SNA route generation using traffic patterns

by S. C. Baade

This paper describes a procedure used by the IBM Information Network to generate optimum routes for a complex Systems Network Architecture (SNA) network by utilizing communication traffic patterns. The Route Table Generator and an understanding of customer locations and available facilities had been the basis for route generation. However, this approach became overwhelming as the network grew. The lack of flexibility required an increasing need to manually override generated routes. The resulting approach could not ensure that network delay had been minimized. The Network Design and Analysis (NETDA) tool developed at the IBM Yorktown Research Center was used as a solution. NETDA orders routes based on static indicators such as number of hops, route distance, and speed of the path components. However, NETDA also selects optimal routes based on network traffic patterns. Traffic data were easily incorporated into NETDA, and the IBM Information Network has optimized its SNA routing using NETDA and actual traffic data. The process was challenging because of the number of network components involved and the difficulty in obtaining portions of the traffic data. The use of NETDA for route generation is discussed, and the data collection methodology is described. Network component utilization and network delay changes are reviewed as a means of showing the benefits of such optimizations.

The task of generating routes with minimal delay in a network with thousands of routes is a complex problem for which few tools exist. The IBM Information Network is just such a large international Systems Network Architecture (SNA) network with more than 1 400 attached networks and over 400 000 attached terminals. When the Information Network began in 1981, the Route Table Generator (RTG) was the tool of choice for route generation. Certain obvious routing problems were corrected utilizing customer locations in conjunction with the RTG program. Although attempts were made to minimize route length, there was very little evidence showing the routes generated to be optimal with respect to delay. Concluding that the system was unacceptable for route generation, we reviewed available tools for their possible application to the route generation process.

Because of its ease of use, flexibility, and analysis capabilities, we began working with the Network Design and Analysis (NETDA) tool in 1983 to perform network modeling and route generation. NETDA[1,2] is an IBM licensed program product used for designing and updating the routing path definitions for Systems Network Architecture telecommunication networks. NETDA permits route selection based on a number of criteria such as route length, route capacity, and route availability. We initially used NETDA to generate routes based on such criteria. However, NETDA will optimize network routing if network traffic is provided. The Information Network began NETDA route optimization using traffic in 1989 as concerns grew over efficient use of network capacity and minimization of network delay. This paper discusses a study which used actual traffic data within the Information Network to optimize the SNA routing.

SNA overview

Before discussing the study presented in this paper, a brief review of a few Systems Network Architecture (SNA) concepts and terms is helpful. SNA is a hierarchical, layered structure which provides for communication between a broad range of IBM products. Hardware and software components implement the functions of the architectural layers. Hardware components include processors, communication controllers, cluster controllers, workstations, and printers. Software elements include telecommunication access methods, applications subsystems, and network control programs.

SNA defines a *node* as the portions of a hardware component, along with its associated software components, that implement the function of the architectural layers. SNA currently defines three types of nodes: host subarea nodes, communication controller subarea nodes, and peripheral nodes. A processor that contains a telecommunication access method is a *host subarea node*. An example of the host node is the node where the Advanced Communication Function/Virtual Telecommunication Access Method (ACF/VTAM™) is present. Host subarea nodes provide the SNA functions that control and manage a network. A communication controller (for example, an IBM 3745) that contains a network control program (for example, ACF/NCP) is a *communication controller subarea node*. Communication controller subarea nodes provide the SNA functions that route and control the flow of data in a network. All other nodes are *peripheral nodes*. A subarea consists of one subarea node and the peripheral nodes that are attached to that subarea node. Adjacent nodes are connected to one another by one or more *links*.

A user gains access to an SNA network through a *logical unit*. Logical units manage the exchange of data between end users, acting as an intermediary between the end user and the network.

The path control layer routes data between source and destination and controls data traffic in the network. *Routes* between source and destination points must be defined using routing tables.

The *SNA network interconnection* (SNI) capability allows users in different networks to communicate with each other. SNI provides a "gateway" between the different networks that permits each network to maintain the network address scheme most appropriate to its configuration.

Network routing: Yesterday and today

Until recently our network route generation process did not include the use of actual traffic data. Future requirements (locations, load) were estimated and a network model was developed which met the requirements while delivering specified availability and response time objectives. The topology determined was entered into NETDA, and routes were generated based on criteria such as minimal IBM 37X5 hops.[3] Actual traffic patterns were not used in route generation.

The network size (with more than 129 nodes) and the routing complexity (containing more than 30 000 routes) resulted in two major concerns: (1) whether network delay was minimized, and (2) whether we were making the most efficient use of the network components. The second concern received much attention as expenses were being minimized. We chose to address these concerns by using the optimization feature of NETDA along with actual traffic data.

Using NETDA to optimize routing appeared to be a simple process. However, collecting the necessary traffic data seemed quite a different story. Precisely what data were needed and over what time frame?

Traffic collection

Different types of traffic flow through the IBM Information Network backbone. The IBM Information Network's data collection techniques make it useful to decompose the data into two types: (1) backbone traffic between interconnected networks, and (2) backbone-only traffic that remains within the IBM Information Network backbone. The former is referred to as *cross network traffic* and the latter as *backbone traffic*.

Cross network traffic. The accounting method for cross network data required that certain session information be collected for all sessions passing through an IBM Information Network IBM 37X5 running the SNA network interconnection (SNI) via the network control program (NCP) exits. Although the data are primarily used for accounting,

they contain elements that are useful in determining traffic flows such as:

- Time stamp
- Origin/destination network addresses
- Byte counters
- Message counters

Since the data are used for accounting, they are retained for extended periods of time and are readily available for processing. Similar data could be collected by using the Gateway Session Collection feature of the Netview Performance Monitor (NPM).

Backbone traffic. Accounting techniques for the backbone traffic do not result in data with the same level of detail as the cross network traffic accounting. Therefore, methods had to be devised to estimate the backbone traffic in much more detail. What data were necessary? We needed message rates and message sizes between all session subarea pairs. There were two types of subareas in the backbone traffic flows: (1) application hosts, and (2) boundary IBM 37X5s with attached peripheral nodes. Traffic to the boundary IBM 37X5s was determined by analyzing line utilization data for all boundary IBM 37X5 lines to peripheral nodes via the NPM. This technique worked because data appearing on peripheral node lines attached to the boundary IBM 37X5s are either being sent by or being received by one of these peripheral nodes—i.e., the boundary IBM 37X5 is the subarea node sink or source for the data. In the case where the boundary IBM 37X5 also serves as a "gateway" (running the SNI program) to other networks, the cross network traffic, if any, would be captured via the method previously described under cross network traffic. Determining the backbone-only traffic contribution of a boundary IBM 37X5 becomes a problem of determining the data volume on all of its peripheral node lines.

As an illustration, suppose an IBM 37X5 node named NCP1 exists in our network. Further, suppose that node NCP1 has one 9600 bit per second line attached to it that supports a terminal cluster. Assume that the line is half-duplex (data are transmitted in only one direction at the time). Assume for the discussion that NPM reported utilization for the line at 50 percent for one hour. We can estimate the backbone-only traffic that flows into and out of the network due to node NCP1 for this hour using basic mathematics. The line can transmit 9600 bits per second or 1200 characters per second (8 bits per character). However, the line averaged 50 percent utilization for the hour, so the average transmission rate per second over the one-hour period was 600 characters per second (1200×0.50). Since the utilization was observed over a one-hour period, then we must multiply the average transmission rate (600 characters per second) times the number of seconds in one hour (3600) to determine the total data transmitted in the one-hour period. The line accounts for 2 160 000 (600×3600) bytes per hour. Since NCP1 has only this one line for peripheral nodes, the traffic into and out of the network contributed by node NCP1 would be 2 160 000 bytes per hour. Again, this is just an illustration, but the methodology was applied to actual data for all IBM Information Network boundary IBM 37X5s and their associated peripheral node lines to determine the total backbone-only traffic contribution due to the boundary IBM 37X5s.

Methods were devised to estimate the backbone traffic.

Because the actual traffic in the application hosts was unknown, the remaining piece to the backbone traffic puzzle was to estimate what percent of the traffic from each application host travels to each of the boundary node IBM 37X5s. A number of estimation techniques were considered using traffic distributions based on:

- Central processor utilization (CPU) in percent
- Sessions by host
- VTAM utilization by host in percent

Using percent CPU utilization by the host as recorded by tools such as the Resource Measurement Facility (RMF) proved a poor technique because we found many instances when the CPU utilization varied significantly but the traffic remained relatively constant. In one case a looping program resulted in high CPU utilization, but there was no change in network traffic. Similar analysis

was performed using accounting data based on percentage of CPU utilization, but the results were as poor as the direct use of CPU utilization. Apportioning traffic to the hosts based on the number of sessions to the respective host also proved a poor technique. The weakness in this approach appears to be that many users establish sessions with hosts and then allow their terminals to sit idle. There was not always a positive correlation between the number of sessions to the host and the traffic volume to and from the host. This problem was especially evident with a virtual machine (VM) host running the Professional Office System (PROFS®). A large number of users logged on to the system but were only making periodic use of the application. The only technique which proved useful was that of using the VTAM percentage CPU utilization for each host. This approach seemed to compensate for the previous shortcomings because VTAM generally consumes much of the CPU power when traffic is being transmitted between session partners.

Once the traffic entering or exiting the network from a specific IBM 37X5 was determined, the number of bytes was divided among the application processors based on a weighted average of the VTAM CPU utilization for each processor. For example, suppose that our network contained host nodes HOST1 and HOST2 with VTAM utilization of the CPU being 6 percent and 10 percent, respectively. Weighted values are calculated as 37.5 percent for HOST1 (6/(6 + 10)) and 62.5 percent for HOST2 (10/(6 + 10)). Again let us illustrate by using our sample node NCP1, which we estimated made a contribution of 2 160 000 bytes per hour. Using the weighted VTAM host values, the amount of traffic between NCP1 and the two hosts can be calculated as:

NCP1-HOST1: 810 000 (.375 × 2 160 000)
NCP1-HOST2: 1 350 000 (.625 × 2 160 000)

Allocating the traffic from all boundary IBM 37X5s to all application processors was performed in the same manner and a table of backbone traffic patterns was determined.

Distributing the traffic based on VTAM host utilization proved the most accurate among all of the techniques described. The accuracy of an approach was measured by comparing actual NPM utilizations of all backbone network links with the utilizations resulting from loading the traffic data into NETDA. This technique seemed meaningful because it permitted selection of the backbone traffic estimation method which minimized the difference between actual and modeled utilization. Data were compared on a component-by-component basis. Four statistics were computed for each approach. First, the absolute value of the difference between the actual and modeled utilization was averaged over all components. We wanted this value to be small (less than 10) since this was an overall measure of how accurately we distributed the traffic among the network components. Second, the sum of all of the differences (including the sign) was averaged over all components. Since we subtracted the actual utilization from the modeled utilization (in that order) the value was either positive or negative. A positive value indicated that the traffic had been overestimated and a negative value indicated that the traffic had been underestimated. We wanted the "average sum" to approach zero because this was a good measure of whether or not our traffic volume matched the actual traffic. If this number was zero it would indicate that we had the correct amount of total network traffic even if the distribution was not precise. Finally, in the last two statistics we summarized the range of the values by the maximum overestimate and maximum underestimate. The quality of the approximation is discussed further in the section on validation. The summary statistics for the three approaches to backbone traffic estimation are contained in Table 1.

Although the average sum of the differences in all cases of Table 1 indicate that our estimate of traffic volume was good, the average difference coupled with the range of the values led us to choose the VTAM utilization allocation method.

The problem of collecting backbone traffic can be simplified by using the Session Collection feature of the Netview Performance Monitor (NPM). Using this feature of NPM would allow collection of all data necessary to determine backbone traffic data flows.

Sampling network data. With a method to collect the data, we turned our attention to determining a time frame from which to sample. Since our capacity planning and performance had used the peak hour time frame, it seemed logical to use peak hour data for the optimization. It was important to decide the time frame in advance so we could be sure to sample "before and after" utili-

Table 1 Comparison of traffic estimation techniques

Method	Actual and Modeled Utilization			
	Average Difference (without sign)	Averaged Sum of Differences (using sign)	Maximum Overestimate	Maximum Underestimate
CPU utilization (percent)	17.1	+6.8	21.8	36.5
Host sessions	20.6	+9.1	18.2	43.7
VTAM utilization of CPU (percent)	6.4	+0.5	13.5	17.6

Table 2 NETDA node input

NODE_ID	SUBAREA(D)
CMC1	1
MVS1	2
MVS2	3
GWNCP1	4
GWNCP2	5
GWNCP3	6
GWNCP4	7
BNNCP1	8

zations from the same time frame. We chose to label the hour of the day with the most network traffic as the peak hour. We summarized all traffic bytes by hour for four weeks. A pattern of mid-afternoon traffic peaks became apparent. The difference between weeks was small. Further, Tuesday through Thursday seemed to have similar peaks that were higher than the other days. Based on our observations, we sampled data from the peak afternoon hour of Tuesday through Thursday for one week and then calculated an average of the three hours.

NETDA

NETDA is an interactive computer program that is used to design SNA telecommunication networks and then analyze and optimize performance for those networks. It may be used to design a new network or to evaluate existing and proposed networks. NETDA provides the tools to define the network, select routes, analyze performance, analyze availability, and generate full or partial path definition statements used by VTAM and NCP to implement routing in the network. In addition, NETDA allows the user to perform "what if" experiments before investing in an expensive upgrading of the network. For additional information on NETDA refer to the NETDA general information publication in Reference 1.

NETDA had been used in the IBM Information Network for five years to generate routes and the associated path definitions, as well as to aid in network design. However, NETDA had been used to perform the generation of routes based on the defined topology; it had not been fed traffic information that would have permitted it to optimize the routes based on load. With traffic data in hand we then entered the values into NETDA.

Feeding NETDA. Traffic data can be entered into NETDA by one of two means: (1) manually via a series of panels, or (2) in file(s) via the NETDA I/O processor.

At the time of optimization the IBM Information Network backbone consisted of 129 subarea nodes and 273 full duplex backbone links. We chose to use the I/O processor with our traffic file of over 3000 records since the volume of our traffic data would have made it very cumbersome to manually enter all of the data.

The NETDA traffic input file has a very specific format and is documented in the NETDA reference manual.[2] The file has three elements. First, one must define all of the nodes for which traffic is to be defined, such as in Table 2.

Second, one must define the message rates and related data for all of the applications. A sample is contained in Table 3. The word "application"

Table 3 NETDA application input

Percent Table Applications							
APPL_ID	M_OUT_HR	M_IN_HR	LENGTH_OUT	LENGTH_IN	PRIO	DIST_ID	COS_____
CMC1	10355	10355	1920	80	M	CMC1	IBMINTER
MVS1	12289	12289	730	120	M	MVS1	IBMINTER
MVS2	105	105	730	120	M	MVS2	IBMINTER
GWNCP1	41484	41484	580	90	M	GWNCP1	IBMINTER
GWNCP2	16781	16781	160	10	M	GWNCP2	IBMINTER
GWNCP3	448	448	1900	500	M	GWNCP3	IBMINTER
GWNCP4	11261	11261	4000	10	M	GWNCP4	IBMINTER

is used loosely, because one usually defines all IBM 37X5s running SNI as applications because their applications lie in attached networks.

Finally, distribution tables must be defined that relate the application message rates to their session end points. The distribution table indicates the percent of traffic between the application and other network nodes. A common distribution table can be used or we could define a unique distribution table for each application. An example of a distribution table for a node named MVS1 may be seen in Table 4.

Once the file was constructed with the three elements, it was quite simple to read the data into NETDA. Given a network topology definition and routes, the procedure to read the data into NETDA is:

- Delete any existing traffic data from NETDA
- Select the I/O processor menu
- Select the traffic input menu
- Read in the traffic data

It took approximately two minutes to execute these steps for our network. NETDA is very accommodating since it will read in multiple network traffic files and permit the user to augment existing traffic data with the new data, rather than automatically overwrite any previously read data. This can be very useful when one must look at changes in the network due to incremental traffic.

Table 4 NETDA distribution table input

Distribution for Node MVS1	
NODE_ID	PERCENT
CMC1	8.00
MVS2	7.00
GWNCP1	10.00
GWNCP2	40.00
GWNCP3	5.00
GWNCP4	25.00
BNNCP1	5.00

Validation

Prior to optimizing the routes, we wanted to ensure that network component utilizations observed in NETDA after loading traffic data closely approximated actual data in the network at the time of sampling. Our method of determining how accurately the true network traffic flows had been captured was to load the data and then compare network component utilizations in NETDA with those obtained from NPM. Because our accounting techniques capture all of the cross network traffic, the comparison is most indicative of the quality of the backbone traffic estimate. The results were promising. A sample of the table constructed for all components is in Table 5.

Table 5 Utilization validation

NETDA Percent Busy	Actual Percent Busy	Difference
26.1	12.6	+13.5*
44.2	34.2	+10.0
43.6	33.8	+9.8
68.4	62.1	+6.3
39.9	37.7	+2.2
33.5	32.8	+0.7
33.8	16.2	−17.6**
29.2	41.7	−12.5
15.2	23.9	−8.7
58.0	62.7	−4.7
15.0	18.2	−3.2
20.4	20.9	−0.5
.	.	.
.	.	.
.	.	.

Number of components:	402
Average difference (without sign):	6.4
Averaged sum of differences (using sign):	+0.5
Maximum component overestimate:	13.5*
Maximum component underestimate:	17.6**

Why did the data not agree exactly? At least four significant reasons are apparent. First, although the estimate of total backbone-only traffic was good, there were certainly errors in our estimate of the distribution because the exact distribution is not known. Hence, there may be a bit too much traffic in one location and a bit too little in another location. Second, overhead data are not included because the statistics used do not reflect such data. For example, we do not generate accounting records for the actual accounting records that flow in the network. Third, because the message lengths (I/O) were averaged into a single value for many applications within one host subarea, the sending and receiving utilizations of full duplex backbone links may have been skewed. Fourth, NETDA requires that the user enter the message processing speed of the communications controller being used. Because these data were not generally available, we had to rely on matching actual utilizations with message rates from our traffic data. Nonetheless, the data match quite well when one considers that utilizations are being compared for 402 components.

Optimization

Before discussing the optimization results, it is valuable to point out that NETDA optimization does not optimize the topology by adding or removing nodes or trunks. NETDA optimized the routes which were generated prior to optimization. Network design alternatives should be evaluated with NETDA prior to optimization. Design alternatives must also be analyzed after optimization because the design may be so poor as to cause bottlenecks.

Having established a base network NETDA model with traffic data closely resembling our network, it seemed logical to look at the results of permitting NETDA to optimize our routing based on the traffic. The process of optimization took approximately 1.5 minutes for our network of over 30 000 routes.

The results of the optimization were encouraging. Given the current topology (no new hardware) the network delays were reduced and a number of very busy resources had their utilizations significantly reduced. The average network path delay during the peak hour was reduced by 6 percent. The peak hour 90th percentile delay decreased by 42 percent. NETDA had projected an average savings of 4.3 percent and does not supply a 90th percentile value. The decrease was accomplished by redistributing the traffic onto different routes with less utilized components. The implication here is that network response time had been decreased without any new network expense. A most interesting view of the effects on path delay is to view a distribution of path delay decrease or increase after optimization as seen in Table 6.

Table 6 shows, for example, that 38 percent (20 + 18 percent) of the routes had a delay reduction between 50 milliseconds and 150 milliseconds. The table also shows that 87 percent of the routes had a delay reduction, whereas 13 percent of the routes had a delay increase. In general the routes with delay increases were routes with very little

traffic. Overall, path delay significantly decreased due to optimization.

Since the backbone links were full duplex, we decided to average the sending and receiving utilizations in order to compute a single statistic per link. This single statistic simplified before and after comparisons. Optimization enhanced both capacity planning and performance efforts by nearly cutting in half the number of links over 41 percent. The distribution of backbone link utilizations after optimization showed a migration toward lower utilizations which can be seen in Table 7.

The distribution of communication controller utilizations did not show a significant reduction in the higher levels but did show redistributions between the middle categories. One reason for a lack of reduction in the busier communication controllers is a design which requires traffic passing through certain intermediate routing communication controllers. The results are shown in Table 8.

Summary statistics for all paths, all communication controllers, and all links can cause us to lose sight of the significant response time improvement for specific network paths. In order to illustrate the significance of the savings, let us look more closely at how optimization affected a specific path: the path from one IBM 3725 to an application which the Information Network manages. The primary route between these two end points showed an improvement of 200 milliseconds after optimization. A message size of over 12 000 bytes is not unusual for this application. Since the application segments the messages into 1K chunks, 12 chunks (messages) would have to be transmitted. Although SNA would likely result in parallel transmission of the segments, the savings may be as high as 2.4 seconds (0.2×12) for the 12 000 byte transaction. Optimizing the routing reduced end-user response time to this application at no additional expense to the Information Network.

Concluding remarks

The benefits from optimizing the network routing using traffic data are significant. First and foremost, the network delay can be reduced. Furthermore, the reduction in delay requires no ad-

Table 6 Distribution of delay changes

Delay Change (msec)	Percent of Optimized Routes with:	
	Delay Decrease	Delay Increase
0–50	37	3
51–100	20	2
101–150	18	5
151–200	9	3
201+	3	0

Table 7 Distribution of link utilization changes

Link Utilization	Percent of Backbone Links	
	Before Optimization	After Optimization
0–10	52.6	56.4
11–20	20.8	22.5
21–30	12.9	10.7
31–40	6.9	6.9
41–50	4.8	2.8
51–60	1.0	0.3
60+	1.0	0.4

Table 8 Distribution of communication controller utilizations

Communication Controller Utilization	Percent of Communication Controllers	
	Before Optimization	After Optimization
0–10	12.6	12.6
11–20	25.2	26.9
21–30	16.0	13.4
31–40	9.2	13.4
41–50	11.8	9.2
51–60	9.2	8.4
61+	16.0	16.0

ditional network expense. Second, backbone resources are used in the most efficient manner because the traffic is balanced across the components in the best manner possible. Optimization gave us a bit of breathing room for resources which appeared to require additional capacity. Prior to optimization, we had three high-speed backbone links with peak hour utilizations of 67, 67, and 64 percent. These were of concern since we had a target of 65 percent peak hour to account for order lead time and containment of spikes during the peak hour. After optimization, their utilizations were reduced to 52, 57, and 58 percent, respectively. Before optimization, two communication controllers had utilizations of 59 and 58 percent. Following optimization, these utilizations were reduced to 37 and 40 percent. Third, once the network is optimized, one has the ability to scrutinize lightly loaded components and their usefulness. In this way expenses may actually be reduced as a result of changes in the network topology after network optimization. In essence, the traffic data can be used to determine a more efficient topology to support the traffic. Fourth, understanding the traffic patterns in the backbone network at the detail necessary for optimization has proved beneficial in problem determination. For example, suppose an IBM 37X5 node has very high utilization (90 percent) and traffic information shows that 50 percent of the message rate through the 37X5 is to a network that is attached to the same 37X5. Then we would conclude that one way of reducing 37X5 utilization is to consider moving the attached network connection to some other less utilized 37X5.

Currently SNA routes are static in the sense that it requires a reloading of tables in the communication controller and VTAM to alter the routes. There are currently many discussions regarding more dynamic or adaptive routing that would not require reloading of tables. Rather, such routing would determine the "best" route at the time a session was established, based on some set of criteria. Our route optimization could be viewed as one iteration in what would be a continuous process of route selection under a dynamic routing scheme. Our work would seem to suggest that some form of dynamic routing which uses feedback regarding traffic patterns and component utilizations could significantly improve network response time, as well as make best use of available network resources.

The weak link in our process of optimization is the estimate of the backbone-only traffic. Since the utilizations in the NETDA model depend on accurate traffic statistics, it would be best if this portion of the traffic did not have to be estimated but could be collected precisely as in the case of the cross network traffic. It has been suggested that NPM session data be used to gather precise statistics, and this possibility is being explored. It would be most desirable to use input from some type of standard product to gather precise information. Again, it would appear that NPM session collection may provide the answer. Furthermore, NPM gateway session collection could be used in place of the current method of cross network accounting. Therefore, NPM session collection would provide an integrated, standard, and supported solution to the problem of traffic data collection.

Cited references and note

1. *Network Design and Analysis General Information*, GC30-3495, IBM Corporation; available through IBM branch offices.
2. *Network Design and Analysis Reference*, SC30-3498, IBM Corporation; available through IBM branch offices.
3. IBM 37X5 represents a family of communications controllers, e.g., IBM 3725, IBM 3745.

Steven C. Baade *IBM Information Network, 3405 West Dr. Martin Luther King Jr. Boulevard, Tampa, Florida 33630.* Since joining IBM in 1983, Mr. Baade has worked in the IBM Information Network at Tampa, Florida. He has had various assignments in network design, planning, and performance. He is currently assigned to network modeling and optimization. Mr. Baade received a B.S. degree in mathematical sciences from Stetson University in 1981 and an M.S. degree in mathematical sciences from Clemson University in 1983.

Reprint Order No. G321-5433.