have been developed recently in order to obtain optimal solutions to this class of problem. These techniques can be considered, in general, as variations of one of two approaches; namely, the linearization approach and the direct nonlinear approach.

In the linearization approach, the nonlinear constraint equations are replaced by an equivalent set of linear inequalities [W4]. This is accomplished by replacing each nonlinear (polynomial) term of the constraint equations by a new binary variable and additional two constraint equations relating (in a linear form) the variables of the replaced terms to the newly introduced binary variable. The nonlinear integer programming problem is thus transformed into an equivalent linear integer pro-

gramming problem which can be solved using any of the techniques described in Section 5.2.

In the direct nonlinear programming approach, the linearization process is avoided and the search for feasible solutions is conducted directly on the nonlinear set of inequalities (constraints). A discussion of this approach is presented in Section 5.3.

The algorithmic treatment of integer programs, both linear and nonlinear, is built upon three key notions: *separation*, *relaxation* and *fathoming*. The meaning of these notions is explained in the following:

Separation

For any optimization problem, (P) , let $F(P)$ denote its set of feasible solutions. Problem (P) is said to be separated into k subproblems $(P_1), (P_2), \dots, (P_k)$ if the following conditions hold:

- . Every feasible solution of (P) is a feasible solution of exactly one of the subproblems $(P_1), (P_2), \dots, (P_k)$.
- . A feasible solution of any of the subproblems $(P_1), (P_2), \dots, (P_k)$ is a feasible solution of (P) .

Separation in many algorithms is used to divide the original problem into a number of subproblems (referred to as candidate problems) which can be conquered separately. The best (feasible) solution found among the solutions of the candidate problems will be the optimal solution of the original problem.

Relaxation

Any constrained optimization problem (P) can be relaxed by loosening its constraints, resulting in a new problem (P_R) . For example, one may simply omit some of the constraints of (P) . The only requirement for (P_R) to be a valid relaxation of (P) is: $F(P) \subseteq F(P_R)$. This implies the following relationships:

- If (P_R) has no feasible solutions, then the same is true of (P) .
- The minimal value of (P) is no less than the minimal value of (P_R) .
- If an optimal solution of (P_R) is feasible in (P) , then it is an optimal solution of (P) .

The reason for relaxing the original problem is that the resulting relaxed problem is usually easier to solve with the hope that (P_R) will yield an optimal solution of (P) or at least as close as possible to it.

Fathoming

The ultimate objective of solving a candidate problem, (P_i) , of the original problem (P) is to determine whether its feasible region $F(P_i)$ may possibly contain an optimal solution of (P) and, if so, to find it. If it can be ascertained by means of some test, that $F(P_i)$ cannot contain a feasible solution better than the best solution yet found, then (P_i) can be dismissed from further considerations. The criteria used for such a test is called the *fathoming criteria* and is generally used to enhance the computational efficiency of the algorithm. There is no general rule for selecting the *fathoming criteria*. The selection rather depends on the structure of the optimization problem.

The notions of *separation*, *relaxation* and *fathoming* are utilized, to varying degrees, in almost every integer programming algorithm. This fact will be further illustrated in Section 5.3.

5.2 Linear Integer Programming Techniques

Linear integer programming algorithms can be classified in general into four categories: *Enumerative Algorithms*, *Benders Decomposition Algorithms*, *Cutting plane algorithms*, and *Group Theoretic Algorithms*. In a recent state-of-the-art survey of integer programming algorithms, Geoffrion and Marsten [G2] presented a unifying framework to facilitate the understanding of the main concepts underlying these four categories of algorithms. The survey also contains a number of currently operational computer-coded algorithms along with computational experience data. For the purposes of this study, it would be sufficient to briefly mention these algorithms. More details are available in the cited references.

Enumerative algorithms [B8], [L2], [A1] include all those algorithms which come under the popular headings of "implicit enumeration" and "branch and bound". Such algorithms methodically search the set of possible integer solutions in such a way that not all possibilities need be considered individually. Operational algorithms based on enumeration methods include: OPHELE MIXED (CDC 6600), RIP30 (IBM 360/91) and MARI-SABETH (UNIVAC 1108).

Benders decomposition algorithms [B9], [G3], [G4] are essentially enumerative techniques that rely on relaxation of the original problem

constraints. These algorithms are particularly useful for solving mixed linear programming problems. An example of operational algorithms that use Bender Decomposition techniques is the MIDAS-2 algorithm (IBM 360/65).

Cutting Plane algorithms [B10], [G5], [Y1]: historically, this was the first general approach taken to solving integer programs. It relies on successively improved relaxations with no use whatever made of separation. Computational experience with this techniques has been largely confined to very small problems.

Group Theoretic algorithms [S2]: in this approach, the original problem is transformed into an equivalent group theory problem, using some relaxation rules. The choice of these rules depends on the algebraic structure of the problem. Algorithms derived from Group Theory (such as shortest path algorithms) are then utilized to solve the equivalent problem. Computational experience with this technique is limited to small and medium size problems.

5.3 Nonlinear Integer Programming Techniques

Approaches for solving polynomial and other nonlinear programs directly, without transforming them to linear integer programs have been proposed by Lawler and Bell [L3], Mao and Wallingford [M2], Taha [T1] and Hanson [H4]. Most of the proposed techniques are tailored to solve particular rather than general problem structures by selecting *fathoming criteria* that are suitable for a given structure.

A more recent approach has been developed by Hammer [H5] for the direct solution of nonlinear integer programming problems. The approach, known as the nonlinear branch and bound technique, combines boolean and branch and bound techniques to solve problems of general structure. An algorithm developed using this approach will be used to solve the resource allocation problem. The methodology of the approach is outlined in the following discussion. A detailed presentation of the algorithm is provided in [H5].

The linear branch and bound technique relies on two important notions: the *characteristic function* and the *directrix* of a given set of constraint equations. The *characteristic function* of a system of inequalities is a boolean function of the bivalent variables of the system. The value of this function is equal to 1 for all those values of the variables corresponding to a feasible solution. The *directrix* of a system of constraints is a boolean function of the bivalent variables of the system which specifies an upper bound of the characteristic function. In other words, a zero value of the directrix implies a zero value of the characteristic function. Thus a necessary condition for the system to be consistent is that the directrix is equal to 1. Methods of obtaining the characteristic function and the directrix for a set of inequalities are given in [H6].

The basic idea of the procedure followed in the nonlinear branch and bound technique is summarized in the following steps:

- a) Construct a directrix of the set of constraints. This is equivalent to a *relaxation* of the original problem (P) since the directrix specifies an upper bound for the characteristic function.
- b) Use the directrix to decompose the original problem (P) into a number of subproblems $(P_1), (P_2), \dots, (P_k)$, each of them being obtained from the original problem (P) by fixing the value of a part of the variables and corresponds to an elementary conjunction of the directrix. This is equivalent to *decomposing* the original problem into a number of subproblems.
- c) Single out one of the possible subproblems, say (P_i) .
- d) Examine the subproblem (P_i) and its complementary problem $(\bar{P}_i)$, where $(\bar{P}_i)$ is the subproblem in which all the variables that were fixed in (P_i) are negated. This is equivalent to applying *separation* to the original problem.
- e) Repeat the above procedure until all possibilities are exhausted "implicitly".

To illustrate the way branching is done in the nonlinear branch-and-bound algorithm, consider the following example. Assume that an optimization problem has six variables, $(x_1, x_2, x_3, x_4, x_5, x_6)$. Also assume that the directrix of the problem is given by

$$(x_1 \bar{x}_2) \cup (x_1 x_4 x_6) \cup (\bar{x}_2 \bar{x}_3) \cup (\bar{x}_3 x_4 x_6) \cup (x_2 x_4 \bar{x}_6)$$

Since the directrix contains five elementary conjunctions, the original optimization problem is divided into five subproblems; each subproblem is obtained by fixing the values of the variables of an elementary conjunction. Branching is done based on fixing the values of the variables of an elementary conjunction, say $\bar{x}_3 x_4 x_6$, and its complement $\overline{\bar{x}_3 x_4 x_6}$. Fixing the values of the variables of an elementary conjunction

leads to a new optimization problem. The directrix of the new problem is then obtained and the procedure is repeated, i.e., the new problem is divided into a number of subproblems; each subproblem is obtained by fixing the variables of an elementary conjunction. The value of the objective function is calculated at each branch point. The search over a given branch is stopped when no further improvement can be attained over this branch. The algorithm stops after "implicitly" searching all branches and the optimal cost value is obtained.

The above algorithm is very similar to branch-and-bound techniques used in linear programming. The significant difference is that the number of variables according to which branching is done is no longer 1, allowing a reduction in the number of subproblems and further enhancing the computational efficiency of the algorithm.

A computer study by Taha [T1] indicates that neither the direct nonlinear nor the transformed linear approach can invariably claim superiority over the other, but that the effectiveness of each depends on the specific problem to be solved. Another study conducted by Pierce [P3] indicates that for both approaches, the computing times tend to increase, in general, approximately exponentially with the number of variables.

For large size networks, solving the resource allocation problem using integer programming techniques tends to be economically prohibitive in terms of the needed computer time and storage resources. This motivated the search for heuristic methods to allow for computationally tract-

able solutions as will be explained in Chapters 6 and 7. However, integer programming techniques are still needed in order to provide a basis for evaluating the solutions yielded by the heuristic methods.

The prime reason for using the direct nonlinear integer programming approach in the resource allocation problem is dictated by the particular structure of the model. The inequality constraints contain a huge number of polynomial terms and their transformation into the linear form leads to excessive storage demands (this is due to the generation of additional constraint equations and replacement variables).

CHAPTER 6

HEURISTIC SOLUTION

6.1 Introduction

In Chapters 4 and 5, it was shown that the problem of optimal file allocation in distributed information networks can be analytically formulated and solved as a bivalent nonlinear programming problem. However, because of the combinatorial structure of the problem, the search for an optimal solution even for moderate size networks supporting a small number of files requires a substantial computational effort and a rather large computer storage space. This difficulty is often encountered in many large scale problems that arise from real situations. Whenever deterministic techniques prove to be computationally intractable for solving these problems, one is forced to resort to near optimal solutions as obtained by combining formal arguments with heuristic reasoning. A general discussion of the meaning and implications of heuristic reasoning is provided in Krone [K2] and Beltrami [B4]. For the purpose of this study, it is sufficient to think of heuristic reasoning as meaning that one brings to bear as much intuition and as many plausible arguments as possible on problems which are either computationally intractable or for which an inadequate theory exists. In the resource allocation problem considered here, the main purpose of the heuristic approach is to find computationally tractable solutions.

In a recently published work, Eswaran [E1] investigated the problem of optimal file allocation in computer networks using Casey's model (see Section 4.5). By applying "Set Covering" Techniques [K5], he proved that the unconstrained file allocation problem is isomorphic to other established classes of combinatorial problems such as vertex coloring problem and the Hamiltonian cycle problem. It has been established that solving these problems using deterministic techniques is extremely difficult. Eswaran's findings thus supports the argument that in solving the optimal file allocation problem, emphasis should be placed on finding good heuristics rather than on improving existing deterministic techniques. The computational experience presented in Chapter 7 will further support this view.

Section 6.2 discusses the methodology of the heuristic algorithm. Steps of the algorithm are given in Section 6.3. An example to illustrate the algorithm is provided in Section 6.4.

6.2 Prologue to the Heuristic Algorithm

The proposed heuristic algorithm starts by generating a number of initial file allocations. An initial allocation of files is obtained by minimizing the storage cost subject to the file availability constraint. Because of the linearity of the storage cost function and since the availability of each file is expressed separately as an inequality constraint, the problem of generating initial solutions can be decomposed into M integer programming subproblems. The solution of these M integer pro-

gramming subproblems generates a set of initial solutions; each of these initial solutions satisfies the set of M availability constraints.

Using the routing tables, the set of link flows can be determined for a given initial file allocation. Link capacities can then be assigned by minimizing the communication cost subject to the delay constraint (the discrete capacity assignment problem). Two techniques have been developed to solve this problem. These are the Dynamic Programming [DP] and Lagrangian Decomposition [LD]. Because these techniques are of central importance to the heuristic algorithm, part of the following discussion is devoted to illustrate them.

Discrete Capacity Assignment Using Dynamic Programming

For each file allocation, a set of flows f_ℓ , $\ell = 1, 2, \dots, 2m$ can be computed using the routing tables. It is required that capacities be assigned to network links so that the communication cost, C_c , (given by eqn. (4.1)) is minimized subject to the delay constraint expressed by eqn. (4.6).

The problem is divided into $2m$ subproblems, called stages, where $2m$ is the total number of links in the network. At stage k , a subnet with k arcs is considered and all the *non-dominant* solutions (the term *non-dominant* will be defined later) with delays varying from 0 to $\bar{T}$ are found. The final solution to the problem is found after solving the final stage and is given by the highest yet allowable delay (and therefore the minimum cost) in the *non-dominant* set.

For each link ℓ , there exists a number of capacity options $C_{\ell,i}$, $i = 1, 2, \dots, N$. Let $t_{\ell,i}$ be the delay contribution of link ℓ when capacity option i is used:

$$t_{\ell,i} = \frac{f_{\ell}}{\gamma} \left(\frac{1}{C_{\ell,i} - f_{\ell}} + \mu(p_i + r_i) \right) \quad (5.1)$$

Also let ∇_k be the table containing the set of all non-dominant solutions at stage k and $\nabla_k(s)$ be the s^{th} entry in the table represented by the delay-cost pair $(T_s, D_s)_k$. It is said that a solution (T_1, D_1) is dominated by another solution (T_2, D_2) if:

$$T_1 > T_2 \quad \text{and} \quad D_1 > D_2$$

and a set of solutions is *non-dominant* if no member is dominated by any other member. The elements of a table ∇_k are said to be *D-sorted* if they are arranged in order of increasing cost, i.e.,

$$D_1 < D_2 < \dots < D_{L_k}$$

where L_k is the size of table ∇_k . If the table contains a *non-dominant* set of solutions, then it follows that the elements are also sorted with respect to the delay, i.e.,

$$T_1 > T_2 > \dots > T_{L_k}$$

A solution in the table which does not follow such rules are dominated and must be removed. A systematic elimination of dominated solutions is called reduction of the table.

According to the principle of optimality due to Bellman [H3], if $(T_s, D_s)_k$ is a non-dominated solution in ∇_k , then it must satisfy the following relation

$$(T_s, D_s)_k = \min_{\substack{j=1, L_{k-1} \\ i=1, N}} \left[(T_j, D_j)_{k-1} + (t_{k,i}, d_{k,i}) \right] \quad (5.2)$$

in other words, table ∇_k can be constructed using only table ∇_{k-1} and $(t_{k,i}, d_{k,i})$, $i = 1, 2, \dots, N$. However, it would be computationally inefficient to verify (5.2) for each candidate solution to ∇_k . Instead ∇_{k-1} is used to construct a set of N auxiliary tables. Each auxiliary table corresponds to a capacity option i , $i = 1, 2, \dots, N$ and is obtained as follows. For capacity option i , each element in auxiliary table i is obtained by adding the delay-cost pair of option i to the corresponding element in ∇_{k-1} (if the delay is greater than $\bar{T}$, the solution is discarded). The set of N auxiliary tables are merged into one table which is D -sorted to form ∇_k . The procedure is repeated until the final stage ($k = 2m$) is reached. The entry in table ∇_{2m} which corresponds to a minimum cost represents the final solution.

The Lagrangian Decomposition approach for discrete capacity assignment is given in [F4]. It has been shown [G1] that the approach yields approximate solutions as opposed to DP which yields exact solutions.

Gerla [G1] conducted a number of numerical simulations to compare DP and LD when applied to actual networks. His results indicate that the choice between DP and LD should be decided on the base of the trade-off between computational efficiency and precision required. The results also indicate that for a specific application, there is a critical value for the network size (in terms of number of links) below which DP is used and above which LD is used.

Starting from an initial file allocation (and initial capacity assignment), the heuristic algorithm progresses by successive addition and deletion of copies of the shared files, (according to a certain procedure) from the network nodes and in such a way that decreases the total cost. The algorithm stops when no further improvement in the total cost can be obtained and is said to have completed a "heuristic run". The output of a heuristic run represents a "local minimum" with respect to the initial file allocation. The final solution corresponds to the outcome of the heuristic run that yields the smallest cost.

A heuristic run consists of two algorithms: the add algorithm and the delete algorithm. In the add algorithm, a test is conducted to see if the addition of an extra file copy to each node of the remaining set of nodes (at any point in the execution of the algorithm, a remaining set of nodes with respect to a file is the set containing all network nodes which are not assigned a copy of the file) would lead to an improvement in the total cost. This is done by calculating the set of incremental flows over the links of the network resulting from the addition of the file

copy to each remaining node. Assume $\Delta f_\ell(i,j)$ to be the incremental flow over link ℓ resulting from the assignment of a copy of file j to node i (this is equivalent to changing the value of variable x_{ij} from 0 to 1). A variable x_{ij} is called a *potential candidate* if a change in its value from 0 to 1 results in a set of incremental flows $\Delta f_\ell(i,j)$, $\ell = 1, 2, \dots, 2m$ that satisfies one or both of the following two inequalities:

$$\Delta d^*(i,j) = \sum_{\ell=1}^{2m} [d_\ell(C_\ell^*) - d_\ell(C_\ell)] + \sum_j b_j \leq 0 \quad (5.3)$$

$$\Delta f(i,j) = \sum_{\ell=1}^{2m} |\ell| \Delta f_\ell(i,j) \leq 0 \quad (5.4)$$

where C_ℓ^* is the minimum capacity required to satisfy the flow constraint, i.e.,

$$C_\ell^* > f_\ell + \Delta f_\ell(i,j), \quad \ell = 1, 2, \dots, 2m$$

Equation (5.3) expresses the minimum incremental cost (storage + communications) required to meet the flow constraint. If we assume $\Delta d(i,j)$ to be the minimum incremental cost required to meet the delay constraint, then,

$$\Delta d^*(i,j) \leq \Delta d(i,j)$$

The above inequality holds true because the communication cost decreases monotonically as the average message delay increases (see Fig. 6.1). Thus the addition of a copy of file j to any network node which

Communication Cost

Figure 6.1

Cost-Delay Function

results in a non-negative $\Delta d(i,j)$ will always lead to an increase in $\Delta d(i,j)$. The reason for using $\Delta d^*(i,j)$ instead of $\Delta d(i,j)$ for conducting the test is that it requires far less computational effort (the computation of $\Delta d(i,j)$ requires the execution of DP or LD routine).

Equation (5.4) represents the total incremental flow-length for the network corresponding to the assignment of file j to node i . The reasoning behind using the incremental flow-length as a criterion for testing the addition of extra file copies can be explained as follows. Consider the case where the cost is a linear function of the capacity and of the distance (length) of the communication link. Assume that the cost d_ℓ of link ℓ is given by:

$$d_\ell = k|\ell| C_\ell$$

where k is a constant, C_ℓ is the capacity of link ℓ , and $|\ell|$ is its length. The solution of the capacity assignment problem for this case is given by the following equation [K3]:

$$C_\ell = f_\ell + \frac{\sum \sqrt{k|\ell|f_\ell}}{\gamma \bar{T}} \sqrt{\frac{f_\ell}{k|\ell|}}$$

thus

$$\sum_{\ell=1}^{2m} k|\ell|C_\ell = \sum k|\ell|f_\ell + \frac{[\sum \sqrt{k|\ell|f_\ell}]^2}{\gamma \bar{T}} \quad (5.5)$$

The assignment of a copy of file j to node i will result in an incremental flow $\Delta f_\ell(i,j)$ and an incremental cost $\Delta d(i,j)$ to meet the delay constraint. If $\gamma \bar{T}$ in eqn. (5.5) is large enough, then

$$\Delta d(i,j) \approx k \sum_{\ell=1}^{2m} |\ell| \Delta f_{\ell}(i,j)$$

which indicates that a negative incremental flow-length leads to an improvement in the communication cost. Although the previous result does not necessarily follow in the discrete capacity case, numerical simulation (see example of Section 6.4) indicated that a negative total incremental flow-length often corresponds to a possible decrease in the total cost.

The above procedure is repeated until all potential candidate variables are found. Potential candidate variables corresponding to smallest values of $\Delta d(i,j)$ and $\Delta f(i,j)$ for all files $j = 1, 2, \dots, M$ are then grouped into one set. If this set contains r variables, then as many as 2^r states can be formed corresponding to the set; each state represents an r -tuple assignment of binary values to the variables. Exhaustive search is then conducted to find the feasible state that leads to a maximum decrease in the total cost. The add algorithm is repeated until no further improvement in the total cost can be attained and control is transferred to the delete algorithm.

In the delete algorithm the procedure followed in the add algorithm is repeated with one exception: the deletion of a file copy may degrade the file availability condition and thus a reliability test must be conducted. In the add algorithm, this was not necessary since the addition of more copies of the file always improves the file availability.

The algorithm iterates between the add and delete routines until no further improvement in the total cost function can be obtained. This signals the end of a heuristic run.

It is not necessary, however, to try all of the elements of the set of initial file allocations since some of these allocations may be generated during the course of other heuristic runs. Elimination of these initial file allocations leads to a substantial reduction in the number of runs and further improves the computational efficiency of the algorithm.

6.3 The Heuristic Algorithm

Initial File Assignment Algorithm

The algorithm finds initial values for the allocation matrix, X , according to the following steps.

1. Find

$$\text{Min}_{\{X\}} \underline{b}^T X \underline{e}$$

subject to

$$\alpha_j \geq \alpha_j', \quad j = 1, 2, \dots, M.$$

The above problem is decomposed into M subproblems, each of which can be stated as follows:

Find

$$\text{Minimum}_{\{x_{ij}, i=1, 2, \dots, n\}} \sum_{i=1}^n b_i x_{ij} e_j$$

for $j = 1, 2, \dots, M$

subject to

$$\alpha_j \geq \alpha_j'.$$

2. The solution of these M integer-programming subproblems will yield an initial value, X^0 , for the allocation matrix, X . Assume r_j^0 to represent the number of copies of file j in the initial allocation matrix X^0 . The set Z_j is defined to be the set of all possible r_j^0 -assignment of file j (an r_j^0 -assignment means r_j^0 copies of file j are stored in r_j^0 network node); with each r_j^0 -assignment corresponding to an element of the set Z_j and at the same time satisfying the reliability constraint:

$$\alpha_j \geq \alpha_j^0$$

Also define the set Z to be the Cartesian product of the sets Z_j , $j = 1, 2, \dots, M$:

$$Z \triangleq Z_1 \times Z_2 \times \dots \times Z_M$$

Elements of the set Z will represent initial solutions for the file allocation problem and the set Z will be called the set of initial file allocations.

Initial Capacity Assignment Algorithm

This algorithm finds initial values for the link capacities, C_ℓ , $\ell = 1, 2, \dots, 2m$, corresponding to a given file allocation according to the following steps:

1. Reset flags 1 and 2.
2. The routing tables together with the next initial file allocation in the initial file allocation set Z are used to calculate link flows f_ℓ , $\ell = 1, 2, \dots, 2m$.
3. Assign link capacities for minimum communication cost, i.e., find

$$\text{Min}_{\{C_\ell\}} \sum_{\ell} d_\ell(C_\ell)$$

subject to

$$\frac{1}{\gamma} \sum_{\ell=1}^{2m} \left[\frac{f_{\ell}}{C_{\ell} - f_{\ell}} + \lambda_{\ell} p_{\ell} \right] \leq \bar{T}$$

$$C_{\ell} > f_{\ell}, \quad \lambda_{\ell} = \mu f_{\ell}$$

(use dynamic programming or lagrangian decomposition).

4. Calculate the initial cost

$$C_T = \sum_{\ell} d_{\ell}(C_{\ell}) + \sum_{i,j} \ell_j b_i x_{ij}$$

corresponding to the file allocation matrix X .

5. Transfer to algorithm add.

Algorithm Add

This algorithm examines the feasibility of adding extra copies of each file to those network nodes that do not contain a copy of the file.

Input Values: The allocation matrix X ,

the set of link flows, f_{ℓ} ($\ell = 1, 2, \dots, 2m$)

the set of link capacities, C_{ℓ} ($\ell = 1, 2, \dots, 2m$)

the total cost, C_T , associated with the above values.

Definitions:

Define S_j to be the set of nodes, each node $i \in S_j$ contains a copy of file j and S_j^c to be the set containing the remaining network nodes (nodes that do not contain a copy of file j), i.e.,

$$S_j \equiv \{i/x_{ij} = 1\}$$

$$S_j' \equiv \{i/x_{ij} = 0\}$$

Let $\Delta f_\ell(i,j)$ be the increase (or decrease) in the traffic flow, f_ℓ , over link ℓ when a copy of file j is assigned to node $i \in S_j'$. Also assume C_ℓ^* to be the lowest available capacity level for link ℓ such that:

$$C_\ell^* > f_\ell + \Delta f_\ell(i,j)$$

For each file j , define the set Q_j of variables, x_{ij} 's, such that each variable $x_{ij} \in Q_j$ satisfies one of two criteria to be defined later.

Steps of Algorithm Add

1. If flag 1 is reset, go to step 2.

If it is set, then test the relationship

$$C_T < C_{T(\min)}$$

If the relationship is true then set:

$$C_{T(\min)} = C_T$$

$$\lambda_{(\min)} = \lambda$$

$$f_{\ell(\min)} = f_\ell \quad \ell = 1, 2, \dots, 2m$$

$$C_{\ell(\min)} = C_\ell \quad \ell = 1, 2, \dots, 2m$$

Go to the initial capacity assignment algorithm.

The following steps, starting from step 2, will be executed for each file j , $j = 1, 2, \dots, M$:

2. Set $Q_j = \{\phi\}$

The following steps will be executed for each node $i \in S_j$.

3. Change $x_{ij} = 0$ to $x_{ij} = 1$ then test the following two conditions:

$$\Delta d(i,j) = \sum_{\ell=1}^{2m} [d_{\ell}(C_{\ell}^*) - d_{\ell}(C_{\ell})] + \ell_j b_i \leq 0$$

$$\Delta f(i,j) = \sum_{\ell=1}^{2m} |\ell| \Delta f_{\ell}(i,j) \leq 0$$

among the nodes of the set S_j satisfying the first condition, find node i that corresponds to the smallest incremental cost $\Delta d(i,j)$ and among the nodes of the set S_j satisfying the second condition, find node k that corresponds to the smallest incremental flow $\Delta f(k,j)$. The corresponding variables x_{ij} and x_{kj} are included in the set Q_j (i and k may refer to the same node).

4. Form

$$Q = \bigcup_{j=1}^M Q_j$$

if the set Q is empty, set flag 1 and go to algorithm delete. Otherwise go to algorithm search add.

Algorithm Search-Add

This algorithm examines the feasible variations of adding copies of the shared files to more nodes and determines the one corresponding to a minimum network cost.

Input Values: The allocation matrix X ,

the set of link capacities, C_{ℓ} ($\ell = 1, 2, \dots, 2m$),

the set of link flows, f_{ℓ} ($\ell = 1, 2, \dots, 2m$),

the total cost, C_T , associated with the above values,
the set Q .

Definitions:

Each variable $x_{ij} \in Q$ may be assigned a value of 1 or 0 depending on whether file j is stored in node i or not. If the set Q contains r elements, then there exist 2^r possible variations (states), each such state corresponds to a given set value assigned to the elements of the set Q . A state will be denoted by χ . When all the variables of the set Q are equal to zero, state χ_0 results. Similarly state χ_t corresponds to the case where all the variables of the set Q are equal to 1. Define $\Delta f_\ell(\chi)$ to be the flow increment over link ℓ corresponding to state χ and $C_\ell^*(\chi)$ to be the lowest available capacity level for link ℓ such that:

$$C_\ell^*(\chi) > f_\ell + \Delta f_\ell(\chi)$$

It is obvious that:

$$\Delta f_\ell(\chi_0) = 0, \quad \ell = 1, 2, \dots, 2m$$

Steps of Algorithm Search-Add

The following steps are repeated for every state χ of the elements of the set Q (state χ_0 is not included).

1. Find $\Delta f_\ell(\chi)$, for $\ell = 1, 2, \dots, 2m$.
2. Find $C_\ell^*(\chi)$, for $\ell = 1, 2, \dots, 2m$.
3. Test the following condition:

$$\Delta d^*(x) = \sum_{\ell=1}^{2m} [d_{\ell}(C_{\ell}^*) - d_{\ell}(C_{\ell})] + \sum_j \lambda_j b_{ij} x_{ij} < 0$$

If the condition is satisfied, go to step 4, if not discard the state being examined.

4. Apply the dynamic programming (or lagrangian decomposition) to find:

$$\text{Min}_{\{C'_{\ell}(x)\}} \sum_{\ell=1}^{2m} d_{\ell}(C'_{\ell}(x)) - d_{\ell}(C_{\ell})$$

subject to:

$$\frac{1}{Y} \sum_{\ell} \frac{f_{\ell} + \Delta f_{\ell}(x)}{C'_{\ell} - [f_{\ell} + \Delta f_{\ell}(x)]} \leq \bar{T}$$

5. Compute:

$$\begin{aligned} \Delta C_T(x) &= \sum_{\ell=1}^{2m} [d_{\ell}(C'_{\ell}(x)) - d_{\ell}(C_{\ell})] \\ &\quad + \sum_j \lambda_j b_{ij} x_{ij} \end{aligned}$$

6. Among all the states that correspond to an incremental cost satisfying the inequality:

$$\Delta C_T(x) < 0$$

find the state x_m corresponding to a minimum incremental cost, $\Delta C_T(x_m)$.

7. If no state is found to satisfy:

$$\Delta C_T(x) < 0$$

then set flag 1 and go to algorithm delete, otherwise go to step 8.

8. Set:

$$\left. \begin{aligned} C_T &= C_T + \Delta C_T(x_m) \\ f_\ell &= f_\ell + \Delta f_\ell(x_m) \\ C_\ell &= C_\ell(x_m) \end{aligned} \right\} \text{ for } \ell = 1, 2, \dots, 2m$$

update X

9. Reset flag 2 and go to Algorithm Add.

Algorithm Delete

This algorithm examines the feasibility of deleting copies of files from network nodes that have been previously assigned copies of the shared files.

Input Values: The allocation matrix X,
the set of link flows, f_ℓ ($\ell = 1, 2, \dots, 2m$)
the set of link capacities, C_ℓ ($\ell = 1, 2, \dots, 2m$)
the total cost, C_T , associated with the above values

Definitions

The sets S_j , S_j^1 and Q_j have the same definitions stated in Algorithm Add. The incremental flow, $\Delta f_\ell(i, j)$, is defined as the increase (or decrease) in the traffic flow, f_ℓ , over link ℓ when a copy of file j is deleted from node $i \in S_j$. The capacity, C_ℓ^* , has the same definition given in Algorithm Add.

Steps of Algorithm Delete

1. If flag 2 is reset, go to step 2, otherwise go to algorithm add.

The following steps, starting from step 2, will be executed for each file j , $j = 1, 2, \dots, M$:

2. Set $Q_j = \{\phi\}$

The following steps will be executed for each node $i \in S_j$

3. Change $x_{ij} = 1$ to $x_{ij} = 0$

then test the following three conditions:

$$\alpha_j \geq \alpha_i$$

$$\Delta d^*(i, j) = \sum_{\ell=1}^{2m} [d_{\ell}(C_{\ell}^*) - d_{\ell}(C_{\ell})] - \ell_j b_i \leq 0$$

$$\Delta f(i, j) = \sum_{\ell=1}^{2m} |\ell| \Delta f_{\ell}(i, j) \leq 0$$

among the nodes of the set S_j satisfying the first and second conditions, find node i that corresponds to the smallest incremental cost $\Delta d(i, j)$ and among the nodes of the set S_j satisfying the first and third conditions, find node k that corresponds to the smallest incremental flow $\Delta f(k, j)$. The corresponding variables x_{ij} and x_{kj} are included in the set Q_j (i and k may refer to the same node).

4. Form

$$Q = \bigcup_{j=1}^M Q_j$$

5. If the set Q is empty, set flag 2 and go to algorithm add. Otherwise go to Algorithm Search-Delete.

Algorithm Search-Delete

This algorithm examines the feasible variations of deleting copies of the shared files from some nodes and determines the one corres-

ponding to a minimum network cost.

Input Values: The allocation matrix X ,
 the set of link capacities $c_\ell (\ell = 1, 2, \dots, 2m)$,
 the set of link flows, $f_\ell (\ell = 1, 2, \dots, 2m)$,
 the total cost, C_T , associated with the above values,
 the set Q .

Definitions

The states corresponding to the values assigned to the elements of the set Q will have the same definition stated in Algorithm Search-Add. The same applies to $\Delta f_\ell(x)$ and $C_\ell^*(x)$. In the Delete case it is obvious that:

$$\Delta f_\ell(x_t) = 0, \quad \ell = 1, 2, \dots, 2m$$

Steps of Algorithm Search-Delete

The following steps are repeated for every state x of the elements of the set Q (state x_t is not included).

1. Find $\Delta f_\ell(x)$, for $\ell = 1, 2, \dots, 2m$.
2. Find $C_\ell^*(x)$, for $\ell = 1, 2, \dots, 2m$.
3. Test the following condition:

$$\Delta d^*(x) = \sum_{\ell=1}^{2m} [d_\ell(C_\ell^*) - d_\ell(C_\ell)] - \sum_x \ell_j b_i x_{ij} < 0$$

If the condition is satisfied, go to step 4, if not discard the state being examined.

4. Apply dynamic programming (or lagrangian decomposition) to find:

$$\text{Min}_{\{C'_\ell(x)\}} \sum_{\ell=1}^{2m} d_\ell(C'_\ell(x)) - d_\ell(C_\ell)$$

subject to

$$\frac{1}{Y} \sum_{\ell} \frac{f_\ell + \Delta f_\ell(x)}{C'_\ell - [f_\ell + \Delta f_\ell(x)]} \leq \bar{T}$$

5. Compute:

$$\Delta C_T(x) = \sum_{\ell=1}^{2m} [d_\ell(C'_\ell(x)) - d_\ell(C_\ell)] - \sum_x \ell_j b_i x_{ij}$$

6. Among all the states that correspond to an incremental cost satisfying the inequality:

$$\Delta C_T(x) < 0$$

find the state x_m corresponding to a minimum incremental cost, $\Delta C_T(x_m)$.

7. If no state is found to satisfy:

$$\Delta C_T(x) < 0$$

then set flag 2 and go to Algorithm Add, otherwise go to step 8.

8. Set:

$$\left. \begin{aligned} C_T &= C_T + \Delta C_T(x_m) \\ f_\ell &= f_\ell + \Delta f_\ell(x_m) \\ C_\ell &= C_\ell(x_m) \end{aligned} \right\} \text{ for } \ell = 1, 2, \dots, 2m$$

update λ .

9. Reset flag 1 and go to Algorithm Delete.

6.4 Example

The network shown in Fig. 6.2 consists of 4 nodes and 4 links, each link consists of two channels (one channel in each direction). The number associated with each link in Fig. 6.2 represents the reliability of that link. Users of the 4 nodes share two different information files. The expected demand for each file by the different nodes (in units of traffic/unit of time) is given in Table 6.1. The cost of storing each file in \$/month in the different network nodes is given in Table 6.2. It is assumed that 4 levels of line capabilities (speeds) are available for the communication channels. These levels correspond to line speeds of 10, 16, 30 and 50 in units of traffic/unit of time. The cost of a communication channel is given by the following formula:

$$A + B \times \text{length (miles)}$$

Values of A and B for different line speeds are given in Table 6.3. The distance (in miles) for each link along with the values of the constants A and B are used to calculate the cost in \$/month for each link corresponding to each of the four available capacity levels. The results are given in Table 6.4.

Requirements

Given the foregoing data, it is required to assign link capacities and file locations (with at least one copy of each file in the net-

Figure 6.2

a 4-node Network

r_{ik} represents the terminal
reliability between nodes i and k

$$r_{12} = 0.9765$$

$$r_{13} = 0.9765$$

$$r_{23} = 0.9715$$

$$r_{34} = 0.7$$

$$r_{14} = 0.68355$$

$$r_{24} = 0.68005$$

TABLE 6.1

File access traffic in units of traffic/unit of time

$$(u_{ij} = u'_{ij}, v_{ij} = v'_{ij})$$

	FILE 1		FILE 2	
	u_{i1}	v_{i1}	u_{i2}	v_{i2}
node 1	4	3	8	3
node 2	13	3	5	2
node 3	3	1	2	1
node 4	4	2	10	2

TABLE 6.2

Storage cost in \$/month

	FILE 1	FILE 2
node 1	200	220
node 2	220	242
node 3	230	253
node 4	190	209

TABLE 6.3

Values of A and B in dollar/month

CAPACITY LEVEL IN UNITS OF TRAFFIC PER UNIT OF TIME	A	B
10	55	9
16	100	12
30	115	15
50	250	22.5

TABLE 6.4

Communication lines leasing cost in dollar/month

	LENGTH IN MILES	CAPACITY LEVEL (units of traffic/unit of time)			
		10	16	30	50
link l_1	70	685	940	1165	1825
link l_2	60	595	820	1015	1600
link l_3	25	280	400	490	812.5
link l_4	30	325	460	565	925

work), to minimize the combined cost of communications and storage, while realizing the availabilities $\{\alpha_1^i, \alpha_2^i\}$ where

$$\alpha_1^i = 0.9, \quad \alpha_2^i = 0.95$$

and incurring an average message delay, T , of no more than 0.5 units of time/unit of traffic.

Solution

The heuristic algorithm will be applied to solve the previous example. The solution is illustrated in the following steps:

Step 1

Apply the initial file assignment algorithm to each of the two shared files. For the first file the solution is to store a copy of the file in node 1. In this case the file availability α_1 will be

$$\begin{aligned} \alpha_1 &= \frac{14 + 32 \times 0.9765 + 8 \times 0.9765 + 12 \times 0.68355}{14 + 32 + 8 + 12} \\ &= 0.93 > \alpha_1^i \quad (\alpha_1^i = 0.9) \end{aligned}$$

a less costly solution would be to store the copy in node 4, but the file availability, α_1 , in this case will be equal to 0.74 which is less than α_1^i , i.e., the availability constraint is not satisfied.

Application of the initial assignment algorithm to the second file, results in the assignment of one copy of the file to each of nodes 1 and 4. The file availability, α_2 corresponding to this assignment is

equal to 0.99 which is satisfactory since $\alpha_2' = 0.95$. It can be easily shown that in the allocation of file 2, no single-copy assignment can be found to satisfy the availability constraint. The initial file allocation corresponds to the following values for the variables x_{ij} 's:

$$\begin{array}{llll} x_{11} = 1 & x_{21} = 0 & x_{31} = 0 & x_{41} = 0 \\ x_{12} = 1 & x_{22} = 0 & x_{32} = 0 & x_{42} = 1 \end{array}$$

The sets Z_1 and Z_2 will be given by

$$\begin{aligned} Z_1 &= \{1, 2, 3\} \\ Z_2 &= \{(1, 4), (2, 4), (3, 4)\} \end{aligned}$$

The set of initial solutions Z defined as,

$$Z = Z_1 \times Z_2$$

contains 9 elements, each element corresponds to an initial file allocation. Elements of the set Z are:

	FILE 1				FILE 2			
	x_{11}	x_{21}	x_{31}	x_{41}	x_{12}	x_{22}	x_{32}	x_{42}
1	1	0	0	0	1	0	0	1
2	1	0	0	0	0	1	0	1
3	1	0	0	0	0	0	1	1
4	0	1	0	0	1	0	0	1
5	0	1	0	0	0	1	0	1
6	0	1	0	0	0	0	1	1
7	0	0	1	0	1	0	0	1
8	0	0	1	0	0	1	0	1
9	0	0	1	0	0	0	1	1

The remainder of the algorithm, starting from step 2, is then applied to each of the above initial file allocations to obtain the corresponding local minimum solutions. Among these local minimum solutions we select the one corresponding to the smallest cost as the final solution to the problem.

Consider the first initial file allocation,

$$\begin{array}{llll} x_{11} = 1 & x_{21} = 0 & x_{31} = 0 & x_{41} = 0 \\ x_{12} = 1 & x_{22} = 0 & x_{32} = 0 & x_{42} = 1 \end{array}$$

Step 2

Using the file access table (Table 6.1) and the shortest distance routing algorithm, the link flows in units of traffic per unit of time corresponding to the initial file assignment (result of step 1) are computed:

$$\begin{array}{lll} f_1 = 20.5 & f_2 = 17 & \gamma = 88 \\ f_3 = 4.5 & f_4 = 17.5 & \end{array}$$

Using dynamic programming, the following values for line capacities and line costs are obtained:

$$\begin{array}{ll} C_1 = 30 & d_1(C_1) = 1165 \text{ $/month} \\ C_2 = 30 & d_2(C_2) = 1015 \text{ $/month} \\ C_3 = 10 & d_3(C_3) = 280 \text{ $/month} \\ C_4 = 30 & d_4(C_4) = 565 \text{ $/month} \end{array}$$

The corresponding time delay, T , is equal to:

$$T = \frac{2}{88} \left(\frac{20.5}{9.5} + \frac{17}{13} + \frac{4.5}{5.5} + \frac{17.5}{12.5} \right)$$

$$= 0.13 \text{ unit of time/unit of traffic}$$

and the total cost, C_T , will be:

$$C_T = 2(1165 + 1015 + 280 + 565) + 200 + 220 + 209$$

$$= 6679 \text{ \$/month.}$$

Flags 1 and 2 are reset before transferring control to Algorithm Add.

Step 3

The Add Algorithm is now executed

$$\begin{array}{lll} S_1 = \{1\} & S'_1 = \{2,3,4\} & Q_1 = \{\phi\} \\ S_2 = \{1,4\} & S'_2 = \{2,3\} & Q_2 = \{\phi\} \end{array}$$

It is obvious that the variables x_{21} , x_{31} , x_{41} , x_{22} and x_{32} should be tested when their values are changed from 0 to 1. For each variable a table is formed to indicate the capacity and cost increments over each of the network links. For variable x_{21} :

	$f_{\ell} + \Delta f_{\ell}(2,1)$	$\Delta f_{\ell}(2,1)$	C_{ℓ}^*	$d_{\ell}(C_{\ell}^*) - d_{\ell}(C_{\ell})$
ℓ_1	10.5	-10	16	-225
ℓ_2	13.5	-3.5	16	-195
ℓ_3	11	6.5	16	120
ℓ_4	19.5	2	30	0

Therefore:

$$\Delta d(2,1) = -380, \quad \Delta f(2,1) = -1375$$

For variable x_{31} :

	$f_{\ell} + \Delta f_{\ell}(3,1)$	$\Delta f_{\ell}(3,1)$	C_{ℓ}^*	$d_{\ell}(C_{\ell}^*) - d_{\ell}(C_{\ell})$
ℓ_1	14	-6.5	16	-225
ℓ_2	15	-2	16	-195
ℓ_3	14	9.5	16	120
ℓ_4	19.5	2	30	0

Therefore:

$$\Delta d(3,1) = -370, \quad \Delta f(3,1) = -555$$

For variable x_{41} :

	$f_{\ell} + \Delta f_{\ell}(4,1)$	$\Delta f_{\ell}(4,1)$	C_{ℓ}^*	$d_{\ell}(C_{\ell}^*) - d_{\ell}(C_{\ell})$
ℓ_1	14	-6.5	16	-225
ℓ_2	14.5	-2.5	16	-195
ℓ_3	14	9.5	16	120
ℓ_4	28.5	11	30	0

Therefore:

$$\Delta d(4,1) = -410, \quad \Delta f(4,1) = -75$$

As a result the set Q_1 contains x_{21} and x_{41} .

For variable x_{22} :

	$f_{\ell} + \Delta f_{\ell}(2,2)$	$\Delta f_{\ell}(2,2)$	C_{ℓ}^*	$d_{\ell}(C_{\ell}^*) - d_{\ell}(C_{\ell})$
ℓ_1	21	0.5	30	0
ℓ_2	16.66	-0.34	30	0
ℓ_3	5.66	1.11	10	0
ℓ_4	16.66	-0.94	30	0

The incremental cost $\Delta d(2,2)$ and the incremental flow $\Delta f(2,2)$ are both greater than zero for the variable x_{22} .

For the variable x_{32} :

	$f_{\ell} + \Delta f_{\ell}(3,2)$	$\Delta f_{\ell}(3,2)$	C_{ℓ}^*	$d_{\ell}(C_{\ell}^*) - d_{\ell}(C_{\ell})$
ℓ_1	19.66	-0.94	30	0
ℓ_2	19	2	30	0
ℓ_3	7.33	2.83	10	0
ℓ_4	17.66	0.16	30	0

The incremental cost $\Delta d(3,2)$ and the incremental flow $\Delta f(3,2)$ are both greater than zero for the variable x_{32} . As a result the set Q_2 is empty.

The set defined as:

$$Q = Q_1 \cup Q_2$$

will be:

$$Q \equiv \{x_{21}, x_{41}\}$$

The Search-Add Algorithm is then executed.

Step 4

Since the set Q contains two elements, three states will be examined by the search-add algorithm. The following table includes these states and the corresponding incremental cost $\Delta d(x)$:

STATE	x_{21}	x_{41}	$\Delta d(x)$
x_1	1	0	-380
x_2	0	1	-410
x_3	1	1	-190

Dynamic programming is then applied to compute the minimum incremental cost $\Delta C_T(x)$ corresponding to each state x for which $\Delta d(x)$ is a negative quantity. The results are given in the following table

STATE	x_1	x_2	x_3
$\Delta C_T(x)$	-380	490	-190

State x_1 corresponds to the minimum total incremental cost.

Therefore:

$$x_m = x_1$$

and the values of the incremental link flows and capacities corresponding to x_m are given below:

	$\Delta f_\ell(x_m)$	$C_\ell(x_m)$
ℓ_1	-10	16
ℓ_2	-3.5	16
ℓ_3	6.5	16
ℓ_4	2	30

Elements of the new allocation matrix X will be:

$$\begin{array}{llll} x_{11} = 1 & x_{21} = 1 & x_{31} = 0 & x_{41} = 0 \\ x_{12} = 1 & x_{22} = 0 & x_{32} = 0 & x_{42} = 1 \end{array}$$

and the new set of link flows and capacities are given by the relationships

$$\begin{aligned} f_{\ell} &= f_{\ell} + \Delta f_{\ell}(x_m) & \ell &= 1, 2, \dots, 2m \\ c_{\ell} &= c_{\ell}(x_m) & \ell &= 1, 2, \dots, 2m \end{aligned}$$

Therefore:

$$\begin{array}{ll} c_1 = 16 & f_1 = 10.5 \\ c_2 = 16 & f_2 = 13.5 \\ c_3 = 16 & f_3 = 11 \\ c_4 = 30 & f_4 = 19.5 \end{array}$$

and the new total cost C_T is equal to:

$$\begin{aligned} C_T &= 6679 - 380 \\ &= 6299 \quad \$/\text{month} \end{aligned}$$

The corresponding time delay, T , is equal to:

$$\begin{aligned} T &= \frac{1}{37} \left[\frac{10.5}{5.5} + \frac{13.5}{2.5} + \frac{11}{5} + \frac{19.5}{10.5} \right] \\ &= 0.34 \text{ units of time/unit of traffic.} \end{aligned}$$

Control is now transferred back to algorithm add.

Step 5

$$S_1 = \{1,2\} \quad S_1' = \{3,4\} \quad Q_1 = \{\phi\}$$

$$S_2 = \{1,4\} \quad S_2' = \{2,3\} \quad Q_2 = \{\phi\}$$

It is obvious that the variables x_{31} , x_{41} , x_{22} and x_{32} should be tested when their values are changed from 0 to 1. Forming incremental cost and flow tables similar to the tables of Step 3 indicated that the resulting incremental cost Δd and flow Δf corresponding to the variables x_{31} , x_{41} and x_{22} are all greater than zero. For the variable x_{32} the incremental cost $\Delta d(3,2)$ was found equal to

$$\Delta d(3,2) = -257$$

and the incremental flow $\Delta f(3,2)$ is actually greater than zero. Thus

$$Q_1 = \{\phi\} \quad \text{and} \quad Q_2 = \{x_{32}\}$$

Applying algorithm search-add to the only resulting state, the minimum incremental cost ΔC_T was computed:

$$\Delta C_T = +253$$

i.e., $\Delta C_T > 0$ and no addition of file copies is found to reduce the total cost further. Flag 1 is set and control is transferred to algorithm delete.

Step 6

Algorithm Delete is now applied.

$$\begin{array}{lll} S_1 = \{x_{11}, x_{21}\} & S'_1 = \{x_{31}, x_{41}\} & Q_1 = \{\phi\} \\ S_2 = \{x_{12}, x_{42}\} & S'_2 = \{x_{22}, x_{32}\} & Q_2 = \{\phi\} \end{array}$$

elements of the sets S_1 and S_2 are tested when their values are changed from 1 to 0. Deleting the copy of the first file from node 2 will obviously result in incremental increases in flow and cost. Deleting any of the two copies of the second file will result in an allocation that does not satisfy the availability constraint of the second file. The only deletion that may be considered then is the one corresponding to changing $x_{11} = 1$ to $x_{11} = 0$. The result is given in the following table:

	$f_\ell + \Delta f_\ell(1,1)$	$\Delta f_\ell(1,1)$	C_ℓ^*	$d_\ell(C_\ell^*) - d_\ell(C_\ell)$
ℓ_1	11.5	1	16	0
ℓ_2	7	-6.5	10	-225
ℓ_3	14.5	3.5	16	0
ℓ_4	17.5	-2	30	0

Therefore:

$$\Delta d(1,1) = -650 \qquad \Delta f(1,1) = -585$$

$$Q_1 = \{x_{11}\}$$

$$Q_2 = \{\phi\}$$

$$Q = Q_1 \cup Q_2$$

$$= \{x_{11}\}$$

and control is transferred to algorithm search-delete.

Step 7

Dynamic programming is applied to compute the minimum incremental cost $\Delta C_T(x)$ when the variable x_{11} is changed from 1 to zero. The results are given in the following table:

	$\Delta f_l(x)$	$C_l(x)$
l_1	1	16
l_2	-6.5	10
l_3	3.5	16
l_4	-2	30

and the incremental cost:

$$\Delta C_T(x) = -650$$

elements of the new allocation matrix X will be:

$$\begin{array}{cccc} x_{11} = 0 & x_{21} = 1 & x_{31} = 0 & x_{41} = 0 \\ x_{12} = 1 & x_{22} = 0 & x_{32} = 0 & x_{42} = 1 \end{array}$$

and the new set of link flows and capacities will be:

$$\begin{array}{cc} C_1 = 16 & f_1 = 11.5 \\ C_2 = 10 & f_2 = 7 \\ C_3 = 16 & f_3 = 14.5 \\ C_4 = 30 & f_4 = 17.5 \end{array}$$

and the new total cost is equal to:

$$\begin{aligned}
 C_T &= 6299 - 650 \\
 &= 5649 \text{ \$/month}
 \end{aligned}$$

The corresponding time delay, T , is equal to:

$$\begin{aligned}
 T &= \frac{1}{35} \left[\frac{11.5}{4.5} + \frac{7}{3} + \frac{14.5}{11.5} + \frac{17.5}{12.5} \right] \\
 &= 0.45 \text{ units of time/unit of traffic.}
 \end{aligned}$$

It is obvious that no more copies of any of the two files may be deleted. Flag is reset while flag 2 is set and control is transferred to algorithm add.

Step 8

$$\begin{aligned}
 S_1 &= \{x_{21}\} & S_1' &= \{x_{11}, x_{31}, x_{41}\} \\
 S_2 &= \{x_{12}, x_{42}\} & S_2' &= \{x_{22}, x_{32}\}
 \end{aligned}$$

The incremental cost Δd and the incremental flow Δf corresponding to each of the elements of the sets S_1' and S_2' were computed (as in Step 3). The results indicate that Δd and Δf are greater than zero for all of the elements of S_1' and S_2' . Thus the addition of more copies of any of the two files to any of the network nodes will not decrease the total cost. Flag 1 is set to indicate the termination of algorithm add. Since the two flags are now set, the algorithm terminates and a solution corresponding to a local minimum is reported.

The above procedure starting from step 2 is repeated for the initial file allocations derived in Step 1 and for each initial file

allocation, a local minimum is obtained. In the previous example, the initial file allocation for which a solution was presented in detail is the one leading to the smallest total cost. Therefore

$$C_{T(\min)} = 5649 \text{ \$/month}$$

CHAPTER 7

COMPUTATIONAL EXPERIENCE

In Chapter 4, a model for the allocation of resources in distributed information networks was formulated. Two algorithms were proposed to solve this model. The first algorithm is based on a nonlinear Branch-and-Bound technique [H5] and represents an exact method of solution (Chapter 5). The second algorithm was discussed in Chapter 6 and represents an approximate (heuristic) method of solution. This chapter examines the following two aspects:

- The computational efficiency (computer processing time and storage requirements) of each of the two algorithms.
- Sensitivity of the resource allocation to changes in the file availability requirements and traffic demand requirements.

The two algorithms were programmed for computer processing and applied to a large number of resource allocation problems. A typical sample of the results is presented in Section 7.1. Section 7.2 examines the sensitivity of resource allocation (assignment of files to nodes and communication capacities to links) to changes in the file availability requirements. Section 7.3 discusses the effects of changing retrieval and update traffic requirements on the allocation of resources.

7.1 Computational Experience

The nonlinear Branch-and-Bound integer programming algorithm and the proposed heuristic algorithm were programmed in APL on an XDS Sigma-7

computer and applied to a large number of resource allocation problems. Five examples are presented here as a typical sample of these problems. The first four examples represent small to medium size problems; the fifth example represents a large size problem (the size of a resource allocation problem is determined by the number of optimization variables, i.e., number of elements of the allocation matrices X and Y).

During the computer run of each example, the value of the total cost was monitored as a function of elapsed CPU time. The initial cost value of the nonlinear Branch-and-Bound algorithm was selected arbitrarily from the set of initial values generated by the heuristic algorithm for each example.

Problem Statement

Example 1

The resource allocation problem of the 4-node network shown in Fig. 6.2 will be considered as an example of small size problems. Problem statement of this example is given in Section 6.4 of the previous chapter.

Example 2

Consider the network shown in Fig. 7.1. There are five nodes and 14 links; assume that two distinct files are involved ($n = 5$, $2m = 14$, $M = 2$). Average message length, μ^{-1} , is equal to 500 bits. Reliability and length (in miles) of each of the seven bidirectional links are shown in Fig. 7.1. Nodes are considered to be perfectly reliable. Table 7.1

Figure 7.1
A 5-NODE NETWORK

shows u_{ij} and v_{ij} ; it is assumed that $u_{ij} = u'_{ij}$ and $v_{ij} = v'_{ij}$. File storage costs are also given in Table 7.1. Available capacity options $\{h_j\}$ are $\{2.4, 4.8, 9.6, 19.2\}$ kilobits/sec. The cost of any link is assumed to be given by the equation

$$A + Bm_\ell$$

where m_ℓ is the length (in mile) of link ℓ , A and B are constants. The values of A and B are given in Table 7.2 for different line speeds. The required file availabilities $\{\alpha_j\}$ are $\{0.95, 0.92\}$ and the maximum admissible average message delay, $\bar{T}$, is equal to 500 milliseconds.

Example 3

Consider the network shown in Fig. 7.2 (Proposed Topology for CANUNET). There are nine nodes ($n = 9$) and 26 links ($2m = 26$). Reliability and length (in miles) each of the 13 bidirectional links are given in Table 7.3. Nodes are considered to be totally reliable. Table 7.4 shows u_{ij} and v_{ij} ; it is assumed that $u_{ij} = u'_{ij}$ and $v_{ij} = v'_{ij}$. File storage costs are also given in Table 7.4. The values of the remaining parameters of the resource allocation problem are:

$$M = 1, \quad \mu^{-1} = 500 \text{ bits}$$

$$\{h_j\} = \{2.4, 4.8, 9.6, 19.2\} \text{ kilobits/sec.}$$

(Table 7.2 is used to compute the capacity-cost matrix)

$$\bar{T} = 300 \text{ milliseconds}, \quad \alpha_j = 0.95$$

TABLE 7.1

File access requirements u_{ij} and v_{ij} (in messages/sec)
and file storage cost (in \$/month) for the
problem of Example 2

NODE	FILE 1			FILE 2		
	u_{i1}	v_{i1}	STORAGE COST	u_{i2}	v_{i2}	STORAGE COST
1	10	4	200	4	2	300
2	4	4	220	6	4	330
3	16	6	210	12	6	315
4	6	4	225	4	2	335
5	4	2	190	16	4	285

TABLE 7.2

Values of constants A and B for different line capacities

k bits/sec	A \$/MONTH	B \$/MONTH/MILE
2.4	180	2.42
4.8	680	2.42
9.6	1080	2.42
19.2	2160	4.84
50	1000	25

m_l is the length (in miles) of link l ,

A and B are constants. The values of
A and B are given in the above table
for 5 different line speeds.

LEGEND

1. VANCOUVER
2. CALGARY
3. SASKATOON
4. WINNIPEG
5. OTTAWA
6. TORONTO
7. MONTREAL
8. QUEBEC
9. HALIFAX

Figure 7.2
A PROPOSED TOPOLOGY FOR CANUNET

TABLE 7.3

Link reliabilities and link lengths

LINK	RELIABILITY	LENGTH. (MILES)
l_1	0.9	775
l_2	0.92	422
l_3	0.95	329
l_4	0.88	751
l_5	0.9	444
l_6	0.87	1046
l_7	0.85	890
l_8	0.9	218
l_9	0.85	313
l_{10}	0.95	104
l_{11}	0.9	144
l_{12}	0.9	399
l_{13}	0.9	491

TABLE 7.4

File access requirements u_{ij} and v_{ij} (in messages/sec)
and file storage cost (in \$/month) for the
problem of Example 3

NODE	1	2	3	4	5	6	7	8	9
u_{ij}	8	3	3	9	6	3	3	10	3
v_{ij}	2	1	0.5	1.5	2	1	1	2	2
STORAGE COST \$/MONTH	350	330	325	365	350	340	350	360	345

Example 4

The network used in Example 3 (Fig. 7.2) is considered again with two files involved in the allocation (instead of one file as in Example 3). Table 7.5 shows u_{ij} and v_{ij} ; it is assumed that $u_{ij} = u'_{ij}$ and $v_{ij} = v'_{ij}$. File storage costs are also given in Table 7.4. The values of the remaining parameters of the resource allocation problem are:

$$M = 2, \quad \mu^{-1} = 500 \text{ bits}$$

$$\{h_j\} = (2.4, 4.8, 9.6, 19.2, 50) \text{ kilobits/sec.}$$

(Table 7.2 is used to compute the capacity-cost matrix)

$$\bar{T} = 400 \text{ milliseconds}, \quad \{\alpha'_j\} = \{0.9, 0.95\}$$

TABLE 7.5

File access requirements u and v (in messages/sec) and file storage cost (in \$/month) for the problem of Example 4

NODE NO.	FILE 1			FILE 2°		
	u_{i1}	v_{i1}	STORAGE COST (\$/MONTH)	u_{i2}	v_{i2}	STORAGE COST (\$/MONTH)
1	8	2	350	1	1	300
2	3	1	330	4	2	283
3	3	0.5	325	1.5	1	280
4	9	1.5	365	2.5	1.5	314
5	6	2	350	2.5	1	300
6	3	1	340	1	1.5	290
7	3	1	350	5	2	300
8	10	2	360	0.5	1.5	310
9	3	2	345	0.5	1	296

EXAMPLE 5

The network shown in Fig. 7.3 will be considered as an example of large size problems. There are 19 nodes and 68 links; assume that five files are involved ($n = 19$, $2m = 68$, $M = 5$). Reliability and length (in miles) of each of the 34 bidirectional links are shown in Table 7.6. Nodes are considered to be perfectly reliable. Table 7.7 shows u_{ij} and v_{ij} ; it is assumed that $u_{ij} = u'_{ij}$ and $v_{ij} = v'_{ij}$. File lengths are $\{2.0, 2.8, 1.2, 2.0, 3.2\} \times 10^8$ bits and node storage costs are given in Table 7.8. The values of the remaining parameters of the resource allocation problem are:

$$M = 5, \quad \mu^{-1} = 300 \text{ bits}$$

$$\{h_j\} = (2.4, 4.8, 9.6, 19.2, 50) \text{ kilobits/sec.}$$

(Table 7.2 is used to compute the capacity-cost matrix)

$$\bar{T} = 500 \text{ milliseconds,} \quad \{\alpha_j\} = \{0.95, 0.98, 0.86, 0.94, 0.92\}$$

Given the foregoing data, it is required (for each example) to assign link capacities and file locations (with at least one copy of each file in the network), to minimize the combined cost of storage and communications, while realizing the file availabilities, $\{\alpha_j\}$, and incurring an average message delay of no more than $\bar{T}$ milliseconds.

Figure 7.3
A 19-NODE NETWORK

TABLE 7.6
Link reliabilities and link lengths

LINK	RELIABILITY	LENGTH (MILES)	LINK	RELIABILITY	LENGTH (MILES)	LINK	RELIABILITY	LENGTH (MILES)
λ_1	0.95	300	λ_{12}	0.85	470	λ_{23}	0.88	220
λ_2	0.95	400	λ_{13}	0.9	1320	λ_{24}	0.9	380
λ_3	0.93	500	λ_{14}	0.9	1300	λ_{25}	0.9	200
λ_4	0.94	740	λ_{15}	0.9	980	λ_{26}	0.88	260
λ_5	0.91	440	λ_{16}	0.92	1200	λ_{27}	0.8	250
λ_6	0.95	420	λ_{17}	0.88	540	λ_{28}	0.8	270
λ_7	0.95	200	λ_{18}	0.88	520	λ_{29}	0.86	420
λ_8	0.90	460	λ_{19}	0.90	780	λ_{30}	0.86	260
λ_9	0.90	480	λ_{20}	0.90	540	λ_{31}	0.90	340
λ_{10}	0.85	720	λ_{21}	0.92	240	λ_{32}	0.92	540
λ_{11}	0.8	500	λ_{22}	0.95	240	λ_{33}	0.93	510
						λ_{34}	0.85	780

TABLE 7.7

Traffic Matrices
(Problem of Example 5)
(u and v are expressed in messages/sec)

NODE	FILE 1		FILE 2		FILE 3		FILE 4		FILE 5	
	u_{i1}	v_{i1}	u_{i2}	v_{i2}	u_{i3}	v_{i3}	u_{i4}	v_{i4}	u_{i5}	v_{i5}
1	0.8	0.2	0.6	0.2	0.5	3	3	0.3	5	1
2	1.1	0.3	0.8	0.4	0.2	2	2	0.2	1	0.2
3	1.5	0.2	1	0.2	0	5	5	1	1	1
4	6	2	1	0.3	0	2	3	0	1	0.2
5	2	0.5	1.2	0.2	0	3	2	0	1.5	0.3
6	1.5	0.7	1	0.4	0	2	1	0	6	0.2
7	2	1	1	0.4	8	2	1	1	2	0.1
8	3	0.4	1.5	0.3	1	1	5	1	2	0.2
9	8	1	7	2	0.2	3	3	1	3	0.2
10	2	0.2	1	0	0.2	3	4	0.1	3	0.5
11	1	0.2	2	0	0.1	2	4	0.2	2	0.5
12	2	0.4	2	0	0.1	1.5	3	0	2	0
13	2	0.5	4	1	0.2	1.5	2	0	1.5	0
14	1	0.2	4	1	0	2	2	0	2.5	0
15	2	0	3	0.5	0	2	3	0.5	1	0.5
16	1	0	2	0.5	1	1	1	0.5	6	0.2
17	1	0	9	2	0	1.5	2	0	7	0.3
18	2	1	1	0.2	1	3	7	1	0.4	0.2
19	2	0.5	1	0	0.5	3	2	0.3	1	0.4

TABLE 7.8
Storage costs in \$/bit/month
(Example 5)

NODE	1	2	3	4	5	6	7	8	9	10
STORAGE COST × 10 ⁻⁶ \$/bit/month	2.5	2.65	2.6	2.4	2.5	2.5	2.4	2.6	2.75	2.5
NODE	11	12	13	14	15	16	17	18	19	
STORAGE COST × 10 ⁻⁶ \$/bit/month	2.35	2.6	2.6	2.55	2.8	2.7	2.6	2.4	2.5	

Computational Results

In the first example, an optimal (total) cost of \$5649 per month was obtained by both algorithms. The optimal cost corresponds to an assignment of one copy of the first file to node 2 and one copy of the second file to each of nodes 1 and 4. The nonlinear Branch-and-Bound algorithm reached the optimal solution in 6 minutes and 20 seconds of CPU time. The heuristic algorithm reached the same solution in 46 seconds. A saving of approximately eight times in CPU time is thus possible with the heuristic algorithm.

In the second example, an optimal (total) cost of \$45,248 per month was obtained by both algorithms. The optimal cost corresponds to an assignment of one copy of the first file to each of nodes 1 and 3, and one copy of the second file to each of nodes 3 and 5. The nonlinear Branch-and-Bound algorithm reached the optimal solution in 14 minutes and

10 seconds of CPU time. The heuristic algorithm reached the same solution in 2 minutes and 5 seconds. A saving of approximately seven times in CPU time is thus possible with the heuristic algorithm.

In the third example, an optimal (total) cost of \$70,985 per month was obtained by both algorithms. The optimal cost corresponds to an assignment of three copies of the file to network nodes; with one copy assigned to each of nodes 1, 4 and 8. The nonlinear Branch-and-Bound algorithm reached the optimal solution in 24 minutes and 45 seconds of CPU time. The heuristic algorithm reached the same solution in 3 minutes and 40 seconds. A saving of approximately six times in CPU time is thus possible with the heuristic algorithm.

The fourth example represents a moderate size problem. Starting from an initial resource allocation corresponding to a total cost of \$129,550 per month, the nonlinear Branch-and-Bound algorithm reached a cost solution of \$93,105 per month after 25 minutes of elapsed CPU time (see Fig. 7.4). For economic reasons, the algorithm was stopped at this point. The heuristic algorithm required 5 minutes and 40 seconds to yield a cost of \$79,360 per month. The result of the heuristic algorithm was then used to start the nonlinear Branch-and-Bound algorithm. Figure 7.4 shows that cost decrease is slow; a cost of \$72,300 per month being reached after 15 minutes of elapsed CPU time (the algorithm was stopped at this point). It is observed that the total cost obtained after 10 minutes of elapsed CPU time remains the same for the next 5 minutes of CPU time, i.e., no cost improvement was obtained between 10 and 15 minutes of elapsed CPU

time. This situation occurs when the Branch-and-Bound search proceeds for a period of time and no feasible solutions leading to cost improvement can be found during this period of time. Since the algorithm did not terminate after 15 minutes of elapsed CPU time (it was stopped at this point), we cannot conclude that an optimal solution has been found.

The fifth example represents a large size problem. Excessive storage requirements prohibit the application of integer programming techniques to solve problems of this size (see *Storage Limitations* Section). The heuristic algorithm is applied to the problem of this example and the resulting solution is compared with a solution obtained by an intuitive engineering (ad hoc) approach.

An intuitive approach to allocate files and communication capacities will be based on the following steps:

1. For each file j , network nodes are arranged in decreasing order of the amount of retrieval traffic. Each of the first q_j nodes in the ordered set of nodes is assigned a copy of file j (q_j is the number of file copies). The number of copies of file j depends on the ratio of total retrieval traffic to total update traffic,

$$\sum_i u_{ij} / \sum_i v_{ij}$$

2. If the assignment of a given file does not satisfy the availability constraint, then an extra copy of the file is assigned to the next node (node $q + 1$) in the ordered set of nodes. The procedure is repeated until the availability constraint is satisfied.
3. Once files are assigned, the traffic flow over each link can be determined using routing tables. Capacities are then assigned to network links to satisfy the flow constraint (capacity of each link must be greater than the

flow over the link). If this assignment of capacities does not satisfy the delay constraint, then the capacities of one or more links are increased (starting with those links that have large delays). The procedure is continued until the delay constraint is satisfied.

Application of the intuitive engineering (ad hoc) approach to the problem of Example 5 results in the file assignment shown in Fig. 7.5. To illustrate how the intuitive approach was applied to obtain this solution, consider the assignment of files 3 and 5 as examples. The retrieval/update traffic ratio for file 3 is equal to 0.3; the same ratio is equal to 8 for file 5. Because the ratio is low for file 3, only one copy of the file is assigned in the network. The copy is located at node 7 which has the highest retrieval traffic with respect to file 3. The retrieval/update traffic ratio is relatively high for file 5 and thus as many as four copies of the file are assigned in the network. The four copies are located at nodes 17, 16, 6 and 1, since these nodes generate the largest amounts of retrieval traffic with respect to file 5. The file assignment shown in Fig. 7.5 corresponds to a total cost of \$192,200 per month. Application of the heuristic algorithm to the same problem resulted in the file assignment shown in Fig. 7.6. Notice the difference between the file assignment obtained using the intuitive engineering approach (Fig. 7.5) and the file assignment obtained using the heuristic algorithm (Fig. 7.6). The latter assignment corresponds to a total cost of \$87,150 per month. The heuristic algorithm required 16 minutes and 20 seconds to reach this solution. A substantial saving of \$105,050 per month in the total cost is thus possible with the heuristic algorithm.

Storage Limitations

In the nonlinear Branch-and-Bound technique, the search for feasible solutions is conducted along the branches of a binary tree. The branches of the tree correspond to elementary conjunctions of the directrix of the polynomial inequalities. The search along a given branch is terminated when it is determined that no feasible solutions can be found on the extensions of this branch. At this point the algorithm backtracks to the previous branching node to examine the "complementary problem". As the algorithm progresses along a given branch, all the information concerning the nodes on this branch must be stored. This indicates that the storage requirement is proportional to 2^s , where s is a number upper-bounded by the total number of variables of the optimization problem. The storage requirement thus grows very rapidly as the number of variables increases.

In the heuristic algorithm, the search (for potential candidate variables) is conducted by calculating the incremental flows over network links corresponding to the assignment of files to each of the network nodes. The assignment of link capacities is carried out using Dynamic Programming (or Lagrangian Decomposition) technique which requires storage of tables with maximum dimensions equal to $(2 m \times N)$. Thus the storage requirement in the heuristic algorithm is directly proportional to the number of variables.

Table 7.9 shows the storage requirements in kilobytes (1 byte = 8 bits) for the five examples considered in this chapter. It is seen that

Figure 7.4
Total Cost (in \$/month) as a Function
of Elapsed CPU Time

Nodes	19				
	18				
	17				
	16				
	15				
	14				
	13				
	12				
	11				
	10				
	9				
	8				
	7				
	6				
	5				
	4				
	3				
	2				
	1				
Files					
	1	2	3	4	5

Figure 7.5

File Assignment Based on an Intuitive (ad hoc) Solution
(Example 5)

Nodes		19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
Files		1	2	3	4	5														
	1																			
	2																			
	3																			
	4																			
	5																			
	6																			
	7																			
	8																			
	9																			
	10																			
	11																			
	12																			
	13																			
	14																			
	15																			
	16																			
	17																			
	18																			
	19																			

Figure 7.6

File Assignment Based on the Application of the

Heuristic Algorithm

(Example 5)

as the number of variables increases, the storage requirement of the integer programming technique grows much faster than that of the heuristic algorithm. For small size problems, as in Example 1, the storage requirement of the integer programming technique is nearly double the storage requirement of the heuristic algorithm. For moderate size problems, as in Example 4, the ratio is approximately 14 to 1. It is estimated that if the integer programming algorithm is applied to solve the problem of Example 5, a storage space on the order of hundreds of Mega-bytes will be needed. Such excessive storage requirements clearly prohibit the use of integer programming techniques to solve problems with large number of variables.

TABLE 7.9
Storage Requirements
(1 byte = 8 bits)

PROBLEM	$(n \times M + 2m \times N)$	MAXIMUM STORAGE REQUIREMENTS (K.BYTES)	
		EXACT METHOD	HEURISTIC METHOD
Example 1	24	80	36
Example 2	38	152	58
Example 3	61	472	84
Example 4	83	1644	120
Example 5	265	(200,000)* (estimated)	350

7.2 Sensitivity of Optimal File Assignment to Changes in File Availability Requirements

The availability constraint (Chapter 4) requires the file availability, α_j , $j = 1, 2, \dots, M$ (given by eqn. (4.11)) to be greater than or equal to a file availability requirement, α'_j , $j = 1, 2, \dots, M$. It was shown in Chapter 4 that the file availability constraint specifies a lower bound for the number of copies of each file in the network.

Consider a resource allocation problem in which copies of a *single file* are assigned to network nodes and communication capacities are assigned to network links. The file assignment which results in a minimum total cost (storage + communication) and at the same time satisfies delay and availability requirements is defined as the optimal file assignment. Let α' denote the availability requirement for the file. The effect of changing α' on the optimal solution of the resource allocation problem will be examined in this section.

For an information network containing n nodes there are as many as $(2^n - 1)$ possible file assignments; each of these file assignments satisfies the inequality:

$$\alpha > 0$$

where α is given by eqn. (4.11).

Define the set S_i to contain all the file assignments that satisfy the inequality:

$$\alpha \geq \alpha'$$

where α' is equal to A_i ($0 < A_i \leq 1$). For any $A_i > 0$, the number of elements (file assignments) of S_i has an upper bound equal to $(2^n - 1)$. In Fig. 7.7, the sets S_0, S_1, S_2, S_3 and S_4 correspond to values of α' equal to A_0, A_1, A_2, A_3 and A_4 respectively. As the value of α' increases, the number of elements in the corresponding sets of file assignments either decreases or remains constant. Thus if A_0, A_1, A_2, A_3 and A_4 are arranged in the following order:

$$0 < A_0 < A_1 < A_2 < A_3 < A_4,$$

then the corresponding sets of file assignments will be related as follows:

$$S_4 \subseteq S_3 \subseteq S_2 \subseteq S_1 \subseteq S_0$$

Let Δ_i denote the optimal file assignment when the availability requirement α' is equal to A_i . Obviously Δ_i is an element of the set of file assignments S_i . Also let C_i denote the total cost corresponding to the optimal file assignment Δ_i (i.e., the minimum total cost of the resource allocation problem). For any two sets S_i and S_j , the relationship $S_i \subseteq S_j$ implies the following two properties:

- (1) If $\Delta_j \in S_i$, then $C_j = C_i$

As an example, in Fig. 7.7 the optimal file assignment Δ_2 is an element of both S_2 and S_3 . Then the optimal cost C_2 is the same for both sets.

- (2) If $\Delta_j \notin S_i$, then $C_j < C_i$

Properties (1) and (2) indicate that as the availability requirement, α' , increases the optimal (total) cost either increases or remains

constant. Figure 7.8 shows the optimal (total) cost as a function of the file requirements, α' , for the values of α' given in Fig. 7.7. Points A_1' , A_2' and A_3' in Fig. 7.8 are called discontinuity (break) points as they represent values of α at which the optimal (total) cost jumps to a higher level.

Example

Consider the resource allocation problem studied in the example of Section 6.4. The network under consideration consists of four nodes connected as shown in Fig. 6.2. The resource allocation algorithm is applied for different values of α_1' (between zero and one) while all other parameters are held constant at the values given in Section 6.4. The resulting optimal (total) cost in \$/month is shown in Fig. 7.9 as a function of α_1' .

For values of α_1' ranging from slightly over zero to 0.932, assignment of a copy of file 1 to node 2 is the optimal solution and the corresponding total cost is \$5649 per month. For values of α_1' between 0.932 and 0.993, an additional copy of the file is needed and the optimal solution will be obtained by assigning one copy of the file to each of nodes 2 and 4. The optimal (total) cost in this case is equal to \$6289 per month. For values of α_1' greater than 0.993, more than two copies of the file are needed. The optimal cost in this case is equal to \$7109 per month and corresponds to the assignment of a copy of the file to each of the four nodes.

Figure 7.7

Sets of File Assignments Corresponding to Different Values of α' .

$$0 < A_0 < A_1 < A_2 < A_3 < A_4 = 1$$

$$S_4 \subseteq S_3 \subseteq S_2 \subseteq S_1 \subseteq S_0$$

$$C_0 < C_1 < C_2 < C_3$$

Figure 7.8

THE OPTIMAL (TOTAL) COST AS A FUNCTION OF α

Figure 7.9

Effect Of Changing the Availability Requirement on the Optimal Cost

In the previous example, the optimal cost, $C_T(\alpha_1' = 0.9, \alpha_2' = 0.95)$, corresponding to the file availabilities, $\alpha_1' = 0.9$ and $\alpha_2' = 0.95$, is equal to \$5649/month. Assume now that a higher value for the availability, α_1' , of file 1 is required, say $\alpha_1' = 0.95$. Figure 7.9 indicates that an extra copy of the file must be added in the network (assigned to node 4) to realize the new file availability, $\alpha_1' = 0.95$ (point X). The optimal cost in this case is equal to \$6289. An alternative solution to realize the new file availability, $\alpha_1' = 0.95$, may be obtained by improving the reliability of some network links. For example, if the reliability of link ℓ_4 is increased from its current value (0.7) to a higher value (0.85), and using the same file assignment that realizes the availabilities, $\alpha_1' = 0.9$, $\alpha_2' = 0.95$, the availability of file 1 will be greater than 0.95. The total cost in this case will be equal to $C_T(\alpha_1' = 0.9, \alpha_2' = 0.95)$ plus the cost of improving the reliability of link ℓ_4 (from 0.7 to 0.85).

7.3 Effect of File Demand Variations on Resource Allocation

The effect of changing retrieval and update traffic matrices, (U, V) , on the allocation of resources in distributed information networks will be examined in the following discussion.

In the derivation of the file availability, α_j , $j = 1, 2, \dots, M$ (Chapter 4), it was shown that α_j is a function of the variables, x_{ij} 's (column j of matrix X) and elements of the traffic matrices, (U, V) . To indicate this, α_j will be written as $\alpha_j(X, U, V)$. Similarly, the average message delay, T , is a function of the allocation matrices (X, Y) and the

traffic matrices, (U,V) , and thus will be written as $T(X,Y,U,V)$. The allocations (X,Y) is feasible if the availability and delay constraints are satisfied, that is if the following two inequalities are satisfied:

$$\alpha_j(X,U,V) \geq \alpha_j^1 \quad j = 1, 2, \dots, M$$

$$T(X,Y,U,V) \leq \bar{T}$$

Let (X^*, Y^*) denote the optimal allocation (files and capacities) matrices and $C_T(X^*, Y^*)$ denote the optimal cost corresponding to traffic (retrieval and update) matrices, (U,V) , in a given information network. For the same information network, a new resource allocation problem results when the traffic matrices, (U,V) , change to $(\tilde{U}, \tilde{V})$. For brevity, a resource allocation problem in which the traffic matrices are represented by (U,V) will be called problem (U,V) .

Assume that the application of the optimal resource allocation algorithm to problem $(\tilde{U}, \tilde{V})$ results in the optimal allocation $(\tilde{X}^*, \tilde{Y}^*)$ and the optimal cost $C_T(\tilde{X}^*, \tilde{Y}^*)$. If the optimal cost $C_T(X^*, Y^*)$ of the original problem (U,V) is less than the optimal cost $C_T(\tilde{X}^*, \tilde{Y}^*)$ corresponding to the new problem $(\tilde{U}, \tilde{V})$, i.e.,

$$C_T(X^*, Y^*) < C_T(\tilde{X}^*, \tilde{Y}^*)$$

then this implies that (X^*, Y^*) is not a feasible solution for problem $(\tilde{U}, \tilde{V})$. It follows that at least one of the following two conditions is not satisfied:

Availability Constraint: $\alpha_j(X^*, \tilde{U}, \tilde{V}) \geq \alpha_j^* \quad j = 1, 2, \dots, M.$

Delay Constraint: $T(X^*, Y^*, \tilde{U}, \tilde{V}) \leq T$

It should be remembered that the optimal solutions (X^*, Y^*) and $(\tilde{X}^*, \tilde{Y}^*)$ were both obtained using the same traffic routing scheme (traffic routing is assumed fixed in the resource allocation model). It is desirable, however, to apply the less costly solution (X^*, Y^*) to problem $(\tilde{U}, \tilde{V})$. This is equivalent to satisfying the availability and delay constraints mentioned above.

Consider the case where the allocation (X^*, Y^*) satisfies the availability constraint for problem $(\tilde{U}, \tilde{V})$, but does not satisfy the delay constraint for the same problem. In some instances, it may be possible to satisfy the delay constraint, which was otherwise unsatisfied, through changing traffic routing scheme. This problem can be stated as follows:

For a given traffic matrices, $(\tilde{U}, \tilde{V})$, and resource allocation, (X^*, Y^*) , find a set of link flows (traffic routing scheme) which satisfies

$$T(X^*, Y^*, \tilde{U}, \tilde{V}) \leq T$$

This is exactly the *flow assignment* problem mentioned in Section 4.5. An algorithm for solving this problem, called the *flow assignment algorithm* has been developed by Gerla [G1]. If a feasible set of link flows can be found using the flow assignment algorithm, then the allocation (X^*, Y^*) can be applied as a solution to problem $(\tilde{U}, \tilde{V})$.

Figure 7.10 shows a flow chart of a proposed procedure for re-allocating file and capacity resources to meet traffic demand variations. The proposed procedure is based on the above discussion.

In actual network environment, it is possible to change the traffic routing scheme automatically, i.e., without the intervention of human operator. A computer system in one of the network nodes will be designated to process the algorithm given by the flow chart of Fig. 7.10. If a change in traffic routing scheme is required, new traffic routing tables for each front-end-processor can be constructed using the results of the flow assignment algorithm. Messages containing these tables can be transmitted over the communication subnetwork from the designated computer to the different front-end-processors. The tables can be loaded into the memory of a front-end-processor using an automatic loader program which is resident in a protected ROM, as shown in Fig. 7.11. Thus the traffic routing scheme can be changed with minimum disruption to the operation of the communication subnetwork.

7.4 Summary

The nonlinear branch-and-bound algorithm and the heuristic algorithm were applied to solve several optimization problems based on the resource allocation model (Chapter 4). Computational results for a number of examples representing problems with small, moderate and large number of variables were reported in Section 7.1. The following conclusions are drawn from computational experience with both algorithms:

INITIAL SOLUTION (X^*, Y^*)

APPLY THE RESOURCE ALLOCATION
(HEURISTIC ALGORITHM) TO
 $(\bar{U}, \bar{V})$ THE RESULT WILL BE THE
NEW ALLOCATIONS $(\tilde{X}^*, \tilde{Y}^*)$

IS
 $C_T(\tilde{X}^*, \tilde{Y}^*) \leq C_T(X^*, Y^*)$
?

YES

TERMINATE: SOLUTION
IS GIVEN BY $(\tilde{X}^*, \tilde{Y}^*)$.

NO

NO

IS AVAILABILITY
CONSTRAINT SATISFIED
FOR $(X^*, \bar{U}, \bar{V})$?

YES

TERMINATE: SOLUTION
IS GIVEN BY (X^*, Y^*)

YES

IS DELAY
CONSTRAINT SATISFIED
FOR $(X^*, Y^*, \bar{U}, \bar{V})$
?

NO

TRY FLOW ASSIGNMENT
ALGORITHM WITH
 $(X^*, Y^*, \bar{U}, \bar{V})$

POSSIBLE
?

NO

YES

TERMINATE: SOLUTION IS
GIVEN BY (X^*, Y^*)

Fig. 7.10. A Procedure for Re-allocation of Resources to Meet Traffic Demand Variations

Figure 7.11
AUTOMATIC LOADING OF TRAFFIC ROUTING SCHEMES

- In most problems with small number of variables, optimal solutions were possible to obtain using any of the two algorithms. However, the heuristic algorithm proved to be more efficient than the integer programming algorithm; savings of 5 to 10 times in CPU time were shown to be possible with the heuristic algorithm.
- In most problems with moderate number of variables, no convergence to the optimal solution was possible with the integer programming algorithm (the algorithm was stopped if no optimal solution was found in 25 minutes of elapsed CPU time). The heuristic algorithm, on the other hand, was shown to yield good sub-optimal solutions for problems of this size, leading to significant savings in both CPU time and computer storage space.
- Excessive storage requirements prohibit the application of the integer programming algorithm to solve problems with large number of variables. When applied to solve these problems, the heuristic algorithm was shown to yield solutions that are considerably less than the cost solutions obtained using an intuitive engineering (ad hoc) approach. The heuristic algorithm required moderate amount of computer processing time and storage space to reach these solutions.

Sensitivity of Optimal Resource Allocations to Variations in File Availability and Traffic Demand Requirements:

In Section 7.2, the effect of changing the file availability requirements on the optimal assignment of resources was examined. It was shown

that the optimal cost increases or remains constant (step-wise function) as the file availability requirement increases. Such a sensitivity analysis is useful in cases where the file availability requirements change; for example, when the resources were originally allocated in the network to realize certain file availability but at some future time, a higher file availability is required and cannot be satisfied with the original allocation of files. Two alternative solutions to realize the higher availability requirement are possible. In the first solution, files are re-allocated so that the new availability requirement is satisfied. This may result in an increase in the number of file copies, and consequently, an increase in the total cost. The same result, i.e., higher file availability can be also obtained by improving the reliability of some network elements while maintaining the original allocation of resources.

The effect of changing the node-file traffic demands on the optimal assignment of resources was examined in Section 7.3. It was shown that changing the file demand matrices, (U,V) , may require re-allocation of resources which leads, in some instances, to a higher total cost. The possibility of reducing the communication cost through varying the traffic routing scheme was studied (traffic routing scheme is assumed fixed in the resource allocation model). A procedure which employs a flow-assignment algorithm to meet the changes in the node-file traffic demand values was proposed in Section 7.3. It can be concluded intuitively that this procedure leads to a better utilization of the total transmission capacity. However, computational experience will be needed to verify its effectiveness.

The main effort in this dissertation has been so far concentrated on the design (resource allocation) problem which is usually associated with early phases of network development. Following the design phase, a number of operational aspects related to network applications and services must be considered. Examples are the methods employed for accessing the distributed file system. Once files are allocated in the network, means must be provided to:

- associate users active jobs and executions with local and/or remote files;
- control and schedule access to files; and
- avoid or resolve the conflicts that may arise as many independent user jobs share access to the same set of files.

Such operational aspects are extremely important in real information network environments as they involve a number of technical problems that must be solved before distributed data bases can be widely used. In fact, the design effort associated with early phases of network development can be justified only if economically feasible and implementable solutions to these problems are available. Development of such solutions to some of the technical problems associated with the operation of distributed information networks will be the subject matter of the following two chapters.

CHAPTER 8

OPERATIONAL ASPECTS OF DISTRIBUTED DATA BASES

In this chapter, alternative methods of creating and realizing a distributed data base are discussed. The different problems associated with some of these methods are defined. This will be followed by an introduction to the approaches used to handle some of these problems. A detailed presentation of these approaches is provided in Chapter 9.

8.1 Methods of Creating a Distributed Data Base

Two types of files may exist in a distributed data base, namely; individual files and common (shared) information files. In information networks containing the second type, grouping, splitting and duplication of information files may be feasible from both an economic and operational point of view.

Individual (private) Information Files

These are built, updated and modified by a user for his own particular use. Each user creates his files at whichever information processor he uses. In practice, users access the closest information processor and as a result, files will be usually optimally located in a geographic sense.

Common (shared) Information Files

A distributed data base which consists of shared information files has a wide number of applications and consequently is much more interesting to investigate. Company files as contrasted to private files can serve as

an example of shared information files. Files which fall into this category include payroll records, order files, customer records, inventory records and other similar data. These files do not belong to an individual user, but to the company, or to any of its divisions.

A distributed data base is made up of at least one, but more often multiple files. The term "File" as defined by Booth [B11] is used to signify a logical entity; a collection of records logically related because of the owner's application.

Methods of creating a distributed data base which includes shared information files will be discussed next.

Grouping Files

The multiple files which form a distributed data base can be separated into groups, and each group attached to one of the information processors in the network. In most applications, files are grouped either geographically or by type. The latter assumes some specialization of the information processors within the network. As an example of grouping by type, all financial files of a company might be at one location while all inventory files at another, and so on.

In grouping files based on geographical locations, tradeoff considerations between storage and transmission costs must be studied. This is essentially a function of the node-demand matrix of the various file groups as well as the type of traffic generated (query and update transactions) as was shown in the resource allocation model (Chapter 4).

It is important to note that while groups of files are distributed in the network, each logical file is kept intact.

File Duplication: The Multiple Copy File System

Duplication of a certain file means that two or more identical copies of the file are stored in different nodes in the information network. Any of these copies may be accessed for record retrieval. All the copies of a file must be updated simultaneously in order to remain identical. Duplication may be needed for one of the following reasons:

- To increase access to the file by providing more paths to it, or more memory and/or processors allowing access to the file.
- To provide rapid back up in case of the failure of a device, communication link or an information processor accessing the file, i.e., to increase the file availability.
- To decrease communication volume and/or dependencies on the communication facilities between the information processors.

It was shown in Chapter 7 that if the query traffic generated from the different nodes is large enough, then duplicating the file system would decrease the amount of generated traffic. Consequently, the capacities of the communication links needed to meet a certain delay constraint will be smaller, leading to a reduction in the total cost.

Logical Splitting of Files

Another method of creating a distributed data base is to split a logical file or files between information processors. Part of a logical file is then attached to one information processor, part to a second one

and so on. Two main reasons for splitting a logical file may exist:

(1) Load size factor and (2) The geographical distribution factors.

As mass storage technology improves and very large capacity devices are built, large information processors today can support storage in the range of hundreds of billions of characters. While this trend is taking place, it is quite noticeable that access paths to these devices have not improved proportionately. If a large file is highly active, i.e., it has a high access rate, then it might not be practically possible for an information processor to support it.

Load, therefore, is one reason why a logical file might be split between two or more information processors. The other main reason for splitting a file is due to the fact that accesses to different parts of it originate in different locations. For example, a file of all customer records for a company would form a logical entity. Because customer orders may originate in different geographical centres, communications economies might make it reasonable to locate information processors in more than one centre. Associated with the concept of splitting logical files, there exist many difficult problems, the solution of which requires additional complex software and consumes part of the computer resources.

8.2 Associating Interactive Executions with the Distributed File Systems

In the operation of information networks supporting distributed data bases, means must be provided to associate jobs or interactive executions with the file(s) needed. Generally, associating jobs with file(s)

can be implemented as a set of processes run under user control or under system (software) control.

Processes Under User Control

In this case all file placement and job addressing for file access is manually controlled. Each user is asked to address all of his input to a specific information processor, usually the one closest to his terminal. When files are created by a user, they will be attached to the information processor to which he has access. If a user wishes to access another user's file, he must find out first where it is located, and then addresses his job or interaction to that information processor.

If a file movement is necessary, users must be notified so that they can change the location to which they send jobs accessing the file.

Processes Under System (software) Control

In some applications where users may require a heavy use of files in multiple information processors accessed by a single job application, and/or heavy use of files remote from the user's local information processor, the above scheme is obviously unworkable. In cases like these, software knowledge of file location is necessary. Jobs or interactions can then be directed automatically to the correct location for execution. Many approaches can be adopted to allow for automatic access under software control of segments of the data base. Examples of these approaches are the "Full-cataloging" approach, "Inter Information Processor Enquiry"

3

OF/DE

4

approach and the "File Name Prefix Use". Discussions of the applicability of these approaches to different cases and configurations of distributed data base systems are detailed by Booth [B11].

8.3 The Distributed File Access Methods

Access requirements in information networks supporting a distributed data base can be categorized as follows:

- (a) The simplest type of job requires access to a file or files connected to a single information processor.
- (b) Another type of job retrieves or updates at one location, and also needs to retrieve a number of records at one or more other locations.
- (c) A third type of job retrieves or updates at one location, and also requires a number of updates at one or more other locations.

The volume of update or query requests for remotely located files greatly influences the choice of the access method that should be adopted in a given network. In the following we shall examine the possible ways of providing the types of access mentioned in (b) and (c).

8.3.1 Transmission of file copies

A complete copy of a file can be transmitted from its home location to a remote information processor, so that it can be used by a job executing there. The decision to do so can be made either automatically by the operating system or manually by the user with the file access requirement. This technique allows access to multiple files, for either enquiry or update, provided that a control is exercised over the file

access so that the file will be locked while remote update is in progress.

There are some obvious restrictions on the use of this technique:

- First, some limit must be set on the size of files which can be transmitted in this way. Depending on the number of accesses required, this method may reduce transmission time and overhead. There is a break-even point at which access volume makes it less costly to send the entire file than to send individual data items (see Fig. 8.1).
- Second, if the user wishes to update the file copy which is transmitted to the remote location, care must be taken to prevent interference with any updates in the home location. This can be done by preventing update at the home information processor until an updated copy is returned from the other location.

8.3.2 Chaining of jobs with multi-access applications

This method [B11] can be used when one application requires access, particularly update access, to files attached to more than one information processor. The job which would otherwise update multiple files at multiple locations can be rearranged as a series of jobs chained together by means of a job activating scheme. Each job in such a chain can update only the files in a specific information processor. As each job finishes, it activates the next job in the chain, generally in a different information processor. This method requires that the user be aware of the exact file distribution among the information processors.

An example of job chaining is shown in Fig. 8.2. The first job might access file A at information processor 1, updating it and extracting data for reporting purposes. This data is included in a new job to be activated. This job is then directed to information processor 2 for

FIGURE 8-1
File Transmission Cost vs. Traffic Volume

- b IS THE BREAKEVEN POINT. IF THE TRAFFIC VOLUME GENERATED IN SOME SESSION IS TO THE LEFT OF b, THEN TRANSMITTING THE ENTIRE COPY OF THE FILE IS MORE COSTLY;
IF IT IS TO THE RIGHT OF b, THEN THE OPPOSITE IS TRUE

Figure 8.2
Job Chaining in Distributed Information Networks

execution where it updates Files B and C. It consolidates the information obtained during both updates. This sequence can be continued as many times as necessary to access all required files. The last job in the chain completes any required reports, and sends them either to the originator, or to any other desired location in the network.

The advantages of using this technique are:

- its simplicity from the operating system's point of view since all complex logic becomes the user's problem;
- avoiding access delay and job interference problems (see Sections 8.3.3 and 8.3.4) since each job execution at each location is independent of the others, except for the passing of control.

The disadvantages of using this technique are:

- the application programmer must continuously monitor the location of each file;
- each job in a chain is independent of the preceding jobs, once it is activated. Thus if an abnormal termination of a job occurs somewhere in the chain, it is difficult to remove the effects of its predecessors. For example, if job number 2 on a chain consisting of three jobs terminates, the updates performed by the first job remain in effect;
- there is no way of presenting an unchanging picture of data base status. This is in contrast to the case of an exclusive access job running in a single information processor where a completely static picture of the entire data base can be obtained.

8.3.3 Software controlled access methods

The information processor's operating system can provide automatic access to file data at other locations, on an individual request basis.

The sequence of events to handle such an access would be roughly as follows:

- User program issues request for file data
- The operating system discovers that the file is attached to another information processor.
- An I/O request is sent to the appropriate information processor, using a very high transmission priority. — —
- The other information processor receives the request, processes it, and returns either the requested data or a denial.

Such an approach, usually referred to as software controlled access, has several attractive features; the most obvious one is its simplicity from the user's point of view since in this mode the user need not be concerned with file location or separation. On the other hand the software controlled access method has two serious problems, viz., job interference (deadlock situations), and the task of controlling access to files.

Job interference cases (deadlock conditions) occur in some instances as when independent jobs accessing the same files are run in different processors. An example of a deadlock condition is shown in Fig. 8.3. Each of the two processors 1 and 2 has a file locked at a different location, while both processors are blocked waiting for access to the other's file.

8.3.4 Delayed and synchronized updating of files with multiple copies

Delayed updating is another mechanism which can be used to provide access to files at multiple locations. It allows multiple jobs to retrieve data concurrently from an apparently unchanging file. Changes made by each job are posted not to the main file, but to private change files. These private change files are inaccessible to any other job.

Figure 8.3

Deadlock State

PROCESS 1 HAS FILE A, AND REQUIRES FILE B

PROCESS 2 HAS FILE B, AND REQUIRES FILE A

Figure 8.4 shows two concurrently updating programs in the same information processor. Since changes are inaccessible until they are applied to the main file, they cannot affect other jobs in concurrent execution. Delayed update changes can be applied to the main file when each job finishes execution successfully, or more often if needed. Multiple delayed update jobs can be run concurrently if there is a program available which can synchronize concurrent changes to the same record within the file. Such a synchronization program must be tailored to the format and conventions of the file type (indexed sequential, inverted, linked, etc.). For applications involving complex file structures, the synchronization program will be complicated.

Synchronizing can also be used, without delayed update, to allow simultaneous updating of multiple copies of a file, all of which are kept constantly current. Updating a file with multiple copies requires a check of the access status of all file copies in the network. File update applications are not allowed to proceed before all copies of the file are synchronized (locked under the control of the updating process).

Another form of job interference occurs when synchronized updating of files with multiple copies is employed. The three-node information network shown in Fig. 8.5 illustrates the problem. Two identical copies, j_1 and j_2 , of file j are stored in the information processors of nodes 1 and 2 respectively. Assume R to be a set of records contained in the two identical copies j_1 and j_2 . Also assume that process P_1 which is resident in information processor of node 1 will start a file update sequence which

Figure 8.4
DELAYED UPDATE

INFORMATION PROCESSOR 1

INFORMATION PROCESSOR 2

INFORMATION PROCESSOR

Figure 8.5

ASYNCHRONOUS UPDATING OF A 2-COPY FILE IN A 3-NODE NETWORK

will change the set of records R to a new set R_1 . Similarly, process P_2 which is resident in information processor of node 2 may change the set R to a new set R_2 . This can be expressed as:

Assume that P_1 and P_2 start their updating sequences at time t . While the copy of the file which is local to the updating process is accessed immediately by the process, the request to update the other copy may reach the remote processor after some time delay (introduced by the network). Thus the order of the sequence of changes may be different in the two copies of the file as follows:

A third process P_3 resident in the information processor of node 3 with a request for query of file j may find different versions of the file depending on whether node 1 or node 2 is accessed.

8.4 Management of tasks and resources in distributed information networks

"The Distributed Data Base Management Facility"

It is apparent from the discussions of Sections 8.3.3 and 8.3.4, that in order to avoid job interference, the operating system providing the software access service must be capable of:

- recognizing dependencies between jobs, detecting and resolving deadlock situations;
- controlling recovery of an abnormally terminated job which was updating files at multiple locations;
- synchronizing the update processes of multiple-copy files within the information network.

Two approaches to the design and implementation of a software controlled access system are proposed. In the first approach, information concerning file access status and active user processes is exchanged between processing nodes, so that the control is *distributed*. In the second approach control resides in a designated node, which is referred to by all active user processes when requests for file access are initiated; control is thus *centralized*. The software system required to control the shared access to the file system will be called *the Distributed Data Base Management Facility* or the DDBMF.

In Chapter 9, three general abstract models for deadlock detection and recovery or deadlock prevention are presented. These models are used to construct the logical functions of the DDBMF. The two approaches proposed for the design and implementation of the DDBMF (the centralized approach and the distributed approach) along with the network protocol problem associated with each approach are also presented in Chapter 9.

CHAPTER 9

SOFTWARE-CONTROLLED ACCESS TECHNIQUES

This chapter examines job interference and file access control problems in distributed information networks. Section 9.1 discusses briefly the access modes and degrees of protection required by files and user processes in a distributed information network environment. Sections 9.2 and 9.3 present two general abstract models for deadlock prevention. A model for deadlock detection and recovery is introduced in Section 9.4. Any of these models can be used to construct the logical functions of the software controlled access system (the *Distributed Data Base Management Facility* or DDBMF). Section 9.5 presents the concepts underlying the centralized and distributed control approaches for the design and implementation of the DDBMF. The protocol problem associated with each approach is examined in Section 9.6. Results of simulation experiments carried out on a network of information processors supporting a group of data files shared by a distributed user group are reported in Section 9.7. The same section examines the two approaches in the light of several performance criteria and suggests guidelines for selecting the software controlled access approach which is most suitable for a given information network.

9.1 Distributed File Access Modes

In general, there exist three access modes and consequently three degrees of protection required by the files (resources) and their user processes (tasks) in a distributed information network environment (see Refs. [M3] and [B11]).

Update or exclusive control: a user process requesting this mode of access will be the only one having an access to all copies of the requested files. This mode of access will be referred to as mode E.

Retrieval or shared control: a user process requesting this mode of access may share control over the requested files along with other processes requesting the same mode of access. The requested files must be protected against any attempt to change its records during the retrieval process, i.e., against any updating process. This mode of access will be referred to as mode S.

Unprotected retrieval: a user process requesting this mode of access does not require any protection against any updating process and does not interfere in any way with other users.

All user processes requesting the first two modes of access seek control of the requested files by name through the DDBMF. It is assumed that requests for a given file are queued on a first-come-first-serve basis. It is the responsibility of the DDBMF then to assign control of requested files to the requesting processes and arrange the file queues properly to avoid the problems of job interference. This will be done based on one of the models described in the following three sections.

9.2 The Maximum Limit (Claim) Model

The system considered in this model consists of a set of n sequential processes $\{P_1, P_2, \dots, P_n\}$ and a set of m different resources $\{R_1, R_2, R_3, \dots, R_m\}$

($n, m \geq 1$). In general, m is a fixed number while n can be a variable.

Each process must state, before any requests for resources are made, its maximum need for each type of resource. (Resources of one type are assumed equal in the sense that they can perform the same tasks.) A process P_i is free to seize and release, in any order, the resources claimed and as a result the allocation strategy takes into consideration the fact that P_i might need all its claimed resources at once.

Definitions

Define the vector $\bar{c} \equiv (c_1, c_2, \dots, c_m)$ whose element c_i , ($i = 1, 2, \dots, m$), represents the total units available of resource i in the system, to be the vector of available resources. Also define the matrix $D \equiv (\bar{d}_1, \bar{d}_2, \dots, \bar{d}_n)$ to be the maximum claim matrix where $\bar{d}_j$, ($j = 1, 2, \dots, n$) represents the maximum claim vector for process j . The component, d_{ij} ($i = 1, 2, \dots, m$), of the vector $\bar{d}_j$ represents the maximum number of resources of type i that will be needed at one time by process P_j . Similarly define the allocation matrix $A \equiv (\bar{a}_1, \bar{a}_2, \dots, \bar{a}_n)$ where $\bar{a}_j$, ($j = 1, 2, \dots, n$) represents the allocation vector of process j . The component a_{ij} ($i = 1, 2, \dots, m$) of the vector $\bar{a}_j$ represents the number of resources of type i allocated to process P_j .

The allocation state, Γ , of the system is determined by specifying the vector $\bar{c}$ and the two matrices D and A . A given allocation state, written as $\Gamma \equiv (\bar{c}, D, A)$, is specified by assigning numerical values to $\bar{c}$, D and A . A transition from one allocation state to another allocation state occurs when some of the elements of the matrix A change their values,

i.e., when processes acquire or release units of available resources. For this reason we write the matrix A as a function of time and $\Gamma(t) \equiv (\bar{c}, D, A(t))$ will describe the allocation state at time t .

An ordering relation, " $\leq$ " will be used between vectors and also between matrices. The relation " $\leq$ " is defined as following:
for any two vectors $\bar{u}$ and $\bar{v}$, where $\bar{u} \equiv (u_1, u_2, \dots, u_m)$ and $\bar{v} \equiv (v_1, v_2, \dots, v_m)$,

$(\bar{v} \leq \bar{u})$ means $v_i \leq u_i$ for all elements $i = 1, 2, \dots, m$.

Also for any two matrices, $F = (\bar{f}_1, \bar{f}_2, \dots, \bar{f}_m)$ and $G = (\bar{g}_1, \bar{g}_2, \dots, \bar{g}_m)$,

$(F \leq G)$ means $\bar{f}_i \leq \bar{g}_i$ for all elements $i = 1, 2, \dots, m$.

An allocation state is said to be a realizable state if it satisfies the following three conditions:

Condition 1: no process claims more resources than are available, i.e.,

$$\bar{d}_j \leq \bar{c} \quad \text{for all } j = 1, 2, \dots, n$$

Condition 2: no process will seize resources more than it has claimed, i.e.,

$$A \leq D$$

Condition 3: at most all resources are allocated, i.e.,

$$\bar{r}(t) \geq 0 \quad (\bar{r}(t) \text{ is called the vector of unused resources})$$

where

$$\bar{r}(t) = \bar{c} - \bar{z} \quad \text{and,}$$

$\bar{z}$ is an $(m \times 1)$ vector whose element z_i is given by:

$$z_i = \sum_{j=1}^n a_{ij} \quad (i = 1, 2, \dots, m)$$

A realizable state does not guarantee that a deadlock cannot occur, i.e., a state may be realizable in the sense that it satisfies the three conditions mentioned above and still contains the deadlock danger.

A realizable state that does not contain the deadlock danger is said to be a safe state; otherwise it is called an unsafe state. In the following, a formal definition for safeness is provided.

A realizable state is said to be safe if the n processes can be ordered in a sequence $j_1, j_2, \dots, j_n$ so that the allocation vectors, $\bar{a}_{j_i}$ ($j_i = 1, 2, \dots, n$), and the unused resources vector $\bar{r}(t)$ satisfy the following set of inequalities:

Condition 4:

$$\bar{d}_{j_1} - \bar{a}_{j_1}(t) \leq \bar{r}(t)$$

$$\bar{d}_{j_2} - \bar{a}_{j_2}(t) \leq \bar{r}(t) + \bar{a}_{j_1}(t)$$

$$\vdots$$

$$\vdots$$

$$\bar{d}_{j_n} - \bar{a}_{j_n}(t) \leq \bar{r}(t) + \bar{a}_{j_1}(t) + \bar{a}_{j_2}(t) + \dots + \bar{a}_{j_{n-1}}(t)$$

The ordered sequence of processes is said to be complete if it contains all n processes.

Condition 4 states that the claim by any process P_i must not exceed the sum of the free resources and these resources which will become free "in due time", when the processes preceding it in the ordered sequence have released theirs. It is worthwhile to mention that the concept "safe" implies that starting from a safe state there is at least one way to allocate the claimed resources to each process P_i , even in the worst case, i.e., when each process P_i asks for all the resources it has claimed and it does not release any resources until it has been allocated all its claimed resources.

Habermann [H7] used the above model to derive algorithms that will test the safety of any realizable state. The algorithms were derived based on the results of three theorems. Statements of these theorems are given below and their proof is available in [H7].

Theorem 1

The processes will not get into a deadlock iff the allocation state is safe provided that no process will release the resources under its control until it has been allocated to all its claimed resources.

An algorithm to check the safety of any realizable stage, $\Gamma(t) \equiv (\bar{C}, D, A(t))$, will try to arrange the processes in some order so as to satisfy condition 4. This is an exhaustive procedure since the processes can be ordered in $n!$ ways. Theorem 2 is useful in reducing this search effort.

Theorem 2

If the allocation state is safe and a subsequence, S , which fulfills condition 4 is found, then S can be extended into a complete safe sequence.

Theorem 2 considerably reduces the task of finding a complete sequence. It implies that for any safe state, any subsequence fulfilling condition 4 can be extended into a safe sequence. As an immediate result, when a subsequence S cannot be extended it is unnecessary to look for another safe sequence under the $n!$ possible sequences since a safe sequence does not exist at all and so no backtracking is required.

As will be shown later, it is important to know whether a transition from one state to another can be done safely, i.e., the new state does not lead to a system deadlock. Suppose that the system is in a safe state, $\Gamma(t) \equiv (\bar{C}, D, A(t))$, and a process, P_i , asks for some of its claimed resources. Process P_i may be allowed to seize the resources if the transformed state is also a safe one. The result of Theorem 2 allows us to construct an algorithm for finding a full sequence fulfilling condition 4 in order to decide whether the transformed state is safe. Theorem 3 will allow further reduction in the search effort.

Theorem 3

If a safe state is transformed by allocating resources to process P_i and if any sequence (or subsequence), S , can be found containing P_i and fulfilling condition 4, then the transformed state is also a safe one.

Theorem 3 allows a further reduction of the state inspection task, especially when resources are not scarce, because many safe sequences may exist and it will be possible then to allow P_i to finish, regardless of the processes after P_i .

Habermann [H7] used the results of Theorems 2 and 3 to construct algorithms that will test the safety of a transformed state and thus allow for deadlock prevention. It should be noted that the algorithms derived do not impose any restrictions on the priority rules assigned by the system to the different processes.

The application of the maximum limit (claim) model to the distributed file problem will now be examined. An information file with all its copies in the information network will be considered as a single unit of resource. The set of distinct information files constitutes the shared resources under consideration. User processes represent the tasks competing to have access to one or more of the information files in order to read, write and/or execute data records. To prevent conflicts arising from simultaneous requests and previous acquisitions of the files by the different processes which may lead to deadlock situations, a mechanism is needed to manage the access to the files in such a way that will prevent the deadlock hazards. The way such a mechanism is implemented (centralized versus distributed) will be discussed later. At the present, the interest is centred around deriving its logic based on the Maximum Limit (claim) model.

Based on the above, and given that the number of the shared information files is m and the number of simultaneously active processes

is n , it follows that each entry in the vector of available resources, $\bar{c}(1 \times m)$, will be equal to 1,

$$\bar{c} = (1, 1, \dots, 1), \quad \bar{c}(1 \times m).$$

The maximum claim matrix D is formed by asking every user process, P_j , entering an active session, to provide information about the files that will be accessed simultaneously during this session. This will be indicated by the entries of the demand vector, $\bar{d}_j$, corresponding to the indices of requested files; a 1 entry in location i of $\bar{d}_j$ means an access to file i is required by process P_j during this active session, and a 0 entry means otherwise. The information carried by vector $\bar{d}_j$ is incomplete since it does not provide the access mode required by the process for a given file. For this reason, an additional vector $\bar{d}_j^*$ ($m \times 1$) is introduced. Element i in vector $\bar{d}_j^*$ ($m \times 1$) corresponds to file i and has one of three possible values; a zero value means file i is not needed at all during the active session of process P_j , a 1 entry means process P_j is demanding a shared control mode of access of file i and a 2 entry means process P_j is demanding an exclusive control (update) mode of access of file i during the process active session. The set of vectors, $\bar{d}_j^*$ ($j = 1, 2, \dots, n$), form the matrix $D^*(m \times n)$. Similarly a matrix $A^*(t)$ can be introduced to complete the information provided by the allocation matrix $A(t)$ defined previously.

The allocation state, $\Gamma(t)$, of the distributed file system at time t is completely specified by $(\bar{c}, D, D^*, A(t), A^*(t))$ and the safety of a transition from a realizable safe state can be tested using algorithm 1.

Algorithm 1

This algorithm tests the transformation of a realizable safe state $\Gamma_1(t)$ by a process P_j to another state $\Gamma_2(t)$ to see if $\Gamma_2(t)$ is also a safe state and thus the transformation is safe, which allows process P_j to acquire its requested resources. The result of Theorem 3 indicates that $\Gamma_2(t)$ is a safe state if a subsequence of processes is found which satisfies condition 4 and contains process P_j . In testing to see if a process, P_k , satisfies any of the inequalities of condition 4, the vector of the unused resources, $r(t)$, will be augmented by another vector $r^*(t)$ to completely specify the information carried by $r(t)$. If the entry i in $r(t)$ is zero, the corresponding entry in $r^*(t)$ will also be zero, indicating that file i is not available. If the entry i in $r(t)$ is 1, then the corresponding entry in $r^*(t)$ may be any positive integer greater than zero. The appearance of 1 in entry i of $r(t)$ and a number, say 3, in the corresponding entry of $r^*(t)$, indicates that file i is available only to processes requesting shared control access (mode S) and that there are three processes presently accessing file i in shared (retrieval) control mode of access. If entry i of $r(t)$ is 2, the corresponding entry in $r^*(t)$ will be zero and this indicates that file i is available to any process requesting any mode of access (mode E or mode S), since no other process is using the file at the time the test was conducted. The flow chart of Algorithm 1 is given in Fig. 9.1.

Figure 9.1
Flow Chart of Algorithm 1

9.3 The Demand Graph Model

A demand graph is a finite directed graph with arcs and nodes. Associated with each arc is a vector with finite dimensions ($1 \times m$), chosen from a set Δ and called the arc demand vector. The set Δ is defined to be the set of m -tuples of non-negative integers for some specified m . A vector called the capacity vector, chosen from the set Δ is associated with the demand graph. If $\bar{c}$ represents the capacity vector and $\bar{d}(\theta)$ refers to an arc demand vector, then the relationship:

$$\bar{d}(\theta) \leq \bar{c}$$

holds for all the arcs of the demand graph. The relation " $\leq$ " has the same definition stated in the previous section. The demand graph considered here consists of a set of disconnected components with every component containing at least one node that has in-degree zero and one node that has out-degree zero. The components of the demand graph are unilateral, i.e., for every pair of nodes at least one node is reachable from the other by a path. For this reason, the components of the demand graph are called chains. The section of the chain between any two nodes is called a segment of the chain. An arc of a chain is obviously a segment of that chain. The demand vectors associated with the first and last arcs of each chain are equal to $\bar{c}$. These arcs are called the initial and terminal arcs of the chain respectively.

The Demand Graph Model is applicable to the class of systems of processes in which resources are shared. The chains of the demand graph

correspond to processes in the system represented by the graph. The nodes correspond to the epochs at which a change in the resource usage occurs and for this reason they are usually called transitions. The arcs of a chain correspond to the phases of activity of the process represented by the chain, i.e., periods of steady resource usage. The demand vectors associated with the arcs of the graph represent the demands for resources associated with the corresponding phases of activities of the processes. The value of element i , ($i = 1, 2, \dots, m$), of a demand vector represents the number of units of resource i required during this phase of activity of the process. As the demands are integers, the processes modelled share different types of resources; each resource forms a common pool with identical units. The value of element i , ($i = 1, 2, \dots, m$), in the capacity vector, $\bar{c}$, represents the total number of available units of resource i in the system or the number of servers in the pool of this resource. The requirement that adjacent arcs have distinct demands is consistent with the fact that the nodes (transitions) represent changes in resource usage.

An example of a demand graph is shown in Fig. 9.2. The chains $P_1, P_2, \dots, P_n$ form the demand graph. The number of arcs of chain P_i is denoted by K_i and the demand vector of arc j on chain P_i is denoted by $\bar{d}_i(\theta_j)$. The first (initial) and last (terminal) arcs on each chain have a $\bar{0}$ demand vectors. The chains of the demand graph model shown in Fig. 9.2 are disjoint from each other. This means that the processes that are represented by these chains are independent. The only interaction between processes is that due to sharing of resources.

$\bar{c} \equiv$ RESOURCE CAPACITY VECTOR

$\bar{d}_i(\theta_j) \equiv$ DEMAND VECTOR OF ARC j ON CHAIN P_i

$K_i \equiv$ NUMBER OF ARCS OF CHAIN P_i

Fig. 9.2. Demand Graph Model

9.3.1 Definitions and notations

Define the *cut set* of a demand graph to be the set of arcs, one arc from each chain; the cut set is said to intersect the chains in the respective arcs. The cut set partitions the nodes of a demand graph into two sets; those that lie above the cut set make up its *predecessor set* and those that lie below the cut set make up its *successor set*. The *initial cut set* of a demand graph consists of the set of initial arcs and the *terminal cut set* consists of the set of terminal arcs of the graph.

In the following discussion, the cut sets of the demand graph will be represented by the symbol γ . The initial and terminal cut sets are denoted by γ_0 and γ_T respectively. The arc from a chain P_i that goes into a cut set γ is represented by $\gamma \sim P_i$ and the demand vector of this arc by $\bar{d}(\gamma \sim P_i)$.

Figure 9.3 shows a demand graph representing a system with one resource, i.e., a scalar resource system, with $c = 10$. The cut set γ intersects the chains P_1 and P_2 in the two arcs whose demands are equal to 3 and 5 resource units respectively. Following the above notations, the demands on these arcs can be expressed as:

$$d(\gamma \sim P_1) = 3 \quad \text{and,}$$

$$d(\gamma \sim P_2) = 5$$

The initial cut set γ_0 and the terminal cut set γ_T are also shown in the demand graph of Fig. 9.3.

The cut sets of a demand graph represent all the states of the system of processes; the arcs composing a cut set indicate which phase each process is in. The state of the system is called the allocation state since in this model demands are considered allocations. This is in contrast with the maximum claim model in which the maximum claim made by a process for resources may be different from the resources allocated to the process in a given state. The allocation state of the system before any process is initiated is represented by the cut set γ_0 . As the processes progress in their activities (after they are initiated) the cut set representing the current state moves to lower and lower positions in the demand graph until the state where all the processes have terminated, i.e., the state represented by the terminal cut set γ_T , is reached.

A *move* on a chain P_i in the demand graph model is defined to be a function whose domain is the set of all cut sets which intersect P_i in a given arc, and whose range is the set of immediate successors of these cut sets on P_i . In Fig. 9.3 a move is said to have been applied to the cut set γ on chain P_i and the cut set γ_1 is the result of this move. The move is represented by the notation $\gamma \rightarrow \gamma_1$. Similarly, the move $\gamma_1 \rightarrow \gamma_2$ is applied to the cut set γ_1 on chain P_2 . Two moves are said to be connected if they can be represented in the form $\gamma \rightarrow \gamma_1$ and $\gamma_1 \rightarrow \gamma_2$, respectively. A *macro-move* is a sequence of moves, every pair of which is connected. The sequence of moves $\gamma_0 \rightarrow \gamma_1 \rightarrow \gamma_2 \rightarrow \gamma_3$ of the demand graph in Fig. 9.3 is an example of a macro-move. A uni-chain macro-move is a macro-move all of whose components are moves on the same chain.

$C = 10$

Fig. 9.3. A Demand Graph Representing a System with One Resource

9.3.2 Feasibility and safety in the demand graph model

A cut set is said to be *feasible* if the sum of the vector demands associated with the arcs in it is no greater than the resource capacity vector $\bar{c}$. For example, the cut set, γ , is feasible if,

$$\sum_{\theta \in \gamma} \bar{d}(\theta) \leq \bar{c}$$

In Fig. 9.3 the cut set γ is feasible since,

$$\sum_{\theta \in \gamma} d(\theta) = 3 + 5 = 8 \leq c = 10$$

A feasible cut set of a demand graph is *safe* if there exists a macro-move from it to the terminal cut set of the graph and if the cut set resulting from the application of each move in the macro-move is feasible. The cut set γ in Fig. 9.3 is feasible since a macro-move, $\gamma \rightarrow \gamma_1 \rightarrow \gamma_2 \rightarrow \gamma_3 \rightarrow \gamma_4 \rightarrow \gamma_5 \rightarrow \gamma_T$, exists and each cut set in this macro-move is feasible.

In a system of processes modelled by the demand graph, a feasible cut set represents a meaningful allocation state and a safe cut set indicates that there exists a safe sequence of moves that leads from the system state represented by the cut set to the state in which all the processes have terminated. A cut set all of whose immediate successors are not feasible represents a deadlock state. It follows that if the current state of the system is not safe, then every sequence of macro-moves applied to the current cut set eventually leads to a cut set which is not feasible, i.e., no schedule exists to permit all the processes to complete their tasks.

and a deadlock condition is inevitable. The foregoing discussion suggests that the occurrence of deadlock can be prevented if an allocation scheme is devised to ensure that the current allocation state is always represented by a safe cut set. Hebalter [H8] derived an allocation scheme which was called the *safeness algorithm*.

The Safeness Algorithm, when applied to scalar demand graphs, i.e., graphs with a single resource tests the safety of a given cut set, γ , by attempting to find a macro-move from the cut set γ to the terminal cut set γ_T . This is done via a sequence of steps. First, the algorithm attempts to construct a uni-chain macro-move down a chain, P_i , from γ so that the slice resulting from each component move is feasible. The attempt is terminated at the first point where a cut set, γ_t , results that satisfies condition c1 given by:

$$d(\gamma_t \sim P_i) \leq d(\gamma \sim P_i)$$

and

$$d(\gamma_t^1 \sim P_i) \leq d(\gamma_t \sim P_i)$$

where γ_t^1 is the immediate successor of γ_t on chain P_i . If the attempt fails, then another chain is tried and so on. If the attempt fails for all the chains of the graph, then it is concluded that the cut set γ is not safe. If the attempt succeeds for any chain, then γ_t replaces γ and the procedure is repeated to test the safety of the new γ . The safeness algorithm terminates when either an attempt to find γ_t for all chains fails, meaning that the original cut set, γ , is not safe, or when the terminal cut set γ_T

is reached, meaning that the cut set, γ , is safe and in this case the algorithm is said to terminate successfully. It has been proven [H8], that the successful termination of the Safeness Algorithm when applied to a cut set γ is both necessary and sufficient for the cut set γ to be safe.

It should be noted that the Safeness Algorithm is adequate for testing the safety of a cut set in scalar demand graphs. When applied to demand graphs representing systems with multiple resources, i.e., vector demand graphs, it sometimes yields wrong results. This can be illustrated by the following example. Consider the demand graph shown in Fig. 9.4 representing a system of processes with two resources. If the Safeness Algorithm is applied to test the safety of the cut set γ , the two moves $\gamma \rightarrow \gamma_1$ and $\gamma \rightarrow \gamma_2$ will be an immediate result. All moves following these two moves are not feasible and the algorithm will conclude that the cut set γ is not safe. However, it can be shown that for this particular example, a macro-move can be found, all of whose components are feasible, leading from γ to γ_T . Such a macro-move is represented by $\gamma \rightarrow \gamma_3 \rightarrow \gamma_4 \dots \rightarrow \gamma_T$ in Fig. 9.4, indicating that the cut set γ is safe.

This obviously contradicts the result obtained by the Safeness Algorithm. The previous example suggests that avoidance of erroneous moves requires changing the condition $c1$ required by the cut set γ_t in the Safeness Algorithm to *condition c2* given by:

$$\bar{d}(\gamma_t \sim P_i) \leq \bar{d}(B \sim P_i)$$

$$\bar{d}(\gamma_t^i \sim P_i) \not\leq \bar{d}(\gamma_t \sim P_i)$$

for all cut sets β that lie between γ and γ_t inclusive.

The Safeness Algorithm with condition c2 replacing condition c1 can then be applied to test the safety of any cut set of a given vector demand graph. Unfortunately the unsuccessful termination of the modified algorithm is not sufficient to indicate that the cut set under test is unsafe. This can be illustrated by considering the demand graph in Fig. 9.5 as an example. If the *Modified Safeness Algorithm* (the Safeness Algorithm with condition c2 replacing condition c1) is applied to test the safety of the cut set γ in the demand graph of Fig. 9.5, then no move will be found that satisfies condition c2 on any of the three chains of the graph, and as a result, the algorithm concludes that the cut set γ is not safe. Consider now the uni-chain macro-move leading from the cut set γ to γ_1 on chain P_1 . The component moves of this macro-move are obviously feasible although the new cut set γ_1 does not satisfy condition c2. Following the move to γ_1 , two uni-chain macro-moves on the chain P_3 can be constructed. The two moves lead to the feasible cut sets γ_2 and γ_3 respectively. A sequence of feasible moves can be easily found leading from the cut set γ_3 to the terminal cut set γ_T . It is concluded then that the cut set γ is safe, a result that contradicts the conclusion reached by the Modified Safeness Algorithm. The example of Fig. 9.5 clearly illustrates the inadequacy of the Modified Safeness Algorithm. The reason that the sequence of moves $\gamma \rightarrow \gamma_1 \rightarrow \gamma_2 \rightarrow \gamma_3$ existed, despite the failure of the Modified Safeness Algorithm, is that the demand vector on the arc $(\gamma_1 \sim P_1)$ given by:

$$\bar{d}(\gamma_1 \sim P_1) = (1,7)$$

is sufficiently low in a crucial component, that is the component equal to 1, which enabled a macro-move down the chain P_3 to reach the terminal arc. It must be noted that the cut set γ_1 does not satisfy condition c2 and for this reason it was not detected by the Modified Safeness Algorithm. The arc $(\gamma_1 \sim P_1)$ helped the macro-move $\gamma_1 \rightarrow \gamma_2 \rightarrow \gamma_3$ to cross the arc $(\gamma_2 \sim P_3)$ which is obviously an obstacle in the way of any feasible uni-chain macro-move attempting to reach the terminal arc $(\gamma_T \sim P_3)$ on chain P_3 . For obvious reasons, an arc playing the role of the arc $(\gamma_1 \sim P_1)$ is called a *crutch arc*, and an arc playing the role of the arc $(\gamma_2 \sim P_3)$ is called a *barrier arc*. In general a crutch arc on a chain P_i with respect to a cut set γ must satisfy the following condition:

$$\bar{d}(\text{crutch arc}) \neq \bar{d}(\gamma \sim P_i)$$

A barrier arc on a chain P_i with respect to a cut set γ is the first arc on the chain P_i such that a uni-chain macro-move down the chain P_i from the cut set γ faces an infeasible move on this arc.

It is obvious now, that whenever the Modified Safeness Algorithm fails to construct sequences that satisfy condition c2, a search must be conducted to find crutch arcs that may assist in crossing barrier arcs. The search must consider all possible combinations of crutches that may be useful. This can be done by appending a special routine to the Modified Safeness Algorithm Routine (the main routine). The function of the special

Fig. 9.4. A Demand Graph Representing a System with Two Resources

Fig. 9.5. An Example to Test the Adequacy of the Safeness Algorithm

routine will be to search for crutches when called by the main routine and then return its findings. Thus the two routines interact with each other in a nested loop. When the main routine calls for the special routine, it passes to it a cut set γ and demands that a search be done to find a crutch(s) for this cut set. The special routine does this search by attempting to construct a uni-chain macro-move down one of the chains, say P_i , with feasible moves. The attempt is terminated when a cut set γ_r is found which satisfies condition c3, given by:

$$\bar{d}(\gamma_r \sim P_i) \neq \bar{d}(\gamma \sim P_i)$$

and

$$\bar{d}(\gamma_r \sim P_i) \neq \bar{d}(\gamma_r^i \sim P_i)$$

where γ_r^i is the immediate successor of γ_r on chain P_i .

The Modified Safeness Algorithm augmented by a special routine to search for crutch arcs will be called Algorithm 2. The flow charts shown in Figs. 9.6(a), (b) and (c) illustrate the details of Algorithm 2. Figure 9.6(a) shows the input-output routines which calls the main routine when it is required to test the safety of a cut set, σ . The flow charts of the main routine and the special routine are given in Figs. 9.6(b) and 9.6(c) respectively.

9.3.3 Application of the demand graph model to the distributed file problem

The demand graph model is useful when applied to Information Networks in which the sharing process of the distributed files follow a pattern more sophisticated than the one described in the maximum limit

Input-Output Routine

T $\equiv$ terminal cut set
 σ $\equiv$ input (cut set to be tested)
 μ, δ, w, x $\equiv$ internal variables
 δp $\equiv$ input when coming
 back from main
 routine

Figure 9.6(a)
Flow Chart of Algorithm 2

$\delta, w, P \equiv \text{input}$
 $P, \delta^1, w^c, x^c, \delta^+, \delta_p$
 $\equiv \text{internal variables}$

Figure 9.6(b)
Flow Chart of Algorithm 2

Special Routine
to search for crutch arco.

input $\equiv w_e, P_C, \delta^+$
internal variables
 $\equiv \delta, \delta^*, w, P$

Figure 9.6(c)

Flow Chart of Algorithm 2

(claim) model application (Section 9.2). Each user process requesting access to the information network files, must provide the management facility with information concerning the sequential phases of its activity, the files that will be accessed during each phase along with the access mode associated with each requested file during this phase. Figure 9.7 shows the demand graph of an information network with four distinct files and two active user processes. The user processes are represented by the chains P_1 and P_2 . Arcs of the two chains P_1 and P_2 represent the phase of activities. The demand vector of each arc has binary entries; a 1 entry indicates that an access to the corresponding file is requested during this phase of activity and a 0 entry indicates otherwise. Along with the demand vectors an *access mode vector* is associated with each arc of the demand graph. The access mode vector entries can be one of three possibilities, E or S or zero. An E entry indicates that the corresponding file is requested in an exclusive update mode of access and an S entry indicates shared or retrieval mode of access. Obviously, a 0 entry in the demand vector of a given arc will always be accompanied by a corresponding 0 entry in the access mode vector, indicating that an access to the file corresponding to the entry in question is not requested during this phase of activity.

A cut set representing a state of the system whose model is given in Fig. 9.7 will be feasible if the demand vectors associated with the arcs comprising it do not lead to a contradiction in the sharing of any file when these demand vectors are considered together. That is, if the entry with index i of the access mode vector of any one arc in the cut set

is an E entry, then all the i^{th} entries of the other arcs of the cut set must be zero, meaning that only one user process can have an exclusive update access to file i . Also, if the entry with index i of the access mode vector of any one arc in the cut set is an S, then all the i^{th} entries of the other arcs of the cut set must be either S or zero, meaning that more than one process can share the file in the retrieval mode of access and the file cannot be updated by any process while the system is in the state represented by this cut set. A safe cut set has the same definition given in Section 9.3.2.

In the system of processes, S_1 , represented by the demand graph shown in Fig. 9.7, the state represented by the cut-set γ is safe since a macro-move,

$$\gamma \rightarrow \gamma_1 \rightarrow \gamma_2 \rightarrow \gamma_3 \rightarrow \gamma_4 \rightarrow \gamma_5 \rightarrow \gamma_6 \rightarrow \gamma_T$$

can be found using Algorithm 2. Assume now that a new user process was activated while the system S_1 is still in the state represented by the cut set γ . The newly introduced user process is represented by the chain P_3 as shown in Fig. 9.8, and the system S_1 is transformed to a new system S_2 along with the newly introduced process represented by chain P_3 . The information on the arcs of chain P_3 is supplied to the management facility which decides whether it is safe to allow the new user process to enter the first phase of activity. This is equivalent to testing the safety of the state represented by the cut set γ^* in Fig. 9.8. Application of Algorithm 2 to cut set γ^* indicates that it represents a deadlock state

Fig. 9.7. Demand Graph of System S_1

since the only possible move is $\gamma^* \rightarrow \gamma^{**}$ as shown in Fig. 9.8, and all the immediate successors of γ^{**} are not feasible. The user process represented by the chain P_3 (the new user process) must then wait for the other two processors to complete some of their phases (of activity) until it is possible to find a feasible and safe cut set that would intersect the first arc of chain P_3 . Algorithm 2 can be used to find such a cut set. In Fig. 9.8, the cut set γ_2 is the first cut set reached that allows the new user process to enter its first phase of activity without causing a system deadlock.

9.4 Deadlock Detection and Recovery: Preemptive Scheme

The algorithms derived based on the maximum limit (chain) and demand graph models, described in the previous two sections, prevent the occurrence of deadlock by allowing the system to enter only those system states that are found "safe". In this section, a different approach is described to deal with the deadlock problems of distributed data bases in information networks. The approach does not prevent deadlock situations from occurring, but instead, it allows the system of processes to detect any deadlock and then recover by simply aborting the activity of one or more of the user processes. The aborted processes will enter the system at a later time in order to finish their uncompleted tasks.

The state of the system of processes at any time, t , can be completely described by providing for each shared file, information concerning the user processes that have control of the file (and have not yet released it); the mode of access of each process (E or S) and the process

Fig. 9.8. Demand Graph of System S_2

rank which indicates the relative position of the process on the file queue. The information given in Table 9.1(a) concerns a system which contains six information files, shared by five user processes. An information file is denoted by, F_j , ($j = 1, 2, \dots, 6$) and a user process by P_i , ($i = 1, 2, \dots, 5$). For each file, the processes requesting control are arranged in a queue, and each process is assigned a rank as follows: the first process in the queue which has control of the file in exclusive control or shared mode of an access, has a 0 rank. If a new process requests a shared mode of access to the file and the previous process in the queue was requesting also a shared mode, then the new process is assigned the same rank as the previous one. Otherwise the rank of the new process is one greater than that of the previous process. This is illustrated by the example shown in Table 9.1(a) which represents a state of a given system of processes. Three processes, P_1 , P_2 and P_5 are listed on the queue of file F_1 . Process P_1 is the first process in the file queue and thus has a 0 rank. Process P_2 is the second process and has a rank equal to 1 since it is requesting mode S (shared control) and this mode of access is different from that of process P_1 (mode E or exclusive control). For the same reason process P_5 has a rank equal to 2. The system state represented by table 9.1(a) is safe in the sense that all the processes will be able to finish their tasks and no process will be left in a permanent wait (deadlock) condition.

Assume now that while the system of processes was still in the state given by Table 9.1(a), Process P_1 issued a request for an exclusive

control access to file F_5 . The request will be placed on the queue of file F_5 and its rank will be equal to 1. The new system state is given by Table 9.1(b). It will be shown soon that the new system state is unsafe, i.e., that it is a deadlock state. Means must be provided to allow the system to detect such a deadlock and then recover from it. Before describing the detection mechanism, some notation will be introduced to facilitate the description.

The relation $P_i \uparrow F_j$ is used to indicate that "the process P_i is listed in the queue of file F_j " (the process has requested control of the file). The rank of any process, P_i , such that $P_i \uparrow F_j$ is denoted by K_{ij} . In Table 9.1(a), process P_2 is listed in the queue of file F_1 which can be written as $P_2 \uparrow F_1$ and has a rank $K_{21} = 1$. Similarly, $P_2 \uparrow F_4$, for process P_2 corresponds to a rank, $K_{24} = 0$. The wait counter, w_i , of a process, P_i , is defined as:

$$w_i = \sum_{\{j/(P_i \uparrow F_j)\}} (K_{ij}) \quad (9.1)$$

As an example, in Table 3.1(a), process P_5 is listed in the queues of files F_1 and F_3 with ranks equal to 2 and 1 respectively. Therefore,

$$w_5 = 2 + 1 = 3$$

Also define Q_w to be the set of all processes whose wait counter w_i is equal to zero, i.e.,

TABLE 9.1(a)

FILE F_j	PROCESS P_i	MODE OF ACCESS	RANK (K_{ij})
F_1	P_1	E	0
	P_2	S	1
	P_5	E	2
F_2	P_3	S	0
F_3	P_3	E	0
	P_5	S	1
F_4	P_2	S	0
	P_3	S	0
	P_4	E	1
F_5	P_4	S	0
F_6	P_2	S	0
	P_4	E	1

E indicates "update access mode (exclusive control)"

S indicates "shared (retrieval) access mode"

TABLE 9.1(b)

FILE F_j	PROCESS P_i	MODE OF ACCESS	RANK (K_{ij})
F_1	P_1	E	0
	P_2	S	1
	P_5	E	2
F_2	P_3	S	0
F_3	P_3	E	0
	P_5	S	1
F_4	P_2	S	0
	P_3	S	0
	P_4	E	1
F_5	P_4	S	0
	P_1	E	1
F_6	P_2	S	0
	P_4	E	1

E indicates "update access mode (exclusive control)"

S indicates "shared (retrieval) access mode"

$$Q_w \equiv \{P_i / w_i = 0\}$$

A process whose wait counter is equal to zero can obviously finish all its tasks since it has an access to all its requested files and the process is said to terminate safely.

The deadlock detection mechanism can be best illustrated by considering as an example, the state of the system of processes described by Table 9.1(b). This table is first transformed to the equivalent format of Table 9.2(a). Each entry in Table 9.2(a) is in the form, P_i / K_{ij} , where P_i is the user process and K_{ij} is its rank with respect to file j . The wait counter, w_i , is computed for all user processes using Table 9.2(a) and eqn. (9.1). Process P_3 is the only one whose wait counter, w_3 , is equal to zero. This is shown in Table 9.2(b) which includes two entries; namely, the user processes and their wait counter. It is clear then that the only member of the set Q_w is the process P_3 , meaning that it is the only process that can terminate its tasks safely. The rest of the processes are in a deadlock state. This is indicated by the fact that none of the wait counters of these processes can be reduced to zero, thus all of them are in a permanent waiting (deadlock) state. Process P_3 can be removed from the system and the file queues are updated (Tables 9.3(a) and (b)).

To resolve the deadlock problem, process P_1 whose request for file F_5 has caused the deadlock (Table 9.1(b)), will be forced to relinquish control of its advanced positions in the queues of its requested file (all files F_j such that $P_1 \uparrow F_j$), and join the tail ends of these queues. This

TABLE 9.2(a)
FILE TABLE

	1	2	3	4	5
F_1	$P_1/0$	$P_2/1$	$P_5/2$		
F_2	$P_3/0$				
F_3	$P_3/0$	$P_5/1$			
F_4	$P_2/0$	$P_3/0$	$P_4/1$		
F_5	$P_4/0$	$P_1/1$			
F_6	$P_2/0$	$P_4/1$			

TABLE 9.2(b)
Process table

PROCESS (P_i)	w_i
P_1	1
P_2	1
P_3	0
P_4	2
P_5	3

$$Q_w = \{P_3\}$$

TABLE 9.3(a)

	1	2	3	4	5
F_1	$P_1/0$	$P_2/1$	$P_5/2$		
F_2					
F_3	$P_5/0$				
F_4	$P_2/0$	$P_4/1$			
F_5	$P_4/0$	$(P_J/1)$			
F_6	$P_2/0$	$P_4/1$			

TABLE 9.3(b)

PROCESS (P_i)	w_i
P_1	①
P_2	1
P_3	0
P_4	2
P_5	2

$$Q_w = \{P_3\}$$

DEADLOCK CONDITION

will be followed by updating the ranks and the wait counters of the remaining processes in all the queues. The state given by Tables 9.4(a) and (b) will be reached. Again, computing the wait counters of the processes P_1, P_2, P_4 and P_5 , the wait counter w_2 of process P_2 will be equal to zero and the set Q_w now contains P_3 and P_2 . Process P_2 is removed from the system and the file queues are updated (Table 9.5(a)). Following the same procedure in the subsequent steps, the case where the set Q_w contains all the processes will be finally reached (Table 9.6(a)). This indicates that all the processes can terminate their tasks successfully.

The above example illustrates how the "detect and recover" algorithm (which will be referred to as Algorithm 3) works. This algorithm is applied to detect the existence of deadlock every time a request for file access control is issued by an active user process which has an access to or is waiting in the queues of other file(s) in the network. If a deadlock condition is detected, the process will abort its previous jobs and issue a new set of requests for the required files. Details of Algorithm 3 are given in the Flow Chart of Fig. 9.10.

When a process P_i terminates all or some of its jobs, a number of files controlled by the process will be released and the process is said to exit from the system. Following such a release, a number of steps must be executed to update the entries of the queues of the released files as well as the wait counters affected by the release. The processes whose wait counters can be reduced to zero are then assigned their requested

TABLE 9.4(a)

	1	2	3	4	5
F_1	$P_2/0$	$P_5/1$	$P_1/2$		
F_2					
F_3	$P_5/0$				
F_4	$P_2/0$	$P_4/1$			
F_5	$P_4/0$	$P_1/1$			
F_6	$P_2/0$	$P_4/1$			

TABLE 9.4(b)

PROCESS (P_i)	w_i
P_1	3
P_2	0
P_3	0
P_4	1
P_5	1

$$Q_w = \{P_3, P_2\}$$

PRE-EMPTYING PROCESS P_1

TABLE 9.5(a)

	1	2	3	4	5
F_1	$P_5/0$	$P_1/1$			
F_2					
F_3	$P_5/0$				
F_4	$P_4/0$				
F_5	$P_4/0$	$P_1/1$			
F_6	$P_4/0$				

TABLE 9.5(b)

PROCESS (P_i)	w_i
P_1	2
P_2	0
P_3	0
P_4	0
P_5	0

$$Q_w = \{P_3, P_2, P_4, P_5\}$$

TABLE 9.6(a)

	1	2	3	4	5
R_1	$P_1/0$				
R_2					
R_3					
R_4					
R_5	$P_1/0$				
R_6					

TABLE 9.6(b)

PROCESS (P_i)	w_i
P_1	0
P_2	0
P_3	0
P_4	0
P_5	0

$$Q_w = \{P_3, P_2, P_4, P_5, P_1\}.$$

files. The steps which must be executed following the release are included in the *Release and Assignment Algorithm* given below.

Release and Assignment Algorithm: This algorithm is executed following the release of one or more files by any active user process. The set S_F contains all the files released by process P_ℓ . The rank of process P_i with respect to file F_j is represented by K_{ij} and its wait counter by w_i .

DO for all released files $F_j \in S_F$:

 remove all the entries $K_{\ell j} = 0$

 DO for all processes P_i remaining on the queue of file F_j :

 If there is no i such that $K_{ij} = 0$, then,

 set $K_{ij} = K_{ij} - 1$ and,

 set $w_i = w_i - 1$

$$w_i = \sum_j K_{ij}$$

 END;

END.

DO for all processes P_j whose wait counter w_j is equal to zero:

 activate process P_j (assign to the process all its requested files).

END.

Application of Algorithm 3 to the system of processes every time a new request for file access arrives is obviously a sufficient condition to guarantee that the system will never fall or remain in a deadlock state. This is because deadlock is cured (by aborting the activity of one or more processes) as soon as it is detected. However, the preemption of an active user process (whose request was bound to cause deadlock) delays the completion of the process tasks. In addition, a cost (penalty) is incurred in some applications when it is essential for the process to repeat some of its phases of activity that were completed prior to, and then damaged by,

the preemption. Algorithm 3 can be modified to include the assignment of different levels of priorities to user processes. Given the cost (penalty) associated with the preemption of each active process, the modified algorithm will find the subset of processes whose preemption is sufficient to recover from deadlock and at the same time incurs a minimum cost. However, achieving such an optimal solution requires additional complex software and computer resources to implement and run the modified algorithm.

Graph representation of the state of a given file system can be used to detect deadlock situations. In the graph of Fig. 9.9, nodes represent processes and directed arcs represent a "wait for" relationship. For example, process P_4 is behind process P_2 in both F_4 and F_6 queues. The existence of a cycle in the graph indicates that the processes represented by the nodes forming the cycle are in a deadlock state. In Fig. 9.4, processes P_1 , P_2 , and P_4 are deadlocked since their representative nodes form a cycle.

9.5 Implementation of the Distributed Data Base Management Facility

The information processor of each node will have a DDBMF software program, perhaps as a part of the operating system, although the actual method of implementation may be different in some processors. All requests for accessing the network files, originated at a given node, are to be sent first to the local DDBMF of the node which interacts with the other DDBMF's in the network in the process of serving the file access requests. The set of DDBMF's of the nodes comprises a *virtual network*,

Figure 9.9
Deadlock State

Figure 9.10
Flow Chart of Algorithm 3

called the DDBMF network. Nodes of the DDBMF virtual network exchange messages over a set of *virtual (logical) links*, according to a certain protocol (Section 9.6). The virtual DDBMF network can be viewed as an entity whose main function is to manage, monitor and control access to all network files. The structure of each node DDBMF and its relation to other nodes of the DDBMF virtual network depend on the file access control approach used (centralized versus distributed control). This will be explained later in this section.

It is proposed that the communication process (the process of exchanging messages) between the DDBMF processes in the different nodes be based on an "outer layer" protocol in a similar way to the approach adopted in the ARPA network [C3]. To illustrate this concept, a brief description of the protocol design concepts in the ARPA network is presented in the following discussion.

The ARPA network provides a capability for geographically separated computers, called hosts, to communicate with each other. Each host computer is connected into the network through a local small computer (IMP). The complete network is formed by interconnecting these IMP's, all of which are virtually identical, through wideband communication lines. Each IMP is programmed to store and forward messages to the neighboring IMP's in the network. Specifications for the physical and logical message transfer between the host and its local IMP define the first level (HOST/IMP) protocol. This protocol is not by itself, however, sufficient to specify meaningful communications between processes running in dis-

similar hosts. Rather, the processes must have some agreement as to the method of initiating communication, the interpretation of transmitted data, etc. A *layered* approach has been adopted in the specification of the communication protocol. The inner layer is the Host/IMP protocol, the next layer, known as the Host/Host protocol specifies methods of establishing communication paths, managing buffer space at each end of a communication path and providing a method of interrupting a communication path.

Processes within a host are envisioned as communicating with the rest of the network through a *Network Control Program* (NCP), resident in the host, which implements the second level (Host/Host) protocol. The primary function of the NCP is to establish communication, break communication and control data flow over any connection between two communicating processes. The NCP's of two communicating hosts exchange information over a fixed number of *logical links* or connections [C3].

In order to accomplish its tasks, the NCP of one host must communicate with the NCP's of the other hosts. To this end a particular link between each pair of hosts has been designated as the *control link* and messages transmitted over the control link are called *control messages*.

The DDBMF virtual network can be viewed as a set of active processes, each process resident in a host computer and communicating with the other processes through the NCP resident in its host. A logical link between each two communicating hosts will be dedicated for the transmission of messages exchanged between the DDBMF processes of the two hosts. All messages received by the NCP of a given host over this link will be delivered to the DDBMF process resident in that host.

Figure 9.11 shows the components of an information processor of a network node in a store and forward computer communication network supporting a distributed data base. Local and remote users demanding access to the file system of the node create processes that run under the control of the operating system of the host computer. Once allowed an access to a given file, the user process interacts directly with the *Data Management System* (DMS) which embodies the logical and physical structures of the file system. The *Data Translator Interface* (DTI) is a program that allows the data management systems in the different host computers to communicate despite the possible dissimilarity in their logical structures. The DDBMF process controls access to the files attached to its local processor and exchanges messages with the corresponding processes in the network through the local NCP over dedicated logical links. The NCP, as mentioned previously, implements the second level (Host/Host) protocol and interfaces with the front end processor which connects the information processor into the communication subnetwork.

In the following, the concepts underlying the centralized and distributed control approaches in the implementation of the DDBMF are presented.

9.5.1 The centralized control approach

In the centralized approach, all job interference tests (deadlock, tests) and consequently all decisions are made by one node, called the control node, designated for that purpose and referred to by other DDBMF's when they receive requests for file access from their local user processes.

Figure 9.11
Details of a Network Node

Legend

FEP	Front End Processor
NCP	Network Control Program
DDBMF	Distributed Data Base Management Facility
DMS	Data Management System
FSS	File Storage System
DTI	Data Translator Interface
UP	User Process
TCP	Terminal Control Program
PCP	Printer Control Program
LP	Line Printer
UT	User Terminal

The following example is provided to illustrate the typical file access procedure in the central approach. Assume that a user process at node i is requesting an access to a file stored in node j . The request is sent first to the DDBMF of node i which relays the message to the central DDBMF. The central DDBMF has an overall picture of the global network status; aided by the deadlock detection or prevention mechanisms, it will place the request properly on the file queue. When the file becomes available, both nodes i and j will be notified to establish a connection. All the file update and retrieval processes will take place directly between nodes i and j . When the file access process is terminated, the central node is notified to update the file status tables. The same sequence is followed even if i and j happen to be the same node, i.e., if a user process is requesting a local file. The example clearly indicates that the central DDBMF has a different structure from the other network DDBMF's. The latter merely convey the requests for file access to the former which is equipped to make the decisions.

It should be noted that the central DDBMF approach is central in the sense that the decision making process concerning the file access central is done by a single node; all other nodes holding requested files have to comply with these decisions. The approach, in fact, is not related to and does not impose any restrictions on the method of routing the network traffic or the topological structure of the network.

9.5.2 The distributed control approach

In the distributed control approach, each DDBMF process will have control over access to any of the files attached to the information processor in which it is operating. To enable each individual DDBMF to make decisions concerning allocating files to local and remote requesting user processes, information must be obtained from other DDBMF processors indicating the status of their files and queue tables. The information will be used by the job interference handling models (deadlock detection or prevention) to construct a global picture of the network and thus enable the individual DDBMF process to make the correct decisions.

Assume that there are n nodes in the network, i.e., n individual DDBMF processes. Each process will transmit $(n-1)$ identical messages simultaneously, with one message addressed to each of the remaining DDBMF processes. Each message contains the most updated information about the status and queues of files at the node in question. The messages will be transmitted periodically at the onset of synchronous clock intervals. Similarly, each DDBMF process will receive periodically $(n-1)$ messages from the other processes. Now assume that a DDBMF process receives a request for access to one of the files under its control from a local or remote user process. If the request is single, i.e., the user process does not have any other present or (pending) past requests for other files, and if the request is not to update a file with multiple copies, it is honored immediately. Otherwise the DDBMF process will delay action until the next time interval, i.e., until receiving updated information about the status

of the network files from other DDBMF processes. The request is then checked against any possible interference (deadlock) and the user process is notified once a decision is made. In this scheme user processes may be assigned different levels of priority.

9.6 The Protocol Problem

It was mentioned in the previous section that the DDBMF virtual network can be viewed as a set of active processes, each process resident in a host computer and communicating with the other processes through the NCP resident in that host. The communication process between any two DDBMF processes takes place as a series of exchanges of messages; *control commands* specifying a request or an order and replies to the requests.

The set of control commands and replies specifies the *protocol* used in the communication process between the DDBMF processes resident in the host computers of the network.

Formats of the control commands are presented in this section for a case study of an information network consisting of 16 information processors (hosts) supporting 128 copies of 82 different files. Each information processor provides up to 250 local users access to the shared network files. The derivation of a suitable protocol for this information network will serve as a pedagogical method to clarify the various aspects of the problem and provide a numerical basis for the simulation conducted to evaluate the centralized and distributed approaches. The numerical simulation and results are presented in Section 9.7.

9.6.1 The centralized control case

The control command message format for the centralized control case is shown in Fig. 9.12. The control command message length is 40 bits. Each control command message begins with an 8-bit opcode. These opcodes have values 0,1,..., each value corresponds to a different command to permit table look up upon receipt of a command. A list of the control commands used in the centralized control protocol is included in Table 9.7. The remaining 32 bits of the control command message consists of the following fields (see Fig. 9.12):

- Bits 9-12 to indicate whether the request for file access is a "retrieval (shared centre) access mode" or "update (exclusive control) access mode".
- Bits 13-16 to indicate if the requesting user process has access to other files simultaneously or not and if the new request is to access one file or more than one file.
- Bits 17-32 Global user process ID - the first 4 bits represents the processor number and the remaining 12 bits represents the user number within the processor.
- Bits 33-40 Global file number - one bit is used to indicate if the file is a multi-copy or single copy file in the network.

The following example represents a typical DDBMF protocol scenario in the centralized control approach and illustrates the sequencing of commands and replies.

A user running in processor i is demanding an update mode of access to a file with two identical copies stored in processors j_1 and j_2 . Pro-

TABLE 9.7
Control Commands for the Centralized Access Control Protocol

COMMAND	SENDER	RECEIVER	COMMENTS
RFA	i	C	A REQUEST ISSUED FOR FILE ACCESS
CR	i	C	CANCEL A PREVIOUS REQUEST FOR FILE ACCESS
WFA	C	i	WAIT FILE IS NOT AVAILABLE NOW
RIC	C	i	CONNECTION IS IMPOSSIBLE TO ESTABLISH
SEC	C	j	AN ORDER TO ESTABLISH CONNECTION WITH i
ACK-SEC	j	C	READY TO ESTABLISH CONNECTION WITH i
RCP	C	i	THE REQUESTED FILE IS READILY AVAILABLE
CAS	i	j	REQUEST FOR FILE ACCESS WAS ACCEPTED "PLEASE START A CONNECTION"
ACK-CAS	j	i	READY TO START A CONNECTION
CIE	j	C	CONNECTION IS ESTABLISHED
EOA	i	j	END OF ACCESS - TERMINATE CONNECTION
CIT	j	C	CONNECTION IS TERMINATED
AC	C	i,j	ABORT A GIVEN CONNECTION
ACK-AC	i,j	C	CONNECTION IS ABORTED
RS	C	ALL PROCESSORS	REQUEST FOR INFORMATION CONCERNING STATUS
STAT	ALL PROCESSORS	C	STATUS MESSAGES

i refers to the processor in which a user process is requesting an access to a file stored in processor j and C represents the processor containing the control JDBMF.

Fig. 9.12. Control Command Message Format

cessor C contains the control DDBMF. The following exchange of messages takes place between the DDBMF's of the four processors (see Fig. 9.13):

1. Processor i sends a request for file access to processor C,

$$i \xrightarrow{\text{RFA}} C$$

2. Processor C replies by a "wait file is not available now" message (the file is found to be under the exclusive control of another process),

$$C \xrightarrow{\text{WFA}} i$$

3. When the file becomes available, processor C asks j_1 and j_2 to make the file available for processor i,

$$C \xrightarrow{\text{SEC}} j_1, j_2$$

Processors j_1 and j_2 acknowledge receipt of the last message.

$$j_1, j_2 \xrightarrow{\text{ACK-SEC}} C$$

4. Processor C informs i that the file is readily available:

$$C \xrightarrow{\text{RCP}} i$$

5. Processor i asks processors j_1 and j_2 to start a connection:

$$i \xrightarrow{\text{CAS}} j_1, j_2$$

and receives acknowledgements:

$$j_1, j_2 \xrightarrow{\text{ACK-CAS}} i$$

Figure 9.13

EXAMPLE TO ILLUSTRATE THE FILE ACCESS PROCESS
IN THE CENTRAL ACCESS CONTROL APPROACH

6. Once the connection is established, processor C is notified:

$$j_1, j_2 \xrightarrow{\text{CIE}} C$$

The access process to update the file continues until processor i notifies j_1 and j_2 that the update process is terminated:

$$i \xrightarrow{\text{EOA}} j_1, j_2$$

7. Processor C is notified of the termination in order to update its tables:

$$j_1, j_2 \xrightarrow{\text{CIT}} C$$

9.6.2 The distributed control case

Two types of command messages are exchanged between the DDBMF's in the distributed access control approach. The first type is similar to the control command messages of the centralized approach and may have the same format shown in Fig. 9.12. The second type consists of the status messages transmitted periodically at the onset of synchronous clock intervals from each DDBMF to the corresponding processes. The status messages carry information concerning the contents of the queue tables of each file at the moment of transmission. The format of the status messages is shown in Figs. 9.15(a) and 9.15(b). The information concerning each file is contained in a number of consecutive 40-bit messages. The first such message (Fig. 9.15(a)) indicates the file name, the moment the information was recorded (the clock period), the number of users on file queue and the number

of users whose requests have not been tested yet for deadlock condition. This message is followed by a number of 40-bit messages indicating the user processes ID and their rank (relative position) in the file queue. The format of the second type of message is shown in Fig. 9.15(b).

The problem of deriving the DDBMF protocols in the distributed control approach is more involved than the corresponding one in the centralized control approach. This is because the decentralization of control in the distributed approach gives rise to *critical race condition problems*. These problems will be illustrated in the following examples. Consider the 3-node network shown in Fig. 9.14(a). Two files, a and b, are accessible to users of the three processors A, B and C. File a has three copies in the network, with one copy assigned to each of the three processors. File b has one copy in the network and it is assigned to processor A. Access requests for each file copy are arranged in a queue associated with that copy. In Fig. 9.14(a), the file queues are Q_{a_1} , Q_b , Q_{a_2} and Q_{a_3} . Each queue contains a number of user processes requesting update or retrieval access. Each user process is assigned a rank indicating its relative position in the queue.

Consider the network state shown in Fig. 9.14(a). Processor A receives a request from user 1 to retrieve some records of file b. At nearly the same time processor C receives a request from user 3 to retrieve some records of file a. If each of processors A and C is unaware of the request received by the other, it will immediately place the request on the tail end of the corresponding queue. The action is taken

since it appears to be a safe one to each of the two processors, based on its local queue tables. It is obvious, however, that following these actions the network will be in a deadlock state since users 1 and 3 will permanently deadlock one another. The deadlock occurred in this case because each of processors A and C did not have enough information about the status of the other before placing the received request on the corresponding file queue.

A similar situation takes place even when the new request for file access is received by one processor only. Consider the network state shown in Fig. 9.14(b). Processor A receives a request for retrieval access of file b from user 5. Based on the contents of its own tables, processor A immediately places the request on the queue of file b (since it appears to be safe). However, if we consider the state of the queues in the other processors, we find that user 4 is waiting for user 1 on Q_{a_3} , user 1 is waiting for user 5 on Q_{a_1} , while user 5, following its new request, will be waiting for user 4 on Q_b . Thus a deadlock condition exists.

Critical race conditions in the distributed control approach are sometimes caused by the random delays associated with message delivery and transmission in message switching computer-communication networks.

The network state shown in Fig. 9.14(c) illustrates a critical race condition introduced by network-traffic delays. Inspecting the queues of the three copies of file a, it is obvious that the update requests of users 1 and 2 are not arranged in the same order relative to one another on all of the three queues. User 2 precedes user 1 on Q_{a_1} while user 1 precedes user 2 on Q_{a_2} and Q_{a_3} . This is obviously a deadlock condition.

To avoid the first two deadlock conditions, all requests for file access (received from user processors who have other requests placed on one or more queues in the network) are not answered until the processor receiving these requests gets the status messages from other processors (at the onset of the next clock interval). Only then the processor is able to test the safety of any action taken in response to the request. Thus all such requests arriving during the synchronous clock intervals are placed on a queue (on a first-come-first-serve basis) before entering the deadlock detection and recovery or deadlock prevention tests. In Figs. 9.14(a), (b) and (c) these queues are represented by Q_A , Q_B and Q_C and will be called *pre-test* queues.

At the beginning of a clock interval, each processor receives information from other processors including the contents of the file queues and the pre-test queue. The processor extracts the contents of the pre-test queues and combines them to construct a global pre-test queue which includes all the requests for file access received by all processors during the previous time interval. The file access requests on the global pre-test queue are tested for deadlock conditions and decisions are then made.

To avoid deadlock situations caused by critical race conditions, the file access requests on the global pre-test queue must be arranged in the same order in all processors. This will obviously prevent situations such as the one mentioned in the example of Fig. 9.14(c). All processors must then follow a predefined routine in constructing the global pre-test

Fig. 9.15(a)

Fig. 9.15(b). File Status Messages

queue. The resulting versions of the global pre-test queue will be identical in all processors at the beginning of every clock interval.

The distributed access control protocol is best illustrated by considering, as an example, a typical DDBMF protocol scenario. Assume that processor i contains an active user process requesting an access to some network files. Consider the following cases:

- i. *The user process is demanding a retrieval mode of access to a file with one or more copies, or an update mode of access to a file with a single copy, and no previous requests have been issued by the process.*

In this case processor i will transmit a "request for file access" message to the processor holding the file (processor j),

If the file is not available the reply of processor j will be a "wait file is not available" message:

Once the file is made available, processor j will send a "ready to establish connection" message to processor i :

- ii. *The user process is demanding an update mode of access to a file with two identical copies stored in processors j_1 and j_2 , and no previous requests have been issued by the process.*

The exchange of messages will proceed as follows:

1. Processor i issues a "request for file access" message,

$$i \xrightarrow{\text{RFA}} j_1, j_2$$

and receive an acknowledgement,

$$j_1, j_2 \xrightarrow{\text{ACK-RFA}} i$$

2. Processors j_1 and j_2 hold the request until the beginning of the next clock interval. The request for file update is first positioned on the global pre-test queue for each processor. The request is then placed on the file copy queue.
3. When the rank associated with the request becomes zero, the processor transmits a "file in question" is available to all processors holding a copy of the file,

$$j_1 \xrightarrow{\text{FQR}} j_2$$

$$j_2 \xrightarrow{\text{FQR}} j_1$$

When a processor receives an FQR command from all remaining processors holding a copy of the file, it transmits a "ready to establish connection" message to processor i ,

$$j_1 \xrightarrow{\text{RSC}} i$$

$$j_2 \xrightarrow{\text{RSC}} i$$

and the file access process starts. If processor i is tired of waiting, it may cancel its previous requests,

$$i \xrightarrow{\text{CR}} j_1, j_2$$

- iii. The user process is demanding an update or retrieval access to a file following requests issued previously by the process for accessing other files.

The exchange of messages will proceed as follows:

1. Processor i issues a "request for file access" message,

and receives an acknowledgement

2. Processor j holds the request until the beginning of the next clock interval. The request is first positioned on the global pre-test queue until a deadlock test is conducted. If it is safe, the request is placed on the tail end of the file queue. When the file rank is equal to zero on all queues, the access process starts.
3. If the deadlock danger is present and the user process is being preempted, the following message is transmitted to the requesting process:

indicating that the request for file access has been shifted to the tail end of the file queue.

The control commands used in the distributed control approach differ from those used in the centralized control approach on the following counts:

1. The RFA command must be acknowledged in the distributed control approach; this is not necessary in the centralized control approach.
2. The six control commands SEC, ACK-SEC, RCP, CAS, ACK-SEC and CIE used in the centralized control approach are replaced by the two commands RSC and ACK-RSC in the distributed control approach.

3. The two commands AC and ACK-AC used in the centralized control approach are not needed in the distributed control approach.
4. Two new commands FQR and RFP are introduced in the distributed control approaches (these are not needed in the centralized control approach).

9.7 Simulation and Results

9.7.1 Performance criteria

The centralized and distributed approaches used for the implementation of the DDBMF were both simulated using a GPDS model in a network of information processors supporting a group of data files shared by a distributed user group (see Section 9.6). Criteria used to compare the performance of the two approaches include:

CPU Time (CPU)

This includes the CPU time required to execute the DDBMF algorithms as well as the time required to update tables and file queues.

Storage Requirements (Q)

This includes the space required to store file access request queues, tables and file status queues (file access request queues are those formed before entering the deadlock testing algorithm, while file status queues refer to those formed at its exit).

Total Volume of Traffic Generated into the Network (V_T)

This includes all the messages exchanged from the moment a request is issued at a local processor until the file is allocated and

assigned to the requesting process. It does not include all query or update transactions since this will be essentially the same irrespective of the DDBMF used, and for this reason they will not be considered as basis for comparison. The total overhead traffic volume, V_T , will be called *the DDBMF virtual network overhead traffic*.

Reliability Considerations

This represents the sensitivity of the DDBMF virtual network operation (the operation of the automatic access control mechanism) to the following factors:

- failure of any of the computing processors;
- failure of the file system supported by a given processor;
- isolation of any processor from the rest of the network due to failure of the communication facilities connecting it to the network.

9.7.2 Simulation model parameters

The information network used in the simulation consists of 16 information processors and 128 shared information file copies. The actual number of distinct information files is 82, divided into four groups. The first group consists of files with single copy in the network, the second group consists of files with two copies and so on. The files are attached to the storage devices of the information processors; each information processor holds up to 8 distinct file copies and provides on-line access service. Up to 250 local users are served by each information processor.

Files located at various centres are accessible to users throughout the network, so they may be read/updated/executed from any point in the network.

The following parameters describe the simulation model:

1. N_c = number of computers supporting the distributed file system.
2. N_u = total number of network users.
3. $(PU)_i$ = percentage of the total number of users that will enter the system (start an activity), i times in a period of time equal to T hours.
4. N_a = number of new requests for file access originated in a period of time equal to Δt minutes. N_a is a random variable whose probability distribution function is assumed to be Poisson. Let $\bar{N}_a \Delta t$ be the average number of new requests originated in Δt ; then:

$$P_r[N_a = j] = \sum_{i=0}^{\infty} \frac{(\bar{N}_a \Delta t)^j}{j!} e^{-\bar{N}_a \Delta t} \delta(i-j)$$

$$P_r[N_a = 0] = e^{-\bar{N}_a \Delta t}$$

$$\bar{N}_a \Delta t = \frac{N_u \Delta t}{T \times 60} \sum_{i=0}^{\infty} i (PU)_i$$

5. N_f = total number of file copies in the network.
6. MCF = number of files with multiple copies.
SCF = number of files with a single copy.
7. $(PC)_i$ = percentage of the newly originated requests requiring access to i file copies simultaneously, ($i = 1, 2, \dots$) during a single session.

8. $(RC)_i$ = percentage of $(PC)_i$ requesting retrieval mode of access.
9. $(UC)_i$ = percentage of $(PC)_i$ requesting update mode of access.
10. $(APC)_i$ = percentage of $(PC)_i$ which will have additional requests for file access while their processes are active.
11. RLF = average number of requests accessing local files
RRF = average number of requests accessing remote files.
12. File retrieval and file update durations will be represented by the two random variables T_r and T_u respectively. The distribution functions of T_r and T_u are assumed exponential with average values $\bar{T}_r$ and $\bar{T}_u$ respectively.

$$\Pr(T_r \leq x) = Fr(x) = 1 - e^{-x/\bar{T}_r}$$

$$f_r(x) = \frac{1}{\bar{T}_r} e^{-x/\bar{T}_r}$$

$$\Pr(T_u \leq x) = Fu(x) = 1 - e^{-x/\bar{T}_u}$$

$$f_u(x) = \frac{1}{\bar{T}_u} e^{-x/\bar{T}_u}$$

13. Let $N_a(t, t + \Delta t)$ represent the newly originated requests entering the network during the interval $(t, t + \Delta t)$ and $N_s(t, t + \Delta t)$ represent the total number of originated requests entering the network during the same interval. It follows that:

$$N_s(t, t + \Delta t) = N_a(t, t + \Delta t)$$

+ requests for accessing additional files issued by processes updating or retrieving other files.

Assume that an active process which may require an access to one or more files in addition to the files under its control, will do so after a time interval $k\Delta t$ from the moment it was originally activated. The interval $k\Delta t$ is a random variable and for a fixed Δt , k is a random variable which may assume any integer value such that:

$$1 \leq k \leq M$$

where M is an upper bound for k . The relationship between $N_s(t, t + \Delta t)$ and $N_a(t, t + \Delta t)$ can be written as:

$$\begin{aligned} N_s(t, t + \Delta t) &= N_a(t, t + \Delta t) \\ &+ \sum_i N_a(t - K\Delta t, t - \Delta t(K-1)) \times (PC)_i \times (APC)_i \end{aligned}$$

14. T = total simulated period of real time.

15. ΔT_s = synchronous clock interval (in the distributed control approach)

9.7.3. The GPDS model

In the GPDS model used to simulate the processes of the DDBMF virtual network, actual entities of the information network such as user requests, files and protocol messages are represented by GPDS entities such as Transactions, Logic Switches, Gates and Queues. The deadlock detection and recovery algorithm is programmed as a FORTRAN subroutine called by the main GPDS program via the HELPF block. The FORTRAN sub-

routine share with the GPDS program a common data base. The data base consists of a number of tables whose contents are being updated constantly to reflect the current status of the file queues.

In the following, a detailed description of the components of the simulation program is provided.

Representation of File Access Requests

File access requests are represented by TRANSACTIONS generated by a single external source. The interarrival time of these TRANSACTIONS is a random variable with an exponential probability distribution function (Section 9.7.2). Each TRANSACTION entering the system has 12 parameters associated with it [G5]. The parameters are assigned values indicating:

- The source computer number
- The requesting user process number
- The number of requested files
- The requested file(s) identification numbers (file names)
- The mode of file access (update or retrieval).

The assignment of TRANSACTION parameter values are based on the values of computational entities (FUNCTIONS and VARIABLES). These FUNCTIONS and VARIABLES are GPDS entities representing the simulation parameters stated in Section 9.7.2. The TRANSACTIONS are processed by the system following the assignment of their parameters.

Representation of Files

Files are represented by LOGIC SWITCH entities and GATE blocks in the simulation model. Each file corresponds to two LOGIC SWITCHES and one GATE Block. The file retrieval and update durations are simulated using the ADVANCE block and exponential distribution FUNCTION entities.

TRANSACTIONS representing access requests to a given file arrive at the queue associated with the GATE corresponding to that file. The GATE block allows the first arriving TRANSACTION to pass to the ADVANCE block. If the TRANSACTION is updating the file, the GATE block prevents other TRANSACTIONS from entering until the update process is terminated. If the TRANSACTION is in retrieval mode of access, the GATE block allows the next TRANSACTION to enter provided that it is requesting a retrieval mode of access, otherwise it is blocked. To implement and control the access process when two modes of access exist, two LOGIC SWITCHES are used with each GATE (representing a file). Each LOGIC SWITCH can be in one of two states: set state or reset state. A GATE may block the arriving TRANSACTION or pass it to the ADVANCE block depending on the STATUS of the LOGIC SWITCHES. To illustrate the steps taken when a TRANSACTION arrives at a GATE block representing a file, assume that G_j represents the GATE, $(LS)_{j,1}$ and $(LS)_{j,2}$ represent the LOGIC SWITCHES corresponding to file j . Also assume that n_j represents the number of TRANSACTIONS sharing access to file j (retrieval mode of access).

TRANSACTION Arriving at GATE G_j Requesting Retrieval Mode of Access

1. Test if $(LS)_{j,1}$ is set, if so, wait in the queue of G_j
2. If $(LS)_{j,1}$ is reset, increment n_j by 1 and set $(LS)_{j,2}$
3. Advance until the retrieval process is terminated
4. Decrement n_j by 1. If n_j is not zero, exit
5. If n_j is zero, reset $(LS)_{j,2}$ and exit.

TRANSACTION arriving at GATE G_j requesting update mode of access

1. Test if $(LS)_{j,1}$ is set, if it is set, wait in the queue of G_j
2. If $(LS)_{j,1}$ is reset, enter and set $(LS)_{j,1}$
3. Test if $(LS)_{j,2}$ is set, if so, wait until it is reset
4. If $(LS)_{j,2}$ is reset, pass to the next block and set $(LS)_{j,2}$
5. Advance until the update process is terminated
6. Reset $(LS)_{j,1}$ and $(LS)_{j,2}$ and exit.

Representation of Protocol Messages

The DDBMF protocols described in Section 9.6 are simulated in the GPDS model in order to calculate the virtual network overhead traffic.

In actual network environment these protocols specify the type and order of the messages exchanged between the network processors in the process of allocating files to user processes. However in the simulation model, these messages are not physically generated in the same way the TRANSACTIONS representing the file access requests are generated. Instead, counters are provided in the different sections of the simulation programs to keep track of the type and number of messages that are exchanged between the

different processors. In the centralized approach, one counter is needed to monitor the command control messages. In the distributed approach, two counters are needed; the first counter monitors the control command messages and the second counter monitors the files status messages. At the end of the simulation run the counters report the total number of DDBMF overhead protocol messages.

To illustrate the way the number of exchanged protocol messages is monitored, in a given situation, consider the following example. In the simulation program of the centralized approach, once a TRANSACTION is generated and its parameters are identified, the message counter is incremented by 1 to account for an RFA message. If the TRANSACTION arrives at the GATE representing the requested file and finds the file available, the message counter is incremented by 6 to account for the SEC, ACK-SEC, RCP, CAS, ACK-CAS and CIE messages (see Section 9.6).

When the TRANSACTION exits, marking the end of the file access, the message counter is incremented by 2 to account for EOA and CIT messages.

The flow charts of Figs. 9.16 and 9.17 illustrate the DDBMF protocol messages generated in the centralized and distributed control approaches respectively.

The Interface with Deadlock Detection Algorithm

The deadlock detection and recovery algorithm (Section 9.4) is implemented in the simulation program. Theoretically, the algorithm can be programmed using the GPDS language statements. However, the result will be a cumbersome and lengthy routine. Execution of this routine, each

Fig. 9.16. Network Messages Generated by the Centralized Control Management Facility.

Fig. 9.17. Network Messages Generated by the Distributed Control Management Facility

time a deadlock situation is suspected will consume a considerable amount of CPU time. Instead, the algorithm is implemented in FORTRAN as a subroutine called by the main GPDS program using the HELPF block (Fig. 9.18). The FORTRAN subroutine has access to the most updated version of the file queue table each time it is called by the main GPDS program to inspect the safety of a transition to a suspected state. The FORTRAN subroutine is executed and the results are returned to the main GPDS program.

9.7.4 Simulation values

$N_c = 16$ computers, $N_u = 4000$ users or 250 users/computer system.

$(PU)_i$ is given in the following table:

i	0	1	2	3	4
$(PU)_i$	15%	30%	25%	20%	10%

$\Delta t = 1$ minute, $T = 6$ hours, $\Delta T_s = .1$ minute

$$\bar{N}_a \Delta t = 4000 \sum_{i=0}^4 i(PU)_i / 6 \times 60$$

$= 20$ requests,

$\bar{N}_a = 20$ requests/minute

$N_f = 128$ file copies. These are distributed as follows:

50 files each with a single copy in the network
 20 files each with 2 copies in the network
 10 files each with 3 copies in the network
 2 files each with 4 copies in the network

Fig. 9.18. Components of the Simulation Model

$$\text{MCF/SCF} = \frac{32}{50} = 0.64$$

The values of $(\text{PC})_i$, $(\text{RC})_i$, $(\text{UC})_i$ and $(\text{APC})_i$ are given in Table 9.8

TABLE 9.8

File request modes

i	1	2	3	4
$(\text{PC})_i$	60%	25%	10%	5%
$(\text{RC})_i$	70%	60%	40%	10%
$(\text{UC})_i$	30%	40%	60%	90%
$(\text{APC})_i$	10%	5%	2%	0%

The ratio RLF/RRF is considered a variable that will assume a fixed value during each run of the simulation.

$$\bar{T}_r = 2 \text{ minutes}, \quad \bar{T}_\mu = 3 \text{ minutes}$$

Also assume that an active process which may require an access to files in addition to ~~the~~ files under its control, will do so after a time interval $k\Delta t$ from the moment it was originally activated. The interval $k\Delta t$ is a random variable which is assumed to have a uniform probability distribution function - the variable K is an integer bounded by the relation:

$$1 \leq K \leq M$$

M is an upper bound for K . In the simulation runs, $M = 10$.

4

OF/DE

4

9.7.5 Simulation results and analysis

A number of simulation runs of the GPDS models were carried out for a simulated real time of six hours per run with the deadlock detection and recovery algorithm (Section 9.4) employed in both models of the DDBMF (centralized and distributed models).

In one set of simulation runs, the ratio RLF/RRF was the only varying parameter with a different value for each run; all other parameters were held constant at the values reported in Section 9.7.4. Performance criteria measures, CPU, Q and V_T were obtained for each simulation run. Graphs representing the values of CPU, Q and V_T as functions of the ratio RLF/RRF for the different simulation runs are shown in Figs. 9.19, 9.20 and 9.21, respectively.

Figure 9.19 indicates that the total CPU time required to execute the various algorithms of the DDBMF virtual network varies with the ratio RLF/RRF for both distributed and centralized implementation techniques. This result is explained by the discussion of Section 9.6 in which it was indicated that allocation of remote files requires more processing by the DDBMF virtual network than does allocation of local files. Also it is noted that the total CPU time required in the centralized approach is considerably less than that required in the distributed case. This result is attributed to the duplication of effort (the execution of similar algorithms simultaneously in the nodes of the DDBMF virtual network) in the case of the distributed control approach, while the total effort is concentrated in one mode in the centralized case.

The results of Fig. 9.19 also indicate that the required CPU time in the distributed control case decreases faster than that in the centralized case as the ratio RLF/RRF decreases. This is because accessing local files in the distributed control approach does not normally require the exchange of protocol messages and consequently eliminates some execution time. Thus increasing the percentage of requests for accessing local files in the distributed control approach leads to a reduction in the CPU time overhead.

A similar argument can be applied to the average total storage, Q , required by the DDBMF virtual network in both cases (Fig. 9.20). In the distributed approach, identical copies of the file status and queue tables are stored in all nodes of the DDBMF virtual network, leading to the need for a storage space substantially larger than that required in the centralized approach where only one node stores these tables. It is also noted that the average total storage, Q , remains nearly constant as the ratio RLF/RRF increases in the graphs of Fig. 9.20. This is because the total required storage of the DDBMF virtual network depends primarily on the total network file access request rate which was held constant during the simulation runs of Fig. 9.20.

Figure 9.21 shows the effect of the ratio RLF/RRF on the DDBMF virtual network overhead traffic. This traffic when added to the retrieval, update transactions, and other overhead traffic (routing, flow control, etc.) form the total communications subnetwork traffic. In the distributed control case, the DDBMF overhead traffic decreases considerably as the

ratio RLF/RRF increases. This is because most of the requests for accessing local files do not generate network overhead traffic. In the centralized case, the change in the total overhead traffic, V_T , with the ratio RLF/RRF is far less noticeable. This is because all requests for file access, including the request for local files, are referred to the central node. The two curves meet at a crossover point (X) corresponding to a ratio, RLF/RRF, of 1.34 in this example (Fig. 9.21).

The ratio RLF/RRF represents an important criterion for comparing the two approaches, particularly for the case where the network average delay in secs/message is considered. Figure 9.22 shows a typical delay-traffic curve of a store and forward computer communication network [K3], for a given set of channel capacities and traffic routing procedure. The average delay per message increases slightly with the increase in the total input traffic until the input traffic approaches the saturation point, i.e., approaches the total channel capacity. At this point the delay increases rapidly with any slight increase in the network input traffic. If the network supporting the distributed file system is operating with an input traffic volume, A_T (Fig. 9.22), then the introduction of a DDBMF overhead traffic, ΔV , would have no or little effect on the average delay per message. In this case the deciding factors in selecting between the two approaches will be the CPU time overhead and storage requirements rather than the DDBMF overhead traffic. In cases where the network is operating near the saturation region (point B_T in Fig. 9.22), the introduction of an overhead traffic ΔV , will seriously affect the average delay

per message. In the latter case, it is essential to select the DDBMF approach that introduces a smaller overhead traffic depending on whether the ratio RLF/RRF is greater or smaller than the crossover point (Fig. 9.21).

Sensitivity of Network Performance to Parameter Variations

Another set of simulation runs were carried out to study the effect of varying the value of some network parameters (such as average update and retrieval durations) on the performance of the network. The results are presented in Figs. 9.23, 9.24, 9.25 and 9.26. In the simulation runs represented by each of these figures, the value of one or more parameters was varied while all other parameters were held constant at the values reported in Section 9.7.4.

The effect of varying the ratio MCF/SCF on the average number of requests in file queues (requests that are being processed as well as those waiting to acquire file access control) is shown in Fig. 9.23 for various values of update/retrieval request ratios, R_u , where R_u is given by:

$$R_u = \frac{\sum_i (PC)_i (UC)_i}{1 - \sum_i (PC)_i (UC)_i}$$

The results of Fig. 9.23 shows that the average number of file requests waiting in file queues increases as the ratio MCF/SCF increases. The increase is larger for higher values of R_u . This result is obvious since an increase in the value of the ratio MCF/SCF coupled with a higher percentage

of requests seeking update file access leads to an increase in the percentage of exclusive control cases. The resultant effect would be an increase in the size of the queue of each file.

The average waiting time as a function of the two parameters, R_u and MCF/SCF is shown in Fig. 9.24. The variation of the average waiting time naturally follows the same trend as the number of requests waiting in file queues. Given the average size of file queues (results of Fig. 9.23), the average waiting time can be also calculated using Little's formula. The formula states that the average number of requests (in queue and service) is equal to the average arrival rate of new requests multiplied by the average time spent by a request (in queue and service). The dotted curves of Fig. 9.24 represent the waiting times calculated using Little's formula.

The number of occurrences of deadlock is affected by many factors, including user traffic, retrieval and update durations and the relative mix of update and retrieval requests. Of particular importance is the occurrence of additional file requests by processes which currently access several files. For instance in Table 9.8, the values of (ACP)_i show that 2% of processes currently accessing three files will generate random requests for an additional file. For the values of (APC)_i given by Table 9.8, simulation of six hours operation resulted in 12 deadlock states (with recovery). Multiplication of the elements of (APC)_i by a factor F affected the number of observed deadlock states, n, as shown below:

F	1	2	3	4	5
n	12	23	39	62	105

It should be noted that the simulation is carried under the assumption that for each newly originated request, the probability of accessing any network file is uniform. This assumption has the effect of reducing the number of deadlock occurrences during a simulation run. A relaxation of this assumption (i.e., concentrating the demand for file access on a subset of files) would certainly lead to an increase in the number of deadlock states during a simulation run.

The effect of increasing the average rate of request arrivals, N_a , on the size of file queues is demonstrated in Fig. 9.25 for different average retrieval and update durations $(\bar{T}_r + \bar{T}_u)/2$. Obviously if the network is offering real time file access service, then larger values of N_a and $(\bar{T}_r + \bar{T}_u)/2$ should be avoided since they correspond to excessive delays (waiting times) in file queues.

Figure 9.26 shows the effect of varying the ratio $\bar{T}_r/\bar{T}_u$ on the average waiting time for requests in file queues, corresponding to four different values of the ratio MCF/SCF (ratio of multiple copy files to single copy files in the network). The results of Fig. 9.26 indicate that increasing file retrieval durations (while decreasing file update durations) will allow more users to share simultaneous access to the file. This reduces the size of the queue of requests waiting for file access. As a result, the average waiting time (delays) encountered by file access requests will be reduced.

Figure 9.19
Processing Time vs. R_s

Figure 9.20

Average Storage Requirements vs. R_s .

Figure 9.21
Overhead Traffic vs. R_s

Figure 9.22
Network Traffic-Delay Characteristic

average number of requests in file queue

Figure 9.23

Average number of requests in file queue vs. the ratio MCF/SCF for different update/retrieval ratios

Figure 9.24

Average waiting time (in minutes) of requests in file queue vs. the ratio MCF/SCF for different update/retrieval ratios.

- average time calculated by simulation
- - - average time calculated using Little's formula

average number of requests in file queue

Figure 9.25

average number of requests in file queue at the
 average number of new requests originally in the file
 at time Δt , for different update and retrieval durations

$$\left(\frac{T_r + T_u}{2} \right)$$

Figure 9.26

Average waiting time (in minutes) of requests vs. the retrieval/update duration ratio for different MCF/SCF values.

9.7.6 Reliability considerations

An important factor enters into consideration when system reliability is introduced as a performance criterion. In this case the distributed approach appears to be superior since it is relatively insensitive to the failure of any network node. In a centralized system failure or isolation of the designated control node will paralyze the operation of the whole system. To ensure the fail-soft operation of the centralized system and thus improve its reliability, safety measures must be employed. As one possible measure, it is proposed that one or more nodes in the network will be designated to act as back-up nodes for the central node.

The central DDBMF software package will be stored in the auxiliary storage devices of the processors of these nodes. In the case of a central node failure, the software system is loaded into the main storage of one of these nodes, which will be the *acting central node* and the other nodes are immediately notified (perhaps by the acting central node itself) of the new change. The network nodes will transmit to the acting central node, information messages concerning the status of the files attached to their processors at the moment the failure of the original central node occurred. Using this information, the acting central node will then be able to construct a global picture of the network files status and queue tables and proceed to function as the new control node.

CHAPTER 10

CONCLUSIONS AND SUGGESTIONS FOR FURTHER RESEARCH

10.1 Thesis Contributions

The major contributions of this research are in the following two areas:

1. The design of information networks supporting distributed data bases (the resource allocation problem).
2. The software controlled access of distributed data bases.

1. *The Design Problem*

A model for the allocation of resources in distributed information networks was formulated. The model allows for the simultaneous allocation of files (to network nodes) and communication capacities (to network links). It was shown (Chapter 4) that the model is in fact a generalization of previously developed resource allocation models and differs from them in several aspects. The main differences are:

- The introduction of the file availability constraint (this was not considered in previous models).
- The ability to handle arbitrary and general topologies (previous models were designed for specific topologies, e.g., fully connected topologies, tree topologies, etc.).
- The representation of more realistic communication and storage cost structures.

The resource allocation model developed in this study results in an optimization problem in which a combined storage and communication cost

function is minimized subject to message delay and file availability constraints. This problem falls into a class of nonlinear integer programming problems which can be solved by a nonlinear branch-and-bound integer programming technique (Chapter 5). The solutions, while theoretically convergent, are computationally cumbersome even for small size problems.

A new algorithm which yields relatively low (but not minimum) cost solutions quite rapidly has been developed (Chapter 6). Numerical results for small and moderate size problems indicate that typical savings of one order of magnitude in computer processing time are possible. Moreover, storage requirements are significantly reduced with respect to the integer programming algorithm. The new algorithm is also useful in obtaining low cost solutions for large size problems where excessive storage requirements prohibit the use of classical integer programming techniques even with the largest computers available to date.

The resource allocation model formulated in this study and the new algorithm developed for its solution should thus allow the practical application of a more comprehensive design approach than has previously been possible.

Complementary to the major contribution cited above, there are two others which can be summarized as follows:

- The time dependent terminal reliability problem was studied for general network topologies (Chapter 2). An expression for the failure rate function between a given pair of nodes was derived for the case where the failure probability distribution functions of all network elements are Weibull. It was shown that for this case, the terminal failure rate function is also Weibull and that it depends on the failure rates of the elements of the minimum cut sets separating the two nodes.
- A queueing model with infinite waiting room, Poisson arrivals and general distribution of message lengths (M/G/1) queue was studied under a head-of-the-line priority service discipline with two priority levels. Using this model, an expression was obtained for the stationary joint probability distribution of the number of high and low priority packets left behind in a front-end-processor by a departing packet. It was shown that the use of a head-of-the-line priority service discipline based on the packet's length, when implemented in a store-and-forward network, reduces the average packet delay (Chapter 3). The average packet delay was shown to decrease as the utilization of the communication lines increases.

2. *The area of software controlled access of distributed data bases*

Two problems were addressed in this dissertation. These are the job interference (deadlock) problem, and the file access control problem.

Three general abstract models for deadlock detection and recovery and deadlock prevention were introduced (Chapter 9). The models were applied to the distributed file access problem and three algorithms were derived. Any of these algorithms can be used to construct the logical functions of file access control systems.

Two approaches to the design and implementation of a software controlled access system were presented in Chapter 9. In the first approach, information concerning file access status and active user processes is exchanged between processing nodes so that the control is *distributed*. In the second approach control resides in a designated node, which is referred to by all active user processes when requests for file access are initiated; control is thus *centralized*. Protocol specifications were derived for each of the distributed and centralized control approaches.

A simulation of the two approaches was carried out on a network of information processors supporting a group of data files shared by a distributed user group. The two approaches were compared under the criteria of computer processing time, storage overhead and network traffic overhead required for implementation. Simulation results indicate that centralized control of file access requires less CPU time and less storage

than does distributed control. As might be expected, however, distributed control gives rise to much less overhead traffic when most files requested are locally stored. The simulation results also demonstrate that the ratio of local to remote access requests is a significant parameter which may be used in the choice of a control scheme.

While centralized control is often efficient, it is not robust, since failure of the central node causes system paralysis. A modified controller is proposed (Section 9.7.7), in which the centre of control can shift its location to adapt to failure of network elements. It is surmised that such an adaptive controller will prove superior in performance to either of the previous alternatives.

10.2 Suggestions for Further Research

This research has uncovered several problems which require further study before distributed data bases can be widely used. Among these problems are:

1. The problem of allocating resources in distributed information networks when some of the variables which were frozen in the model of this study (such as network topology and message routing strategy) are introduced as design variables. Based on the computational experience reported here as well as the theoretical results obtained by Eswaran [E1], it is recommended that, in future attempts to solve more general design problems, emphasis should be placed on developing good heuristic approaches rather than on refining existing deterministic (exact) approaches.

2. The problem of interfacing data bases that have dissimilar logical structures. This is the problem of designing the *"Data Translator Interface"* explained in Section 9.5.
3. The problem of developing efficient tools (analysis, simulation and measurement) to compare the cost-performance of the methods used to handle job interference (deadlock detection and recovery, and deadlock prevention) in various file access applications.

REFERENCES

- [A1] Agin, N., *Optimum seeking with branch and bound*, Management Science, Vol. 13, No. 4, December 1966, pp. 8-176-185.
- [B1] Becker, H.B., *Functional Analysis of Information Networks*, John Wiley, New York, 1973.
- [B2] Banerjee, S.K. and K. Rajamani, *Parametric representation of probability in two dimensions - A new approach in system probability evaluation*, IEEE Trans. on Reliability, Vol. R21, pp. 56-60, February 1972.
- [B3] Burtin, Y. and G. Pittel, *Asymptotic estimates of the reliability of a complex system*, Engineering Cybernetics, Vol. 10, No. 3, May-June 1972.
- [B4] Beltrami, E.J. and L.D. Bodin, *Networks and Vehicle Routing for Municipal Waste Collection*, Networks, 4:65-94, 1974.
- [B5] Boehm, B.W. and R.L. Mobley, *Adaptive Routing Techniques for Distributed Communication Systems*, Rand Corporation Memorandum, RM-4781-PR, 1966.
- [B6] Burke, P.J., *The output of a queueing system*, Operations Research, Vol. 4, pp. 699-704, 1956.
- [B7] Basket, F. and R. Muntz, *Queueing network models with different classes of customers*, IEEE Computer Conference Proceedings, pp. 205-209, San Francisco, California, September 1972.
- [B8] Bales, E., *An additive algorithm for solving linear programs with zero-one variables*, Operations Research, Vol. 13, No. 4 (July - August 1965), pp. 517-516.
- [B9] Baliniski, M.L., *Integer programming: Methods, uses, computations*, Management Science, Vol. 12, No. 3 (November 1965), pp. 253-313.
- [B10] Bowman, J. and G. Nemhauser, *Deep Cuts in Integer Programming*, Technical Report No. 8 (February 1968), Dept. of Statistics, Oregon State University, Corvallis, Oregon.
- [B11] Booth, G.M., *The use of distributed data bases in information networks*, Proceedings of the First Computer Communication Conference, Washington, D.C., pp. 371-376, October 1972.
- [B12] Barker, D.L., *The European computer network project*, The First International Conference on Computer Communications, pp. 192-200, ACM/IEEE, October 1972.
- [C1] Chen, P.P.S., *Optimal file allocation in multi-level storage systems*, Proceedings of the National Computer Conference, pp. 277-282, 1973.

- [C2] Chu, W.W., *Optimal file allocation in a multiple computer system*, IEEE Trans. on Computers, Vol. C-18, No. 10, October 1969.
- [C3] Carr, C.S., S.D. Crocker and V.G. Cerf, *Host-host communications protocol in the ARPA network*, AFIPS Conference Proceedings, 36: 589-597, SJCC, 1970.
- [C4] Crocker, S.D., J.F. Heafner, R.M. Metcalfe and J.B. Postel, *Function-oriented protocols in the ARPA computer network*, AFIPS Conference Proceedings, 40:271-279, SJCC, 1972.
- [C5] Cowan, D.D., N.D. Brewer, C.C. Gotlieb and L.F. Macrae, *A Preliminary Study of the Methods of Using Computer Networks and of the Applications which Might be Available Over a Computer Network*, Report for the CANUNET Committee, March 1972.
- [C6] Casey, R.G., *Allocation of copies of files in an information network*, Proceedings of the Spring Joint Computer Conference, May 1972.
- [D1] Dixon, R.D., *Computer networking - the giant step in data communications*, Data Communication Systems, pp. 25-31, September 1973.
- [D2] Demercado, J.B. and N. Spyratos, *A Capacity Allocation Problem in Distributed Networks*, To appear in Networks, 1974.
- [D3] Demercado, J.B., S. Mahmoud and N. Spyratos, *A summary of methods for the reliability modelling of computer communication networks*, IEEE Trans. Rel., to be published.
- [D4] Demercado, J.B., N. Spyratos and B.A. Bowen, *A method for the exact calculation of network reliability*, IEEE Trans. Rel., to be published.
- [D5] Demercado, J.B., *Reliability prediction studies of complex systems having many failed states*, IEEE Trans. Rel., Vol. R-20, pp. 223-230, November 1971.
- [D6] Davies, D.W., *The control of congestion in packet switching networks*, Proceedings of the Second AMC IEEE Symposium in the Optimization of Data Communication Systems, Palo Alto, California, October 1971.
- [D7] DeSilva, J.S., *Dynamic Assignment of Priorities in a Computer Network*, Report of the Communications Planning Branch, Department of Communications, Ottawa, Canada, June 1973.
- [E1] Eswaran, K.P., *Placement of records in a file and file allocation in a computer network*, IFIP Conference Proceedings, pp. 304-307, Stockholm, Sweden, August 1974.
- [F1] Frank, H., R.E. Kahn and L. Kleinrock, *Computer communication network design experience with theory and practice*, AFIPS Conference Proceedings, 40:225-270, SJCC, 1972.

- [F2] Farber, D., *Data Ring Oriented Computer Networks*, Computer Networks, edited by R. Rustin, Prentice-Hall, Englewood Cliffs, N.J., 1972.
- [F3] Fuchs, E. and P.E. Jackson, *Estimates of distributions of random variables for certain computer communications traffic models*, Communications of the ACM, Vol. 13, No. 12, pp. 752-757, December 1970.
- [F4] Fox, B., *Discrete optimization via marginal analysis*, Management Science, 13: 210-216, November 1966.
- [F5] Fultz, G.L., *Adaptive Routing Techniques for Message Switching Computer-Communication Networks*, Computer Science Department, UCLA, ENG 7252, July 1972.
- [G1] Gerla, M., *The Design of Store and Forward Networks for Computer Communications*, UCLA Technical Report UCLA-ENG-F319, Computer Science Department, University of California, Los Angeles, January 1972.
- [G2] Geoffrion, A.M. and R.E. Marsten, *Integer Programming Algorithms: A Framework and State of the Art Survey*, Third Operations Research Symposium, Montreal, Quebec, 24-26 August 1970.
- [G3] Geoffrion, A.M., *Elements of large scale mathematical programming*, Management Science, Vol. 16, No. 11 (July 1970), pp. 652-691.
- [G4] Geoffrion, A.M., *Generalized Benders Decomposition*, Working Paper No. 159 (April 1970), Western Management Science Institute, UCLA.
- [G5] Gomory, *Some polyhedra related to combinatorial problems*, Journal of Linear Algebra and Applications, Vol. 2, No. 4 (October 1969), pp. 451-558.
- [G6] Greenberg, S., *GPSS Primer*, John Wiley, New York, 1972.
- [H1] Hansler, E., G.K. McAuliffe and R.S. Wilkov, *Exact Calculation of Computer Network Reliability*, Fall Joint Computer Conference, 1972.
- [H2] Hansler, E., *A fast recursive algorithm to calculate the reliability of a communication network*, IEEE Trans. on Communications, Vol. COM-20, No. 3, pp. 637-640, June 1972.
- [H3] Hsu, J.C. and A.U. Mayer, *Modern Control Principles and Application*, McGraw-Hill, 1968.

- [H4] Hansen, P., *Nonlinear 0-1 Programming by Implicit Enumeration*, paper presented at the 7th Mathematical Programming Symposium, The Hague, September 1971.
- [H5] Hammer, P., *A B-B-B Method for Linear and Nonlinear Bivalent Programming*, Israel Institute of Technology, Operations Research, Statistics and Economics, Memoograph Series No. 48.
- [H6] Hammer, P. and S. Rudeanu, *Boolean Methods in Operation Research and Related Areas*, Springer-Verley, New York, 1968.
- [H7] Habermann, A.N., *Prevention of system deadlocks*, Communications of the ACM, Vol. 12, No. 7, pp. 373-377, 1969.
- [H8] Hebalkar, Prakash, *Deadlock-free Resource Sharing in Asynchronous Systems*, (Ph.D. Dissertation), Electrical Engineering Department, MIT, Mass., September 1970.
- [H9] Holt, R., *Some deadlock properties of computer systems*, Computing Surveys, Vol. 4, No. 3, September 1973.
- [K1] Karp, D. and S. Seroussi, *A communications interface for computer networks*, Second Symposium on Problems in the Optimization of Data Communications Systems, ACM/IEEE, pp. 117-123, October 1971.
- [K2] Krone, M.J., *Heuristic Programming Applied to Scheduling Problems*, Doctoral dissertation, Princeton University, 1970.
- [K3] Kleinrock, L., *Stochastic Message Flow and Delay*, Dover Publications, 1972.
- [K4] Kahn, R.E. and W.R. Crowther, *A Study of the ARPA Computer Network Design and Performance*, Bolt, Beranek and Newman, Inc., Cambridge, Mass., Report No. 2161, August 1971.
- [K5] Karp, R.M., *Reducibility Among Combinatorial Problems, Complexity of Computations*, Plenum Press, 1972.
- [L1] Lowe, T.C., *The influence of data base characteristics and usage on direct access file organization*, JACM, Vol. 15, pp. 535-548, October 1968.
- [L2] Lenke, C.E. and K. Spielberg, *Direct search algorithms for zero-one and mixed-integer programming*, Operations Research, Vol. 15, No. 5 (September-October 1967), pp. 892-914.
- [L3] Lawler, E. and M. Bell, *A Method for solving discrete optimization problems*, Operations Research 15, 1098-1112 (1967).

- [M1] Mahmoud, S.A. and J.S. DaSilva, *The output of a priority service queue and its applications to computer networks*, Proceedings of the Second Computer Conference, 44: 1-6, Austin, Texas, November 1973.
- [M2] Mao, J.C. and B.A. Wallingford, *An Extension of Lawler and Bell's Method of Discrete Optimization with Examples from Capital Budgeting*, Management Science 15, Series B, 51-60 (1968).
- [M3] Murphy, J.E., *Resource allocation with interlock detection in a multi-task system*, Fall Joint Computer Conference Proceedings, Vol. 33, pp. 1169-1176, 1968.
- [M4] Merten, E., *Lugriffs synchronisation bei mehrfach geführten dateien in rechnernetzen*, Diplomarbeit, Universität Karlsruhe, March 1973 (German).
- [P1] Postel, J.B., *A Graph Model Analysis of Computer Communications Protocols*, Computer Science Department, UCLA, UCLA-ENG-7410, January 1974.
- [P2] Prosser, R.T., *Routing procedures in communication networks - Part I: Random procedures*, IRE Transactions on Communication Systems, CS-10: 322-329, 1962.
- [P3] Pierce, J.F., *Application of combinatorial programming to a class of all-zero-one integer programming problems*, Management Science, Vol. 15, No. 3 (November 1968), pp. 191-209.
- [R1] Ramamoorthy, C.V. and K.M. Chandy, *Optimization of Memory Hierarchies in Multiprogrammed Systems*, JAGM, July 1970.
- [R2] Roberts, L.G. and B.D. Wessks, *Computer network development to achieve resource sharing*, AFIPS Conference Proceedings, 36: 543-599, SJCC, 1970.
- [S1] Severance, D.G., *Some Generalized Modeling Structures for Use in Design of File Organization*, Ph.D. Dissertation, Department of Computer and Communication Sciences, University of Michigan, 1972.
- [S2] Shapiro, J.F., *Dynamic programming algorithms for the integer programming problem*, Operations Research, Vol. 16, No. 1, (January-February 1962), pp. 103-121.
- [S3] Shashani, A. and E.G. Coffman, *Detection, Prevention and Recovery from Deadlock in Multiprogrammed Multiple Resource Systems*, Technical Report No. 80, Department of Electrical Engineering, Princeton University, October 1969.

- [S4] Scantlebury, R.A. and P.T. Wilkinson, *The design of a switching system to allow remote access to computer services by other computers and terminal devices*, Second Symposium on Problems in the Optimization of Data Communications Systems, ACM/IEEE, pp. 160-167, October 1971.
- [T1] Taha, H.A., *A Balasian-Based Algorithm for 0-1 Polynomial Programming*, Research Report No. 70-2, University of Arkansas, May 1970.
- [W1] Whitney, V.K.M., *A Study of Optimal File Assignment and Communication Network Configuration in Remote-Access Computer Message Processing and Communications Systems*, University of Michigan SEL Technical Report No. 48, September 1970.
- [W2] Walden, D.C., *A system for interprocess communication in a resource sharing computer network*, *Communications of the ACM*, Vol. 15, No. 4, pp. 221-230, April 1972.
- [W3] Wilkov, R.S., *Reliability Considerations in Computer Network Design*, Proceedings of the IFIP Congress, Ljubljana, Yugoslavia, August 1971.
- [W4] Walters, L.J., *Reduction of integer polynomial programming problems to zero-one linear programming problems*, *Operations Research*, Vol. 15, pp. 1171-1174, December 1967.
- [Y1] Young, R.D., *New Cuts for a Special Class of 0-1 Integer Programs*, Research Report in Applied Mathematics and Systems Theory (November 1968), Rice University, Houston.

APPENDIX A

DERIVATION OF TERMINAL FAILURE RATE FUNCTION

In this appendix, the expression given in eqn. (2.6) for $\theta(t)$ is derived. The derivation is done in two steps. First, it is shown that under the assumption of very small values of μ_i , ($i = 1, 2, \dots, n$), then $F(t) \rightarrow 0$ and the failure rate $\theta(t)$ defined previously as

$$\theta(t) = \frac{f(t)}{1 - F(t)}$$

will be asymptotically equal to $f(t)$. An expression is then derived for

$$\theta(t) = f(t) = \frac{dF(t)}{dt}$$

In every state $x(t)$ some of its elements will be in a failure state for which $x_i(t) = 0$ and other elements in an operating state for which $x_i(t) = 1$. If the set S_1 contains all the elements of the vector $x(t)$ in the operating state and the set S_2 contains all those elements in the failure state, i.e.,

$$S_1 \equiv \{i | x_i(t) = 1\} \quad \text{and} \quad S_2 \equiv \{i | x_i(t) = 0\}$$

Then the expression for $F(t)$ can be written as

$$F(t) = \sum_{\{x(t) | \psi[x(t)] = 0\}} \left[\prod_{S_1} e^{-\epsilon v_i t^k} \prod_{S_2} \left(1 - e^{-\epsilon v_i t^k} \right) \right]$$

where

$$\mu_i = \epsilon v_i \quad \text{for } i = 1, 2, \dots, n.$$

We note that as ϵ tends to zero the term:

$$\sum_{S_1} e^{-\epsilon v_i t^k}$$

is equal asymptotically to 1 and the term:

$$\sum_{S_2} \left(1 - e^{-\epsilon v_i t^k} \right)$$

is equal asymptotically to 0.

In the previous expression for $F(t)$, each term of the summation is in the form:

$$\prod_{S_1} e^{-\epsilon v_i t^k} \prod_{S_2} \left(1 - e^{-\epsilon v_i t^k} \right)$$

Since the sum is made over all the failing states, the set S_2 must contain at least one failing element for all the terms in the summation, which means all these terms tend to zero as $\epsilon \rightarrow 0$, i.e., $F(t) \rightarrow 0$ as $\epsilon \rightarrow 0$. Then it follows that the failure rate, $\theta(t)$, defined previously as:

$$\theta(t) = \frac{f(t)}{1 - F(t)}$$

will be equal to $f(t)$ as $\epsilon \rightarrow 0$.

The density function $f(t)$ is the derivative w.r.t. time of the distribution function $F(t)$. Taking the derivative

$$\begin{aligned} F(t) &= \frac{d}{dt} F(t) \\ &= \sum_{\{x(t) | \psi[x(t)] = 0\}} \prod_{S_1} e^{-\epsilon v_i t^k} \prod_{S_2} \left(1 - e^{-\epsilon v_i t^k} \right) \\ &\quad \times \left[\sum_{S_1} -\epsilon v_i k t^{k-1} + \sum_{S_2} \frac{\epsilon v_i k t^{k-1} e^{-\epsilon v_i t^k}}{1 - e^{-\epsilon v_i t^k}} \right] \end{aligned}$$

Then investigating the terms of the above equation as $\epsilon \rightarrow 0$, it is found that

$$\prod_{S_1} e^{-\epsilon v_i t^k} \rightarrow 1 \quad \text{as } \epsilon \rightarrow 0.$$

$$\begin{aligned} \prod_{S_2} \left(1 - e^{-\epsilon v_i t^k} \right) &= \prod_{S_2} \left[1 - (1 - \epsilon v_i t^k + o(\epsilon)) \right] \\ &= \prod_{S_2} \left(\epsilon v_i t^k + o(\epsilon) \right) \quad \text{as } \epsilon \rightarrow 0, \end{aligned}$$

$$\sum_{S_2} \frac{\epsilon v_i k t^{k-1} e^{-\epsilon v_i t^k}}{1 - e^{-\epsilon v_i t^k}} = \sum_{S_2} \frac{\epsilon v_i k t^{k-1} e^{-\epsilon v_i t^k}}{1 - (1 - \epsilon v_i t^k + o(\epsilon))}$$

$$= \sum_{S_2} \frac{k}{t} \quad \text{as } \epsilon \rightarrow 0$$

Substituting in the expression for $f(t)$,

$$f(t) = \sum_{\{x(t) | \psi[x(t)] = 0\}} \left\{ \prod_{S_2} \epsilon v_i t^k \left[\sum_{S_1} - \epsilon v_i k t^{k-1} + \sum_{S_2} \frac{k}{t} \right] \right\} \text{ as } \epsilon \rightarrow 0.$$

$$= \sum_{\{x(t) | \psi[x(t)] = 0\}} \left\{ \prod_{S_2} \mu_i t^k \left[\sum_{S_2} \frac{k}{t} \right] \right\} \text{ as } \epsilon \rightarrow 0.$$

assume that the number of failing elements in the state $x(t)$, i.e., the number of element of S_2 is $\ell(x)$, then

$$f(t) = \sum_{\{x(t) | \psi[x(t)] = 0\}} \left\{ t^{k\ell(x)} \left[\prod_{S_2} \mu_i \right] \frac{k\ell(x)}{t} \right\}$$

$$f(t) = \sum_{\{x(t) | \psi[x(t)] = 0\}} \left\{ k\ell(x) t^{k\ell(x)-1} \prod_{S_2} \mu_i \right\}$$

In computer communication networks, elements of the set $\{x(t) | \psi[x(t)] = 0\}$ corresponds to all the mixed cut sets between a pair of nodes. As an approximation we can consider in the above expression only

the vectors $x(t)$ for which $\ell(x)$ is a minimum. This is a valid approximation since for very small μ

$$\mu^{\ell(x)+r} \ll \mu^{\ell(x)} \quad \text{for } r \geq 1$$

In computer communication networks, this approximation corresponds to considering only the minimum mixed cut sets. If we assume the set ζ to contain all these minimum mixed cut sets, $S(\zeta_j)$ to be the set of elements of the cut set ζ_j and m is the number of elements in each, then,

$$f(t) = \theta(t) \approx \sum_{\zeta} \left\{ kmt^{km-1} \prod_{S(\zeta_j)} \mu_{\zeta_j} \right\}$$

APPENDIX B

JOINT PROBABILITY DISTRIBUTION OF NUMBER OF MESSAGES IN QUEUE

In the following we shall derive an expression for the joint probability distribution for the number of messages in a non-break in priority M/G/1 queue with 2 priority levels.

Arrival processes of both types of messages are Poisson processes with the probability distribution function,

$$A^{(i)}(t) = 1 - e^{-\lambda_i t} \quad i = 1, 2$$

where

i is the priority class number (the lower the index, the higher the priority level)

λ_i is the average arrival rate of messages of priority class i

$A^{(i)}(t)$ is the probability distribution function of inter-arrival times for messages of priority class i .

Also define $B^{(i)}(t)$ as the service time distribution for the messages of priority class i . $\beta^{(i)}(s)$ is the Laplace transform of $B^{(i)}(t)$ and is given by:

$$\beta^{(i)}(s) = \int_0^{\infty} e^{-st} dB^{(i)}(t)$$

$$1/\mu_i = \int_0^{\infty} t dB^{(i)}(t)$$

If we denote the total number of messages left behind in the system by the n^{th} departing message by x_n and let $(v_j, j = 0, 1, \dots)$ be its stationary distribution, we can define the following Z transform for $|Z| \leq 1$:

$$V(Z) \stackrel{\text{def}}{=} \sum_{j=0}^{\infty} Z^j v_j = \lim_{n \rightarrow \infty} E\{Z^{x_n}\}$$

Denote the number of high and low priority messages left behind in the system by the n^{th} departing message after $t=0$ by $x_n^{(1)}$ and $x_n^{(2)}$. Further, $v_j^{(i)}$ will denote the number of i^{th} priority messages arriving during a service time of a j^{th} priority message $i = 1, 2, j = 1, 2$. We note that if at some moment t_0 the system is empty,

$$\frac{\lambda_i}{\lambda_1 + \lambda_2}$$

is the probability that the first message arriving after t_0 is an i^{th} priority, for $|Z| \leq 1$ we can write the following equality for the messages of priority class (1):

$$\begin{aligned} E \left\{ Z^{x_{n+1}^{(1)}} \mid x_0 = 0 \right\} &= E \left\{ Z^{x_n^{(1)} + v_n^{(1)}} \left(x_n^{(1)} > 0 \right) \mid x_0 = 0 \right\} \\ &+ E \left\{ Z^{v_2^{(1)}} \left(x_n^{(1)} = 0 \right) \left(x_n^{(2)} = 0 \right) \mid x_0 = 0 \right\} \end{aligned}$$

$$\begin{aligned}
 & + E \left\{ Z_1^{v_1^{(1)}} \left(x_n^{(1)} = 0 \right) \left(x_n^{(2)} = 0 \right) \mid x_0 = 0 \right\} \frac{\lambda_1}{\lambda_1 + \lambda_2} \\
 & + E \left\{ Z_2^{v_2^{(1)}} \left(x_n^{(1)} = 0 \right) \left(x_n^{(2)} = 0 \right) \mid x_0 = 0 \right\} \frac{\lambda_2}{\lambda_1 + \lambda_2} \quad (B.1)
 \end{aligned}$$

Let us denote the stationary distribution of the number of first priority messages left behind in the systems by a departing message by $(v_j^{(1)}, j = 0, 1, \dots)$. It follows for $\lambda_1/\mu_1 + \lambda_2/\mu_2 < 1$ that:

$$\begin{aligned}
 \lim_{n \rightarrow \infty} P_n \left\{ x_n^{(1)} = 0, x_n^{(2)} = 0 \mid x_0 = 0 \right\} \\
 & = 1 - \frac{\lambda_1}{\mu_1} - \frac{\lambda_2}{\mu_2} \\
 & = 1 - \rho_1 - \rho_2 \quad (B.2)
 \end{aligned}$$

where

$$\rho_1 = \frac{\lambda_1}{\mu_1}$$

$$\rho_2 = \frac{\lambda_2}{\mu_2}$$

By definition it follows that $v_0^{(1)}$ is the probability that no messages of priority (1) are left behind in the system by a departing message:

$$v_0^{(1)} = \lim_{n \rightarrow \infty} P_r \left\{ x_n^{(1)} = 0, x_n^{(2)} > 0 \mid x_0 = 0 \right\} \\ + \lim_{n \rightarrow \infty} P_r \left\{ x_n^{(1)} = 0, x_n^{(2)} = 0 \mid x_0 = 0 \right\}$$

Therefore

$$\lim_{n \rightarrow \infty} P_r \left\{ x_n^{(1)} = 0, x_n^{(2)} > 0 \mid x_0 = 0 \right\} = v_0^{(1)} - (1 - \rho_1 - \rho_2) \quad (B.3)$$

If we define

$$v^{(1)}(z) = \sum_{j=0}^{\infty} z^j v_j^{(1)} = \lim_{n \rightarrow \infty} E \left\{ z^{x_n^{(1)}} \mid x_0 = 0 \right\}$$

$$v^{(1)}(1) = \sum_{j=0}^{\infty} v_j^{(1)} = 1$$

then

$$\lim_{n \rightarrow \infty} E \left\{ z^{x_{n+1}^{(1)}} \mid x_0 = 0 \right\} = v^{(1)}(z) \quad (B.4)$$

$$\lim_{n \rightarrow \infty} E \left\{ z^{x_{n-1}^{(1)} + v_1^{(1)}} \left(x_n^{(1)} > 0 \right) \mid x_0 = 0 \right\}$$

$$= \lim_{n \rightarrow \infty} E \left\{ x_n^{(1)} > 0, z^{x_{n-1}^{(1)}} \mid x_0 = 0 \right\} \lim_{n \rightarrow \infty} E \left\{ z^{v_1^{(1)}} \mid x_0 = 0 \right\}$$

$$= Z^{-1} \left[v^{(1)}(Z) - v_0^{(1)} \right] \lim_{n \rightarrow \infty} E \left\{ Z^{v_1^{(1)}} \mid x_0 = 0 \right\} \quad (B.5)$$

It can be easily shown that

$$\lim_{n \rightarrow \infty} E \left\{ Z^{v_1^{(1)}} \mid x_0 = 0 \right\} = \beta^{(1)}(q)$$

and

$$\lim_{n \rightarrow \infty} E \left\{ Z^{v_2^{(1)}} \mid x_0 = 0 \right\} = \beta^{(2)}(q)$$

where

$$\beta^{(1)}(q) = \int_0^{\infty} e^{-qt} dB^{(1)}(t)$$

and

$$q = \lambda_1(1-Z)$$

Furthermore

$$\begin{aligned} & \lim_{n \rightarrow \infty} E \left\{ \left\{ x_n^{(1)} = 0 \right\} \left\{ x_n^{(2)} > 0 \right\} Z^{v_2^{(1)}} \mid x_0 = 0 \right\} \\ &= \lim_{n \rightarrow \infty} E \left\{ \left\{ x_n^{(1)} = 0 \right\} \left\{ x_n^{(2)} > 0 \right\} \mid x_0 = 0 \right\} \lim_{n \rightarrow \infty} E \left\{ Z^{v_2^{(1)}} \mid x_0 = 0 \right\} \\ &= \left[v_0^{(1)} - (1 - \rho_1 - \rho_2) \right] \beta^{(2)}(q) \quad (B.6) \end{aligned}$$

Similarly:

$$\begin{aligned} \lim_{n \rightarrow \infty} E \left\{ \left(x_n^{(1)} = 0 \right) \left(x_n^{(2)} = 0 \right) z^{v_1^{(1)}} \mid x_0 = 0 \right\} \frac{\lambda_1}{\lambda_1 + \lambda_2} \\ = (1 - \rho_1 - \rho_2) \beta^{(1)}(q) \frac{\lambda_1}{\lambda_1 + \lambda_2} \end{aligned} \quad (B.7)$$

and

$$\begin{aligned} \lim_{n \rightarrow \infty} E \left\{ \left(x_n^{(1)} = 0 \right) \left(x_n^{(2)} = 0 \right) z^{v_2^{(1)}} \mid x_0 = 0 \right\} \frac{\lambda_2}{\lambda_1 + \lambda_2} \\ = (1 - \rho_1 - \rho_2) \frac{\lambda_2}{\lambda_1 + \lambda_2} \beta^{(2)}(q) \end{aligned} \quad (B.8)$$

Substituting (B.4), (B.5), (B.6), (B.7) and (B.8) into (B.1):

$$\begin{aligned} v^{(1)}(z) = z^{-1} \beta^{(1)}(q) \left[v^{(1)}(z) - v_0^{(1)} \right] + \left[v_0^{(1)} - (1 - \rho_1 - \rho_2) \right] \beta^{(2)}(q) \\ + (1 - \rho_1 - \rho_2) \beta^{(1)}(q) \frac{\lambda_1}{\lambda_1 + \lambda_2} + (1 - \rho_1 - \rho_2) \frac{\lambda_2}{\lambda_1 + \lambda_2} \beta^{(2)}(q) \end{aligned}$$

Hence

$$\begin{aligned} v^{(1)}(z) \left[1 - z^{-1} \beta^{(1)}(q) \right] = v_0^{(1)} \left[\beta^{(2)}(q) - z^{-1} \beta^{(1)}(q) \right] - \beta^{(2)}(q) \\ (1 - \rho_1 - \rho_2) \frac{\lambda_2}{\lambda_1 + \lambda_2} + \beta^{(1)}(q) (1 - \rho_1 - \rho_2) \frac{\lambda_1}{\lambda_1 + \lambda_2} \end{aligned}$$

therefore

$$v^{(1)}(Z) = v_0^{(1)} \left[\frac{\beta^{(1)}(q) - Z \beta^{(2)}(q)}{\beta^{(1)}(q) - Z} \right] + (1 - \rho_1 - \rho_2) \frac{\lambda_1}{\lambda_1 + \lambda_2} \frac{\beta^{(2)}(q) - \beta^{(1)}(q)}{\beta^{(1)}(q) - Z} Z \quad (B.9)$$

where

$$\beta^{(i)}(q) = \int_0^{\infty} e^{-\lambda_1(1-Z)t} dB^{(i)}(t)$$

If we take the $\lim_{Z \rightarrow 1}$ of eqn. (B.9) noting that

$$\lim_{Z \rightarrow 1} v^{(1)}(Z) = 1$$

$$\lim_{Z \rightarrow 1} \beta^{(i)}(q) = \int_0^{\infty} dB^{(i)}(t) = 1$$

$$\lim_{Z \rightarrow 1} \frac{d}{dZ} \beta^{(i)}(q) = \lambda_1 \int_0^{\infty} t dB^{(i)}(t) = \frac{\lambda_1}{\mu_1}$$

then eqn. (B.9) after taking the limit will be reduced to

$$1 = v_0^{(1)} \left[\frac{\frac{\lambda_1}{\mu_1} - 1 - \frac{\lambda_1}{\mu_2}}{\frac{\lambda_1}{\mu_1} - 1} \right] + (1 - \rho_1 - \rho_2) \frac{\lambda_1}{\lambda_1 + \mu_2} \frac{\frac{\lambda_1}{\mu_2} - \frac{\lambda_1}{\mu_1}}{\frac{\lambda_1}{\mu_1} - 1}$$

from which we can obtain the value of $v_0^{(1)}$ as

$$v_0^{(1)} = 1 - \frac{\lambda_1}{\mu_1 + \mu_2} \left(\frac{\lambda_1}{\mu_1} + \frac{\lambda_2}{\mu_2} \right)$$

Substituting in eqn. (B.3)

$$\begin{aligned} \lim_{n \rightarrow \infty} P_r \left\{ x_n^{(1)} = 0, x_n^{(2)} > 0 \mid x_0 = 0 \right\} &= 1 - \frac{\lambda_1}{\lambda_1 + \lambda_2} (\rho_1 + \rho_2) - (1 - \rho_1 - \rho_2) \\ &= \rho_1 + \rho_2 \left(1 - \frac{\lambda_1}{\lambda_1 + \lambda_2} \right) \\ &= \rho \frac{\lambda_2}{\lambda} \end{aligned}$$

$$\begin{aligned} \lim_{n \rightarrow \infty} P_r \left\{ x_n^{(1)} > 0, x_n^{(2)} \geq 0 \mid x_0 = 0 \right\} &= 1 - \left(1 - \rho + \rho \frac{\lambda_2}{\lambda} \right) \\ &= \rho \left(1 - \frac{\lambda_2}{\lambda} \right) \\ &= \rho \frac{\lambda_1}{\lambda} \end{aligned}$$

To summarize:

$$\lim_{n \rightarrow \infty} P_r \left\{ x_n^{(1)} = 0, x_n^{(2)} = 0 \mid x_0 = 0 \right\} = 1 - \rho$$

$$\lim_{n \rightarrow \infty} P_r \left\{ x_n^{(1)} = 0, x_n^{(2)} > 0 \mid x_0 = 0 \right\} = \rho \frac{\lambda_2}{\lambda}$$

$$\lim_{n \rightarrow \infty} P_r \left\{ x_n^{(1)} > 0, x_n^{(2)} \geq 0 \mid x_0 = 0 \right\} = \rho \frac{\lambda_1}{\lambda}$$

APPENDIX C

RELATION OF FILE AVAILABILITY MEASURE a'_{ij} TO ACTUAL FILE AVAILABILITY

In this appendix, we shall prove that the availability measure, a'_{ij} , is an upper bound to the actual file availability, a_{ij} , where a'_{ij} is given by eqn. (4.9) as

$$a'_{ij} = \left[1 - \prod_{k=1}^n (1 - r_{ik} x_{kj}) \right], \quad r_{ii} = 1 \quad (C.1)$$

In eqn. (C.1), all node components are assumed to be perfectly reliable. We shall prove the following inequality:

$$a'_{ij} \geq a_{ij} \quad (C.2)$$

Following the notations of Section 2.3, let the binary vector, x , given by

$$x \triangleq (x_1, x_2, \dots, x_{2m})$$

Characterize the state of the network ($2m$ is the number of links). Also define $\psi_{ik}(x)$ to be the structural function for nodes i and k . Define Q_{ik} to be the set of network states such that each of these states corresponds to possible communications between nodes i and k . The set Q_{ik} can be

expressed as,

$$Q_{ik} \triangleq \{x | \psi_{ik}(x) = 1\}$$

Finally, we define $p(x^{(l)})$ to be the probability of system state $x^{(l)}$,

$$p(x^{(l)}) = \prod_{i=1}^{2m} [x_i p_i^{(l)} + (1 - x_i) q_i^{(l)}]$$

where $p_i^{(l)}$ is the probability of operation of element i , ($i = 1, 2, \dots, 2m$), in state $x^{(l)}$ and $q_i^{(l)} = 1 - p_i^{(l)}$.

If we assume that s copies of file j are stored in as many nodes, the actual availability, a_{ij} , of file j to node i (node i does not include a copy of file j) will be given by

$$a_{ij} = \sum_{x^{(l)} \in (Q_{ik_1} \cup Q_{ik_2} \cup \dots \cup Q_{ik_s})} p(x^{(l)}) \quad (C.3)$$

and file availability measure, a_{ij} , can be expressed as,

$$a_{ij} = 1 - \left\{ \left[1 - \sum_{x^{(l)} \in Q_{ik_1}} p(x^{(l)}) \right] \times \left[1 - \sum_{x^{(l)} \in Q_{ik_2}} p(x^{(l)}) \right] \times \dots \right. \\ \left. \dots \times \left[1 - \sum_{x^{(l)} \in Q_{ik_s}} p(x^{(l)}) \right] \right\} \quad (C.4)$$

Consider the following two cases:

Case 1: The sets $Q_{ik_1}, Q_{ik_2}, \dots, Q_{ik_s}$ are not mutually exclusive. The expression given by eqn. (C.3) considers the union of these sets while the expression given by eqn. (C.4) assumes that they are mutually exclusive. Thus (C.4) over-estimates the file availability, i.e.,

$$a'_{ij} > a_{ij}$$

Case 2: The sets $Q_{ik_1}, Q_{ik_2}, \dots, Q_{ik_s}$ are mutually exclusive. In this case the expressions (C.3) and (C.4) are equivalent. Thus

$$a'_{ij} = a_{ij}$$

It follows that in general,

$$a'_{ij} \geq a_{ij}$$

END

1901

FIN