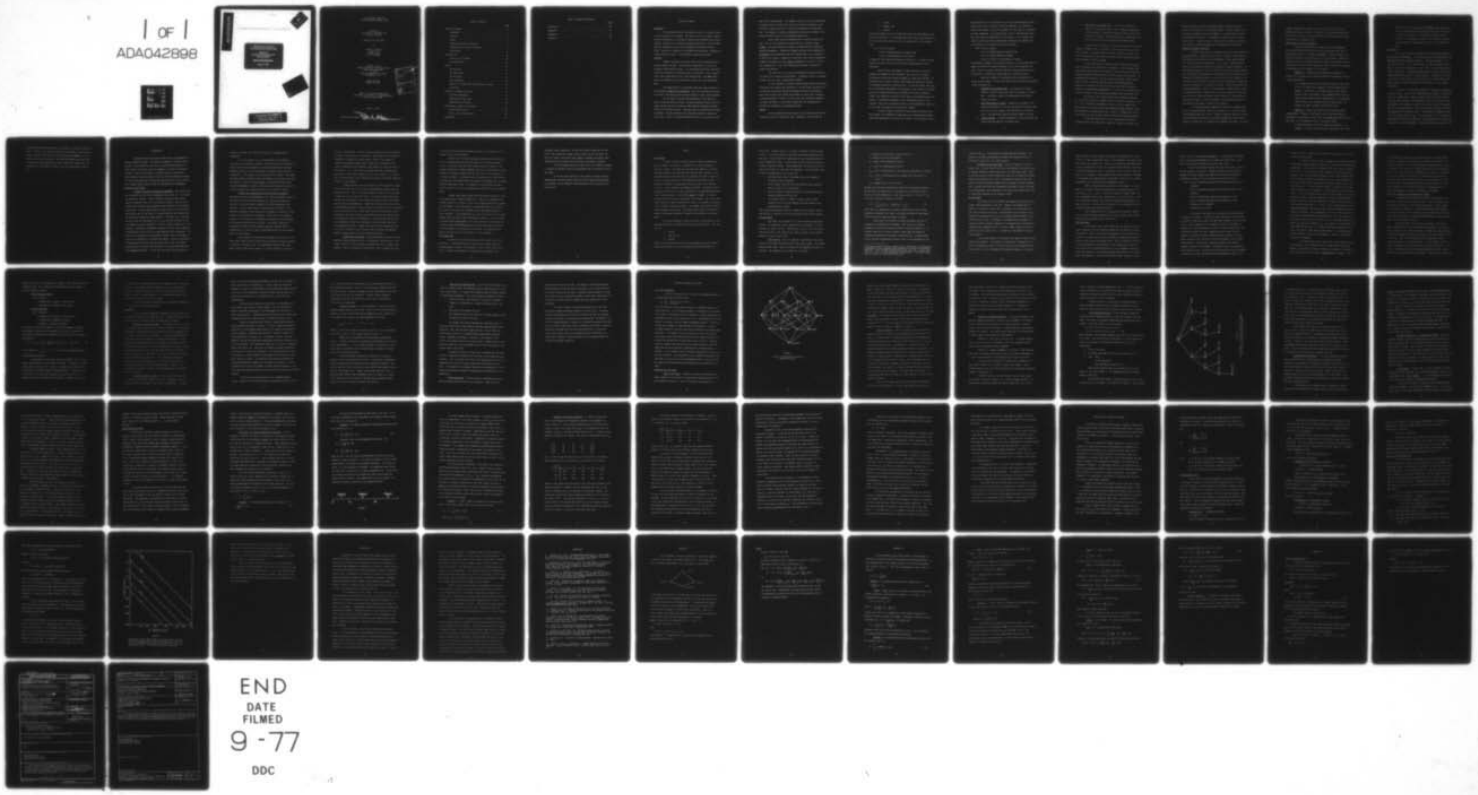
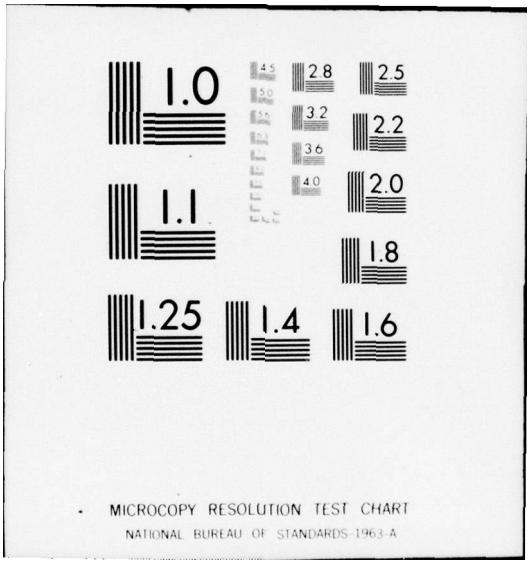
AD-A042 898 ILLINOIS UNIV AT URBANA-CHAMPAIGN CENTER FOR ADVANCED--ETC F/G 9/2
RESEARCH IN NETWORK DATA MANAGEMENT AND RESOURCE SHARING. NETWO--ETC(U)
AUG 76 G G BELFORD, J D DAY, E GRAPA DCA100-75-C-0021
UNCLASSIFIED UIUC-CAC-DN-76-203 CCTC-WAD-6506 NL

1 OF 1
ADA042898



END
DATE FILMED
9-77
DDC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

Center for Advanced Computation

CAC Document Number 203
CCTC-WAD Document Number 6506

Research in
Network Data Management and
Resource Sharing

Network File Allocation

August 2, 1976

CAC Document Number 203
CCTC-WAD Document Number 6506

Research in Network Data Management and Resource Sharing

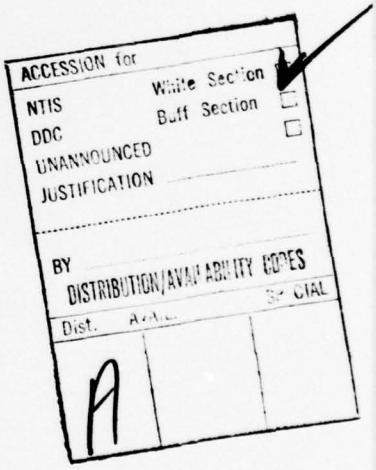
Network File Allocation

by

Geneva G. Belford
John D. Day
Enrique Grapa
Paul M. Schwartz

Prepared for the
Command and Control Technical Center
WWMCCS ADP Directorate
of the
Defense Communications Agency
Washington, D.C.

under contract
DCA100-75-C-0021



Center for Advanced Computation
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

August 2, 1976

Approved for release: Peter A. Alsberg

Peter A. Alsberg, Principal Investigator

(Rev 1493)

Table of Contents

	Page
Executive Summary	1
Background	1
Overview	1
Models	2
Finding the Optimal Allocation	6
Applications to Special Situations	7
Conclusions	8
Introduction	10
History of the Problem	10
The Present Work	13
Models	15
Introduction	15
The Casey Model	16
The Chu Model	18
The Levin Model	19
Model for a Primary-Copy Synchronization Scheme	22
Constraints	24
Finding the Optimal Allocation	29
Zero-One Programming	29
Searching the Cost Graph	29
Simplifying the Problem	38
Applications to Special Situations	47
Single Update Source	48
Use of a Local Partial Copy	50
Conclusions	54

Table of Contents (continued)

	Page
References	56
Appendix 1	57
Appendix 2	59
Appendix 3	63

Executive Summary

Background

This document presents the results to date of a research study of file allocation in a network. The study is part of a larger, comprehensive investigation of problems in network data management and resource sharing. The goal is to develop techniques applicable to the World-Wide Military Command and Control System (WWMCCS) Intercomputer Network (WIN). The work is supported by the WWMCCS ADP Directorate, Command and Control Technical Center, of the Defense Communications Agency.

Overview

Computer networks can provide rapid access to geographically distant computer systems. This creates an opportunity to set up an automated distributed data base. In a distributed data base, different files can be located at different sites in the network. Also, copies of files can be replicated at one or more backup sites. The WWMCCS data base is an example of such a multi-file, multi-copy, distributed data base.

The engineering of a distributed data base leads naturally to the problem of network file allocation. This is the problem of placing the files at the various sites in some optimal (or near optimal) manner.

In our study of the allocation problem we first develop a model for how the files are used. The model carefully defines the file usage process that we are trying to make efficient through an optimal choice of allocation strategy. The next step is to define what is meant by optimal. In most studies in the literature optimal is defined as least cost. "Cost" is a broadly defined term that can include far more

than dollar considerations. For example, dollar costs can be multiplied by weight factors to reflect the scarcity of certain resources or the priority of special functions. There are alternatives to using least cost. For example, it might be worthwhile to define the optimum as the allocation which results in the minimum response time.

Once a model has been developed and it has been decided that (say) cost is to be minimized, the next step is to develop a cost formula. The cost formula will reflect the model and will include all essential parameters. Additional formulas may have to be developed for constraints in the case where the cost formula alone is not sufficient to define the optimum. Examples of constraints that would be imposed in a WWMCCS environment are the a priori assignment of a file to a site or the elimination of a site from consideration because of security or command requirements.

The last step is to solve the optimization problem defined by the model, cost formula and constraints. In general, this is a straightforward, but very large, computational problem.

In this document we examine several possible models and discuss associated cost formulas and constraints. We then study the question of solving the optimization problem and present several new techniques which will in many cases reduce the problem to a size that is computationally tractable. Finally, we look at two file allocation questions for which the model is sufficiently simple that the problem may be solved with essentially no computational work.

Models

We have identified three aspects of file handling that must be included in any file allocation study. Basically, the file must be

1. stored,

2. updated, and

3. queried.

The precise impact on cost of these basic processes will depend on how they are carried out. In the models which one usually sees discussed in the file allocation literature, the cost formula is of the following form:

Cost = Cost of Storage

+ Cost of Sending Queries to Nearest Copy

+ Cost of Sending Updates to All Copies

A formula of this type was introduced by Casey [4]. It varies in some respects from that used by Chu [6] in his pioneering work on file allocation.

Notice that as the number of copies increases, the cost of updates and storage will also increase. But the cost of querying decreases as closer sites become available to receive the queries. We therefore have a classic tradeoff, which makes the problem of finding the cheapest allocation a nontrivial one. In case the reader feels that not all relevant costs appear to be included in this formula, we remark that only a slight modification is needed for this basic scheme to reflect such additional features as the costs of processing queries and updates. Furthermore, parameters can be interpreted broadly. The "nearest" copy is not necessarily nearest in the geographical sense, but the one that can respond to the query most cheaply.

We have, however, identified a more serious difficulty with this scheme. The assumption is made that all sites generating updates send them independently to all the file copies. In our parallel work on

synchronization [1] and resiliency [2], we have identified many difficulties with such a "broadcast" model for updating. We find that a primary-copy scheme, in which all updates are first sent to a "primary" has many advantages. The primary plays a role in ensuring both resiliency and proper synchronization of the updates. For a primary-copy scheme in which the primary subsequently broadcasts the updates, the cost formula will be something like the following:

Cost = Cost of Storage
+ Cost of Sending Queries to Nearest Copy
+ Cost of Sending Updates to Primary
+ Cost for Primary to Send Updates to Copies

Surprisingly, although this formula looks like it would probably lead to higher cost than the broadcast formula, this is not necessarily the case. That is, in many situations lower-cost file allocations can be obtained for a primary-copy scheme than for a broadcast scheme.

There are a number of constraints which one might want to impose on the problem. The four most useful ones that we find discussed in the literature are:

1. Forbidden or prescribed sites. For historical or policy reasons, a copy of a data base might be required at a given site. Conversely, security considerations could forbid certain locations.
2. Prescribed number of copies. Primarily for reasons of increased availability [3], it may be desirable to specify that there be, say, two copies of the data base, or of certain files. One might also specify a minimal number of copies.
3. Limited storage. It may be necessary to take into account the limited storage capacity of certain sites.

4. Upper bound on response time. If the access process is modeled in considerable detail, and if enough information on system parameters is known, the expected time to access each file may be computed. The constraint can be imposed that this access time be less than some prescribed maximum value.

Of these four constraints, the first two appear to be the most relevant to WWMCCS needs. Fortunately, they also turn out to be very easy to impose. Straightforward conditions on number of copies or on file locations reduce the options and hence tend to reduce the computation necessary to find an optimum.

On the other hand, the third and fourth constraints actually tend to increase the difficulty of the problem. This is because these constraints couple together the impact of all the files in the system. Storage limitations obviously must take into account all the files. In response-time constraints, the file locations interact in more subtle ways, such as in increased network traffic and resulting delays. Without such coupling, the best location for each file (or group of files) can be determined independently. It turns out to be much easier to solve a number of small allocation problems than one gigantic one.

One further point should be made about storage constraints. It is not only difficult to impose them, but it may be unrealistic to do so. If it is otherwise worthwhile to store a file at a given site, there seems to be no good reason why additional storage capacity could not be added to accommodate that file. That is, "storage cost" in the cost formula can just as well reflect the cost of new storage capacity.

If one takes this point of view, then the only classical constraint which is troublesome to impose is a bound on response time.

Further study may be useful to determine whether response-time constraints can be handled in some simple fashion - and also whether they are really worthwhile in most environments. It may be that some more straightforward constraint - for example, on network traffic between various sites - would be easier to apply and at the same time have the same effect on system performance as the response-time constraint.

Finding the Optimal Allocation

If there are n sites in a network, and for each site there are two possibilities - either it has a copy or it doesn't - then there are 2^n different file allocations to be considered. (This number becomes 2^n-1 if we omit the null allocation, i.e. no copy.) Computationally, there are no foolproof shortcuts to finding the optimal allocation. As the network increases in size, the work to solve the allocation problem will in general grow exponentially, doubling with every additional site. Consider the following example. Casey [4] notes that a program run on the 360/91 took less than 10 seconds to solve six different single-file allocation problems for a network of 19 sites. (This time included the Fortran compilation.) Thus, if we assume that it takes about 2 seconds to optimally allocate a single file in a 19-site network, it will take over an hour in a 30-site network, about 48 days in a 40-site network, and about 136 years in a 50-site network. These figures must then be multiplied by the number of files to be allocated.

It therefore becomes imperative to reduce the computational work in any way possible. Casey [4] and Urano et al. [13] have given some theorems which help in this. Taking a different approach, we have developed three theorems which determine *a priori* that certain sites should, or should not, be included in any optimal allocation. Each such

a priori determination reduces by a factor of two the size of the remaining allocation problem. We therefore feel that our theorems can be enormously useful in reducing the file allocation problem for large networks down to manageable size.

Applications to Special Situations

Following another line of research, we have investigated certain simple models and related allocation questions which may be particularly useful in the WWMCCS environment. First we assumed that network transmission cost is the same for all site pairs. However, we added the costs of processing queries and updates to the model, and we assumed that these may differ from site to site. Here are the questions we investigated, with summaries of our results.

Question 1. Suppose that all updates are generated at a single site, but queries may originate anywhere in the network. What is the best allocation?

In a homogeneous network, the optimal allocation often turns out to be a single copy, located at the site generating the updates. Specifically, this happens when it costs more to maintain (store and update) a copy elsewhere than it does to answer remote queries from the updating site. In a heterogeneous network, more tradeoffs enter into the picture. We present two simple strategy rules for this case.

Question 2. Suppose that there are only two sites under consideration - a remote site holding a copy of the data base and responsible for updating it, and a local site generating queries for the data base. Is it worthwhile to have a local data cache?

A simple inequality suffices to answer this question. For example, if we save $0.10 per query by having the local cache,

and the cost of local storage is $100/month, then the local cache is cost effective for all query rates over (100/0.1)/mo. or about 1.4 per hour. If the savings per query is only $0.01; i.e., decreases by a factor of 10, then the crossover point increases by the same factor of 10, to 14 queries per hour.

Conclusions

It is not very hard to formulate a file allocation problem in any particular environment. It is only necessary to understand the file handling process well enough to model it. For small networks - say less than 20 sites - it is a simple matter to find the optimal allocation, by brute force computation of the "costs" of all possible allocations if necessary. Considerable headway has been made on reducing the computational difficulties for larger networks. In addition, in particular situations the problem may be simplified to such an extent that general rules of thumb are available to yield good - if not optimal - allocations.

In short, the theoretical problems of optimal file allocation are well in hand. The data base administrator who wishes to distribute files in a network must, however, address several practical problems. What model best reflects his environment? What are the values of the parameters? (For example, how should the various "costs" be computed?) What are the usage patterns? (That is, what query and update volumes are generated at the various sites?) The question of parameter values is a particularly difficult one. In some respects, it is a problem of measurement and statistical analysis. Only measurement can yield usage patterns for a real environment. Statistical analysis can help to say

whether those patterns seem static, oscillatory, or have some long-term trend. "Costs" can be even more difficult to determine, since they may reflect not only "real" costs, but some subjective judgment as to the value or scarcity of resources. The data base administrator must play an active role in determining the "best" network file allocation. The researcher can only supply the basic tools and instructions on their use.

Introduction

Computer networks can provide rapid access to geographically distant computer systems. This opens up a number of attractive possibilities. One possibility is to set up a distributed data base, in which copies of different files (or segments of files) are located at different sites in the network. An obvious question then arises: where should the files be placed? This is the file allocation problem. It is not surprising that this problem, being basic as well as obvious, has a longer history than any other in distributed data management.

History of the Problem

Optimal allocation by zero-one programming. The earliest work on the network file allocation problem was done by Chu [6]. Chu states the problem as follows: "Given a number of computers that process common information files, how can we allocate files so that the allocation yields minimum overall operating costs subject to the following constraints: (1) The expected time to access each file is less than a given bound, and (2) the amount of storage needed at each computer does not exceed the available storage capacity." Variables X_{ij} to describe the allocation are introduced; $X_{ij}=1$ if the jth file is stored in the ith computer and $X_{ij}=0$ otherwise. In order to apply constraint (1), Chu develops a reasonably comprehensive formula for access time, including queueing delays and the effect of intercomputer traffic congestion. The overall cost expression to be minimized includes costs for storage as well as for transmission of queries and updates. Since the variables to be determined can take on only the values zero or one, the optimal allocation may be found as the solution to a so-called nonlinear zero-one programming problem. (In fact Chu notes that the problem may be

reduced to a linear one, which may be solved by straightforward techniques.)

In a later paper, Chu [7] discusses how an availability constraint can be added to the model. The main idea is to determine in advance (from simple assumptions on failure probabilities) how many redundant copies of a file are required to achieve a desired level of availability. This number is then inserted into the model. The scheme otherwise remains unchanged, since Chu assumed in his earlier work that the number of copies was given a priori. The difficulty with using zero-one programming to solve the file allocation problem is that it is so time-consuming as to seem impractical for a large file system.

Optimal and suboptimal allocation by search procedures. Believing that zero-one programming is too costly an approach, Casey [4] developed an efficient search procedure for finding a minimal-cost solution, as well as heuristic methods for finding acceptably good solutions. Casey's model differs in some respects from Chu's. One important difference is that Casey lets the number of copies of a file, as well as its locations, be variables. Notice that as the number of copies of a file increases, the expense of querying the file decreases, but storage and updating costs increase. Thus Casey's approach to optimization strikes a balance between these two opposing trends. A disadvantage, of course, is that the minimization may not yield enough copies for reliability.

Casey applied his optimization algorithm to real data for the ARPA network (when it had 19 sites) and thus showed the process feasible for networks of moderate size. His experiments indicated that when update traffic equals query traffic, it is most efficient to store the

file at a central node. As query traffic increases relative to updates, storage at multiple nodes is indicated. These results are intuitively reasonable. Although one always expects several local minima in a complex, multivariable minimization problem, it is noteworthy that Casey's experiments reveal extemely large numbers of them (over 100 in some cases). It is clear that any optimal allocation procedure must take care to avoid being trapped in such a local minimum. It is possible, of course, that a local minimum may provide an acceptable suboptimal solution to the problem.

Morgan and Levin have criticized both Chu's and Casey's models on the basis that they do not allow for dependencies between files and programs [10,11]. That is, a program (which is itself a file) may need to make use of one or more data files. The fact that these files must interact with one another is not taken into account in the older models which assume file independence. Morgan and Levin also point out that in a heterogeneous network it may not always be possible to store a particular file at an arbitrary site. Their model takes into account this type of constraint, which also includes the possibility that the allocation may be restricted by security considerations. The algorithm used to solve the optimization problem is a systematic search procedure, along the lines suggested by Casey. Dynamic features were also introduced; that is, costs to change the file allocation were considered and balanced against savings expected from the improved allocation.

Simplifying the problem. Model development has therefore been extensive and has produced models of considerable sophistication. More recent research in file allocation has emphasized ways to simplify the problem. This is an important consideration, since in general the work

to solve the file allocation problem increases as 2^n, where n is the number of sites in the network.

Urano et al. [13] have developed theorems which allow a network to be partitioned into what one might call "proximity groups" of sites. At most one site from each proximity group can appear in any optimal allocation. Roughly speaking, the costs of querying the various members of a proximity group (from any other site in the network) are all about the same. If querying costs are proportional to the distance the query must be shipped across the network, then members of a proximity group are physically close. For example, in a nation-wide network, a group of sites in, say, the Los Angeles area might form such a proximity group.

Another step towards making the problem more tractable is the very recent work of Chandy and Hewes [5]. They first reformulate the problem so that it is linear, at the cost of adding additional variables and constraints. They then ignore the requirement that the variables X_{ij} be zero or one and use a standard linear programming approach to find a minimum. If the minimizing X_{ij}'s turn out to have (as very often happens) zero-one values, then the problem is solved. If not, at least a possibly useful lower bound to the true minimum cost is obtained, and a linear integer-programming approach (with all the troubles involved for large numbers of constraints) can be tried. More detailed work needs to be done here, but the approach is very promising.

The Present Work

We here report on the work that we have carried out on file allocation. In the next section we discuss the different models in varying degrees of detail. In particular, Casey's model, and a variation of it which we developed for a primary-copy environment, are

discussed fairly completely. We felt that Casey's model has the simplicity and flexibility needed to form a basis for our own work. We also list various constraints which might be imposed, and assess their importance and the relative ease with which they might be applied.

In the following section we take up the very difficult problem of finding the optimal allocation and present some contributions that we have made.

In the two final sections of this report we briefly discuss applications to several special cases for which simply analyzed models are possible, and we summarize some conclusions we have drawn from our work to date.

Models

Introduction

In order to make a rational decision, based on quantitative data, on where to put files in a network, it is first necessary to construct a model. The model must include those aspects of file handling which affect cost, response time, and any other features of the distributed data system which the file allocator may wish to take into account. Once the process of file handling is defined by a model, it is possible to identify the parameters which enter into cost, response time, etc., and to generate the necessary formulas. The reader should be aware of the fact that a model inevitably is a simplified representation of the real world. A good model should be simple. It should include the key features of the process to be modeled. But at the same time, the level of detail should not be such as to preclude its application to slightly varying situations. For example, one should be able to modify parameter definitions. One should also be able to include more detail by developing submodels to compute the parameters in the basic model.

With these thoughts in mind, we identify three aspects of file handling which must be included in any file allocation study. The file must be

1. stored,
2. updated, and
3. queried.

Effects of these basic processes must be included in any cost formula. Their precise impact on cost will depend on how the processes are

carried out. Response time is, of course, a function of the querying process. It will also depend on processor and network characteristics and loads. The possibility of modeling at an ever increasing level of detail immediately arises. In this report we will restrict ourselves to flexible, high level models. Where appropriate, we will point out how lower level detail may enter into the parameters. The four models that we will be considering in this section are

1. the Casey model, which is perhaps the one most commonly discussed in the literature,
2. the Chu model, which was the first network file allocation model to appear in the literature,
3. the Levin model, an ambitious effort to include interaction between programs and files, and
4. a new model (really a variant of Casey's) that we have developed for use in environments that apply primary copy synchronization schemes.

After discussing parameters and cost formulas for the models, we will look briefly at formulas for various constraints which may be imposed.

The Casey Model

This model, as presented in [4], may be briefly described as follows. For simplicity, only a single file is considered. Any site may query or update the file. Updates must be sent to all file copies. Queries are sent to the "closest" one, where "closest" can be broadly defined in terms of lowest cost.

Cost function. The cost computed is essentially total operating cost over some chosen time period (day, week, month). The formula contains three terms: one for storage, one for updating, and one for querying. The parameters in the model are as follows:

I = index set of sites with a copy of the file

n = number of sites in the network

Ψ_j = update load originating at site j

λ_j = query load originating at site j

d_{jk} = cost of communication of one query unit from site j to site k

d'_{jk} = cost of communication of one update unit from site j to site k

σ_k = storage cost of file at site k

Storage costs and query loads must be those for the chosen time period. An allocation is specified by I, the index set of sites with a copy. That is, if sites numbered 1, 4, and 5 have a copy, $I = \{1,4,5\}$.* The cost $C(I)$ of a given allocation I is then

$$C(I) = \sum_{j=1}^{n} \left[\sum_{k \in I} \Psi_j d'_{jk} + \lambda_j \min_{k \in I} d_{jk} \right] + \sum_{k \in I} \sigma_k \qquad (1)$$

Since this cost formula and the notation defined above will be used frequently throughout this report, this page is printed on light green stock so that the reader may locate it easily.

Notice that as the number of copies increases, the cost of storage and of updating also increases. But the cost of querying decreases as closer sites become available to send the queries to. We therefore have here a classic tradeoff, which makes the problem of finding the minimal-cost allocation a nontrivial one. A complicating factor, from the computational point of view, is the appearance in the

---

* Although brackets are most commonly used to denote sets in mathematical discussions, we will sometimes use parentheses to indicate an allocation index set. Thus, we might alternatively write I=(1,4,5). We will also refer to I (or to (1,4,5)) simply as an "allocation". This seems easier than to continue to talk about "index sets".

formula of min d_{jk}. This makes the problem seriously nonlinear. The problem of actually determining the minimal cost allocation will be discussed in some detail in a later section.

Extension to multiple files. The cost formula (1) is for a single file. If more than one file is to be allocated, the overall cost is readily obtained by adding the costs of the individual files. This is possible since the model assumes that there is no interaction between usage of the various files. Thus, to minimize the overall cost, one need only find the optimal (minimal cost) allocation for each individual file. We shall see that when constraints are added, or when file usage is not independent, this separation of the multi-file problem into a set of single-file problems may no longer be valid.

The Chu Model

Like Casey, Chu [6,7] assumes that updates must be sent to all copies, while queries go to just one. However, Chu assumes that, if there are r copies of a file, then a fraction 1/r of the queries for that file are sent to each copy. Alternatively, one may think of Chu's approach as averaging over the cost of sending queries randomly to any site which has a copy. Furthermore, Chu assumes that the number r of copies is fixed in advance, generally by considerations of availability. (Relaxing this assumption would, of course, simply require one to consider in turn the cases r=1,2,...,n (where n is the number of sites in the network).)

Chu has some details in his model which appear to make it less flexible than Casey's. For example, he envisions a "transaction" to a file as resulting in a flow of u units of data from the remote site to the user site. Such a "transaction" corresponds roughly to a query in

Casey's model. But then updates are handled by assuming that a certain fraction of these transactions result in the shipment of the same u data units back to the remote site. That is, Chu models the updating process as one of sending for the file (or relevant portion thereof), modifying it, and then shipping it back. This makes it easy for Chu to apply his model to, for example, a multiprocessor with memory hierarchy. However, it is unlikely that updates would be performed in this way in a network environment with a distributed data management system.

There is another limiting feature of Chu's model. In Casey's model, it is easy to include processing costs by adding them into the factors d_{ij} or d'_{ij}. Thus, if we wish to include in Casey's model the cost of querying a local copy, we need only let d_{jj} be that cost. Chu's cost formulation, however, has a built-in limitation that makes local query cost zero. (We will not write down Chu's cost formula here; the interested reader may consult his paper.)

Chu's formulation is of the multiple-file problem. However, his cost function readily separates into a sum of terms, each of them being the cost associated with a single file.

The Levin Model

In his thesis [10], Levin argues that "in order to process a given transaction both the relevant program and the relevant file must be accessed." Thus, a user request originating at one site may have to be sent to a second site (the one containing the application program), which in turn accesses the needed files. Since the optimal location of a file then depends on the locations of the programs which access it, Levin feels that "an optimizing model for file assignments should consider the location of the relevant programs and optimize their assignments simultaneously." This is an attractive idea. However, it does

lead to a model of considerable complexity. In particular, the assumption of file-usage independence is no longer valid when some of those "files" are actually programs which access other files.

Levin's cost formula contains the usual three parts - costs of storage, updates, and queries. However, because of the inclusion of programs and because updates and queries are transmitted through an intermediate program, the cost function ends up with six terms:

Cost = Cost of transmitting queries from initiating sites to the programs
+ Cost of transmitting updates from initiating sites to the programs
+ Cost of transmitting queries from programs to files
+ Cost of transmitting updates from programs to files
+ Storage cost of programs
+ Storage cost of files.

In any model there seems to be a problem with deciding where to transmit the queries. Recall that Casey sends them to the copy that is closest in some sense, while Chu sends them randomly to any site with a copy. Levin's approach is to introduce a new set of variables to describe the "routing discipline"; that is, to which program copy, and from there to which file copy, transactions from a given site are shipped. The values of these routing variables are then to be determined, along with the locations of files and programs, in the optimization process. This approach increases enormously the number of unknown variables in the problem. Furthermore, since one program may access several files - and one file may be accessed by several programs - locations of all files and programs are interdependent, and it is no

longer possible to simplify the problem by decomposing it into a set of single-file allocations.

In order to carry out such a decomposition, and hence make the optimization computationally tractable, Levin makes several simplifying assumptions. In particular, he assumes that storage costs of programs are zero. In a very real sense this eliminates the program location problem, since it then becomes possible to freely place programs wherever it seems useful (and is possible) to do so. Once programs are located, one may readily optimize the routing discipline for any given file allocation. The file allocation itself is carried out by using Casey's suggested search procedure. (This procedure will be discussed later.) It is not surprising that Casey's solution method is ultimately applicable, since, by adding assumptions to make the problem computationally tractable, Levin has simplified his model to the point where it looks very much like Casey's. In fact, Levin's simplified model can be obtained immediately from Casey's model if some precomputing is done to reflect the routing. Each transmission from user to file will have to touch one of the preestablished program sites. Thus, if neither the user site nor the file site contains a copy of the needed program, then Casey's d_{ij} should be changed to $d_{ik} + d_{kj}$, where k is the program site which minimizes that sum. (If copies of necessary programs are at all sites, then Casey's model needs no modification. In a heterogeneous network, however, it may not be reasonable to assume an unrestricted allocation of programs.)

As we have noted, Levin's basic model, before simplifications are introduced, is too complicated to be computationally tractable. However, Levin goes on to map out elaborate extensions of his model. He first notes that the basic model is <u>deterministic</u> and <u>static</u>. That is,

access patterns are assumed to be known, and costs incurred through dynamically changing the allocation are not taken into account. He therefore indicates (1) how the deterministic assumption may be relaxed by treating access patterns as random variables and using forecasting theory, and (2) how the static assumption may be relaxed by including the costs of changing allocations in a multi-period cost function. These are interesting ideas which may be of some importance in future work on dynamic, self-organizing, distributed data systems. In this report, however, we will restrict ourselves to examining the file allocation problem in a static, deterministic environment.

Model for a Primary-Copy Synchronization Scheme

In examining the literature on network file allocation, we find Casey's model the most attractive one. It has a simplicity and flexibility which makes it more readily applicable to a wide variety of situations. In addition, its simplicity provides clear insights into the key features of the problem, which are easily obscured in the more complicated models. We have therefore decided to build on Casey's model in our file allocation work.

When we examine Casey's model in the light of our work on update synchronization [1], we find one serious problem. That model - like all others in the current literature - assumes that the site where the update originates broadcasts the update to all sites holding a copy of the file. Thus the update message flow corresponds to that of Johnson's scheme for synchronization [9]. In that scheme, each site adds a timestamp (its own clock time) to the update and sends it to the copies. We have identified certain disadvantages with such a scheme which led us to study primary-copy schemes. In primary-copy schemes all

updates are sent to a distinguished, or primary, site, which subsequently broadcasts them. We may summarize the difference which this makes in the cost formula as follows.

Casey (broadcast) model:

Cost = Storage

+ Transmission of Queries to Nearest Copy

+ Transmission of Updates to All Copies.

Primary-copy model:

Cost = Storage

+ Transmission of Queries to Nearest Copy

+ Transmission of Updates to Primary

+ Broadcast of Updates by Primary.

The first two terms are the same; it is only the handling of updates that changes. Assuming that the primary is at site 1, we may write (using Casey's notation) the single-file cost function for the primary-copy model as

$$C(I) = \sum_{k \in I} \sigma_k + \sum_{j=1}^{n} \lambda_j \min_{k \in I} d_{jk} + \sum_{j=1}^{n} \Psi_j d'_{j1} + \Psi \sum_{k \in I} d'_{1k} \qquad (2)$$

In this formula, $\Psi = \sum_{j=1}^{n} \Psi_j$; i.e., Ψ is the total update load which the primary must broadcast.

It may appear to the reader that this formula, since it looks longer, yields higher costs than Casey's formula. In looking at examples, we find that this is not true. That is, a primary copy scheme may be cheaper than a timestamp scheme for synchronization. Intuitively, this occurs because a site generating updates need send them only to the primary; and it may be cheaper for the primary to transmit them to the

other copies than it would have been for the updating site to send them directly. (For example, when the parameters d'_{jk} are proportional to distance, the primary site may be closer to the other sites in the allocation than is the updating site.) This interesting result will be discussed in more detail in a later report.

As in Casey's model, multiple files are handled by summing the cost functions for the individual files.

Constraints

It is not always reasonable to minimize the cost function over all possible allocations of copies to sites. In this section we look at various kinds of constraints which one might want to impose.

Forbidden or prescribed sites. The simplest type of constraint is to require that certain sites have a copy of the file. In the WWMCCS environment, certain data bases are currently held at associated sites as a matter of policy and/or convenience. It would be unreasonable to throw such sites into a mindless file-allocation algorithm. In such situations, the algorithm would only be applied to determine the best locations for extra copies. The cost formula would, of course, include the information that a copy is at some prescribed site, since this affects the query cost. Similarly, it may not be feasible to consider putting certain data at some sites. This could be for reasons of security, local system overload, etc. We can then simply omit such sites when we apply the file-allocation procedure. However, if a forbidden site may query the data, this information must be included in the cost formula.

Prescribed number of copies. As was previously noted, Chu's cost function includes the number of copies as a parameter. Chu [7] suggests choosing this number on the basis of availability. A simple

model can be used to determine the number of copies that are needed to yield a given level of availability. In many cases, two copies are adequate. If one site is prescribed, then one need only determine which of the (n-1) sites remaining is the best location for the extra copy. Given the cost function, it is a simple matter to carry out this determination.

One could also assume that availability considerations set a lower bound on the number of copies needed. This will increase the number of possible allocations to be investigated, but may lead to a lower-cost allocation. (Adding sites can decrease total cost when query transmission costs are large relative to storage and update costs.)

There is one problem with letting a simplified availability model determine the number of copies. A correct, complete availability analysis for a real network would involve such things as the detailed network topology, a combinatorial analysis of relevant "cutsets" and their probability of failure, etc. The number of copies r that are needed to satisfy an availability requirement would then in general vary with copy location. And availability would not, of course, be independent of the user's site. Some sort of average or minimal availability would have to be used in the constraint. One could envisage an iterative process. A rough estimate of availability could be used to determine a number of copies. An allocation could then be determined and expected availability computed for that specific allocation. If this availability is too low, the number of copies could be increased. This is an attractive approach which, as far as we know, has not been previously proposed.

It may also be that one wants to set a maximum number of copies. However, the most obvious basis for doing this would be that of

cost, and the allocation optimization process automatically selects the most economical solution. Thus, it is not necessary (though it may save computation time) to announce a priori that all allocations of more than four copies (say) are "too expensive". Indeed, without extensive computations of multi-copy allocation costs it may be impossible to verify such a pronouncement.

Storage constraint. It may be necessary to take into account limited storage capacity at the sites. That is, it may not be physically possible to store all the files that one might want to at a given site. In his model, Chu introduces storage constraints of the form

$$\sum_{j \in F_i} L_j \leq S_i \qquad (i = 1,2, \ldots, n),$$

where F_i is the set of files to be stored at site i, L_j is the length of the jth file, and S_i is the storage capacity at site i.

Adding such a constraint has an unfortunate computational effect. It makes it necessary to consider the allocations of all of the files simultaneously. It is no longer possible to allocate each file in turn, since the ultimate total file distribution is likely to violate one or more storage constraints.

Including storage constraints may seem to be a necessity. However, this is not necessarily the case. It may be more reasonable to assume that additional storage space can be purchased as needed. The cost of storing a file at a given site can reflect "new storage" cost as well as "old storage" cost. Indeed, unless some sites have a lot of unused storage capacity which management elects to think of as "free", there seems to be no good reason to consider newly purchased storage capacity to have a cost basis different from the old.

Upper bound on response-time. Chu [6,7] also includes in his model the condition that "the expected time to access each file is less than a given bound." To compute this expected time, Chu makes a number of simplifying assumptions. The access process is modeled as follows.

1. The user request, after waiting in a queue for a network channel, is transmitted to a remote site that holds a file copy.
2. The remote site accesses the file.
3. The response is put in the queue for a network channel, and is ultimately sent back to the user.

Chu assumes that request messages are relatively short, so that they can be given a high priority and their queueing delay can be neglected. With this and other simplifications, Chu can treat the entire access process as a straightforward single-server queueing system. Response time becomes a function of line capacity, line traffic, and request rates to the various sites. Request rates to sites and line traffic are in turn dependent upon the file allocation and the request rates to the files.

The important thing to notice about a response-time constraint is that it inherently includes all of the files in the system and their locations. This is because line traffic and response delay due to line congestion must take into account transactions involving all of the files. Not being able to decompose the multi-file problem into a set of single-file problems increases the computational intractability of the optimization.

Other constraints. The four types of constraints discussed above are those which one sees in the literature. There are other

possibilities which come to mind. For example, limited channel bandwidth between two sites may make it necessary to set an upper bound on the traffic between the two sites. Although limited bandwidth is taken into account in Chu's response-time model, a more direct imposition of a constraint on traffic might be simpler and more appropriate in some situations.

As another example, one might sometimes want to take into account the limited processing capacity at certain sites. Including terms for processing costs in the cost function can help to eliminate the need for putting a constraint on processing time. For example, the "cost" of sending queries or updates to an already overloaded site could be set at such a large figure that no additional constraint is needed to effectively eliminate that site from the allocation. To take into account more subtle effects of processing capacity limitations, one would have to develop a realistic model for the processing needed for the query and update operations.

Finding the Optimal Allocation

Zero-One Programming

In Chu's formulation [6,7] of the file allocation problem, the variables defining an allocation are

$$X_{ij} = \begin{cases} 1 & \text{if file } j \text{ is at site } i, \\ 0 & \text{otherwise.} \end{cases}$$

The cost function is then readily written as a quadratic function of these variables. By increasing the number of variables in the problem, Chu is able to reformulate his cost function as a linear combination of variables that take on the values zero or one. Minimizing the cost then amounts to solving a zero-one linear programming problem. In general, solving such problems is computationally very expensive. On many occasions heuristics are used, sacrificing optimality for computational efficiency. For example, Levin [10] points out that Chu's approach would produce about 9000 zero-one variables with 18000 constraints for an ARPA-like network with 30 sites and at least 10 files. Although there are computational difficulties for large systems, it should be emphasized that this approach is mathematically straightforward. In addition, it is relatively easy to formulate constraints in terms of the zero-one variables. The constraints are also handled automatically by algorithms to solve the zero-one programming problem. This is how Chu may readily add response-time and storage-capacity constraints to his model.

Searching the Cost Graph

Casey's cost graph. Finding the optimal allocation can be simply thought of as a process of computing and comparing the costs of all allowable allocations. The set of possible allocations for a

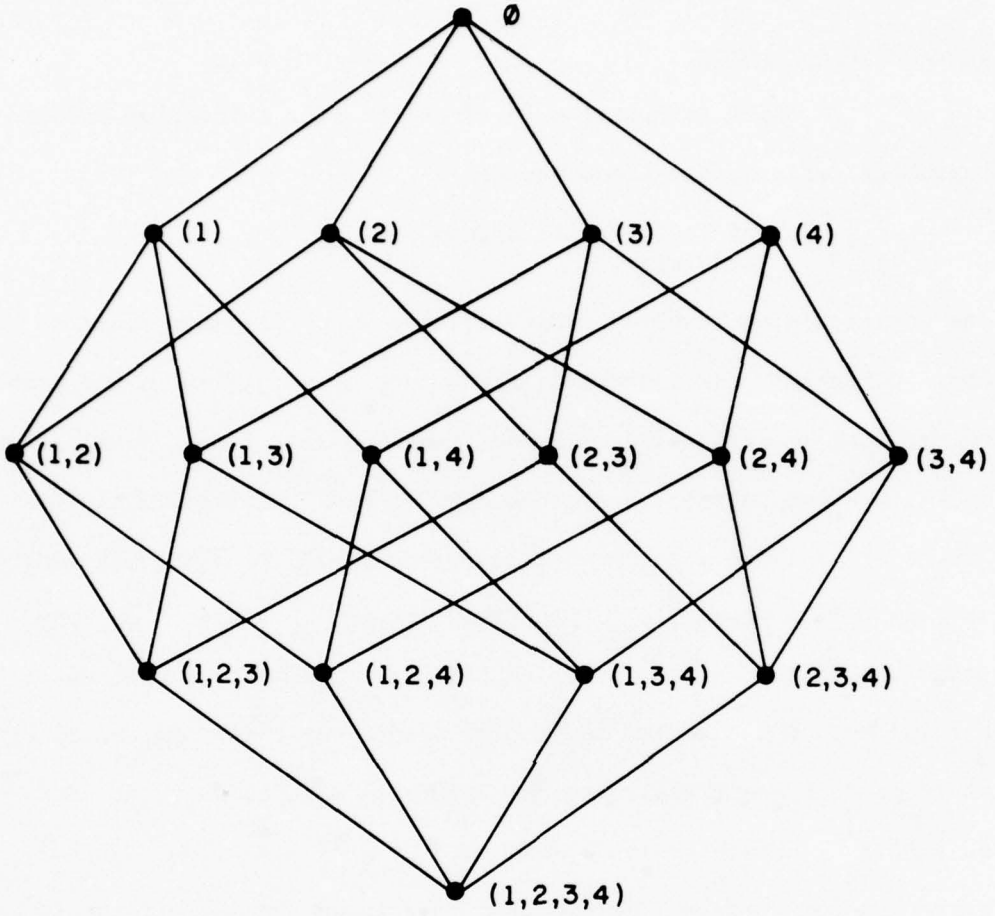


Figure 1

Graph of all possible allocations
among four sites.

single-file, unconstrained problem can be pictured in graphical form. Figure 1 shows such a graph for four-site allocations. For symmetry, the top node denotes the null allocation ($\emptyset$), in which no sites have a copy. At level i in the graph we have all the allocations of i copies (each node representing an allocation). A branch is drawn from a node at level i to one at level i+1 if and only if the latter allocation can be obtained by adding a single copy to the former. If we associate the cost of each allocation with each node, we have what Casey [4] calls the cost graph. In the general n-site graph, the ith level contains $\binom{n}{i}$ nodes. The total number of nodes in the graph (including the null allocation) is 2^n. Each site added to the network therefore doubles the number of costs that must be computed, in general.

Search procedures. Casey [4] suggests that the minimum cost node be located by carrying out a systematic search through the cost graph. In Casey's words, "A computer algorithm can be implemented in several different ways to select file nodes [sites] one at a time up to the optimum configuration. One approach is to follow all paths in parallel through the cost graph, stepping one level per iteration. This method is computationally efficient, but may require a great deal of storage in a large problem. Alternatively, a program can be written to trace systematically one path at a time. Such a technique uses less storage, but may require redundant calculations since many different paths intersect each vertex". In general we agree with this statement, but we dispute the last sentence. We believe that no node must be visited more than once in a "path at a time" approach. We will shortly defend this belief.

First, let us make more specific the memory requirements of the level-by-level approach. At least $\binom{n}{n/2}$ memory locations (for n

even) are needed to save the cost values of the previous level in the cost graph. (This maximum occurs at the middle level (i=n/2) in the cost graph.) Some additional working space is needed for the evaluation of the current level. Note that $\binom{n}{n/2}$ is 184,756 for n=20 and 155,117,520 for n=30. As networks increase in size, the memory requirement of a level-by-level search quickly becomes prohibitive. This makes it important to devise a path-at-a-time search in which effort is not wasted.

Theorems to truncate the search. In the worst possible case - when the optimal allocation is to put a copy everywhere - all nodes of the cost graph must be evaluated in any search procedure. Usually, however, one can truncate the search. Casey presents the following two theorems for doing so. The reader should refer back to the description of Casey's model for definitions of the notation.

Theorem C1. Let $d_{jk}=d'_{jk}$ for all j,k. If for some integer r ($\leq$n), $\Psi_j>\lambda_j/(r-1)$ for all j, then any r-site file assignment is more costly than the optimal one-site assignment.

Theorem C2. Suppose assignment I is optimal. Then along any path in the cost graph from the null node to the node corresponding to I, cost is monotonically non-increasing from one node to the next.

Actually Casey's Theorem C2 is somewhat more general and is not stated in terms of the notions of "paths" and "nodes". These notions belong to the cost graph visualization of the optimal allocation problem. (See figure 1.)

Theorem C1 gives us an upper bound (r) on the level beyond which there is no need to search; i.e. a gross stopping criterion. To see how this theorem may reduce our search, notice that Ψ_j/λ_j is the

ratio of updates to queries generated at site j. If for all sites the volume of updates is, say, more than half that of queries, then the inequality in theorem C1 holds with r=3, and we know that we need not search below level 2 in the cost graph.

Theorem C2 gives us a more precise stopping criterion. If, while following a path down the cost graph, we find that the cost increases, then no further search along this path will be of any use.

An efficient path search. We have devised a scheme for searching the cost graph one path at a time, but without revisiting nodes. The reader can probably understand the algorithm best from an example. The search order for four sites goes:

(1), (1,2), (1,2,3), (1,2,3,4), (1,2,4),
(1,3), (1,3,4), (1,4),(2), (2,3), ..., (4).

More precisely, the following algorithm will print out the search order for n sites. Let V(i) be an array with index i. Let the instruction "print" be such that it will print the values of V(1) to V(k) contiguously.

1. V(1)=1; k=1; print.
2. If V(k)≠n, then k=k+1; V(k)=V(k-1)+1; print; go to 2.
3. Else k=k-1;
   If k=0, then halt.
   Else V(k)=V(k)+1; print; go to 2.

What we have actually done is to convert the cost graph (figure 1) to a tree (figure 2). Our algorithm follows a preorder search of this tree.

As the reader will observe, there are many paths that lead to the node (1,2,3,4) in figure 1, but only one in figure 2. There remains

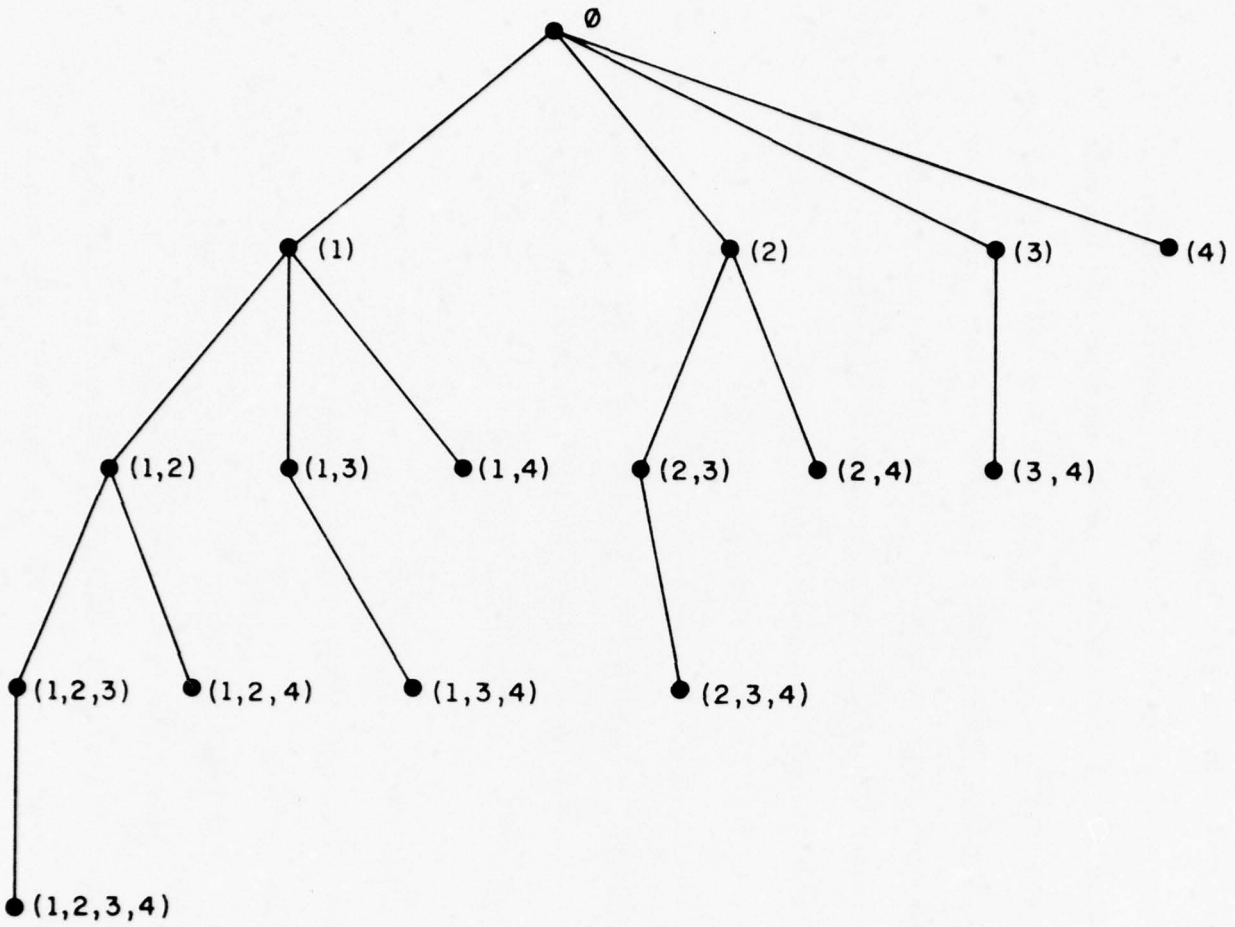


Figure 2

Rearrangement of four-site allocations (figure 1)
into a tree for efficient preorder search.

an important question. Is it possible that by following only one path to a given node we could fail to recognize the optimum? The answer to this question is "No". In order for a node to have minimum cost, it must, by definition of minimum, have a cost no greater than that of any other node, in particular than that of any node in the previous level. This fact, in conjunction with Casey's theorem C2, produces the condition that cost is nonincreasing along all paths leading to the optimum node. Thus, our simplified tree of figure 2 contains one such path that will lead to the optimum. We therefore conclude that our restricted search is sufficient to detect the optimum.

As the reader will realize, our algorithm requires in the worst case (i.e. along the longest path) only one memory location per level, or n memory locations. In implementing the algorithm, we find that some efficiency can be gained by increasing the memory requirement to $O(n^2)$. However, it is clear that we have achieved our goal. Memory requirements are much more reasonable than that of the level-by-level search. But at the same time we have preserved computational efficiency by ensuring that each node cost is computed only once.

Searching with constraints. The search algorithms described above have been for the unconstrained cost minimization problem. If constraints are imposed on the single-file problem, the effect is to cut down on the number of nodes of the cost tree which must be included in the search. This may or may not be helpful. Casey's theorem C2, for example, applies only to the overall minimum of the cost graph. More extensive computing, therefore, may be necessary to identify the minimum cost over a restricted set.

If constraints (e.g., response time or storage) are imposed which couple the allocations of several files, the cost graph expands to

include all possible allocations of all files. This expanded graph has 2^{nf} nodes, where f is the number of different files in the system. One arrives at this number of nodes by noting that (in Chu's notation) the problem has nf variables X_{ij}, each of which may take on the value zero or one. Casey's theorems for truncating the search are clearly no longer applicable. It is hard to see how one could avoid the tedious process of checking all allocations to see which ones satisfy the constraints, and evaluating the costs of all that do. The alternative is to use Chu's formulation and a zero-one programming algorithm. It appears to us that Chu's approach, being straightforward, is the best one to take if interactions between the allocations of different files must be taken into account.

The best allocation for a primary copy scheme. The search procedure needed to find the optimum allocation for the primary copy model is much like that for a constrained problem in which a copy is prescribed at one site. That is, given a location for the primary, we search that portion of the cost graph covering all allocations that include the primary site. This determines the best allocation with that particular site for the primary. By trying each site in turn as the primary, we can find the lowest cost of all and hence the best site for the primary.

Local minima. In many cases, it is not feasible to search a large part of the cost graph for a true global minimum. One must content oneself with a local minimum - i.e., an allocation which is less (or no more) costly than any "neighboring" allocation.

As one such procedure for finding a local minimum we have tried the following. Let the sites be numbered 1,2,3,... in any arbitrary order. (In the next subsection, we will discuss a good strategy

for choosing this order.) Begin by computing the cost of allocation (1); i.e. one copy at site 1. Then proceed cyclically through the list of sites, testing the effect of a change in allocation state (i.e., whether or not the site has a copy) of each site in turn. If the cost is decreased, make the change; otherwise proceed to the next site. Continue in this way until a local minimum is reached. A local minimum is found if we are able to make a complete cycle of the list of sites without any cost improvement. That is, if we have a node which represents a local minimum, then all adjacent nodes will have a higher cost.

An example might be helpful. After computing the cost C(1) of allocation (1), we next compute C(1,2). If this is larger than C(1), we next try C(1,3), etc. Suppose that C(1,4) is less than C(1). We then use (1,4) as a new prospective local minimum, and test the neighboring allocations (1,4,5), (1,4,6) ... (1,4,n), (4), (1,2,4), (1,3,4). If none of these has a lower cost than (1,4), then (1,4) is a local minimum. It may help to visualize the process to note that neighboring nodes are defined in this algorithm as those joined to the given node by a branch of the cost graph. (See figure 1.)

The reader might feel that, for example, allocation (1,3) should be considered as neighboring (1,4). But notice that these differ by two changes in allocation state - deletion of the copy at 3 and addition of the copy at 4. Thus these allocations are said to be a distance 2 from each other. It is quite possible to define a local minimum as being of lower cost than all allocations within distance 2 - or possibly within a farther distance. This simply increases the search required to locate and verify a local minimum. The usual sort of trade-off is involved here. By increasing the cost of the search, one can generally obtain a lower cost allocation. Indeed, we find that for many

examples a local optimum that has lower cost than all allocations within distance 2 is actually a global optimum. Chandy and Hewes [5] also report that such a "2-distance heuristic ... is a near optimal heuristic."

Simplifying the Problem

No matter how efficient a search procedure is devised, it remains true in general that the problem of finding the optimum grows like 2^n, where n is the number of sites. Indeed, a rigorous proof to essentially this effect has recently been presented by Eswaran [8]. (Technically, Eswaran proved the problem "polynomial complete".) Let us see what this really means in terms of computation time. Casey [4] notes that a program run in the 360/91 took less than 10 seconds to solve six different optimal allocation problems for a network of 19 sites. (This time included the Fortran compilation.) Thus, if we assume that it takes about 2 seconds to optimally allocate a file in a 19-site network, and every additional site doubles the time, then it will take over an hour in a 30-site network, about 48 days in a 40-site network, and about 136 years in a 50-site network. It is therefore worthwhile to take other approaches to trying to cut down on the amount of work.

In one such approach, Urano et al. [13] have shown that the sites may be partitioned into sets of "neighboring" sites in such a way that at most one member of each set should be included in an optimal allocation. (The geographic interpretation of their result is based on visualizing the d_{ij}'s as distances and was discussed in more detail in the Introduction.) More recently, Chandy and Hewes [5] have reformulated what is essentially Casey's model as a simple linear integer programming

problem. They find that solving the problem by a standard linear programming algorithm (without the integrality constraints) leads in many cases to correct integer solutions. (The same phenomenon has been noted in similar optimization problems [12], and hence appears to be more than just an artifact of the particular examples tried.)

New results on a priori exclusion or inclusion. We have attempted to decrease the size of the problem by a new approach; namely, by deriving rules for determining a priori that certain sites will (or will not) be included in an optimal allocation. Three theorems providing rules of this sort are presented in this section. Formal proofs of the theorems may be found in appendix 2. Casey's model and cost formula are assumed. (See equation (1), p. 17.) For simplicity, we shall also assume that $d'_{jj}=d_{jj}=0$ for all j. Thus we neglect the costs of queries and updates applied locally. Relaxing this assumption, however, as well as making other minor alterations in the formulation, requires only small compensating alterations in our theorems.

Our first theorem essentially states that if the cost of having a local file copy is smaller than the smallest possible cost of sending the locally generated queries elsewhere, then a local copy should unquestionably be included in the optimal allocation. The cost of storing and updating a copy at site i is an important quantity; we shall denote it by Z_i. That is, we let

$$Z_i = \sigma_i + \sum_{j=1}^{n} \psi_j d'_{ji}.$$

Theorem 1. All optimal allocations will include site i if

$$\lambda_i \min_{j \neq i} d_{ij} > Z_i. \qquad (3)$$

Our second theorem gives the other side of the coin. If it costs more to maintain a local copy than we can possibly save by having one, then we do not want one.

Theorem 2. No optimal allocation including more than one site will include site i if

$$Z_i > \sum_{j=1}^{n} \lambda_j (\max_k d_{jk} - d_{ji}). \qquad (4)$$

Theorems 1 and 2 can be summarized as follows. Let

$$m_i = \lambda_i \min_{j \neq i} d_{ij}, \text{ and}$$

$$M_i = \sum_{j=1}^{n} \lambda_j (\max_k d_{jk} - d_{ji}).$$

Then for each i the real line is partitioned by m_i and M_i into three regions, as shown in figure 3. If Z_i falls in region 1, then it should unquestionably be included in any optimal allocation. If it falls in region 3, it will generally be excluded. An exceptional case is when all sites fall in region 3; i.e., satisfy inequality (4). In this case, all optimal allocations will be single-copy ones, and we need only choose the best of these. Sites whose costs Z_i fall in region 2 (including the boundary points) must be considered further.

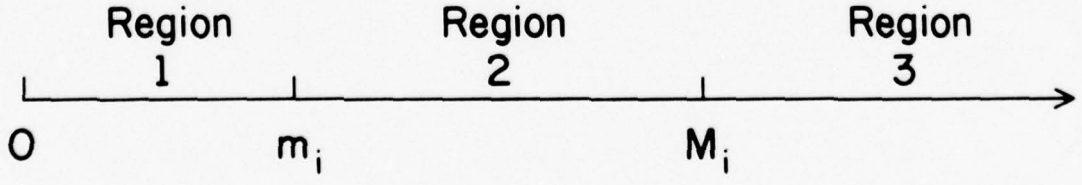


Figure 3

One other comment should be made. If network transmission costs are independent of the sites involved, so that $d_{ij}=d$ for all $i\neq j$, then $m_i=M_i=\lambda_i d$, and region 2 collapses to a single boundary point. Unless the cost Z_i of some site lies on the boundary, every site is *a priori* either included or excluded. Immediately we see that then the set of sites in region 1 forms the optimal allocation, unless the exceptional case noted above obtains. Computation is therefore reduced to a minimum (essentially n cost evaluations) for many real network environments. (For example, most commercial packet-switched networks charge per packet, irrespective of the distance the packet is to be sent.) There are, of course, real cost differentials incurred in the longer lines between sites, but from the user's point of view it is the network charging policy that matters.

Our third theorem is somewhat in the spirit of the approach to simplification taken by Urano et al. [13]. However, Urano's results depend on a more complex model, involving "relay costs" at network nodes and restrictive hypotheses on the d_{ij}. Also, instead of determining that no more than one of some group of geographically close sites can be included in an optimal allocation, we show that certain sites may *a priori* be precluded from being in an optimal allocation by the existence of a "better" site nearby. Thus, again we have a theorem which allows us to eliminate initially certain sites from any further consideration in the search for the optimum.

Theorem 3. A site i cannot be included in any optimal allocation if there exists another site k in the network such that

$$Z_i - Z_k > \sum_{j=1}^{n} \lambda_j (d_{jk} - d_{ji})_+ \qquad (5)$$

where $(f)_+ = \begin{cases} f \text{ if } f \geq 0, \\ 0 \text{ if } f < 0. \end{cases}$

Examples and further discussion. In order to assess their value, we have applied our theorems to several of the examples in Casey's paper [4]. Casey assumes throughout that it costs the same to transmit a query as an update ($d'_{ij}=d_{ij}$), that transmission costs are symmetric ($d_{ij}=d_{ji}$), and that storage costs are zero. In Casey's 5-site example, the data are as follows. The matrix of transmission costs d_{ij} is

$$\begin{bmatrix} 0 & 6 & 12 & 9 & 6 \\ 6 & 0 & 6 & 12 & 9 \\ 12 & 6 & 0 & 6 & 12 \\ 9 & 12 & 6 & 0 & 6 \\ 6 & 9 & 12 & 6 & 0 \end{bmatrix}$$

All λ_j's are 24, and the vector of Ψ_j's is (2,3,4,6,8). In order to apply theorems 1 and 2, we tabulate Z_i, m_i, and M_i for all sites i.

Site	1	2	3	4	5
Z_i	168	180	174	126	123
m_i	144	144	144	144	144
M_i	648	648	576	648	648

Theorem 1 then states that both sites 4 and 5 should be included in any optimal allocation. Theorem 2 does not eliminate any sites, so the effect of adding sites 1,2,3 would have to be checked. However, the problem has been reduced from an optimization among 32 ($=2^5$) choices to one among 8 ($=2^3$). (The optimal allocation I is (1,4,5).) It is virtually obvious by inspection that costs (Z_i) are sufficiently similar and query loads and transmission costs sufficiently large that theorem 3 will not be useful. More will be said about this later.

As another example of the usefulness of theorem 1, consider Casey's 5-site example reduced to four sites by the elimination of site 5. The table of Z_i, m_i, and M_i is then

Site	1	2	3	4
Z_i	120	108	78	78
m_i	144	144	144	144
M_i	504	576	576	504

Since for all i $Z_i < m_i$, theorem 1 says that the optimal allocation consists of all four sites, and no further computation is necessary.

In another of Casey's examples - the 19-site ARPA network example with d_{ij} equal to distance - theorem 3 does turn out to be useful. This is what is to be expected in a geographically widespread network in which there are groups of sites that are very close together (e.g., in the Boston and Los Angeles areas). We will not give the details (the interested reader may consult the data and notation in Casey's paper), but we find that theorem 3 eliminates six sites. Specifically, sites 3, 4 and 5 are excluded by site 2, site 15 by site 14, and sites 17 and 19 by site 18. This reduces by a factor of 64 the maximum size of the search required to find the optimum.

The reader might question whether it is worthwhile, in general, to carry out the computations needed to check the conditions for our theorems. In this connection, we point out that the costs Z_i are needed in any cost optimization algorithm and it only increases efficiency to pre-compute them. Computation of m_i for i=1,...,n is easy, requiring $O(n^2)$ operations altogether (n to find the minimum d_{ij} for each i). Application of theorem 1 is therefore highly advisable, since for each

site successfully identified as satisfying inequality (1) the search is reduced by 50 percent. Furthermore, it may happen that the set of sites satisfying (1) forms an acceptable sub-optimal allocation, so that no further search is necessary.

Computation of M_i is more time-consuming, requiring $0(n^2)$ operations for <u>each</u> i. It may be that one would only want to try theorem 2 selectively, say when Z_i (or Z_i/λ_i) is "big". Testing for theorem 3 is even more time-consuming because of the dependence on k. However, the quantities involved in the test (the right sides of inequality (5)) are sufficiently similar to M_i that it is not difficult to work out an efficient scheme for carrying out the computations needed for theorems 2 and 3 together. Again, to save time one can readily eliminate from the process many cases where checking inequality (5) is (or is likely to be) useless. For example, when $Z_i < Z_k + \lambda_i d_{ik}$, (5) clearly cannot be satisfied. This simple condition suffices to show conclusively that theorem 3 is of no help in the 5-site example discussed above.

Our intuitive idea of theorem 3 - the elimination of more expensive, neighboring sites by the existence of a given site - can be used to identify situations when theorem 3 is likely to be useful. Thus, if a number of network sites are clumped together (in the sense of small d_{ij}), then the "cheapest" of these (small Z_i/λ_i) may serve to eliminate others of the clump. Notice also that the minimization carried out in the test for theorem 1 identifies the "closest" site to site i in the sense of smallest d_{ij}. This information can be saved and used to identify promising pairs for the theorem 3 test.

Finally we note that all of the results and comments of this section will hold for the primary-copy model (equation (2)), provided that we redefine Z_i as

$$Z_i = \sigma_i + \Psi d'_{1i}.$$

This is intuitively reasonable. The cost of shipping all updates to the primary is the same for all allocations and hence has no effect on the cost comparisons. Therefore the appropriate cost of maintaining a copy is just the sum of storage cost plus the cost of shipping all updates to that copy from the primary.

The quantity Z_i/λ_i appeared above as a measure of the cheapness of site i. Since an optimal allocation is likely to include the cheaper sites, this quantity is also a rough measure of the probability that site i will be included in an optimal allocation. We have successfully used Z_i/λ_i as a heuristic "search factor" to expedite search procedures for optimal or near-optimal allocations. For example, in the search for local minima which we described earlier, we have obtained considerable improvement by numbering the sites 1,2,... in order of increasing Z_i/λ_i. That is, the use of this search factor generally seems to lead one to a better (i.e. lower cost) allocation with less effort than if the sites are numbered arbitrarily.

If constraints are imposed on the problem, then our theorems may not be immediately applicable. If there are constraints excluding or including sites, or setting a minimum number of copies, our theorems are still useful as long as they are not applied mindlessly. Thus, one must make adjustments if theorem 1 would include some of the wrong sites - or too many sites - or if theorem 2 excludes too many sites. In some cases, one might want to consider the proofs of the theorems

(see appendix 2) in determining how they might be applied, since the proofs provide the basic cost comparisons which follow from the theorem hypotheses.

For example, suppose we wish to find the best 2-site allocation, and one copy is required to be at site 1. Then it may be possible to exclude a number of sites from consideration as the second site by applying lemma 1 (appendix 2) with k=1. If all sites are "excluded" by this lemma, one is reduced to brute-force comparisons of the possibilities. On the other hand, theorem 1 may show that some site - say site 2 - should appear in any globally optimal allocation. Then I=(1,2) is certainly a "good" allocation, and application of lemma 2 with k=2, i=3, ..., n, may in fact serve to prove that (1,2) is the best 2-site allocation containing site 1.

The example above is, of course, very simplistic. Application of the theorems in this case is likely to save little over the brute-force cost computations and comparisons. The example is only designed to show the reader the kinds of variations on our results which are obtainable for application in some non-standard situations.

Constraints may take on many forms, and for some forms it may not be clear that anything like our theorems can be derived. We believe, however, that for any file allocation problem it is worth investigating the possibility of deriving as much a priori information on the optimum as possible. The savings may well be enough to turn an impractically large computing problem into one that is of acceptable size.

Applications to Special Situations

In addition to altering Casey's model to handle a primary-copy synchronization scheme, we have also, following another line of research, developed models for certain special situations which may be particularly relevant to the WWMCCS environment. In this section we report on two such studies.

First, we consider the optimal allocation of a file that is updated by a single site (the site that generates the updates by its data collection activities), but is queried by other sites for its information. We will initially consider this situation under the assumptions of a uniform query load (i.e., each site generates the same number of queries) and a homogeneous environment. We then relax the homogeneity condition. An interesting feature of this study is that a partial (and sometimes complete) solution to the file allocation problem may be obtained by inspection of simple inequalities. The lengthy computation and comparison of multitudes of allocation costs are just not needed in many simple situations.

Second, we consider the tradeoffs found in the local caching of data (as might occur in an intelligent terminal application.) Here we will consider a local system that maintains a partial copy of a file or data base. We assume that the system can satisfy some of its queries locally and the rest by using the remote copy. We then derive the conditions that must be satisfied for this situation to be cost-effective.

We will slightly modify Casey's model by assuming that the costs d_{jk} and d'_{jk} are the sums of two components - system (processing) cost and a constant cost of transmission. Assuming a constant cost of

transmission seems reasonable, since most packet-network charges are independent of distance. Including processing costs will allow us to investigate effects of system load on the problem. Thus we will assume that

$$d'_{jk} = \begin{cases} u_k & \text{for } j=k \\ u_k+N_u & \text{for } j\neq k \end{cases}$$

and

$$d_{jk} = \begin{cases} q_k & \text{for } j=k \\ q_k+N_q & \text{for } j\neq k \end{cases}$$

where u_k is the cost of processing an update on the kth system,

N_u is the total network cost to transmit an update,

q_k is the cost of processing a query on the kth system, and

N_q is the total network cost to transmit a query.

Single Update Source

In this section we wish to determine the optimal allocation of a file which is updated by only one node but queried by all. To represent this situation, we will assume that all updates originate from the node j=1. (In other words, $\Psi_1 \neq 0$ and $\Psi_j = 0$ for $1 < j \leq n$.) Initially we will assume that $q_k=q$, $u_k=u$, $\sigma_k=\sigma$ and $\lambda_k=\lambda$ for all k, and that $N_u=N_q=N$. This last condition is not really necessary, but will make the results somewhat clearer. Thus our homogeneity assumptions include equal storage costs, query loads, and query and update processing costs. Under these assumptions we obtain the following.

Strategy Rule 1. [homogeneous network]

If $\Psi_1(u+N) + \sigma > \lambda N$,

then the cheapest allocation is to put a single copy at site 1.

Intuitively, this result says that if it costs more to store and update the copy at site k than it does to send k's queries to site 1, then it does not pay to put a copy at site k. A rigorous proof of this result is in appendix 3.

Now suppose that we introduce some heterogeneity into the system. Assume that for some site k in the network, $u_1 \neq u_k$, $\sigma_1 \neq \sigma_k$, and $q_k < q_1 < q_k + N$. (This last inequality ensures that locally generated queries will be processed locally.) When will C(k) be less than C(1)? Straightforward computation yields

$$C(k) - C(1) = \Psi_1(N + u_k - u_1) + n\lambda(q_k - q_1) + \sigma_k - \sigma_1.$$

From this we immediately conclude:

Strategy Rule 2 [heterogeneous network].

If some site k is so much cheaper than site 1 that

$$\Psi_1 u_1 + n\lambda q_1 + \sigma_1 > \Psi_1(N + u_k) + n\lambda q_k + \sigma_k,$$

then it is cheaper to put a copy at site k than at site 1, even though site 1 is generating all the updates.

Finally, we might ask when, in a heterogeneous network with best single-site allocation (1), it is worthwhile to add a second copy. We will first assume $q_k < q_1 < q_k + N$ and compute

$$C(1) - C(1,k) = \lambda N + (n-1)\lambda(q_1 - q_k) - \sigma_k - \Psi_1(u_k + N).$$

From this we obtain

Strategy Rule 3 [heterogeneous network]

If site k is sufficiently cheap that

$$\sigma_k + \Psi_1(u_k + N) < \lambda N + (n-1)\lambda(q_1 - q_k),$$

then the two-site allocation (1,k) is cheaper than allocation (1).

If q_k is still smaller, so that $q_k+N<q_1$ (which makes it economical for even site 1 to send its queries to site k), then the condition in strategy rule 3 becomes

$$n\lambda(q_1-q_k) > \psi_1(u_k+N)+\sigma_k.$$

Notice that these last two rules do not give us the complete answer to the file allocation problem. For example, rule 3 says that we can save by putting a second copy at site k, but it does not select among several alternative sites which may satisfy the inequality, nor does it preclude greater savings from adding a third copy.

Use of a Local Partial Copy

In this section, we investigate the situation that would exist when an intelligent terminal acts as a front-end to a distributed data management system. Specifically, we consider the case where some queries can be satisfied by a local partial copy of a data base, while the rest must be answered by querying a remote copy. We will compare the cost of this two-copy allocation (one complete copy and one partial copy) with that of a single remote copy. We assume that all updates originate at the remote site (site 1), and that all queries originate at the local site (site 2).

Let us also assume that the storage cost of the partial copy is $p\sigma_1$ where p is less than 1. Further, let

ψ_1 = the volume of updates relevant to the remote copy but not to the local (partial) copy, and

ψ_2 = the volume of updates relevant to the local copy.

Then ψ, the total update volume, is $\psi_1+\psi_2$.

Let λ_1 = the query volume that must be sent to the remote copy, and

λ_2 = the query volume that can be satisfied locally (i.e. by the partial copy).

With these assumptions, the cost of having only the remote copy is

$$C(1) = \Psi u_1+(\lambda_1+\lambda_2)(q_1+N)+\sigma_1.$$

Adding the partial copy gives

$$C(1,2) = \Psi u_1+\Psi_2(u_2+N)+\lambda_1(q_1+N)+\lambda_2 q_2+(1+p)\sigma_1$$

Therefore

$$C(1)-C(1,2) = \lambda_2(q_1-q_2+N)-\Psi_2(u_2+N)-p\sigma_1,$$

and the local partial copy becomes cost-effective when

$$\lambda_2(q_1-q_2+N) > \Psi_2(u_2+N)+p\sigma_1.$$

There is one immediate point to be noted here. If $q_2>q_1+N$, so that the local cost of processing queries is greater than that of sending them across the network and processing them at the primary site, then it never pays to have the local copy. This observation increases the importance of designing intelligent terminals to be as inexpensive as possible.

Another interesting observation may be made. Let the query cost differential q_1-q_2+N be denoted by Q, and suppose that the locally cached data is relatively static, so that $\Psi_2=0$. Then the condition for cost-effectiveness becomes

$$\lambda_2 Q > p\sigma_1.$$

To visualize this condition, in figure 4 we have plotted crossover points. Each curve is for a fixed value of $p\sigma_1$, and gives values of (λ_2,Q) such that $\lambda_2 Q=p\sigma_1$. On the graph the units for Q are dollars per query, of λ_2 are queries per hour, and of $p\sigma_1$ are dollars per month. Because of the conversion factors among these units, the actual quantity graphed is $\lambda_2=(p\sigma_1/720Q)$. For example, if the differential querying cost Q is \$0.10, and the cost $p\sigma_1$ of the local storage cache is \$10 a

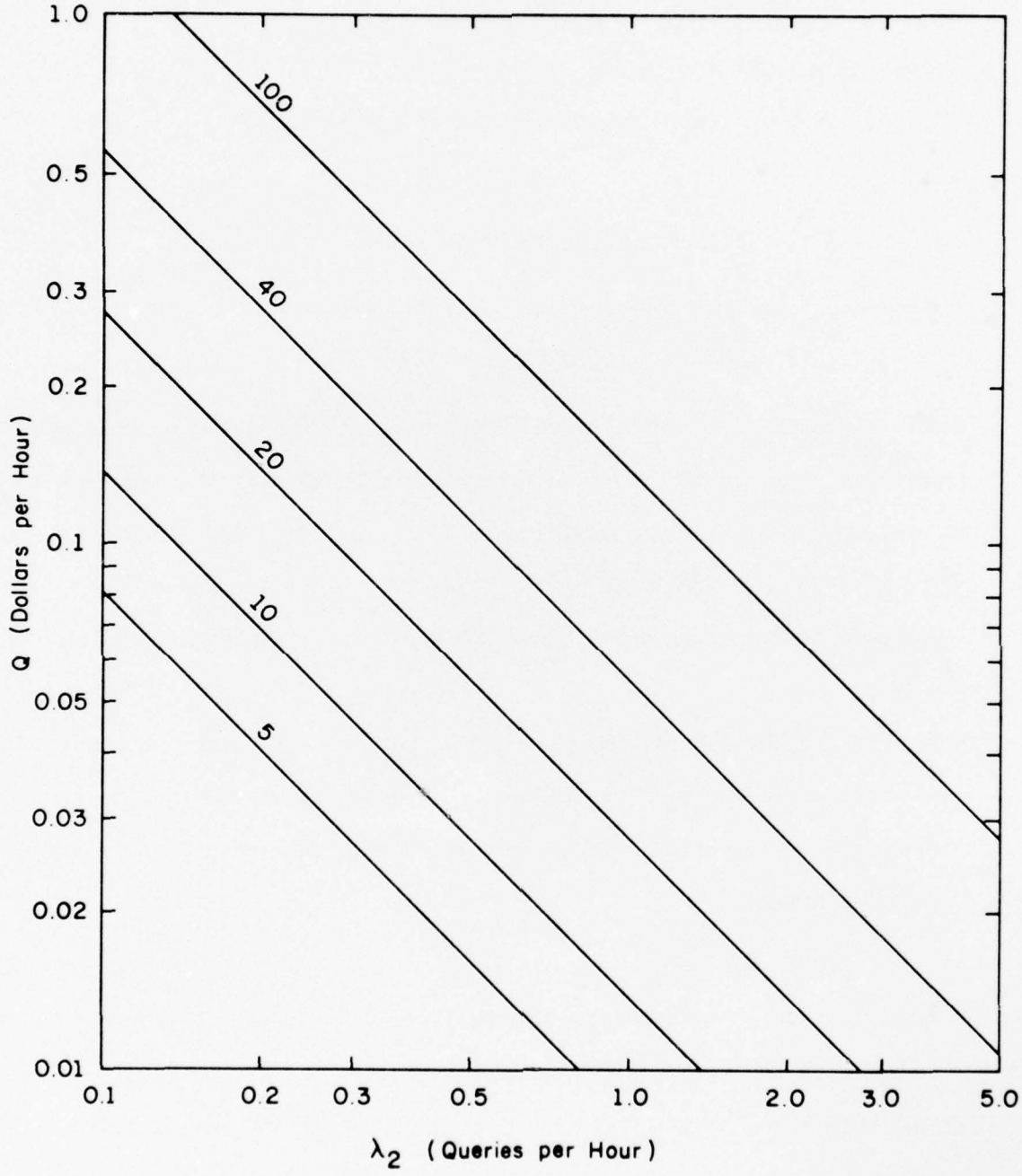


Figure 4

Minimum query rates which make a local cache cost effective. Q is the savings per query caused by having the local cache. Curves are labeled with storage cost of the local cache (in dollars per month). See text for detailed assumptions.

month, the crossover point is at $\lambda_2 \approx 0.14$ queries per hour. For any query rate over this, the local cache becomes increasingly cost-effective. If the differential querying cost Q is only \$0.01; i.e., decreases by a factor of 10, the crossover point increases by the same factor of 10, to 1.4 queries per hour.

There are many cost factors which enter into the quantity Q. Careful analysis and/or measurements would have to be carried out to get a good estimate of Q in any given situation. This very simple application of file allocation points up a basic problem. Good data on costs and usage patterns must be available before allocation optimization can be at all meaningful.

Conclusions

Theoretically, the file allocation problem seems to be well understood. However, the models and associated cost formulas to be found in the literature do not necessarily reflect the realities of distributed data management. Witness the total lack of concern with the synchronization problems which the broadcasting of updates to multiple copies causes. Nevertheless, we find that it is not very difficult to develop new models to handle, say, a primary-copy scheme. Obtaining models and associated cost formulas for any specific environment is largely a matter of having the data access process carefully defined. In a way, the model is such a careful definition.

If the network is not very large - say 20 sites or less - the best allocation can be found by brute force, if necessary. That is, the costs of all possible allocations can be computed and compared. As networks grow in size, this computational problem rapidly becomes too large to be feasible. Considerable progress has been made in making the problem more tractable. Doubtless there is room for more work in this area. We believe that one useful approach is the development of very simple models and resulting decision rules that answer specific questions. Examples of this kind of approach were provided in the section just preceding.

A serious problem - and one which research alone can not solve - is the determination of the parameters to be put into the model. We have indicated that "costs" need not (and perhaps should not) be based on real dollars, but instead might reflect the data base manager's judgment as to the "value" of the various resources, or the relative importance of the various terms in the overall cost formula. In short,

cost factors can be thought of as general weight factors, which may reflect subjective judgments as well as objective cost accounting. The information on data base usage is also specific to the system. Measurements must be taken of query and update volumes before the model can yield valid results. The researcher can aid in providing statistical analyses of the measurements. Such analyses might yield information on whether usage patterns appear to be static or oscillatory, or show long-term trends. But the burden is again on the data base manager to supply the raw data needed to put real numbers in the formulas.

Finally, we note that it might be reasonable to attack the file allocation problem from an entirely different point of view than that of minimizing "cost". One might instead want to minimize, say, response time. Minimization of response time does have some attractions, especially in an environment where rapid response may be critical. The problem must be constrained, of course. Otherwise the fastest response is obtained by putting a copy at each site. Reasonable constraints might be to put a bound on storage and update costs, or to set a maximum number of copies.

In our discussion of response time as a constraint, however, we have already implicity identified two difficulties with this approach. First, response models are complex. More data and parameters are needed to compute response. No readily usable response model seems to currently exist in the literature; additional research is needed on this problem. Second, response models necessarily couple together the effects of all the files in the system. Unless some clever way can be found to get around this problem, computation of a minimal response allocation will be intractable even for small networks. In spite of these difficulties, we believe that research along these lines would be definitely worthwhile.

References

1. Alsberg, P.A. et al. Synchronization and Deadlock. CAC Document No. 185 (CCTC-WAD Document No. 6503), Center for Advanced Computation, University of Illinois at Urbana-Champaign, March 1976.

2. Alsberg, P.A., Belford, G.G., Day, J.D., and Grapa, E. Multi-Copy Resiliency Techniques. CAC Document No. 202 (CCTC-WAD Document No. 6505), Center for Advanced Computation, University of Illinois at Urbana-Champaign, May 1976.

3. Belford, G.G., Schwartz, P.M., and Sluizer, S. The Effect of Backup Strategy on Data Base Availability. CAC Document No. 181 (CCTC-WAD Document No. 6501), Center for Advanced Computation, University of Illinois at Urbana-Champaign, February 1976.

4. Casey, R.G. Allocation of copies of a file in an information network. AFIPS Conference Proceedings 40, AFIPS Press, Montvale, N.J., 1972, pp. 617-625.

5. Chandy, K.M. and Hewes, J.E. File allocation in distributed systems. Proc. International Symp. on Comp. Performance Modeling, Measurement and Evaluation (March 1976), pp. 10-15.

6. Chu, W.W. Optimal file allocation in a multi-computer information system. IEEE Trans. on Computers C-18 (1969), pp. 885-889.

7. Chu, W.W. Optimal file allocation in a computer network. In Computer-Communications Networks, N. Abramson and F. Kuo (Eds.), Prentice-Hall, Englewood Cliffs, N.J., 1973.

8. Eswaran, K.P. Placement of records in a file and file allocation in a computer network. Information Processing 74, North-Holland Publishing Co., Amsterdam, 1974, pp. 304-307.

9. Johnson, P.R. and Thomas, R.H. The Maintenance of Duplicate Databases." RFC #677, NIC #31507, January 1975. (Available from ARPA Network Information Center, Stanford Research Institute-Augmentation Research Center, Menlo Park, CA.)

10. Levin, K.D. Organizing distributed data bases in computer networks. Ph.D Dissertation, University of Pennsylvania (1974).

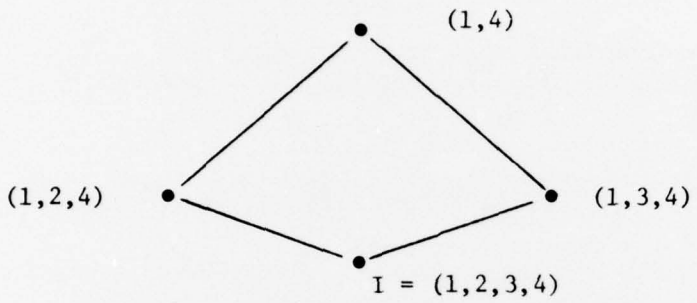
11. Morgan, H.L. and Levin, K.D. Optimal Program and Data Locations in Computer Networks, Report 74-10-01, Dept. of Decision Sciences, The Wharton School, University of Pennsylvania, 1974.

12. Spinetto, R.D. A facility location problem. SIAM Review 18 (1976), pp. 294-5.

13. Urano, Y., Ono, K., and Inoue, S. Optimal design of distributed networks. Second International Conf. on Comp. Comm. (1974), pp. 413-420.

Appendix 1

In this appendix, we prove the analog of Casey's key lemma [4, p. 620] for the primary copy model (equation (2)). This lemma deals with a diamond-shaped subset of the cost graph, as shown below.



It says that if the cost of I is less than (or no more than) that of its two predecessors in this subgraph, then the cost of the top node can not be smaller than that of those predecessors. The lemma holds in general for any four nodes in this configuration. The specific labels on the allocations are designed to clarify their relationship. Just as in Casey's model, this lemma leads straightforwardly to the fact that costs are monotonic non-increasing along paths to the minimum.

Lemma. Without loss of generality, let $I = \{1,2,\ldots,r\}$

If $C(I) \leq C(I \sim \{k\})$, $k = 2,3$;

then

$C(I \sim \{k\}) \leq C(I \sim \{2,3\})$, $k = 2,3$.

(The notation $I \sim \{.\}$ means the set I with the set of elements in the brackets deleted from it.)

Proof.

Let $X_k = C(I\sim\{k\}) - C(I)$, and

$$Y_k = C(I\sim\{2,3\}) - C(I\sim\{k\})$$

We wish to show that if $X_2 \geq 0$ and $X_3 \geq 0$, then $Y_2 \geq 0$ and $Y_3 \geq 0$.

This will be true if $Y_3 - X_2 \geq 0$ and $Y_2 - X_3 \geq 0$.

Now $$X_2 = -\sigma_2 - \Psi d'_{12} + \Sigma\lambda_j (\min_{k\in I\sim\{2\}} d_{jk} - \min_{k\in I} d_{jk})$$

$$Y_3 = -\sigma_2 - \Psi d'_{12} + \Sigma\lambda_j (\min_{k\in I\sim\{2,3\}} d_{jk} - \min_{k\in I\sim\{3\}} d_{jk})$$

So

$$Y_3 - X_2 = \Sigma\lambda_j (\min_{k\in I\sim\{2,3\}} d_{jk} - \min_{k\in I\sim\{3\}} d_{jk} - \min_{k\in I\sim\{2\}} d_{jk} + \min_{k\in I} d_{jk}).$$

The remainder of the proof follows exactly Casey's proof; we will not copy it here. Essentially, one sees easily that the quantity in parentheses is positive for all j; and the inequality $Y_2 - X_3 \geq 0$ follows by permuting indices.

Appendix 2

In this appendix we give formal proofs of the theorems on exclusion and inclusion which were presented and discussed earlier. We are assuming Casey's cost formula (equation 1) and are also assuming that $d_{jj}=d'_{jj}=0$ for all j. The cost of maintaining a copy at site i is given by

$$Z_i = \sigma_i + \sum_{j=1}^{n} \Psi_j d'_{ji}.$$

Theorem 1. All optimal allocations will include site i if

$$\lambda_i \min_{j\neq i} d_{ij} > Z_i. \qquad (A1)$$

Proof. Assume that site i satisfies (A1) and that there is an optimal allocation I which does not include i. Let $I'=I\cup\{i\}$. By straightforward computation,

$$C(I') = C(I) + Z_i + S,$$

where $S = \sum_{j=1}^{n} \lambda_j (\min_{k\in I'} d_{jk} - \min_{k\in I} d_{jk})$.

Clearly each term in S is nonpositive, since taking a minimum over a larger set cannot increase that minimum. Furthermore, since for j=i the contribution to S is $-\lambda_i \min_{k\in I} d_{ik}$, it follows that

$$S \leq -\lambda_i \min_{k\in I} d_{ik} \leq -\lambda_i \min_{j\neq i} d_{ij}.$$

Therefore (from (A1)), $S+Z_i<0$, and hence $C(I')<C(I)$. This contradicts the assumed optimality of I and proves the theorem.

Theorem 2. No optimal allocation including more than one site will include site i if

$$Z_i > \sum_{j=1}^{n} \lambda_j (\max_k d_{jk} - d_{ji}). \qquad (A2)$$

Proof. Given an allocation $I \neq \emptyset$ (the null allocation), let $I' = I \cup \{i\}$. Just as in the proof of theorem 1,

$$C(I') = C(I) + Z_i + S.$$

Making a term by term comparison, we see that the right side of inequality (A2) is greater than or equal to (-S) provided that

$$\max_k d_{jk} - d_{ji} + \min_{k \in I'} d_{jk} - \min_{k \in I} d_{jk} \geq 0 \qquad \text{(A3)}$$

for all j. Examining (A3), we see that if

$$\min_{k \in I'} d_{jk} = d_{ji},$$

then the second and third terms cancel and (A3) is clearly satisfied. Otherwise, the last two terms cancel and (A3) is again satisfied. It then follows from (A2) that

$$Z_i + S > 0.$$

Hence $C(I') > C(I)$, and the theorem is proved.

Theorem 3. A site i cannot be included in any optimal allocation if there exists another site k in the network such that

$$Z_i - Z_k > \sum_{j=1}^{n} \lambda_j (d_{jk} - d_{ji})_+ \qquad \text{(A4)}$$

where $(f)_+ = \begin{cases} f \text{ if } f \geq 0, \\ 0 \text{ if } f < 0. \end{cases}$

This theorem is most readily proved in two parts. We will give these parts as two preliminary lemmas. As in the previous theorems, let I be an arbitrary allocation (but not inlcuding i or k). Let $I' = I \cup \{k\}$ and $I'' = I' \cup \{i\}$. The first lemma states that if site i is sufficiently costly, then adding site i to an allocation which already includes k increases the total cost.

Lemma 1. If site i satisfies

$$Z_i > \sum_{j=1}^{n} \lambda_j (d_{jk} - d_{ji})_+$$

for some site k in the network, then $C(I'')>C(I')$.

Proof. By straightforward computation,

$$C(I'') - C(I') = Z_i + \sum_{j=1}^{n} \lambda_j (\min_{m\in I''} d_{jm} - \min_{m\in I'} d_{jm}).$$

Similarly to the proof of theorem 2, the lemma is proved if, for all j,

$$(d_{jk} - d_{ji})_+ + \min_{m\in I''} d_{jm} - \min_{m\in I'} d_{jm} \geq 0. \qquad \text{(A5)}$$

The inequality (A5) follows easily from consideration of two cases.

(i) If $\min_{m\in I''} d_{jm} \neq d_{ji}$, then the last two terms cancel and clearly the left side is positive.

(ii) If $\min_{m\in I''} d_{jm} = d_{ji}$, then (A5) reads

$$d_{jk} - d_{ji} + d_{ji} - \min_{m\in I'} d_{jm} \geq 0,$$

which again is clearly satisfied.

Our second lemma says that, if (A4) is satisfied, replacing site k by site i in an allocation also increases the cost.

Lemma 2. Let $I'''=I\cup\{i\}$. If sites i and k satisfy inequality (A4), then $C(I''')>C(I')$.

Proof. By straightforward computation,

$$C(I''') - C(I') = Z_i - Z_k + \sum_{j=1}^{n} \lambda_j (\min_{m\in I'''} d_{jm} - \min_{m\in I'} d_{jm}).$$

Similarly to the preceding lemma, the proof follows from showing that

$$\sum_j \lambda_j (d_{jk} - d_{ji})_+ \geq \sum_j \lambda_j (\min_{m\in I'} d_{jm} - \min_{m\in I'''} d_{jm}).$$

And this inequality holds if, for all j, we have

$$(d_{jk} - d_{ji})_+ - \min_{m\in I'} d_{jm} + \min_{m\in I'''} d_{jm} \geq 0. \qquad \text{(A6)}$$

Inequality (A6) can be verified by checking cases.

(i) If $\min_{m\in I''} d_{jm} \neq d_{ji}$ or d_{jk}, then the last two terms cancel and (A6) obviously holds. (Recall that $I''=I\cup\{i,k\}$.)

(ii) If $\min_{m\in I''} d_{jm} = d_{ji}$, then (A6) becomes

$$d_{jk} - d_{ji} - \min_{m\in I'} d_{jm} + d_{ji} \geq 0.$$

(iii) If $\min_{m\in I''} d_{jm} = d_{jk}$, then the left side of (A6) becomes $0 - d_{jk} + \min_{m\in I'''} d_{jm}$, which must be nonnegative by the assumption defining this case.

Proof of Theorem 3. If inequality (A4) holds, then lemmas 1 and 2 both apply. Since lemma 2 says that an allocation containing i is always improved by replacing i by k, and lemma 1 eliminates any allocation containing both i and k from being optimal, nothing more is needed.

Appendix 3

Proof of Strategy Rule 1

Under the assumptions given in the discussion of this rule in the text, the cost of allocation (1) is

$$C(1) = \Psi_1 u + (n-1)\lambda N + n\lambda q + \sigma.$$

The cost of allocation (1,k), where k is any other site, is

$$C(1,k) = \Psi_1(u+N) + \Psi_1 u + (n-2)\lambda N + n\lambda q + 2\sigma.$$

Therefore

$$C(1,k) - C(1) = \Psi_1(u+N) - \lambda N + \sigma.$$

Hence

$C(1,k) > C(1)$ if and only if

$$\Psi_1(u+N) + \sigma > \lambda N, \qquad \text{(A7)}$$

which is the condition in strategy rule 1. Casey's theorem C2 can be applied to show that allocation (1) is the only allocation containing site 1 which could possibly be optimal.

To complete the proof, we must show

(i) that allocation (1) is cheaper than any other single-site allocation, and

(ii) that adding a site to any other single-site allocation increases cost (so that Casey's theorem C2 may again be applied to show that the optimum can not lie below level 1 in the cost graph).

Now $C(k)$ for $k \neq 1$ is $\Psi_1(u+N) + \sigma + (n-1)\lambda N + n\lambda q$. To verify (i) we simply note that

$$C(k) - C(1) = \Psi_1 N.$$

To verify (ii), we compute C(k,j) for the case where neither k nor j is 1.

$$C(k,j) = 2\Psi_1(u+N) + 2\sigma + (n-2)\lambda N + n\lambda q.$$

Therefore

$$C(k,j) - C(k) = \Psi_1(u+N) + \sigma - \lambda N.$$

The condition for $C(k)<C(k,j)$ is precisely that inequality (A7) be satisfied, which is our hypothesis. Thus the result is proved.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER
CAC Document Number 203
CCTC-WAD Document Number 6506

2. GOVT ACCESSION NO.

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)
Research in Network Data Management and Resource Sharing. 7 Network File Allocation.

5. TYPE OF REPORT & PERIOD COVERED
Research rept.,

6. PERFORMING ORG. REPORT NUMBER
CAC #203

7. AUTHOR(s)
G.G. Belford, J.D. Day, E. Grapa, and P.M. Schwartz

8. CONTRACT OR GRANT NUMBER(s)
DCA100-75-C-0021

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Center for Advanced Computation
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS
Command and Control Technical Center
WWMCCS ADP Directorate
11440 Isaac Newton Sq., N., Reston, VA 22090

12. REPORT DATE
August 2, 1976

13. NUMBER OF PAGES
68 (12) 70 p.

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)
UIUC-CAC-DN-76-203, CAC-203

15. SECURITY CLASS. (of this report)
UNCLASSIFIED

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Copies may be obtained from the
National Technical Information Service
Springfield, Virginia 22151

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

No restriction on distribution

18. SUPPLEMENTARY NOTES

None

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

File allocation
Distributed data bases
Network resource sharing

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

This report contains results to date of a study of file allocation in a network. Models and algorithms contained in the literature are surveyed. Some new models (for special situations and for update distribution through a primary site) are developed. Some new theorems for simplifying the computational problem are presented.

407227

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

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

BIBLIOGRAPHIC DATA SHEET	1. Report No. UIUC-CAC-DN-76-203	2.	3. Recipient's Accession No.
4. Title and Subtitle Research in Network Data Management and Resource Sharing - Network File Allocation			5. Report Date August 2, 1976 6.
7. Author(s) G.G. Belford, J.D. Day, E. Grapa, and P.M. Schwartz			8. Performing Organization Rept. No. CAC #203
9. Performing Organization Name and Address Center for Advanced Computation University of Illinois at Urbana-Champaign Urbana, Illinois 61801			10. Project/Task/Work Unit No. 11. Contract/Grant No. DCA100-75-C-0021
12. Sponsoring Organization Name and Address Command and Control Technical Center WWMCCS ADP Directorate 11440 Isaac Newton Square, N. Reston, Virginia 22090			13. Type of Report & Period Covered Research 14.

15. Supplementary Notes
None

16. Abstracts

This report contains results to date of a study of file allocation in a network. Models and algorithms contained in the literature are surveyed. Some new models (for special situations and for update distribution through a primary site) are developed. Some new theorems for simplifying the computational problem are presented.

17. Key Words and Document Analysis. 17a. Descriptors

File allocation
Distributed data bases
Network resource sharing

17b. Identifiers/Open-Ended Terms

17c. COSATI Field/Group

18. Availability Statement	19. Security Class (This Report)	21. No. of Pages
No restriction on distribution Available from the National Technical Information Service, Springfield, Virginia 22151	UNCLASSIFIED	68
	20. Security Class (This Page) UNCLASSIFIED	22. Price