RC 8017 (#34783) 12/17/79
Communications 20 pages

Research Report

ON THE GENERATION OF EXPLICIT ROUTING TABLES

Kiyoshi Maruyama
George Markowsky

Computer Sciences Department
IBM Thomas J. Watson Research Center
P. O. Box 218
Yorktown Heights, N.Y. 10598

Proceedings of the Fifth International Conference on Computer Communication

ICC Atlanta 27-30, October 1980
pp. 90-95

RC 8017 (#34783) 12/17/79
Communications 20 pages

ON THE GENERATION OF EXPLICIT ROUTING TABLES

Kiyoshi Maruyama
George Markowsky

Computer Sciences Department
IBM Thomas J. Watson Research Center
P. O. Box 218
Yorktown Heights, N.Y. 10598

Abstract:

Explicit routing offers some good features for packet-switching. This paper considers three table-driven explicit routing algorithms depending on the information carried in the packet header and how the packet is steered at the intermediate nodes. These are the ON/DN/ERN algorithm, the DN/ERN algorithm and the DN/ERN/IX algorithm. This paper assumes that the routes to be defined in the routing tables are given and for each of the above routing algorithms, describes algorithms for generating the corresponding routing tables. In discussing the DN/ERN algorithm, we show that the ERN assignment problem is NP-complete and discuss a number of heuristic approaches to its solution. All the algorithms described have been coded in PL/1 and run successfully.

1. INTRODUCTION

The problem of routing in store-and-forward packet switched communication networks has been one of the most intensively studied areas in the field of computer communications in the past decade. A large variety of routing procedures have been developed and some are actually implemented in existing networks. We classify these routing procedures, for the purpose of this paper, as deterministic routing and stochastic routing.

In deterministic routing, a complete route is selected at the packet origin node, and each intermediate node ensures that the packet follows the selected route unless some elements along the route fail. If the routing procedure is table-driven, then the table at each intermediate node is not adaptive to changes in traffic, and are updated (either centrally or locally) only when some changes in network topology occur. The initial route selection at the packet origin node, however, can be adaptive to the traffic changes, and the route selection may be performed for each packet, message, session or virtual route. Examples of these routing procedures are the minimum-hop routing, TYMNET routing [RA78] and explicit routing [JU76].

In stochastic routing, a complete route is not selected at the packet origin node nor at any intermediate node, and the routing decision of the packet is entirely up to an intermediate node whose routing table is, in general, updated when changes in traffic or topology occur. An example of such routing is the ARPANET routing [RO72].

Both routing procedures have advantages and disadvantages. Some advantages of deterministic routing are: FIFO packet delivery so long as link level packet delivery is FIFO; easy failed element diagnosis; easy determination of the loss of connectivity; less administrative traffic; the ability to use parallel links for load balancing. Some advantages of stochastic routing are: adaptivity to changes in traffic and topology; easy to implement in a decentralized fashion.

In this paper we are mostly interested in deterministic routing procedures known as explicit routing [JU76]. In explicit routing, a set of routes, say up to k routes, is pre-selected for each pair of communication nodes, and only these routes can be used for communication unless the set is re-selected. The routing decision in explicit routing is therefore the selection of a route from the set of routes at the packet origin node. Such a route selection at the origin node can be adaptive to changes in traffic and the selection can be made for every packet or for every session or for every virtual connection between the origin and destination nodes.

An *explicit route* is an ordered sequence of nodes and transmission groups from an origin node to a destination node. A *transmission group* (TG) is a set of one or more physical links grouped together to form a connection between a pair of nodes and usually identified by a transmission group number (TGN). The transmission group numbers assigned to transmission groups between the same node pair must be different. Fig. 1.1. illustrates a simple example of a six-node network. This network consists of nine transmission groups each of which is labelled with a transmission group number (TGN). Each of these TGs may consists of one or more physical links. Between node A and node E, for example, there exist following six explicit routes:

A(1),B(1),E
A(1),B(1),C(1),E
A(1),B(1),C(1),D(1),E
A(1),D(1),E
A(1),D(1),C(1),E
A(1),D(1),C(1),B(1),E

Dependent on a particular implementation, some or all of the above explicit routes may be defined in the routing table.

The above concept of the explicit routing has been adopted in IBM's Systems Network Architecture (SNA) to support multiple active routes between SNA sub-area nodes such as communication controllers and hosts [GR79,AH79]. Multiple explicit routes between a pair of nodes provide alternate routing capability for additional reliability, availability and possible traffic load splitting among routes.

Explicit routing algorithms may be classified by the way packets are steered into header-driven algorithms or table-driven algorithms. In header-driven algorithms, the entire route information from the origin to the destination nodes is stored in the packet header. Thus, an intermediate node steers the packet by referencing the route information stored in the packet header. In table-driven algorithms, only partial route information is stored in the header. Thus, an intermediate node steers the packet by referencing both the partial route information in the packet header and the routing table stored at that node.

Although, header-driven algorithms provide conceptual simplicity, thay have some serious drawbacks; the packet header can be very large for a large network, and each origin node must store complete route information to all destination nodes unless otherwise generated when it becomes necessary. The size of the routing table is proportional to k(n-1)m, where k is the number of explicit routes between a origin node to a destination node, n is the number of nodes in the network, and m is the average length of an explicit route measured in the number of transmission groups in a route.

In this paper, we are interested in table-driven explicit routing algorithms. We consider three different implementations for table-driven explicit routing algorithms depending on what information is carried in the packet header and how the packet is steered at the intermediate nodes (or how an explicit route is defined). These will be identified as the ON/DN/ERN algorithm, the DN/ERN algorithm and the DN/ERN/IX algorithm. Here ON and DN denote origin and destination nodes, and ERN and IX denote explicit route number and route index, respectively. In the following, we assume that the explicit routes which should be used for the network operation are pre-determined for each pair of nodes using some route selection criteria and selection algorithms such as those discussed in [NA79], and discuss routing table generation algorithms for each of the above three different explicit routing algorithms.

2. THE ON/DN/ERN EXPLICIT ROUTING APPROACH

In this approach, an explicit route is identified by the triplet (ON,DN,ERN). Each packet carries in its header, among others, these three values. The routing table at each node consists of ON, DN and ERN fields and an additional field which indicates the next intermediate node to be visited in order to reach the given destination node. This is depicted in Fig. 2.1. The value given by (ON,DN,ERN) in the packet header is used at each intermediate node to locate the corresponding entry in the routing table at that node to find the next node (NN) to be visited and the transmission group (TGN) to be used in order for the packet to reach the given destination node DN. In an actual routing table implementation, the NN.TGN field in the figure may indicate the pointer to the corresponding transmission group send queue.

The algorithm for generating routing tables for ON/DN/ERN explicit routing is as follows:

ALGORITHM 2.1

For each pair of nodes in the network, repeat the following two steps:

(1) Assign a different explicit route number (ERN) to each route between the selected pair of nodes. One may assign a smaller number to a preferred route. However, ERN assignments can be arbitrary as long as assigned ERNs to routes between pairs of nodes are all different.

(2) Repeat the following for each explicit route: let ERN=r, ($1 \leq r \leq k$, where k denotes the number of routes predetermined to be supported by the routing tables) be assigned to an explicit route:

$$N_1\ (g_1), N_2\ (g_2), \ldots\ldots, N_i\ (g_i), \ldots\ldots, N_m$$

where N_1 is the origin node, N_2 through N_{m-1} are intermediate nodes, N_m is the destination node and g_i denotes the transmission group number (TGN) which is traversed from N_i to N_{i+1} for $i=1,2,\ldots, m-1$. For each j, $j=1,2,\ldots,m-1$, create a routing entry in the routing table at the node N_j which consists of

$$ON=N_1,\ DN=N_m,\ ERN=r,\ NN=N_{j+1},\ \text{and}\ TGN=g_j.$$

Fig. 2.2 illustrates a portion of the routing table at node C which was generated by ALGORITHM 1 for the explicit routes shown in Table 1. Table 1 shows two pre-determined routes from all other nodes to the destination node E of the network illustrated in Fig. 1.1. In the first step of ALGORITHM 1, we assigned ERN=1 to the short route and ERN=2 to the longer route between each origin-destination node pair.

The advantages of this algorithm are; it is very easy to generate routing tables because it is easy to assign ERNs to routes; it allows arbitrary selection of explicit routes between each pair of nodes. The major drawback, compared to the other routing algorithms we will describe, is that the routing table is large, since its size is proportional to the number of explicit routes going through as well as starting from that node. In the worst case, it is proportional to k(n-1)(n-2), where k is the number of explicit routes between a pair of nodes and n is the number of nodes in the network.

3. THE DN/EKN EXPLICIT ROUTING APPROACH

In this approach, an explicit route is identified by the pair (DN,ERN). Each packet header contains the same information fields as the ON/DN/ERN approach, however, the routing table consists of only three fields: DN, ERN and NN.TGN. This is depicted in Fig. 3.1. The value given by (DN,ERN) in the packet header is used at each intermediate node to identify the next node to be visted and the transmission group to be used in order for the packet to reach the given destination node. Clearly, this approach is an origin independent routing scheme, while the ON/DN/ERN approach is an origin dependent routing scheme. An origin independent routing does not necessarily imply any particular method of route selection. This approach allows the sharing of routes from an intermediate node to a destination node (one may consider this as sharing of routing entries) if ERN numbers are assigned to routes appropriately.

Since the packet steering is performed using only the (DN,ERN) information stored in the packet header regardless of its origin node, special care must be paid when assigning ERN numbers to the routes. It requires that when an ERN is assigned to a route, all route segments that comprise the route to the destination node must be assigned the same ERN. Let an explicit route be denoted by:

$$N_1\ (g_1),N_2\ (g_2),\dots,N_i\ (g_i),\dots,N_{m-1}\ (g_{m-1}),N_m$$

and let r be the assigned ERN. Then the node N_i must support the route segment:

$$N_i\ (g_i),\dots,N_{m-1}\ (g_{m-1}),N_m$$

with ERN=r, for i=1,2,...,m-1. Because of this requirement, intermediate nodes often have to assign multiple ERNs to the same physical route; the route segment $N_i\ (g_i),\dots,N_m$ can be a route segment of another explicit route which carries a different ERN number.

A set of routes to a particular destination node is said to be a *prime set* of routes if no route in the set is contained in another route in that set. A route in a prime set of routes is called a *prime route*. For example, a set of routes:

A(1),B(2),E
B(2),E

is not a prime set since the first route contains the second route. A set of routes:

A(1),B(2),E
A(1),B(1),E

is a prime set (remember that an explicit route is an ordered sequence of node names and TGNs). The prime set for the explicit routes shown in Table 1 consists of the following prime routes:

A(1),B(1),E
C(1),B(1),E
B(1),C(1),E
D(1),C(1),E
A(1),D(1),E
F(1),E
F(2),E

A trivial way to assign ERNs to routes is to assign a different ERN to each prime route. However this numbering has a drawback that the total number of ERNs required can be as large as k(n-1). An important problem associated with the DN/ERN approach is to number routes using the least number of ERNs. Unfortunately, as the following theorem shows, this problem is NP-complete since it is polynomially equivalent to graph coloring which is known to be NP-complete (see [AH74], Chapter 10). As is well-known, there are no "good" algorithms for NP-complete problems, although a number of "good" approximate algorithms are known in many cases.

Theorem 3.1

The ERN assignment problem for the DN/ERN explicit routing algorithm is an NP-complete problem, being equivalent to graph coloring.

Proof:

I. Any route numbering problem (i.e., ERN assignment to explicit routes) can be transformed into a graph coloring problem, so that a solution to the graph coloring problem yields a solution to the route numbering problem. We first describe a transformation of the ERN assignment problem into a graph coloring problem.

From a prime set of routes, we construct a graph, G, which we call a *contention graph*, in the following manner. A node of G represents a prime route. An edge is inserted between a pair of nodes if the two routes represented by the pair of nodes can not assume the same ERN number, i.e., if some node occurs in both routes but has a different successor in the two cases. Clearly, deriving the contention graph is a polynomial time procedure. Thus the ERN assignment problem is the class NP.

II. Next, we want to show that given any graph G with N nodes, we can construct a network $\mathcal{N}$ having N^2-N+1 nodes and specifies N routes in $\mathcal{N}$ such that the contention graph of these routes is isomorphic to G. The procedure for constructing $\mathcal{N}$ and the N routes is simple and can be done in polynomial time. Thus the route numbering problem will be shown to be NP-complete.

Let G have the nodes $n_1,\ldots,n_N$. Let $\mathcal{N}$ have the nodes n_{ij}, $i,j=1,\ldots,N$, $(i\neq j)$ and a distinguished node w. We can connect any two nodes in $\mathcal{N}$ together or we can just insert enough edges into $\mathcal{N}$ in order to support the paths $P_1,\ldots,P_N$ which we will now define. For $1\leq i\leq N$, let

$$s(i)=\{j \mid 1\leq j\leq N \text{ and there is an edge between } n_i \text{ and } n_j\}, \text{ and}$$

$$P_i=\underbrace{n_{ji}\ldots\ldots}_{j\in s(i)}n_{i1}\ldots\ldots n_{iN}w.$$

If $j\in s(i)$, then P_i and P_j have both n_{ij} and n_{ji} in common. Since they are in a different order, P_i and P_j have an edge in their contention graph. If $j\notin s(i)$, then $i\notin s(j)$ and we see that P_i and P_j have only w in common. Thus, in the contention graph there is no edge between P_i and P_j. Hence, in G, there is an edge between n_i and n_j iff P_i and P_j cannot be colored the same color.

This shows that if we could solve the route numbering problem in polynomial time we could solve the graph coloring problems in polynomial time. Since graph coloring is known to be NP-complete, route numbering is thus seen to be NP-complete by I. □

Once ERNs are assigned to explicit routes, the following algorithm can be used for the construction of routing tables:

Let an ERN=r be assigned to an explicit route:

$$N_1(g_1),\ldots,N_i(g_i),N_{i+1}(g_{i+1}),\ldots,N_{m-1})(g_{m-1}),N_m \ .$$

Then at node N_i, create the routing entry such that

$$DN=N_m,\ ERN=r,\ NN=N_{i+1} \text{ and } TGN=g_i,$$

for i=1,2,..,m-1.

There are several heuristic algorithms for the ERN assignment problem which might be worthwhile noting. They differ somewhat in their strategies and depending on the presence of additional constraints (if there are any) they might have significantly different performance.

ALGORITHM 3.1 (Based on Graph Coloring)

Construct the contention graph as in Theorem 3.1 and then color the graph using your favorite heuristic for this problem.

Fig. 3.2 shows the contention graph of the set of routes given in Table 1. Also shown in the figure is a possible graph coloring by ERN numbers.

ALGORITHM 3.2 (Based on routing tree decomposition)

First construct a routing tree T(DN) from the prime set of routes to destination node DN. Then decompose T(DN) into the minimum number of routing subtrees such that each subtree contains the root node DN and other nodes each of which appears at most once in the subtree. Finally, assign different ERN number to each subtree. All routes contained in a subtree assume the ERN number assigned to that subtree.

Fig. 3.3 illustrates the routing tree T(E) for the explicit routes given in Table 1. Fig. 3.4 illustrates a decomposition of T(E) into three routing subtrees each of which carries a different ERN number.

ALGORITHM 3.2 suggests a new approach for selecting routes and assigning ERN numbers. Namely, construct k routing trees from a network such that each routing tree contains those nodes of the network each of which appears at most once in the tree. Such a tree can be a spanning tree of the network rooted at a given destination node. This approach eliminates the problem of ERN assignment, however, it has the serious drawback that it is difficult to control the quality of the selected explicit routes. In particular, the same physical route may appear more than one routing tree.

ALGORITHM 3.3 (Based on modified bin packing)

First provide sufficiently many bins each of which carries a different ERN number, and arrange them in order of increasing ERN numbers.

Proceeding through the prime routes in some sequence, assign each prime route to the bin with the smallest ERN number into which it can fit with no conflicts with any of routes already been assigned to that bin. This packing is a "first-fit" packing. One may consider any other packing such as "best-fit" packing by defining a measure for packing efficiency.

In the implementation of ALGORITHM 3.3, one may use the bin illustrated in Fig. 3.5. Here, N_1 through N_n denote the names of nodes in the network, and each entry in the bin will indicates the next node to be visited using a given transmission group. For example, $(N_j \cdot g_j)$ in the figure indicates that the bin contains the route segment from node N_i to N_j using the transmission group g_j.

The following procedure can be used to pack an explicit route into a bin.

Let $N_1(g_1),\ldots,N_i(g_i),N_{i+1}(g_{i+1}),\ldots,N_{m-1}(g_{m-1}),N_m$ denotes an explicit route to be packed into the bin with ERN=r. Then repeat the following for i=m-1 to 1.

Get a segment $(N_i(g_i),N_{i+1})$ and put $(N_{i+1} \cdot g_i)$ into the bin location N_i if the location is not yet occupied (if the location is occupied by $(N_{i+1} \cdot g_i)$ then the packing so far is successful). If the location is occupied by something other than $(N_{i+1} \cdot g_i)$, then the packing of the route fails (in this case, the bin must be reset to the condition just before the packing, and the next bin is examined).

Fig. 3.6 illustrates the bin packing of those routes shown in Table 1 into three bins each of which is identified by the ERN number. One should note from this figure as well as Figures 3.2 and 3.4 that 3 ERN numbers were required to identify 2 explicit routes. Fig. 3.7 shows a portion of the DN/ERN routing table at node C for the routes shown in Table 1.

The advantages of the DN/ERN explicit routing algorithm are, it is conceptually simple and requires less storage for routing tables compared to that for ON/DN/ERN approach. The size of the routing table S is bounded by $k(n-1)(n-2)/2>S\geq k(n-1)$, and its actual size is dependent on the routes to be define in the routing tables. The major drawback is that the associated ERN numbering problem is an NP-complete problem; the number of ERNs, r, required to distinguish k routes between a pair of nodes can only be bounded by $k(n-1)>r\geq k$. In practice, the maximum r is limited by a particular network implementation (in most cases, it is less than $k(n-1)$). Thus, when r is fixed where $k\leq r$, it is quite possible that r ERNs are not sufficient to identify k routes (or not all routes can be defined in the routing table). Since one can not pre-determine whether or not r ERNs are sufficient to identify k routes between each pair of nodes, one must first run a ERN assignment algorithm and then reduce the number of routes to be defined in the routing table if more than r ERNs are required, which is a computationally expensive process.

4. THE DN/ERN/IX EXPLICIT ROUTING APPROACH

In this approach, an explicit route is identified by the pair (DN,ERN) at the packet origin node, and by the pair (DN,IX) at an intermediate node. The explicit route index, IX, is introduced here to allow the maximum sharing of route segments to a particular destination node. Each packet header carries ON, DN, ERN and IX. The routing table consists of fields for DN, ERN/IXR, IXS and NN.TGN. Here, IXR and IXS denote the received IX and sent IX, respectively. ERN/IXR indicates that the field can be accessed either by ERN or by IXR. This is depicted in Fig. 4.1.

At the packet origin node, the pair (DN,ERN) is used to locate the corresponding IXS and NN.TGN values. The IXS value will be stored in the IX field of the packet header, and then the packet will be sent to the next node NN using TGN. At an intermediate node, the pair (DN,IX) in the packet header is used to match the fields DN and IXR in the routing table to locate the corresponding routing entry. The corresponding IXS value will be stored in the IX field of the packet header, and then sent to the next node NN using TGN. This routing process may be called an IX swapping operation. As can be seen, this algorithm allows the sharing of route indexes by swapping indexes at each intermediate node visited.

The size of the routing table is equal to (s+k)(n-1), where s is the number of support routes at the node, k is the number of explicit routes between a pair of nodes and n is the number of nodes in the network. A route from a node is said to be a support route if it is strictly used by the packet originated at some other nodes and is not used by the packet originated at that node. The sizes of indexes: IX, IXR and IXS, are all equal to (s+k).

The algorithm for the generation of routing tables for DN/ERN/IX explicit routing algorithm is described next. It is optimal in the sense that it requires the minimum number of ERNs and IXes for identifying given explicit routes.

ALGORITHM 4.1

(1) Construct a routing tree T(DN) which contains all given routes to the destination node DN such that each route from a leaf node to the root node in the tree represents a prime route. Then assign indexes to the nodes of the tree in the following manner:

Assign an index 1 to the root of the tree. Assign different indexes to those with the same node name (one may assign a smaller index value to the node in the tree which represents a preferred route).

(2) Construct routing tables in the following manner:

For each parent-child node pair in the tree, create a routing entry in the routing table at the node represented by the child node such that

DA=(the root of the tree),
NN=(the name of the parent node),
TGN=(TGN of the child node),
IXR=(the index assigned at the child node),
ERN=IXR, and
IXS=(the index assigned at the parent node).

Fig. 4.2 illustrates an example of IX assignment on the routing tree T(E) for the explicit routes shown in Table 1. Fig. 4.3 shows a portion of the DN/ERN/IX routing table at node A for the IX assignment given in Fig. 4.2.

As can be seen from ALGORITHM 4.1, ERN assignment and IX assignment are carried on simultaneously and simply. The IX assignment algorithm described is optimal in that it requires the least number of indexes. Furthermore, the number of IXes and ERNs required in the algorithm is always less than or equal to the number of ERNs required for the DN/ERN algorithm (this is due to the fact that the DN/ERN algorithm requires ERN assignments to routes while the DN/ERN/IX algorithm requires IX assignments to nodes). The number of IXes required is exactly equal to the number of routes to a particular destination node that the node must support, which is the number of node appearances of the same node name in the routing tree. Thus, if k routes were pre-determined from a particular node to a destination node and if that node name appears (k+s) time in the routing tree, then s indicates the number of support routes that the node must provide.

Fig. 4.4 shows the number of ERNs (or IXes),r, actually required to identify given number of explicit routes between every pair of nodes for different routing algorithms. Although, we do not show the actual networks examined, we considered 3 different size mesh connected networks. For each pair of nodes in the example, k explicit routes which maximize the point-to-point reliability (the probability that two nodes are connected) were selected assuming statistically independent network element (nodes and links) failures. Furthermore, for each destination node, routes were selected so that no support route exists (i.e., s=0). Fig. 4.4 shows that both the ON/DN/ERN and the DN/ERN/IX routing algorithms require the same number of ERNs (or IXes) as the number of explicit routes to be defined in the routing tables. It also shows that the DN/ERN routing algorithm requires more ERNs than the actual number of routes defined in the routing tables. This requirement is measured by the slope of the curve and seems to increase as the size of the network increases.

5. CONCLUSION

We have described three basic ways to implement explicit routing algorithms: the ON/DN/ERN algorithm, the DN/ERN algorithm and the DN/ERN/IX algorithm. For each we have described a routing table generation algorithm assuming that the routes to be defined in the routing tables were given. These generation algorithms were all implemented in PL/1 and tested on an IBM370/168.

The ON/DN/ERN algorithm offers some advantages over the other two algorithms; it is simple, it allows arbitrarily route selection between every pair of communication nodes, and its routing table generation is trivially simple. However, it has the serious drawback that its routing table size is proportional to the square of the number of communication nodes in the network. The DN/ERN algorithm offers routing simplicity, and some reduction of the routing table size. However, it has a drawback that multiple ERNs are often used up for identifying a single physical route. This means one must either provide large number of ERNs to define given number of routes in the routing tables or one can only support limited number of routes when the number of available ERNs is fixed. Another drawback of this algorithm is the complexity associated with the ERN assignment in the process of generating routing tables; the ERN assignment problem is NP-complete, and thus one must rely on a heuristic ERN assignment algorithm such as the ones discussed in Section 3. The DN/ERN/IX algorithm offers the most compact routing tables and its routing table generation is fairly simple. Its drawbacks are its conceptual complexity because of swapping IX values at intermediate nodes, and possible additional cost associated with the IX swapping.

In conclusion, it seems that the most practical, most elegant and most powerful implementation of the explicit routing algorithm of those considered in this paper is the DN/ERN/IX routing algorithm.

REFERENCES

[AH74] Aho, A. V., Hopcroft, J. E. Ullman, J. D., 'The design and analysis of computer algorithm', Addison-Wesley, 1974.

[AH79] Ahuja, V. 'Routing and flow control in Systems Network Architecture', IBM Systems Journal, Vol.15, No.2, 1979, pp.298-314.

[GR79] Gray, J. P. and McNeill, T. B., 'SNA multiple system networking', IBM Systems Journal, Vol.15, No.2, 1979, pp.263-297.

[JU76] Jueneman, R. R. and Kerr, G. S., 'Explicit path routing in communications networks', Proc. 3rd ICCC, Toronto, Canada, Aug. 3-6, 1976, pp.340-342.

[NA79] Natarajan, K. S., Tang, D. T. and Maruyama, K., 'On the selection of communication paths in computer networks', Computer and Networking Symposium 1979, Gathersburg, Md, Dec. 12, 1979.

[RA78] Rajaraman, A., 'Routing in TYMNET', Proc. ECC 1978, London, England, May 9-12, 1978, pp.9-21.

[RO72] Roberts, L. G. and Wessler, B., 'The ARPA computer network', Computer Communication Networks, ed. by N.A. Abramson and F.F. Kuo, Prentice Hall, 1972, pp.485-500.

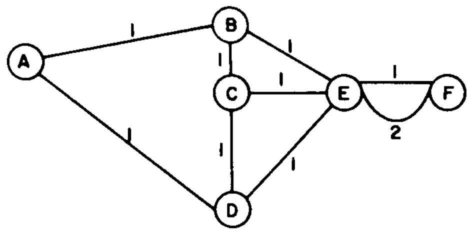

Fig. 1.1 A Simple Network

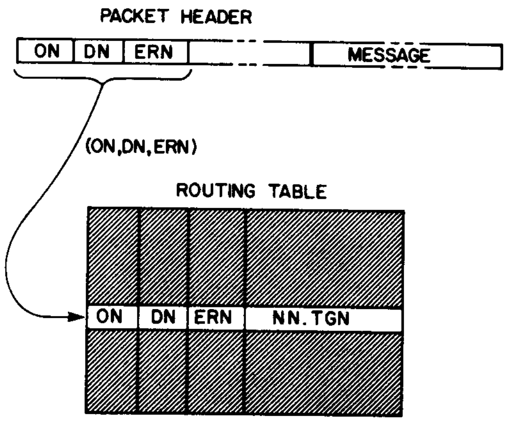

Fig. 2.1 The ON/DN/ERN Routing Algorithm

ON	DN	ERN	NN.TGN
A	E	1	B.1
		2	D.1

Fig. 2.2 ON/DN/ERN Routing Table at Node A

FROM	TO	EXPLICIT ROUTE
A	E	A(1),B(1),E A(1),D(1),E
B	E	B(1),E B(1),C(1),E
C	E	C(1),E C(1),B(1),E
D	E	D(1),E D(1),C(1),E
F	E	F(1),E F(2),E

Table 1 Two Explicit Routes to Destination Node E

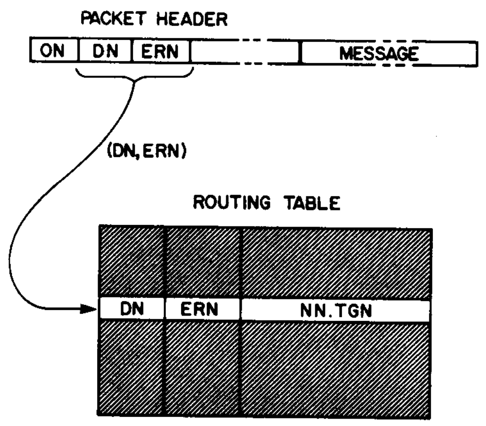

Fig. 3.1 The DN/ERN Routing Algorithm

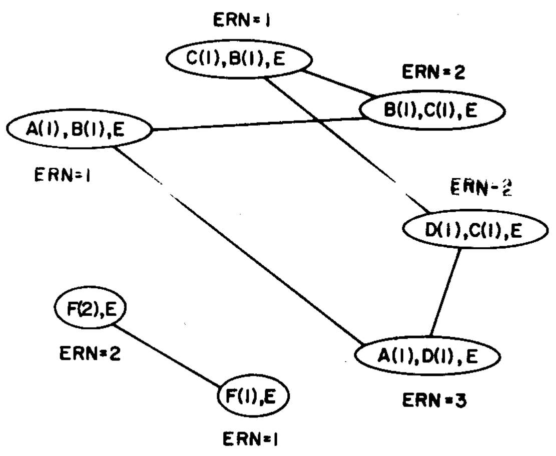

Fig. 3.2 The Contention Graph for The Routes in Table 1

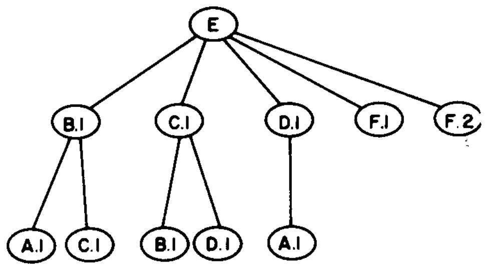

Fig. 3.3 Routing Tree T(E)

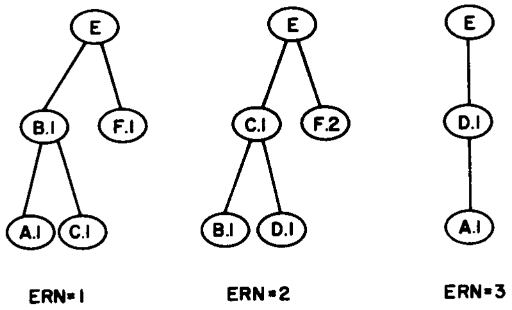

Fig. 3.4 Routing Tree Decomposition of T(E) in Fig. 3.3

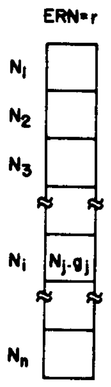

Fig. 3.5 A Bin Configuration

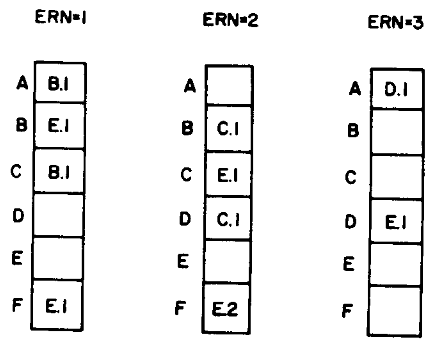

Fig. 3.6 Bin Packing of The Routes in Table 1

DN	ERN	NN.TGN
E	1	B.1
E	3	D.1

Fig. 3.7 The DN/ERN Routing Table at Node A

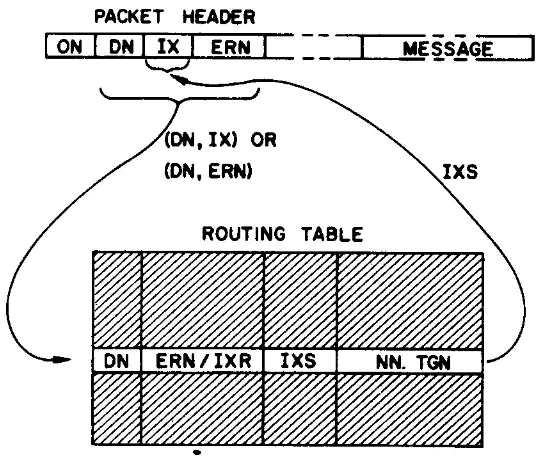

Fig. 4.1 The DN/ERN/IX Routing Algorithm

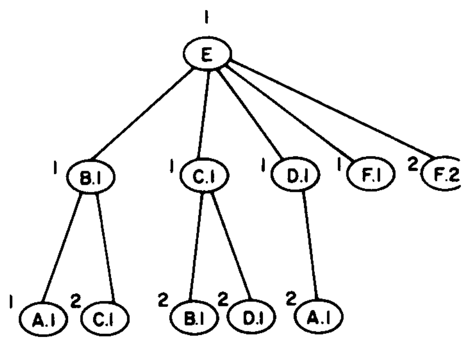

Fig. 4.2 The IX Assignment on The Routing Tree T(E)

DN	ERN/IXR	IXS	NN.TGN
E	1	1	B.1
E	2	1	D.1

Fig. 4.3 The DN/ERN/IX Routing Table at Node A

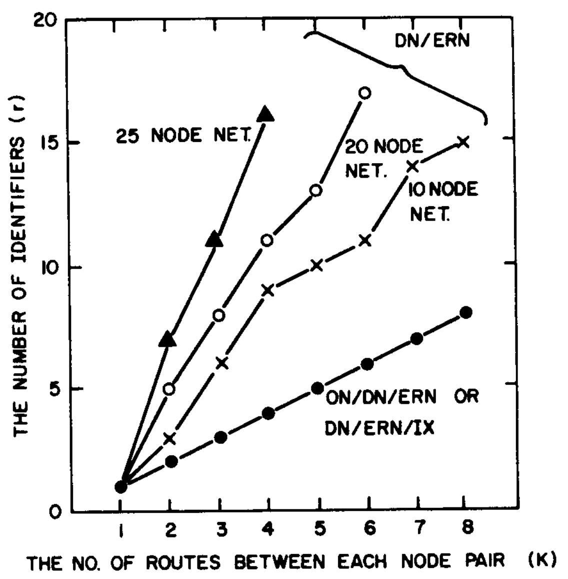

Fig. 4.4 The Number of Identifies (r) Required to Define k routes