Fourth International Conference on Data Engineering, Feb.1, 1988, pp. 92-99.

FILE ALLOCATION ON HOMOGENEOUS LOCAL COMPUTER SYSTEMS WITH TWO-LEVEL MULTIACCESS NETWORKS

Yao-Nan Lien, * *Yih-Long Chang,* ** *and Benjamin W. Wah* ***

ABSTRACT

Various versions of the file allocation problem on homogeneous two-level local multiaccess networks are examined in this paper. The problems associated with file allocation are more simple on two-level networks than on general point-to-point networks because the communication cost for accessing a file is reduced to a three-value variable. File allocation with storage-limit constraints is still NP-hard. The optimal solution, given these conditions, involves transformation into a pure zero-one integer programming problem and a search algorithm 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 polynomially solvable. An efficient heuristic algorithm with guaranteed performance is proposed for this special case.

INDEX TERMS: Distributed computing, distributed database, file allocation, local computer system, two-level local multiaccess network.

1. Introduction

This paper studies the file allocation problems on two-level local multiaccess networks. As the technology of mini-computers and local-area networks (LANs) advances, distributed computing systems (DCS) based on local-area computer networks are becoming a very attractive means to provide information processing to local user communities. Universities, hospitals, and buildings with branch offices are examples of institutions which can profit from DCS. In environments such as these, both users and resources are physically dispersed, requiring special strategies for system control and resource management in order to deliver information processing capability to users. Processor power, memory storage, extended secondary storage, and the communication bandwidth of the network are important considerations in a distributed computing system. In most cases, these resources are not evenly utilized and overall system efficiency is effected by resulting bottlenecks. Among many tunable design parameters, the allocation of files is considered to be a very important one to improve system performance. A good file allocation strategy can significantly increase data availability and reduce the remote data-access overhead.

Allocation of files has been extensively studied as the *file allocation problem* (FAP) in distributed databases[1,2,3]. The FAP entails the distribution of possibly replicated files to a set of sites on a computer network to minimize the overall overhead. To reduce remote data-access overhead and to increase data availability, multiple copies of files are allocated to sites that demonstrate strong access locality at the cost of increased overhead on consistency maintenance and data storage. FAP was originally investigated by Chu[2], who studied it with respect to multiple files on a multiprocessor system. He considered the problem as an integer programming problem in which the objective is to minimize the overall operating overhead in conjunction with the constraints of response time and available storage capacity. Subsequently, much work has been done in this area[3,4,1,5]. A special case of FAP is the *simple file allocation problem* (SFAP)[6], 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.

Both FAP and SFAP on general point-to-point networks are known to be NP-hard. Recently, these problems in a local area environment were studied by Wah and Lien[7] , and some interesting results were discovered for systems connected by a local multiaccess network. By considering the broadcast characteristics of multiaccess networks, more efficient algorithms have been found.

The FAP in a local-area environment is important since the locality of access of data is usually much lower than that in a wide-area environment. We define the *process-data distance* between a process and its required data to be zero when they are located in the same site. Otherwise it is one. The average process-data distance in a local-area environment is very high since users are able to access the system from many sites, and since locations of processes and data are controlled by the system for reasons such as load balancing and location transparency. This high user/process/data mobility may introduce a significant remote data access overhead. A good file allocation algorithm is, therefore, important in such an environment.

Although efficient file allocation algorithms in local broadcast networks have been found, they are not applicable for larger and more complex local-area systems. The number of sites that a single local broadcast network can support is normally under 100. New communication architectures will have to be used to support larger systems. A very attractive approach is to interconnect a set of local networks with a backbone network through gateways. The Andrew system of Carnegie-Mellon University is an example of this approach[8]. Other examples were discussed at the recent ACM SIGOPS Workshop on Accommodating Heterogeneity[9].

A model for this problem is suggested 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 algorithm based on Balas's additive algorithm. The algorithm is suitable for evaluation of small to moderate sized problems. In Section 4, the problem of a communication-dominated system is presented.

2. Models

2.1. One-Level Local Broadcast Networks

A one-level system is formed by connecting a set of sites or nodes to a single communication medium. Messages transmitted by a site can be addressed to one or more destinations using broadcast capability. Transmission of messages is controlled by an access control protocol. The effective distance of a local-area network is no more than a few kilometers, and the data rate is no higher than 10 MBPS unless special technologies, such as fiber optics, are used. CSMA/CD networks (e.g. Ethernet) and some ring networks (e.g. Token Ring) are the most popular networks in this category. Their logical properties are as follows.

* Department of Computer and Information Science, The Ohio State University, Columbus, OH 43210. (614) 292-5236; lien@cis.ohio-state.edu, cbosgd!osu-cis!lien.

** Department of Management and Information Systems, University of Arizona, Tucson, AZ 85721. (602) 621 7497; chang@arizmis.BITNET.

*** Department of Electrical and Computer Engineering and the Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL 61801. (217) 333-3516; wah@aquinas.csl.uiuc.edu. Research supported by National Science Foundation Grant DMC 85-19649.

International Conference on Data Engineering, IEEE, Los Angeles, California, February 1988.

their corresponding overheads may also be different. Local queries do not involve networking overhead. For each data unit in a neighboring query, a cost of $d_\beta+2d_p$ is incurred. The cost for an internet query is $(d_\alpha + 2(d_\beta + d_p + d_g))$.

The cost of an update is different from that of a query in two respects. First, the per-unit cost of updates may be higher than that of queries. Second, the number of network transmissions may depend on the number of data 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. It is $d_\beta' + d_p'$ for each site having a copy of the file when only sites in the home network have the file. The cost for an internet update is more complicated: one local transmissions on network n, one transmission on the backbone network, and one transmission on each remote network having at least one copy of the updated file are needed.

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

$$\textit{minimize}\ C(I) = \sum_{f,n,s}\left[\lambda^f_{n,s}d^f_{n,s}(I) + \phi^f_{n,s}d'^f_{n,s}(I) + \sigma^f_{n,s}I^f_{n,s}\right] \qquad (1)$$

$$\textit{subject to}\quad \sum_{n\in N}\sum_{s\in S_n} I^f_{n,s} \ge 1\,, \quad \forall\ f\in F$$

$$\sum_{f\in F} l^f I^f_{n,s} \le C_{n,s}\,, \quad \forall\ n\in N, s\in S\,,$$

$$I^f_{n,s} = \begin{cases} 1 & f \text{ is allocated to site } (n,s) \\ 0 & \text{otherwise} \end{cases},$$

where

$$d^f_{n,s}(I) = \begin{cases} 0 & I^f_{n,s} = 1 \\ d_\beta + 2d_p & Y^f_n = 1 \text{ and } I^f_{n,s} = 0 \\ d_\alpha + 2\cdot d_\beta + 2d_p + 2d_g & Y^f_n = 0 \end{cases}$$

$$d'^f_{n,s}(I) = \begin{cases} 0 & I^f_{n,s} = 1 \text{ and } \sum_{n'\in N}\sum_{s'\in S_{n'}} I^f_{n',s'} = 1 \\ d_\beta'+d_p'(1+\sum_{s'\in S_n, s'\ne s} I^f_{n,s'}) & \sum_{s'\in S_n, s'\ne s} I^f_{n,s'} \ge 1 \text{ and } \sum_{n'\in N} Y^f_{n'} = 1 \\ d_\alpha'+\left[1+\sum_{\substack{n'\in N\\ n'\ne n}} Y^f_{n'}\right](d_\beta'+d_g') & \text{otherwise.} \\ \quad + d_p'\left[1+\sum_{\substack{n'\in N\\ s'\ne s}} I^f_{n',s'}\right] & \end{cases}$$

Notice that in Eq. (1) the number of remote networks that have file f is $\left[\sum_{n'\in N} Y^f_{n'} - 1\right]$ if $Y^f_n = 1$, and is $\left[\sum_{n'\in N} Y^f_{n'}\right]$ otherwise.

2.4. 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 for one-level homogeneous local broadcast networks[7]. In general, the problem cannot be solved efficiently and optimally by combinatorial algorithms or mathematical programming methods. Specialized optimization algorithms and faster heuristic solutions are needed.

One way to reduce the complexity of the problem is to identify the important parameters and ignore all others. Two extreme cases are found in systems dominated by the communication overhead and in systems dominated by the processing overhead. Only the first of these systems is examined in this paper.

Although local-area communication network technology is available for most systems, a high-speed local-area network is still very expensive. For the immediate future, LANs with a speed of around 10 MBPS will be most popular. Due to the networking overhead, the effective end-to-end speed may be as low as 1 MBPS. Since bandwidth is shared by many sites, the average bandwidth per site is much lower. On the other hand, processing capacity is easier to increase. With the advance of VLSI and multiprocessing technologies, processing speed has increased significantly in the past 15 years and will continue to increase in the near future. With this increase, the processing capacity of each site can be made much greater than the communication bandwidth. Thus, communication overhead will dominate the overall system overhead. This is especially true in a high user/data/process mobility environment. The FAP for such systems is referred to as the *communication-dominated FAP*. An example of this is CMU's Andrew system, in which the domination by network overhead is predicted after the processors of the file servers are upgraded. Domination in this case can be traced to the lack of a high-speed local network[8].

3. File Allocation with Storage Capacity Constraints

In cases involving storage constraints, the problem, which we will call P, is NP-hard and can be converted to a pure zero-one problem. The transformation is shown in Section 3.1, and some approaches for reduction of problem size are discussed in Section 3.2.

3.1. Linear Integer Programming Formulation

Eq. (1) is rephrased as follows. Since Y^f_n is a binary variable indicating the existence of at least one copy of file f on network n, two constraints can be added.

$$\sum_{s\in S_n} I^f_{n,s} \le M\cdot Y^f_n\,, \quad \forall\ n\in N, f\in F \qquad (2)$$

$$\sum_{s\in S_n} I^f_{n,s} \ge Y^f_n\,, \quad \forall\ n\in N, f\in F \qquad (3)$$

where M is a non-Archimedean large number to force Y^f_n become 1 when $\sum_{s\in S_n} I^f_{n,s} \ge 1$ in Eq. (2). The per-unit cost of query and update can be rewritten as

$$d^f_{n,s}(I) = (d_\alpha+2d_\beta+2d_p+2d_g)(1 - Y^f_n) + (d_\beta+2d_p)(Y^f_n - I^f_{n,s}) \qquad (4)$$

and

$$d'^f_{n,s}(I) = d_\alpha'+(\sum_{n'\in N} Y^f_{n'} - Y^f_n+1)(d_\beta'+d_g')+d_p'\left[\sum_{\substack{n'\in N\\ s'\ne s}} I^f_{n',s'}+1\right] \qquad (5)$$
$$- (d_\alpha'+d_\beta'+d_p'+d_g')Z^f_{n,s} - (d_\alpha'+d_g')W^f_{n,s}$$

where

$$(\sum_{n'\in N} Y^f_{n'} - 1) \le M(1 - Z^f_{n,s} - W^f_{n,s}), \quad \forall\ n\in N, s\in S, f\in F; \qquad (6)$$

$$(\sum_{s\in S_n} I^f_{n,s} - 1) \le M(1 - Z^f_{n,s}), \quad \forall\ n\in N, s\in S, f\in F; \qquad (7)$$

$$Z^f_{n,s} \le I^f_{n,s}\,, \quad \forall\ n\in N, s\in S, f\in F; \qquad (8)$$

$$W^f_{n,s} \le \sum_{s\in S_n} I^f_{n,s} - Z^f_{n,s}\,, \quad \forall\ n\in N, s\in S, f\in F; \qquad (9)$$

$Z^f_{n,s}$ and $W^f_{n,s}$ are 0-1 variables.

Eq. (6) forces $Z^f_{n,s}$ and $W^f_{n,s}$ to become 0 when $\sum_{n'\in N} Y^f_{n'} > 1$, which is a part of the third case of $d'^f_{n,s}(I)$. $Z^f_{n,s} + W^f_{n,s}$ will be equal to 1 when $\sum_{n'\in N} Y^f_{n'} = 1$, because of the minimization in Eq. (5). Further, Eq's (7) and (9) will result in $Z^f_{n,s} = 1$ when $I^f_{n,s} = 1$ and $\sum_{n'\in N}\sum_{s'\in S} I^f_{n',s'} = 1$, since the coefficient of $Z^f_{n,s}$ is less than that of $W^f_{n,s}$ in Eq. (5). Therefore, Eq. (5) represents the original values of $d'^f_{n,s}(I)$ in problem P.

After combining Eq's (2) and (9) into problem P, we can restate it as a pure zero-one problem, namely, Q, as follows.

Single-network allocation. In single-network allocation, where τ remains the same as for single-copy allocation, cost reduction is

$$\Delta_n = \sum_{s \in S_n} \left[\lambda_{n,s} (d_\alpha + d_\beta) + \phi_{n,s} (d_\alpha' + d_\beta') \right] + \sum_{s \in S_n} \left[\lambda_{n,s} d_\beta - \sigma_{n,s} \right] I_{n,s}. \quad (21)$$

The first term in Eq. (21) is the reduction of cost from neighboring queries and updates. The second term is the additional cost reduction from local queries. There is no local update since a neighboring update is inevitable for every update made in the home network. Optimally, the second term is maximized for network n and all sites having positive values of $\left[\lambda_{n,s} d_\beta - \sigma_{n,s}\right]$ have a copy of the file. In networks where one or no sites have positive values for $\left[\lambda_{n,s} d_\beta - \sigma_{n,s}\right]$, multiple copies would not be allocated. In this case, Eq. (21) is no longer appropriate and a single-copy allocation should be made. Except in cases where there is one or no sites, the cost reduction for τ in an optimal allocation for network n is

$$\Delta_n = \sum_{s \in S_n} \left[\lambda_{n,s} (d_\alpha + d_\beta) + \phi_{n,s} (d_\alpha' + d_\beta') \right] + \sum_{s \in S_n} D \left[\lambda_{n,s} d_\beta - \sigma_{n,s} \right],$$

where $D(v) = \begin{cases} v & v > 0 \\ 0 & v \le 0 \end{cases}$.

The network with the maximum Δ_n is selected, and every site in that network with positive $\left[\lambda_{n,s} d_\beta - \sigma_{n,s}\right]$ is allocated a copy of the file. Unless all local networks are discarded, the cost of a single-network multiple-copy allocation is then equal to

$$C(I) = \tau - \max_{n \in N} \left[\Delta_n \right].$$

Multiple-network allocation. A local network retaining a copy of the file is referred to as an *allocated local network*. In multiple-network allocation, the cost of update depends on the number of local networks to which the file is allocated. The equation is

$$d'_{n,s}(I) = \begin{cases} 0 & I_{n,s} = 1 \text{ and } \sum_{n' \in N} \sum_{\substack{s' \in S_{n'} \\ s' \neq s}} I_{n',s'} = 0 \\ d_\alpha' + q \cdot d_\beta' & \sum_{s' \in S_n} I_{n,s'} > 0 \\ d_\alpha' + (q+1) \cdot d_\beta' & \sum_{s' \in S_n} I_{n,s'} = 0 \end{cases}$$

where q is the number of allocated local networks ($q = \sum_{n \in N} Y_n$). The above equation can be solved by considering all possible permutations of allocating copies of a file to all local networks. The case in which exactly q local networks are allocated at least one copy is called the *q-network-allocation* and can be solved easily.

If τ_q denotes the total communication cost when all queries are internet queries, and the copies in all q local networks are updated by internet communications in a q-network-allocation, the equation is

$$\tau_q = \sum_{n \in N} \sum_{s \in S} \left[\lambda_{n,s} \left[d_\alpha + 2 \cdot d_\beta \right] + \phi_{n,s} \left[d_\alpha' + (q+1) \cdot d_\beta' \right] \right]. \quad (22)$$

When at least one copy is allocated to network n, the cost reduction of remote queries and updates is

$$\sum_{s \in S_n} \left[\lambda_{n,s} (d_\alpha + d_\beta) + \phi_{n,s} d_\beta' \right]. \quad (23)$$

The first term in the summation is the reduction for the cost of remote queries. The second term is the reduction for the cost of neighboring updates since only $(q-1)$ copies are stored in other local networks. For each copy allocated to site (n,s), a cost of $\left[\lambda_{n,s} d_\beta - \sigma_{n,s}\right]$ can be saved as well. Obviously, the optimum allocation for each network is to allocate a copy to those sites with positive $\left[\lambda_{n,s} d_\beta - \sigma_{n,s}\right]$. If there is no such site, it is best to either not allocate the file to the network or allocate a copy to the site with the maximum $\left[\lambda_{n,s} d_\beta - \sigma_{n,s}\right]$ if

$$\sum_{s' \in S_n} \left[\lambda_{n,s} (d_\alpha + d_\beta) + \phi_{n,s} d_\beta' \right] + \max_{s \in S_n} \left[\lambda_{n,s} d_\beta - \sigma_{n,s} \right] > 0.$$

If fewer than q networks can be allocated, the q-network-allocation case should be discarded since it is dominated by either the $(q-1)$-network case or the single-network case. The optimal solution for multiple-network allocation can be obtained by comparing the costs of all q-network cases. As with single-network allocation, multiple-network allocation should be discarded if there are one or fewer networks to which copies are allocated. The cost of an optimum allocation is then

$$C(I) = \tau - \sum_{n \in N} Y_n \left[\sum_{s \in S_n} \left[\lambda_{n,s} (d_\alpha + d_\beta) + \phi_{n,s} d_\beta' \right] \right] - \sum_{s \in S} \left[\lambda_{n,s} d_\beta - \sigma_{n,s} \right] I_{n,s}.$$

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

The solution algorithm is summarized in algorithm 2BCSFAP 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 beyond what a system can handle in a reasonable amount of time and when the access locality changes rapidly. It is not difficult to obtain good heuristics for file allocation problems. Performance, however, is generally difficult to evaluate. In this section we will discuss several approaches to this problem and propose a heuristic solution with guaranteed performance.

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

Approach 1. Allocate each file independently until all files are allocated. Reallocation of some files may be necessary during the later stages of this process.

Approach 2. Allocate one copy of each file to the system to generate an initial solution, then improve the solution by adding further copies. Allocation of extra copies to each site can be determined by a standard zero-one knapsack problem.

Approach 3. Allocate files to each site as a standard zero-one knapsack problem. This may result in one or more files which are not allocated to any place in the system. Therefore, it is necessary to identify the missing files and allocate them so that at least one copy exists.

Approach 4. A combination of the above approaches.

It is very difficult to compare these approaches. One problem of Approach (1) and (3) is that they may generate infeasible solutions in the fist phase such that they rely on a reallocation procedure in the second phase to make infeasible solutions feasible. The transformation may be difficult or even impossible. In contrast, one problem with Approache (2) is that the advantage of allocating a file is unknown

Theorem 1. The largest possible error that algorithm HC2BFAP can produce is no greater than

(a) $\sum_{n\in N}\sum_{s\in S_n}\left[P'_{n,s}+\sum_{f\in F}\overline{p^f_{n,s}}\right]$, or

(b) $P^*\left[\sum_{n\in N}\sum_{s\in S_n}\left[\varepsilon_{n,s}\frac{P_{n,s}(KNAP)}{P^*}\right]\right]$ if $\sum_{f\in F}\overline{p^f_{n,s}} < P'_{n,s}, \ \forall f$,

where $\varepsilon_{n,s}=\frac{(q+1)L}{C_{n,s}}$, $q=\lceil\frac{|F|}{|S|}\rceil$, and $L=\max\{l^f\}$.

Proof. The proof for this theorem is not shown here due to space limitation[19]. □

To estimate the error using Theorem 1, several instances of the knapsack problem were evaluated. Although knapsack problems are not difficult to solve, part (b) may be easier to use than part (a) in some cases. To use part (b) of Theorem 1, the summation of $\overline{p^f_{n,s}}$ is required to be small when compared to $P'_{n,s}$. This may not be true in practice. However, after examining the definition of $p^f_{n,s}(1)$, $p^f_{n,s}(2)$, $p^f_{n,s}(3)$, and $p^f_{n,s}(4)$, we found that $p^f_{n,s}(4)$ occurs most frequently. Thus, $P_{n,s}(KNAP)$ can be considered to be the maximum profit of a knapsack problem instance that takes $p^f_{n,s}(4)$ as the profit of allocation f to site (n,s). The resulting error estimation will be close to the real upper bound. Furthermore, if the difference between P^* and $P^*(KNAP)$ is small, $\left[\varepsilon_{n,s}\frac{P_{n,s}(KNAP)}{P^*}\right]$ can be replaced by $\left[\varepsilon_{n,s}\frac{P_{n,s}(KNAP)}{P^*(KNAP)}\right]$, making the solution even easier. The relationship between the estimated upper bound of the deviation and the various parameters is demonstrated in Figure 2 on the following conditions:

(a) $P_{n,s}(KNAP)$ is similar in all sites.

(b) $\sum_{f\in F}\overline{p^f_{n,s}} < P'_{n,s}, \ \forall f$.

(c) $P^*(KNAP)\cong P^*$.

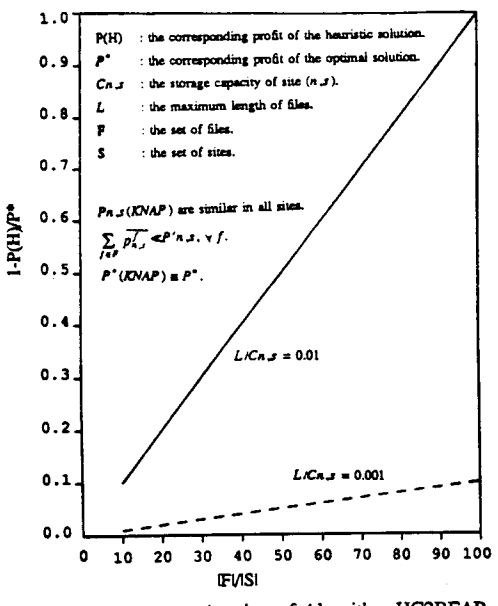


Figure 2 Error estimation of Algorithm HC2BFAP.

5. Conclusion

This article has dealt with file allocation problems on homogeneous two-level local multiaccess networks in the following way. The communication cost for reading a file is reduced to a three-value variable, making the problem simple in comparison to similar problems on general networks. The general problem is transformed into a linear zero-one integer programming problem which is solved by a search algorithm based on Balas's additive algorithm. For a special case, in which the communication overhead is dominant, the simple file allocation problem becomes polynomially solvable and is solved in $O(|S|\,|N|)$ time complexity, where $|S|$ is the total number of sites in the network and $|N|$ is the total number of local networks in the system. The general file allocation with storage constraints is still NP-hard. Since these problems are generally difficult to solve in large systems, an efficient heuristic algorithm with guaranteed performance is proposed.

References

1. B. W. Wah, "File Placement on Distributed Computer Systems," *IEEE Computer*, vol. 17, no. 1, pp. 23-32, Jan. 1984.
2. W. W. Chu, "Multiple File Allocation in a Multiple Computer System," *IEEE Trans. on Comp.*, vol. C-18, no. 10, pp. 885-889, Oct. 1969.
3. L. W. Dowdy and D. V. Foster, "Comparative Models of the File Assignment Problem," *ACM Computing Surveys*, vol. 14, no. 2, pp. 287-313, June 1982.
4. Yao-Nan Lien and Benjamin Wah, "Design and Performance Study of DDBLMN: A Distributed Database on a Local Computer System," *Proceedings of HICSS-20, Volume 2*, pp. 407-418, Kona, Hawaii, Jan. 1987.
5. B. W. Wah 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.
6. C. V. Ramamoorthy and B. W. Wah, "The Isomorphism of Simple File Allocation," *IEEE Trans. On Computers*, vol. C-32, no. 3, pp. 221-232, March 1983.
7. B. W. Wah and Y. N. Lien, "Design of Distributed Databases On Local Computer Systems With A Multiaccess Network," *IEEE Trans. On Software Engr.*, vol. SE-11, no. 7, pp. 606-619, July 1985.
8. James H. Morris, Mahadev Satyanarayanan, Michael H. Conner, John H. Howard, David S. H. Rosenthal, and F. Donelson Smith, "ANDREW : A Distributed Personal Computing Environment," *Communications of the ACM*, vol. 29, no. 3, pp. 184-201, March 1986.
9. David Notkin, Norman Hutchinson, Jan Sanislo, and Michael Schwartz, "Heterogeneous Computing Environments: Report on The ACM SIGOPS Workshop on Accommodating Heterogeneity," *CACM*, vol. 30, no. 2, pp. 132-140, Feb. 1987.
10. Gia Toan Nguyen, "Distributed Query Management for a Local Network Database System," *Proceedings of the Second International Conference on Distributed Computing Systems Paris*, April 1981.
11. M. G. Gouda and U. Dayal, "Optimal Semi-join Schedules for Query Processing in Local Distributed Database Systems," *Proc. ACM SIGMOD Conference*, pp. 164-175, May 1981.
12. Larry Kerschberg, 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. S. Masuyama, 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.
14. G. M. Sacco, "Distributed Query Evaluation In Local Area Networks," *Proc. of Int'l Conf. On Data Engineering*, pp. 510-516, Los Angeles, CA, April 1984.
15. A. R. Hevner, 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.

(a) The per-unit cost of inter-site communication is site independent due to the multiaccess/broadcast capability.

(b) The communication cost of updating a file is independent of the number of times that a file is replicated in a broadcast network. (This does not include the associated computational overhead.)

Some networks, such as Ethernet, have the following additional property.

(c) Since the message order is preserved in all sites, synchronization and consistency control is simplified.

Although the file allocation problem does not depend on property (c), it is important to point out that this property is very useful in a distributed environment since it makes the synchronization, concurrency control , and dynamic query processing much easier than that in point-to-point networks[10, 11, 12, 13, 14, 7, 15, 16, 4]. The communication overhead is assumed to be proportional to the volume of data to be transmitted. Although this assumption may not be true for short messages in which the constant overhead in each data unit may be significant when compared to overhead for the actual data, it is a reasonable assumption when a large volume of data is transmitted.

2.2. Homogeneous Two-Level Local Broadcast Networks

The network with which we are concerned in this paper consists of three components: *a backbone network*, a set of *local networks*, and a set of homogeneous *local systems* (sites). The per-unit disk-access costs and the processing costs are identical for all sites. Both the backbone network and the set of local networks are one-level local broadcast networks as described in the last section, except that the backbone network may have a higher data rate. The networks are organized into two-levels: each local system is connected to a local network; and all local networks are connected to the backbone network through gateways. Properties (b) and (c) of one-level networks are not preserved in two-level broadcast 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 broadcast network is shown in Figure 1.

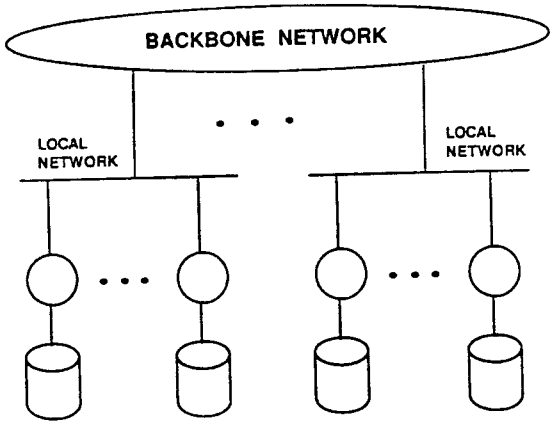


Figure 1 An example of Two-level Local Multiaccess Networks.

Since transmissions on different networks are independent, the total bandwidth is higher than the bandwidth of a single local network. Buffering may be necessary in the gateways that connect local networks to the backbone network. The local network to which a particular site is connected is called the *home network* of the site. With respect to this site, other local networks are called *remote networks*. 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 network, and an *internet query* (resp., *update*) if it is directed to files in a remote network. Neighboring and internet queries (resp., *update*) can be called *remote* queries (resp., *updates*).

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

2.3. 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 disk overhead depends on the number of copies in an update, it is usually insignificant when compared to the overhead in identifying target data and concurrency control. The problem can therefore be simplified by not considering disk overhead. Since accessing local data does not involve networking, it can be said to involve no overhead.

The cost consideration of data movement on a two-level network is different from the consideration of a wide-area-network since the communication cost is dominant in the latter case. The cost parameters in the proposed model consist of overheads in processors, local networks, backbone networks, and gateways. Since local sites and local networks are homogeneous, a unit cost for every "resource" involved in moving a data unit is counted.

Notation definitions are as follows.

N - set of local networks

S - set of sites in the system

S_n - set of sites in the local network n

F - set of files in the database, $|F| = m$

(n,s)- site s of network n, $s \in S_n$

$\lambda^f_{n,s}$ - query load originating at site (n,s) for file f per unit time

$\phi^f_{n,s}$ - update load originating at site (n,s) for file f per unit time

d_α - per-unit cost on the backbone network for a remote query

d_β - per-unit cost on the local network for a remote query

d_p - per-unit processing cost for intersite communications in a remote query

d_g - per-unit gateway overhead for intersite communications in a remote query

d_α' - per-unit cost on the backbone network in a remote update

d_β' - per-unit cost on the local network in a remote update

d_p' - per-unit processing cost for intersite communications in a remote update

d_g' - per-unit gateway overhead for intersite communications in a remote update

$\sigma^f_{n,s}$ - storage cost per unit time of file f at site (n,s)

l^f - length of file f

$C_{n,s}$ - storage capacity at site (n,s)

I - an allocation

$$I^f_{n,s} = \begin{cases} 1 & \text{if file } f \text{ is allocated to site } (n,s) \\ 0 & \text{otherwise} \end{cases}$$

$$Y^f_n = \begin{cases} 1 & \sum_{s \in S_n} I^f_{n,s} \geq 1 \\ 0 & \text{otherwise} \end{cases}$$

The cost to transmit a request to the site containing the requested data is assumed nil; only transmission of the requested data is counted. Since the protocols for handling queries and updates may be different,

Problem Q

$$\text{minimize } C(I) = \sum_{f,n,s} (a^f_{n,s} Y^f_n + b^f_{n,s} Z^f_{n,s} + c^f_{n,s} W^f_{n,s} + d^f_{n,s} I^f_{n,s} + e^f_{n,s}) \quad (10)$$

$$\text{subject to } \sum_{n\in N}\sum_{s\in S_n} I^f_{n,s} \geq 1, \quad \forall f\in F; \quad (11)$$

$$\sum_{f\in F} l^f I^f_{n,s} \leq C_{n,s}, \quad n\in N, s\in S_n; \quad (12)$$

$$M\cdot Y^f_n - \sum_{s\in S_n} I^f_{n,s} \geq 0, \quad \forall n\in N, f\in F; \quad (13)$$

$$\sum_{s\in S_n} I^f_{n,s} - Y^f_n \geq 0, \quad \forall n\in N, f\in F; \quad (14)$$

$$M(1 - Z^f_{n,s} - W^f_{n,s}) - \sum_{n'\in N} Y^f_{n'} + 1 \geq 0, \quad (15)$$

$$\forall n\in N, s\in S, f\in F;$$

$$M(1 - Z^f_{n,s}) - \sum_{s\in S_n} I^f_{n',s} + 1 \geq 0, \quad \forall n,s,f; \quad (16)$$

$$I^f_{n,s} - Z^f_{n,s} \geq 0, \quad \forall n\in N, s\in S, f\in F; \quad (17)$$

$$Z^f_{n,s} + W^f_{n,s} \leq \sum_{s\in S_n} I^f_{n,s}, \quad \forall n\in N, s\in S, f\in F; \quad (18)$$

and Y^f_n, $Z^f_{n,s}$, $W^f_{n,s}$, and $I^f_{n,s}$ are 0-1 valued, where

$$a^f_{n,s} = -(d_\alpha + d_\beta + 2d_g)\lambda^f_{n,s} + (|N|-1)(d_\beta' + d_g')\phi^f_{n,s},$$

$$b^f_{n,s} = -(d_\alpha' + d_\beta' + d_p' + d_g')\phi^f_{n,s},$$

$$c^f_{n,s} = -(d_\alpha' + d_g')\phi^f_{n,s},$$

$$d^f_{n,s} = \sigma^f_{n,s} - (d_\beta + 2d_p)\lambda^f_{n,s} + (|S|-1)d_p'\phi^f_{n,s},$$

$$e^f_{n,s} = (d_\alpha + 2d_\beta + 2d_p + 2d_g)\lambda^f_{n,s} + (d_\alpha' + d_\beta' + d_p' + d_g')\phi^f_{n,s},$$

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 networks 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.

Small to medium sized problems can be solved by special zero-one algorithms. Balas's additive algorithm[17] 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[18].

3.2. File Preallocation and Reduction of Problem Size

In many real systems, the number of files may make it impossible to obtain an optimum solution 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[7] may be modified to use a heuristic knapsack algorithm instead of the optimal algorithm.

(b) For those systems operating under steady state, there is no need 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 (less than 1/30).

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

4. File Allocation on Communication Dominated Systems

In communication dominated systems, processing cost is ignored, and $d^f_{n,s}(I)$ and $d'^f_{n,s}(I)$ read as follows.

$$d^f_{n,s}(I) = \begin{cases} 0 & I^f_{n,s} = 1 \\ d_\beta & Y^f_n = 1 \text{ and } I^f_{n,s} = 0 \\ d_\alpha + 2\cdot d_\beta & Y^f_n = 0 \end{cases}$$

$$d'^f_{n,s}(I) = \begin{cases} 0 & I^f_{n,s} = 1 \text{ and } \sum_{n'\in N}\sum_{s'\in S_{n'}} I^f_{n',s'} = 1 \\ d_\beta' & \sum_{\substack{s'\in S_n \\ s'\neq s}} I^f_{n,s'} \geq 1 \text{ and } \sum_{n'\in N} Y^f_{n'} = 1 \\ d_\alpha' + \left[\sum_{\substack{n'\in N \\ n'\neq n}} Y^f_{n'} + 1\right] d_\beta' & \text{otherwise.} \end{cases}$$

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 linealization 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.

$$C(I) = \sum_{n\in N}\sum_{s\in S_n} \left[\lambda_{n,s} d_{n,s}(I) + \phi_{n,s} d'_{n,s}(I) + \sigma_{n,s} I_{n,s}\right]$$

In a communication dominated system, the update cost is dependent on the number of local networks that have copies of the file, but not on the number of sites within each network that have copies of the file. Only one transmission on each network is needed to que y or update a file. As a result, the problem is polynomially solvable with respect to the total numbers of sites and local networks.

In solving the problem, three allocation alternatives are considered.

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

(b) *Single-network allocation:* Multiple copies of the file are allocated to a single local network after determining the best location. No internet update is needed for the changes initiated by a site in the network in which the file resides.

(c) *Multiple-network allocation:* The file is allocated to more than one local network. Update cost depends on the number of local networks that have copies of the file.

In each of these alternatives, placement is a prime consideration. The optimal solution for a given problem can be discovered by comparing the total cost for each alternative. An analysis of the equations for comparing each option follows.

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

$$\tau = \sum_{n\in N}\sum_{s\in S} \left[\lambda_{n,s}\left[d_\alpha + 2\cdot d_\beta\right] + \phi_{n,s}\left[d_\alpha' + 2\cdot d_\beta'\right]\right].$$

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

$$\Delta_{n,s} = \sum_{s'\in S_n} \left[\lambda_{n,s}(d_\alpha + d_\beta) + \phi_{n,s}(d_\alpha' + d_\beta')\right] + \left[\lambda_{n,s} d_\beta + \phi_{n,s} d_\beta'\right] - \sigma_{n,s}. \quad (19)$$

The first term in the above equation is the reduction of cost from neighboring queries and updates. The second term is additional reduction from requests initiating from site (n,s). The third term is the cost of storing the file. Obviously, the file should be allocated to the site with the maximum $\Delta_{n,s}$. The cost of single-copy allocation is then

$$C(I) = \tau - \max_{\substack{s\in S \\ n\in N}} \left[\Delta_{n,s}\right]. \quad (20)$$

before the number of allocated copies is known. However, it is reasonable to assume that the number of single-copy and single-network files is small after an appropriate preallocation; hence, the second approach can be adopted by simply assuming that all files are allocated to more than one network. The following heuristic is derived from Approach (2).

Heuristic Algorithm HC2BFAP

(a) Assume that all files are allocated to more than one network and that the profit of allocating file f to site (n,s) is $\left[\lambda^f_{n,s} d_\beta - \sigma^f_{n,s}\right]$.

(b) An arbitrary site is selected for allocation of each file such that total length of all files in a given site is no greater than $\lceil \frac{|F|}{|S|} \rceil L$, where L is the maximum length of all files.

(c) For each site, additional files are allocated using a standard 0-1 knapsack algorithm to fill the remaining storage capacity. □

For simplicity, we assume the storage capacity of every site to be no less than $\lceil \frac{|F|}{|S|} \rceil L$. Otherwise, Step 2 must be modified. Under this assumption, Step 1 is always feasible since each site can hold at least $\lceil \frac{|F|}{|S|} \rceil$ files. In Step 2, the arbitrary site in which the first copy of a file is stored can be selected using the single-copy allocation algorithm of SFAP developed in Section 3.1. In step 3, either optimal or heuristic 0-1 knapsack algorithms can be used. Pseudo polynomial-time optimal algorithms are useful in finding optimal solutions. For our purposes here, an optimal algorithm is used.

Performance of algorithm HC2BFAP. Although HC2BFAP is not the best of its type, its performance can be estimated based on the properties of knapsack problems developed elsewhere[19]. 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 classes of items are allocated to knapsacks with the goal of maximizing the total profit. Each class has an unlimited number of identical items. At most, one item from each class can be allocated to a knapsack; and at least one item from each class must be allocated to one of the knapsacks. When allocating an item to a particular knapsack, one of the following profits is possible.

(a) $$p^f_{n,s}(1) = \sum_{s' \in S_n} \left[\lambda^f_{n,s}\cdot(d_\alpha + d_\beta) + \phi^f_{n,s}\cdot d_\alpha'\right] + \left[\lambda^f_{n,s} d_\beta + \phi^f_{n,s} d_\beta' - \sigma^f_{n,s}\right] - \sum_{\substack{n' \in N \\ n' \neq n}} \sum_{s' \in S_{n'}} \phi^f_{n',s'}\cdot d_\beta',$$

(b) $$p^f_{n,s}(2) = \sum_{s' \in S_n} \left[\lambda^f_{n,s}\cdot(d_\alpha + d_\beta) + \phi^f_{n,s}\cdot d_\alpha'\right] + \left[\lambda^f_{n,s} d_\beta - \sigma^f_{n,s}\right] - \sum_{\substack{n' \in N \\ n' \neq n}} \sum_{s' \in S_{n'}} \phi^f_{n',s'}\cdot d_\beta',$$

(c) $$p^f_{n,s}(3) = \sum_{s' \in S_n} \left[\lambda^f_{n,s}\cdot(d_\alpha + d_\beta)\right] + \left[\lambda^f_{n,s} d_\beta - \sigma^f_{n,s}\right] - \sum_{\substack{n' \in N \\ n' \neq n}} \sum_{s' \in S_{n'}} \phi^f_{n',s'}\cdot d_\beta',$$

(d) $$p^f_{n,s}(4) = \left[\lambda^f_{n,s} d_\beta - \sigma^f_{n,s}\right].$$

To calculate the overall cost from the profit in the transformed problem, the following fixed cost is added to each instance of the problem:

$$\tau = \sum_{f,n,s} \left[\lambda^f_{n,s}\left[d_\alpha + 2\cdot d_\beta\right] + \phi^f_{n,s}\cdot(d_\alpha' + d_\beta')\right].$$

This is the cost when all queries and updates are assumed to involve internet communications. This does not present a complete picture of total communication costs, however, since only the communication cost on the home network of the originating site is counted for each update. Whenever a file is allocated to a site, there is a certain cost reduction which can be mapped to the profit in MCMKP. Maximizing the profit of an MCMKP instance minimizes the cost of the corresponding instance in FAP. The profit $p^f_{n,s}(1)$ is the cost reduction when allocating file f to site (n,s) as the unique copy in an allocation. The third term in $p^f_{n,s}(1)$ is the communication cost on network n for updates generated by remote networks. This cost is not counted in τ. Profit $p^f_{n,s}(2)$ is the cost reduction when file f is allocated to site (n,s) where network n is the only network that has the file and site (n,s) is the first site in the network to have the file. The first term is the cost reduction from neighboring queries and updates generated from the local network. The second term is the additional cost reduction from local queries and updates. Profit $p^f_{n,s}(3)$ is the cost reduction when allocating file f to site (n,s) where more than one network has the file and site (n,s) is the first site in network n to have the file. Profit $p^f_{n,s}(4)$ is the cost reduction when allocating file f to site (n,s) where the conditions in $p^f_{n,s}(1)$, $p^f_{n,s}(2)$, and $p^f_{n,s}(3)$ are not applicable.

After the transformation, the problem becomes the maximization of the total profit since (total cost) = τ – (total profit). In the rest of this section, the minimization of the overall cost is referred to as the maximization of the total profit. The evaluation of HC2BFAP involves finding the maximum deviation between the profit generated by HC2BFAP and the profit generated by an optimal algorithm. Since it is difficult to evaluate HC2BFAP against the optimal algorithm directly, the evaluation is done by comparison to another value: the total profit of $|N|$ 0-1 knapsack problem (one for each site). In this case, each site is solved as an independent 0-1 knapsack problem. This allocation is called *many-knapsack allocation* (MANY-KNAPSACK) here for convenience. The profit of allocating file f to site (n,s) is the maximum of $p^f_{n,s}(1)$, $p^f_{n,s}(2)$, $p^f_{n,s}(3)$ and $p^f_{n,s}(4)$. We denote P^*, $P(H)$, and $P^*(KNAP)$ as the maximum profit generated by an optimum solution of FAP, by algorithm HC2BFAP, and by a MANY-KNAPSACK allocation, respectively. We also denote $P_{n,s}(KNAP)$ as the profit generated by solving the allocation of site (n,s) as a standard 0-1 knapsack problem and $P_{n,s}(H)$ as the profit produced by site (n,s) in algorithm HC2BFAP. In other words, $P^*(KNAP) = \sum_{n \in N} \sum_{s \in S_n} P_{n,s}(KNAP)$ and $P(H) = \sum_{n \in N} \sum_{s \in S_n} P_{n,s}(H)$. The following steps are then taken to complete the evaluation.

(1) The maximum difference between $P_{n,s}(H)$ and $P_{n,s}(KNAP)$ is evaluated for each site (n,s). This difference is referred to as the *maximum deviation* of site (n,s).

(2) The maximum difference between $P^*(KNAP)$ and $P(H)$ is estimated by adding the maximum deviation of all sites together.

(3) It is easy to prove that P^* is always less than or equal to $P^*(KNAP)$ since the solution of FAP is also a feasible solution of the MANY-KNAPSACK allocation[19]. Therefore, the value obtained in Step 2 is an upper bound of the deviation between $P(H)$ and P^*.

The following notations are used in the theorems derived in this section.

$$p^f_{n,s}(0) = \max\{p^f_{n,s}(1), p^f_{n,s}(2), p^f_{n,s}(3), p^f_{n,s}(4)\}.$$
$$\overline{p^f_{n,s}} = p^f_{n,s}(0) - \min\{p^f_{n,s}(1), p^f_{n,s}(2), p^f_{n,s}(3), p^f_{n,s}(4)\}.$$

Lemma 1. In a FAP instance, if the profit of allocating a file f to site (n,s) is the maximum of $p^f_{n,s}(1)$, $p^f_{n,s}(2)$, $p^f_{n,s}(3)$, $p^f_{n,s}(4)$, then the maximum profit produced by an optimal solution is no greater than the maximum profit produced by solving the allocation in each site as a standard 0-1 knapsack problem, i.e.,

$$P^* \le P^*(KNAP) = \sum_{n \in N} \sum_{s \in S} P_{n,s}(KNAP)$$

Proof. Allocating each site as a standard 0-1 knapsack problem is also a feasible solution for an optimum allocation of FAP[19]. □

$P'_{n,s}$ is the profit produced by allocating files to site (n,s) when the storage capacity of the site is $\left[1+\lceil \frac{|F|}{|S|} \rceil\right] L$ and the profit of allocating a file f to site (n,s) is $p^f_{n,s}(0)$, where L is the maximum length of all files.

16. Yao-Nan Lien, *Distributed Databases On Local Multiaccess Computer Systems*, Ph.D. Dissertation, School of Electrical Engineering, Purdue University, West Lafayette, IN, Aug. 1986.

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

18. Yao-Nan Lien, Yih-Long Chang, and Benjamin Wah, "File Allocation Problems On Homogeneous Two-Level Local Broadcast Networks," *Technical Report #OSU-CISRC-8/87-TR24*, Dept. of Comp. and Info. Sci., The Ohio State Univ., Columbus, Ohio , Aug. 1987.

19. Y.-N. Lien, "Some Properties of 0-1 Knapsack Problems," *Proc. Conference on Combinatorics and Complexity*, p. 105, Chicago, IL, 1987.

APPENDIX A

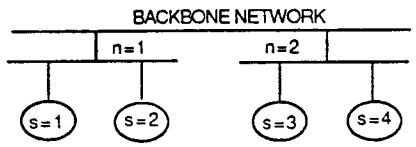
```
/*-------------------------------------------------------------------
      Algorithm 2BCSFAP: to solves SFAP
      on two-level local multiaccess networks.
-------------------------------------------------------------------*/
/* (Single-copy Allocation) */
   maxΔ = -∞
   FORALL n IN N DO {
      FORALL s IN  S_n  DO {
         IF ( Δ_{n,s} > maxΔ )      /*  Eq. (19) */
         THEN { maxΔ = Δ_{n,s}; sc.allocation = (n,s) }
      } }
   sc_cost = τ - maxΔ

/* (Single-netwotk Allocation) */
   maxΔ = -∞
   FORALL n IN N DO {
      number_of_copy = 0
      Δ_n = Σ_{s∈S_n} [λ_{n,s}(d_α+d_β)+φ_{n,s}(d_β'+d_α')]   /* Eq. (21) */
      FORALL s IN  S_n  DO {
         saving =  [λ_{n,s} d_β - σ_{n,s}]
         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] = ∅;}
      IF  (Δ_n > maxΔ)
      THEN { maxΔ = Δ_n ; sn_allocation = allocation[n]}
      } }
   sn_cost = τ - maxΔ

/* (Multiple-netwotk Allocation) */
   mn_cost = ∞
   FOR q FROM 2 TO N DO {
      number_of_network = 0
      FORALL n IN N DO {
         number_of_copy = 0
         Δ_{q,n} = Σ_{s∈S_n} [λ_{n,s}(d_α+d_β) + φ_{n,s} d_β']   /* Eq. (23)  */
         maxΔ = -∞
         FORALL s IN  S_n  DO {
            saving =  [λ_{n,s} d_β - σ_{n,s}]
            IF ( saving > 0 )
            THEN {
               Δ_{q,n} = Δ_{q,n} + 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; s_max = (n,s) }
            } /* s */
         IF ( number_of_copy = 0 )
         THEN {
            Δ_{q,n} = Δ_{q,n} + maxΔ
            IF ( Δ_{q,n} > 0 )
            THEN {
               net_allocation[n] = net_alocation [n] ∪ s_max
               number_of_network = number_of_network + 1
            } }
          ELSE number_of_network = number_of_network + 1
         } /* n */
      IF (number_of_network ≥ q)
      THEN {
         picked_network = first q networks in descending order of Δ_{q,n}
         cost[q]=Σ_{n,s} [λ_{n,s} [d_α+2·d_β]+φ_{n,s} [d_α'+(q+1)·d_β']]   /* Eq.(22) */
         FORALL n IN picked_network DO {
            q_allcation[q] = q_allocation[q] ∪ net_allocation[n]
            cost[q] = cost[q] -Δ_{q,n}
         } }
      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
L=30	λ	6	7	4	5
	φ	3	6	2	4
	σ	3	5	2	4

Single-copy allocation: $\tau = 230$, $\Delta_{n,s} = [115, 120, 78, 81]$

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

Single-network allocation: $\tau = 230$, $\Delta_n = [111, 75]$

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

Multiple-network allocation: $\tau_q = 260$ (q can only be 2 in this example.)

site 1,2 in network 1 and site 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.