FILE ALLOCATION ON LOCAL COMPUTER SYSTEMS WITH TWO-LEVEL MULTI-ACCESS NETWORKS

ABSTRACT

The file allocation problem (FAP) on local computer systems with two-level multi-access networks is studied in this paper. FAP is less complicated, although it is still NP-hard, on such networks than that on general networks because the communication overhead for remote file access is reduced from a random matrix to a three-value variable. If the communication overhead dominates other overheads, the simple file allocation problem becomes polynomially solvable. For optimal solution, the problem is transformed into a pure zero-one integer programming problem and solved using a variation of Balas's additive search algorithm. A simple heuristic algorithm based on the 0-1 knapsack problem is proposed with analytical and empirical evaluations. The performance degradation of the proposed heuristic algorithm is shown to be small in both evaluations.

Keywords: Distributed computing, distributed database, file allocation, local computer system, two-level local multi-access network.

Accepted to appear in the Information Science, IS-A-95-1559-A.

FILE ALLOCATION ON LOCAL COMPUTER SYSTEMS WITH TWO-LEVEL MULTI-ACCESS NETWORKS

Yao-Nan Lien
AT&T Bell Laboratories
101 Crawfords Corner Rd.
Holmdel, NJ 07733
E-mail: yaonan.lien@att.com

Yih-Long Chang*
School of Management
Georgia Institute of Technology
Atlanta, GA 30332-0520
E-mail: YihLong.Chang@som.gatech.edu

Benjamin W. Wah
Department of Electrical and Computer Engineering
and the Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801
E-mail: wah@aquinas.csl.uiuc.edu

1. INTRODUCTION

This paper studies the file allocation problems on local computer systems with two-level multi-access networks. As computer and local-area networks (LANs) technologies advance, users are moving toward "downsizing" their mainframe systems to distributed computing systems (DCS) which interconnects a number of smaller computers with a local area network. In such environment, both users and resources are physically dispersed in a local area, it requires special system control and resource management strategies to deliver the effective aggregated computing capability to the users. One particular example of the DCS is Merrill Lynch case reported in Berman and Nigam [1992]. To avoid a premature performance saturation, it is important to efficiently manage the following scarce resources with a balanced utilization: processing capacity (including processor and memory), disk, and network. Among many other resource management strategies, a good file allocation strategy is essential to reduce remote data-access overhead and to increase data availability. Compared to wide-area-networks (WAN), the performance of DCS on LAN environment is much more sensitive to the quality of file allocation strategy due to its lower space locality. (This is because (1) users are able to access the system from many different nodes and (2) the locations of processes and data are often controlled by the system for reasons such as load balancing and location transparency.)

The placement of files has been extensively studied as the *file allocation problem* (FAP) in distributed databases (Apers, 1988; Chu, 1969; Dowdy and Foster, 1982). The FAP entails the distribution of replicated files to a set of sites on a computer network to minimize the overall overhead or cost. To reduce remote data-access overhead and to increase data availability, multiple copies of files are allocated to sites where demands are strong, however, at the expense of extra overhead in consistency maintenance and data storage. Originated by Chu (1969), FAP was formulated as an integer programming problem in which the objective is to minimize the overall operating overhead with the constraints of response time and storage. Subsequently, much work has been done in this area (Lien and Wah, 1987; Lien et al., 1988; Wah, 1984; Wah and Lien, 1984). A special case of FAP is the *simple file allocation problem* (SFAP) (Ramamoorthy and Wah, 1983), in which multiple copies of a single file are to be allocated and the effects of queries, updates, and data storage are represented as costs. Another variation of FAP is to dynamically migrate files to requesting sites in order to preserve or improve system performance. The file migration problems have also been widely studied (Smith, 1981a,b,c; Procar, 1982; Liu Sheng, 1986, 1992a, 1992b). This paper focuses on the static file allocation problems on local computer systems with two-level multi-access networks.

Both FAP and SFAP on general point-to-point networks are known to be NP-hard. These problems in local area environments were studied by Wah and Lien (1985). They found that, along with some other interesting results, the broadcast characteristics of this kind of networks significantly reduces the complexities of FAP and SFAP, resulting more efficient solutions.

The scalability of a single LAN system may be limited by the LAN bandwidth. It is a natural step to scale up the system capacity by expanding a LAN system into a multi-LAN system where a higher bandwidth backbone network is used to interconnect a number of regular LANs. (Because of the enhanced network capacity and user population, such a network environment may spread around a metropolitan area referred to as a MAN. This paper makes no difference between the two as far as the physical and logical characteristics are concerned.) This paper studies the file allocation problem on this kind of network environments. (Note that the traditional diskless workstation and single file sever environments do not need to solve FAP problem because all files are located in the file server.)

The model of the problem is proposed in Section 2. In Section 3, the general FAP problem is transformed into a linear pure zero-one integer programming problem and is solved by a search

method based on Balas's additive algorithm [Balas, 1965]. The algorithm is suitable for evaluation of small to moderate sized problems. In Section 4, the problem of a communication-dominant system is presented. The practical considerations of the file allocation problems in real systems are discussed in Section 5.

2. MODELS

We assume that a data site takes the following steps to process a remote read-only query:

1. interpret the query (processing overhead),
2. search data from disk (disk I/O overhead), and
3. transmit the result back to the originating site (communication overhead.)

Assume that an update is processed in the following simplified steps:

1. A data site interpret the query (processing overhead), and search data from disk (disk I/O overhead in one site);
2. All data sites proceed with a concurrency and transaction control (communication overhead); and
3. All data sites update the data without a search (nominal I/O overhead on each data site).

The transaction control in a real DBMS is much more complicated than the above procedure. However, since FAP only concerns with the statistical characteristics of all overhead components, there is no need to elaborate the implementation details.

2.1 Throughput vs. Response Time

Two major optimization objectives of a DCS system design are maximizing throughput and minimizing response time. This paper chooses to minimize the response time for the following reasons:

1. The two objectives are not completely independent to each other. If a good scheduling strategy is in place, optimizing one may not necessarily compromise the other significantly.
2. The throughput is easier to scale up in a DCS environment by adding new nodes into the system. On the other hand, adding new nodes may not necessarily reduce response time unless the system is always saturated.

2.2 Cost Analysis

2.2.1 Disk I/O Overhead vs. Communication Overhead

Compared to WANs, the communication overheads are less dominant in LAN environments. (Disk access time is in the range of 10 ms and the transfer rate is in the range of 5 Mbytes; and transmitting a short message from one process to a remote process on the same LAN is in the range of 5-20 ms.) It is a natural guess that disk overhead should be a major cost component in FAP. A careful investigation which will be shown in the following paragraph tells us that this is not quite true. Disk I/O overhead actually can be ignored from the FAP cost model

simply because it is not sensitive to the file location. Except some extreme cases, the disk I/O overhead is actually independent of where the data is stored. In other words, FAP cannot help to reduce disk I/O overhead even though it is very important. (There are many ways to reduce I/O overhead such as indexing, using good query processing algorithms, adding new disks, using faster disks, etc.)

The following discussion will explain why disk I/O overhead is not sensitive to file allocation. First, the access time are fairly uniform across all types of disks (assuming they all use up-to-date technology.) Secondly, for read-only queries, approximately the same amount of disk access will be generated whether it is stored locally or remotely when the same query algorithm is used in each of the data sites. For update requests, this assumption may not be severely violated even when the system is update-intensive and data is replicated because (1) it is the concurrency control and transaction control that make an update request require longer response time, and (2) only one data site needs to process the update request to identify the data items (tuples) that are to be updated.

2.2.2 Processing Overhead

Processing overhead referred to the CPU (or processors) overhead. It was ignored in all other FAP study, either in WAN or in LAN environments, based on the following assumptions:

1. In a WAN environment, the processing overhead is so insignificant as compared to other overheads.
2. In a LAN environment, even it is comparable with other overheads, it will become very insignificant in the near future as the processor speed almost doubled every two or three years.

Unfortunately, after a careful study, we found that the second assumption may not be valid. For the following reasons, the processing overhead may not always be ignored in LAN environments.

The recent memory technology advances are all on the storage density enhancement, not on the access speed enhancement. Typical memory access time is about 50-80 ns at chip level and 100-500 ns at bus level. In the context of FAP, the processing overhead will impact the communication cost which will be explained in the following section. For the same reason as I/O overhead, the impact to the query processing itself is insignificant.

2.2.3 Communication Overhead and High Bandwidth Networks

It is another surprise that higher bandwidth networks such as FDDI (Fiber Distributed Data Interface) or ATM (Asynchronous Transfer Mode) does not significantly reduce communication overhead (in terms of response time).

In general, the communication cost consists of four components: propagation delay, transmission delay, queuing delay, and protocol delay. Propagation delay is negligible on LAN environment; transmission delay is a function of bandwidth; queuing delay depends on the network workload; while protocol delay depends on the complexity of protocols and processing speed. Higher bandwidth network can only reduce transmission delay and queuing delay. Standard communication protocols such as TCP/IP require several layers of program interface before a message can be transferred from the program context to the communication media and vise versa. This procedure may induce a significant overhead due to the memory copying procedure and program execution in each layer. As mentioned above, memory access overhead can not be ignored and, therefore, communication overhead cannot be ignored even if a higher bandwidth LAN is used.

Because the communication overhead may depend on processing speed, we separate the protocol processing overhead from communication overhead in our model to obtain a more accurate model. However, in real systems, there is no practical way to measure them separately. It would be more practical to measure the communication delay regardless of how it is constituted. A simplified model that considers only communication overhead can be used, which will be described in Section 4, although it may be less accurate.

2.3. One-Level Local Multi-access Networks

On a one-level local multi-access network, any message transmitted to the network can be received by every site on the network directly with approximately the same communication overhead. A message can be received by all sites (broadcast) or some selected sites (multicast) simultaneously. The transmission of messages is controlled by an access control protocol. Detailed physical characteristics can be easily found elsewhere. Their logical properties with respect to the data intensive distributed computing applications are as follows.

1. The per-unit cost of inter-site communication is independent of source and destination sites due to the multi-access/broadcast capability.
2. The communication overhead is assumed proportional to the volume of data transmitted. (Shorter controlling messages can be ignored.)
3. The communication cost of updating a file is independent of the number of times that file is replicated (see section 2.2.1).
4. The per-unit disk-access costs and the processing (CPU) costs are identical for all sites (homogeneous environments).
5. Synchronization and consistency maintenance, such as concurrency control, is simplified because the message order is preserved in all sites.
   (Although the FAP does not depend on this property, it is worth to mention that this property is very useful in a distributed environment. It makes the synchronization, concurrency control, and dynamic query processing much easier than that in point-to-point networks (Gouda and Dayal, 1981; Hevner et al., 1985; Kerschberg et al., 1982, Lien, 1986; Lien and Wah, 1987; Masuyama et al., 1983; Sacco, 1984; Wah and Lien, 1985)
6. The overhead to update a file is independent of system load.
   (On point-to-point networks, the update overhead may be very sensitive to the system load due to the extra synchronization needed for increased resource contention. However, on local multi-access networks, simplified concurrency control protocols that are insensitive to the system load by taking the advantage of broadcast/multicast capability and property (5) has been found (Wah and Lien, 1985)

2.4. Homogeneous Two-Level Local Multi-access Networks

The network concerned in this paper consists of three components: a *backbone network*, a set of *local subnets*, and a set of homogeneous *local systems* (sites). The per-unit disk-access costs and the processing costs are identical for all sites. The backbone network and all local subnets are one-level local multi-access networks as described in the last section, except that the backbone network may have a higher data rate. The networks are organized into a two-level hierarchy: each local system is connected to a local subnet; and all local subnets are connected to

the backbone network through gateways (or routers).* Properties (1) and (5) of one-level networks are not preserved in two-level multi-access networks, but many problems involving distributed control can still be simplified. This will be discussed later in the paper. An example of a two-level local multi-access network is shown in Figure 1.

Since the transmissions on different subnets are independent, the total bandwidth is higher than the bandwidth of a single local subnet. A message may have to be stored in the buffer storage at a gateway before it can be forwarded from one network to another. The local subnet to which a particular site is connected is called the *home subnet* of the site. With respect to this site, other local subnets are *remote subnets*. A query (resp., update) initiated from a site is called a *local query* (resp., *update*) if it is directed to local files, a *neighboring query* (resp., *update*) if it is directed to files located in another site in the home subnet, and an *internet query* (resp., *update*) if it is directed to files in a remote subnet. Neighboring and internet queries (resp., update) are *remote queries* (resp., *updates*). With respect to a file f, the sites (resp. local subnets) that are allocated with the file are referred to as the *f-filed sites* (resp., *f-filed subnets*) or *filed sites* (resp., *filed subnets*) in short.

Only one transmission is needed to transmit a message from one site to another site on the same local subnet. Processors in both sites and the home subnet are involved in this transmission. Three transmissions are needed to transmit a message from a given site to a remote subnet: one on the home subnet, one on the backbone network, and one on the target subnet. The message is first transmitted to the home subnet in which the sender resides. It is then forwarded to the backbone network, through one gateway, and, finally, to the remote subnet through another gateway. Processors in both sites, the two local subnets, the backbone network, and the gateways for the two local subnets are involved in this transmission.

2.5. A Model for File Allocation Problems

Assuming homogeneity, the disk overhead to access or modify a file is independent of the location of the file. Although the disk overhead depends on the number of file copies in an update, it is usually insignificant when compared to the overhead in identifying the target data and in executing concurrency control. The problem can therefore be simplified by ignoring the disk overhead (see Section 2.2.1). Thus, local file access has a zero cost since it does not involve any communication overhead.

The cost parameters in the proposed model consist of overheads in processors, local subnets, backbone networks, and gateways. We do not see the cost as a real dollar value but the overhead or time spent. The following notations are defined for the modeling purpose.

N - set of local subnets
S - set of sites in the system
S_n - set of sites in the local subnet n
F - set of files in the database, $|F| = m$
(n,s) - site s of subnet n, $s \in S_n$
λ_{fns} - query load originating at site (n,s) for file f per unit time
ϕ_{fns} - update load originating at site (n,s) for file f per unit time
d_α - per unit communication cost on the backbone network for a remote query
d_β - per unit communication cost on a local subnet for a remote query
d_p - per-unit processing cost for intersite communications in a remote query
d_g - per-unit gateway processing cost for intersite communications in a remote query
d_α' - per-unit communication cost on the backbone network in a remote update
d_β' - per-unit communication cost on a local subnet in a remote update
d_p' - per-unit processing cost for intersite communications in a remote update
d_g' - per-unit gateway processing cost for intersite communications in a remote update
σ_{fns} - storage cost per unit time of file f at site (n,s)
ρ_f - length of file f
C_{ns} - storage capacity at site (n,s)
I_{fns} - 1 if file f is allocated to site (n,s); 0, otherwise
Y_{fn} - 1 if $\Sigma_{s \in S_n} I_{fns} \geq 1$; 0, otherwise

Since the protocols for handling queries and updates may be different, their corresponding overheads may also be different. Local queries do not involve networking overhead. A cost of $d_\beta+2d_p$ is incurred for each data unit in a neighboring query and $(d_\alpha+ 2(d_\beta + d_p+ d_g))$ for an internet query. The cost of an update is different from that of a query in two aspects. First, the per-unit cost of updates may be higher than that of queries. Secondly, the number of network transmissions is dependent on the number of file copies. Considering the update cost for each data unit, there is no cost if a unique copy is allocated at the site where the update is initiated. When the home subnet is the unique f-filed subnet, the overhead is d_β' and a cost of d_p' for each f-filed site except the initiation site. The cost for an internet update is more complicated: one transmissions on the home subnet, one transmission on the backbone network, and one transmission on each of the remote filed subnets.

Obviously, at least one copy of every file must be allocated in the system. Further, the total size of the allocated files in a site must not exceed the storage capacity of the site. The problem is formulated as follows.

Problem P

$$\text{Minimize } C(I) = \Sigma_{n \in N} \Sigma_{s \in S_n} \Sigma_{f \in F} (\lambda_{fns} d_{fns}(I) + \phi_{fns} d'_{fns}(I) + \sigma_{fns} I_{fns}) \quad (1)$$

$$\text{Subject to} \quad \Sigma_{n \in N} \Sigma_{s \in S_n} I_{fns} \geq 1 \qquad \forall f \in F \quad (2)$$

$$\Sigma_{f \in F} \rho_f I_{fns} \leq C_{ns} \qquad \forall n \in N,\ s \in S_n \quad (3)$$

$I_{fns} = 1$ if file f is allocated to site (n,s); 0, otherwise

where

$$d_{fns}(I) = \begin{cases} 0 & \text{if } I_{fns} = 1; \\ d_\beta+2d_p & \text{if } I_{fns} = 0 \text{ and } Y_{fn} = 1 \\ d_\alpha+ 2(d_\beta + d_p+ d_g) & \text{if } Y_{fn} = 0 \end{cases}$$

$$d'_{fns}(I) = 0 \qquad \text{if } I_{fns} = 1 \text{ and } \Sigma_{n' \in N} \Sigma_{s' \in S_{n'}} I_{fn's'} = 1;$$

---

* The interface between two communication networks is usually referred to as a *router* if two networks use the same communication protocol; otherwise, it is called a *gateway*.

$= d_\beta' + d_p'(1+\Sigma_{s' \neq s}\ I_{fn's'})$ if $\Sigma_{s' \neq s}\ I_{fns'} \geq 1$ and $\Sigma_{n' \in N}\ Y_{fn'} = 1$;
$= d_\alpha' + (1+\Sigma_{n' \neq n}\ Y_{fn'})(d_\beta' + d_g') + d_p'(1+\Sigma_{s' \neq s}\ I_{fns'})$, otherwise

Note that the number of remote f-filed subnets is $(\Sigma_{n' \in N}\ Y_{fn'} - 1)$ if $Y_{fn} = 1$, and $(\Sigma_{n' \in N}\ Y_{fn'})$ otherwise.

2.6. Remarks

The general FAP on the proposed network model is NP-hard. This can be easily proved by reducing it from the standard 0-1 knapsack problem. The proof will not be shown here since it is a general case of the FAP on one-level homogeneous local multi-access networks (Wah and Lien, 1985). In general, the optimal solutions cannot be obtained efficiently by combinatorial algorithms or mathematical programming methods due to the large size of the problem. Specialized optimization algorithms and faster heuristic solutions are needed for practical purpose.

Further, as mentioned in Section 2.2.3, it may not be practical to measure the protocol processing overhead in communication delay. Communication delay may be better represented as an atomic communication overhead. In that case, the model in Section 4, communication dominant system, will be more practical.

3. FILE ALLOCATION WITH STORAGE CAPACITY CONSTRAINTS

In the cases involving storage constraints, the problem, named *P*, is NP-hard and can be converted to a pure zero-one problem. Eq. (1) is rephrased as follows. Since Y_{fn} is a binary variable indicating the existence of file f on subnet n, two constraints can be added.

$$\Sigma_{s \in S_n}\ I_{fns} \leq M\, Y_{fn} \qquad \forall\ n \in N, f \in F \qquad (4)$$

$$\Sigma_{s \in S_n}\ I_{fns} \geq Y_{fn} \qquad \forall\ n \in N, f \in F \qquad (5)$$

where M is a non-Archimedean large number to force Y_{fn} become 1 when $\Sigma_{s \in S_n}\ I_{fns} \geq 1$ in (4). Let Z_{fns} and W_{fns} be binary variables with the following relationships:

$$\Sigma_{n' \in N}\ Y_{fn'} - Y_{fn} \leq M\,(1 - Z_{fns} - W_{fns}), \qquad \forall\ n \in N,\ s \in S_n, f \in F \qquad (6)$$

$$Z_{fns} + W_{fns} \leq 1, \qquad \forall\ n \in N,\ s \in S_n, f \in F \qquad (7)$$

$$\Sigma_{s' \in S_n}\ I_{fns'} - I_{fns} \leq M\,(1 - Z_{fns}), \qquad \forall\ n \in N,\ s \in S_n, f \in F \qquad (8)$$

Then per-unit cost of query and update can be rewritten as:

$$d_{fns}(I) = (d_\alpha + 2(d_\beta + d_p + d_g))(1 - Y_{fn}) + (d_\beta + 2d_p)(Y_{fn} - I_{fns}) \qquad (9)$$

and

$$\begin{aligned} d'_{fns}(I) = \ & d_\alpha' + (\Sigma_{n' \in N}\ Y_{fn'} - Y_{fn} + 1)(d_\beta' + d_g') \\ & + d_p'(\Sigma_{n' \in N}\ \Sigma_{s' \in S_{n'}}\ I_{fn's'} - I_{fns} + 1) \\ & - (d_\alpha' + d_\beta' + d_p' + d_g')Z_{fns} - (d_\alpha' + d_g')W_{fns} \end{aligned} \qquad (10)$$

Given $I_{fns} = 1$, Y_{fn} is 1 from Eq. (4), and $d_{fns}(I)$ is 0 in QE. (9). Given $Y_{fn} = 1$ and 0,I) is + $2d_p$ in Eq. (9). If Y_{fn} is 0, I_{fns} will be 0 from Eq. (4) and $d_{fns}(I)$ will be $d_\alpha + 2d_\beta + 2d_p + 2d_g$. Given $I_{fns} = 1$ and $\Sigma_{n' \in N}\ \Sigma_{s' \in S_{n'}}\ I_{fn's'} = 1$, Eqs. (6) to (10) will force $Z_{fns} = 1$ and $W_{fns} = 1$ since the coefficient of Z_{fns} is less than that of W_{fns} in Eq. (10), and $d'_{fns}(I)$ will be 0. Given the second case of $d'_{fns}(I)$, Eq. (8) will force $Z_{fns} = 0$ and W_{fns} will be 1 since the minimization of Eq. (10). Given the third case of $d'_{fns}(I)$, Eq. (6) will force $Z_{fns} = 0$ and $W_{fns} = 0$, and that $d'_{fns}(I)$ in Eq. (10) represents the update cost. Therefore, Eqs. (9) and (10) represent the original $d_{fns}(I)$ and $d'_{fns}(I)$ in problem P. By combining Eqs. (4) through (10) into problem P, we can restate it as a pure zero-one formulation, namely *Q*, shown below.

Problem Q

$$\text{Minimize } C(I) = \Sigma_{n \in N}\ \Sigma_{s \in S_n}\ \Sigma_{f \in F}\,(a_{fns}Y_{fn} + b_{fns}Z_{fns} + c_{fns}W_{fns} + d_{fns}I_{fns} + e_{fns}) \qquad (11)$$

subject to (2) to (8)

and Y_{fn}, Z_{fns}, W_{fns}, and I_{fns} are 0-1 valued, where

$$a_{fns} = -(d_\alpha + d_\beta + 2\,d_g)\lambda_{fns} - (d_\beta' + d_g')\phi_{fns} + (d_\beta' + d_g')\Sigma_{n \in N}\ \phi_{fn's}$$

$$b_{fns} = -(d_\alpha' + d_\beta' + d_p' + d_g')\phi_{fns}$$

$$c_{fns} = -(d_\alpha' + d_g')\phi_{fns}$$

$$d_{fns} = \sigma_{fns} - (d_\beta + 2\,d_p)\lambda_{fns} - d_p'\phi_{fns} + d_p'\Sigma_{n' \in N}\ \Sigma_{s' \in S_{n'}}\ \phi_{fn's'}$$

$$e_{fns} = (d_\alpha + 2d_\beta + 2\,d_p + 2\,d_g)\lambda_{fns} + (d_\alpha' + d_\beta' + d_p' + d_g')\phi_{fns}$$

and *M* is a non-Archimedean large number.

Problem *Q* itself is a combinatorial problem. For large problems, the number of 0-1 variables or constraints may be huge. However, in most practical systems the number of local subnets and sites tends to be small, and the scale of a problem can be further reduced by excluding those files that can be pre-allocated (Wah, 1984).

Small to medium sized problems can be solved by special zero-one algorithms. Balas's additive algorithm (Balas, 1965) has demonstrated its efficiency in solving pure zero-one linear problems. We have developed a modified additive algorithm based on the Balas's algorithm to solve Problem Q (Lien et al., 1988)

4. FILE ALLOCATION ON COMMUNICATION DOMINANT SYSTEMS

In communication dominant systems, the processing cost is ignored, and $d_{fns}(I)$ and $d'_{fns}(I)$ read as follows:

$$\begin{aligned} d_{fns}(I) &= 0 && \text{if } I_{fns} = 1; \\ &= d_\beta && \text{if } I_{fns} = 0 \text{ and } Y_{fn} = 1 \\ &= d_\alpha + 2d_\beta && \text{if } Y_{fn} = 0 \end{aligned}$$

$$\begin{aligned} d'_{fns}(I) &= 0 && \text{if } I_{fns} = 1 \text{ and } \Sigma_{n' \in N} \; \Sigma_{s' \in Sn} \; I_{fn's'} = 1; \\ &= d_\beta' && \text{if } \Sigma_{s' \neq s} \; I_{fns'} \geq 1 \text{ and } \Sigma_{n' \in N} \; Y_{fn'} = 1; \\ &= d_\alpha' + (1 + \Sigma_{n' \neq n} \, Y_{fn'}) d_\beta' && \text{otherwise} \end{aligned}$$

As we will show in Section 4.1, the SFAP in this case is polynomially solvable. The general problem with storage constraints is still NP-hard. A slight modification on the linearization shown in Section 3 can be used here. In Section 4.2, an efficient heuristic solution with guaranteed performance is developed.

4.1. Simple File Allocation

In the following discussion, the index *f* is dropped because the problem is defined with respect to a single file.

$$C(I) = \Sigma_{n \in N} \; \Sigma_{s \in Sn} \, (\lambda_{ns} \, d_{ns}(I) + \phi_{ns} \, d'_{ns}(I) + \sigma_{ns} \, I_{ns})$$

In a communication dominant system, the update cost is dependent on the number of filed subnets, but not on the number of filed sites. Only one transmission on each subnet is needed to query or update a file. As shown below, the problem becomes polynomially solvable with respect to the number of sites and the number of local subnets.

To solve the problem, three mutual exclusive allocation alternatives are considered:

(a) *Single-copy allocation*: One copy of the file is allocated to a single site after the optimal placement is determined. In this case, no remote update is needed for changes initiated by the site.

(b) *Single-subnet allocation*: Multiple copies of the file are allocated to a single local subnet after the best subnet is determined. No internet update is needed for the changes initiated by the filed site.

(c) *Multiple-subnet allocation*: The file is allocated to more than one local subnet. Update cost depends on the number of filed subnets.

The optimal solution for a given problem can be discovered by comparing all alternatives.

Single-copy allocation. In the following equation for single-copy allocation, τ denotes the total communication cost when all queries and updates are internet. τ is defined as:

$$\tau = \Sigma_{n \in N} \; \Sigma_{s \in Sn} \, (\lambda_{ns} \, (d_\alpha + 2d_\beta) + \phi_{ns} \, (d_\alpha' + 2d_\beta'))$$

Δ_{ns} denotes the reduction of cost if the file is only allocated to site $(n,s.)$ The equation is

$$\Delta_{ns} = \Sigma_{s' \in Sn} \, (\lambda_{ns'} \, (d_\alpha + 2d_\beta) + \phi_{ns'} \, (d_\alpha' + 2d_\beta')) + \lambda_{ns} \, d_\beta + \phi_{ns} \, d_\beta' - \sigma_{ns} \qquad (12)$$

The first term in Eq. (12) is the reduction of the cost for neighboring queries and updates. The second term is the additional reduction for the requests initiated from site $(n,s.)$ The third term is the storage cost. Obviously, the file should be allocated to the site with the maximum Δ_{ns}. The cost of single-copy allocation becomes:

$$C(I) = \tau - \text{Max}_{n \in N \; s \in Sn} \, (\Delta_{ns})$$

Single-subnet allocation. In the single-subnet allocation, where τ remains the same. The cost reduction is

$$\Delta_n = \Sigma_{s \in Sn} \, (\lambda_{ns} \, (d_\alpha + d_\beta) + \phi_{ns} \, (d_\alpha' + d_\beta')) + \Sigma_{s \in Sn} \, (\lambda_{ns} \, d_\beta - \sigma_{ns}) \, I_{ns} \qquad (13)$$

The first term in Eq. (13) is the reduction of the cost for neighboring queries and updates. The second term is the additional cost reduction for local queries. There is no local update since a neighboring update is inevitable for every update made in the home subnet. To maximize Eq. (13) for subnet *n*, every site in subnet *n* having positive values of $(\lambda_{ns} \, d_\beta - \sigma_{ns})$ should be allocated with a copy. In the subnets that have zero or one filed site, Eq. (13) is not applicable and such subnets should not be considered in the single subnet allocation since such an allocation will not be better than the single-copy solution. The cost reduction for the filed subnet *n* is:

$$\Delta_n = \Sigma_{s \in Sn} \, (\lambda_{ns} \, (d_\alpha + d_\beta) + \phi_{ns} \, (d_\alpha' + d_\beta')) + \Sigma_{s \in Sn} \, \text{Max}(0, (\lambda_{ns} \, d_\beta - \sigma_{ns}))$$

The subnet with the maximum Δ_n should be selected as the filed subnet. Every site in that subnet with positive $[\lambda_{ns} d_\beta - \sigma_{ns}]$ should be allocated a copy. Unless all local subnets are discarded, the cost of a single-subnet multiple-copy allocation is then:

$$C(I) = \tau - \text{Max}_{n \in N} \; (\Delta_n)$$

Multiple-subnet allocation. In multiple-subnet allocation, the cost of update depends on the number of filed subnets. The equation becomes:

$$\begin{aligned} d'_{fns}(I) &= 0 && \text{if } I_{ns} = 1 \text{ and } \Sigma_{n' \in N} \; \Sigma_{s' \in Sn \; s' \neq s} \; I_{n's'} = 0; \\ &= d_\alpha' + q \, d_\beta' && \text{if } \Sigma_{s' \neq s} \; I_{ns'} \geq 1; \\ &= d_\alpha' + (1+q) d_\beta' && \text{otherwise} \end{aligned}$$

where q is the number of filed subnets ($q=\Sigma_{n \in N} Y_{fn}$). The case in which there are exactly q filed subnets, called the *q-subnet-allocation*, can be solved easily. For a particular q, the multiple-subnet allocation can be solved by selecting the best q subnets. (The allocation of each subnet can be solved in a way similar to the single-subnet allocation.). The original multiple-subnet allocation can be solved by comparing all n cases.

Let τ_q be the total communication cost when all queries are internet queries, and the copies in all q filed subnets are updated by internet communications in a q-subnet-allocation. Thus,

$$\tau_q = \Sigma_{n \in N} \Sigma_{s \in Sn} (\lambda_{ns} (d_\alpha + 2d_\beta) + \phi_{ns} (d_\alpha' + (q+1)d_\beta')) \quad (14)$$

If subnet n is a filed subnet, the reduction of the cost for remote queries and updates is:

$$\Sigma_{s \in Sn} (\lambda_{ns} (d_\alpha + d_\beta) + \phi_{ns} d_\beta') \quad (15)$$

The first term in the summation is the reduction of the cost for remote queries. The second term is the reduction of the cost for neighboring updates since there are only (q-1) filed subnets. For each copy allocated to site (n,s), a cost of $(\lambda_{ns} d_\beta - \sigma_{ns})$ can be saved as well. Obviously, the optimum allocation for each subnet is to allocate a copy to each site that has a positive $(\lambda_{ns} d_\beta - \sigma_{ns})$. If there is no such site, it is best to allocate a copy to the site with the maximum of $(\lambda_{ns} d_\beta - \sigma_{ns})$ if $\Sigma_{s' \in Sn} (\lambda_{ns'} (d_\alpha + d_\beta) + \phi_{ns'} d_\beta') + \text{Max}_{s \in Sn} (\lambda_{ns} d_\beta - \sigma_{ns}) > 0$. or not to allocate the file to the subnet. If fewer than q subnets can be allocated, the q-subnet-allocation case should be discarded since it is dominated by either the (q')-subnet case, where q'<q, or by the single-subnet allocation. The optimal solution for the multiple-subnet allocation can be obtained by comparing the costs of all q-subnet allocations. Similar to the single-subnet allocation, the multiple-subnet allocation should be discarded if there are one or fewer filed subnets. The cost of an optimum allocation is then:

$$C(I) = \tau - \Sigma_{n \in N} Y_n \Sigma_{s \in Sn} (\lambda_{ns} (d_\alpha + d_\beta) + \phi_{ns} d_\beta') - \Sigma_{s \in Sn} (\lambda_{ns} d_\beta - \sigma_{ns}) I_{ns}$$

Global optimum solution. A global optimum solution can be obtained by comparing the results obtained from the three different allocations. The time complexity for single-copy allocation and single-subnet allocation is O(|S|), where |S| is the total number of sites. The complexity for multiple-subnet allocation is O(|S||N|), where |N|is the number of local subnets. Therefore, the complexity for the algorithm is O(|S||N|).

The solution algorithm is summarized in algorithm SFAP-OPT in Appendix A. A demonstration of the algorithm is shown in Appendix B.

4.2. Heuristic Solutions for File Allocation Problems

Fast heuristic algorithms are needed when the number of variables is more than what a system can handle in a reasonable amount of time and when the locality changes rapidly. It is not difficult to obtain good heuristics for file allocation problems. Their performances, however, are

generally difficult to evaluate. In this section we will discuss several approaches for heuristic algorithm design and propose one with guaranteed performance.

The following approaches all belong to the the category of "divide-and-conquer", a methodology frequently used to obtain approximate solutions for complicated combinatorial problems.

Approach 1. Allocate each file independently using simple file allocation algorithms until all files are allocated. It may be necessary to reallocate some files if the storage capacity constraints are violated.

Approach 2. Allocate the first copy of each file to the system to generate an initial solution, then improve the solution by adding more copies. Allocation of extra copies to each site can be solved as a standard 0-1 knapsack problem using either optimal algorithms or heuristic algorithms.

Approach 3. Allocate files to each site as a standard 0-1 knapsack problem (optimally or heuristically), then, identify the missing files and modify the solution by adding missing files and removing redundant files.

Approach 4. A combination of the above approaches.

Although the knapsack problem itself is an NP-hard problem, there does exist pseudo-polynomial time optimal algorithms (Garey and Johnson, 1979) and good polynomial time suboptimal algorithms. Since the FAP is a design problem, the time to solve FAP with an pseudo-polynomial optimal algorithms may be tolerable. It is difficult to compare these approaches. In the rest of this section, a heuristic algorithm based on Approach 2 will be shown in the rest of this section. It is simply because we can estimate how good a solution will be.

Heuristic Algorithm FAP-HEUR

(a) An arbitrary site is selected to allocate the first copy of each file such that the total length of all files in a given site is at most $\left\lceil \frac{|F|}{|S|} \right\rceil L$, where L is the maximum length of the files. (The single-copy allocation algorithm developed for SFAP in Section 3.1 can be used.)

(b) Additional copies are allocated to each site as a standard 0-1 knapsack problem to fill the remaining storage capacity. In solving the 0-1 knapsack problem, the files are considered the objects; the storage capacity of a site is the capacity of the corresponding knapsack; the length of a file is the weight of the corresponding object; and the profit of allocating a file to site (n,s) is $(\lambda_{fns} d_\beta - \sigma_{fns})$, which is the cost reduction when file f is allocated to more than one subnet.

For simplicity, we assume that the storage capacity of each site is at least $\left\lceil \frac{|F|}{|S|} \right\rceil L$. Otherwise, Step (a) must be modified. Under this assumption, Step (a) is always feasible since the number of files each site can hold is at least $\left\lceil \frac{|F|}{|S|} \right\rceil$. In step (b), either optimal or heuristic 0-1 knapsack algorithms can be used as we have mentioned.

4.3. How Good is Algorithm FAP-HEUR?

Although FAP-HEUR is not the best of its type, its performance can be estimated based on the properties of knapsack problems (Lien, 1987). The evaluation is explained as follows. To evaluate the algorithm, the FAP is first transformed into a *multiple-choice-multiple-knapsack problem* (MCMKP), in which some classes of objects are to be allocated to a set of knapsacks with a goal of maximizing the total profit. Each class has an unlimited number of identical objects. At most one object from each class can be allocated to a knapsack; and at least one object from each class must be allocated to one of the knapsacks. When allocating an object to a particular knapsack, one of the following four profits is obtained:

(a) $p_{fns}(1) = \Sigma_{s' \in S_n} (\lambda_{fns'} (d_\alpha + d_\beta) + \phi_{fns'} d_{\alpha'}) + (\lambda_{fns} d_\beta + \phi_{fns} d_{\beta'} - \sigma_{fns}) - \Sigma_{n' \in N,\, n' \neq n} \Sigma_{s' \in S_n} \phi_{fn's'} d_{\beta'}$

(b) $p_{fns}(2) = \Sigma_{s' \in S_n} (\lambda_{fns'} (d_\alpha + d_\beta) + \phi_{fns'} d_{\alpha'}) + (\lambda_{fns} d_\beta - \sigma_{fns}) - \Sigma_{n' \in N,\, n' \neq n} \Sigma_{s' \in S_n} \phi_{fn's'} d_{\beta'}$

(c) $p_{fns}(3) = \Sigma_{s' \in S_n} (\lambda_{fns'} (d_\alpha + d_\beta)) + (\lambda_{fns} d_\beta - \sigma_{fns}) - \Sigma_{n' \in N,\, n' \neq n} \Sigma_{s' \in S_n} \phi_{fn's'} d_{\beta'}$

(d) $p_{fns}(4) = (\lambda_{fns} d_\beta - \sigma_{fns})$

We denote, among the four different profits, the maximum, the minimum, and the one with respect to a particular optimal allocation as max_p_{fns}, min_p_{fns}, and $p_{fns}(*)$, respectively. Each of these profits is nothing more than the cost reduction from the following fixed cost when one object (file) is allocated to a knapsack.

$$\tau = \Sigma_{n \in N} \Sigma_{s \in S_n} \Sigma_{f \in F} (\lambda_{fns} (d_\alpha + 2d_\beta) + \phi_{fns} (d_{\alpha'} + d_{\beta'}))$$

Maximizing the profit of an MCMKP instance minimizes the cost of the corresponding instance in FAP. The profit $p_{fns}(1)$ is the cost reduction of allocating file *f* to site (n,s) when it is the only copy allocated. The third term in $p_{fns}(1)$ is the communication cost on subnet *n* for the updates generated by remote subnets, which is not counted in τ. Profit $p_{fns}(2)$ is the cost reduction when file *f* is allocated to site *(n,s)* where subnet *n* is the unique f-filed subnet and site (n,s) is the first f-filed site in the subnet. Profit $p_{fns}(3)$ is the cost reduction of allocating file *f* to site *(n,s)* when there are two or more f-filed subnets and site (n,s) is the first f-filed site in subnet *n*. Profit $p_{fns}(4)$ is the cost reduction of allocating file *f* to site *(n,s)* when the conditions in $p_{fns}(1)$, $p_{fns}(2)$, and $p_{fns}(3)$ are not applicable.

After the transformation, the problem becomes the maximization of total profit since (total cost) = τ - (total profit). In the rest of this section, the minimization of overall cost is referred to as the maximization of total profit. The evaluation of FAP-HEUR involves finding the maximum discrepancy between the profit generated by FAP-HEUR and the optimal solution, referred to as the *performance degradation* of FAP-HEUR. The performance degradation is bounded by the discrepancy between the FAP-HEUR and the relaxed-FAP, which is the original FAP without

availability constraints. The relaxed-FAP can be solved optimally by allocating files to each site as a standard 0-1 knapsack problem. The profit of allocating file *f* to site *(n,s)* is $p_{fns}(*)$.

We denote P^*, P^h, and P^{**} as the total profit generated by an optimal solution of FAP, by algorithm FAP-HEUR, and by an optimal solution of relaxed-FAP, respectively. We also denote P^{**}_{ns} as the profit of site *(n,s)* as a knapsack problem instance and P^h_{ns} as the profit produced by site *(n,s)* in algorithm FAP-HEUR. In other words, $P^{**} = \Sigma_{n \in N} \Sigma_{s \in S_n} P^{**}_{ns}$ and $P^h = \Sigma_{n \in N} \Sigma_{s \in S_n} P^h_{ns}$. It is easy to prove that $P^{**} - P^h \leq \Sigma_{n \in N} \Sigma_{s \in S_n} (P^{**}_{ns} - P^h_{ns})$.

Denote P'_{ns} as the profit produced by allocating files to site *(n,s)* when the storage capacity of that site is $(1 + \left\lceil \frac{|F|}{|S|} \right\rceil)L$ and the profit of allocating a file *f* to site *(n,s)* is max_p_{fns}, where L is the maximum length of all files.

Theorem 1. The performance degradation of algorithm FAP-HEUR is at most

(a) $\Sigma_{n \in N} \Sigma_{s \in S_n} (P'_{ns} + \Sigma_{f \in F} \Delta p_{fns})$, or

(b) $P^* \Sigma_{n \in N} \Sigma_{s \in S_n} (\varepsilon_{ns} P^{**}_{ns} / P^*)$ if $\Sigma_{f \in F} \Delta p_{fns} < P'_{ns}$, $\forall f \in F$

where $\Delta p_{fns} = (max\_p_{fns} - min\_p_{fns})$, $\varepsilon_{ns} = (q+1)L/C_{ns}$, $q = \left\lceil \frac{|F|}{|S|} \right\rceil$, and $L = \max \{\rho_f\}$.

Proof. The proof for this theorem is shown in Appendix C.

In practice, part (a) is difficult to use since several instances of the knapsack problem must be evaluated. In contrast, part (b) is much easier to use in some cases. To use part (b), the summation of Δp_{fns} is required to be small when compared to P'_{ns}. This seems to be too strong in practice. However, after examining the derivation of Theorem 1, we found that Δp_{fns} really presents an upper bound of the profit estimation error in allocating file *f* to site *(n,s)*. Since we expect that $p_{fns}(4)$ occurs more frequently than $p_{fns}(1)$, $p_{fns}(2)$, and $p_{fns}(3)$, the real value of $\Sigma_{f \in F} \Delta p_{fns}$ is probably negligible as compared to P'_{ns}. If the discrepancy between P^* and P^{**} is small, $\varepsilon_{ns} P^{**}_{ns} / P^*$ can be replaced by $\varepsilon_{ns} P^{**}_{ns} / P^{**}$, making the solution even easier. The relationship between the estimated upper bound of performance degradation and the various parameters is demonstrated in Figure 2 on the following conditions: (a) P^{**}_{ns} is similar in all sites; (b) $\Sigma_{f \in F} \Delta p_{fns} << P'_{ns}$, $\forall f$; and (c) $P^{**} \equiv P^*$.

4.4 SIMULATION EXPERIMENTS

To compare the heuristic method and zero-one formulation (Problem Q), we randomly generated multiple problem instances of various sizes. The details for these data sets are as follows.

(1) For |N| = 2 and |S| = 2, 10 instances for each of the parameter |F| = 2, 3, 4, 6, 8, 10, 15, 20, 30, and 40 were generated. That is, we randomly generated 100 instances in total for different number of files.

(2) For each problem, the coefficients λ_{fns}, ϕ_{fns}, σ_{fns}, ρ_f, and C_{ns} were uniformly determined as follows:.

(a) λ_{fns} = U(10, 100) (i.e. uniform distribution from 10 to 100.)

(b) ϕ_{fns} = U(1, 50)

(c) ρ_f = U(1, 100)

(d) $\sigma_{fns} = \rho_f \cdot U(0.1, 1) + 1$

(e) $C_{ns} = 100|F|/3 + 100\,|F|\,U(0, 1)$

The result of the experiment is shown in Table 1. Since the zero-one formulation is NP-hard, it is not possible to obtain optimal solutions in practicable time when |F| is greater than 10. Nonetheless, the experiment does show that the heuristic performs better when the problem size increases and the heuristic solution can be found in a reasonable time although it is not a polynomial algorithm. It also shows that the computation time of the heuristic is very stable.

Table 1. Experiment results for |N| =2 and |S| =2.

|F|	Problem Q		HC2BFAP	
	Objective function[1]	CPU Time[3]	Objective function[2]	CPU Time[4]
2	100	22 sec.	150	38 sec.
3	100	55 sec.	140	43 sec.
4	100	4 min.	149	45 sec.
6	100	10 min.	122	42 sec.
8	100	35 min.	121	42 sec.
10	100(*)	40 min.	114	43 sec.
15	100(*)	50 min.	109	54 sec.
20	100(*)	60 min.	106	55 sec.
30	100(*)	90 min.	100	68 sec.
40	100(*)	120 min.	96	93 sec.

1. 10 problems average, normalized to 100%
2. 10 problems average, compared to Problem Q solutions
3. Run on a VAX 8600 with a VMS operating system
4. Run on a Paramid 98x with a UNIX 4.2BSD operating system

*. Not solved to optimal

5. IMPLEMENTATION CONSIDERATIONS

5.1. Applicability of The FAP Model

Not every distributed computing system can benefit from FAP. The model is best suited to the environment in which most of the files are shared by a number of users such as database systems. In the environment in which most of the files are private files, a simple strategy such as cache-on-demand may be better than the complicated integer programming model. A careful evaluation must be made before the model is actually applied to a real system.

5.2. Dynamic File Allocation

The instantiation of the model seems to be a serious problem in applying the FAP model to a real system. A real system is very unlikely to have stable access rates for the whole period between two consecutive file reorganizations. Furthermore, the accuracy of access rate estimation presents another problem in applying the model, especially, in the local area environments where the user/process/data mobility is higher than wide-area-networks as mentioned in Section 1. However, the dynamic file allocation strategy, known as the *file migration problem* (FMP), to relocate the files periodically in response to the changes in the file access pattern is very useful to alleviate the problem. The basic model of FMP is to reallocate the files as a file allocation problem at the end of each time "window", which is a fixed length time interval, based on the observed references during the last window (Gavish and Liu Sheng, 1990a, 1990b; Levin, 1982; Levin and Morgan, 1978; Liu Sheng, 1986, 1992a, 1992b; Porcar, 1982; Segall, 1976; Segall and Sandell, 1979).

5.3. File Preallocation and Reduction of Problem Size

In practice, the large number of files may make FAP impossible to solve optimally in a reasonable amount of time. Optimal algorithms, however, are still useful for the following reasons:

(a) Optimal algorithms provide a boundary within which one can arrive at approximate solutions. For example, the optimal algorithm we have proposed earlier may be modified to use a heuristic knapsack algorithm instead of an optimal algorithm.

(b) For those systems operating under steady state, it is not necessary to solve the file allocation problem in real time. The allocation is only needed when the file system or the database is reorganized. Therefore, a long execution time to solve the FAP is tolerable as long as the ratio of the execution time to the mean time between reorganization is reasonable.

Problem size can be reduced by allocating the following files separately: (1) rarely used files; (2) files with strong locality in a few sites; and (3) frequently-used files with weak locality. For rarely used files, a single copy should be allocated. Files in category (2) should have a copy allocated to each site. For the files in category (3), replicating files in each system can be beneficial. Simple allocation heuristics such as a best-first search are useful in these preallocations.

6. CONCLUSION

The file allocation problem on homogeneous local computer systems with two-level local multi-access networks is studied in this paper. The problem is less complicated on two-level networks than on general point-to-point networks because the communication cost of remote file access is reduced from a random matrix to a three-value variable. The general file allocation with storage-limit constraints is still an NP-hard problem. It is transformed into a linear zero-one integer programming problem which is solved by a search method based on Balas's additive algorithm. In the case in which the system is dominated by the communication cost, the simple file allocation problem is solved in O(|S||N|) time complexity, where |F| is the total number of sites in the network and |N| is the total number of local subnets in the system. A simple heuristic algorithm based on the 0-1 knapsack problem is proposed with analytical and empirical evaluations. The performance degradation of the proposed heuristic algorithm is shown to be small in both evaluations.

References

1. Apers, Peter, "Data Allocation in Distributed Database Systems," *ACM Trans. on Database Systems*, vol. 13, No. 3, pp. 263-304, Sept. 1988.

2. Balas, Egon, "An Additive Algorithm for Solving Linear Programs with Zero-one Variables," *Operating Research*, vol. 13, pp. 517-549, July-August 1965.

3. Berman, L., and Nigam, R., "Optimal Partitioning of Data Bases across Multiple Servers in a LAN," *Interface*, vol. 22, no. 2, pp. 18-27, March-April, 1992.

4. Chu, W. W., "Multiple File Allocation in a Multiple Computer System," *IEEE Trans. on Comp.*, vol. C-18, no. 10, pp. 885-889, Oct. 1969.

5. Dowdy, L. W. and D. V. Foster, "Comparative Models of the File Assignment Problem," *ACM Computing Surveys*, vol. 14, no. 2, pp. 287-313, June 1982.

6. Fortin, Michael, S. J. Kao, and Douglas Kerr, "Benchmarking Workstations from the User's Perspective: Taking into Account the Environment," Working Paper, Dept. of Comp. and Info. Sci., The Ohio State Univ., Columbus, Ohio, Sept. 8, 1988.

7. Garey, Michael R. and David S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freemen, Reading, England, 1979.

8. Gavish, B., and Liu Sheng, O. R., "Adaptive Heuristics for Dynamic File Migration Policies in Distributed Computer Systems", *Working Paper*, University of Arizona, 1990a.

9. Gavish, B., and Liu Sheng, O. R., "Dynamic File Migration Policies in Distributed Computer Systems", *Comm. ACM* , Vol. 33, no. 2, pp. 177-189, 1990b.

10. Gouda, M. G. and U. Dayal, "Optimal Semi-join Schedules for Query Processing in Local Distributed Database Systems," *Proc. ACM SIGMOD Conference*, pp. 164-175, May 1981.

11. Hevner, A. R., O. Q. Wu, and S. B. Yao, "Query Optimization on Local Area Networks," *ACM Trans. on Office Information Systems*, vol. 3, no. 1, pp. 35-62, Jan. 1985.

12. Kerschberg, Larry, Peter D. Ting, and S. Bing Yao, "Query Optimization in Star Computer Networks," *ACM Transactions on Database Systems*, vol. 7, no. 4, pp. 678-711, Dec. 1982.

13. Levin, K. D. and H. L. Morgan, "A Dynamic Optimization Model For Distributed Databases," *Oper. Res.*, vol. 26, no. 5, pp. 824-835, Sept.-Oct. 1978.

14. Levin, K. D., "Adaptive Structuring of Distributed Database," Proc. National Computer Conf., pp. 691-696, 1982.

15. Lien, Yao-Nan, *Distributed Databases on Local Multi-access Computer Systems*, School of Electrical Engineering, Purdue University, Aug. 1986.

16. Lien, Yao-Nan, "Some Properties of 0-1 Knapsack Problems," *Conference on Combinatorics and Complexity*, p. 105, Chicago, Il., May 1987.

17. Lien, Yao-Nan and Benjamin Wah, "Design and Performance Study of DDBLM: A Distributed Database on a Local Computer System," *Proceedings of HICSS-20*, Volume 2, pp. 407-418, Kona, Hawaii, Jan. 1987.

18. Lien, Y. N., Y. L. Chang, and B. W. Wah, "File Allocation Problems on Homogeneous Two-level Local Broadcast Networks," *Proc. of Fourth Int'l Conf. on Data Engineering*, pp. 92-99, Los Angeles, CA, Feb. 1988.

19. Liu Sheng, O. R., *Models for Dynamic File Migration in Distributed Computer Systems*, Ph.D. Dissertation, W. E. School of Business Administration, Univ. of Rochester, New York, 1986.

20. Liu Sheng, O. R., "Analysis of Optimal File Migration Policies in Distributed Computer Systems", *Management Science*, Vol. 38, No. 4, April, 1992a, pp. 459-482.

21. Liu Sheng, O. R., "Optimization of File Migration in Distributed Computer Systems", *Computer and Operation Research* , (forthcoming), 1992b.

22. Masuyama, S., S. Muro, T. Mizutani, T. Ibaraki, and T. Hasegawa, "Shortest Semijoin Schedules for Local Area Distributed Database Systems," *Proc. 16th Annual Hawaiian Int'l Conf. on System Sciences*, pp. 284-293, IEEE, 1983.

23. Morris, James H., Mahadev Satyanarayanan, Michael H. Conner, John H. Howard, David S. H. Rosenthal and F. Donelson Smith, "ANDREW: A Distributed Personal Computing Environment," Commun. of the ACM, vol. 29, no. 3, pp. 184-201, March 1986.

24. Nguyen, Gia Toan, "Distributed Query Management for a Local Network Database System," *Proceedings of the Second International Conference on Distributed Computing Systems Parts*, April 1981.

25. Notkin, David, Norman Hutchinson, Jan Sanislo, and Michael Schwartz, "Heterogeneous Computing Environments: Report on The ACM SIGOPS Workshop on Accommodating Heterogeneity," *Commun, ACM*, vol. 30, no. 2, pp. 132-140, Feb. 1987.

26. Porcar, J. M., *File Migration in Distributed Computer Systems,* Ph.D. Dissertation, Lawrence Berkeley Lab., Univ. of California, Berkeley, July 1982.

27. Ramamoorthy, C. V. and B. W. Wah, "The Isomorphism of Simple File Allocation," *IEEE Trans. on Computers,* vol. C-32, no. 3, pp. 221-232, March 1983.

28. Sacco, G. M., "Distributed Query Evaluation in Local Area Networks," *Proc. of Int'l Conf on Data Engineering,* pp. 510-516, Los Angeles, CA, April 1984.

29. Segall, A., "Dynamic File Assignment in a Computer Network," *IEE Trans Auto, Control,* vol. AC-21, no. 2, April 1976.

30. Segall, A. and N.R. Sandell Jr., "Dynamic File Assignment in a Computer Network - Part II: Decentralized Control," *IEEE Trans Auto, Control,* vol. AC-24, no. 5, Oct. 1979.

31. Smith, A. J., "Analysis of Long Term File Reference Patterns for , Application to File Migration Algorithms," *IEEE Trans. Soft Eng.,* vol. 7, no. 4, pp. 403-417, July 1981.

32. Smith, A. J., "Long Term File File Migration: Development and Evaluation of Algorithms," *Comm. ACM,* 24 (1981b), pp. 521-532.

33. Smith, A. J., "Optimization of I/O Systems by Cache Disks and File Migration: A Summary," *Performance Evaluation,* 1 (1981c), pp. 249-262.

34. Wah, B. W., "File Placement on Distributed Computer Systems," *IEEE Computer,* vol. 17, no. 1, pp. 23-32, Jan. 1984.

35. Wah, B. W. and Y. N. Lien, "The File-Assignment and Query-Processing Problems in Local Multiaccess Networks," *Proc. of Int'l Conf. on Data Engineering,* pp. 228-235, Los Angeles, CA, April 1984.

36. Wah, B. W. and Y. N. Lien, "Design of Distributed Databases on Local Computer Systems with A Multiaccess Network," *IEEE Trans. on Software Eng.*, vol. SE-11, no. 7, pp. 606-619, July 1985.

APPENDIX A

```
/*-----------------------------------------------------------------
   Algorithm SFAP-OPT: to solve SFAP
   on two-level local multi-access networks.
-----------------------------------------------------------------*/

/* (Single-copy Allocation) */
  max (Δ = -∞
  FORALL n IN N DO {
    FORALL s IN  S_n DO {
      IF ( Δ_ns > maxΔ )    /*  Eq. (12) */
      THEN { maxΔ = Δ_ns; sc.allocation = (n,s) }
    } }
  sc_cost = τ - maxΔ

/* (Single-network Allocation) */
  maxΔ = -∞
  FORALL n IN N DO {
    number_of_copy = 0

       Δ_n =Σ_{s ∈ S_n} (λ_ns (d_α+ d_β) + φ_ns (d_α'+d_β'))    /*  Eq. (13) */

    FORALL s IN  S_n DO {

      saving = (λ_ns d_β - σ_ns)
      IF ( saving > 0 )
      THEN {
            Δ_n = Δ_n + saving

            allocation[n] = allocation[n] ∪  (n,s)
            number_of_copy = number_of_copy + 1
      } }
       IF ( number_of_copy <= 1 )
       THEN { Δ_n = -∞; allocation[n] = ø;}
       IF (Δ_n > maxΔ)
       THEN { maxΔ = -∞; sn_allocation = allocation[n]}
         } }
  sn_cost = τ - maxΔ

/* (Multiple-network Allocation) */
     mn_cost = ∞
     FOR q FROM 2 TO N DO {
          number_of_subnet = 0
          FORALL n IN N DO {
             number_of_copy = 0
```

$\Delta_{qn} = \Sigma_{s \in Sn} (\lambda_{ns} (d_\alpha + d_\beta) + \phi_{ns} d_\beta')$ /* Eq. (15) */

```
        maxΔ = -∞
        FORALL s IN Sn DO {
          saving = (λns dβ - σns)
          IF ( saving > 0 )
          THEN {
          Δqn = Δqn + saving
          net_allocation[n] = net_allocation[n] ∪ (n,s)
          number_of_copy = number_of_copy + 1
          }
      IF (maxΔ < saving < 0 AND number_of_copy = 0 )
      THEN { maxΔ = saving; Smax = (n,s) }
      } /* s */
     IF ( number_of_copy = 0 )
     THEN {
       Δqn = Δqn + maxΔ
       If (Δqn > 0)
       THEN {
          net_allocation[n] = net_allocation [n] ∪ Smax
          number_of_subnet = number_of_subnet + 1
     } }
       ELSE number_of_subnet = number_of_subnet + 1
     } /* n */
IF (number_of_subnet ≥ q)
THEN {
    picked_subnet = first q subnet in descending order of Δqn
    cost[q]=Σn∈ N Σs ∈ Sn (λns (dα+ 2dβ) + φns (dα'+(q+1)dβ')) /* Eq. (14) */
    FORALL n IN picked_subnet DO {
     q_allocation[q] = q_allocation[q] ∪ net_allocation[n]
     cost[q] = cost[q] - Δqn
   } }
 ELSE cost[q] = ∞
 IF ( cost[q] < mn_cost )
 THEN { mn_cost = cost [q]; mn_allocation = q_allocation[q] }
 }    /* q */
/* Compare all cases */

Select the minimum among sc_cost, sn_cost, and mn_cost
}
```


APPENDIX B

An example showing the simple file allocation algorithm

d_β	d_β'	d_α	d_α'
1	2	3	4

		n=1		n=2	
		s=1	s=2	s=3	s=4
C		50	40	50	40
	λ	6	7	4	5
|F|=1	φ	3	6	2	4
L=30	σ	3	5	2	4
	λ	1	4	5	6
|F|=2	φ	7	3	5	2
L=20	σ	2	1	4	2

File 1 is to be allocated in the simple file allocation.

Single-copy allocation:

$\tau = 230$

Δ_{ns} = [115, 120, 78, 81]

The file is allocated to site 2 with C(I) = 110.

Single-network allocation:

$\tau = 230$

Δ_n = [111, 75]

The file is allocated to site 1 and 2 in network 1 with C(I)=119.

Multiple-network allocation:

$\tau_q = 260$

sites 1,2 in network 1 and sites 3,4 in network 2 are allocated with C(I)=134.
Finally, three cases are compared and the single copy is allocated to site 2 with a cost 110.

APPENDIX C
Proof of Theorem 1

The properties found in Lien (1987) are very useful to prove Theorem 1 shown in Section 4.

C.1. Some Properties of 0-1 Knapsack problems.

The standard 0-1 knapsack problem is as follows: given a set of objects $N = \{1, ..., n\}$, the associated positive weights and profits, $\{w_i \mid i \in N\}$, $\{p_i \mid i \in N\}$, and a positive capacity c, find $x_1, x_2, ..., x_n$ so as to maximize the total profit as follows:

$$Z = \max \left\{ \sum_{i=1} p_i x_i \mid \sum_{i=1} w_i x_i \le c;\ x_i \in \{0,1\};\ i \in N \right\}.$$

For easy to refer, the instance of allocating objects N to a knapsack with a capacity c is denoted as (c, N); an optimal allocation of the instance and the associated total profit are denoted as A_N^c and Z_N^c respectively.

Theorem C1 described below is useful in answering the following question: given a knapsack instance (c, N), what will be the maximum profit degradation if the objects in Q, a subset of N, is preallocated in the knapsack? The following notations are defined for Theorem C1.

W_q $\quad \sum_{i \in Q} w_i$.

Z_1 $\quad Z_N^c$.

Z_2 $\quad \left[Z_{N-Q}^{c-W_Q} + \sum_{i \in Q} p_i \right]$,

where $Z_{N-Q}^{c-W_Q}$ is the maximum profit of allocating the remaining objects $(N\text{-}Q)$ to the knapsack with the remaining capacity $(c\text{-}W_Q)$ when all the objects in Q have been allocated in the knapsack already. Actually, Z_2 can be restated as follows.

$$Z_2 = \max \left\{ \sum_{i=1}^{n} p_i x_i \mid \sum_{i=1} w_i x_i \le c;\ x_i \in \{0,1\},\ i \in N\text{-}Q;\ x_i = 1,\ \forall i \in Q \right\}.$$

Theorem C1:

For any knapsack instance (c, N), and Q, a subset of N,

(a) $Z_2 + Z_N^{W_Q+k} \ge Z_1$,

(b) $Z_2 \ge \left[1 - \frac{W_Q+k}{c}\right] Z_1$, and

(c) either $\frac{Z_2}{Z_1} > \frac{c-W_Q}{c}$ or $\frac{Z_2}{Z_1} \cong \frac{c-W_Q}{c}$, if $c\text{-}W_Q >> k$,

where $k = \max_{i \in N} \{w_i\} \le c$.

Theorem C2 described below is useful in answering questions such as the following. Imagine that everyone in a supermarket contest is to fill a 10 lb. bag with food. The price tags were taken away and everyone has to make his best guess. The one who packs the highest dollar value of food wins. Assume one solves the problem using an optimal knapsack algorithm, then what is the discrepancy between the value he actually earns and the value he could earn if he knew the prices in advance? This scenario can be thought as a 0-1 multiple-choice knapsack problem solved as a standard 0-1 knapsack problem. Therefore, the properties in Theorem C2 are useful to estimate this discrepancy.

A formal description of the problem is as follows. Given two problem instances (c, N_1) and (c, N_2), assume that the objects in N_1 and N_2 are the same with different profits, (the objects in N_1 and N_2 are considered the same if they have the same index), what would be the possible profit degradation as compared to the maximum profit of (c, N_1) if the knapsack is filled with N_1 in such a way: an object of N_1 is allocated to the knapsack if the corresponding object in N_2 is in $A_{N_2}^c$, which is an optimal allocation of (c, N_2).

More notations for Theorem C2 are defined as follows.

N_t the t-th set of objects, $N_t = \{i_t \mid i=1,2,\ldots,n\}$, $t \in \{1,2\}$;

w_{i_t} the weight of object i_t;

p_{i_t} the profit of object i_t;

$A_{2\text{-}1}$ $\{i_1 \mid i_1 \in N_1;\ i_2 \in A_{N_2}^c\}$, which is the set of objects of N_1 corresponding to the objects in $A_{N_2}^c$;

$Z_{2\text{-}1}$ $\quad \sum_{i_1 \in A_{2\text{-}1}} p_{i_1}$.

Theorem C2:

Given two knapsack problem instances (c, N_1) and (c, N_2), $|N_1| = |N_2| = n$, if $w_{i_1} = w_{i_2}$, $i = 1,2, \ldots, n$, *then*

(a) $Z_{2\text{-}1} + \sum_{i=1}^{n} \max\{0, (p_{i_2} - p_{i_1})\} \geq Z^{c}_{N_2}$,

(b) $Z_{2\text{-}1} + \sum_{i=1}^{n} |p_{i_1} - p_{i_2}| \geq Z^{c}_{N_1}$.

C.2. Proof of Theorem 1

Define the basic-relaxed-FAP as the relaxed FAP except that the profit of allocating file f to site (n,s) is $(\lambda_{fns} d_{\beta} - \sigma_{fns})$. Denote the maximum profit of a basic-relaxed-FAP and the maximum profit associated with each site as P^{***} and P^{***}_{ns} respectively. Therefore,

$P^{**} = \Sigma_{n \in N} \Sigma_{s \in S_n} P^{**}_{ns} \geq P^{***} = \Sigma_{n \in N} \Sigma_{s \in S_n} P^{***}_{ns} \geq P^{h} = \Sigma_{n \in N} \Sigma_{s \in S_n} P^{h}_{ns}$.

$P^{**} - P^{h} \leq \Sigma_{n \in N} \Sigma_{s \in S_n} (P^{**}_{ns} - P^{h}_{ns})$.

(a)

(a.1) From Theorem 2C, we know $P^{**}_{ns} - P^{***}_{ns} \leq \Delta p_{fns}$.

(a.2) From Theorem 1C, we get $P^{***}_{ns} - P^{h}_{ns} \leq P'_{ns}$.

(a.3) From (a.1) and (a.2), we get $P^{**} - P^{h} \leq \Sigma_{n \in N} \Sigma_{s \in S_n} (\Delta p_{fns} + P'_{ns})$.

(b) Since the first step of algorithm FAP-HEUR is to allocate files to each site with a size at most $\left\lceil \frac{|F|}{|S|} \right\rceil L$ so that $W_Q \leq \left\lceil \frac{|F|}{|S|} \right\rceil L$, where W_Q is the total length of the files allocated to a site in the first step. From Theorem C1 and the assumption that $\Sigma_{f \in F} \Delta p_{fns} << P'_{ns}$, we know that $P^{**}_{ns} - P^{h}_{ns} = P^{***}_{ns} - P^{h}_{ns} \leq P^{*}_{ns} \frac{W_Q + k}{C_{ns}}$, where P^{*}_{ns} is the part of P^{*} that is associated with site (n,s). (That is. $P^{*} = \Sigma_{n \in N} \Sigma_{s \in S_n} P^{*}_{ns}$.) It is not difficult to prove that:

$$P^{*} - P^{h} \leq P^{*} \left(\Sigma_{n \in N} \Sigma_{s \in S_n} \left(\varepsilon_{ns} \frac{P^{**}_{ns}}{P^{*}}\right)\right),$$

where $\Delta p_{fns} = (max\text{-}p_{fns} - min\text{-}p_{fns})$, $\varepsilon_{ns} = \frac{(q+1)L}{C_{ns}}$, and $q = \left\lceil \frac{|F|}{|S|} \right\rceil$.

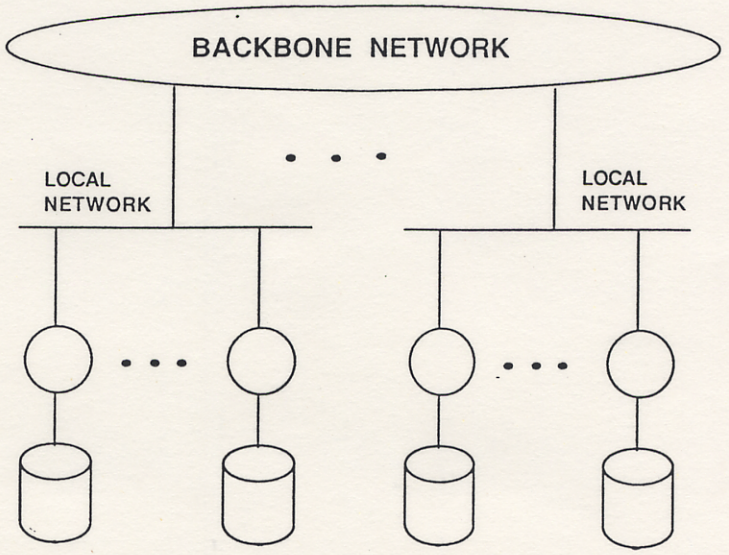

Figure 1 An example of Two-level Local Broadcast Networks.

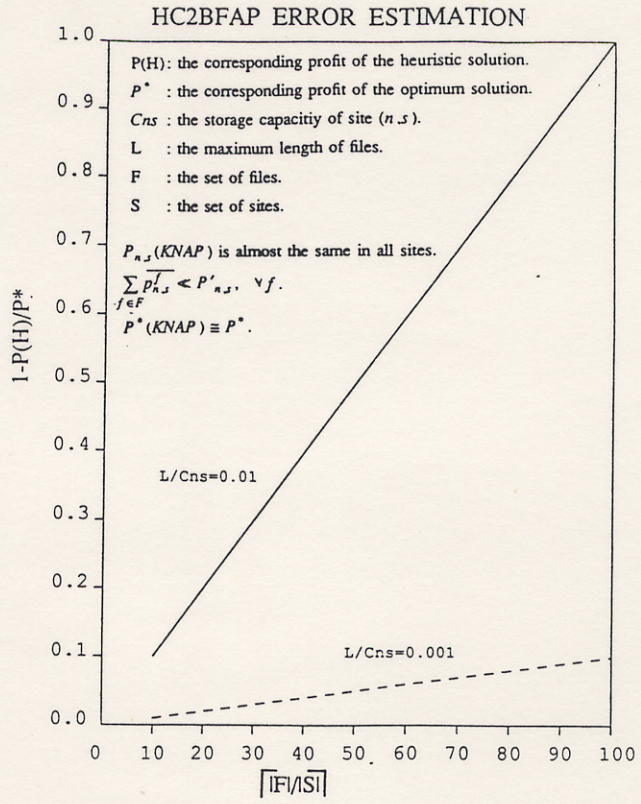

Figure 2 Error Estimation of Algorithm HC2BFAP.