Defining routing tables for SNA networks

by K. Maruyama

This paper addresses three basic problems associated with the definition process for the routing tables of IBM's Systems Network Architecture (SNA). The paper then introduces a program called the Routing Table Generator (RTG) and describes how these problems were solved with RTG. Also discussed are some approaches on how to use RTG in managing routing tables for growing networks.

Since the first announcement of Systems Network Architecture (SNA) in 1974, many new functional enhancements and improvements have been introduced to SNA users in the form of new versions and releases.[1] Prior to the announcement of Advanced Communication Function (ACF) Release 3 products,[2] the topological configurations and networking functions were rather limited. In the pre-ACF Release 3 products, the network configuration could be either a "tree" configuration, as illustrated in Figure 1, or a "multitree" configuration, as illustrated in Figure 2, both with the distinction of "local" and "remote" communications controllers (or ACF/NCPs, where NCP is the Network Control Program). ACF Release 3 products were the first SNA products that eliminated many configurational and functional limitations by introducing the multiple path routing capability, parallel links, multiple transmission groups, etc., bringing SNA into the "true" mesh networking as depicted in Figure 3. Further functional enhancements, which include the host intermediate node,[3] channel-to-channel adaptors, and SNA Network Interconnection, were incorporated into SNA and were announced as ACF Version 2 products.[4]

With respect to routing in SNA,[5] a major change occurred when the ACF Release 3 products were introduced in 1979; single path routing advanced to multiple path routing, known as explicit path routing.[6] Explicit path routing provides many advantages[7] over single path routing, such as added communication path availability between communication node pairs, increased throughput by separating sessions over multiple paths, improved response time, and the physical separation of one type of traffic (e.g., batch) from other types (e.g., interactive). However, the implementation of explicit path routing in SNA created complexities in the definition, generation, and management of routes and routing tables.

This paper addresses problems associated with the definition and generation of the explicit path routing tables known as transit routing tables.[1,7] It then describes a program called the Routing Table Generator[8] that was introduced in 1981 to aid in the process of route definition. The paper further describes the techniques used by this product to handle the network definition problems.

Routing in ACF R3 networks

In SNA, a logical connection called a "session" is set up between each pair of network addressable units that desire to exchange messages. Such a session is assigned to a single physical route, called an *explicit route*, at the time of the session initiation. The session will stay on the route until either session

Figure 1 "Tree" configuration

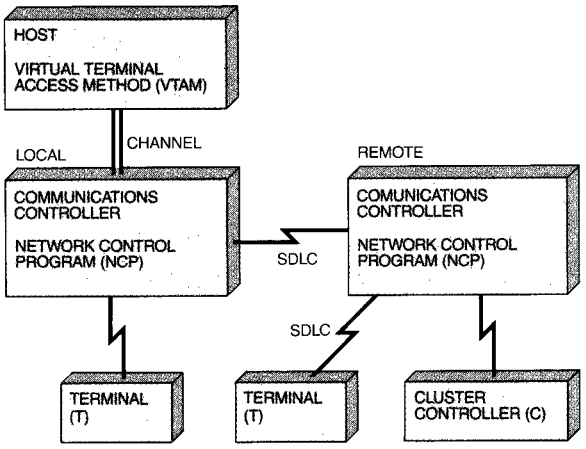
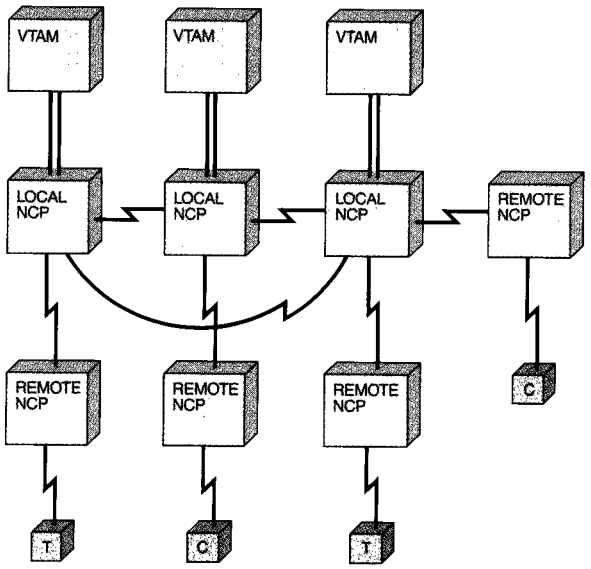


Figure 2 "Multitree" configuration



termination or route failure occurs. The mapping of sessions to explicit routes is determined using a set of system-generated tables and the availability of network resources. At the session initiation time, a control point called SSCP (System Services Control Point) that assists session establishment uses a log-on-mode table (in the case of ACF/VTAM) or a bind-image table (in the case of ACF/TCAM) to determine the name of a proper Class of Service (COS) table. A COS table contains a preferential list of potential routes called *virtual routes* (VRs). A virtual route is a logical route between two end points and provides among other things the end-to-end flow control. Each session will be assigned to the very first operational virtual route in the identified COS table. In the existing SNA products, each virtual route is mapped to a unique explicit route (ER), and up to eight explicit routes can be defined and activated between each pair of subarea nodes such as hosts and ACF/NCPs. An explicit route is an ordered sequence of subarea nodes and of links

In the current SNA products, the definition of routing tables is a part of the system definition/generation process.

called *transmission groups* (TGs) from the subarea node of its origin to its destination subarea node. A transmission group is a grouping of parallel links between adjacent subarea nodes and is viewed as a logical link. Each explicit route, which is unidirectional, is identified by a number called an *explicit route number* (ERN). It is required that each explicit route defined in the routing tables must be accompanied by its reverse explicit route, i.e., the same physical path in the opposite direction. However, the ERN assigned to one direction of a route need not be the same as the number assigned to the reverse direction. An explicit route accompanied by its reverse route is called a physically reversible ER. The use of paired, physically reversible ERs simplifies the failure notification in the network since it causes both directions of traffic flow to fail simultaneously.[1]

Although an explicit route is defined as an ordered sequence of subareas and transmission groups from the subarea of its origin to its destination subarea, no single subarea node has an understanding of the complete sequence. Each explicit route is broken down into one or more route segments that are stored in a subarea node table called a *transit*

routing table.[1] A transit routing table consists basically of three fields: destination subarea (DSA) field, ERN field, and next node/transmission group (NN/TG) field, which indicates the outgoing transmission group queue leading to the next node (NN). When a message called a *Path Information Unit* (PIU) is processed for transmission at a subarea routing node, the node finds a routing entry that corresponds to the destination subarea number (DSA) and the ERN stored in the message transmission header (TH). The message is then placed in an appropriate outgoing transmission group queue for transmission. This situation is illustrated in Figure 4.

In the current SNA products, the definition of routing tables is a part of the system definition/generation process. Therefore, the system/network administrator must create input statements, called *PATH macros*, to be used for the system generation of the transit routing tables. The definition of routing tables, however, is considered to be the most complex, time-consuming and human-error-prone process in SNA system generation.[9] This paper will discuss those problems associated mainly with the definition of PATH macros.

Basic problems involved in defining routing tables

There are three basic processes involved in the definition of the transit routing tables. These are

1. The route selection process
2. The route numbering process
3. The PATH macro generation process

The route selection process. The route selection process involves the selection of physically reversible explicit routes between each subarea node pair where SNA sessions will be created. Each reversible explicit route must satisfy SNA product constraints. Depending on the version, release, and physical environment, a subarea node may or may not support the message forwarding function known as the intermediate network node (INN) function and/or the multiple path routing function. Table 1 summarizes the functional difference among some products. If a subarea node does not support the INN function, one must not select a route that traverses that subarea. Such a subarea can only be the end node of explicit routes. If a subarea node does not support the multiple path routing function, one must not select more than one route that traverses or originates from that node to each destination node.

Figure 3 "Mesh" configuration

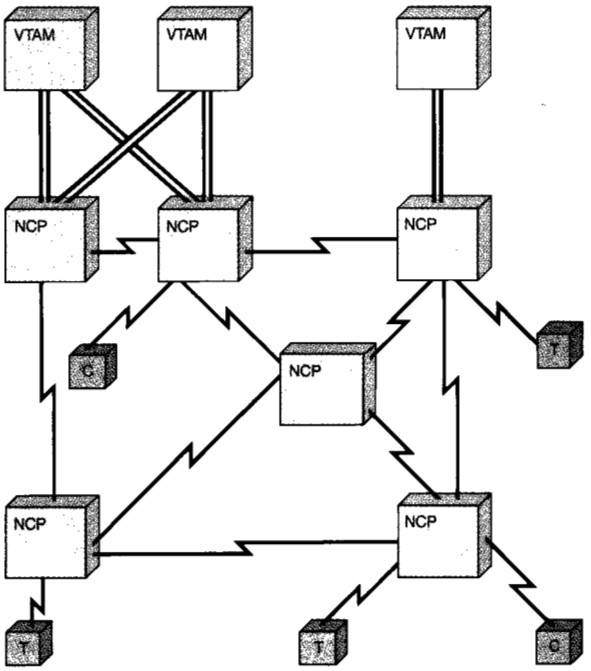
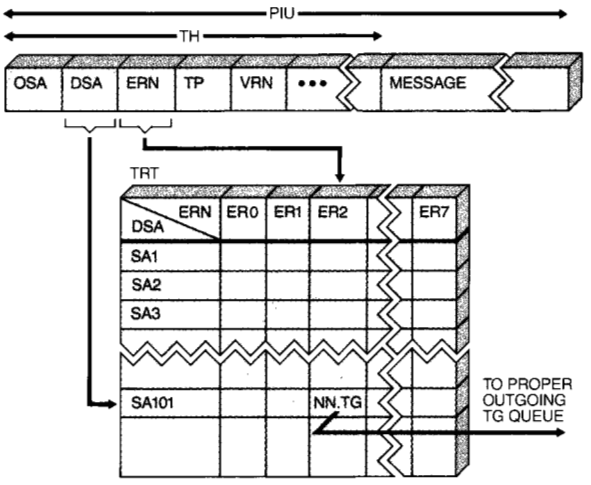


Figure 4 Explicit path routing in ACF R3



Among those routes that satisfy the above product constraints, one may select some routes for each node pair by considering some route characteristics derived from the characteristics of components (nodes, links) in the network. There are two types of characteristics: those associated with an individual

Figure 5 A simple network

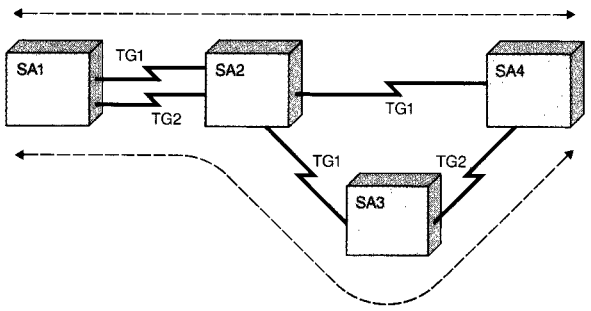


Table 1 Functional differences among ACF products

Product	INN	Multiple Paths
ACF/VTAM/V1R1	yes	
ACF/VTAM/V1R2		
ACF/VTAM/V1R3		yes
ACF/VTAM/V2R1	yes	yes
ACF/VTAM/V2R2	yes	yes
ACF/VTAME	yes	
ACF/VTAM/V2R1/CA	yes	yes
ACF/TCAM/V1	yes	
ACF/TCAM/V2R1		
ACF/TCAM/V2R2		
ACF/TCAM/V2R3		yes
ACF/TCAM/V2R4		yes
ACF/NCP/V1R1	yes	
ACF/NCP/V1R2	yes	
ACF/NCP/V1R2.1	yes	
ACF/NCP/V1R3	yes	yes
ACF/NCP/V1R4	yes	yes
ACF/NCP/V2R1	yes	yes
ACF/NCP/V2R2	yes	yes

route and those associated with a set of routes between a node pair.[10,11] The characteristics that may be considered for selecting an individual end-to-end route are the physical distance of the route, the route length in terms of the number of transmission groups along the route, the route capacity or speed, the route availability/reliability, the route error rate, the route security level, and the estimated delay on the route. The characteristics that may be considered for the selection of a set of routes between a node pair are some sort of averages of those characteristics considered for the individual route selection (e.g., the average route length), the end-to-end route availability (the probability that at least one route is available/operational), and the disjointedness among routes.

It is quite plausible to perform a manual selection of a set of reversible explicit routes for each node pair for relatively small networks. It, however, becomes quite a time-consuming and potentially error-prone process as the size of the network grows. Unfortunately, this route selection process is further aggravated by its dependence on the route numbering.

The route numbering process. Route numbering is the most time-consuming and human-error-prone process in the definition of SNA routing tables. In order to carry out the route numbering (or the ERN assignment) process properly, one must pay special attention to the following characteristics and restrictions:

a. ACF/R3 routing is "source-independent, destination/route-number"-based routing.
b. Explicit routes (ERs) must be physically reversible.
c. Only eight ERNs are available for each destination node. (SNA architecture supports 16 ERNs.)
d. The ERN assigned to an ER can be different from the ERN assigned to its reverse ER.
e. The migration route[12] must be numbered zero (or ER0) for both directions.

Characteristic a implies that each routing entry for a particular destination in a transit routing table will be used/shared by many routes that originate from other nodes. This condition can be seen from Figure 5, which illustrates the transit routing table used by an SNA subarea node. From this table, one can observe that the only entries used to route messages are the destination subarea number (DSA) and the explicit route number. The source information (OSA) of the message is not used for the routing decision. Therefore, the numbering of one route, which is in a sense equivalent to creating all routing entries associated with that route, affects the numbering of other routes to the same destination. This effect requires that the route numbering process must simultaneously consider all of the routes selected for the same destination node.

Characteristic b aggravates the process of route numbering since it requires a successful route number assignment to a pair of physically reversible explicit routes. The route numbering can be successful all of the time if an unlimited number of

route numbers are available. This, however, is not the case, as restriction c states.

Restriction c implies that when routes are numbered, the assignment of more than one route number to the same physical route should be avoided, though such avoidance may not be possible in some cases. The phenomenon that requires more route numbers than the number of physically different routes defined in the routing tables for each node pair is often called "ERN Starvation" or "ERN Explosion."[9] Characteristic d relaxes the numbering restrictions and reduces the rate at which a physical reversible route requires a new route number for assignment.

The PATH macros are the external representation of SNA's transit routing tables.

In ACF Release 3 (ACF/R3), a route is said to be a migration route if it traverses one or more pre-ACF/R3-level nodes. Restriction e states that such a migration route must be numbered zero for both directions because the pre-ACF/R3 products did not use the transit routing table indexed by DSA and the ERN, but by DSA only. By using ER0, the sending ACF/R3 node can delete ER0 and create a message header understood by the receiving pre-ACF/R3 node.

PATH macro generation process. The PATH macros are the external representation of SNA's transit routing tables. The macros are used in the generation process by IBM products to generate the transit routing tables at subarea nodes. The basic format of a PATH macro is

```
PATH      DESTSA=sa1 | (sa1,sa2, ....),
          ER0=(adjsa,tgn),
          ER1=(adjsa,tgn),
           .
           .
           .
          ER7=(adjsa,tgn)
```

where sa, sa1, sa2, and adjsa indicate subarea addresses and tgn indicates a transmission group number. The PATH macros for ACF/VTAM, ACF/TCAM, and some ACF/NCP releases contain additional information for mapping virtual routes (VRs) to explicit routes (ERs). The format for such a mapping function is

```
VR0=ern,
VR1=ern,
 .
 .
 .
VR7=ern
```

An example of a PATH macro is

```
PATH      DESTSA=103,
          ER0=(101,1),ER1=(101,2),
          ER2=(102,1),ER3=(101,1),
          VR0=3,VR1=0,VR2=2,VR3=1
```

The above macro indicates the definition of four explicit routes and four virtual routes from the point of view of the subarea node where this macro is processed to the destination node 103. When messages are sent to destination node 103 using ER0, for example, the message will be transmitted to the next subarea node 101 via transmission group 1. There are two types of messages that use the explicit route ER0: the messages originating at the node where the macro is processed and sent via VR1, and the messages arriving at the node from other source nodes and using the node as an intermediate network node (INN) to get to destination 103. The second type of messages exist when the node processing the PATH macro supports the INN function.

Once the route numbering process is completed successfully, generating PATH macros becomes a trivial mechanical process. For example, let us consider the following two routes between subareas SA1 and SA4 of the network illustrated in Figure 5:

To SA4		To SA1
ER0	SA1 - TG1 - SA2 - TG1 - SA4	ER0
ER1	SA1 - TG2 - SA2 - TG1 - SA3 - TG2 - SA4	ER2

The PATH macros for the subarea SA2, for example, can be generated from the above explicit routes by looking at SA2 and its adjacent subareas and transmission groups. The PATH macros at SA2 are

```
PATH      DESTSA=4,
          ER0=(4,1),ER1=(3,1)
PATH      DESTSA=1,
          ER0=(1,1),ER2=(1,2)
```

Table 2 Topological information

Type	Parameter	Description
Subarea	SANAME	Name of the subarea node
	SANUM	Address of the subarea node
	PULVL	Name of the software product
Link	LNAME	Name of the link
	ADJSAS	Adjacent subarea pairs
	TGN	Transmission group number
	MED	Link medium
	PROTOCOL	Full duplex or half duplex
	PDLY	Propagation delay of the link
	ERR	Error rate of the link
	AVL	Availability of the link
	SEC	Security level of the link
	TGC	TG classes of the link
	VAL	Any value assigned to the link

Table 3 Control parameters

Type	Parameter	Description
Route	ER	Explicit route
	ERN	Forward ER number
	RERN	Backward (reverse) ER number
	VRN	Virtual route number
Declare	ERDEF	Defining ERs between node pairs
	ERUND	Not defining ERs between node pairs
	VRNUM	Usable VRNs
	ERNUM	Usable ERNs
	ERLEN	The maximum ER length
	ERSEL	Optimal and back-up route selection
	SUBNET	Defining the third-party-owned network and/or negating the INN function from a node
	VTAMPM	Controlling the format of VTAM PATH macros
	ERPRINT	Printing ERs using names or addresses of subarea nodes
	LISTING	Controlling the amount of output information
	PMFILE	Creating or not creating a PATH macro file
	RSFILE	Creating or not creating a ROUTE statement file

As will be seen later, the proper selection of the data structure for the explicit routes in a routing table generation program will further simplify the process of PATH macro generation.

Routing table generator—Assumptions and requirements

For the generation of PATH macros, a software product called the Routing Table Generator (RTG) was developed.[8] Since RTG was aimed at various SNA users with varying degrees of constraints and requirements, certain assumptions were made.

It was assumed that the only information to be made available to RTG was the network topology without workload information. There were a few reasons behind adopting this assumption: RTG was to be simple to use without requiring a large amount of input from users; the workload information (session types, characteristics, and the number of sessions initiated between each node pair) is not easily obtainable or known, at times, prior to the installation of the networks; and the workload changes with time. It was concluded that it is wise not to perform optimization based on uncertain and time-dependent information. Another assumption was that no two network administrators/designers would have the identical routing requirements even for the same network topology. Therefore, some ways should be available to tell RTG the specific user requirements on the PATH macros that it generates. Some of the user requirements for which functions are available in RTG are listed below:

Three cases of route selection are handled by RTG.

a. The user should be able to tell RTG to include specific explicit routes (with or without specific explicit route and virtual route numbers) in the routing tables.
b. The user should be able to tell RTG to select routes between certain node pairs.
c. The user should be able to control the number and the maximum length (hop count) of selected routes for each node pair.
d. The user should be able to specify an arbitrarily defined route selection criterion (or objective

function) for the selection of optimal and back-up routes. (The reason is that different users generally want the routes with different route characteristics.)

e. The user should be able to choose not to use the intermediate network node function from certain nodes. (One may use this function to avoid the definition of explicit routes which go through the nodes.)
f. The user should be able to control the usage of VRNs and ERNs. (Such control allows the reservation of certain route numbers for later use as the network topology changes.)
g. The user should be able to control the amount of the output information from RTG and should be able to specify the kind of information wanted.

The functions for the above requirements were made available in RTG in the form of "control parameters" to RTG. The overall conceptual view of RTG and the network topology input and the control parameters are illustrated in Figure 6. The detailed information on the network topology, the control parameters, and RTG outputs is summarized in Table 2, Table 3, and Table 4, respectively. In the "type" column of Table 3, requirement a is satisfied by Route and requirements b through g are satisfied by Declare.

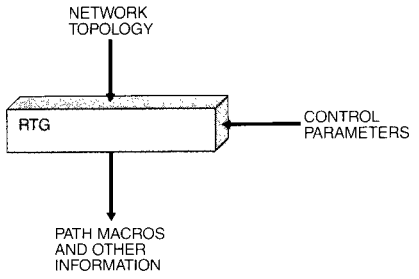
Route selection processes in RTG

Three cases of route selection are handled by RTG: the selection of optimal routes with or without back-up routes, the selection of a set of routes for each node pair, and the selection of "migration" routes.

Table 4 Output

Type	Description
Output Report	Listing of the input to RTG Diagnostic messages Address-name table Link-characteristic table TG-characteristic table VRN-ERN-ER table VRN-characteristic table VRN-performance table TG-usage table
PATH Macros	PATH macros for each subarea node
ROUTE Statements	Listing of all explicit routes supported by the generated PATH macros

Figure 6 Conceptual view of RTG



The selection of an optimal route with or without back-ups. Sometimes it is desirable for the RTG users to be able to select an optimal route[13] for each node pair by simply specifying an optimality criterion. Some optimality criteria are additive (such as distance, length, delay), and some are nonadditive, requiring minimization, maximization, or multiplication of the weights along a route (such as capacity, reliability, availability, security, and error rate). Since it is not practical to implement an optimal route selection algorithm for each route characteristic, generalized versions of the Floyd[14] and Dijkstra[15] algorithms were developed[11] and used in RTG. The user specifies in ERSEL (see Table 3) an ordered list of route characteristics; the characteristics placed in the second, third, etc., positions are used for tie-breaking among optimal routes. Those route characteristics available in RTG are listed in Table 5. For example, the user may specify ERSEL [route length (RL), route TG capacity (TC)] for the selection of an optimal route for each node pair. RTG selects minimum hop routes. When there is more than one minimum hop route between a node pair, RTG will use the route capacity information to break ties among routes and select the one with the maximum TG capacity.

There is always the possibility that the user's desirable route characteristic is not among those in Table 5. To cope with such a problem, RTG provides a way to define any new route characteristic, called the "preferred route" definition. Another possibility is that the user may want RTG to select optimal routes based on a certain subset of the network and not on the entire network. Such a function is also supported by RTG and is called "restricted route" selection.

Table 5 Route characteristics

Route Characteristics	Notation
Route length (hop count)	RL
Short PIU delay	SD
Long PIU delay	LD
Short PIU transmission time	ST
Long PIU transmission time	LT
Propagation time	PT
Route TG capacity	TC
Route link capacity	LC
Route availability	RA
Route quality	RQ
Route security level	RS

In addition to the route characteristics, the user may specify one or more virtual route numbers (VRNs). The first VRN will be assigned to the optimal route between each node pair. When more than one VRN is specified, which implies the selection of back-up routes, RTG will select the next optimal route and assign the next VRN. This process continues when many VRNs are specified. For example, ERSEL (VR0, VR1, VR2, route-length) instructs RTG to select three minimum-hop routes and assign them VR0, VR1, and VR2 in the order of increasing length for each node pair.

The selection of a set of routes for each node pair. Similar to the selection of optimal routes, one can consider the "average" characteristic such as the average delay and the average route capacity of a set of routes to determine the optimal set for each node pair. The use of this average route characteristic for the selection of a set of routes has a serious drawback in its tendency to select similar routes. Another approach is to select disjoint (either node-disjoint or link-disjoint or a mixture) routes for each node pair by maximizing the total capacity or by minimizing the average length, etc.[10] Modified versions of the Min-cut, Max-flow algorithm[16] are often used for the selection of disjoint routes.

Another approach, which has been adopted for RTG, considers both disjointness and the quality of routes and is a hybrid of the above two approaches. Instead of enforcing the selection of disjoint routes, it enforces the selection of routes that maximize the end-to-end availability, which is defined as the probability that at least one route in the set is operational. This approach of selecting routes by maximizing the end-to-end availability makes sense because (a) some disjoint routes are not always desirable; (b) the disjoint routes do not necessarily offer the maximum total capacity; and (c) no disjoint route may exist between some node pairs.

One can see easily that the brute-force implementation of this technique is computationally prohibitive and requires substantially large storage space. In RTG, up to 128 candidate routes are enumerated for each node pair, and from them eight routes are selected by using a technique called "sequential availability maximization." This technique begins with an optimal route and selects the next route which maximizes the pair-wise availability sum (the summation of the availabilities computed from all possible route pairs). Once the second route is determined, the algorithm looks for the third route, which again maximizes the pair-wise availability sum and continues this sequential maximization process until all eight routes have been selected. The reasons for using the pair-wise availability sum instead of computing the exact end-to-end availability are that (a) the exact computation is computationally expensive, and (b) it is not necessary to compute the exact end-to-end availability but rather to select routes which maximize availability.

The control parameter available in RTG for instructing route selection/definition between user-specified subarea node pairs is ERDEF (see Table 3). For example, let us consider the selection of up to four routes of length no more than five between subarea nodes 101 and 102, and the selection of up to eight routes of any length between subarea nodes 101 and 103. The complete ERDEF describing such route selection is ERDEF ((101,102:4:5) (101,103:8)).

The selection of migration routes. In SNA, a route is called a migration route[12] if it traverses at least one subarea node with the pre-ACF/R3 product. A migration route is required to be numbered ER0 for both directions. In RTG, if a network contains one or more pre-ACF/R3 products, it first selects one route for each node pair and assigns ER0 for both directions whether or not the route is a true migration route. Once a migration route (here in the sense that the route has ER0s for both directions) is supported for each node pair, it becomes a lot easier to introduce pre-ACF/R3 products (see Table 1) into the network at any later time. This is because the attachment of a pre-ACF/R3 product to a product that has migration routes to other nodes in the network requires simple extensions of existing migration routes. However, the attachment of a

pre-ACF/R3 product to a product that does not have migration routes to other nodes in the network requires the reassignment of ERNs to explicit routes, leading to a major system regeneration.

Two conditions must be satisfied by the migration routes in the network: (1) each route must be physically reversible (as any other explicit route) and (2) for each destination node, the union of all migration routes going toward the destination nodes must form a "tree" rooted at the node (i.e., the union of these routes is a graph characterized by the property that every node except the root node has exactly one outgoing link). For example, let us consider the following three routes going toward the destination node SA4:

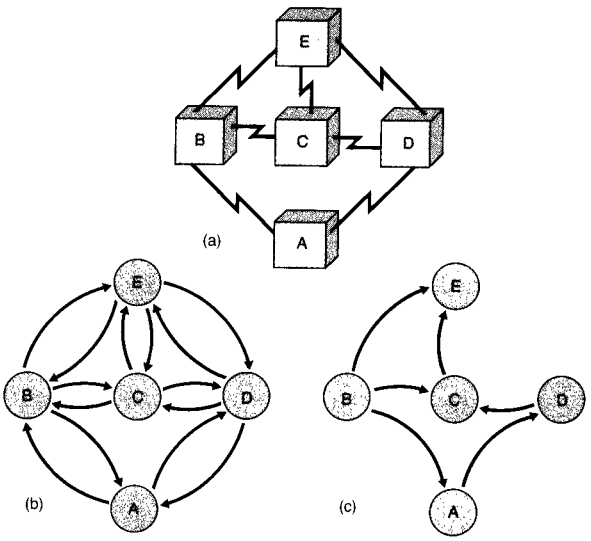
a. SA1 - TG1 - SA2 - TG1 - SA4
b. SA5 - TG1 - SA2 - TG1 - SA4
c. SA2 - TG1 - SA3 - TG1 - SA4

The union of routes a and b does form a tree rooted at SA4. However, the union of routes a and c does not form a tree rooted at SA4 since the graph formed by the union of these routes has two outgoing links (or transmission groups) at node SA2 going toward the destination node SA4. Condition 2 makes it possible to assign a single explicit route number, ER0, to all of the migration routes. The question here is how to select optimal routes which meet these conditions.

As one can see, it is trivial to select optimal routes that satisfy either the reversibility requirement or the tree-forming requirement. It is, however, not so trivial to satisfy these two requirements simultaneously. A simple application of an optimal route selection algorithm, such as the Floyd[14] and Dijkstra[15] algorithms, does not guarantee the selection of routes that satisfy those requirements simultaneously. This situation results because an optimal route for a node pair is not necessarily unique.[17]

Two approaches exist for selecting optimal routes that meet these two conditions. The first is to introduce a special tie-breaking rule into an optimal route selection algorithm so that the algorithm will guarantee a unique optimal route selection. There are two ways to break ties among optimal routes: (1) by lexicographical comparison which assigns a unique label to each transmission group or (2) by introducing an infinitesimally small perturbation into the weights of the transmission groups.[11] The second approach is to explicitly enforce the reversibility and tree-forming properties among selected routes.

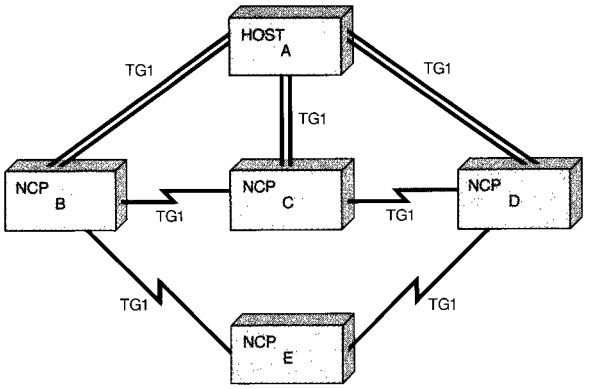
Figure 7 Imbedding hand-picked route



After a careful study of implementation alternatives, the second approach was adopted for RTG. This approach first selects optimal reversible routes for all node pairs and then modifies them to form a tree for each destination without losing the reversibility property.[18]

There are cases in which the RTG users want to hand-pick some migration routes. The problem in such a case is to select optimal migration routes for the remaining node pairs without destroying the hand-picked routes. A simple technique used in RTG is to force the route selection algorithm to choose the hand-picked routes by giving it no other choice. This technique imbeds hand-picked routes into the network from which migration routes will be selected.[11] One might think that an even simpler approach would be to assign the most favorable weights to those links in the network that are contained in the hand-picked routes, and then to select routes from the resulting network. However, such an approach sometimes fails on certain optimality criteria since some other links in the network happened to carry the most favorable weights from the beginning. Figure 7 shows the route imbedding process. Part a shows a simple five-node network, and Part b shows the corresponding directed graph. Let us consider the destination node E and one hand-picked route

Figure 8 A simple network



from node A to node E, A-D-C-E. Part c shows the graph resulting from imbedding the hand-picked route and removing all of the outgoing edges from nodes A, D, C, and E except those edges contained in the hand-picked route. This way, when a route selection algorithm is applied on the graph of Part c, the algorithm will be forced to select only one route from A to E, which is the hand-picked one.

The route numbering process in RTG

Whenever one decides to define one or more explicit routes between subarea node pairs, one faces the problem of properly assigning route identification numbers (ERNs) to explicit routes. In order to fully understand the route numbering (or ERN assignment) problem, we consider the network illustrated in Figure 8. We assume two routes for each node pair as shown in Table 6. Here, to simplify the description, the transmission group (TG) numbers are omitted from the expression of each explicit route because all are TG1. Since there are only two routes defined from Host A to each of the NCP nodes B, C, D, and E, and since route numbering depends on the destination, two ERNs are sufficient to distinguish these routes as shown in the fourth column of Table 6.

Table 6 Explicit route number assignment

ERN toward A		Routes	ERN from A
Arbitrary Assignment	By Resolving Conflicts		
1	1	B—A	1
2*	2	B—C—A (p)	2
1	1	C—A	1
2*	3	C—B—A (p)	2
1	1	D—A	1
2*	2	D—C—A (p)	2
1	1	E—B—A (p)	1
2*	3	E—D—A (p)	2

*indicates numbering conflict.
(p) indicates prime routes going toward destination A.

Since there are two routes from each NCP node to Host A, at least two ERNs are required to distinguish these eight routes. Let us number those routes from the NCP nodes to Host A as one, two, one, two, and so forth, as shown in the first column of Table 6. This route numbering is the simplest scheme; however, it causes route numbering conflicts between the following pairs of routes:

B-C-A and C-B-A
C-B-A and D-C-A
D-C-A and E-D-A

Any route numbering scheme that considers only a subset of routes for the numbering conflict resolution fails.[19] As a matter of fact, when a number is assigned to a route going toward a destination, one must verify such assignment against all those routes whose ERNs have already been assigned and which have the same destination. The second column in Table 6 indicates the ERN assignment obtained from resolving the numbering conflicts by assigning the next higher number and by not changing any prior ERN assignments. This sequential route numbering algorithm does not yield in general an optimal ERN assignment, optimal in the sense of requiring the minimum number of ERNs. The problem of determining the minimum number of ERNs required for the definition of explicit routes is NP-complete.[9] Let us next investigate and compare two distinct ways of performing the route numbering—the graph coloring and routing tree decomposition approaches.

Graph coloring approach. In this approach,[9] the route numbering problem is converted into a graph coloring problem that can be solved by any graph coloring algorithm available in the literature.[20] To convert the route numbering problem into a graph coloring problem, one must first derive graphs that describe routes and their numbering conflicts. For each destination node, first determine "prime" routes. An explicit route contained in any other explicit route is not a prime route. Next construct a

graph (known as a "contention" graph) of the prime routes by making each prime route a node in the graph and by inserting an edge between a node pair whenever the routes represented by the pair cannot assume the same route number.[9] Only the prime routes in the contention graph are considered, for no other reason than to reduce computation time by reducing the size of the graph. Notice that there will be as many contention graphs as the number of destinations in the network.

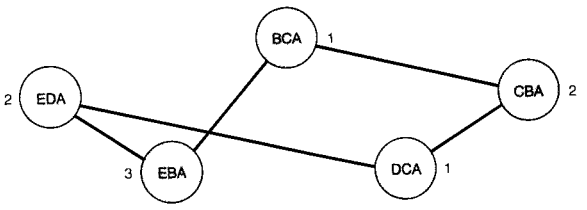
In Table 6, for example, route B-A is not a prime route since it is contained in routes C-B-A and E-B-A. The prime routes going toward destination A are indicated by (p) in the third column of the table. The contention graph for destination node A covering those prime explicit routes is illustrated in Figure 9. In this graph, for example, the node BCA and the node CBA have numbering conflicts because the route BCA and the route BA which is contained in CBA must be given different route numbers.

A routing tree is a rooted tree containing all routes to a particular destination node which is the root of the tree.

Once the contention graph is obtained for each destination node, the next step is to execute the graph coloring process. This process assigns a color (or ERN) to each node in the graph such that no adjacent node pair, the node pair with a direct edge between them, has the same color. The objective is to minimize the total number of different colors used.

Although the route numbering problem can be treated as a graph coloring problem, the real ERN assignment problem in SNA is not quite as simple as that since the total number of ERNs is limited to eight. There are cases where a given contention graph requires more than eight colors (a color here is the same as an ERN) and thus requires the decision of which nodes to color and which nodes not to color. This decision process is equivalent to the determination of which routes are to be supported and which routes are not to be supported by the routing tables. In general, the graph coloring approach for solving the ERN assignment problem does not offer a flexible route "design" process, though it offers a good analysis capability (the capability to determine the total number of ERNs needed to support all routes in the routing tables).

Figure 9 A contention graph—requires three colors (1, 2, 3)

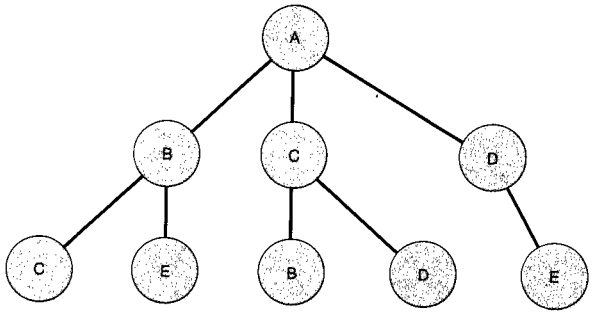


Routing tree decomposition approach. This approach[9] takes advantage of some intrinsic properties involved in routing rather than treating the route numbering problem as an abstract one like a graph coloring problem. For each destination node, a routing tree is constructed. A routing tree is a rooted tree containing all routes to a particular destination node, which is the root of the tree. In such a routing tree, any route from a leaf node to the root node represents a prime route. The routing tree for destination node A of those routes in Table 6 is illustrated in Figure 10.

The next step, which is equivalent to the route numbering process, is to decompose the routing tree into the minimum number of routing subtrees such that each subtree contains the root node and other nodes, each of which appears at most once in the subtree. Once the routing tree is decomposed into subtrees, a unique ERN can be assigned to each subtree. All routes contained in a subtree assume the ERN assigned to that subtree. Figure 11 shows a possible decomposition of the routing tree of Figure 10 into three routing subtrees. Since each subtree can be represented by a vector, which will be described next, it becomes extremely simple to implement the above decomposition scheme.

Consider a vector called a "routing bin," which is illustrated in Figure 12. Such a bin[9] corresponds to a collection of routing entries at different nodes to

Figure 10 A routing tree for destination A



one particular destination node. Here, N_1 through N_n denote the names (or addresses) of nodes in the network, and each entry in the bin will indicate the routing segment. The ith entry indicates that the route from node N_i visits the next node NN_i via the transmission group TG_i to eventually reach the destination node N. Each such routing bin is going to store one routing subtree when the routing tree decomposition is performed successfully. The tree decomposition process thus can be called a form of bin packing.[21] Let us next describe this process, which forms a base for the ERN assignment solution of the RTG.

Routing-bin packing process. This process has two steps:

1. For each destination node, provide one bin for each different ERN. Arrange them in the order of increasing numbers.
2. Proceeding with the prime routes having the same destination in some sequence, place each prime route in the smallest-numbered bin by not overwriting (i.e., avoiding numbering conflicts with) those routes that have been imbedded already.

The placement of a route in a routing bin is successful only when every route segment from its node of origin to its destination node can be placed in the same routing bin. The placement of a route segment into an entry of a routing bin is successful if the entry is vacant or if it is occupied by the same route segment that is being placed there.

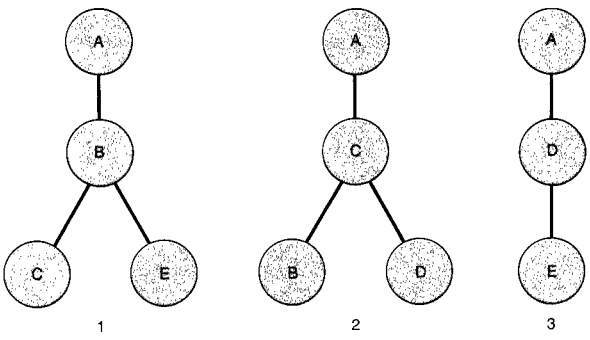
It is well known that the efficiency of bin packing is greatly influenced by the order in which elements are imbedded (or packed) into bins.[21,22] Among those routes that are of equal importance, the best packing can be achieved by imbedding them in lexicographically decreasing order. This makes sense from the route availability point of view since it imbeds the longer routes first. It is also well known that bin packing efficiency is greatly influenced by the order in which bins are tried for packing. In Step 2 the smallest-numbered bin is tried first for packing. This method is often called "first-fit" packing. Any other packing strategy such as "best-fit" packing may be used by defining a goodness of packing. A significant improvement in packing efficiency has been observed by first trying the bin into which the previous route was imbedded and then trying the first-fit packing method. Figure 13 shows the bin-packing results of those routes in Table 6 obtained by first-fit packing. By comparing the routing subtrees of Figure 11 and the routing bins of Figure 13, one can observe that a routing bin is an equivalent representation of a routing subtree.

The routing tree decomposition (or routing-bin packing) approach has several advantages over the graph coloring approach:

- The route numbering conflict is checked automatically at the time of route imbedding, thus avoiding the expensive preprocessing of generating contention graphs.
- The routing bin representation offers a very natural means for creating PATH macros.[23]
- The routing bin provides a simple means for determining which routes are to be dropped when not all routes can be numbered using available ERNs.

The ERN assignment algorithm used in RTG performs both analysis and synthesis. It uses two types of routing-bin packing algorithms: bidirectional and

Figure 11 Routing subtrees



unidirectional. The strategy of the bidirectional packing algorithm is to pack both the forward and backward explicit routes of a route at the same time (but not necessarily packing them into bins with the same ERN). The unidirectional packing algorithm packs explicit routes into bins without concern for the packing of their reverse routes. In RTG, the bidirectional packing algorithm is first used to determine roughly which routes can be defined into the routing table using the available ERNs. Once such routes are identified, better route numbering on those identified routes is obtained by applying the unidirectional packing algorithm. Finally, the bidirectional packing algorithm is once again applied on the remaining unpacked routes to increase the total number of routes to be defined in the routing tables.

Managing routes for an expanding network using RTG

In ACF Release 3 networks, when changes occur in the network topology (such as the insertion of new TGs and the addition of new communication nodes into the network), the generation of new PATH macros and the loading of them at each subarea node are required if the network administrator wants to provide the best communication paths for the new network. Unfortunately, the best use of the network for communication requires some changes in almost all routing tables in the network. This requirement implies a substantial amount of work for generating routing tables and some nontrivial network downtime. It is impossible to require the update of all routing tables simultaneously, especially when a large number of subareas are involved. When all routing tables cannot be updated simultaneously, one must update one (or some) routing table(s) at a time. This approach may not provide adequate communications while routing tables are updated unless the new routing tables support some explicit routes that were supported by the old routing tables.

There are several methods for dealing with the above problem.[8] Below, some interesting approaches of using RTG for generating routing tables for an expanding network are discussed.

Approach 1

1. Provide all ERNs and VRNs to RTG.
2. Run RTG for the original network and save the routes generated by RTG.

Figure 12 A routing bin

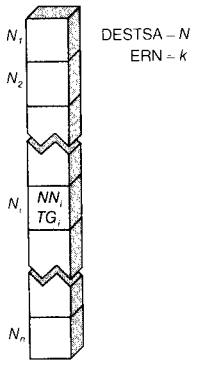
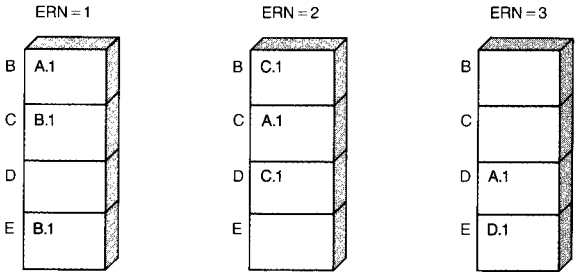


Figure 13 Routing-bin packing



3. When the network is changed, tell RTG to include all those routes selected from the original network in Step 2, run RTG for the new network, and save the newly generated routes.

The advantage of this approach is simplicity. The major disadvantage is that it is possible that all ERNs will have been used for the definition of routes in the original network; thus it may not be possible to define an adequate number of additional routes that reflect the topological changes in the new network. It is quite possible in some cases that no new additional route may be defined.

Approach 2

1. Reserve some ERNs and/or VRNs.
2. Run RTG for the original network and save the routes generated by RTG.
3. When the network is changed, tell RTG to include all those routes selected from the original network in Step 2, allocate some or all of the reserved ERNs and/or VRNs, run RTG for the new network, and save the generated routes.

The advantages of this approach are simplicity and the possibility of delaying the total system generation requirement until all of the reserved ERNs and VRNs are used up. This approach has a similar disadvantage to the first approach in that eventually all ERNs may be used up and no new routes may be defined. Since some ERNs are reserved for future system generation, a proper set of routes may not be supported by the limited ERNs in the early stage of the network.

Approach 3

1. Provide all ERNs and VRNs for RTG.
2. Run RTG for the original network and save the generated routes.
3. Reduce the number of routes for each node pair to two or three (e.g., save the first two or three routes for each node pair and discard the rest).
4. Tell RTG to include the reduced set of routes, run RTG for the new network, and save the generated routes.

The advantages of this approach are simplicity and the ability to provide both old and new routes. The disadvantage is that unless new subareas are added, the reduced set of routes tends to become "saturated" (not able to reflect other types of topological changes such as the addition of new TGs) unless some meaningful reduction method is used in Step 3.

Approach 4

1. Run RTG for the original network using only half of the ERNs (say, ERNs 0, 1, 2, and 3) and save the generated routes.
2. Run RTG for the new network using only the other half of the ERNs (say, ERNs 4, 5, 6, and 7) and save the generated routes.
3. Merge the two sets of routes.
4. Use the merged routes as part of the RTG input, run RTG for the new network, and generate PATH macros.
5. Update routing tables one node at a time.
6. After all routing tables are updated, change the VRN-to-ERN mapping table from VRN → (0, 1, 2, 3) to VRN → (4, 5, 6, 7).
7. Eventually, every VR uses an ER with ERN 4 or 5 or 6 or 7. Thus, ERN 0, 1, 2, and 3 become available again for new ER definition.
8. Repeat the process with reverse use of ERNs if the network topology changes.

In this method, if a network contains one or more pre-ACF/R3 subarea nodes, ERN0 should be reserved for the support of the migration routes and the remaining ERNs for the above method. The major advantages of this approach are that it is almost nondisruptive to the ongoing communications, it can cope with any topological changes, and the routing tables can be updated node by node. The major disadvantage is that only half of the available ERNs are used at any time, so that an adequate number of alternate explicit routes may not be defined.

Summary

The multiple explicit path routing of SNA that has been used since the announcement of the ACF Release 3 products provides many advantages over other routing techniques.[24] Unfortunately, however, the way this routing algorithm was implemented (i.e., the origin-independent, nonswap routing) and the way the routing tables are defined (i.e., creating routing tables by system generation rather than creating them dynamically) expose the complexity of managing the routing tables for SNA networks to the network administrators/designers. It is expected, however, that the future implementation of the SNA routing functions[5] will either reduce this complexity close to zero or hide it from the network administrators/designers.

In this paper we addressed three basic problems associated with the definition process of the SNA routing tables: the route selection problem, the route numbering problem, and the PATH macro generation problem. We then discussed a program called the Routing Table Generator (RTG) and how these three problems were solved with RTG. We also discussed some approaches on how to use RTG in managing routing tables for growing networks.

Although we did limit our discussion to those problems strictly associated with the definition of the transit routing tables, there are other networking problems which must be solved for the SNA networks. These problems are related to the definition of a session to class of service (COS) mapping tables and the definition of COS tables.[25] At this point, RTG does not provide explicit solutions to these problems, though it does provide some output information that can be used by the RTG users for the definition of these tables. RTG was announced as an extended Field Developed Program (FDP) in July 1981, and it

has been enhanced since then. It is expected that both functional and usability enhancements to RTG will continue as the SNA routing mechanism evolves.

Acknowledgments

The author has consulted with many people in IBM and has received many valuable comments and suggestions during the implementation of the Routing Table Generator program from the following individuals: V. Ahuja, J. H. Benjamin, G. M. Benson, L. D. Bower, L. Colle, P. DeBacker, S. L. Dilly, F. D. George, J. P. Gray, B. J. Heldke, G. Huff, J. Jackson, G. W. Krens, J. Link, B. Maney, T. B. McNeill, E. Miller, C. Pulley, D. A. Stamper, and R. A. Weingarten of the Communication Products Division; D. N. Crockett, R. H. Gleaton, J. J. Lucas, D. Shorter, and C. Van Winkle of the Information Systems Group; I. McGregor, A. Meijer, and P. Peeters of World Trade; and G. Markowsky and K. S. Natrajan of the Research Division. H. Colle and W. Kooij of World Trade, and K. Milliken and D. T. Tang of the Research Division contributed to the development of algorithms used in RTG. The author wants to express a special acknowledgment to R. M. Sackowitz of the Information Systems Group, who provided his constant support and many valuable suggestions and criticisms during the RTG development. Without his support it would have been difficult to bring RTG into existence.

Cited references and notes

1. J. P. Gray and T. B. McNeill, "SNA multiple-system networking," *IBM Systems Journal* **18,** No. 2, 263–297 (1979).
2. Some of the ACF Release 3 products are ACF/VTAM/V1R3, ACF/TCAM/V2R3, and ACF/NCP/V1R3.
3. The host intermediate network node (INN) function was not available in the ACF Release 2 and 3 products, though the function was once available in the ACF Release 1 host access methods.
4. Examples of the ACF Version 2 products are ACF/VTAM/V2R1 and V2R2 for MVS/VS1, ACF/VTAM/V2R1 for DOS/VSE, and ACF/NCP/V2R1 and V2R2.
5. J. M. Jaffe, F. H. Moss, and R. A. Weingarten, "SNA routing: Past, present and possible future," *IBM Systems Journal* **22,** No. 4, 417–434 (1983, this issue).
6. R. R. Juenemman and G. S. Kerr, "Explicit path routing in communications networks," *Proceedings of the 3rd ICCC*, Toronto (August 1976), pp. 340–342.
7. V. Ahuja, "Routing and flow control in Systems Network Architecture," *IBM Systems Journal* **18,** No. 2, 298–314 (1979).
8. *Routing Table Generator—Program Description/Operations Manual*, SB21-2806-1, IBM Corporation; available through IBM branch offices.
9. K. Maruyama and G. Markowsky, "On the generation of explicit routing tables," *Proceedings of the 5th ICCC*, Atlanta (October 1980), pp. 90–95.
10. K. S. Natrajan, D. T. Tang, and K. Maruyama, "On the selection of communication paths in computer networks," *Computer and Networking Symposium*, Gaithersburg, MD (December 1979), pp. 65–72.
11. D. T. Tang and K. Maruyama, *On Criteria and Generation of Optimal Paths*, Research Report RC 8411, IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598 (1980).
12. In ACF Release 3, a route is said to be a migration route if it traverses one or more pre-ACF Release 3 nodes. It is required that each migration route be numbered zero because the products do not support multiple ERs nor the virtual route concept.
13. What or which route is optimal is dependent on one's definition of optimality. If the optimality criterion is distance, for example, the shortest route is an optimal route.
14. R. Floyd, "Algorithm 97, shortest path," *Communications of the ACM* **5,** No. 6, 345 (June 1962).
15. E. W. Dijkstra, "A note on two problems in connection with graphs," *Numerische Mathematik* **1,** 269–271 (1959).
16. L. R. Ford and D. R. Fulkerson, *Flows in Networks*, Princeton University Press, Princeton, NJ (1962).
17. When an optimality criterion such as the minimum hop count or the maximum capacity is used, there tends to be more than one optimal route between many node pairs, especially when the network is highly connected.
18. M. A. Bonuccelli and K. Maruyama, *An Algorithm to Enforce Treeness in a Set of Optimal Paths,* Research Report RC 8998, IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598 (1981).
19. A pair of explicit routes to the same destination node is said to be in conflict with respect to SNA route numbering if the assignment of the same route number to these two routes leads to the assignment of the same route number to two physically different routes going to that destination node.
20. R. D. Dutton and R. C. Brigham, "A new graph coloring algorithm," *The Computer Journal* **24,** No. 1, 85–86 (1981).
21. A. R. Brown, *Optimum Packing and Depletion*, MacDonald, London, and American Elsevier, New York (1971).
22. K. Maruyama, S. K. Chang, and D. T. Tang, "A general packing algorithm for multidimensional resource requirements," *International Journal of Computer and Information Science* **6,** No. 2, 131–149 (1977).
23. The collection of the *i*th entries from the routing bins for the same destination node forms the PATH macro for the *i*th node.
24. TYMNET routing, ARPANET routing, and DECNET routing are some examples.
25. A COS table contains an ordered list of virtual routes that provide a certain level of service to sessions. A virtual route is defined by a pair of subarea nodes, a virtual route number, and a transmission priority.

Reprint Order No. G321-5204.

Kiyoshi Maruyama *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598.* Dr. Maruyama joined the Research Center as a Research Staff Member in 1972. His research areas have included combinatorial algorithms, parallel processing, data bases, communica-

tions network architecture, communications network design, and communications network management. He is currently manager of the communications and systems management group in the Computer Sciences Department at the Research Center. He has received two Outstanding Contribution Awards for his work related to communications networking. Dr. Maruyama received his B.S.E.E. degree from Nihon University, Tokyo, in 1968, and his Ph.D. in computer science from the University of Illinois, Champaign-Urbana, in 1972.