DOMESTIC LONG DISTANCE COMMUNICATIONS NETWORK STUDY

Communications Systems Engineering

SYN: A COMPUTER PROGRAM FOR LONG-RANGE NETWORK PLANNING

by

G.A. Neufeld
Systems Modelling & Analysis Group

August 1974

DOMESTIC LONG DISTANCE COMMUNICATIONS NETWORK STUDY

Communications Systems Engineering

SYN: A COMPUTER PROGRAM FOR LONG-RANGE NETWORK PLANNING

by

G.A. Neufeld
Systems Modelling & Analysis Group

CRC PROGRAM MANAGER: W.L. Hatton
PROJECT LEADER: A.R. Kaye

DLDCNS Report No.: 15
Date of Issue: August 1974

TABLE OF CONTENTS

1. INTRODUCTION

The purpose of this report is to describe a computer program called SYN which is an implementation of an algorithm described in DLDCNS Report No. 14. SYN is a tool for routing end-to-end message and television traffic, through a network model with step-like link costs, at minimum cost.

The purpose of this report is to describe a computer routine called SYN. SYN is an implementation of the minimum cost routing algorithm described in DLDCNS Report No. 14 (see Neufeld [2]). Since the implementation parallels the description of the algorithm (see Neufeld [2]), the object of this report is more concerned with how the algorithm was implemented and how to use it. Before going on, a general remark on the implementation should be made. The reader may from time to time question the reason for having implemented certain parts of the algorithm as they are in SYN. The only comment is that it reflects the "state of the art" in software engineering (for example, see [1]). There is little theoretical basis for development of software whose requirements are being defined in parallel with its development. SYN evolved over a period of time. During this time, new features and constraints were added, during the course of the long-distance communications study. As a result, the program was continually being modified.

The basic problem, for which SYN is designed, is described in Section 3 of DLDCNS Report No. 14 (see Neufeld [2]). SYN provides the user with a rather general tool for network planning and its application is, to some degree, open to the imagination of the user. SYN is written in FORTRAN IV and is presently set up so as to be called as a subroutine by a control program (see DLDCNS Report 14).

2. INPUT FILES FOR PROGRAM SYN

SYN was written in the form of a FORTRAN subroutine. The input to SYN consists of four files as well as input parameters via the subroutine call statement.

The following is a description of all the input files to program SYN. All files are read in under G Format. Thus data elements can be specified in any format and separated by a blank character(s).

The file, that contains all the link cost functions along with some other information, is attached to Unit 4. Basically the file contains two records for each link, followed by two records containing an end of file mark, followed by a record containing some further information. Thus a schematic of the file with n links is as follows:

Record 1	}data for link 1
Record 2	
Record 3	}data for link 2
Record 4	
Record 2NE-1	}data for link NE
Record 2NE	
Record NE+1	}end of file
Record NE+2	
Record NE+3	}other information

Let us describe the records (2I-1,2I) for a typical link I. (It is assumed that each link cost function will have at most four "steps", although this could readily be increased by adding more records in file for each link followed by expanding the corresponding dimension statement in the program and changing the format statement.) Suppose link k is incident to nodes n_1 and n_2 with $n_1 < n_2$ and its cost function is given in Figure 1. Clearly each step i is completely specified by the parameters (w_i, x_i, y_i, z_i), i=1,2,3, and (w_4, x_4) where

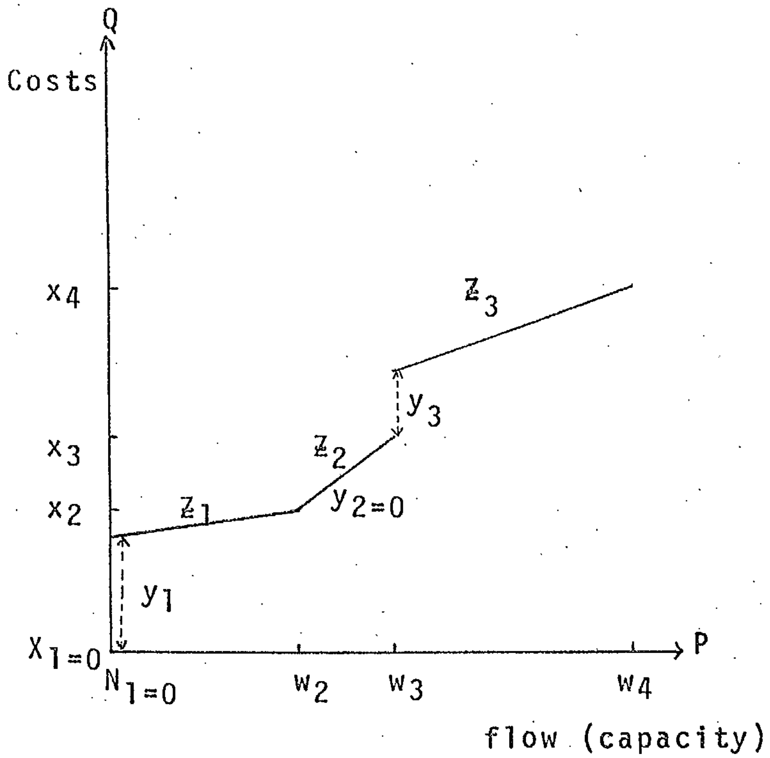

FIGURE 1

w_i is the P coordinate of the "bottom" of step i, x_i is the Q coordinate of the "bottom" of step i, y_i is the size of the "step" or discontinuity, z_i is the slope of the linear portion of step i, w_4 is the P coordinate of the maximum capacity of link k, and x_4 is the cost for w_4 flow units in link k. The number of parameters n_3 is dependent upon the number of steps, s, in the cost function: $n_3 = (sx4)+2$. (Note the function is actually over-specified this was done for purposes of convenience within the program SYN.) If a link cost function has 4 steps, then record 2I-1 and 2I are as follows:

Record 2I-1: n_1 n_2 n_3 w_1 x_1 y_1 z_1 w_2 x_2 y_2 z_2 w_3

Record 2I: x_3 y_3 z_3 w_4 x_4 e_1 e_2

Record 2I-1 contains the first 12 data elements and the remainder are on record 2I. The data element e_1 is equal to

-1 if link k is a TCTS analogue link,

-2 if link k is a CNCP link,
-3 if link k is a TCTS digital link
1 if link k is incident to the satellite node and is restricted to message traffic,

2 if link k is incident to the satellite node N and is restricted to television transmit/receive traffic (omni television traffic and broadcast television traffic being transmitted to the satellite node),

3 if link k is incident to the satellite node and is restricted to television traffic being received from the satellite, and

0 otherwise.

k data element e_2 is equal to n_2.

Record n+1 contains an end-of-file mark, namely "9999" and record n+2 is empty.

Before going on to describe the last record (n+3) in the file, let us state some assumptions regarding the ordering of the links:

i) As already noted, $n_1 < n_2$ for each link k.

ii) The (node) n_1 for link k (in record 2k-1) is less than or equal to the n_1 for link ℓ (in record $2\ell-1$) for all $k<\ell$.

iii) If (node) n_1 for link k (in record 2k-1) is equal to the n_1 for link ℓ (in record $2\ell-1$) for some $k<\ell$ then n_2 for link k (in record 2k-1) is less than the n_2 for link ℓ (in record $2\ell-1$).

If N is the number of nodes in the network model, then the format of record n+3 is as follows:

s_1 s_2 b_1 b_2 X.......... b_N where

s_1 is the cost per voice circuit for flow routed through the satellite space segment, s_2 is the number of voice circuits that are essentially equivalent to a television channel when routed through the satellite space segment, and b_i corresponds to node i in the network model and b_i=0 unless node i corresponds to a node within the satellite model, that is i is either the "satellite" node itself or i is incident to a link that is incident to the satellite node.

The data elements in this file correspond to the following variables in program SYNTHESIS.

For link I, corresponding to record 2I-1 and 2I,

ISPEC (I,1) = n_1,
ISPEC (I,2) = n_2,
ISPEC (I,3) = n_3,
SPEC (I,1),......, SPEC (I, n_3)

correspond to w_1, x_1,........ y_t, z_t, w_{t+1}, X_{t+1}

where $t = (n_3-2)/4$. Furthermore

EDGE(I,4) = e_1 and DTNCI) = e_2.

Corresponding to the very last record in the file, STCT = s_1, ITVVCEQ=s_2, and NODTYP(J)=b_j, J=1,.....,N where N is the number of nodes in the model.

The next file contains all the flow requirements and it is attached to unit 5 in the program. Basically the file contains one record for each message flow requirement, followed by a record containing an end of message flow requirements marker, followed by five records for each television flow requirement. Thus a schematic for a file with f_1 message flow requirements and f_2 television flow requirements is as follows:

Record 1	data for message flow requirement 1.
⋮	
Record f_1	data for message flow requirement f_1
Record f_1+1	end of message flow requirement marker
Record f_1+2 ⋮ Record f_1+6	data for television flow requirement 1.
⋮	⋮
Record $f_1+2+f_2 \cdot 5$ ⋮ Record $f_1+f_2 \cdot 5+6$	data for message flow requirement f_2.
Record $f_1+f_2 \cdot 5+7$	end-of-file marker.

For each message flow requirement i, the corresponding record i is as follows:

Record i: n_i n_i' t_i n_i''

where n_i and n_i' are the source-sink nodes for the i^{th} flow requirement. Data element t_i is number of voice circuits to be routed between n_i and n'_i. Data element n''_i is -1, -2, or -3 according to whether the traffic requirement must be routed on TCTS analogue, CNCP, or TCTS digital links where it is routed terrestrially. Record $f_1 + 1$ contains "9999" for purpose of indicating the previous record (f_1) as the last message flow requirement. The nodes in the network are assumed to be labelled such that if there are s nodes in the network that are either a source (or sink) for message flow, then $1 \le n_i, n'_i \le s$ for all message flow requirements i.

The data pertaining to each television flow requirement k is kept in groups of five records a, b, c, d, e. Consider first the case for a half-duplex television flow requirement k. Record "a" contains four data elements k_1, k_2, k_3, and k_4 as follows:

Record "a":

k_1 k_2 k_3 k_4.

Data element k_1 equals the number of half-duplex channels in the k^{th} television flow requirement. Data element k_2 equals 2 to indicate it as being half-duplex television. Data element k_3 represents the source node from which distribution takes place. Since every point receiving a half-duplex channel is also a distribution point, k_3 is set equal to 0. Data element k_4 is -1, -2, or -3 according to whether the traffic requirement must be routed on TCTS analogue, CNCP, or TCTS digital links where it is routed terrestrially. Records b, c, d and e are described making direct reference to Neufeld [2] using the same variables.

Records "b" and "c" are as follows:

Record "b": $n^{1}_{1k}\ n^{1}_{2k} \ldots\ 9999\ n^{2}_{1k}\ \ldots\ 9999\ n^{2}_{1k}\ \ldots\ 9999\ 9999$

Record "c": $m^{1}_{1k}\ m^{1}_{2k} \ldots\ 9999\ m^{2}_{1k}\ \ldots\ 9999\ m^{P}_{1k}\ \ldots\ 9999$

where

$$\{n^{i}_{1k},\ n^{i}_{2k}, \ldots\} = V_{A_i^k} \cap V_k \quad \text{and}$$

$$\{m^{i}_{1k},\ m^{i}_{2k}, \ldots\} = V_{A_i^k} \cap V.$$

Records "d" and "e" are as follows:

Record "d": $n^{1}_{1k}\ n^{1}_{2k}\ \ldots\ 9999\ n^{2}_{1k} \ldots\ 9999$
$n^{Q'}_{1k}\ n^{Q'}_{1k}\ \ldots\ 9999\ 9999$

Record "e": $u^{1}_{1k}\ \ldots\ 9998\ u^{1}_{1k}\ \ldots 9999\ \ldots$
$9999\ u^{Q'}_{1k}\ \ldots\ 9998\ u^{Q'}_{1k} \ldots\ 9999$

where

$$\{n^{i}_{1k},\ n^{i}_{1k}, \ldots\} = \{V_{C_i^k} \cap V\} - \{N\}.$$

$$\{u^{i}_{1k},\ u^{i}_{2k}, \ldots\} = V \cap V_{B_i^k}.$$

Some of the sets of nodes $\{m^{i}_{1k}, m^{i}_{2k}, \ldots\}$, $\{u^{i}_{1k}, u^{i}_{2k}, \ldots\}$, and $\{u^{\bar{1}}_{1k}, u^{\bar{i}}_{2k}, \ldots\}$ may be empty and this is designated with a space character ( ). If for some i, Steiner tree. $S_{C_i^k}(V_k \cap V_{C_i^k})$ is known to correspond to some Steiner tree $S_{A_j^k}(V_k \cap V_{A_j^k})$ then $\{u_{1k}^i, u_{2k}^i, \ldots\}$ may be replaced in the file record by {o j}. The result is an increase in efficiency in the program because the Steiner tree $S_{C_i^k}(V_k \cap V_{C_i^k})$ is not determined for a second time.

The specification for a simplex television flow requirement k_1 and k_4 are as for half-duplex television with a few exceptions. First, k_2 is equal to 3. Data element k_3 is the node from which the simplex television is distributed. Furthermore, the record "a" contains some additional data pertaining to the source or node from which the television channel is to be distributed. Thus record "a" is as follows:

$$k_1 \; k_2 \; k_3 \; k_4 \; \ell_1 \; \ldots \; \ell_L \; 9999$$

where k_1 is as before, k_2=3, k_3 is the source node, k_4 equals -1, -2, -3 as described above, and $\ell_1, \ldots \ell_L$ are the nodes that the user wishes to be considered as points from which to transmit the television channel to the satellite if it is used. It is assumed that all the nodes $\ell_1, \ldots, \ell_L$ are in $V_{B_1^k}$.

The data elements in this file containing the flow requirements correspond to the following variables in program SYN. For message flow requirement I, corresponding to the I^{th} record in the file, IRM(I,1) = n_I, IRM(I,2) = n_I^1, and RM(I,1) = t_I. For half-duplex television flow requirement K, corresponding to the k^{th} group of five records that follow the data pertaining to message flow requirements, TVREQ(k) = k_1, TYPE(k) = k_2, TVREQ1(k) = k_3, and TVCR(k) = k_4 in record "a"; SBGRPA(I), I=1,... corresponds to n^1_{1k}, n^1_{2k}, ... 9999 9999 in record "b"; SBGRPB(I),I=1,... correspond to m^1_{1k}, m^1_{2k}, ... 9999 in record "c"; SBGRPC(I),I=1,... correspond to r^1_{1k}, r^1_{2k}, ... 9999 9999 in record "d"; SBGRPD(I),I=1,... correspond to u^1_{1k}, u^2_{2k}, ... 9999 in record "e". For the case of a simplex television flow requirement k then TVREQ(K), TYPE(K), TVREQ1(K), SEND(k,1),... SEND(K,L), SEND(K,L+1) correspond to k_1, k_2, k_3, ℓ_1,...ℓ_L, 9999 in record "a".

There is input to program SYN that consists of various parameters which may be changed from run to run. They are passed to SYN via the subroutine call statement. There are five parameters, NET, SAT, ALP, BET, and ALP1. NET is really of no significance and is always set equal to -3. SAT equal 0 or 1 according to whether the satellite system cannot or can be used for routing traffic. ALP and BET corresponds to the alpha (α) and beta (β) parameters discussed in DLDCNS Report No. 14 [2]. ALP1 corresponds to an "alpha" value that applies only to the links corresponding to the satellite system.

The next input file is attached to unit 12. The file specifies an initial load or flow for each of the NE link in the model as well as the satellite. There is a one-to-one correspondence between the first NE records in this file and the records in the file read in on unit 4 (see above). A schematic of the file is as follows:

RECORD 1   Data for link 1

∘
∘
∘
∘
∘

RECORD NE   Data for link NE
RECORD NE+1 Data for satellite

Each record i has the following format:

29 alphanumeric characters ℓd_i.

The first twenty-nine characters may be blank or contain some identifier. They are followed by the amount of flow ℓd_i to be pre-loaded on link i, or the satellite when i=NE+1, before SYN begins to route traffic. When all the links and the satellite are being preloaded with zero traffic, then a file containing only the first record, with $\ell d_i = 10^6$, is sufficient.

The file read in from unit 13 specifies constraints in routing traffic through the network so as to achieve a more reliable network. Each constraint i specifies a set of links I = $\{\ell_i^1, \ell_i^2, \ldots\}$ of which a given number q_i must be loaded to some percent p_i of their capacity before any one of the links in I may be loaded to more than p_i percent of their capacity. Each record of the file specifies one constraint and the last record in the file contains a "9999" end-of-file marker. The format of each record i specifying the i^{th} constraint is

where n_1^{i1}, n_2^{i1}, and e_1^{i1} correspond to n_1, n_2, and e_1 for link ℓ_i^j in the file read in from unit 4 (see above).

3. OUTPUT FROM PROGRAM SYN

In addition to summary output to the printer (teletype), there are two output files from SYN. One output file pertains to the total flow on each link in the model, and the other pertains to information about the traffic routed through the satellite.

The total network cost and the amount of traffic routed through the satellite is output on unit 6. The load on each link in the network is output on a file attached to unit 9. There is a one-to-one correspondence between the records in this file and those in the file read in from unit 4.

Information pertaining to traffic routed through the satellite is output on unit 7. Every time SYN routes message traffic through the satellite, it is identified as to its source-sink nodes, its type (TCTS analogue, TCTS digital, CN/CP), and the ground stations through which it was routed. Each time, three new records are added to the file attached to unit 7. The format of these three records is as follows:

RECORD 1: a_1 a_2 a_3 a_4

RECORD 2: b_1

RECORD 3: c_1

where a_1 is the amount of message traffic routed through the satellite, a_3, a_4, are the source-sink nodes of the message traffic, a_2 is the type of message traffic (TCTS analogue -1, CN/CP -2, TCTS Digital -3), and b_1, c_1 are the nodes in the terrestrial network at which the traffic is routed through the ground stations. On the other hand, for each television channel of a television traffic requirement routed through the satellite, only one new record is added to the file attached to unit 7:

where a_1 is the number of channels of the b_1^{th} television traffic requirement routed, c_1 is the type of television channel (simplex 3, half-duplex 2), and d_1, d_2,.......are the nodes in the terrestrial network at which the television channel has been routed through a ground station.

4. STRUCTURE OF PROGRAM SYN

SYN is a subroutine and is itself made up of several subroutines. The purpose of this section is to provide the necessary information to understand the structure of SYN and the function of its various components.

Program SYN consists of a main subroutine along with twelve subroutines. Figure 2 shows how all the routines interrelate. A link between a pair of routines indicates one routine involking the use of another by means of a subroutine call statement. The links are directed to show the hierarchy; control is always passed from one routine to another in the direction indicated on each link. For example, the main subroutine SYN involkes the use of subroutine FPTH (not vice-versa). Control is always given back to the calling routine. The number of call statements by which each routine calls another is given by the number adjacent to each link in Figure 2. The areas shown in Figure 2 (indicated by broken lines) give a correspondence between the function of different groups of routines to the algorithm described in Section 4 of DLDCNS Report #14 (see Neufeld [2]). In the remainder of this section we describe the input, function, and output of each subroutine.

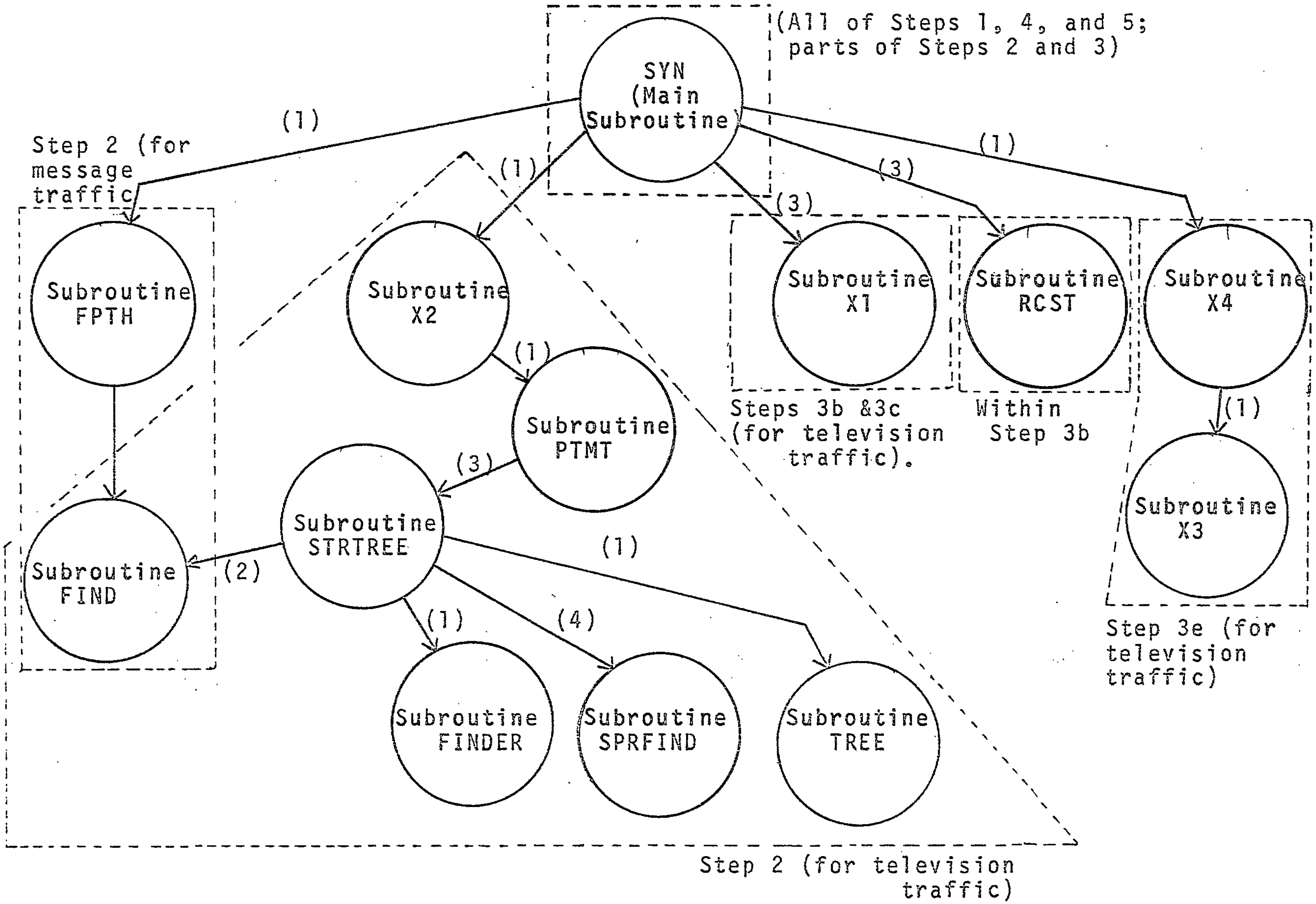

FIGURE 1 SCHEMATIC FLOW CHART OF SYN

4.1 SUBROUTINE FPTH

Input:

NE - is the number of links in the graph;

NREQNDS - is the number of nodes in the graph that are either a source or a sink for some flow requirement;

N - is the number of nodes in the graph.

EDGE (M,1),M=1,...,NE - is equal to the weight y_m assigned to link M (see Neufeld [2]).

NREQ - is the number of flow requirements corresponding to message traffic.

NODTYP(I),I=1,...,N - as defined above (MAIN PROGRAM - input).

SAT - as defined above (MAIN PROGRAM - input).

NET - as defined above (MAIN PROGRAM - input).

EDGE(J,4),J=1,...,NE -as defined above above (MAIN PROGRAM - input).

IRM(J ,1),IRM(J,2),IRM(J,3),J=1,...,NREQ - as defined above (MAIN PROGRAM - input).

RM(J,1),J=1,...,NREQ - as defined above (see MAIN PROGRAM - input) except that RM(J,1) is the amount of flow requirement J that remains to be permanently routed (see Neufeld [2] - corresponds to f_j in DLDCNS Report #14).

TRBL - is equal to 1.

Function:

i) To find the shortest path in the graph between all pairs of nodes IRM (J,1) and IRM(J,2) for all J=1,...,NREQ.

ii) To record the length of each shortest path.

iii) To determine the total flow on each link in the graph when the flow requirements J, each with RM(J,1) units of unrouted flow, J=1,...,NREQ are routed along the shortest paths determined in i.

Output:

IPTHLG(J),J=1,...,NREQ - is the number of links in the shortest path between nodes IRM(J,1) and IRM(J,2).

IEDGPTH(J,K),K=1,...,IPTHLG(J) - are the links in the shortest path between nodes IRM(J,1) and IRM(J,2). The links are stored in consecutive with IEDGTH(J,1) being the link incident to node IRM(J,2) and IEDGPTH(J,IPTHLG(J)) being the link incident to node IRM(J,1).

RM(J,2),J=1,...,NREQ, - is equal to 0 if RM(J,1) = 0 and is equal to 1 if RM(J,1)>0.

EDGE(M,2),M=1,...,NE - is the total flow on link M when the message flow requirements J, each with RM(J,1) units of unrouted flow, J=1,...,NREQ are routed along the shortest paths.

TRBL - is equal 0 if all the required shortest paths exist and is equal to 1 otherwise.

4.2 SUBROUTINE FIND

Input:

N1,N2 - are two nodes in the graph with N1<N2.

NODVTR(N1,1) - is the first row in the array ISPEC such that ISPEC(NODVTR)N1,1),1)=N1.

NODVTR(N1,2) is the number of rows in the array ISPEC with ISPEC(NODVTR(N1,1),1)=N1.

EDGE(J,4),J - NODVTR(N1,1), NODVTR(N1,1) + NODVTR(N1,2) as defined above.

TYP - is equal to 1, 2, or 3, depending upon the calling routine. TYP is related to the type of traffic, namely 1 for message traffic, 2 for omni television, and 3 for broadcast television.

Function:

To determine the existence of a link that is incident to nodes N1 and N2 and that may be used for the type of traffic specified in the variable TYP.

Output:

FOUND - is equal to 1 if the required link exists in the graph and is equal to 0 otherwise.

IPOS1 - is equal to the row of the array ISPEC that corresponds to the required link, if it exists.

4.3 SUBROUTINE X2

Input:

NE - as defined above.

NOTVREQ - is equal to the number of flow requirements corresponding to television traffic.

TVREQ(I),I=1,...,NOTVREQ - is equal to the number of channels remaining to be routed for the I^{th} flow requirement corresponding to television traffic.

TYPE(I),I=1,...,NOTVREQ - is the type of flow requirement I corresponding to television traffic. TYP(I) = 2 if the flow requirement corresponds to omni television and = 3 for broadcast television.

NODTYP(I),I=1,...,N (defined).

ITVVCEQ - is the number of voice circuits per television channel in any of the links associated with the satellite system model (see Neufeld [2]).

ITVTCTS - is the number of voice circuits per television channel in any of the links I associated with the TCTS system (i.e. EDGE(I,4)= -1).

ITVCNCP - is the number of voice circuits per television channel in any of the links I associated with the CNCP system (i.e. EDGE(I,4)= -2).

TVCR(I),I=1 - is the type of terrestrial link that may be used where the I^{th} television requirement is routed terrestrially.

Function:

i. To determine the minimum cost tree through which to distribute each flow requirement I,I=1,...,NOTVREQ.

ii. To determine the total flow on each link in the graph when the flow requirements I, each with TVREQ(I) channels of unrouted flow, I=1,...,NOTVREQ, are routed through the minimum cost trees determined in i.

Output:

TRBL i is equal to 1 if not all the required minimum cost trees exist and is equal to 0 otherwise.

TMPEDG(J),J=1,...,NE - is the total flow on link J when the televisio traffic flow requirements K, each with TVREQ(K) channels of unrouted flow, K=1,...,NOTVREQ, are routed through the minimum cost trees.

TVRQPTR(I),I=1,...,NOTVREQ - equals 1 if TVREQ(I)>0 and equals 0 otherwise.

4.4 SUBROUTINE PTMT

Input:

SBGROA(K,J), SBGRPB(K,J), SBGRPC(K,J), and SBGRPD(K,J),J=1,...,K=1,...,NOTVREQ. These arrays contain all the information pertaining to nodes in the various subgraphs A_i^k, B_i^k, C_i^k discussed earlier in this report under input to Main Program - SYNTHESIS.

I1 - corresponds to the I1th television traffic flow requirement being considered, $1 \leq$ I1 $\leq$ NOTVREQ.

SEND(I1,J),J=1,...., - as defined above (MAIN PROGRAM-input).

Function:

To determine the minimum cost Steiner tree through which to distribute the I1th television flow requirement.

Output:

GRP4 _ is a constant that is $\leq$ 1.0E30 unless there does not exist a tree through which to route the I1th television flow requirement.

STREDG(I1,I),I=1,....,NE - is equal to 1 if link I is part of the minimum cost Steiner tree for the I1th television flow requirement and is equal to 0 otherwise.

SNDND - equals 0 unless the flow required corresponds to broadcast television in which case SNDND equals one of the nodes stored in the I1th row of the array SEND (input to Main Program).

4.5 SUBROUTINE STRTREE

Input:

IGRP1(1) - equals the number of nodes that must be in the Steiner tree.

IGRP1(2) - equals the number of nodes that ma be in the minimum cost Steiner tree that interconnects the nodes IGRP1(I),I-3,...,IGRP1(1)+2.

IGRP1(I),I=3,...,IGRRP1(1)+2 - are all the terrestrial nodes, arranged in increasing order, that must be in the Steiner tree that is to be determined.

IGRP2(I),I=1,...,IGRP1(2) - are the nodes that may be in the minimum cost Steiner tree.

BUFF - equals 1 if node N is to be in the Steiner tree that is to be determined in subroutine STRTREE and is equal to 0 otherwise.

SNDND - defined above.

I1 - input from subroutine PTMT.

TYPE(I1) - as defined above (MAIN PROGRAM-input).

NODVTR(I,J),I=1,...,N,J=1,2 - as defined above (Subroutine FIND-input).

NODTYP(I),I=1,...,N - as defined above (MAIN PROGRAM-input).

EDGE(I,4),I=1,...,NE - as defined above (MAIN PROGRAM-input).

Function:

To determine the minimum cost Steiner tree in the network model that interconnects the nodes stored in IGRP1(I),I=3,...,IGRP1(1)+2, as well as the node N if BUFF equal 1, and that uses any of the nodes stored in IGR2(I),I=1,...,IGRP1(2), as intermediate nodes if there is a resulting reduction in cost.

Output:

STREDG(I1,I),I=1,...,NE - equals 1 if link I is in the minimum cost Steiner tree and is equal 0 otherwise.

GRP4 - equals the cost of the minimum cost Steiner tree.

4.6 SUBROUTINE TREE

Input:

IPOS2 - equals the number of links in the subgraph for which a minimum cost (weight) spanning tree is to be found.

TABLE(I,1) and TABLE (I,2),I=1,...,IPOS2, - are the nodes to which the I^{th} link in the subgraph is incident (Note: "I^{th} link" here does not refer in any way to the I^{th} link in the network model (i.e. stored in the I^{th} row of the arrays ISPEC and SPEC.

TABLE(I,3) - is the weight assigned to the I^{th} link in the subgraph for which a minimum cost spanning tree is to be found.

Function:

To determine the minimum cost spanning tree in the subgraph defined by the IPOS2 links stored in the array TABLE.

Output:

COST - is the cost of the minimum cost (weight) spanning tree.

TBL(I,3),I=1,...,IPOS2 - equals 1 if the link between nodes TABLE(I,1) and TABLE(I,2) is in the minimum spanning tree and is equal to 0 or 2 otherwise. Note that the "link" between nodes TABLE(I,1) and TABLE(I,2) may not actually be in the model but represent two links in the model that are in series.

4.7 SUBROUTINE SPRFIND

Input:

N1,N2 - two nodes in the network model with N1<N2.

IPOS2 - is a pointer to the last (row) entry made in the array TABLE.

NODVTR(N1,I),I=1,2 - as defined above (subroutine FIND-input).

DTN(I),I=NODVTR(N1,1),...,NODVTR(N1,1)-1 + NODVTR(N1,2) - as defined above (MAIN PROGRAM-input).

NET - as defined above (MAIN PROGRAM-input).

EDGE(IX,1), EDGE(IX,4),IX = NODVTR(N1,1), NODVTR(N1,1)-1 + NODVTR(N1,2) - as defined above (MAIN PROGRAM-input).

ITVCNCP, ITVTCTS - as defined above (Subroutine X2-input.

Object:

To find the minimum cost link I, if it exists, that is incident to Node N1 and such that DTN(I)=N2.

Output:

FOUND - equal 1 if such a link exists in the network model and equals 0 otherwise.

IPOS1 - equals I where link I is the minimum cost link that is incident to N1 and for which DTN(I)=N2. No value is assigned to IPOS1 when FOUND=0.

TABLE(IPOS2+1,4) - is not specified when FOUND=0. TABLE(IPOS2+1,4) is an indicator or pointer to enable identification of the link just found. TABLE(IPOS2+1,4) is the i^{th} link, in a sequential search beginning at the first row in arrays ISPEC and DTN, for which ISPEC(i,1) = N1, DTN(i) = N2, and that corresponds to the link just found, namely has the minimum cost of all such links.

ZMIN - is the cost of the link identified b N1,N2, and TABLE(IPOS2+1,4).

METCOST - is always equal to 0 (May have some future purpose).

4.8 SUBROUTINE FINDER

Input:

N1,N2 - are terrestrial nodes in the network model with N1<N2.

I - is an integer, $1 \le I \le NE$.

NODVTR(N1,I),I=1,2 - as defined above (Subroutine FIND-input).

EDGE(IX,1),EDGE(IX,4),DTN(IX),IX = NODVTR(N1,1),NODVTR(N1,1)-1 + NODVTR(N1,2) - as defined above.

TABLE(I,4) - as defined above.

Function:

To identify the link I in the model that is incident to node N1, for which DTN(i)=N2, and such that it is the i^{th} such link, in a sequential search beginning at the first row of arrays ISPEC and DTN, where i = TABLE(I,4). Furthermore, subroutine FINDER identifies the link between nodes N2 and DTN(I) when N2 ≠ DTN(I).

Output:

STREDG(I1,I) - is set equal to 1 for those values of I that corresponds to the link(s) identified in subroutine FINDER.

4.9 SUBROUTINE X1

Input:

NOTVREQ - defined above.

TVRQPTR(I),I=1,....,NOTVREQ - equals 1 if TVREQ(I)>0 unless TVRQPTR(I) has been set equal to 0 during a previous call to subroutine X1 after subroutine X2 was last called.

STREDG(I,J),I=1,...,NOTVREQ,J=1,...,NE - defined above.

EDGE(I,1),I=1,...,NE - is the weight y_I assigned to link I (see [2]).

NODTYP(I),I=1,...,N - defined above.

EDGE(I,1),I=1,...,NE - is as defined above except that it equals 1.0×10^{29} if for link I it was found that the expected cost of the temporary flow was not within a specified range of the actual cost (see [2]).

ITVCNCP - equals the number of voice circuits that are equivalent to one television channel (radio channel) on the CNCP terrestrial systems.

ITVTCTS - as for ITVCNCP except that it applies to TCTS terrestrial systems.

Function:

To determine those television flow requirements I, $1 \leq I \leq$ NOTVREQ and for which TVRQPTR(I)=1, that have been temporarily routed through a link J for which EDGE(J,1) = 1.0×10^{29}, that is through a link for which the expected cost is not within a specified range of the actual cost.

Output:

TMPEDG(I),I=1,...,NE - is as defined above (Subroutine X2-Output) except that TMPEDG(I) is updated to reflect the removal of those television flow requirements that were previously temporarily routed by subroutine X2.

TVRQPTR(I),I=1,...,NOTVREQ - is the same as input except TVRQPTR(I) is set equal to 0 if television flow requirement J was found to be routed through a link K with EDGE(K) = 1.0×10^{29}.

4.10 SUBROUTINE RCST

Input:

I - an integer such that $1 \leq I \leq$ NE.

EDGE(I,2) - is the current total amount of flow temporarily routed through link I.

ISPEC(I,3) - defined above (MAIN PROGRAM-input).

EGFL(I,1) - is the current total amount of flow permanently routed through link I.

SPEC(I,J),J=1,...,ISPEC(I,3)*4+2 - as defined above (MAIN PROGRAM-input).

STREDGA(I) - is the real or actual cost of routing the EDGE(I,2) units of temporary traffic through link I (taking into account that EGFL(I,1) units have already been permanently routed through link I.

Function:

To determine to ratio of the expected cost, of routing EDGE(I,2) units (voice circuits) of flow through link I, to the real cost.

Output:

BUFF - is the described ratio.

4.11 SUBROUTINE X4

Input:

NOTVREQ - as defined above (Subroutine X2-input).

TVREQ(I),I=1,...,NOTVREQ - as defined above (Subroutine X2-input).

TVRQPTR(I),I=1,...,NOTVREQ - as defined above (Subroutine X1-input and output).

SLNNET - as defined above (MAIN PROGRAM-input).

ISPEC(J,1),ISPEC(J,2),J=1,...,NE - as defined above (MAIN PROGRAM-input).

EDGE(I,3),I=1,...,NE - is the maximum amount of flow to be routed through link I during the current iteration (see definition of variable U_I in [2]).

SATTRFK - is the current total amount of flow (in voice circuits) through the satellite node N.

TVSTTFK - is the current total amount of television flow (in voice circuits) through the satellite node N.

ITVVCEQ - as defined above (MAIN PROGRAM-input).

IREQCTR - is the current total number of of flow requirements that have been completely routed "permanently" that is routed as in the final solution.

Function:

To route as much as possible of each television flow requirement I, $1 \subset I \subset$ NOTVREQ, for which TVRQPTR(I)=1. The total amount of flow through any link J, $1 \subset J \subset$ NE, must be less than or equal to EDGE(J,3).

Output:

SATTRFK - possibly updated.

TVSTTFK - possibly update.

IREQPTR - possibly updated.

EDGE(I,3),I=1,....,NE, - may possibly be updated to reflect the television flow that may have been routed through link I during the current iteration.

4.12 SUBROUTINE X3

Input:

I is an integer, $1 \subset I \subset$ NOTVREQ.

Function:

To route as much as possible of the so far unrouted television flow requirement I. The flow is of course established in those links J, for which STREDG(I,J)=1, and the amount may be limited by EDGE(J,3).

Output:

FLG - is the number of channels of television flow requirement I that have just been routed and become part of the overall solution.

TMPEDG(I),I=1,....,NE - is the total amount of flow (in voice circuits) that was routed through link I.

REFERENCES

1. Dijkstra, E.W., Structured Programming, Academic Press, 1972.

2. Neufeld, G.A., "An Applied Graph Theoretic Approach to Network Synthesis For Long-Distance Communications," DLDCNS Report No. 14, April 1974.

APPENDIX A

The following is a source program listing of SYN. The comment statements in the source listing become most useful when taken together with the comments in the last section of the above report.

```
      SUBROUTINE SYN(NET,SAT,
     *          ALP,BET,ALP1)
C
*
* Program SYN routes flow requirements through a network (graph) at
* minimal cost. This program is an implementation of the algorithm
* described in CRC serial document #14. It will be assumed that the
* reader is familiar with this document. Most of the variables and
* arrays in the following Common Block are briefly described when
* they are first used.
*
      COMMON ISPEC(144,3),SPEC(144,20),EGFL(144,6),IEGFL(144,3),
     1IRM(285,3),RM(285,4),EDGE(144,4),VRTX(39,2),IEDGPTH(285,34),
     2IPTHLG(285),NE,NREQ,N,NREQNDS,RMLPTR(144),STORAGE(144,2),DTN(144)
     4,TVRQPTR(10),FLG,IGRP1( 25),IGRP2( 15),IGRP3(15),GRP4,TYP,
     5TABLE(85,4),TBL(85,3),NODVTR(41,2),IPOS1,IPOS2,I1,FOUND,CTR1,
     6CTR2,CTR3,N1,N2,STCT,COST,I2,NODTYP(39),TYPE(12),TVREQ(12),
     7TMPEDG(144),STREDG(10,144),FLWW,REQCTR,IPTR(10),X44,XCES(10)
     8,ISTACK(60,2),SEND(12,7),ISTKPTR,TVREQ1(12),STKVTR(80),K,ITVVCEQ
     9,METCOST,IRELVTR(12),SBGRPA(12,20),SBGRPB(12,20),SBGRPC(12,20),
     9SBGRPD(12,20),COSTA(10),COSTB(10),STREDGA(144),STREDGB(144),BUFF
     9,SNDND,TVSTTFK,I,ZZMIN,IPROTCT(23,30),IPROPTR(23)
     9,ITRFKTP(5),TRTP,TVCR(12)
      WRITE(6,101)NET,SAT,ALP,BET,ALP1
101   FORMAT(5G.5,3A4)
      AFLAG=1
      BFLAG=1
      CFLAG=1
      ITVTCTS=1500
      ITVCNCP=2100
      ITVDIG=1344
      OUTPUT=0
      ISUPER=1
C               READ IN THE FILE SPECIFYING  THE CROSS SECTIONS AND THE
C               PROTECTION TO BE APPLIED ACROSS THEM.
      DO 13020 J=1,9999
CIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
C                                                     INPUT:UNIT 13
CIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
      READ(13,8) (IPROTCT(J,I),I=1,30)
      IF(IPROTCT(J,1).EQ.9999) GO TO 13025
```

```
13020 CONTINUE
13025 NCS=J-1
 5000 ZMNCT=1.0E50
      STORAGE(1,1)=9999
      RCDR=0
      TVSTTFK=0
C0000000000000000000000000000000000000000000000000000000000000000
C                                                        OUTPUT:UNIT 6
C000000000000000000000000000000000000000000000000000000000000000
      ZMNCT=1.0E50
      ITRNCTR=0
      MAXITRN=1
C
C
C
C INPUT COST FCNS AND COMPUTE INITIAL MATRIX A
C
   50 REWIND  4,5
      DO 10 M=1,1000
CIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
C                                                             INPUT:UNIT 4
CIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
*
* Read in tne link cost functions from the file attached to unit 4.
* Each link is incident to two nodes stored in ISPEC(M,1) and
* ISPEC(M,2).  ISPEC(M,3) specifies the number of data elements but
* is then reset equal to the number of steps in the cost function.
* There are two additional data elements read in with each cost
* function:  EDGE(M,4) equals -1 if the link belongs to the TCTS
* (TRANS CANADA) network, -2 if it belongs to CNCP, 1 if it belongs
* to the satellite system and is restricted to message traffic,
* 2 if it is restricted to television transmit/receive traffic,
* and 3 if it is restricted to receiving broadcast television traffic.
* Otherwise EDGE(M,4) = 0.  DTN(M) equals the real end node, rather
* than ISPEC(M,2) which may be an intermediate dummy node (see
* DLDCNS Report #8, Figure 3).  SPEC(M,I), I=1,...contains the
* data specifying the Mth link cost function.
*
      READ(4,30) (ISPEC(M,J),J=1,3),(SPEC(M,J),J=1,20)
      IF(ISPEC(M,1).EQ.9999) GO TO 9
      EDGE(M,4)=SPEC(M,ISPEC(M,3)+1)
      DTN(M)=SPEC(M,ISPEC(M,3)+2)
12001  ISPEC(M,3)=(ISPEC(M,3)-2)/4
   10 CONTINUE
*
* NE equals the number of links in the model.
*
    9 NE=M-1
      ZMIN=0
*
* Determine the number of nodes N in the model.
*
```

```
      DO 12000 M=1,NE
      IF(ISPEC(M,2).LE.ZMIN) GO TO 12000
      ZMIN=ISPEC(M,2)
12000 CONTINUE
      N=ZMIN
CIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
C                                                         INPUT:UNIT 4
CIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
*
* Read in some more data from Unit 4
*       STC =   offering price or cost for the satellite space
*               segment (see DLDCNS Report No. 8 - Section 8).
*
*       ITVVCEQ =   Number of voice circuits that equal one
*                   Television channel in the satellite space
*                   system.
*
*       NODTYP(I) = 1 if node I is in the satellite model and
*                   equals 0 otherwise.
*
      READ(4,8) STCT,ITVVCEQ, (NODTYP(I),I=1,N)
8     FORMAT(60G.0)
      WRITE(6,11002) STCT
11002 FORMAT('1 SATELLITE SPACE COST PER VOICE CIRCUIT IS   ',F15.2)
*
* Reset STCT = cost per voice circuit for television traffic through
* the satellite space system in terms of the cost through the
* TCTS links.
*
      STCT=(STCT*ITVVCEQ)/ITVTCTS
*
* Create a table for looking-up a link cost function.
*       NODVTR(I,1) = First row in the array SPEC for which
*                     ISPEC(M,1) = I.
*
*       NODVTR(I,2) = Number of rows in the array SPEC for which
*                     ISPEC(M,1) = I.
*
      NODVTR(1,1)=1
      DO 11000 I=1,N
      K=0
      DO  11005 J=NODVTR(I,1),NE
      IF(ISPEC(J,1).NE.I) GO TO 11010
11005 K=K+1
11010 NODVTR(I,2)=K
      NODVTR(I+1,1)=J
      IF(J.GT.NE) GO TO 11020
11000 CONTINUE
11020 CONTINUE
      DO 11007 J=I+1,N
11007 NODVTR(J,2)=0
C               INITIALIZE A VECTOR WHOSE ELEMENT I CORRESPONDS TO
```

```
C                   CROSS SECTION I.
      DO 13040 I=1,20
13040 IPROPTR(I)=0
C                   FOR EACH CROSS SECTION, PLACE THE TEMPORARY CAPACITY
C                   ON EACH OF ITS LINKS
      IF(NCS.EQ.0) GO TO 21
      DO 13045 I=1,NE
13045 IPTHLG(I)=0
      DO 13290 I=1,NCS
      DO 13100 J=1,9999
      IF(IPROTCT(I,J*3).EQ.9999)   GO TO 13290
      DO 13050 K=NODVTR(IPROTCT(I,J*3),1),NODVTR(IPROTCT(I,J*3),1)+
     1NODVTR(IPROTCT(I,J*3),2)-1
      IF(IPTHLG(K).EQ.1) GO TO 13050
      IF((ISPEC(K,2).EQ.IPROTCT(I,J*3+1))
     1.AND.(EDGE(K,4).EQ.IPROTCT(I,J*3+2))) GO TO 13060
13050 CONTINUE
13060 Z=SPEC(K,ISPEC(K,3)*4+1)*IPROTCT(I,2)/100
      IPTHLG(K)=1
      DO 13070 L=1,ISPEC(K,3)
      IF(SPEC(K,L*4+1).GT.Z) GO TO 13080
13070 CONTINUE
      GO TO 13100
13080 ISPEC(K,3)=L
      SPEC(K,L*4+1)=Z
      SPEC(K,L*4+2)=SPEC(K,(L-1)*4+2)+SPEC(K,(L-1)*4+3)+
     1(Z-SPEC(K,(L-1)*4+1))*SPEC(K,(L-1)*4+4)
13100 CONTINUE
13290 CONTINUE
   21 REWIND  4,5
      DO 16 M=1,1000
CIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
C                                                        INPUT:UNIT 5
CIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
*
* Read in the message traffic flow requirements from Unit 5.
* IRM(M,1) and IRM(M,2) are the source-sink nodes for the
* M th flow requirement and RM(M,1) is the amount of traffic.
*
      READ(5,8)  IRM(M,1),IRM(M,2),RM(M,1),IRM(M,3)
      IF(IRM(M,1).EQ.9999) GO TO 17
   16 CONTINUE
*
* Set NREQ = Number of message flow requirements.
*
   17 NREQ=M-1
      ITRFKTP(1)=IRM(1,3)
      ITRFKTP(5)=1
      DO 73000 I=2,NREQ
      DO 73001 J=1,ITRFKTP(5)
      IF(IRM(I,3).EQ.ITRFKTP(J)) GO TO 73000
73001 CONTINUE
```

```
      ITRFKTP(5)=ITRFKTP(5)+1
      ITRFKTP(ITRFKTP(5))=IRM(I,3)
73000 CONTINUE
      DO 11 M=1,1000
CIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
CIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
CIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
C                                                     INPUT:UNIT 5
CIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
CIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
CIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
*
* Read in the television flow requirements from Unit 5.
* For the M th television flow requirement,
*      TVREQ(M) =  Number of Television Channels
*      TYPE(M) =   2 for half-duplex television
*                  3 for simplex television
*
*      TVREQ1(M) = source node for simplex television.
*      SEND(M,I) = Ith potential node from which the simplex
*                  television flow requirement is to be
*                  considered being transmitted to the
*                  satellite (corresponds to nodes n
*                  (subscript L) - see DLDCNS Report No. 14,
*                  Section 5).
*
* Again referring to DLDCNS Report No. 14, data pertaining to the
* subgraphs A is read into arrays SBGRPA and SBGRPB where SBGRPA
* contains the nodes that must be in the Steiner tree and SBGRPB
* contains the Steiner nodes.  Similarly, data pertaining to the
* subgraphs B and C is read into arrays SBGRPC and SBGRPD.
* For further information see DLDCNS Report No. 14.
*
      READ(5,8,END=12) TVREQ(M),TYPE(M),TVREQ1(M),TVCR(M)
     1,(SEND(M,I),I=1,7)
      IF(TVREQ(M).EQ.9999)   GO TO 12
      READ(5,8) (SBGRPA(M,I),I=1,20)
      READ(5,8) (SBGRPB(M,I),I=1,20)
      READ(5,8) (SBGRPC(M,I),I=1,20)
      READ(5,8) (SBGRPD(M,I),I=1,20)
   11 CONTINUE
*
* Set NOTVREQ = Number of television flow requirements.
*
   12 NOTVREQ=M-1
      ZMIN=0
      DO 12010 M=1,NREQ
      IF(IRM(M,2).LE.ZMIN) GO TO 12010
      ZMIN=IRM(M,2)
12010 CONTINUE
      NREQNDS=ZMIN
   31 FORMAT(3G.0,NG.0,40G.0)
```

```
      REWIND  12
CIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
C                                                      INPUT:UNIT 12
CIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
*
* Read in the initial capacity to which the links are loaded before
* any traffic is routed.
*      EGFL(I,1) = current load (amount of flow) on link I.
*
      READ(12,12120) EGFL(1,1)
12120 FORMAT(29X,G.0)
      IF(EGFL(1,1).EQ.1.0E6) GO TO 12101
      REWIND  12
      DO 12100 I=1,NE
*
* Read in the parameters ALPHA and BETA (see DLDCNS Report No. 14,
* Section 4).
*      ALP =   ALPHA value that applies to all links except those
*              corresponding to the satellite ground station and
*              backhaul where ALP1 is the ALPHA value used.
*              BET equals the BETA value and it applies to all links.
*              ALP and BET are initial values for ALPHA and BETA.
*
CIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
C                                                      INPUT:UNIT 12
CIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
      READ(12,12120) EGFL(I,1)
      DO 12098 J=1,50
      IF(EGFL(I,1).LE.SPEC(I,1+J*4))     GO TO 12099
12098 CONTINUE
12099 IEGFL(I,1)=J
12100 CONTINUE
      GO TO 12105
12101 DO 12103 I=1,NE
      IEGFL(I,1)=1
12103 EGFL(I,1)=0
*
* SATTRFK equals total amount of flow through the satellite node
* which is always considered assumed to be the node labelled N
* where N = number of nodes in the network model.
*
12105 READ(12,8) SATTRFK
      DO 33 M=1,NE
*
* For each link M, determine the initial lower bound y (subscript M)
* (see DLDCNS Report No. 14, Section 4) and set EDGE(M,1) = Y
* (subscript M).
*
* Several other parameters (initial conditions) for each link M
* are specified.  They are as follows:
*
*       IEGFL(M,1) = Step I of the cost function into which the range
```

```
*                  of the current flow EGFL(M,1) falls.
*
*       IEGFL(M,2) = Step I of the cost function into which the range
*                  of the desired flow falls (desired flow = U
*                  (subscript M) - see DLDCNS Report No. 14,
*                  Section 4).
*
*       EGFL(M,2) = Previously mentioned desired flow.
*
*       EGFL(M,3) = is set to 1 prior to determining the required
*                  paths and trees.  Later when the expected cost
*                  of flow through link M is not realistic
*                  (comparison is made with the real cost and the ratio
*                  of the two costs is less than alpha - see DLDCNS
*                  Report No. 14, Section 4), then EGFL(M,3) is reset
*                  = 0.  EGFL(M,6) = 1 if link M is full and = 0
*                  otherwise.
*
   30 FORMAT(12G.0,/,11G.0)
      ZMIN=1.0E30
      IF(ISPEC(M,3).EQ.0) GO TO 23
      DO 20 K=1,ISPEC(M,3)
      IF(ISPEC(M,3).EQ.1) GO TO 12147
      IF(SPEC(M,K*4+1).LE.EGFL(M,1))     GO TO 20
12147 L=IEGFL(M,1)
      DEL=0
      IF(EGFL(M,1).EQ.SPEC(M,1+(L-1)*4)) GO TO 12149
      DEL=1
12149 Z1=(SPEC(M,2+4*K)-(SPEC(M,2+(L-1)*4)+SPEC(M,3+(L-1)*4))*DEL
     1-SPEC(M,4+(L-1)*4)*(EGFL(M,1)-SPEC(M,1+(L-1)*4)))
     1/(SPEC(M,1+4*K)-EGFL(M,1))
      IF(Z1.GT.ZMIN) GO TO 20
      ZMIN=Z1
      KO=K
   20 CONTINUE
   23 EDGE(M,1)=ZMIN
      ALPHA=1.0
      BETA=1.0
      IEGFL(M,2)=KO
      EGFL(M,2)=SPEC(M,1+KO*4)
      EGFL(M,3)=1
      EGFL(M,6)=0
      IEGFL(M,3)=0
   33 CONTINUE
*
* IREQCTR equals the total number of flow requirements that have been
* completely routed.
*
      IREQCTR=0
      IATEMP=0
*
* TRFKCTR equals the number of message traffic flow requirements
```

```
* that have been routed.
*
      TRFKCTR=0
      IF((NREQ+NOTVREQ).EQ.0) GO TO 800
C
C CALL FPTH TO FIND SHORTEST PATHS
C
C            FIRST KEEP IMAGE OF MATRIX A
C
      SLNNET=1
      TRBLCTR=1
      GO TO 45
*
* The following code pertains to improving a solution from a
* previously generated solution (see DLDCNS Report No. 14,
* Section 4). The capacity of some selected link is temporarily
* reduced in the hope that a solution of lower cost will be found.
* (Note: The present arrangement is such that this code pertaining
* to improving a solution is not used.)
*
 7000 IF(COST.GE.ZMNCT) GO TO 7020
      DO 7010 I=1,NE
      STORAGE(I,1)=ISPEC(I,3)
      STORAGE(I,2)=EGFL(I,1)
 7010 RMLPTR(I)=0
      ZMNCT=COST
      ITRNCTR=0
      PRCT=0
      ZMAXSTR=0
      IO=1
      GO TO 7030
 7020 WRITE(6,7047) TYPE(1),TYPE(2)
 7047 FORMAT(' PLEASE TAKE PRECAUTIONS-IMPOSSIBLE TO GET A SOLUTION',
     1/,'  NOTE TRAFFIC REQUIREMENT T=',F3.0,'  NO.=',F3.0,
     2/,'                  (T=0 IS MESSAGE,T=1 IS TV)')
      GO TO 800
 7037 DO 7035 I=1,NE
      ISPEC(I,3)=STORAGE(I,1)
 7035 EGFL(I,1)=STORAGE(I,2)
      PRCT=PRCT-.2
      ITRNCTR=ITRNCTR+1
      IF(ITRNCTR.GE.MAXITRN) GO TO 5000
 7030 IRMLPTR=0
      DO 7080 I=IO,5
      PRCT=PRCT+.2
      DO 7070 J=1,NE
      IF(RMLPTR(J).EQ.1)  GO TO 7070
      IF(EGFL(J,1).EQ.0) GO TO 7070
      IF(((EGFL(J,1)-SPEC(J,(IEGFL(J,1)-1)*4+1))/
     1(SPEC(J,IEGFL(J,1)*4+1)-SPEC(J,(IEGFL(J,1)-1)*4+1)))
     2.GT.PRCT) GO TO 7070
      IF((SPEC(J,(IEGFL(J,1)-1)*4+3)+(EGFL(J,1)-
```

```
     1SPEC(J,(IEGFL(J,1)-1)*4+1))*SPEC(J,(IEGFL(J,1)-1)*4+4))
     2.LE.ZMAXSTR) GO TO 7070
      IRMLPTR=J
      STRR=ISPEC(J,3)
      ZMAXSTR=SPEC(J,(IEGFL(J,1)-1)*4+3)+(EGFL(J,1)-
     1SPEC(J,(IEGFL(J,1)-1)*4+1))*SPEC(J,(IEGFL(J,1)-1)*4+4)
 7070 CONTINUE
      IF(IRMLPTR.NE.0) GO TO 7075
 7080 CONTINUE
      IF(IRMLPTR.EQ.0) GO TO 5000
 7075 ISPEC(IRMLPTR,3)=IEGFL(IRMLPTR,1)-1
      RMLPTR(IRMLPTR)=1
C0000000000000000000000000000000000000000000000000000000000000000000
C                                                      OUTPUT:UNIT 6
C000000000000000000000000000000000000000000000000000000000000000000
      WRITE(6,2050)
 2050 FORMAT(1H0,'ARE IMPROVEMENTS TO SOLUTION TO BE ATTEMPTED')
CIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
C                                                       INPUT:UNIT 6
CIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
      READ(6,8) I
      IF(I.EQ.0) GO TO 5000
      GO TO 21
C000000000000000000000000000000000000000000000000000000000000000000
C                                                          OUTPUT:UNIT 6
C0000000000000000000000000000000000000000000000000000000000000000000
 7085 WRITE(6,7086)
 7086 FORMAT(1H0,'THERE MAY NOT EXIST A SOLUTION')
      GO TO 800
*
* We are now ready to perform Step 2 of the algorithm described in
* Section 4 of DLDCNS Report No. 14.
*
   45 M=0
      FLWW=0
   46 FORMAT(1H0,'   SPECIFY ALPHA & BETA & ALPHA1')
*
* First we set up the ALPHA and BETA values to be used.
* TRBLCTR equals the number of passes through Steps 2 and 3
* without there having been any flow "permanently" routed.
* The initial value of TRBLCTR equals 0.
*
      ALPHA=ALP**TRBLCTR
      BETA=BET**TRBLCTR
C
C              NOW CALL FPTH
C
      TRBL=1
      IF(NREQ.EQ.0) GO TO 54
*
* Now find the required shortest paths in the network for routing
* message traffic.
```

```
*
      CALL FPTH
      IF(TRBL.EQ.1) GO TO 7020
      TRBL=1
      IF(OUTPUT.EQ.0) GO TO 54
      DO 70000 I=1,NREQ
      IF(RM(I,1).EQ.0) GO TO 70000
      WRITE(1,70001) I,(IEDGPTH(I,J),J=1,IPTHLG(I))
70001 FORMAT('1YYYY',80(I3,1X))
70000 CONTINUE
*
* Next find all the minimum cost trees for the television traffic.
*
   54 CALL X2
      IF(TRBL.EQ.1) GO TO 7020
      IF(TRBLCTR.LE.200) GO TO 49
C000000000000000000000000000000000000000000000000000000000000000000
C                                                      OUTPUT:UNIT 6
C000000000000000000000000000000000000000000000000000000000000000000
      WRITE(6,7850)
 7850 FORMAT(1H0,'THERE EXIST DIFFICULTIES IN FINDING A SOLUTION')
      GO TO 800
C
*
* The following code corresponds to Step 3 of the algorithm
* described in Section 4 of the DLDCNS Report No. 14.
*
* We begin by doing some bookkeeping, namely by setting
* EDGE(M,2) equal to the total "temporary" flow through each link M.
* Furthermore, some conditions are checked with respect to
* preventing nonconvergence.
*
C** PART A ****************************************
C FINDS LINKS WHICH ARE UNDERLOADED ON THEIR OWN ACCORD
C
C
C ELIMINATE ALL REQ. PAIRS THAT FLOW THRU AN UNDER
C LOADED LINK. ALSO ELIMINATE THE CORRESPONDING LINKS
C
C                    FIRST KEEP IMAGE OF PRESENT MATRIX A
C                    AS WELL AS RSTORE ORIGINAL MATRIX A
C
   49 M=0
      IF(AFLAG.EQ.0) GO TO 9999
C
C                FROM MATRIX FM,SET COST OF UNDERLOADED LINK TO
C                                     1.0E30
C
      IF(OUTPUT.EQ.0) GO TO 77778
      WRITE(1,77777) (EGFL(I,3),I=1,NE)
77778 CONTINUE
*
```

```
* Now compare the expected cost of flow in each link I to the
* actual cost (see Step 3b, Section 4, DLDCNS Report No. 14).
*
      DO 53 I=1,NE
      IF(EGFL(I,1).NE.0) GO TO 59000
      STREDGA(I)=0
      GO TO 59003
59000 DO 59001 J=1,ISPEC(I,3)
      IF(EGFL(I,1).LE.SPEC(I,J*4+1)) GO TO 59002
59001 CONTINUE
59002 STREDGA(I)=SPEC(I,(J-1)*4+2)+SPEC(I,(J-1)*4+3)
     1+SPEC(I,(J-1)*4+4)*(EGFL(I,1)-SPEC(I,(J-1)*4+1))
59003 EDGE(I,2)=EDGE(I,2)+TMPEDG(I)
      IF((EGFL(I,2)-EGFL(I,1)).LE.0) GO TO 50000
      IF(NODTYP(ISPEC(I,2)).GT.0) GO TO 53
      IF(TMPEDG(I).EQ.0) GO TO 53
      IF(TMPEDG(I).NE.EDGE(I,2)) GO TO 53
      TP=ITVTCTS
      IF(EDGE(I,4).EQ.-1) GO TO 59005
      TP=ITVCNCP
      IF(EDGE(I,4).EQ.-2) GO TO 59005
      TP=ITVDIG
59005 IF((SPEC(I,1+4*ISPEC(I,3))-EGFL(I,1)).GE.TP)   GO TO 53
50000 EGFL(I,6)=1
   53 CONTINUE
*
* Then find flow requirements passing through links for which the
* expected cost is too unrealistic (see Step 3c, Section 4,
* DLDCNS Report No. 14).
*
      DO 80 I=1,NE
      EGFL(I,4)=EDGE(I,2)
      SALPHA=ALPHA
      IF(NODTYP(ISPEC(I,2)).LE.0) GO TO 57
      SALPHA=ALP1
   57 CALL RCST
      IF(BUFF.GE.SALPHA) GO TO 80
      EDGE(I,1)=1.0E29
      EGFL(I,3)=0
      IEGFL(I,3)=1
   80 CONTINUE
C
C                    FIND REQ PAIRS PASSING THRU UNDERLOADED LINKS
C
      DO 8099 I=1,NREQ
      IF(RM(I,2).EQ.0) GO TO 8098
      DO 8090 J=1,IPTHLG(I)
      IF(EDGE(IEDGPTH(I,J),1).GE.1.0E29) GO TO 8098
 8090 CONTINUE
      GO TO 8099
 8098 RM(I,2)=0
 8099 CONTINUE
```

```
      CALL X1
      IF(OUTPUT.EQ.0) GO TO 9999
      WRITE(1,77777) (EGFL(I,3),I=1,NE)
77777 FORMAT(1H0,'XXXXX',4(30(I1,1X),/))
C
*
* This next section repeats steps 3b-3c.  It is a repeat of the
* code just executed.  The object is to those flow requirements
* passing through links for which the expected cost is too
* unrealistic as a result of the flow requirements that were chosen
* to be ignored.
*
C** PART B  ***********************************************
C  FIND LINKS WHICH ARE UNDER LOADED DUE TO PATHS GOING THRU
C  UNDERLOADED LINKS FOUND IN PART A
C
 9999 IF(BFLAG.EQ.0) GO TO 9998
      DO 8120 I=1,NE
 8120 EDGE(I,2)=0
      DO 8130 I=1,NREQ
      IF(RM(I,2).EQ.0) GO TO 8130
      DO 8125 J=1,IPTHLG(I)
      EDGE(IEDGPTH(I,J),2)=EDGE(IEDGPTH(I,J),2)+RM(I,1)
 8125 CONTINUE
 8130 CONTINUE
      DO 8132 I=1,NE
 8132 EDGE(I,2)=EDGE(I,2)+TMPEDG(I)
C
C           FIND UNDERLOADED LINKS
C
      DO 8000 I=1,NE
      SALPHA=ALPHA
      IF(NODTYP(ISPEC(I,2)).LE.0) GO TO 8159
      SALPHA=ALP1
 8159 CALL RCST
      IF(BUFF.GE.SALPHA) GO TO 8000
      EDGE(I,1)=1.0E29
      EGFL(I,3)=0
      IEGFL(I,3)=1
 8000 EDGE(I,3)=EGFL(I,2)-EGFL(I,1)
C
C              FIND REQ. PAIRS PASSING THRU UNDERLOADED LINKS
C
      DO 8199 I=1,NREQ
      IF(RM(I,2).EQ.0) GO TO 8198
      DO 8190 J=1,IPTHLG(I)
      IF(EDGE(IEDGPTH(I,J),1).GE.1.0E29) GO TO 8198
 8190 CONTINUE
      GO TO 8199
 8198 RM(I,2)=0
 8199 CONTINUE
      CALL X1
```

```
      IF(OUTPUT.EQ.0) GO TO 9998
      WRITE(1,77777) (EGFL(I,3),I=1,NE)
C
*
* Now repeat steps 3b and 3c but with the added restriction that the
* total flow b (subscript M) plus t (subscript M) does not exceed
* U (subscript M) - for more details see step 3d of the algorithm
* in Section 4, DLDCNS Report No. 14).
*
* First establish some initial conditions.  Then, for each flow
* requirement that is still eligible for further consideration
* during this pass through the algorithm, find the maximum
* allowable flow through the corresponding shortest path.
* Flow requirements corresponding to message traffic are considered
* first, followed by those corresponding to television traffic.
*
C** PART C *****************************************************
C  FIND LINKS WHICH ARE UNDERLOADED DUE TOPATHS GOING THRU A
C  LINK WITH RESTRICTED CAPACITY
C
C
C                           SET MATRIX FM=0
C
 9998 IF(CFLAG.EQ.0) GO TO 9997
      DO 90 I=1,NE
      EDGE(I,2)=0
   90 CONTINUE
C
C                          FOR EACH REQ. PAIR FIND MAX PATH
C                          (IGNORE REQ. WHOSE CORRESP. PATH PASSES
C                          THRU UNDERLOADED LINK)
C
      DO 150 I=1,NREQ
      IF(RM(I,2).EQ.0) GO TO 149
      ZMIN=1.0E30
      DO 140 J=1,IPTHLG(I)
      IF(EDGE(IEDGPTH(I,J),1).GE.1.0E29) GO TO 149
      IF(EDGE(IEDGPTH(I,J),3).GE.ZMIN) GO TO 140
      ZMIN=EDGE(IEDGPTH(I,J),3)
  140 CONTINUE
  142 IF(RM(I,1).GE.ZMIN) GO TO 141
      ZMIN=RM(I,1)
  141 DO 147 J=1,IPTHLG(I)
      EDGE(IEDGPTH(I,J),2)=EDGE(IEDGPTH(I,J),2)+ZMIN
  147 CONTINUE
      GO TO 150
  149 RM(I,2)=0
  150 CONTINUE
      IF(NOTVREQ.EQ.0) GO TO 6095
      DO 6085 I=1,NE
 6085 TMPEDG(I)=0
      X44=0
```

```
      DO 6090 I=1,NOTVREQ
      IF(TVREQ(I).EQ.0) GO TO 6090
      IF(TVRQPTR(I).EQ.0) GO TO 6090
      CALL X3
 6090 CONTINUE
 6095 CONTINUE
C
*
* Now flag all those links for which the expected cost is
* too unrealistic.
*
C                      AGAIN NOTE UNDERLOADED LINKS AND SET THEIR
C                      COST = 1.0E29
C
      DO 167 I=1,NE
  167 EDGE(I,2)=EDGE(I,2)+TMPEDG(I)
      DO 170 I=1,NE
      SALPHA=ALPHA
      IF(NODTYP(ISPEC(I,2)).LE.0) GO TO 168
      SALPHA=ALP1
  168 CALL RCST
      IF(BUFF.GE.SALPHA) GO TO 170
      EDGE(I,1)=1.0E29
      EGFL(I,3)=0
  170 CONTINUE
C
*
* Next find the flow requirements that were temporarily routed
* through links that were flagged above.
*
C                     FIND THE REQ. PAIRS WHOSE CORRESP. SHORTEST
C                     PATH CONTAINS AN UNDERLOADED LINK
C
      DO 173 I=1,NE
      EDGE(I,2)=0
  173 CONTINUE
      DO 200 I=1,NREQ
      IF(RM(I,2).EQ.0.0) GO TO 200
      DO 176 J=1,IPTHLG(I)
      IF(EDGE(IEDGPTH(I,J),1).GE.1.0E29) GO TO 180
  176 CONTINUE
      DO 178 J=1,IPTHLG(I)
  178 EDGE(IEDGPTH(I,J),2)=EDGE(IEDGPTH(I,J),2)+1
      GO TO 200
  180 RM(I,2)=0
  200 CONTINUE
      CALL X1
      IF(OUTPUT.EQ.0) GO TO 9997
      WRITE(1,77777) (EGFL(I,3),I=1,NE)
C
*
* Next we execute code corresponding to Step 3c of the algorithm
```

```
* in Section 4 of DLDCNS Report No. 14.
*
* First as much television traffic as is possible is routed.
*
C FOR EACH LINK TO BE CONSIDERED FIND THE REQ. PAIRS WHOSE
C SHORTEST PATH PASSES THRU THE LINK
C
C                             TO FACILITATE THE ABOVE SOME PRELIMINARIES
C                             MUST BE PERFORMED (FOR EASE OF COMPUTATION)
C
C                             ALSO FIND THE MIN AND MAX OF FM(I,J)
C                           (CORRESPONDING TO THE LINKS BEING CONSIDERED)
C
 9997 CALL X4
*
* For message traffic those flow requirements, whose shortest
* path contains a link with the fewest number of flow
* requirements temporarily routed through it, are considered
* first for being routed permanently.  So first find the
* "least used" link.
*
      MIN=10000000
      MAX=0
      DO 220 I=1,NE
      IF(EDGE(I,4).GE.2) GO TO 220
      IF(EGFL(I,3).EQ.0) GO TO 220
      IF(EDGE(I,2).GE.MIN) GO TO 206
      IF(EDGE(I,2).EQ.0) GO TO 206
      MIN=EDGE(I,2)
  206 IF(EDGE(I,2).LE.MAX) GO TO 220
      MAX=EDGE(I,2)
  220 CONTINUE
      IF(MIN.GT.MAX) GO TO 399
C
*
* Now find the flow requirements routed through the least used
* links and permanently route as much as possible of these
* flow requirements.
*
C                        FIND REQ. PAIRS WHOSE CORRESP. PATHS PASS
C                        THRU LINKS THAT ARE BEING USED MIN TIMES
C
  210 DO 300 I=1,NREQ
      BUFF=0
      IF((RM(I,1).EQ.0).OR.(RM(I,1).GT.0.02)) GO TO 211
      RM(I,1)=0
      IREQCTR=IREQCTR+1
      TRFKCTR=TRFKCTR+1
      GO TO 235
  211 IF(RM(I,2).EQ.0) GO TO 300
      ZMIN=1.0E29
      KK=0
```

```
      DO 225 J=1,IPTHLG(I)
      IF(EDGE(IEDGPTH(I,J),2).NE.MIN) GO TO 223
      KK=1
  223 IF(EDGE(IEDGPTH(I,J),3).GE.ZMIN) GO TO 225
      ZMIN=EDGE(IEDGPTH(I,J),3)
      IF(ZMIN.EQ.0) GO TO 235
  225 CONTINUE
      IF(KK.EQ.0) GO TO 300
      GO TO 240
  235 RM(I,2)=0
      GO TO 300
  240 RM(I,2)=0
      IF(RM(I,1).GT.ZMIN) GO TO 244
      ZMIN=RM(I,1)
      RM(I,1)=0
      IREQCTR=IREQCTR+1
      TRFKCTR=TRFKCTR+1
      FLWW=1
      GO TO 245
  244 IF(ZMIN.EQ.0) GO TO 300
      RM(I,1)=RM(I,1)-ZMIN
      FLWW=1
  245 IF(SLNNET.EQ.0) GO TO 247
      IF(SLNNET.EQ.2) GO TO 243
      DO 16500 J=1,IPTHLG(I)
      KJ=ISPEC(IEDGPTH(I,J),2)
      IF(NODTYP(KJ).EQ.1) GO TO 243
16500 CONTINUE
      GO TO 247
  243 WRITE(7,246)  ZMIN,IRM(I,3),IRM(I,1),IRM(I,2)
  246 FORMAT(4I)
  247 DO 248 J=1,IPTHLG(I)
      EDGE(IEDGPTH(I,J),3)=EDGE(IEDGPTH(I,J),3)-ZMIN
      KI=ISPEC(IEDGPTH(I,J),1)
      KJ=ISPEC(IEDGPTH(I,J),2)
      IF((NODTYP(KI).EQ.1).AND.(NODTYP(KJ).EQ.1).AND.
     1(KJ.NE.N)) GO TO 251
      IF(KJ.NE.N) GO TO 241
  251 BUFF=1
  241 IF(SLNNET.EQ.0) GO TO 248
      IF((NODTYP(KI).EQ.0).AND.(NODTYP(KJ).EQ.0)) GO TO 248
      IF((NODTYP(KI).EQ.1).AND.(NODTYP(KJ).EQ.1)) GO TO 248
      WRITE(7,246) KI
  248 CONTINUE
      IF(BUFF.EQ.0) GO TO 300
      SATTRFK=SATTRFK+ZMIN
  300 CONTINUE
      IF(MIN.GE.MAX) GO TO 399
C
*
* Then find the next least used link and repeat the above
* code to find the corresponding flow requirements and permanently
```

```
* routing them (This is repeated until all the links have
* been considered).
*
C                         NOW FIND NEXT LARGEST VALUE FOR MIN
C
      KK=MIN+1
      MIN=10000000
      DO 350 I=1,NE
      IF(EGFL(I,3).EQ.0) GO TO 350
  307 IF(EDGE(I,2).GE.MIN) GO TO 350
      IF(EDGE(I,2).LT.KK) GO TO 350
      MIN=EDGE(I,2)
  350 CONTINUE
      GO TO 210
  399 CONTINUE
C
*
* Having completed Step 3 of the algorithm, we are now about
* to start steps 4 and 5. Before doing so, we determine those
* parallel links whose capacity can be increased because
* all the required conditions pertaining to protection have been
* met.
*
C UPDATE LINK INFORMATION AND READJUST THE INCREMENTAL COSTS
C I.E. UPDATE MATRIX A
C
  400 DO 401 I=1,NREQ
      IF(RM(I,1).EQ.0) GO TO 401
      RM(I,2)=1
  401 CONTINUE
      IF(OUTPUT.EQ.0) GO TO 389
  388 FORMAT(1H0,'ZZZZZZZZZZZZZZZZZZZZZZZZZ',I15)
  389 DO 16005 I=1,NE
16005 EGFL(I,5)=EGFL(I,2)-EDGE(I,3)-EGFL(I,1)
      DO 16006 I=1,NE
16006 IPTHLG(I)=0
C              DETERMINE WHETHER OR NOT THE CAPACITY RESTRICTION ON THE
C      SYSTEMS WITHIN EACH CROSS SECTION CAN BE REMOVED
C
C              ARE THERE ANY CROSS SECTIONS TO BE CONSIDERED?
      IF(NCS.EQ.0) GO TO 16300
C              IF SO, CONSIDER THEM.
      DO 16260 I=1,NCS
C              CHECK IF THE CAPACITY RESTRICTION ON THE SYSTEMS IN  CROSS
C     SECTIONS HAVE ALREADY BEEN REMOVED
      IF(IPROPTR(I).EQ.1) GO TO 16260
C              DETERMINE THE LINKS IN THE CROSS SECTION I
      ZMIN=0
      DO 16190 J=1,9999
      IF(IPROTCT(I,J*3).EQ.9999)   GO TO 16200
      DO 16050 K=NODVTR(IPROTCT(I,J*3),1),NODVTR(IPROTCT(I,J*3),1)+
     1NODVTR(IPROTCT(I,J*3),2)-1
```

```
      IF(IPTHLG(K).EQ.1) GO TO 16050
      IF((ISPEC(K,2).EQ.IPROTCT(I,J*3+1))
     1.AND.(EDGE(K,4).EQ.IPROTCT(I,J*3+2))) GO TO 16060
16050 CONTINUE
16060 IRELVTR(J)=K
      IPTHLG(K)=1
C                HAS ALL THE MESSAGE BEEN ROUTED?(WE MUST DISTINGUISH FOR
C                PURPOSE OF OVERALL CONVERGENCE)
      IF(NREQ.EQ.TRFKCTR) GO TO 16100
C                NO IT HAS NOT.
      TP=1
      GO TO 16120
C                ONLY TV REMAINING TO BE ROUTED.
16100 TP=ITVTCTS
      IF(EDGE(K,4).NE.-1) GO TO 16120
      TP=ITVCNCP
      IF(EDGE(K,4).EQ.-2) GO TO 16120
      TP=ITVDIG
C                IS THE LOAD ON LINK K APPROACHING ITS RESTRICTED CAPACITY?
16120 IF((SPEC(K,ISPEC(K,3)*4+1)-EGFL(K,1)-EGFL(K,5)).GT.TP)GO TO 16190
      ZMIN=ZMIN+1
16190 CONTINUE
C                DETERMINE IF CAPACITY RESTRICTIONS SHOULD BE LIFTED
16200 IF(ZMIN.LT.IPROTCT(I,1)) GO TO 16260
C                YES, THEY SHOULD AS ALL THE LINKS ARE LOADED TO THEIR
C                RESTRICTED CAPACITY
      REWIND 4
      MIN=0
      DO 16240  MN=1,J-1
      MAX=9999
      DO 16210 K=1,J-1
      IF(IRELVTR(K).LE.MIN) GO TO 16210
      IF(IRELVTR(K).GE.MAX) GO TO 16210
      MAX=IRELVTR(K)
16210 CONTINUE
      L=MAX-MIN-1
      IF(L.EQ.0) GO TO 16223
      DO 16220 K=1,L*2
16220 READ(4,8) ZMIN
16223 READ(4,30) (ISPEC(MAX,M),M=1,3),(SPEC(MAX,M),M=1,20)
      ISPEC(MAX,3)=(ISPEC(MAX,3)-2)/4
      EGFL(MAX,6)=0
      MIN=MAX
16240 CONTINUE
      IPROPTR(I)=1
16260 CONTINUE
16300 CONTINUE
      IF(IREQCTR.LE.RCDR) GO TO 30011
      WRITE(6,30009) IREQCTR,SATTRFK
      RCDR=IREQCTR
30009 FORMAT(1H0,'NO. OF TRAFFIC REQ. ROUTED IS',I4,
     1'(SATELLITE TRAFFIC-',F7.0,')')
```

```
30011 IF(OUTPUT.EQ.0) GO TO 30000
      WRITE(1,30008) IREQCTR,TRFKCTR
30008 FORMAT(1H0,'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX',I10,5X,F10.1)
30000 CONTINUE
*
* Now consider each link I in the network.
*
      DO 699 I=1,NE
      FLG=0
  403 IF(ISPEC(I,3).EQ.0) GO TO 437
      IF(EGFL(I,6).EQ.1) GO TO 437
      EGFL(I,3)=1
      IF(FLWW.EQ.0) GO TO 440
*
* In this case some permanent flow was actually routed through
* the network (see Step 5 of the algorithm in  Section 4 of
* DLDCNS Report No. 14).
*
      TRBLCTR=1
C
C                 CONSIDER THE CASE WHERE SOME ADDITIONAL FLOW WAS
C                 ACTUALLY IMPOSED ON NETWORK
C
C                            FIRST UPDATE CURRENT FLOW
C
      IF((EGFL(I,1)+EGFL(I,5)).EQ.EGFL(I,2)) GO TO 430
      GO TO 421
C
C                           OTHERWISE EXPECTED FLOW . DESIRED FLOW
C
  405 IF(EGFL(I,2).EQ.SPEC(I,(IEGFL(I,2)-1)*4+1))
     1GO TO 406
      R1=SPEC(I,(IEGFL(I,2)-1)*4+2)
     1+SPEC(I,(IEGFL(I,2)-1)*4+3)
     2+(EGFL(I,2)-SPEC(I,(IEGFL(I,2)-1)*4+1))
     3*SPEC(I,(IEGFL(I,2)-1)*4+4)
      GO TO 407
  406 R1=SPEC(I,(IEGFL(I,2)-1)*4+2)
  407 IF(EGFL(I,1).EQ.SPEC(I,(IEGFL(I,1)-1)*4+1)) GO TO 408
      R2=SPEC(I,(IEGFL(I,1)-1)*4+2)
     1+SPEC(I,(IEGFL(I,1)-1)*4+3)+(EGFL(I,1)-
     2SPEC(I,(IEGFL(I,1)-1)*4+1))*SPEC(I,(IEGFL(I,1)-1)*4+4)
      GO TO 409
  408 R2=SPEC(I,(IEGFL(I,1)-1)*4+2)
  409 RUN=EGFL(I,2)-EGFL(I,1)
      SLP1=(R1-R2)/RUN
      GO TO 500
C
C                         WAS THE FLOW INCREMENTED?
C
  410 IF(EGFL(I,5).GT.0) GO TO 420
C
```

```
C                                  OTHERWISE ACTUAL FLOW WAS NOT INCRMENTED
C
      GO TO 405
C
C            COMPARE ACTUAL FLOW TO DESIRED FLOW
C
  420 IF((EGFL(I,1)+EGFL(I,5)).GE.EGFL(I,2)) GO TO 430
C
C                                        OTHERWISE ACTUAL FLOW INCREMEMNTED
C                                     WAS . DESIRED FLOW
C
  421 DO 423 L=1,ISPEC(I,3)
      IF((EGFL(I,1)+EGFL(I,5)).LE.SPEC(I,L*4+1))     GO TO 425
  423 CONTINUE
      L=L-1
  425 IEGFL(I,1)=L
      EGFL(I,1)=EGFL(I,1)+EGFL(I,5)
      GO TO 405
C
C                    HENCE THE DESIRED FLOW WAS ATTAINED
C
  430 IEGFL(I,1)=IEGFL(I,2)
  438 MIN=IEGFL(I,1)
      IF((EGFL(I,1)+EGFL(I,5)).NE.SPEC(I,IEGFL(I,2)*4+1))
     1GO TO 431
      IF(MIN.EQ.ISPEC(I,3)) GO TO 436
      MIN=IEGFL(I,1)+1
  431 SLP1=1.0E30
      DO 432 L=MIN,ISPEC(I,3)
      R1=SPEC(I,L*4+2)
      IF(EGFL(I,2).EQ.SPEC(I,(IEGFL(I,2)-1)*4+1)) GO TO 433
      R2=SPEC(I,(IEGFL(I,2)-1)*4+2)+SPEC(I,(IEGFL(I,2)-1)*4+3)
     1+(EGFL(I,2)-SPEC(I,(IEGFL(I,2)-1)*4+1))*
     2SPEC(I,(IEGFL(I,2)-1)*4+4)
      GO TO 434
  433 R2=SPEC(I,(IEGFL(I,2)-1)*4+2)
  434 RUN=SPEC(I,L*4+1)-EGFL(I,2)
      IF(((R1-R2)/RUN).GT.SLP1) GO TO 432
      SLP1=(R1-R2)/RUN
      KO=L
  432 CONTINUE
      IF(FLG.EQ.1) GO TO 435
      EGFL(I,1)=EGFL(I,1)+EGFL(I,5)
  435 IEGFL(I,2)=KO
      EGFL(I,2)=SPEC(I,(KO )*4+1)
      GO TO 500
  436 EGFL(I,1)=EGFL(I,1)+EGFL(I,5)
      EGFL(I,2)=EGFL(I,1)
  439 EGFL(I,6)=1
  437 SLP1=1.0E30
      GO TO 500
C
```

```
*
* The following code updates the weights of link I as required
* for the case where no permanent flow was routed through the network
* (corresponding to Step 4 of the algorithm).
*
C                    NOW CONSIDER THE CASE WHERE NO ADDITIONAL FLOW WAS
C                     IMPOSED ON THE NETWORK
C
  440 IF((EGFL(I,4)+EGFL(I,1)).LE.EGFL(I,2)) GO TO 450
C
C                             HENCE EXPECTED LOAD EXCEEDED DESIRED LOAD
C
      IF(EGFL(I,2).EQ.SPEC(I,ISPEC(I,3)*4+1)) GO TO 405
  444 IF(EGFL(I,2).LT.SPEC(I,IEGFL(I,2)*4+1)) GO TO 442
      IF(IEGFL(I,2).EQ.ISPEC(I,3)) GO TO 442
      MIN=IEGFL(I,2)+1
      GO TO 443
  442 MIN=IEGFL(I,2)
  443 EGFL(I,2)=EGFL(I,1)
      GO TO 431
13000 IF((SPEC(I,ISPEC(I,3)*4+1)-EGFL(I,1)).LT.TP)   GO TO 439
      EGFL(I,2)=EGFL(I,1)+TP-.1
      DO 17020 J=1,ISPEC(I,3)
17020 IF(EGFL(I,2).LE.SPEC(I,J*4+1)) GO TO 17030
17030 IEGFL(I,2)=J
      GO TO 444
C
C                         THUS ADJUST (INCREASE) THE COST OF THIS LINK
C
C
C                             IF EXP LOAD WAS  ZERO THEN DO NOT ADJUST SLOPE
C
  450 IF(EGFL(I,4).EQ.0) GO TO 405
  451 IF(EGFL(I,1).EQ.SPEC(I,(IEGFL(I,1)-1)*4+1)) GO TO 453
      R2=SPEC(I,(IEGFL(I,1)-1)*4+2)+SPEC(I,(IEGFL(I,1)-1)*4+3)+
     1(EGFL(I,1)-SPEC(I,(IEGFL(I,1)-1)*4+1))*
     2SPEC(I,(IEGFL(I,1)-1)*4+4)
      GO TO 454
  453 R2=SPEC(I,(IEGFL(I,1)-1)*4+2)
  454 RISE=R1-R2
  458 ZMIN=(EGFL(I,2)-EGFL(I,1)-EGFL(I,4))*BETA
     1+EGFL(I,1)+EGFL(I,4)
      DO 460 L=1,ISPEC(I,3)
      IF(ZMIN.LE.SPEC(I,L*4+1))      GO TO 465
  460 CONTINUE
      L=L-1
  465 R1=SPEC(I,(L-1)*4+2)+SPEC(I,(L-1)*4+3)+(ZMIN-SPEC(I,(L-1)*4+1))
     1*SPEC(I,(L-1)*4+4)
      SLP1=(R1-R2)/(ZMIN-EGFL(I,1))
      EGFL(I,2)=ZMIN
      IEGFL(I,2)=L
      GO TO 500
```

```
  480 SLP1=SLP2
      EGFL(I,2)=EGFL(I,1)+EGFL(I,4)
      IEGFL(I,2)=L
  500 IF(SLP1.GT.0) GO TO 504
      SLP1=.001
  504 EDGE(I,1)=SLP1
      IF(NREQ.NE.TRFKCTR) GO TO 503
      FLG=1
      IF(EDGE(I,1).EQ.1.0E30) GO TO 503
      TP=ITVTCTS
      IF(EDGE(I,4).GE.-1) GO TO 507
      TP=ITVCNCP
      IF(EDGE(I,4).EQ.-2) GO TO 507
      TP=ITVDIG
  507 IF((EGFL(I,2)-EGFL(I,1)).LT.TP)   GO TO 13000
  501 FORMAT(24HOUPDATED SLOPE IS  XXXXX,F10.5,7H  XXXXX)
  503 EGFL(I,4)=0
      EGFL(I,5)=0
      IEGFL(I,3)=0
  699 CONTINUE
*
* Increment TRBLCTR if no flow was permanently routed during the
* previous pass through steps 2 and 3 of the algorithm.
* If not all the flow requirements are routed then go back to
* repeat steps 2 and 3.
*
      IF(FLWW.EQ.1) GO TO 701
      TRBLCTR=TRBLCTR+1
  701 IF(IREQCTR.EQ.(NREQ+NOTVREQ)) GO TO 800
      IF(OUTPUT.EQ.0) GO TO 45
      WRITE(1,31000) (EDGE(I,1),I=1,NE)
31000 FORMAT(1H0,20(9(F10.3,2X),/))
  800 CONTINUE
      COST=TVSTTFK*STCT*ITVTCTS/ITVVCEQ
*
* A complete solution has been attained - the total load on
* each link, as well as the total flow through the satellite
* node, is output on Unit 7.
*
      DO 899 I=1,NE
      IF(EGFL(I,1).EQ.0) GO TO 899
      COST=COST+SPEC(I,(IEGFL(I,1)-1)*4+2)
     1+SPEC(I,(IEGFL(I,1)-1)*4+3)
     2+(EGFL(I,1)-SPEC(I,(IEGFL(I,1)-1)*4+1))
     3*SPEC(I,(IEGFL(I,1)-1)*4+4)
  899 CONTINUE
C000000000000000000000000000000000000000000000000000000000000
C000000000000000000000000000000000000000000000000000000000000
C                                              OUTPUT:UNITS 6,7,9
C0000000000000000000000000000000000000000000000000000000000000000
C00000000000000000000000000000000000000000000000000000000000000
  902 FORMAT(1H0,'  TOTAL SATELITE TRAFFIC EQUALS  ',F10.1,
```

```
     1'      (TV =',F10.1,'   )')
 949 FORMAT(1H0,'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX')
     WRITE(6,910) COST
 910 FORMAT(23H0  COST OF NETWORK IS  ,E15.6)
     WRITE(6,902) SATTRFK,TVSTTFK
     DO 985 I=1,NE
 985 WRITE(9,983) ISPEC(I,1),ISPEC(I,2),EGFL(I,1)
 983 FORMAT(5X,I5,6X,I5,5X,F12.1)
     WRITE(9,902) SATTRFK,TVSTTFK
     WRITE(9,910) COST
     WRITE(9,949)
     WRITE(7,987)
 987 FORMAT('9999')
     RETURN
 950 STOP
*
* Subroutine FPTH finds all the shortest paths for routing the flow
* requirements.  Every link M has a weight Y (subscript M) assigned
* to it.  The subroutine has two parts.  The first part finds all
* the shortest paths in the network from node I to all the other
* nodes in the network.  The array VRTX is used for this purpose.
* The shortest paths are found using an algorithm that appears on
* page 193 in the book "Communication, Transmission, and
* Transportation Networks" by Frank and Frisch.  The second part records
* the required shortest paths:  The shortest path for flow
* requirement J is stored i row J of array IEDGPTH.  The amount of
* flow on each link k is stored in EDGE(k,2).
*
     SUBROUTINE FPTH
     DO 3 I=1,NE
   3 EDGE(I,2)=0
     DO 1007 II=1,ITRFKTP(5)
     DO 1000 I=1,NREQNDS-1
     IF(I.EQ.1) GO TO 5
     DO 1 J=1,I-1
     VRTX(J,1)=0
   1 VRTX(J,2)=1.0E50
   5 VRTX(I,1)=0
     VRTX(I,2)=0
     IF(I.EQ.N) GO TO 10
     DO 8 J=I+1,N
     VRTX(J,1)=0
   8 VRTX(J,2)=1.0E50
  10 K=0
     DO 18 J=1,NE
     IF((NODTYP(ISPEC(J,2)).GT.0).AND.(SAT.EQ.0)) GO TO 18
   9 IF(EDGE(J,4).GE.2) GO TO 18
     IF((EDGE(J,4).GE.0).OR.(ITRFKTP(II).EQ.0)) GO TO 27
     IF(EDGE(J,4).NE.ITRFKTP(II)) GO TO 18
  27 IF((VRTX(ISPEC(J,1),2)+EDGE(J,1)).LT.VRTX(ISPEC(J,2),2)) GO TO 16
  11 IF((VRTX(ISPEC(J,2),2)+EDGE(J,1)).GE.VRTX(ISPEC(J,1),2)) GO TO 18
     K=K+1
```

```
      VRTX(ISPEC(J,1),1)=ISPEC(J,2)
      VRTX(ISPEC(J,1),2)=VRTX(ISPEC(J,2),2)+EDGE(J,1)
      GO TO 18
   16 K=K+1
      VRTX(ISPEC(J,2),1)=ISPEC(J,1)
      VRTX(ISPEC(J,2),2)=VRTX(ISPEC(J,1),2)+EDGE(J,1)
      GO TO 11
   18 CONTINUE
      IF(K.GT.0) GO TO 10
      DO 80 J=1,NREQ
      IF(IRM(J,3).NE.ITRFKTP(II)) GO TO 80
      IF(IRM(J,1).NE.I) GO TO 80
      IF(RM(J,1).EQ.0) GO TO 79
      RM(J,2)=1
      K2=IRM(J,2)
      IF(VRTX(K2,2).GE.1.0E30) GO TO 1011
      Z=K2
      DO 70 K=1,NE
      K1=VRTX(K2,1)
      IF(K1.LT.K2) GO TO 22
      N1=K2
      N2=K1
      GO TO 23
   22 N1=K1
      N2=K2
   23 TYP=1
      TRTP=IRM(J,3)
      CALL FIND
      IEDGPTH(J,K)=IPOS1
      EDGE(IPOS1,2)=EDGE(IPOS1,2)+RM(J,1)
      IF(K2.EQ.Z) GO TO 50
      IF(K2.EQ.IRM(J,1)) GO TO 75
      GO TO 69
   50 IF(K1.EQ.IRM(J,1)) GO TO 75
      K2=K1
   69 Z=K2
   70 CONTINUE
   75 IPTHLG(J)=K
      GO TO 80
   79 RM(J,2)=0
   80 CONTINUE
 1000 CONTINUE
 1007 CONTINUE
      TRBL=0
      GO TO 1010
 1011 TYPE(1)=0
      TYPE(2)=J
 1010 RETURN
*
* Subroutine STRTREE finds the minimum cost Steiner tree to connect
* specified nodes in a graph, using other nodes in the graph if there
* is a reduction in cost.  The subroutine STRTREE is called upon to
```

```
* determine the Steiner trees of the (sub)graphs A, B, and C discussed
* in Sections 5 and 6 of DLDCNS Report No. 14.  The basic strategy
* used to find the required Steiner tree is found on page 118 in
* Networks, Volume 1, Number 2 (in a paper by Hakimi entitled
* "Steiner's Problems in Graphs).  Subroutine STRTREE does all
* the bookkeeping, setting up an array called TABLE in which
* are stored the links of the subgraph whose nodes are to be
* connected by a minimum cost tree.  The tree itself is
* found in a subroutine called TREE.  Subroutine STRTREE must
* determine TABLE for every different subgraph, of the graph
* in which a Steiner tree is to be found, that contains all
* the nodes that must be interconnected by the Steiner tree.
* The nodes that are to be connected by the Steiner tree
* are presented to STRTREE in elements 3,...in the vector
* called GRP1.    IGRP1(1) = number of nodes in IGRP1(3),
* IGRP1(4),...etc.  IGRP1(2) = number of nodes in the vector
* IGRP2.  Array GRP2 contains all intermediate nodes that may
* or may not be in the Steiner tree.  From this information,
* STRTREE consider all possible subgraphs in the network that
* contain all the nodes in IGRP1 and some nodes in IGRP2.
* For each subset of nodes, all links in the network that join a
* pair of nodes in this subset are found and stored in the array
* TABLE.  Special provision has been built into the routine for
* temporarily eliminating some intermediate nodes (see
* discussion on special techniques discussed in Section 5 of DLDCNS
* Report No. 14).  All nodes between the terrestrial network model and
* the satellite node are eliminated.  Also, all nodes that lie
* between one of the nodes incident to a link and the real end
* (or destination) node (see the variable DTN(M) read in earlier
* from Unit 4) are eliminated.  To ensure that subroutine TREE is always
* able to find a tree, links with infinite weight are used.  Brief comments
* are made below to indicate what function various sections within the
* subroutine perform.
*
       SUBROUTINE STRTREE
      SNDCOST=0
      GRP4 =1.0E30
 3000 IPOS2=0
*
* Determine links incident to pairs of nodes in IGRP1.
*
      DO 3050 I=3,IGRP1( 1)+1
      DO 3049 J=I+1,IGRP1( 1)+2
      N1=IGRP1( I)
      N2=IGRP1( J)
      CALL SPRFIND
      IF(FOUND.EQ.0) GO TO 3049
      IPOS2=IPOS2+1
      TABLE(IPOS2,1)=N1
      TABLE(IPOS2,2)=N2
      TABLE(IPOS2,3)=METCOST+ZMIN
 3049 CONTINUE
```

```
 3050 CONTINUE
      DO 3055 I=4,IGRP1( 1)+2
       N1=IGRP1( 3)
      N2=IGRP1( I)
      CALL SPRFIND
      IF(FOUND.EQ.1) GO TO 3055
      IPOS2=IPOS2+1
      TABLE(IPOS2,1)=IGRP1( 3)
      TABLE(IPOS2,2)=IGRP1( I)
      TABLE(IPOS2,3)=1.0E50
 3055 CONTINUE
*
* Branch to statement 3526 if the satellite node N is in IGRP1.
* Then initialize various bookkeeping type variables.
* Basically, IGRP3 is a boolean vector to record which subsets
* of nodes in IGRP2 have been considered as intermediate nodes.
* The remaining variables are pointers.
*
      IF(BUFF.EQ.1) GO TO 3526
 3057 CTR4=IPOS2
      DO 3399 I2=0,IGRP1(  2)
      IF(I2.EQ.0) GO TO 3210
      DO 3070 I=1,IGRP1( 2)
 3070 IGRP3(I)=0
      DO 3075 I=1,I2
 3075 IGRP3(I)=1
      ICTR1=1
      CTR1=I2
      IPOS2=CTR4
      GO TO 3077
 3076 IPOS2=IPTR(ICTR1-1)-1
 3077 CTR2=0
      DO 3200 I=ICTR1,IGRP1( 2)
      FLW=0
      IF(IGRP3(I).EQ.0) GO TO 3200
*
* If the satellite node is not in IGRP1, then branch to
* Statement #3890.  Otherwise determine links in the network
* model that exist between the nodes in IGRP2 and the satellite
* node N.
*
      IF(BUFF.EQ.0) GO TO 3890
      IF(NODVTR(IGRP2(I),2).EQ.0) GO TO 3890
      IF(IGRP2(I).NE.TVREQ1(I1)) GO TO 3082
      IF((SNDND.NE.IGRP2(I)).AND.(TYPE(I1).EQ.3)) GO TO 3890
 3082 DO 3830 J=NODVTR(IGRP2( I),1),
     1NODVTR(IGRP2( I),1)+NODVTR(IGRP2( I),2)-1
      IF(NODTYP(ISPEC(J,2)).GT.0) GO TO 3850
 3830 CONTINUE
      GO TO 3890
 3850 IF(IGRP2(I).NE.SNDND) GO TO 3856
      TP=2
```

```
      GO TO 3857
 3856 TP=TYPE(I1)
 3857 DO 3860 K=NODVTR(ISPEC(J,2),1),
     1NODVTR(ISPEC(J,2),1)+NODVTR(ISPEC(J,2),2)-1
 3860 IF(EDGE(K,4).EQ.TP)        GO TO 3870
      GO TO 3890
 3870 IPOS2=IPOS2+1
      TABLE(IPOS2,1)=IGRP2( I)
      TABLE(IPOS2,2)=N
      TABLE(IPOS2,3)=(EDGE(J,1)+EDGE(K,1))*ITVVCEQ/ITVTCTS
      IPTR(I)=IPOS2
      FLW=1
      IF(IGRP2(I).NE.SNDND) GO TO 3890
      SNDCOST=TABLE(IPOS2,3)
      TABLE(IPOS2,3)=.0001
 3890 IF(I.EQ.1) GO TO 3090
*
* Find all the links incident to pairs of nodes in IGRP2 and links
* incident to a node in IGRP1 and to another node in IGRP2.
*
      N2=IGRP2( I)
      DO 3085 J=I-1,1,-1
      IF(IGRP3(J).EQ.0) GO TO 3085
      N1=IGRP2( J)
      CALL SPRFIND
      IF(FOUND.EQ.0) GO TO 3085
      IPOS2=IPOS2+1
      IF(FLW.EQ.1) GO TO 3079
      IPTR(I)=IPOS2
 3079 TABLE(IPOS2,1)=N1
      TABLE(IPOS2,2) =N2
      TABLE(IPOS2,3)=METCOST+ZMIN
      FLW=1
 3085 CONTINUE
 3090 DO 3110 J=3,IGRP1( 1)+2
      IF(IGRP2( I).GT.IGRP1( J))          GO TO 3093
      N1=IGRP2( I)
      N2=IGRP1( J)
      GO TO 3095
 3093 N2=IGRP2( I)
      N1=IGRP1( J)
 3095 CALL SPRFIND
      IF(FOUND.EQ.0) GO TO 3110
      IPOS2=IPOS2+1
      IF(FLW.EQ.1) GO TO 3097
      IPTR(I)=IPOS2
 3097 TABLE(IPOS2,1)=N1
      TABLE(IPOS2,2)=N2
      TABLE(IPOS2,3)=METCOST+ZMIN
      FLW=1
 3110 CONTINUE
      IF(FLW.EQ.1) GO TO 3200
```

```
      IPOS2=IPOS2+1
      IPTR(I)=IPOS2
 3103 TABLE(IPOS2,1)=N1
      TABLE(IPOS2,2)=N2
      TABLE(IPOS2,3)=1.0E50
 3200 CONTINUE
*
* Call subroutine TREE to find the minimum cost tree connecting
* the nodes incident to the links stored in the first IPOS2 rows
* of the array TABLE. There is one link for each row I of
* array TABLE, I=1,....,IPOS2, where the I Th link is incident
* to the nodes stored in TABLE(I,1) and TABLE(I,2).
* The weight Y (subscript I) assigned to the I Th link is
* stored in TABLE(I,3).
*
 3210 CALL TREE
*
* A record of the links in the tree just found is kept if the
* cost of the tree is better than any previously found tree.
* Otherwise branch ahead to Statement #3305.
*
      COST=COST+SNDCOST
      IF(COST.GE.1.0E30)GO TO 3305
 3213 IF(COST.GE.GRP4 )    GO TO 3305
 3217 GRP4 =COST
 3218 DO 3220 I=1,NE
 3220 STREDG(I1,I)=0
      DO 3240 I=1,IPOS2
      IF((TBL(I,3).EQ.0).OR.(TBL(I,3).EQ.2)) GO TO 3240
      IF(TABLE(I,3).EQ.1.0E50) GO TO 3305
      N1=TABLE(I,1)
      N2=TABLE(I,2)
      IF(N2.EQ.N) GO TO 3230
      CALL FINDER
      GO TO 3240
 3230 DO 3250 J=NODVTR(N1,1),NODVTR(N1,1)+NODVTR(N1,2)-1
      IF(NODTYP(ISPEC(J,2)).GT.0) GO TO 3255
 3250 CONTINUE
 3255 N2=ISPEC(J,2)
      CALL FIND
      STREDG(I1,IPOS1)=1
      TP=TYP
      IF(N1.NE.SNDND) GO TO 3237
      TYP=2
 3237 N1=N2
      N2=N
      CALL FIND
      TYP=TP
      STREDG(I1,IPOS1)=1
 3240 CONTINUE
*
* Before going back to determining the next tree, some bookkeeping
```

```
* must be done:  The vector IGRP3 is updated; the next subset of
* nodes to be considered, from those in IGRP2, is determined.
*
 3305 CTR1=0
      IF(I2.EQ.0) GO TO 3399
      DO 3310 I=IGRP1( 2),1,-1
      IF(IGRP3(I).EQ.0) GO TO 3310
      CTR1=CTR1+1
      IF(I.EQ.IGRP1( 2))    GO TO 3310
      IF(IGRP3(I+1).EQ.0) GO TO 3320
 3310 CONTINUE
      GO TO 3399
 3320 IGRP3(I)=0
      IGRP3(I+1)=1
      ICTR1=I+1
      IF(CTR1.EQ.1)  GO TO 3076
      IF((I+1).EQ.IGRP1( 2))    GO TO 3076
      DO 3330 J=I+2,IGRP1( 2)
 3330 IGRP3(J)=0
      DO 3340 J=I+2,I+CTR1
 3340 IGRP3(J)=1
      GO TO 3076
 3399 CONTINUE
      GO TO 3505
*
* The following code determines links which are in the network
* model and that connect nodes in vector IGRP1 to the satellite
* node N.
*
 3526 DO 3590 I=3,IGRP1( 1)+2
      IF(IGRP1(I).NE.TVREQ1(I1)) GO TO 3519
      IF((SNDND.NE.IGRP1(I)).AND.(TYPE(I1).EQ.3)) GO TO 3590
 3519 IF(IGRP1(I).NE.SNDND) GO TO 3528
      TP=2
      GO TO 3529
 3528 TP=TYPE(I1)
 3529 IF(NODVTR(IGRP1(I),2).EQ.0) GO TO 3590
      DO 3530 J=NODVTR(IGRP1( I),1),NODVTR(IGRP1( I),1)
     1+NODVTR(IGRP1( I),2)-1
      IF(NODTYP(ISPEC(J,2)).GT.0) GO TO 3550
 3530 CONTINUE
      GO TO 3590
 3550 DO 3560 K=NODVTR(ISPEC(J,2),1),
     1NODVTR(ISPEC(J,2),1)+NODVTR(ISPEC(J,2),2)-1
 3560 IF(EDGE(K,4).EQ.TP)        GO TO 3570
 3570 IPOS2=IPOS2+1
      TABLE(IPOS2,1)=IGRP1( I)
      TABLE(IPOS2,2)=N
      TABLE(IPOS2,3)=(EDGE(J,1)+EDGE(K,1))*ITVVCEQ/ITVTCTS
      FLW=1
      IF(IGRP1(I).NE.SNDND) GO TO 3590
      SNDCOST=TABLE(IPOS2,3)
```

```
      TABLE(IPOS2,3)=.0001
 3590 CONTINUE
      IF(FLW.EQ.1) GO TO 3596
      IPOS2=IPOS2+1
      TABLE(IPOS2,1)=IGRP1( 3)
      TABLE(IPOS2,2)=N
      TABLE(IPOS2,3)=1.0E50
 3596 CTR4=IPOS2
      GO TO 3057
 3500 CONTINUE
 3505 CONTINUE
      RETURN
*
* Subroutine FIND determines whether or not there is a link
* in the network model that is incident to nodes N1 and
* N2, and furthermore that the link can carry the type of
* flow requirement (message, half-duplex television, simplex
* television).
*
      SUBROUTINE FIND
      FOUND=0
      IF(NODVTR(N1,2).EQ.0) GO TO 15
      ZMIN=1.0E50
      DO 10 IX=NODVTR(N1,1),NODVTR(N1,2)-1+NODVTR(N1,1)
      IF(ISPEC(IX,2).NE.N2) GO TO 10
      IF((EDGE(IX,4).GT.0).AND.(EDGE(IX,4).NE.TYP)) GO TO 10
      IF((EDGE(IX,4).GE.0).OR.(TRTP.EQ.0)) GO TO 5
      IF(EDGE(IX,4).NE.TRTP) GO TO 10
    5 FOUND=1
      IF(EDGE(IX,1).GE.ZMIN) GO TO 10
      ZMIN=EDGE(IX,1)
      IPOS1=IX
   10 CONTINUE
      GO TO 20
   15 CONTINUE
   20 RETURN
*
* Subroutine Tree finds the minimum cost tree in a graph
* according to an algorithm given on page 207 in a book
* entitled Communications, Transmission, and Transportation
* Networks by Frank and Frisch.
*
      SUBROUTINE TREE
      DO 3550 I=1,IPOS2
      TBL(I,1)=TABLE(I,1)
      TBL(I,2)=TABLE(I,2)
 3550 TBL(I,3)=0
      DO 3599 I=1,IGRP1( 1)+I2-1+BUFF
      ZMIN=1.0E52
      DO 3570 J=1,IPOS2
      IF(TABLE(J,3).GE.ZMIN) GO TO 3570
      IF(TBL(J,3).NE.0) GO TO 3570
```

```
      ZMIN=TABLE(J,3)
      K=J
 3570 CONTINUE
      TBL(K,3)=1
      DO 3580 J=1,IPOS2
      IF(J.EQ.K) GO TO 3580
      IF(TBL(J,1).NE.TBL(K,2))   GO TO 3572
      TBL(J,1)=TBL(K,1)
 3572 IF(TBL(J,2).NE.TBL(K,2))   GO TO 3578
      TBL(J,2)=TBL(K,1)
 3578 IF(TBL(J,1).NE.TBL(J,2)) GO TO 3580
      TBL(J,3)=2
 3580 CONTINUE
 3599 CONTINUE
      COST=0
      DO 3585 J=1,IPOS2
      IF(TBL(J,3).NE.1) GO TO 3585
      COST=COST+TABLE(J,3)
 3585 CONTINUE
      RETURN
*
* Subroutine X1 determines those flow requirements,
* corresponding to television traffic, that have
* temporarily been routed through a link for which the
* expected cost of flow is too unrealistic compared to
* the real cost.
*
      SUBROUTINE X1
      IF(NOTVREQ.EQ.0) GO TO 8400
      DO 8395 I=1,NOTVREQ
      IF(TVRQPTR(I).EQ.0) GO TO 8395
      DO 8390 J=1,NE
      IF(STREDG(I,J).EQ.0) GO TO 8390
      IF(EDGE(J,1).LT.1.0E29) GO TO 8390
      TVRQPTR(I)=0
      DO 8370 L=1,NE
      IF(STREDG(I,L).EQ.0) GO TO 8370
      IF(NODTYP(ISPEC(L,1)).NE.1) GO TO 8356
      TV=ITVVCEQ
      GO TO 8360
 8356 TV=ITVTCTS
      IF((EDGE(L,4).EQ.-1).OR.(NODTYP(ISPEC(L,2)).EQ.1))
     1GO TO 8360
      TV=ITVCNCP
 8360 TMPEDG(L)=TMPEDG(L)-TVREQ(I)*TV
 8370 CONTINUE
      GO TO 8395
 8390 CONTINUE
 8395 CONTINUE
 8400 CONTINUE
      RETURN
*
```

```
* Subroutine X2 is the main subroutine for determining the required
* Steiner trees for television traffic.
*
y
      SUBROUTINE X2
      DO 7920 I=1,NE
 7920 TMPEDG(I)=0
      IF(NOTVREQ.EQ.0) GO TO 7963
      DO 7950 I1=1,NOTVREQ
      IF(TVREQ(I1).EQ.0) GO TO 7950
      TVRQPTR(I1)=1
      TYP=TYPE(I1)
      TRTP=TVCR(I1)
      CALL PTMT
      IF(GRP4.GE.1.0E30)      GO TO 7970
      DO 7945 J=1,NE
      IF(STREDG(I1,J).EQ.0) GO TO 7945
      IF(NODTYP(ISPEC(J,1)).NE.1) GO TO 7941
      TV=ITVVCEQ
      GO TO 7942
 7941 TV=ITVTCTS
      IF((EDGE(J,4).EQ.-1).OR.(NODTYP(ISPEC(J,2)).EQ.1))
     1GO TO 7942
      TV=ITVCNCP
      IF(EDGE(J,4).EQ.-2) GO TO 7942
      TV=ITVDIG
 7942 TMPEDG(J)=TMPEDG(J)+TVREQ(I1)*TV
 7945 CONTINUE
 7950 CONTINUE
 7963 TRBL=0
      GO TO 7965
 7970 TYPE(1)=1
      TYPE(2)=I1
 7965 CONTINUE
      RETURN
*
* Subroutine X3 determines how the maximum number of television
* channels that can be routed through the corresponding tree
* without violating the constraint that the total flow through
* each linok be less than or equal to U (subscript M)
* (see Step 3 of the algorithm described in Section ?
* of DLDCNS Report No. ?).
*
      SUBROUTINE X3
      ZMIN=1.0E40
      DO 6050 J=1,NE
      IF(STREDG(I,J).EQ.0) GO TO 6050
      IBUFF=EDGE(J,3)
      IF(NODTYP(ISPEC(J,1)).NE.1) GO TO 6031
      IBUFF=(IBUFF/ITVVCEQ)*ITVTCTS
      GO TO 6035
 6031 IF((EDGE(J,4).EQ.-1).OR.(NODTYP(ISPEC(J,2)).EQ.1))
```

```
     1GO TO 6035
      IBUFF=(IBUFF/ITVCNCP)*ITVTCTS
      IF(EDGE(J,4).EQ.-2) GO TO 6035
      IBUFF=(IBUFF/ITVDIG)*ITVTCTS
 6035 IF(IBUFF.GE.ZMIN)     GO TO 6050
      ZMIN=IBUFF
      JO=J
 6050 CONTINUE
      K=TVREQ(I)
      DO 6070 J=1,K
      IF((J*ITVTCTS).GT.ZMIN) GO TO 6075
 6070 CONTINUE
 6075 J=J-1
      FLG=J
      IF(FLG.NE.0) GO TO 6076
      GO TO 6085
 6076 DO 6080 L=1,NE
      IF(STREDG(I,L).EQ.0) GO TO 6080
      IF(NODTYP(ISPEC(L,1)).NE.1) GO TO 6090
      TV=ITVVCEQ
      GO TO 6077
 6090 TV=ITVTCTS
      IF((EDGE(L,4).EQ.-1).OR.(NODTYP(ISPEC(L,2)).EQ.1))
     1GO TO 6077
      TV=ITVCNCP
       IF(EDGE(L,4).EQ.-2) GO TO 6077
      TV=ITVDIG
 6077 TMPEDG(L)=TMPEDG(L)+J*TV
 6080 CONTINUE
 6085 RETURN
*
* Subroutine X4 is a control routine for permanently routing
* television traffic.
*
      SUBROUTINE X4
      X44=1
      IF(NOTVREQ.EQ.0) GO TO 6195
      DO 6190 I=1,NOTVREQ
      IF(TVREQ(I).EQ.0) GO TO 6190
      IF(TVRQPTR(I).EQ.0) GO TO 6190
      DO 6130 J=1,NE
 6130 TMPEDG(J)=0
      CALL X3
      IF(FLG.EQ.0) GO TO 6190
      BUFF=0
      IL=0
 6147 DO 6150 J=1,NE
      IF(STREDG(I,J).EQ.0) GO TO 6150
      IF(ISPEC(J,2).NE.N) GO TO 6173
      BUFF=1
 6173 IF(SLNNET.EQ.0) GO TO 6150
      IF((NODTYP(ISPEC(J,1)).NE.0).OR.(NODTYP(ISPEC(J,2)).NE.1))
```

```
      1GO TO 6150
       IL=IL+1
       IGRP1(IL)=ISPEC(J,1)
 6150  EDGE(J,3)=EDGE(J,3)-TMPEDG(J)
       IF(SLNNET.EQ.0) GO TO 6170
       IF((SLNNET.EQ.1).AND.(IL.EQ.0)) GO TO 6170
       WRITE(7,6171) FLG,I,TVREQ1(I),(IGRP1(J),J=1,IL)
 6171  FORMAT(20I)
 6170  TVREQ(I)=TVREQ(I)-FLG
       FLWW=1
       IF(BUFF.EQ.0) GO TO 6155
       SATTRFK=SATTRFK+FLG*ITVVCEQ
       TVSTTFK=TVSTTFK+FLG*ITVVCEQ
 6155  IF(TVREQ(I).GT.0) GO TO 6190
       TVRQPTR(I)=0
       IREQCTR=IREQCTR+1
 6190  CONTINUE
 6195  CONTINUE
       RETURN
       SUBROUTINE SPRFIND
       FOUND=0
       IF(NODVTR(N1,2).EQ.0) GO TO 20
       ZMIN=1.0E50
       IZMIN=0
       IF(NODVTR(N1,2).EQ.0) GO TO 20
       DO 10 IX=NODVTR(N1,1),NODVTR(N1,2)-1+NODVTR(N1,1)
       IF(DTN(IX).NE.N2)     GO TO 10
       IZMIN=IZMIN+1
       IF((EDGE(IX,4).GE.0).OR.(TRTP.EQ.0)) GO TO 5
       IF(TRTP.NE.EDGE(IX,4)) GO TO 10
    5  ZZMIN=EDGE(IX,1)
       IF(EDGE(IX,4).GE.-1) GO TO 11
    9  ZZMIN=EDGE(IX,1)*ITVCNCP/ITVTCTS
       IF(EDGE(IX,4).GE.-1) GO TO 11
       ZZMIN=EDGE(IX,1)*ITVDIG/ITVTCTS
   11   IF(ZZMIN.GE.ZMIN)      GO TO 10
       FOUND=1
       IPOS1=IX
       ZMIN=ZZMIN
       TABLE(IPOS2+1,4)=IZMIN
   10  CONTINUE
       METCOST=0
   20  RETURN
       SUBROUTINE FINDER
       IZMIN=0
       DO 10 IX=NODVTR(N1,1),NODVTR(N1,2)-1+NODVTR(N1,1)
       IF(DTN(IX).NE.N2)     GO TO 10
       IZMIN=IZMIN+1
       IF(TABLE(I,4).EQ.IZMIN)  GO TO 40
   10  CONTINUE
   40  STREDG(I1,IX)=1
       IF(ISPEC(IX,2).EQ.N2) GO TO 20
```

```
      N1=N2
      N2=ISPEC(IX,2)
      CALL FIND
      STREDG(I1,IPOS1)=1
   20 RETURN
      SUBROUTINE PTMT
      DO 19005 I=1,NE
      STREDGA(I)=0
19005 STREDGB(I)=0
      KC=1
      KAA=1
      KBB=1
      SNDND=0
19007 DO 19010 KA=KAA,1000
      IF(SBGRPA(I1,KA).EQ.9999) GO TO 19015
19010 IGRP1(KA-KAA+3)=SBGRPA(I1,KA)
19015 IGRP1(1)=KA-KAA
      KAA=KA+1
      IF(IGRP1(1).EQ.0) GO TO 19101
      DO 19020 KB=KBB,1000
      IF(SBGRPB(I1,KB).EQ.9999) GO TO 19025
19020 IGRP2(KB-KBB+1)=SBGRPB(I1,KB)
19025 IGRP1(2)=KB-KBB
      KBB=KB+1
19030 DO 19028 I=1,NE
19028 STREDG(I1,I)=0
      BUFF=0
      CALL STRTREE
      DO 19040 I=1,NE
      IF(STREDG(I1,I).EQ.0) GO TO 19040
      STREDGA(I)=KC
19040 CONTINUE
      COSTA(KC)=GRP4
      KC=KC+1
      GO TO 19007
19101 CSTA=0
      DO 19103 I=1,KC-1
19103 CSTA=CSTA+COSTA(I)
      IF(SAT.EQ.0) GO TO 19345
      KC=1
      COSTB(KC)=1.0E60
      KAA=1
      KBB=1
      GO TO 19190
19107 DO  19110 KA=KAA,1000
      IF(SBGRPC(I1,KA).EQ.9999) GO TO 19115
19110 IGRP1(KA-KAA+3)=SBGRPC(I1,KA)
19115 IGRP1(1)=KA-KAA
       IF(IGRP1(1).EQ.0) GO TO 19320
      IF(SBGRPD(I1,1).NE.10000) GO TO 19118
      IGRP1(2)=0
      GO TO 19127
```

```
19118 DO 19120 KB=KBB,1000
      IF(SBGRPD(I1,KB).EQ.9998) GO TO 19125
19120 IGRP2(KB-KBB+1)=SBGRPD(I1,KB)
19125 IGRP1(2)=KB-KBB
      KBB=KB+1
      IF(IGRP2(1).EQ.0) GO TO 19150
19127 DO 19128 I=1,NE
19128 STREDG(I1,I)=0
      BUFF=0
      CALL STRTREE
      DO 19135 I=1,NE
      IF(STREDG(I1,I).EQ.0) GO TO 19135
      STREDGB(I)=KC
19135 CONTINUE
      COSTB(KC)=GRP4
      GO TO 19190
19150 COSTB(KC)=COSTA(IGRP2(2))
      DO 19160 I=1,NE
      IF(STREDGA(I).NE.IGRP2(2)) GO TO 19160
      STREDGB(I)=KC
19160 CONTINUE
19190 L=0
      DO 19210 KA=KAA,1000
      IF(SBGRPC(I1,KA).EQ.9999)GO TO 19215
      IF(KAA.EQ.1) GO TO 19213
      DO 19212 I=1,KAA-2
19212 IF(SBGRPC(I1,I).EQ.SBGRPC(I1,KA)) GO TO 19210
19213 L=L+1
      IGRP1(L+2)=SBGRPC(I1,KA)
19210 CONTINUE
19215 IGRP1(1)=L
19217 KAA =KA+1
      DO 19220 KB=KBB,1000
      IF(SBGRPD(I1,KB).EQ.9999) GO TO 19225
19220 IGRP2(KB-KBB+1)=SBGRPD(I1,KB)
19225 IGRP1(2)=KB-KBB
      KBB=KB+1
      SNDFLG=0
      DO 19240 I=3,IGRP1(1)+2
19240 IF(TVREQ1(I1).EQ.IGRP1(I)) GO TO 19260
      SNDND=0
      GO TO 19275
19260 SNDFLG=1
      DO 19270 I=1,1000
      IF(SEND(I1,I).EQ.9999) GO TO 19271
19270 CONTINUE
19271 NOSNDND=I-1
      DO 19299 KA=1,NOSNDND
      SNDND=SEND(I1,KA)
19275 BUFF=1
      DO 19227 I=1,NE
19227 STREDG(I1,I)=0
```

```
      CALL STRTREE
      IF(COSTB(KC).LE.GRP4) GO TO 19298
      COSTB(KC)=GRP4
      DO 19290 I=1,NE
      IF(STREDGB(I).NE.KC)  GO TO 19280
      STREDGB(I)=0
19280 IF(STREDG(I1,I).NE.1) GO TO 19290
      STREDGB(I)=KC
19290 CONTINUE
19298 IF(SNDFLG.EQ.0) GO TO 19300
19299 CONTINUE
19300 KC=KC+1
      GO TO 19107
19320 CSTB=0
      DO 19340 I=1,KC-1
19340 CSTB=CSTB+COSTB(I)
      CSTB=CSTB+STCT
      IF(CSTB.LT.CSTA) GO TO 19360
19345 GRP4=CSTA
      DO 19350 I=1,NE
      STREDG(I1,I)=0
      IF(STREDGA(I).EQ.0) GO TO 19350
      STREDG(I1,I)=1
19350 CONTINUE
      GO TO 19380
19360 GRP4=CSTB
      DO 19370 I=1,NE
      STREDG(I1,I)=0
      IF(STREDGB(I).EQ.0) GO TO 19370
      STREDG(I1,I)=1
19370 CONTINUE
19380 RETURN
      SUBROUTINE RCST
      IF(EDGE(I,2).NE.0) GO TO 60004
      BUFF=0
      GO TO 60003
60004 IF((EGFL(I,1)+EDGE(I,2)).LT.EGFL(I,2)) GO TO 60000
      BUFF=1
      GO TO 60003
60000 DO 60001 M=1,ISPEC(I,3)
      IF((EDGE(I,2)+EGFL(I,1)).LE.SPEC(I,M*4+1)) GO TO 60002
60001 CONTINUE
60002 BUFF=SPEC(I,(M-1)*4+2) +SPEC(I,(M-1)*4+3)
     1+SPEC(I,(M-1)*4+4)*(EDGE(I,2)+EGFL(I,1)-SPEC(I,(M-1)*4+1))
       BUFF=(EDGE(I,1)*EDGE(I,2))/(BUFF-STREDGA(I))
60003 CONTINUE
      IF((OUTPUT.EQ.0).OR.(EDGE(I,2).EQ.0)) GO TO 60009
      WRITE(1,60010) I,EGFL(I,1),EGFL(I,2),EDGE(I,2),BUFF,SALPHA
60010 FORMAT(1H1,I3,1X,5( F11.3,1X))
60009 RETURN
      END
```