THE BELL SYSTEM
TECHNICAL JOURNAL

DEVOTED TO THE SCIENTIFIC AND ENGINEERING
ASPECTS OF ELECTRICAL COMMUNICATION

Volume 50 October 1971 Number 8

On the Addressing Problem for Loop Switching

By R. L. GRAHAM and H. O. POLLAK

(Manuscript received March 25, 1971)

The methods used to perform the switching functions of the Bell System have been developed under the fundamental assumption that the holding time of the completed call is long compared to the time needed to set up the call. In considering certain forms of communication with and among computers the possibility arises that a message, with its destination at its head might thread its way through a communication network without awaiting the physical realization of a complete dedicated path before beginning on its journey. One such scheme has been proposed by J. R. Pierce and may be called "loop switching." We imagine subscribers, perhaps best thought of as computer terminals or other data generating devices, on one-way loops. These "local" loops are connected by various switching points to one another as well as to other "regional" loops which are in turn connected to one another as well as to a "national" loop. If a message from one loop is destined for a subscriber on another loop it proceeds around the originating loop to a suitable switching point where it may choose to enter a different loop, this process continuing until the message reaches its destination. The question naturally comes up, how the message is to know which sequence of loops to follow. It would be desirable for the equipment at each junction to be able to apply a simple test to the destination

address at the head of the message which would determine which choice the message should make at that junction.

In this paper we propose a method of addressing the loops which has several attractive features:

(*i*) *It permits an extremely simple routing strategy to be used by the messages in reaching their destinations.*
(*ii*) *By using this strategy, a message will always take the shortest possible path between any two local loops in the same region.*
(*iii*) *The method of addressing applies to any collection of loops, no matter how complex their interconnections.*

The addressing scheme we propose will be applied primarily to local loops where the mutual interconnections may be quite varied. If a certain amount of hierarchical structure is introduced into the regional and national loop structure, as suggested by J. R. Pierce,[1] it is possible to achieve addressings which are both compact and quite efficient.

I. INTRODUCTION

The methods used to perform the switching functions of the Bell System have been developed under the fundamental assumption that the holding time of the completed call is long compared to the time to set up the call. It is thus sensible to hold portions of a route while the attempt is made to establish the connection. In considering certain forms of communication with and among computers, as well as the consideration of many schemes for time division switching, the possibility arises that a message, with its destination at its head, might thread its way through a communication network without awaiting the physical realization of a complete dedicated path before beginning on its journey.

One such scheme has been proposed by J. R. Pierce,[1] and may be called "loop switching." We imagine subscribers, perhaps best thought of as computer terminals or other data generating devices, on one-way loops. If a meassage is destined for a subscriber on another loop it proceeds around the originating loop to a suitable switching point where it may choose to enter a different loop and continue the process until it reaches its destination.

The question now comes up, how the message is to know which sequence of loops to follow. A sufficiently complicated memory in the originating loop might, of course, look up an appropriate route, and then attempt to seize a complete path; but this is the old and perhaps in-

appropriate solution. It would be more convenient and sometimes preferable if the equipment at each junction could apply some simple test to the destination address at the head of the message which would determine which choice the message should make at that junction.

In the nation-wide loop switching system as conceived by Pierce, we can envisage local loops, regional loops, and a national loop. The simplest imaginable structure is one in which each local loop has an interchange only with its regional loop and with no other loop; similarly each regional loop interchanges with the national loop and otherwise only with its local loops. How does it work? Suppose that a message originates in local loop X, and has its destination in local loop Y, where X and Y may or may not be identical. When the message comes to the interchange between X and X's regional loop, it exits onto the regional loop if and only if $Y \neq X$. It later exits onto the national loop if Y's region is different from X's region; otherwise the message stays on X's regional loop until it reaches Y. Therefore what should addresses look like? We see that if a portion of the loop address represents the regional loop, and another portion the local loop, routing decisions will be made on the basis of identity or nonidentity of certain portions of the sending and the receiving addresses.

The loop configuration just described is perhaps too special to be practical. For example, it provides for no alternate routing, and for no special direct connections between two local loops with high mutual traffic. Pierce has shown how each of these difficulties can, to some extent, be alleviated. There remain, however, the further problems of the configuration of local loops belonging to a given region, and of the configuration of regional loops themselves. It is quite likely that the local loops attached to a given regional loop have many mutual switching points among themselves, so that calls within one region are not normally expected to use the regional loop. How should we address such local loops so as to make routing easy? Much of the rest of this paper will be devoted to this problem. We shall, in this and the next two sections, speak of "loops" generally, but mean a system of *local* loops as the most likely realization. We note in passing that a completely general national configuration of loops on which no hierarchical structure has been imposed will have the same addressing problem—but probably a much larger number of loops. We return to the hierarchical situation in Section IV.

In some very simple arrangements of loops it is easy to see how addressing might successfully be accomplished. Consider, for example, Fig. 1 showing four loops which touch as if they were circles of radius $\frac{1}{2}$ at

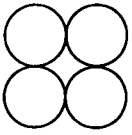


Fig. 1—Simple arrangement of loops.

the vertices of the unit square. If the address of each loop were just the two-digit numeral ij, i, $j = 0$ or 1, representing the coordinates of its center, then routing could be done in the following extremely simple manner: at each junction, go into the new loop if this decreases the Hamming distance* between where you are and your destination. If it doesn't decrease the Hamming distance, don't go. Thus, if you wish to go from loop 10 to loop 11 then the Hamming distance is 1. You will not take the exit from 10 to 00 if you reach it first, for this increases rather than decreases the Hamming distance. You will, however, exit into 11 when you reach that junction. To go from 10 to 01 either exit, to 00 or to 11, improves the Hamming distance and either routine is equally good.

A simple potential routing scheme can thus be described as follows. Each loop has a binary address, n bits long. You make an exit from one loop to another if and only if it decreases the Hamming distance between where you are and where you want to go. If several exits do the same job then each one must lead to an equally short optimal path from sending loop to receiving loop. Furthermore, the number of loops traversed should, if possible, be exactly the Hamming distance between sender and receiver, with each transfer decreasing the distance from the receiver by exactly 1.

Can such an addressing scheme be devised for every collection of loops with whatever adjacency structure? A little reflection shows that there will certainly be difficulties. Let's think of the collection of loops abstractly as a graph, with each loop a vertex, and two vertices connected if and only if the two loops have a mutual transfer point. Thus, the graph of the previous example is as shown on Fig. 2. We have numbered each vertex with a pair of binary digits so that adjacent vertices differ in exactly one position, the number of edges required to pass from one point to another is exactly the Hamming distance between the corresponding numberings, and all shortest paths between two points are achieved by following routes of decreasing Hamming distance to the destination. Another example (Fig. 3): if we wanted a collection of six

* The Hamming distance between two n-place binary numbers is the number of places in which they differ.

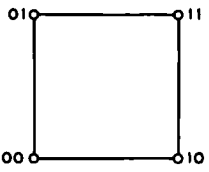


Fig. 2—Graph of Fig. 1.

loops arranged cyclically we could use the numbering 000, 100, 101, 111, 011, and 010. We see that we are looking for a closed path on the 3-dimensional cube with the additional property that two points are exactly as far apart in Hamming distance as the number of edges to be traversed between them—otherwise, the routing logic would be ruined. Thus we can use the realization for a cycle of six loops shown in Fig. 4a. The realization shown in Fig. 4b, however, would *not* be a valid solution. In this latter picture, 100 and 110 have Hamming distance 1 and therefore should be directly connected. The path between them, however, has length 3, the first link out increases rather than decreases Hamming distance, and therefore would not represent a useful addressing scheme.

We thus see a difficulty caused by points coming too close together on the cube for the addressing scheme to work, but there are even deeper difficulties. Suppose we wish to construct an addressing scheme for a system consisting of three pairwise adjacent loops (see Fig. 5). This can never be drawn on a cube of *any* dimension. For any closed path of edges on a cube has even length, and 3 is odd. Is the scheme therefore kaput?

Not quite. We can still imagine the 3-cycle embedded on a cube in an appropriate dimension (in this case a square) if we are willing to generalize what we mean. We shall attach to A the code 00, B the code 10, and to C *both* 11 and 01. We shall denote the pair 11 and 01 by the symbol $d1$, where d means "don't care." Hamming distance between two n-tuples of 0's ,1's, or d's is computed by crediting 1 for every position at

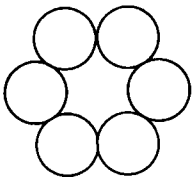


Fig. 3—Cyclic arrangement of six loops.

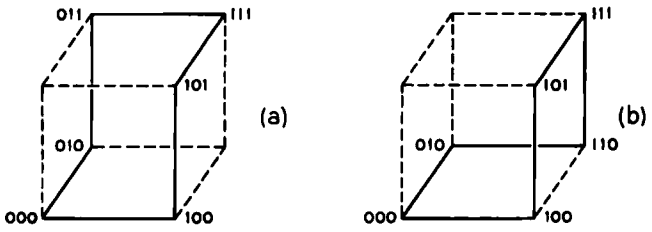


Fig. 4—Realizations for a cycle of six loops: (a) a valid solution; (b) not a valid solution.

which one n-tuple has a 0 and the other a 1, and 0 for every other position. Thus, the Hamming distance between 01d1d0 and 11d010 is 2, with the contributions coming from the first and fourth positions. With this convention, the Hamming distance between any two of the three addresses 00, 10, and d1 is certainly 1, and correct routing still consists exactly of decreasing by 1 the Hamming distance at each junction at which a transfer is made.

We now have a number of fundamental questions to answer. Can *every* collection of loops be numbered by assigning to each loop an address consisting of a sequence of 0's, 1's, and d's? We require that every shortest route between two loops can be found automatically by moving from a loop to an adjacent one if and only if this decreases the Hamming distance to the final destination by 1. How many bits long would such an address have to be? Let's state right away the fundamental theorem of this paper: Every collection of n loops, with maximum distance s between any two loops, can indeed be realized by giving each loop an address of no more than $s(n-1)$ 0's, 1's, or d's. In fact, we know of no example where more than $(n-1)$ "bits" are needed, and we shall give a construction that has found addresses no more than $n-1$ bits long in every case on which it has been tried. The construction, however, is not quite an algorithm and we do not have a proof that it can always be done with as few as $n-1$ bits.

Fig. 5—Three pairwise adjacent loops.

How is this routing algorithm going to work in practice? Here we have only the very earliest and simplest suggestions. The basic idea of the scheme is to obtain the greatest possible simplicity of routing strategy at the expense of the length of the loop address. Thus, for example, you could physically realize addresses consisting of 0's, 1's, and d's by encoding 0 as 00, 1 as 01, and d as either 10 or 11. The logic then says: If the $2k$ − 1st digit of both addresses is 0 then compute the Hamming distance between the $2k$th digits. If the $2k$ − 1st digit of either address is 1, ignore it. Add up over all k, and see if going into the new loop decreases Hamming distance to the destination. This could be very easy to mechanize; the arbitrary bit following a 1 in an odd position could be used for parity checks or other purposes.

It is not immediately clear who assigns the loop address to an individual message. The "phone book" may contain a shorter code that is translated in the first junction you come to, or the sending computer itself may use the destination's correct loop address. This problem is connected with that of system growth. How many numbers do you have to change if a loop is added to the system? The consequent desire for a hierarchical loop address structure is to a large extent fulfillable and will be discussed in Section IV.

Before we proceed with the general theory, let's see how a particular and not so simple example works out. Thus, consider the system of loops in Fig. 6. The distance between pairs of vertices is given by the following (symmetric) table:

	A	*B*	*C*	*D*	*E*	*F*
A	0	2	1	3	1	2
B	2	0	2	1	1	2
C	1	2	0	2	1	1
D	3	1	2	0	2	1
E	1	1	1	2	0	2
F	2	2	1	1	2	0

We shall assign a sequence of five 0's, 1's, and d's to each vertex in such a way that the Hamming distance between the 5-tuples corresponding to two vertices is exactly the distance in the table. One solution, as the reader should verify, is the following:

Fig. 6—System of loops.

A—1111d

B—001dd

C—11d0d

D—000d1

E—10dd0

F—010dd

In the sequel, we shall see how such a solution can in fact be found for every possible system of loops. A really surprising amount of interesting mathematics seems, at present, to be involved in the problem.

In order to see how a set of satisfactory loop addresses can always be constructed, let us analyze the previous example in more detail. The first column of the solution is:

A—1

B—0

C—1

D—0

E—1

F—0

We see that A, C, and E have the value 1 at this coordinate while B, D, and F have the value 0. Thus, this coordinate will contribute a 1 to the Hamming distance from any of ACE to any of BDF. We may denote

this as $ACE \times BDF$. Therefore, the first column makes the following contribution to the overall distance matrix.

	A	B	C	D	E	F
A	0	1	0	1	0	1
B	1	0	1	0	1	0
C	0	1	0	1	0	1
D	1	0	1	0	1	0
E	0	1	0	1	0	1
F	1	0	1	0	1	0

The second column may be written as $ACF \times BDE$ and contributes the following to the distance matrix.

	A	B	C	D	E	F
A	0	1	0	1	1	0
B	1	0	1	0	0	1
C	0	1	0	1	1	0
D	1	0	1	0	0	1
E	1	0	1	0	0	1
F	0	1	0	1	1	0

The first two columns (i.e., coordinates) then contribute the sum of the previous matrices to the distance matrix.

	A	B	C	D	E	F
A	0	2	0	2	1	1
B	2	0	2	0	1	1
C	0	2	0	2	1	1
D	2	0	2	0	1	1
E	1	1	1	1	0	2
F	1	1	1	1	2	0

The third column is:

A—1

B—1

C—d

D—0

E—d

F—0

It will contribute 1 between A or B and D or F. Since C and E have the third coordinate value d, it cannot contribute to the Hamming distance from C or E to any other point. We can write $AB \times DF$ and obtain the following contribution to the distance matrix:

	A	B	C	D	E	F
A	0	0	0	1	0	1
B	0	0	0	1	0	1
C	0	0	0	0	0	0
D	1	1	0	0	0	0
E	0	0	0	0	0	0
F	1	1	0	0	0	0

The first three columns (coordinates) thus contribute the following to the distance matrix:

	A	B	C	D	E	F
A	0	2	0	3	1	2
B	2	0	2	1	1	2
C	0	2	0	2	1	1
D	3	1	2	0	1	1
E	1	1	1	1	0	2
F	2	2	1	1	2	0

The last two columns are $A \times C$ and $D \times E$ respectively. If the cor-

responding 1's are added to the distance matrix, we obtain the matrix of our example. We see, therefore, that we can think of the distance matrix for our example as generated by the sum of the products $ACE \times BDF$, $ACF \times BDE$, $AB \times DF$, $A \times C$, $D \times E$. Notice that in each product we assign a 0 to each element of one multiplier, a 1 to each element of the other, and a d to any possible multiplier which does not occur. Which set you make 0 and which set you make 1 does not matter. If we carry this out we obtain the coordinates for A through F given previously.

The same mathematics works in general. Take the system of loops for which we wish to find an addressing scheme, and find the abstract graph in which each loop represents a vertex and two vertices are connected if and only if the loops touch. Now write down the (symmetric) distance matrix for this graph. If the vertices of the graph are $A_1, A_2, \cdots, A_n$, and the Hamming distance between A_i and A_j is d_{ij}, then we may take d_{ij} copies of $A_i \times A_j$ and then sum over all i and j. The contribution to the address of each A_k will be d_{ij} coordinates 1 to A_i, 0 to A_j, and d to all other vertices. Therefore, the total contribution to the distance matrix will be d_{ij} in the (i, j) position, and 0 everywhere else. Thus, the resulting complete set of coordinates for the A_k will consist of 0's, 1's, and d's calculated from each necessary copy of each $A_i \times A_j$ and will produce the desired distance matrix.

This proves that the addressing scheme is always possible, but we have used a ridiculously large number of coordinates, perhaps

$$\frac{sn(n-1)}{2}$$

where s is the largest point-to-point distance in the distance matrix. We can save a factor $n/2$ if we take

$$A_1 \times (A_2, A_3, \cdots, A_n) + A_1 \times (A_{i_1}, A_{i_2}, \cdots, A_{i_{k_1}}) + A_1 \times (A_{j_1}, \cdots, A_{j_{k_2}}) + \cdots$$

where $A_{i_1}, \cdots A_{i_{k_1}}$ are all those vertices for which $d_{1,i_t} \geqq 2$, $A_{j_1}, \cdots A_{j_{k_2}}$ are all those vertices for which $d_{1,j_t} \geqq 3$, etc. We then repeat for $A_2 \times (A_3, \cdots A_n) \cdots$, and so on up to $d_{n-1,n}$ copies of $A_{n-1} \times A_n$. This time we have at most $s(n-1)$ products, and therefore have found a set of at most $s(n-1)$ coordinates for the A_i such that loop addressing will work in the desired way. We have proved:

Theorem 1: Given any system of n loops so that the maximum distance between any two loops is s, a system of addresses such that every minimal path between loops is obtained by switching to an adjacent loop if and only if

the Hamming distance to the destination is decreased by 1 can always be found. The length of each address can be taken to be no more than $s(n-1)$.

Let us remark right away that we believe the right answer to be $(n-1)$ rather than $s(n-1)$. We have no proof and we have no counterexamples. We will, however, prove the following theorems in the sequel.

Theorem 2: If the abstract graph of the loop system is the complete graph on n vertices, then addresses of length $(n-1)$ are best possible.

Theorem 3: If the abstract graph of the loop system is a tree on n vertices, addresses of length $(n-1)$ are best possible.

Theorem 4: If the abstract graph of the loop system is a cycle of length n, then addresses of length $n/2$ are best possible if n is even, and addresses of length $(n-1)$ are best possible if n is odd.

II. MATHEMATICAL DEVELOPMENT

Let us summarize what we have proven so far. Let (d_{ij}) be the distance matrix of the abstract graph G with vertices A_i. Let

$$\sum_{\alpha=1}^{N(G)} (A_{i_{\alpha,1}} \cdots A_{i_{\alpha,p_\alpha}}) \times (A_{j_{\alpha,1}} \cdots A_{j_{\alpha,r_\alpha}}) \tag{1}$$

represent the graph G in the sense that A_i and A_j appear on opposite sides of products exactly d_{ij} times. The number of coordinates which we must assign to each vertex of G is the minimum of $N(G)$ over all decompositions that satisfy the above conditions.

The problem is equivalent to a problem in quadratic forms. Write

$$\sum_{1 \le i \le j \le n} d_{ij} x_i x_j = \sum_{\alpha=1}^{N} (x_{i_{\alpha,1}} + \cdots + x_{i_{\alpha,p_\alpha}})(x_{j_{\alpha,1}} + \cdots + x_{j_{\alpha,r_\alpha}}). \tag{2}$$

Since $d_{ii} = 0$, no single x_k can appear in both factors of any single product. The equivalence is immediate since either decomposition will immediately yield the other. Our problem then is to find the minimum number N for any given quadratic form whose coefficients d_{ij} are the distance matrix of a graph. We shall prove the following lemma due to H. S. Witsenhausen.

Lemma 1: Let n_+, n_- be respectively the number of strictly positive and strictly negative eigenvalues of the distance matrix (d_{ij}). Then

$$N \geqq \max (n_+, n_-).$$

Proof: Let $Q(x_1, \cdots, x_n)$ denote the quadratic form $\sum_{1 \le i \le j \le n} d_{ij}x_ix_j$. As we have seen, the existence of a length k addressing of G is equivalent to the existence of a decomposition of Q into the sum of k products of the form $(x_{i_1} + \cdots + x_{i_r})(x_{j_1} + \cdots + x_{j_s})$. But we see that

$$\begin{aligned} Q &= \sum_{u=1}^{k} (x_{i_{u,1}} + \cdots + x_{i_{u,r(u)}})(x_{j_{u,1}} + \cdots + x_{j_{u,s(u)}}) \\ &= \frac{1}{4} \sum_{u=1}^{k} \{(x_{i_{u,1}} + \cdots + x_{i_{u,r(u)}} + x_{j_{u,1}} + \cdots + x_{j_{u,s(u)}})^2 \\ &\quad - (x_{i_{u,1}} + \cdots + x_{i_{u,r(u)}} - x_{j_{u,1}} - \cdots - x_{j_{u,s(u)}})^2\} \end{aligned}$$

so that we have represented Q as a sum of k squares minus another sum of k squares. However, it is an easy consequence of the theory of quadratic forms (cf. Ref. 2) that for any representation of Q as a sum of p squares minus a sum of q squares, we must have

$$p \geqq \text{index } Q = n_+ ,$$

$$q \geqq \text{rank } Q - \text{index } Q = n_- .$$

Therefore, $k \geqq \max(n_+, n_-)$ and the lemma is proved. For most simple examples equality seems to hold in the above lemma. However, most unfortunately, in general $N \neq \max(n_+, n_-)$. For the graph given in Fig. 7, we have $n_+ = 1$, $n_- = 5$, but a computer search of possible decompositions has shown $N = 6$.

Lemma 1 is strong enough to settle the best N in many cases. In preparation let us prove Lemma 2, due to E. N. Gilbert.

Lemma 2: If the $n \times n$ distance matrix (d_{ij}) is cyclic (meaning $d_{ij} = a(j - i) \bmod n$), then the eigenvalues of (d_{ij}) are the values of

$$P(z) = \sum_{0}^{n-1} a_i z^i$$

at each nth root of unity.

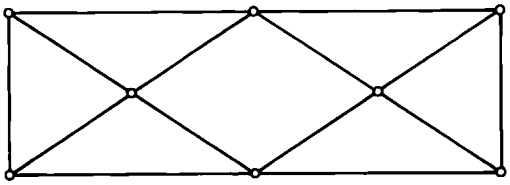


Fig. 7—Graph for Lemma 1.

Proof of Lemma 2: Let the cyclic matrix M be

$$M = \begin{pmatrix} a_0 & a_1 & \cdots & a_{n-1} \\ a_{n-1} & a_0 & \cdots & a_{n-2} \\ & & \vdots & \\ a_1 & a_2 & & a_0 \end{pmatrix}.$$

If we try an eigenvector of the form

$$\begin{pmatrix} 1 \\ z \\ \vdots \\ z^{n-1} \end{pmatrix},$$

then

$$M\begin{pmatrix} 1 \\ z \\ \vdots \\ z^{n-1} \end{pmatrix} = \begin{pmatrix} a_0 + a_1 z + \cdots + a_{n-1}z^{n-1} \\ a_{n-1} + a_0 z + \cdots + a_{n-2}z^{n-1} \\ \vdots \\ a_1 + a_2 z + \cdots + a_0 z^{n-1} \end{pmatrix}.$$

If $z^n = 1$, then the latter matrix equals

$$(a_0 + a_1 z + \cdots + a_{n-1}z^{n-1})\begin{pmatrix} 1 \\ z \\ \vdots \\ z^{n-1} \end{pmatrix}.$$

Thus the values of $a_0 + a_1 z + \cdots + a_{n-1}z^{n-1}$ if $z^n = 1$ are eigenvalues of the matrix M. Since they are n in number, and since M has only n eigenvalues, they are *all* the eigenvalues of M.

Theorems 2, 3, and 4 may now be proved by using these lemmas. Let us prove a statement equivalent to:

Theorem 2: If G is the complete graph on n vertices, then $N(G) = n - 1$.*

Proof: For this graph, $d_{ij} = 1$, $1 \leqq i < j \leqq n$. The corresponding quadratic form is

$$\sum_{1 \leqq i < j \leqq n}^{n-1} x_i x_j ,$$

* i.e., any two vertices of G are joined by an edge.

which is equal to

$$\sum_{i=1}^{n-1} x_i(x_{i+1} + \cdots + x_n).$$

Hence $N(G) \leqq n - 1$. To obtain an inequality in the opposite direction, we examine the eigenvalues of the (d_{ij}) matrix. By Lemma 2, they are the values of

$$P(z) = z + z^2 + \cdots + z^{n-1}$$

when $z^n = 1$. But $P(z) = [z(z^{n-1} - 1)/(z - 1)]$, so that if $z^n = 1$ and $z \neq 1$, $P(z) = -1$. If $z = 1$, $P(z) = (n - 1)$. Hence $n_+ = 1$, $n_- = (n - 1)$, and, by Lemma 1, $N(G) \geqq n - 1$. Hence $N(G) = n - 1$.

Theorem 3: If the graph G is a tree with n vertices, then $N(G) = n - 1$.*

Proof: We first examine the distance matrix D_n for a tree with n vertices. Consider a terminal vertex v_i, i.e., a vertex which is distance 1 from just *one* other vertex, say v_j. By a suitable relabeling we can assume $i = n$ and $j = n - 1$.† Thus, $d_{n,k} = 1 + d_{n-1,k}$ for $1 \leqq k \leqq n - 1$. Hence, the matrix D_n has the form

$$D_n = \begin{bmatrix} 0 & d_{12} & \cdots & d_{1n-1} & 1 + d_{1n-1} \\ d_{12} & 0 & \cdots & d_{2n-1} & 1 + d_{2n-1} \\ \vdots & \vdots & & \vdots & \vdots \\ d_{1n-1} & & \cdots & 0 & 1 \\ 1 + d_{1n-1} & & \cdots & 1 & 0 \end{bmatrix}.$$

We wish to evaluate the determinant $\det(D_n)$ of D_n. Certainly we can subtract column $n - 1$ from column n and row $n - 1$ from row n of D_n without changing $\det(D_n)$. This leaves us with a matrix D_n' with the form

$$D_n' = \begin{bmatrix} 0 & \cdots & d_{1n-1} & 1 \\ & & & 1 \\ \vdots & & \vdots & \vdots \\ d_{1n-1} & \cdots & \cdots & 1 \\ 1 & \cdots & 1 & -2 \end{bmatrix}.$$

* i.e., G is connected and has no cycles.

† We have chosen $j = n - 1$ to simplify the exposition of the first part of the proof. In fact, any j, $1 \leqq j \leqq n - 1$, is acceptable. This generality is required later in the proof.

But, we now imagine removing the vertex v_n from G, forming a tree G_{n-1} with $n - 1$ vertices. The interpoint distances in G_n are given exactly by the upper-left $(n - 1)$-by-$(n - 1)$ submatrix of D_n'. As before, we can suitably relabel the vertices of G_{n-1} so that v_{n-1} is a terminal vertex adjacent only to v_{n-2}. The corresponding rearranged matrix D_n'' now has the form

$$D_n'' = \begin{bmatrix} 0 & d_{12} & \cdots & d_{1n-2} & 1 + d_{1n-2} & 1 \\ d_{12} & 0 & \cdots & d_{2n-2} & 1 + d_{2n-2} & 1 \\ \vdots & & & \vdots & \vdots & \vdots \\ d_{1n-2} & & \cdots & 0 & \cdot & 1 \\ 1 + d_{1n-2} & & \cdots & 1 & 0 & 1 \\ 1 & & \cdots & 1 & 1 & -2 \end{bmatrix}.$$

By subtracting column $n - 2$ from column $n - 1$ and row $n - 2$ from row $n - 1$ we obtain

$$D_n''' = \begin{bmatrix} 0 & d_{12} & \cdots & d_{1n-2} & 1 & 1 \\ d_{12} & 0 & \cdots & d_{2n-2} & 1 & 1 \\ \vdots & \vdots & & \vdots & \vdots & \vdots \\ d_{1n-2} & d_{2n-2} & \cdots & 0 & 1 & 1 \\ 1 & 1 & \cdots & 1 & -2 & 0 \\ 1 & 1 & \cdots & 1 & 0 & -2 \end{bmatrix}.$$

It is not difficult to see that this process can be continued until we reach the matrix

$$D_n^* = \begin{bmatrix} 0 & 1 & 1 & 1 & \cdots & 1 & 1 \\ 1 & -2 & 0 & 0 & \cdots & 0 & 0 \\ 1 & 0 & -2 & 0 & \cdots & 0 & 0 \\ 1 & 0 & 0 & -2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots \\ 1 & 0 & 0 & 0 & \cdots & -2 & 0 \\ 1 & 0 & 0 & 0 & \cdots & 0 & -2 \end{bmatrix}.$$

The first (surprising) conclusion we draw is that det (D_n) depends

only on the number of vertices n and not on the *structure* of the tree G. By expanding det (D_n^*) along the last column it is easy to get the recurrence

$$D_n = (-1)^{n-1}2^{n-2} - 2D_{n-1}, \quad D_1 = 0, \quad D_2 = -1$$

from which it follows that

$$D_n = (-1)^{n-1}(n-1)2^{n-2}, \quad n \geqq 1.$$

We next note that if we relabel the vertices of G, according to the relabeling used to get the matrix D_n^*, in the corresponding distance matrix $\overline{D_n}$ (which is a permutation of the original distance matrix D_n) the upper left-hand k-by-k submatrix $\overline{D_k}$ of $\overline{D_n}$ is just the distance matrix for some k vertex subtree of G. Hence,

$$\det(\overline{D_k}) = (-1)^{k-1}(k-1)2^{k-2}, \quad k \geqq 1.$$

Finally, the sequence of determinants

$$1, \det(\overline{D_1}), \det(\overline{D_2}), \cdots, \det(\overline{D_n}) \tag{3}$$

is just

$$1, 0, -1, 4, -12, 32, \cdots, (-1)^{n-1}(n-1)2^{n-2}.$$

Hence, the number of *permanences of sign* of this sequence (where 0 is fixed as either positive or negative) is just *one*! By a theorem in matrix theory (cf. Ref. 2), the number of permanences in sign of the sequence (3) is exactly the number of positive eigenvalues of $\overline{D_n}$ which we have seen is just one. Since $\overline{D_n}$ is nonsingular for $n \geqq 1$, then $\overline{D_n}$ has no *zero* eigenvalues and hence, $\overline{D_n}$ must have $n - 1$ negative eigenvalues. Therefore D_n also has $n - 1$ negative eigenvalues and by Lemma 1, $N(G) \geqq n - 1$.

The construction which gives $N(G) \leqq n - 1$ has an easy recursive definition: Each time we choose the next vertex v_i in the tree to assign an address to,* make sure that it is adjacent to a vertex v_j which is already addressed, and let $A(v_i) \rightarrow A(v_j)$ 1 and $A(v_k) \rightarrow A(v_k)$ 0 for the previously addressed vertices (i.e., 1 and 0 are adjoined to the previous addresses). Thus, after all vertices have been addressed, all addresses will have length $n - 1$ and, in fact, no d's are used. Therefore, $N(G) = n - 1$ and the theorem is proved.

Theorem 4: If G is a cycle on n vertices, then $N(G) = n/2$ if n is even and $(n - 1)$ if n is odd.

* Where we assign 0 to the first vertex and 1 to the second vertex.

Proof: If $n = 2m$, then the vertices $A_1, \cdots A_{2m}$ can be coordinatized as follows:

$$A_s = (\underbrace{1 \cdots 1}_{s-1}, \underbrace{0 \cdots 0}_{m-s+1}) \quad \text{if} \quad 1 \leqq s \leqq m+1,$$

and

$$A_s = (\underbrace{0 \cdots 0}_{s-m-1}, \underbrace{1 \cdots 1}_{2m-s+1}) \quad \text{if} \quad m+2 \leqq s \leqq 2m.$$

Clearly $d_{ij} = \min(|i-j|, 2m - |i-j|)$ is the number of places in which A_i and A_j differ, and is the correct distance on a cycle. Hence $N(G) \leqq m$. On the other hand, $d_{1,m+1} = m$, and hence A_1 and A_{m+1} must differ in exactly m coordinates. Therefore there must *be* at least m coordinates, and hence $N(G) \geqq m$. Thus $N(G) = m$.

If $n = 2m + 1$, consider the following addresses:

$$\begin{array}{ll} & \overbrace{\qquad\qquad}^{2m} \\ & \overbrace{\quad}^{m} \quad \overbrace{\quad}^{m} \\ A_0 & -000\cdots00\cdots000 \\ A_1 & -d00\cdots00\cdots001 \\ A_2 & -dd0\cdots00\cdots011 \\ & \vdots \\ A_m & -ddd\cdots d1\cdots111 \\ A_{m+1} & -1dd\cdots d1\cdots110 \\ A_{m+2} & -d1d\cdots d1\cdots100 \\ A_{m+3} & -dd1\cdots d1\cdots000 \\ \vdots & \\ A_{2m} & -ddd\cdots10\cdots000 \end{array}$$

We see that:

(*i*) if $0 \leqq i \leqq j \leqq m$, $d_{ij} = j - i$;
(*ii*) if $m < i \leqq j \leqq 2m$, $d_{ij} = j - i$;
(*iii*) if $0 \leqq i \leqq m$ and $j = m + s$ where $s > 0$, then consider separately $i > s$ and $i \leqq s$. If $i > s$ then the first m coordinates contribute 0 and the second m contribute $j - i$. If $i \leqq s$ the

first m coordinates contribute 1 and the second m contribute $i + 2m - j$, so that together they give $2m + 1 - j + i$ which is the correct cyclic distance.

We thus know that $N(G) \leqq 2m$.

To prove $N(G) \geqq 2m$, we use Lemmas 1 and 2.

$$P(z) = z + 2z^2 + \cdots + mz^m + mz^{m+1} + (m-1)z^{m+2} \cdots + z^{2m},$$

and we consider z such that $z^{2m+1} = 1$.

If $z_k = \exp(2\pi ik/2m + 1)$, then

$$P(z_k) = 2 \sum_{j=1}^{m} j \cos \frac{2j\pi k}{2m+1}, \qquad k = 0, 1, 2, \cdots, 2m.$$

$P(1) > 0$; we shall prove $P(z_k) < 0$ for all other k. We find that if we define

$$g(x) = \sum_{j=1}^{m} \sin \frac{2j\pi x}{2m+1},$$

then

$$g(x) = \frac{1}{2}\left[\frac{\cos \frac{\pi x}{2m+1} - \cos \pi x}{\sin \frac{\pi x}{2m+1}}\right].$$

Therefore

$$g'(x) = \frac{\pi}{2} \frac{-\frac{1}{2m+1} + \sin \pi x \sin \frac{\pi x}{2m+1} + \frac{1}{2m+1} \cos \pi x \cos \frac{\pi x}{2m+1}}{\sin^2 \frac{\pi x}{2m+1}}$$

is $\pi/(2m + 1)$ times the desired series if x is $1, 2, \cdots 2m$. But $g'(x) < 0$ at all of these points. Hence $n_+ = 1$, $n_- = 2m$, and $N(G) \geqq 2m$ by Lemma 1. The theorem is proved.

III. ADDRESSES OF MINIMUM LENGTH

We describe an algorithm which is guaranteed to produce a valid addressing for any graph G. This algorithm has always succeeded in finding an addressing of length $\leqq n - 1$ for every graph G on n vertices to which it has been applied. However, no proof that this will always happen is currently known.

The algorithm proceeds as follows:

(*i*) Number the n vertices of G with integers $\{1, 2, \cdots, n\}$ *so that for* $k > 1$, *the vertex numbered* k *is adjacent to some vertex with a smaller number*. Since G is connected, this is always possible. Let $v(k)$ denote the vertex to which k has been assigned.

(*ii*) Assign the (partial) addresses of 0 to $v(1)$ and 1 to $v(2)$.

(*iii*) In general, suppose we have assigned (partial) addresses to $v(1), v(2), \cdots, v(k)$, say, $A(i)$ has been assigned to $v(i)$, so that $d_{i,j} = d_H(A(i), A(j))$, $1 \leqq i < j \leqq k$, where d_H denotes the Hamming distance and $d_{i,j}$ denotes the distance between $v(i)$ and $v(j)$ in G. We next search for an address $A(k+1)$ (of the same length as the $A(i)$) with the property that $\max_{1 \leqq i \leqq k} (d_{i,k+1} - d_H(A(i), A(k+1))) \equiv m_{k+1}$ is as small as possible under the constraint

$$\min_{1 \leqq i \leqq k} (d_{i,k+1} - d_H(A(i), A(k+1))) \geqq 0. \qquad (*)$$

Of course, we can always find *some* address which satisfies (*), namely the all d's address. Typically we can choose $A(k+1)$ so that $m_{k+1} = 1$. In fact, it is usually possible to do this by choosing $A(k+1)$ to be a slightly perturbed copy of some $A(l)$ where $v(l)$ is adjacent to $v(k+1)$. This is intuitively reasonable since in this case $|d_{i,k+1} - d_{i,l}| \leqq 1$.

After $A(k+1)$ has been chosen, we then *adjoin* m_{k+1} symbols to each of the partial addresses $A(i)$, $1 \leqq i \leqq k+1$, as follows. To $A(k+1)$ we adjoin m_{k+1} 1's. To $A(i)$ we adjoin $m_{k+1} - (d_{i,k+1} - d_H(A(i), A(k+1)))$ d's and $d_{i,k+1} - d_H(A(i), A(k+1))$ 0's. It is easy to check that for the new augmented addresses $A'(i)$, $1 \leqq i \leqq k+1$, we have

$$d_{i,j} = d_H(A'(i), A'(j)), \quad 1 \leqq i < j \leqq k+1.$$

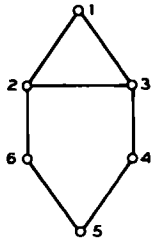
We continue in this manner until the addressing is completed. By construction, the terminal addresses will form a valid addressing for G of length $1 + m_3 + \cdots + m_n$.

As an example, we construct an addressing for the graph in Section I by this process. In Fig. 8 we show this graph with a particular "adjacent-numbering" chosen and also the distance matrix for the graph.
We start with

vertex address

1—0
2—1

Adjoining vertex 3, we see that any partial address of length one will give $m_3 = 1$. We choose 0.



	1	2	3	4	5	6
1	0	1	1	2	3	2
2	1	0	1	2	2	1
3	1	1	0	1	2	2
4	2	2	1	0	1	2
5	3	2	2	1	0	1
6	2	1	2	2	1	0

$D = (d_{ij})$

Fig. 8—Addressing example.

vertex address

1—0
2—1
3—0

We next adjoin $m_3 = 1$ 1's to $A(3)$ and augment $A(1)$ and $A(2)$ accordingly.

vertex address

1—00
2—1*d*
3—01

Now adjoin vertex 4, choose partial address 01, calculate that $m_4 = 1$, and augment the partial addresses accordingly.

vertex address

1—000
2—1*d*0
3—010
4—011

Continue this for two more steps. Each time $m_k = 1$.

vertex address	*vertex address*
1—0000	1—0000*d*
2—1*d*0*d*	2—1*d*0*dd*
3—0100	3—0100*d*
4—0110	4—01100
5—0111	5—0111*d*
	6—1*d*1*d*1

The last array gives a length 5 addressing for G. Of course, different partial addresses or a different initial vertex numbering will result in different addressings for G.

As we have previously stated, we have no general proof that $N(G) \leqq n - 1$ in all cases although a number of partial results in that direction have been given as well as a heuristic construction.

IV. ADDRESSING IN RESTRICTED LOOP SYSTEMS

The addressing scheme we have been describing has the very great power of being able to handle an arbitrary configuration of loops, and to provide alternate routing in an optimal way without any supervisory memory. The price we have paid for this generality is in the length of the address—typically $n - 1$ "bits" for n loops in the simplest encoding—and in possible complications under system growth. It is clear that if a new loop is added which greatly shortens the distance between many pairs of loops, then many addresses may change a good deal. There would be various ways of handling this, but it is obviously a problem. It arises essentially because the numbering in its full generality is not hierarchical.

Typical Bell System loop configurations, as we noted in the introduction, will not be arbitrary collections of loops, but will have a hierarchical structure.

By correspondingly restricting the allowable adjacency graphs G, it is possible to modify the routing algorithm and effectively take advantage of a natural "product" construction, as pointed out by J. R. Pierce.[1] In this system, as we saw, loops are partitioned into three classes—national, regional, and local. The address portion of the message is subdivided into three corresponding portions. The routing algorithm now consists of three steps: (*i*) First apply the previous Hamming distance algorithm to the "national" portions of the sending and the destination addresses; (*ii*) When the distance in i becomes zero, then apply the Hamming distance algorithm to the "regional" portions of the addresses; (*iii*) Finally, when the distance in ii is zero, apply the Hamming distance algorithm to the "local" portions of the address.

This scheme combines the efficiency of the Hamming distance algorithm with the savings in address lengths resulting from the hierarchical structure. As an example, the network in Fig. 9 has 44 local vertices. For a direct Hamming algorithm addressing we should expect addresses to have length of around 59. By distinguishing national, regional, and local loops (capital letters, lower case letters, and integers

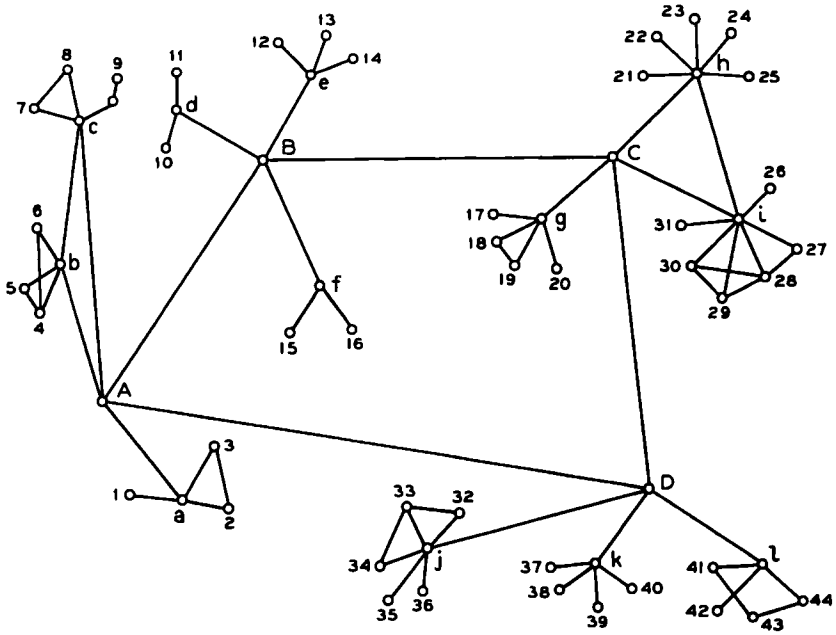


Fig. 9—Network example.

respectively), with a small additional computed cost in routing (several extra conditional transfers) we can have addresses of length $\leqq 11$. For example, let N, R, L denote national, regional, local, respectively. One possible addressing begins:

	N	R	L
A—	(00,	000	--)
B—	(01,	000,	---)
C—	(10,	000,	---)
D—	(11,	000,	---)
a—	(00,	001,	000)
b—	(00,	010,	000)
c—	(00,	1d0,	000)
d—	(01,	001,	000)
	⋮		⋮

1—(01, 001, 001)

2—(01, 001, 010)

3—(01, 001, 1*d*0)

⋮ ⋮

44—(11, 000, 001)

Moreover, to add additional local stations to a regional station it is a very simple matter to modify just the *neighboring* local addresses to obtain a correct addressing for the augmented network.

The restriction on local loops in the above addressing is that each one must interchange directly with one and only one regional loop. If a local loop meets no regional loop directly, but only other local loops, then the addressing must make special provision for routing calls to other regions properly. If a local loop meets more than one regional loop—really a violation of the hierarchical concept—then routing becomes more difficult, and must assure that a call to a different region exits the local loop properly. As J. R. Pierce has pointed out,[1] a special trunk loop connecting a local loop in one region to a local loop in another (i.e., a preferred alternate route in a special case to the national loop) is no problem. The exit from the local loop is just before the regional interchange, and the entrance to the local loop just after. Exit is made only if the total loop address matches exactly. Alternate routes more generally are perhaps most easily provided by duplicating portions of regional or natural loops.

V. SOME VARIANTS OF THE ADDRESSING PROBLEM

The purpose of this section is to record very briefly some other alternatives that have been considered.

(*i*) We have required that in every alternate route between the loops, Hamming distance decrease by exactly 1 at each transfer. One could consider the alternate problem in which *any* exit which decreases Hamming distance is valid—even if it decreases it by *more* than 1. Under special conditions, this can lead to shorter addresses, but we do not have a solution for this alternate problem.

(*ii*) Since the introduction of *d*'s causes some complication of the address codes, it is interesting to consider the possibility of getting rid of them. They arose originally because of the need for *odd* cycles, as in the case of a 3-cycle. One way out of this example would be to *double* all the

distances. If these vertices were located at 000, 110, and 101 respectively, the Hamming distance between any pair is two, and correct routing would be possible without any d's in the addresses.

Unfortunately, this technique of doubling all the distances to get rid of d's does not generalize. Consider the graph in Fig. 10a. We double all

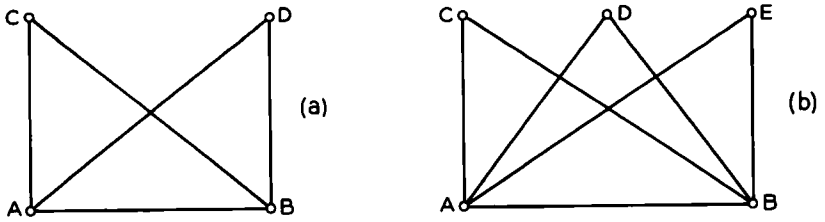


Fig. 10—Graphs to illustrate one variant of the addressing problem.

distances, so that $AB = AC = AD = BC = BD = 2$, $CD = 4$. Then $A = 00 \cdots$, $B = 11 \cdots$, where the coordinates are identical from the third onward. Now C must differ from each of A and B by 2. It therefore must differ in one of the first two columns, and in one other, say the third. Thus, we may assume $A = 000 \cdots$, $B = 110 \cdots$, $C = 101 \cdots$, where the coordinates are identical from the fourth onward. D must also differ in exactly two places from A and B and in four places from C. Hence $A = 0000 \cdots$, $B = 1100 \cdots$, $C = 1010 \cdots$, $D = 0101 \cdots$.

So far so good. If we now require yet another point E (Fig. 10b) such that $EA = EB = 2$, $EC = ED = 4$, we have no possible coordinates for E left. E's address must begin with 01 or 10 in order to differ from A and B by equal amounts, say with 10. To differ from C and D by equal amounts the first four coordinates must be 1001. But it now differs from A and B by 2 and from C and D by 2; no additional coordinates can make $EC = ED = 4$ without destroying $EA = EB = 2$. Thus doubling distances will not get rid of d's.

Similar arguments show that even if we are allowed to multiply all distances by a fixed number $m > 2$, we still cannot get along without d's.

REFERENCES

1. Pierce, J. R., "Network for Block Switching of Data," unpublished work.
2. Jones, B. W., "The Arithmetic Theory of Quadratic Forms," CARUS Mathematical Monograph No. 10, published by MAA, 1950.