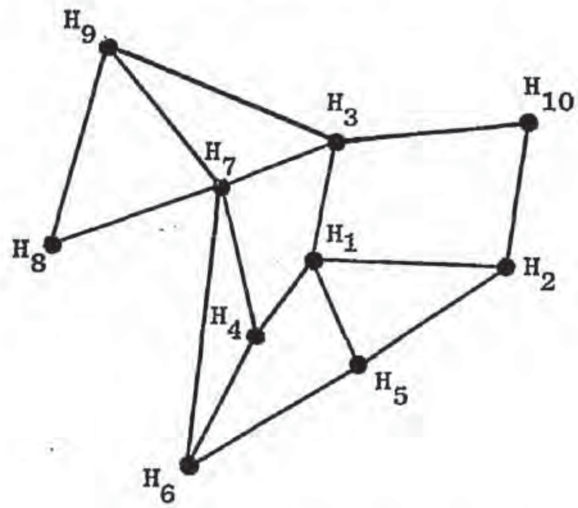


Assume that a file F is at Host 1 and that the appropriate row of U there is

File \ Host	1	2	3	4	5
F	15	30	25	25	2
⋮					

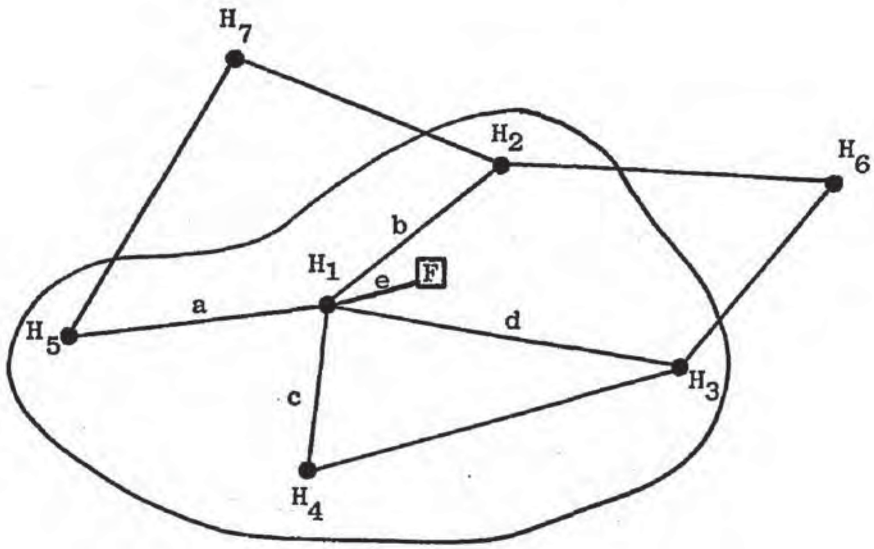
The decision to transfer the file to Host 2 would be sub-optimal if the distribution of actual requests for F were

File \ Host	1	2	3	4	5	6	7	8	9	10
F	30	30	1	5	2	0	10	0	12	0
⋮										

Assuming infinite line capacity and minimum distance (hop) routing, the cost (request rate × distance*) of keeping the file at 1 is (15 + 30 + 25 + 25 + 2) = 97 units while at host 2 it would be (30 + 15 + 2 + 10 + 2 + 30 + 36) = 125 units.

*Assume the distance for satisfying requests to locally resident files is 1/2 unit while link costs are equal to 1 unit.

Figure E.2. SUB-OPTIMAL DECISION USING ALGORITHM I.



a, b, c, d, e are link costs.

Communication costs L as seen by host 1 are:

$L(1) = e$; $L(2) = b + e$; $L(3) = d + e$;

$L(4) = c + e$; $L(5) = a + e$

Figure E.3. HOST 1 AND ITS PHYSICAL NEIGHBORS.

file after transfering it to neighbor n, is equal to the difference between the cost of keeping the file at the current host and at neighbor n. The cost of using a file i at its current residence is equal to the sum of its utilizations from different neighbors, and is equal to

$$\sum_{g=1}^{NPN} u(i,g). \qquad (E.2)$$

Now, if the file were to move to neighbor n, the cost of using it at n would be different. In figure E.3, if the file were to move from host 1 to 2, then the utilization from host 5 would increase to (L(5)-L(1)+L(2))/L(5) of its original value. This is under the assumption that traffic passes through the switching node to which host 1 is connected, and that L(1) relative to all hosts is the same i.e. the communication cost of retrieving a file from local file storage is the same for all hosts. Generalizing, we conclude that when the file moves to neighbor n, the utilization from all neighbors g, $g \neq n$, increases to (L(g)-L(1)+L(n))/L(g) times its original value. The utilization from n itself increases to L(1)/L(n) times its original value. Therefore the cost of using a file i at a neighbor n is equal to

$$u(i,n)*L(1)/L(n) + \sum_{\substack{g=1 \\ g\neq n}}^{NPN} u(i,g)*(L(g)-L(1)+L(n))/L(g). \qquad (E.3)$$

The savings in cost of using the file i when it has moved to n is given by equation (E.2) minus equation (E.3). Simplifying, elements of S are given by:

$$s(i,n) = u(i,n)*(L(n)-L(1))/L(n) - \sum_{\substack{g=1 \\ g\neq n}}^{NPN} u(i,g)*(L(n)-L(1))/L(g)] - d(i,n) \quad ;n=2,\ldots,NPN$$

$$s(i,n) = 0 \quad ;n=1 \qquad (E.4)$$

The savings are relative to the cost of keeping the file at this host.

E.2.3 Algorithm III

This algorithm is an extension of the previous one, and attempts to take into account in greater detail the path taken by file traffic from the neighbors once the file has been moved. If the traffic does not pass through the node corresponding to the original host (as in Algorithm II), then it must pass through some other node such that the cost is less (by virtue of the routing algorithms of the communication subnet). These alternate routes depend on the topology and dynamically changing load conditions. These factors could be taken into account in the following way:

$$s(i,n) = u(i,n)*(L(n)-L(1))/L(n) - \sum_{\substack{g=1 \\ g\neq n}}^{NPN} r(g,n)*u(i,g)*(L(n)-L(1))/L(g) - d(i,n) \quad ;n=2,\ldots,NPN$$

$$s(i,n) = 0 \quad ;n=1 \qquad (E.5)$$

where $r(g,\bar{n})$ is the topology and routing factor for each neighbor pair. This factor is always less than or equal to one. R is the <u>topology and routing matrix</u>. It is of dimension NPNxNPN. Further research is required to determine how these factors are calculated, i.e. should they be calculated dynamically or can they be assumed to be heuristic constants?

E.3 Discussion

It is hoped that this distributed optimization will approximate the same optimum as would be predicted by an algorithm that has complete knowledge of the topology and request rates. This is greatly dependent on the rate of variation of the request rates and the speed with which such an algorithm can track the changing optimum. Further, the topology and routing algorithms of the communication network may be such that the stepwise optimization causes a file to get stuck at a local minimum. The effect of network topologies and routing algorithms on the success of this optimization technique is a topic for future research.

E.4 Finite Storage and Differing Storage Costs

Section E.2 indicated that when the algorithm for placing the files in the network is as distributed as the step-wise optimization proposed, the cost for utilizing the file has to be based on the time to successfully transfer the file, the rate of usage from each host, and the size of the file. This is because these are the only parameters that the file system on which the file resides can measure.

Local file systems in reality will not have infinite file storage, and so this fact must be incorporated into the distributed algorithm, if possible. We describe a possible technique. When the file system

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.

WHAT WILL YOU BUILD? | **sales@docketalarm.com** | **1-866-77-FASTCASE**

fastcase®
Smarter legal research.