B.S.T.J. BRIEF

A Table Look Up Approach to Loop Switching

By L. H. BRANDENBURG and B. GOPINATH

(Manuscript received July 24, 1972)

In this paper we consider some questions of implementation of a scheme described in Section V of Ref. 1 for addressing message blocks in the Pierce loop system.[2] The scheme consists of using a stream of binary digits (0 and 1) as the address of the destination loop of a message such that the scalar product* of the address with a stream of binary digits of equal length stored at a loop gives the distance between the loop and the destination. The message is routed along a path that minimizes distance between source and destination. The binary streams used in this scheme can be obtained by factoring the distance matrix D of the graph representing the connection of loops into two binary-valued matrices P and Q such that

$$D = PQ^t,$$

where the superscript "t" stands for matrix transposition. The kth row of P represents the address to be prefixed to a message destined for loop k. The kth row of Q represents a binary sequence to be stored in loop k. The scheme discussed in Section V of Ref. 1 provides a particular way of factoring D. For completeness, we give a description of that factorization as follows.

Let s be the diameter of a graph on n vertices and let D be its distance matrix. (Note: $s = \max_{ij} D_{ij}$). We consider a PQ^t factorization of D with P and Q matrices each of order $n \times ns$. The ith row of P, P_i, is zero everywhere except in positions $(s(i-1)+1)$ to si where it is one. The ith row of Q, Q_i, is constructed as follows: for every j, there are exactly D_{ji} 1's in positions $(s(j-1)+1)$ to sj with the rest of the entries of Q_i equal to zero.

---

* The scalar product of two binary streams of equal length is the number of places they both have a "one".

$$P = \begin{array}{c} \begin{array}{cccccccc} & s & & 2s & & 3s & & ns \\ & \downarrow & & \downarrow & & & & \end{array} \\ \begin{bmatrix} 1\cdots1 & 0\cdots0 & 0\cdots0 & \cdots & 0\cdots0 \\ 0\cdots0 & 1\cdots1 & 0\cdots0 & & \\ & & & & \\ 0\cdots0 & 0\cdots0 & 0\cdots0 & \cdots & 1\cdots1 \end{bmatrix} \end{array}$$

$$Q = \begin{array}{c} \begin{array}{cccccccccc} & s & \overbrace{\quad} & & 2s & \overbrace{\quad} & & 3s & & \overbrace{\quad} & ns \\ & \downarrow & D_{12} & & \downarrow & D_{13} & & \downarrow & & D_{1n} & \downarrow \end{array} \\ \begin{bmatrix} 0\cdots0 & 1\cdots1 & \cdots & 0 & 1\cdots1 & \cdots & 0 & \cdots & 1\cdots1 & \cdots & 0 \\ 1\cdots10 & 0 & \cdots & 0 & 1\cdots1 & \cdots & 0 & & 1\cdots1 & \cdots & 0 \\ & & & & & & & & & & \\ \underbrace{1\cdots10}_{D_{n1}} & \underbrace{1\cdots1}_{D_{n2}} & \cdots & 0 & \underbrace{1\cdots1}_{D_{n3}} & \cdots & 0 & & 0 & \cdots & 0 \end{bmatrix} \end{array}$$

Obviously the length of addresses in this scheme is ns. The P matrix defined above, rows of which are addresses to be prefixed onto messages, is the same for all graphs with diameter s. The Q matrix, rows of which are stored in the loops in the network, contains the information that identifies a particular graph.

As will be discussed further, a simple mechanization of this scheme reveals that it is essentially an obvious table look up scheme. In a table look up scheme, each loop has stored the distance between it and every other loop in the networks, and each address essentially provides a signal to look up or read out a particular distance that is stored.

The above PQ^t factorization can be transformed into a table look up scheme in the following way. The integer i ($\leqq n$) uniquely specifies the ith row of P; thus, a message address need only consist of the binary representation of i, which, of course, requires at most $[\log_2 n]$* bits. Each loop, instead of storing simply a row of matrix Q, stores a binary array consisting of n rows and $[\log_2 s]$ columns. The ith row of this array contains the binary representation of the distance between the loop in question and the ith loop. The binary array can be implemented by a read only memory the ith row of which is accessed by the binary sequence representing integer i. Note that in the original PQ^t form of this scheme, both matrices P and Q were dependent on s, the diameter of the graph. In the table look up mechanization, s appears

* $[x]$ means "the least integer greater than x".

only in the dimension of the stored memory. (The authors had originally considered a table look up mechanization for minimum distance routing but had concentrated their early efforts on the more mathematically fruitful schemes discussed in Refs. 1 and 3. It is interesting to note at this point that the table look up mechanization has evolved in a natural way from a special case of one of those schemes.)

The table look up scheme meets the following important criteria:

(*i*) The simplicity of constructing addresses is important in the case of large graphs with no special structure. Even if one were to find minimum length addresses similar to the ones described in Sections I, II, and III of Ref. 1, we suspect that the construction of these addresses would be very complicated. In the present case, each address is obtained trivially as the binary representation of an integer.

(*ii*) Since present plans for length of message blocks envisage lengths of perhaps a few thousand bits, the length of the message address is an important parameter in any large loop system. The present method guarantees minimum length addresses, $[\log_2 n]$, provided that the graph is not constrained to have a particular structure.

(*iii*) The scheme can be adapted by prefixing some control digits to do alternate routing.

(*iv*) It is simple to determine the necessary changes in the stored memory required by updating (adding loops) or modifying the network, since such changes can be identified by inspection of the distance matrix.

(*v*) The speed of operation of this scheme can be very fast compared to the speed of the message.

(*vi*) If the graph is hierarchical in the sense of Pierce,[2] except that interconnections among loops at the local level are allowed, then routing can be accomplished by using the Pierce scheme at higher levels, and the present scheme at the local level.

At first it might appear that the present scheme requires longer address lengths than the Pierce addressing scheme[2] for the case of strictly hierarchical graphs. For most cases, we show this is not so. A hierarchical graph of k levels can be represented as in Fig. 1.

N_i represents the maximum of the number of branches connecting each vertex at the $(i-1)$st level, to vertices at the ith level. Pierce's scheme obviously requires addresses of length $\sum_{i=1}^{k} [\log N_i]$ bits. The present scheme requires at most

$$m = [\log (1 + N_1 + N_1N_2 + \cdots + N_1N_2 \cdots N_k)] \text{ bits.}$$

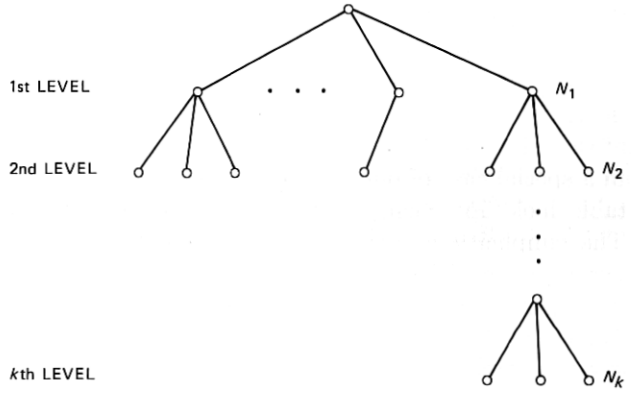

Fig. 1—A hierarchical graph of k levels.

But

$$m = \left[\log N_1 N_2 \cdots N_k \left(1 + \frac{1}{N_k} + \frac{1}{N_k N_{k-1}} + \cdots + \frac{1}{N_k N_{k_1} \cdots N_1}\right)\right].$$

If, at each level, there is at least one vertex with two or more branches descending to the next level (this will always be the case if the levels "fan out") then $N_i \geqq 2$ for each i. Then

$$m \leqq \left[\log N_1 N_2 \cdots N_k \left(1 + \frac{1}{2} + \frac{1}{2^2} + \cdots + \frac{1}{2^k}\right)\right]$$

$$\leqq [\log 2N_1 N_2 \cdots N_k] \leqq [\log N_1 N_2 \cdots N_k] + 1.$$

Therefore, if each N_i is a power of 2, then Pierce's scheme gives addresses at most one bit shorter than those of the present scheme. However, if even one of the N_i's is not a power of 2, then the present method has address lengths at most equal to those of Pierce's scheme. In fact, even if each N_i is a power of 2, the Pierce scheme has the one bit advantage if and only if every vertex at the ith level has exactly N_{i+1} branches connected to the $(i + 1)$st level, for all levels.

In order to compare the present scheme with other schemes described in Ref. 1, we assume that the important parameters are of the following orders:

s = diameter of the graph ≈ 6
n = number of loops $\approx 10^3$
length of message block $\approx 4 \times 10^3$ bits.

Then for the present scheme, the address length $\approx$ 10 bits, or 1/4 percent of the message block length. The memory requirement at each loop is n $\log_2 s$ or $\approx 3 \times 10^3$ bits. For these numbers, the presently available memories, either core or the more advanced semiconductor memories, could be used to realize very high speed routing of messages.

For the other schemes in Refs. 1 and 3, the address length was conjectured not to exceed $2(n - 1)$ bits $\approx 2 \times 10^3$ bits, or 50 percent of the message block length. The memory requirement is of the same order or about 2/3 of that of the present scheme.

With regard to time of computing distance at each junction, the table look up scheme using semiconductor memory might require on the order of 100 nanoseconds; on the other hand, the schemes discussed in Refs. 1 and 3, based on inner product or Hamming distance calculations, might require on the order of 5,000 nanoseconds.

These numbers, together with the six points listed previously, provide evidence of the practical advantage of the table look up scheme over the other schemes in Ref. 1. We have, of course, omitted discussion of such important questions as reliability and the implementation of updating the stored memory in each loop due to changes in the network. However, these questions are of equal importance in the present scheme and the schemes in Ref. 1, since the basic routing rule is the same (minimum distance) and the sizes of the stored memories are comparable as shown above.

The authors are indebted to H. S. McDonald for helpful discussions concerning some of the practicalities of implementing a loop switching system.

REFERENCES

1. Brandenburg, L. H., Gopinath, B., and Kurshan, R. P., "On the Addressing Problem of Loop Switching," B.S.T.J., *51*, No. 7 (September 1972), pp. 1445–1470.
2. Pierce, J. R., "Network for Block Switching of Data," B.S.T.J., *51*, No. 6 (July–August 1972), pp. 1133–1145.
3. Graham, R. L., and Pollak, H. O., "On the Addressing Problem for Loop Switching," B.S.T.J., *50*, No. 8 (October 1971), pp. 2495–2519.