# A BONeS Model of a Telephone System

**Dan S. Reznik**
**Victor S. Frost**

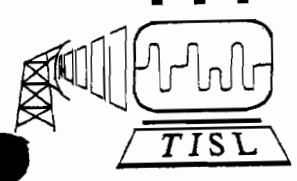
**TISL Technical Report TISL-8670-05**

Prepared for
COMDISCO Systems Inc.
1310 Wakarusa Drive, Suite A
Lawrence, Kansas 66049

and

Kansas Technology Enterprise Corporation (KTEC)
112 W 6th, Suite 400
Topeka, Kansas 66603

August 1990




Telecommunications and Information Sciences Laboratory
The University of Kansas Center for Research, Inc.
2291 Irving Hill Drive Lawrence, Kansas 66045

# A BONeS Model of a Telephone System

Dan S. Reznik

Victor S. Frost

TISL Technical Report TISL 8670-5

Prepared for:

COMDISCO Systems, Inc.
1310 Wakarusa Drive, Suite A
Lawrence, Kansas 66049

and

Kansas Technology Enterprise Corporation (KTEC)
112 W. 6th, Suite 400
Topeka, Kansas 66603

August 1990

## Abstract

A set of BONeS objects (data structures, modules, simulations, and plots) has been developed, re-used, enhanced, and integrated into a BONeS model for a telephone network. Results extracted from system's level simulations include call processing visualization, blocking ratios, and unbalanced-network performance. Both high reusability of BONeS objects, and overall modularity of the block-oriented paradigm have resulted in a compact and rapid implementation of the proposed model. The BONeS model developed here provides a basis for the modeling of CCITT Signaling System 7 and Intelligent Networks.

# Table of Contents

## Table of Contents (Cont.)

# List of Figures

## List of Figures (Cont.)

## List of Tables

# 1. Introduction

## 1.1 Overview of the Model

This report describes a BONeS model for a simple telephone system. The modeling methodology is based on Figure 1.1. Major functional units represented in this figure include: local offices, circuit-switching tandems, and telephone traffic sources.

Telephone traffic sources generate call traffic which is input to a corresponding local office. Traffic units correspond to call-initiation requests directed to random remote locations.

Local offices process input traffic by setting up connections with the requested locations. Connections are setup over a network of circuit-switching tandems; local offices and tandems are interconnected by a finite number of trunks.

Figure 1.2 (extracted from [3]) depicts the signaling scheme used to setup/teardown telephone calls (i.e., call processing). In this figure, a source local-office initiates a call by sending a request to its neighboring nodal processor (tandem). The request is forwarded over multiple hops to the destination local office. An "answer" (call accepted) signal is then returned to the initiating end and the connection is established. When the call is ready to be cleared (e.g., a two-party conversation is over), the source local office transmits a clearing signal to the remote end and the connection is torn down.

Note: The current model requires an additional clear confirmation signal from the receiving end before a call can be completely cleared.

All the major elements of telephone call processing are modeled. The BONeS modules developed here provide the basis for the modeling of CCITT Signaling System 7 as well as Intelligent Networks.

## 1.2 Parameters Under Study

The system's level model allows for the following parameters to be studied:

(i) Call processing visualization:

The activity that takes place at any local-office-to-tandem interface shows the handshaking procedures involved in setting-up, conducting, and tearing-down a connection. This is useful to understanding the dynamics of call processing, viewing trunk utilization, and tracking the progression of individual calls.

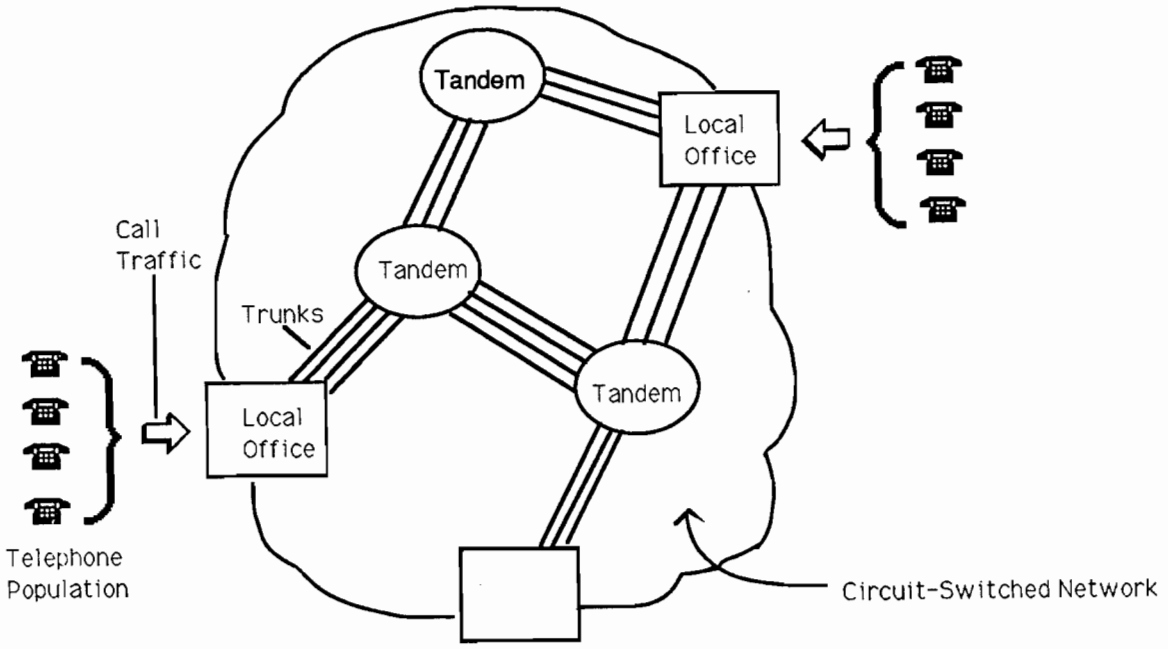



Figure 1.1 - Functional Units on a Simple Telephone System

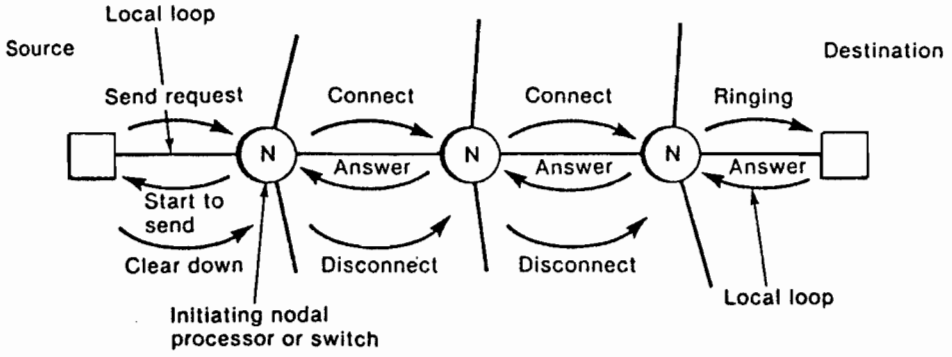



Figure 1.2 Call Processing Signalling [3].

(ii) Blocking ratios (local & remote):

Blocking results from frustrated call-initiation requests. This occurs when a local office is unable to setup a requested call because all the paths to the remote destination are busy.

Note: The current model assumes that tandems are non-blocking. Therefore, blocking situations can only occur at local-office-to-tandem interfaces, and never between tandems. Local blocking is attributed to trunk overflow at call-initiating local-offices; remote blocking is associated with the absence of incoming trunks at receiving-end local-offices.

(iii) Call setup time:

This measures the time (sampled over a large number of calls) it takes for a local-office to successfully achieve a connection.

### 1.3 System's Control Parameters

The following control parameters can be adjusted to study various aspects of the system performance:

(i) Number of trunks:

The number of trunks (physical connections) available at any local-office-to-tandem interface in the system can be set on a per-local-office basis.

(ii) Call traffic intensity

The call traffic intensity offered to individual local offices can be easily set.

(iii) Tandem processing speed

Network performance is a function of the speed at which tandems can setup connections. This parameter is also easily set in the current model.

(iv) Distribution of call flow

The remote destinations of call-initiation requests can be distributed arbitrarily. This also allows for the system's performance parameters to be studied in a variety of unbalanced-traffic situations.

### 1.4 Documentation Structure

The major aspects of the current model are distributed over sections 2 through 4. In section 2, "Model Overview", the basic modeling techniques and assumptions of the

telephone-system model are outlined. The system is viewed at its highest level of abstraction. In section 3, "Implementation Description", the BONeS objects used to construct system's level simulations are described. In section 4, "Simulation Systems & Results", the specific system's level simulations used for the current model are discussed. Parameter settings and result probing techniques are also explained.

## 2. Model Overview

### 2.1 An Abstract View of the System's Level Model

An abstract view of the telephone-system model is presented in Figure 2.1. As shown, the system is composed of 6 local-office units (labeled L0-L5) receiving call traffic from individual call traffic sources (labeled CTSs). CTSs produce traffic equivalent to that of a large population of telephones.

The circuit-switched network is composed of four tandems (labeled S0-S3) interconnected in a star-like fashion (note that this topology was chosen arbitrarily).

Tandems are two-way, three-port units. Each port in a tandem is labeled P1-P3. Every local-office-to-tandem interface admits a finite number of trunks. This imposes a limit on the number of local-office-to-tandem connections that can coexist at any given time. However, inter-tandem connections do not impose such limitations -- tandems are modeled as non-blocking.

### 2.2 Top-Level BONeS Units & General System's Functionality

Figure 2.2 depicts the top-level BONeS diagram of the proposed model. Call traffic sources, local offices, and tandems are one-to-one related to Figure 2.1's units.

#### 2.2.1 Call Traffic Sources (CTSs)

Call traffic sources generate a stream of call-initiation request (CIReq) data structures (DS's). Each such DS's contain fields for the issuing time, the sending/destination addresses, and the desired conversation time. CTSs are controlled by important modeling parameters such as: total number of traffic-generating subscribers, frequency of calls per subscriber, and percent calls needing outgoing trunks.

Each CIReq issued by a CTS has its "destination" field set randomly over a user-specified range of remote addresses. This allows for network traffic to be distributed in a variety of ways.

#### 2.2.2 Local Offices

The internal structure of the "Local Office" module is shown in Figure 2.3. This module is composed of two component blocks, "CallProc (Tel)" and "CP-Net Interface (Tel)". These blocks are responsible for managing call processing and network interfacing procedures, respectively.

Call processing tasks include call setup, resource holding time, and call teardown of both incoming and outgoing calls. Note that many such calls may need to be carried out simultaneously. Network interfacing must be performed by the network-end of

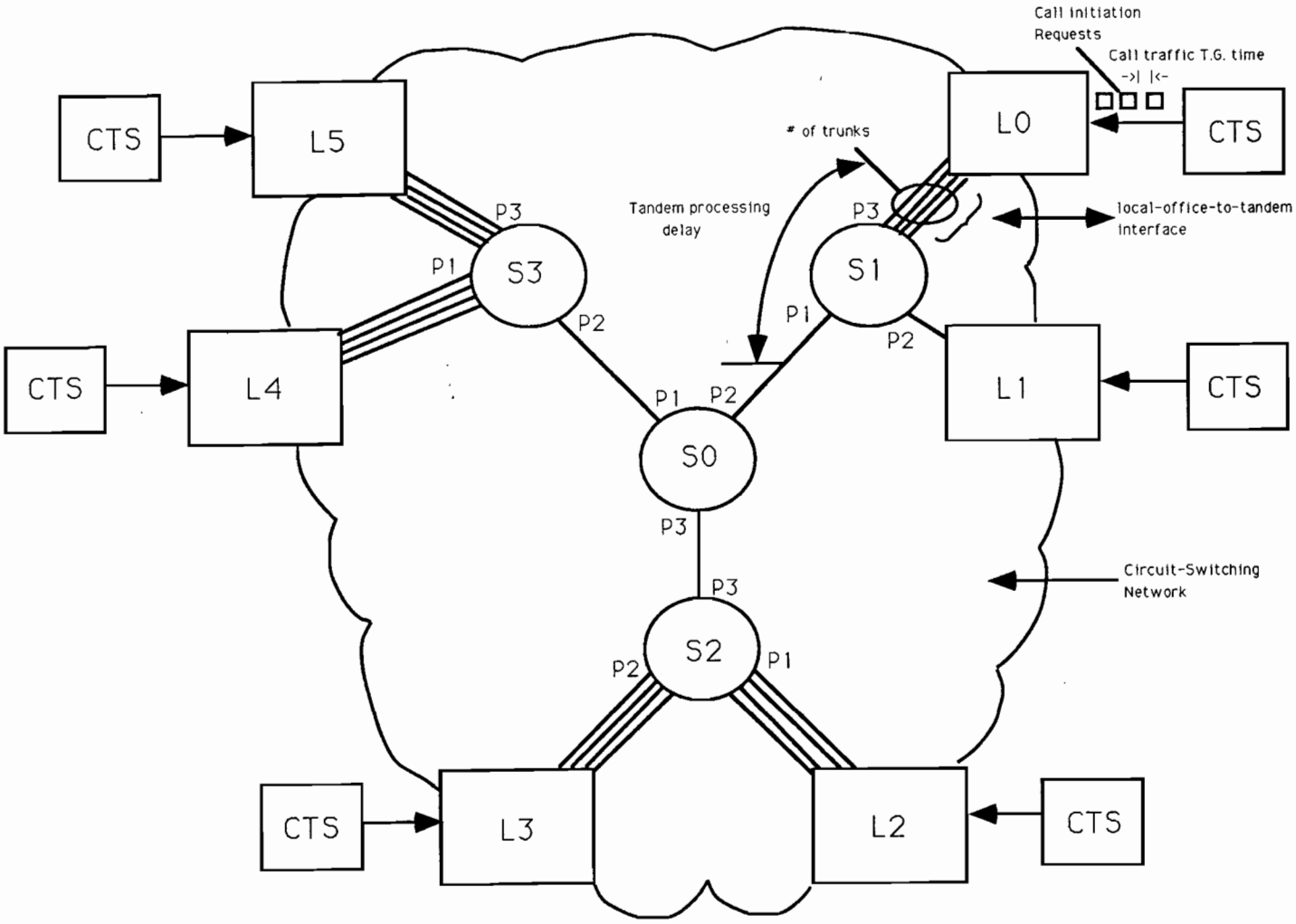



Figure 2.1 Abstract View of the System's Level Model.

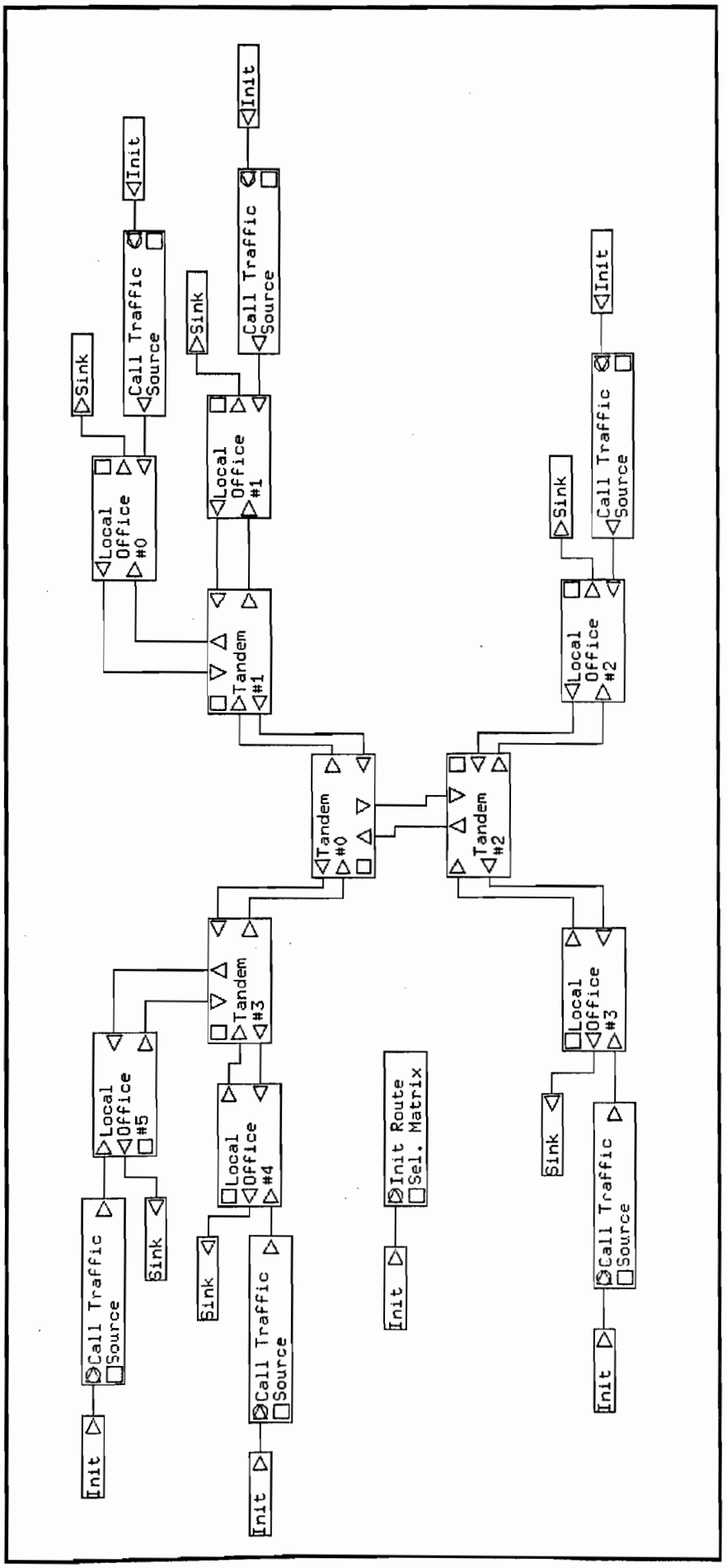



Figure 2.2 Top Level Diagram of the Telephone System.

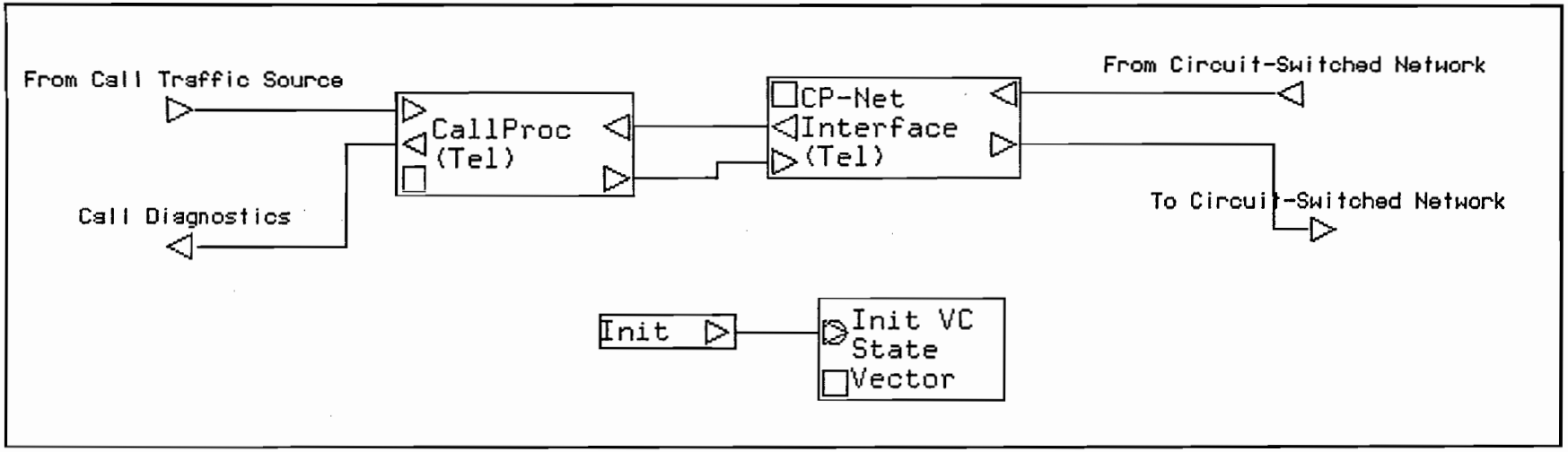



Figure 2.3 The "Local Office" Module.

the local-office so that call-processing procedures can be made compatible with the network's internal conventions. Interfacing tasks include data structure encapsulation/de-encapsulation, routing assignments, and trunk allocation/de-allocation.

Communication between the call processor and the call-processor-to-network interface takes place via a number of logical "channels" (trunks), which are allocated on a per-call basis. Thus, another important task of the network interface is to properly map such trunks into incoming/outgoing calls. The number of trunks available to a local office is a user-defined parameter.

### 2.2.3 Tandems

Tandems have been described earlier as being two-way, three-port units with non-blocking capabilities. These modules are responsible for routing the call setup and teardown signals to the proper destinations. They must also ensure that a connection will remain physically active until its corresponding call is cleared.

An important user-defined parameter exported by the "Tandem" module is its processing delay. This allows the user to control the speed at which protocol signals are piped through the network.

## 3. Implementation Description

### 3.1 Introduction

A considerable number of objects used in the BONeS implementation of telephone model have been either derived or directly re-used from a previous model, the BONeS model for an X.25 network [2].

This section shall concentrate both on the changes and enhancements made to X.25 objects, and on the objects constructed specifically for the telephone system.

Note: familiarity with the previous modeling work [2] is assumed throughout this section.

### 3.2 Objects of the Telephone System

Figure 3.1 depicts the module tree for the telephone system. The tree only shows modules that were either specifically created or altered (out of the X.25 model) for integration with the telephone system. Altered X.25 modules appear labeled with the "(Tel)" suffix. An exception to this is the presence of the "Call Traffic Source" and "Tandem" nodes in the module tree. These objects correspond to unaltered (though relabeled) instances of X.25's "File Transfer Requestor" and "Dumb Switch" modules, respectively.

A total of twelve modules were specifically created for the telephone-system. Additionally, nine X.25 modules were altered/enhanced for integration purposes. A total of eight BONeS databases were used to store user-defined objects and provide library primitives to the model. Database usage is depicted on Table 3.1.

**Table 3.1 - Database Usage for the Telephone Model**

| Database name | Contents |
|---|---|
| DS-base-types | Built-in BONeS data structures. |
| MOD-primitives | BONeS module primitives. |
| MOD-utilities | BONeS utility modules. |
| DS'S In Call Setup | Data structures from the X.25 model. |
| DTE | X.25 modules (call processing portion). |
| Packet Switch Net | X.25 modules (switching network portion). |
| Telephone Model | Modules specifically created for the telephone model. |
| Telephone Sims | Simulation definitions and plots from the telephone model. |

The work associated with constructing, developing, and integrating telephone-model objects into top-level systems is described next.

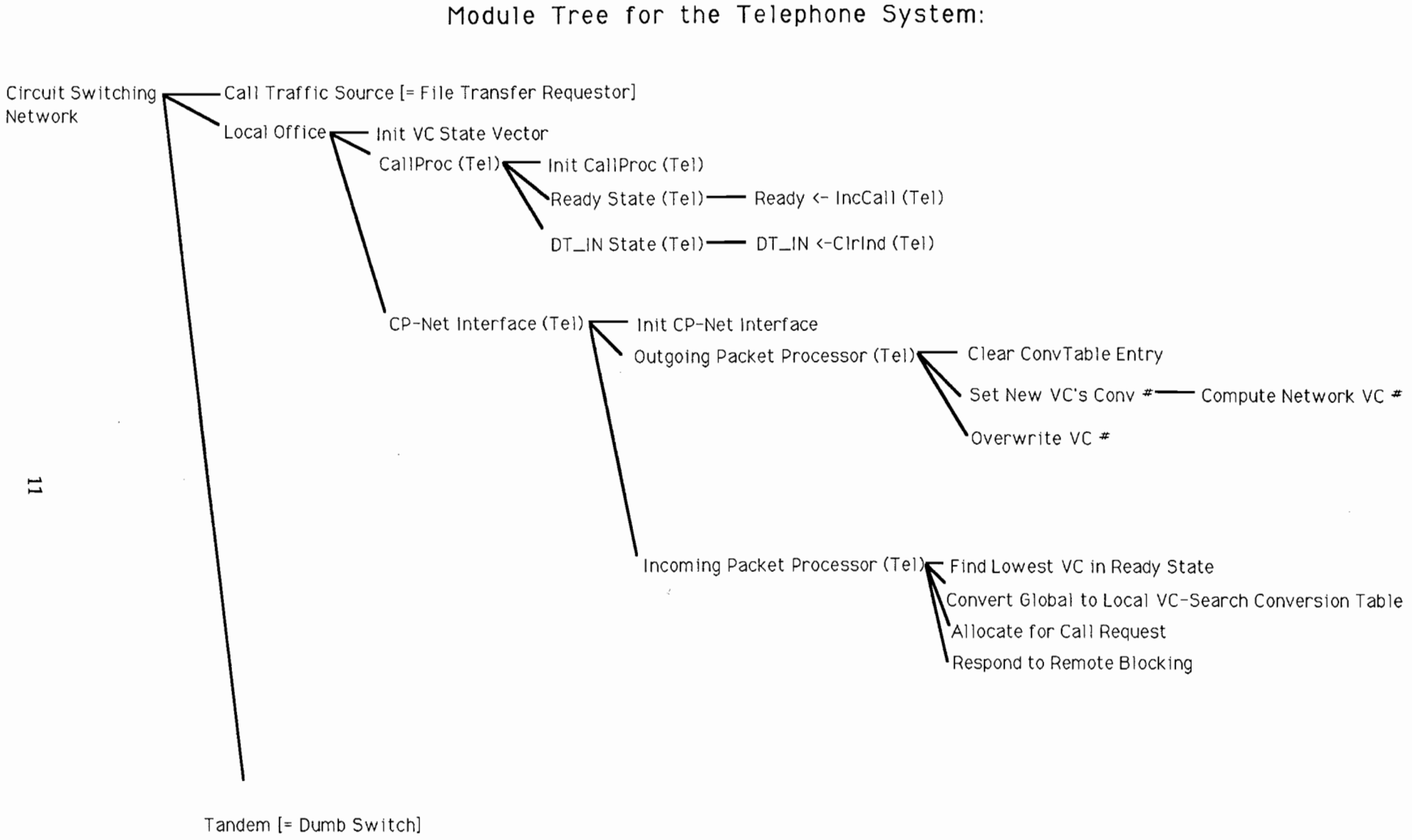



Figure 3.1 Module Tree for the Telephone System.

### 3.3 Top-Level Diagram

The top-level block diagram of the telephone system is shown in Figure 3.2. This diagram reflects the abstract model proposed by Figure 2.1. Namely, six local offices receive call traffic from six independent call traffic sources. Communication between local offices is established over a network of four circuit-switching tandems.

### 3.4 Call Traffic Generation

The first of Figure 3.2's elements to be analyzed is "Call Traffic Source". The function of this module is to generate a stream of call-initiation requests (CIReq's) originated from a large population of telephones.

Each CIReq must contain the following relevant information: address of the call-originating local-office, address of the called local-office, and the time at which the request was issued.

The "File Transfer Requestor" (F.T.R.) module from the X.25 model was used for the above purposes. This module generates a stream of file transfer requests (FTReq). As shown on Table 3.2, FTReq's can accommodate all the information needed for CIReq's.

**Table 3.2 - FTReq-to-CIReq Field Reassignment**

| FTReq Field Name | CIReq Interpretation | Comments |
|---|---|---|
| Time Issued | Same | used for stat. gathering |
| Time Completed | Same | used for stat. gathering |
| Reject Status | Same | used for stat. gathering |
| Sender | Call Initiating L.O. | 0 to 5 range |
| Receiver | Destination L.O. | 0 to 5 range |
| File Length | Not used | set to zero (see sec. 3.6) |

Note: The "Time Completed" and "Reject Status" fields of an FTReq are initialized by the call processing block when a call terminates; at this point, the call-originating FTReq is returned to the sender for statistical gathering purposes [2]. Also, the "FTReq Intergeneration Time" parameter exported by an F.T.R. is used to control the rate at which call traffic is generated.

### 3.5 The Circuit-Switching Network

The network topology proposed for the current model is shown in Figure 3.3. Four switches are connected to six local-offices. Local offices are assigned addresses 0 through 5, clockwise around the network; switches are labeled "S0" through "S3" -- ports in a switch are labeled "P1" through "P3".

As in the X.25 model [2], a system of routes between end-nodes (in this case, 6 local

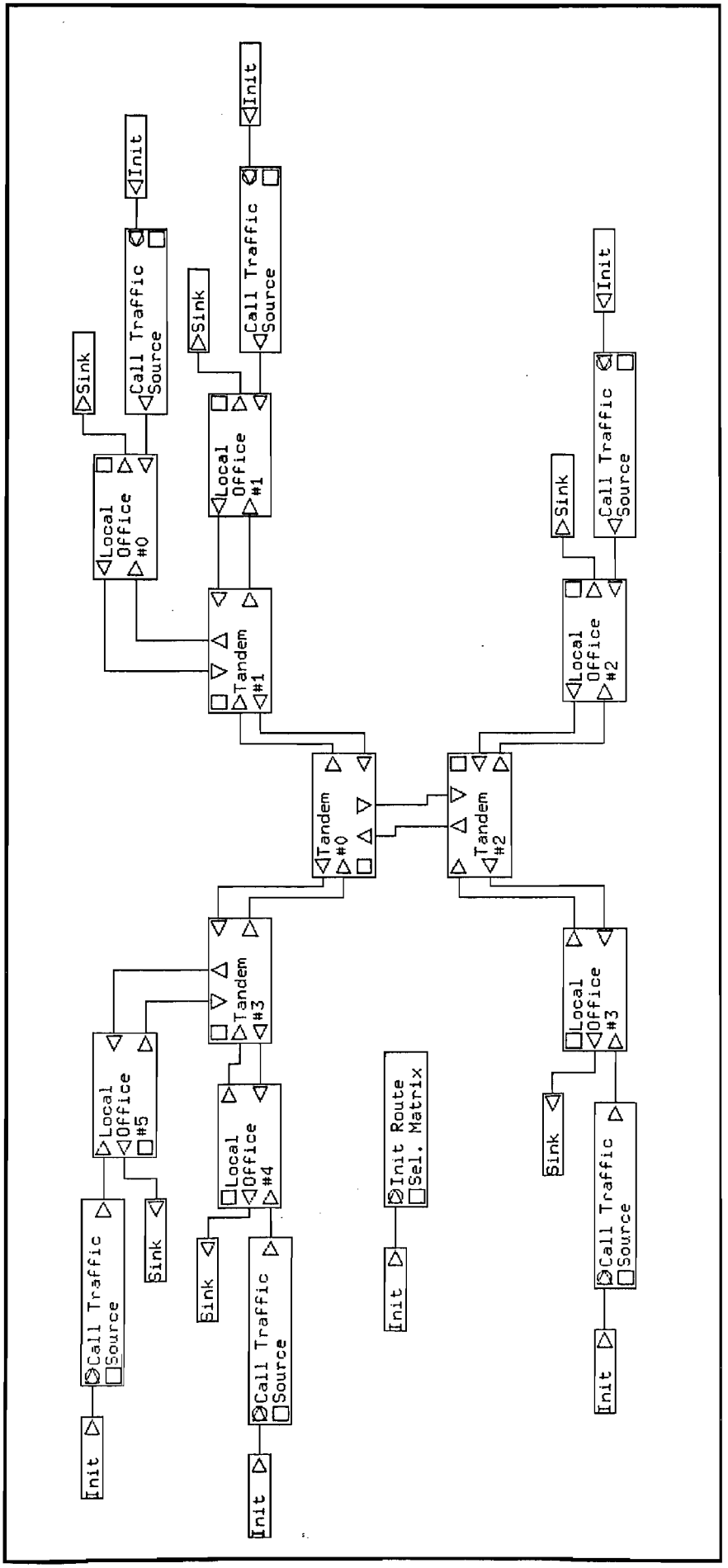



Figure 3.2 The "Circuit-Switching Network" System.

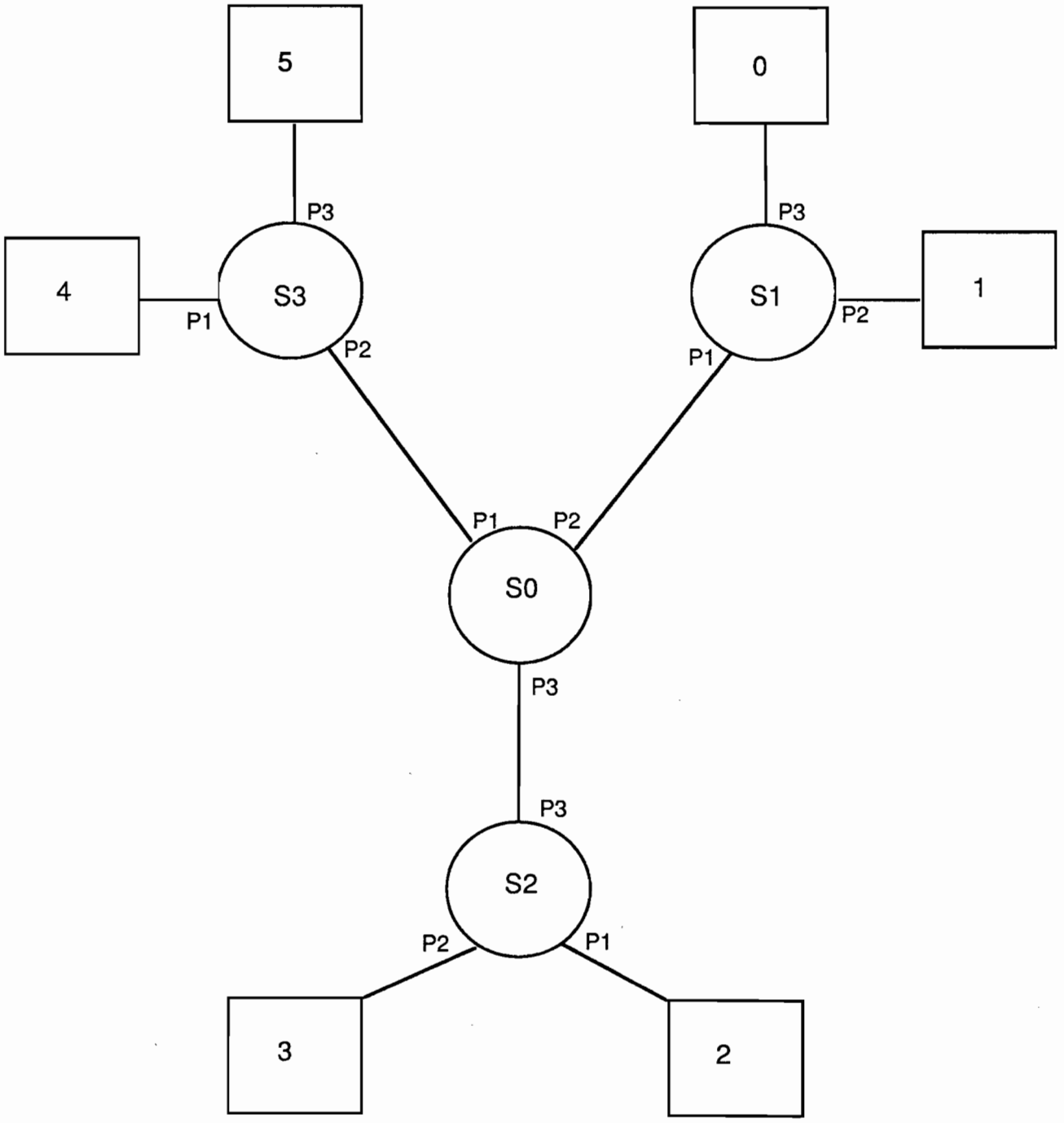



Figure 3.3 A View of the Network Topology.

offices) is constructed. A total of 6*(6-1)/2 = 15 distinct route numbers is needed for the present case. These are assigned as shown on Table 3.3.

**Table 3.3 - Route # Assignment on the 6-Local-Office Network**

| End-Node Pair | Route # | End-Node Pair | Route # |
|---|---|---|---|
| 0-1 | 0 | 1-5 | 8 |
| 0-2 | 1 | 2-3 | 9 |
| 0-3 | 2 | 2-4 | 10 |
| 0-4 | 3 | 2-5 | 11 |
| 0-5 | 4 | 3-4 | 12 |
| 1-2 | 5 | 3-5 | 13 |
| 1-3 | 6 | 4-5 | 14 |
| 1-4 | 7 | | |

Note: a Route Selection Matrix, shown in Appendix B, is used to provide route numbering information to the appropriate modules [2].

Routes are associated with the (typically unique) shortest path between two end nodes. For example, a signal going from local-office-3 to local-office-1 (i.e., route # 6) would flow as follows:

from local-office-3 to switch-2's port-2;
from switch-2's port-3 to switch-0's port-3;
from switch-0's port-2 to switch-1's port-1;
from switch-1's port-2 to local-office-1.

X.25's "dumb switch" module was used as the switching element. Dumb switches were relabeled as "Tandems" at system's level diagrams of the telephone model. As noted in [2], dumb switches require individual 3-row routing matrices. The number of columns in such matrices is given by the number of routes in the system. For the present network, a total of four (one per switch), 3 x 15 matrices is required. These matrices are shown in Appendix A. The algorithm to generate a routing matrix is described in [2].

Switches in the network are characterized by their processing time. This parameter is exported by the "dumb switch" modules and is of easy definition at system's level simulations.

### 3.6 Local Office Functions

The last functional unit to be analyzed in this section is the local office. An expansion of this module is shown in Figure 3.4. The two major functions performed by this module are call processing and network interfacing. These

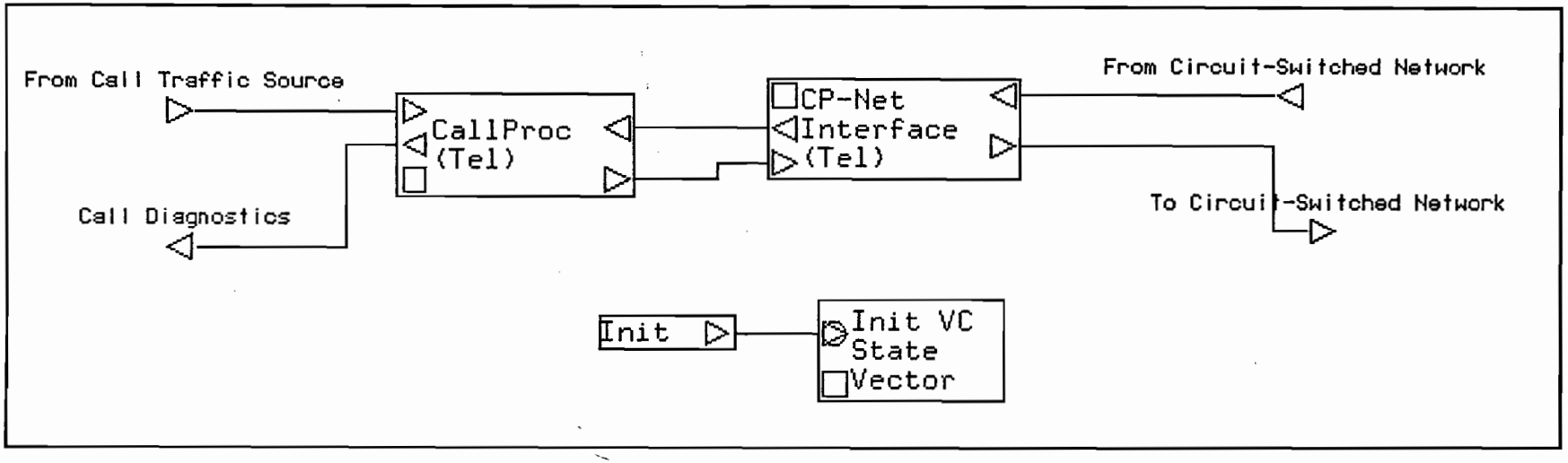



Figure 3.4 The "Local Office" Module.

functions are performed by the "CallProc (Tel)" and "CP-Net Interface (Tel)" modules, respectively (see Figure 3.4).

The component modules of a local-office operate on the contents of a common structure, the "VC State Vector". This keeps track of the state of all the calls active at the particular local office.

Note: The concept of a call's "state" is inherited from the X.25 model, in which the call processing unit is modeled after a state machine [2]. Also, in the telephone-system paradigm, entries in the "VC State Vector" correspond to trunks at a particular local office.

The "VC State Vector" is initialized by the "Init VC State Vector" module, appearing in Figure 3.4. This module initializes all the trunks of a particular local office to "idle".

### 3.6.1 Call Processing

The call processing functions of a local office are executed by the "CallProc (Tel)" module, whose expansion is shown in Figure 3.5. This module is almost identical to the "Call Processor" module used in the X.25 model [2]. The same basic principle applies here: call processing is modeled after a state-transition diagram.

The state-transition diagram used by the telephone model is shown in Figure 3.6. This diagram is very similar to the one used for the X.25 protocol [2]; X.25-call-processing states have direct correspondence to states in the telephone model, as shown on Table 3.4.

**Table 3.4 - State Correspondence Between the X.25 and Telephone Models**

| Telephone State | X.25 State |
|---|---|
| Idle | Ready |
| L.O. Waiting | DTE Waiting |
| Hold Resource | DT_Out |
| L.O. Clearing | DTE Clearing |
| Listen | DT_In |

Only a few functional changes were made to the original call processor when adapting it to the telephone model. The first of these changes has to do with the "Hold Resource" to "L.O. Clearing" state transition of Figure 3.6. Unlike X.25, in which a connection is used for data-packet transfer, a telephone call will hold its connection continually, over a period of time. This is accomplished as follows:

(i) Force a zero-byte data transfer -- as described in 3.4, all CIReq's carry a "File Length" field set to zero. This prevents call processor's DT_Out state from generating any data packets.

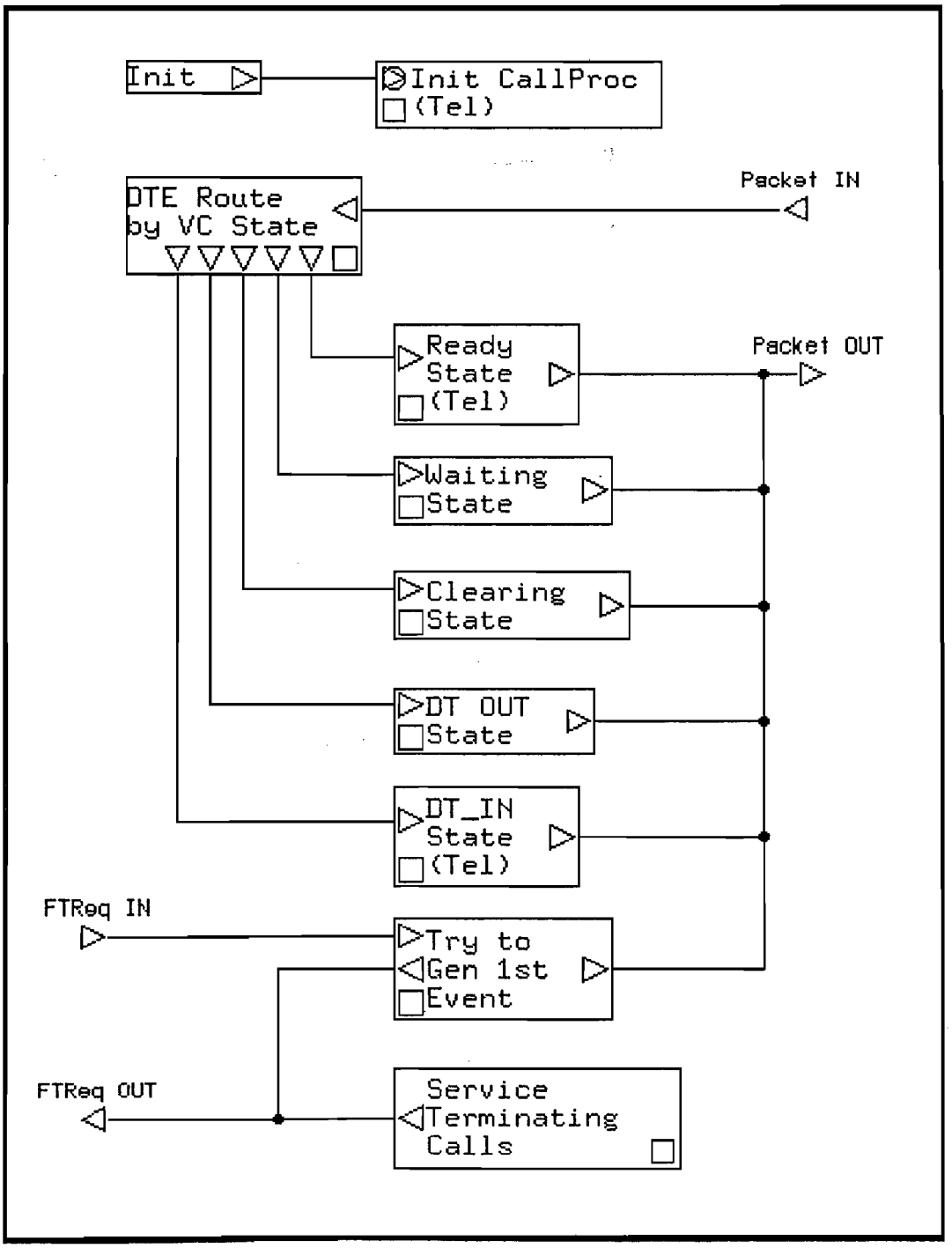



Figure 3.5 The "CallProc (Tel)" Module.

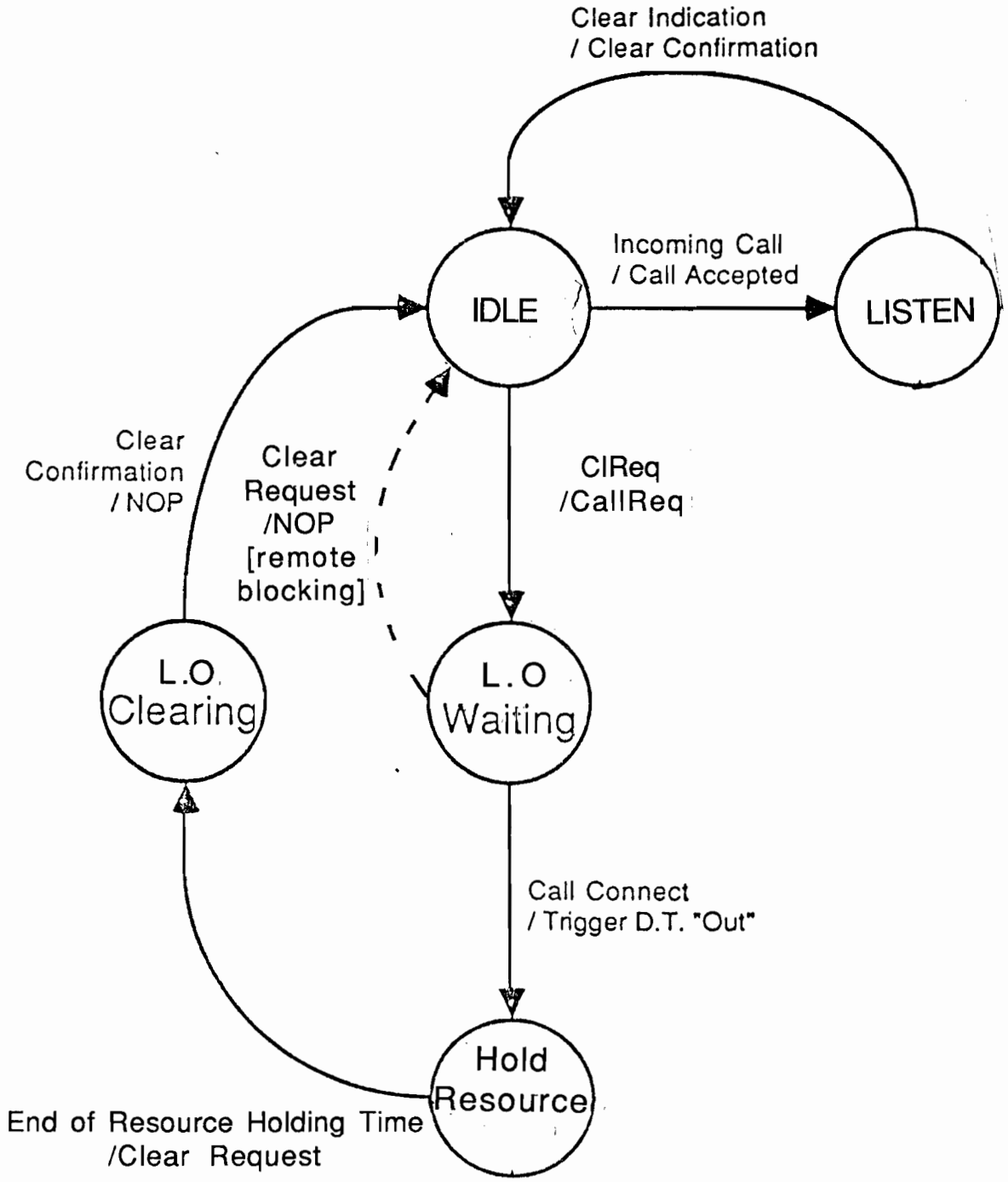



Figure 3.6 State Transition Diagram used by the "CallProc (Tel)" Module.

(ii) Rename DT_Out's data packet intergeneration (I.G) time to "Mean Conversation Time" -- Since no data packets are produced by "DT_Out State", the data packet I.G time will effectively control the delay between entering and leaving the Hold Resource state.

The second change made to the call processor was the removal of all the functions related to the NAIC (Number of Available Incoming Channels) memory. In the X.25 model, NAIC was a single-integer storage used to keep track of how many incoming channels were available at any given time [2]. This enabled the call processor to produce blocking at the receiving end of a call (remote blocking). As it will be further shown, remote blocking becomes a feature of the network interface portion of the local office, and therefore, NAIC is no longer needed.

### 3.6.2 Network Interfacing

Interfacing the network protocol with call processing functions is accomplished by module "CP-Net Interface (Tel)", expanded in Figure 3.7. The interfacing process in the telephone system involves the following tasks:

- Modeling of processing delays associated with transferring packets across the interface;
- DS-type conversions (for BONeS purposes);
- Route # selection & management;
- VC # conversion;
- Interface-level Blocking.

The first three tasks have already been implemented by the X.25's "CP-Net Interface" module. Therefore, the only enhancements needed are the introduction of VC # conversion, and blocking at the interface level.

#### 3.6.2.1 CP-Net Interface Initialization

"CP-Net Interface (Tel)" is initialized by "Init CP-Net Interface", shown in Figure 3.8. Two main structures are used: the route vector and the conversion table, i.e., "ConvTable". Route vector management is fully described in [2]. On the other hand, ConvTable is an added feature used to store VC # conversion information.

VC # conversion allows different local-office-to-network interfaces to refer to ranges of trunk #'s that have local significance only. For instance, local-office-0 may use its trunk-0 for some call "A" while local-office-2 is also using its trunk-0 for a totally independent call, "B".

The concept of locally significant trunk numbers requires a translation scheme between the local interface and the network. This scheme must assign a network-unique identifier (i.e., a "circuit number") to every trunk carrying an active call.

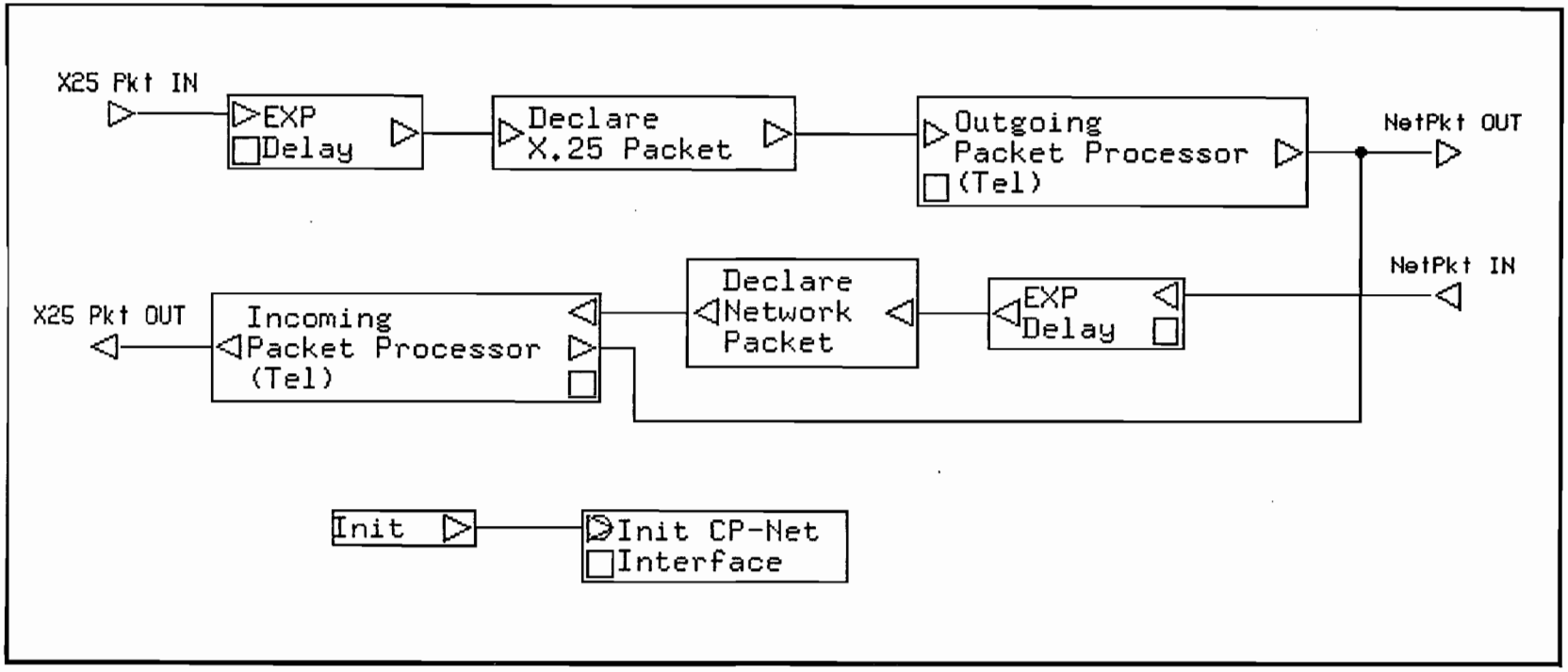



Figure 3.7 The "CP-Net Interface (Tel)" Module.

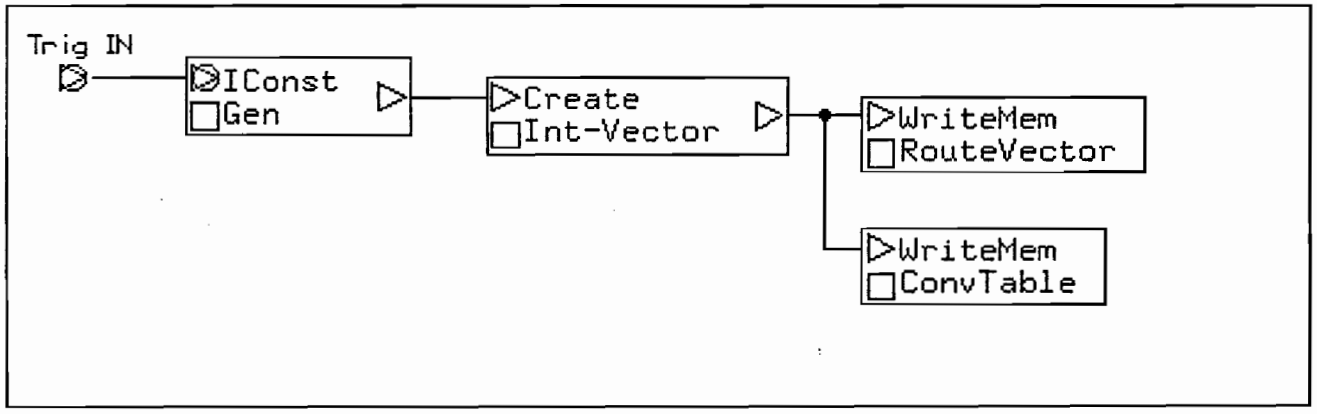



Figure 3.8 The "Init CP-Net Interface" Module.

The ConvTable vector is used for the above purposes. Namely, for an "active" trunk "N", ConvTable[N] will contain the unique circuit number associated with that trunk.

### 3.6.2.2 Outgoing-Packet Processing

The first functional unit in "CP-Net Interface (Tel)" to be analyzed is "Outgoing Packet Processor (Tel)", expanded in Figure 3.9. This module has evolved from its X.25 predecessor, to allow for VC # conversion.

In Figure 3.9, the main input path leads to a packet-router ("DTE Route by Event", [2]) which selects one of three actions depending on the type of the input signal.

Namely, if the input signal is one of the following:

(i) Call request:

Call request signals indicate to the outgoing-call processor that a call is initiating. The call is routed to module "Set New VC's Conv", expanded in Figure 3.10. It is this module's task to come up with a network-unique circuit number and associate it with the incoming call's trunk #.

Note: "X.25 Packet" data structures (and all its children DS's, see [2]) are used to convey call processing signals in the telephone model. The "VC #" field in such DS's is used to convey trunk number information at the local interface, and circuit number information within the network.

Unique circuit numbers are computed by module "Compute Network VC #", expanded in Figure 3.11. The formula used by this module is shown in equation 3.1.

$$global\_VC = local\_VC * l.o's\_in\_network + local\_address \quad [eq.\ 1]$$

where:

global_VC is the network-unique identifier;
local_VC is the original Call Request's VC #;
l.o's_in_network is the total # of local offices in the network;
local_address is the address of the particular local office.

The above procedure guarantees that each local-office-to-network interface will generate a unique set of circuit numbers. This avoids conflicts between calls originated at different local offices.

Once the unique VC # has been computed, it can be stored in ConvTable; this step is performed by module "Set Route Entry (used to set VC Conv)", appearing as one of Figure 3.10's component modules.

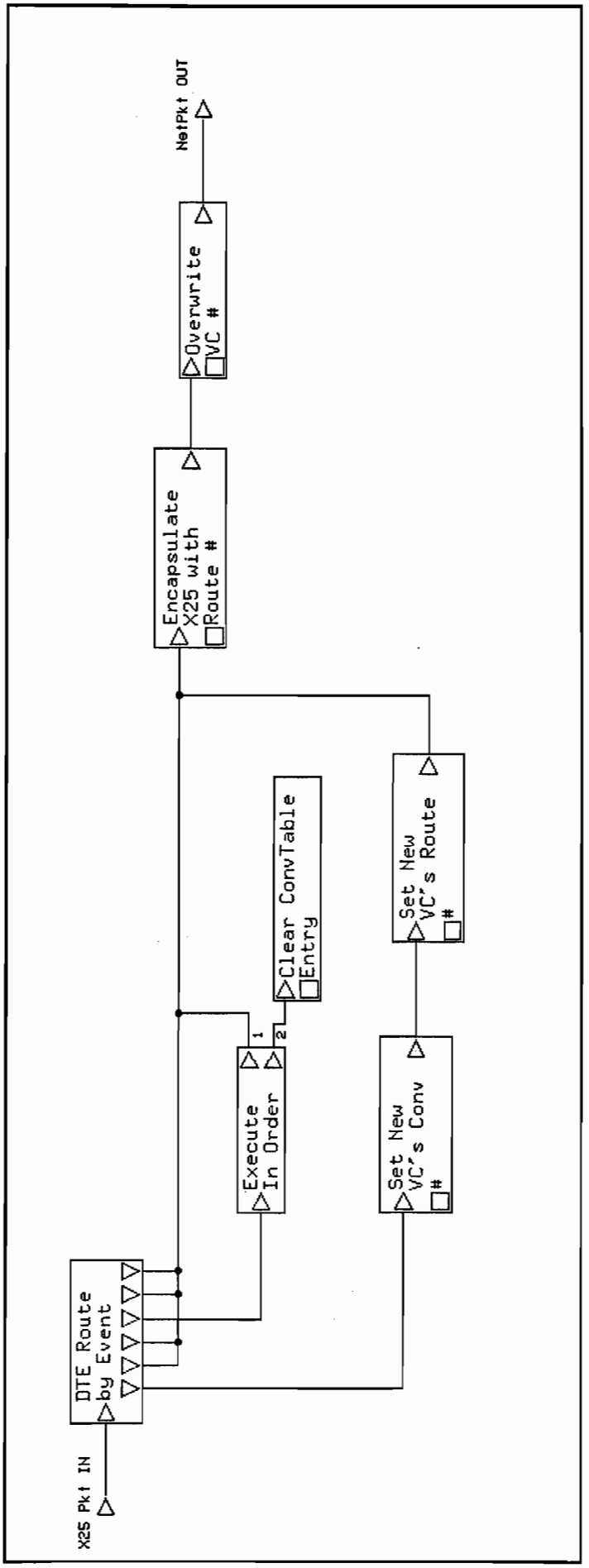



Figure 3.9 The "Outgoing Packet Processor" Module.

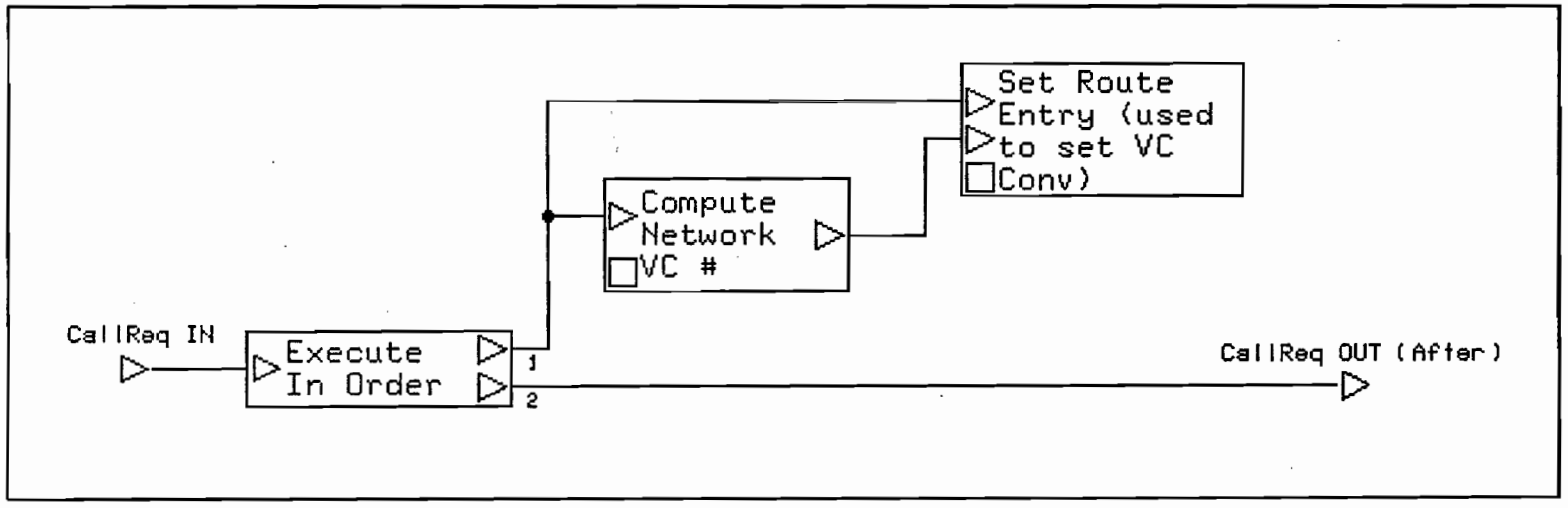



Figure 3.10 The "Set New VC's Conv #" Module.

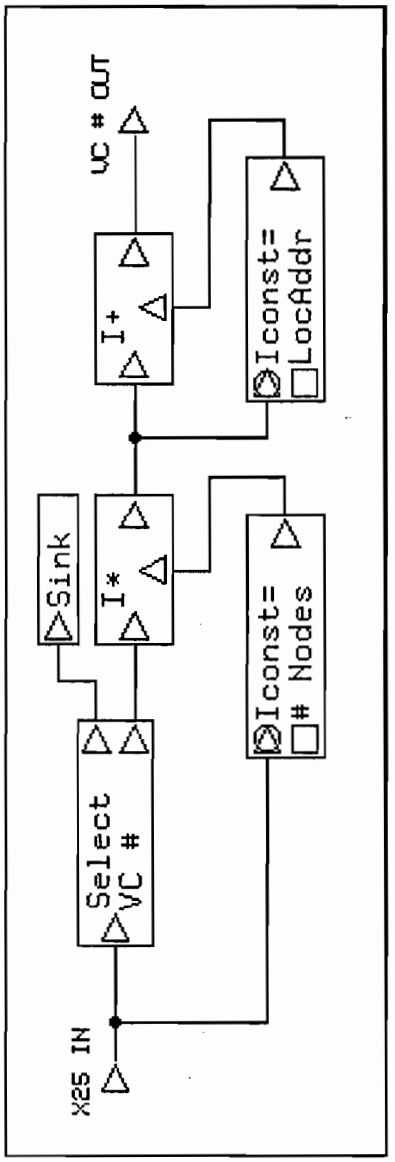



Figure 3.11 The "Compute Network VC #" Module.

In Figure 3.9, the CallReq signals must go through both the "Set New VC's Conv #" and the "Set New VC's Route #" [2] modules. At that point, CallReq is encapsulated with a route # and is further initialized with a global VC #. Note that the packet's original VC # (which has local significance) is overwritten by the global one.

VC # overwriting is done by module "Overwrite VC #", expanded in Figure 3.12. The original VC # is used as an index to ConvTable; the extracted element is used as the new VC # for the output signal.

(ii) Clear confirmation:

If a ClrConf is input to the outgoing call processor (Figure 3.9), the corresponding call is interpreted as terminating, and and its entry on ConvTable must be cleared. Cleared entries on ConvTable have "-1" values. Note that when ConvTable is initialized (see 3.6.2.1), all its entries are set to "-1".

However, before the entry is cleared, the ClrConf signal must be sent into the network through both the route encapsulation and VC overwriting blocks, of Figure 3.9. Note that if the ConvTable entry was cleared prior to the packet's flow through "Overwrite VC #", the latter would have no way of performing the conversion. This explains the suggested order of actions, namely, first convert the VC, and then clear the entry.

The "Clear ConvTable Entry" module performs the function indicated by its name. This module is expanded in Figure 3.13.

(iii) Other packet types

If the packet input to Figure 3.9's outgoing call processor is neither a CallReq nor a ClrConf, it shall simply go through the route encapsulation and VC # overwriting steps, assuming that its particular ConvTable entry is correctly initialized.

### 3.6.2.3 Incoming-Call Processing

The "Incoming Packet Processor (Tel)" has evolved from its X.25-model predecessor. This module is used by the CP-Net interface (Figure 3.7) on the network-to-call-processor direction of flow.

Aside from route # management, the incoming packet processor provides VC conversion services to the particular network-to-local-office interface. Namely, this module is responsible for converting network-unique circuit numbers of incoming packets into locally significant trunk #'s.

Figure 3.14 shows an expansion of "Incoming Packet Processor (Tel)". This module is two-way interfaced to the network (one input and one output), and has one output directed to the call processor. The sequence of operations performed on each

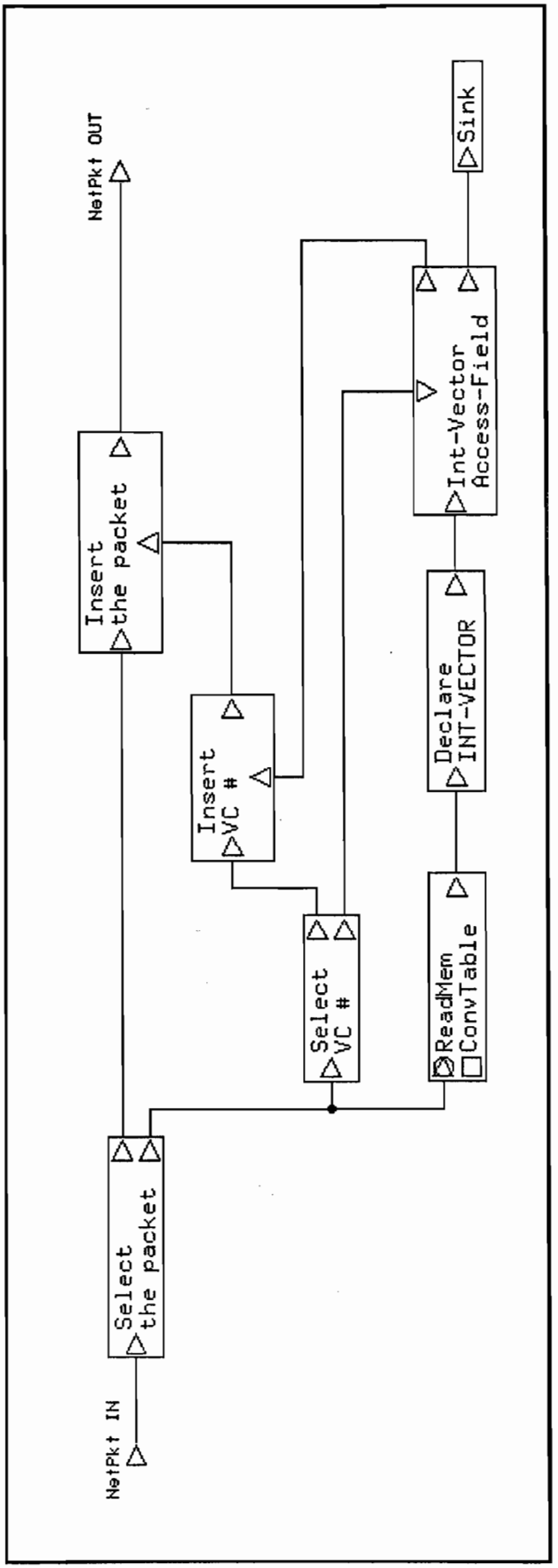



Figure 3.12 The "Overwrite VC #" Module.

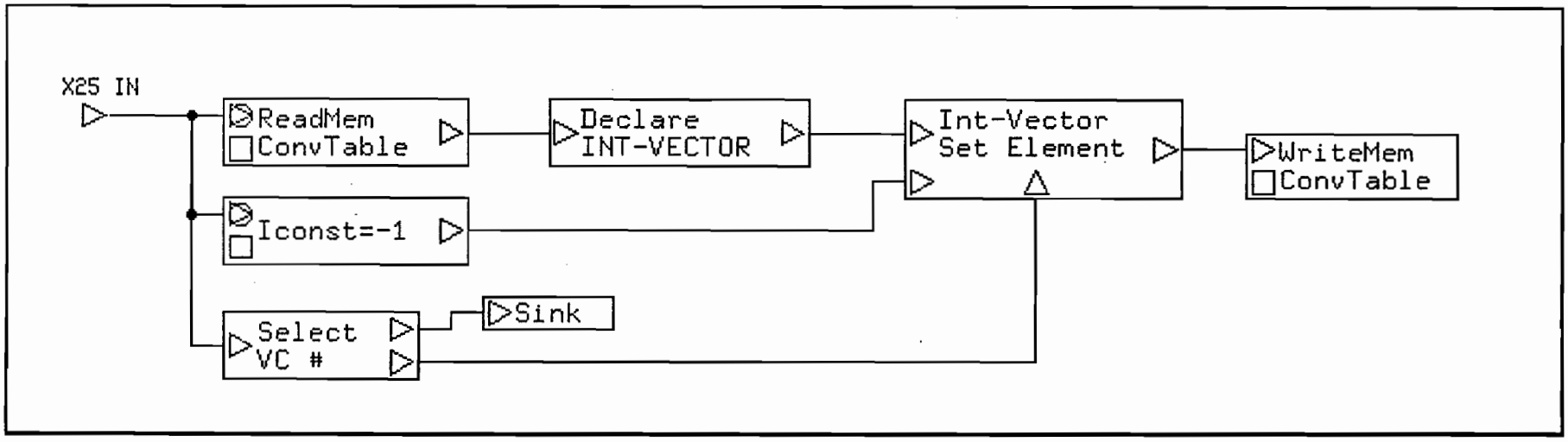



Figure 3.13 The "Clear ConvTable Entry" Module.

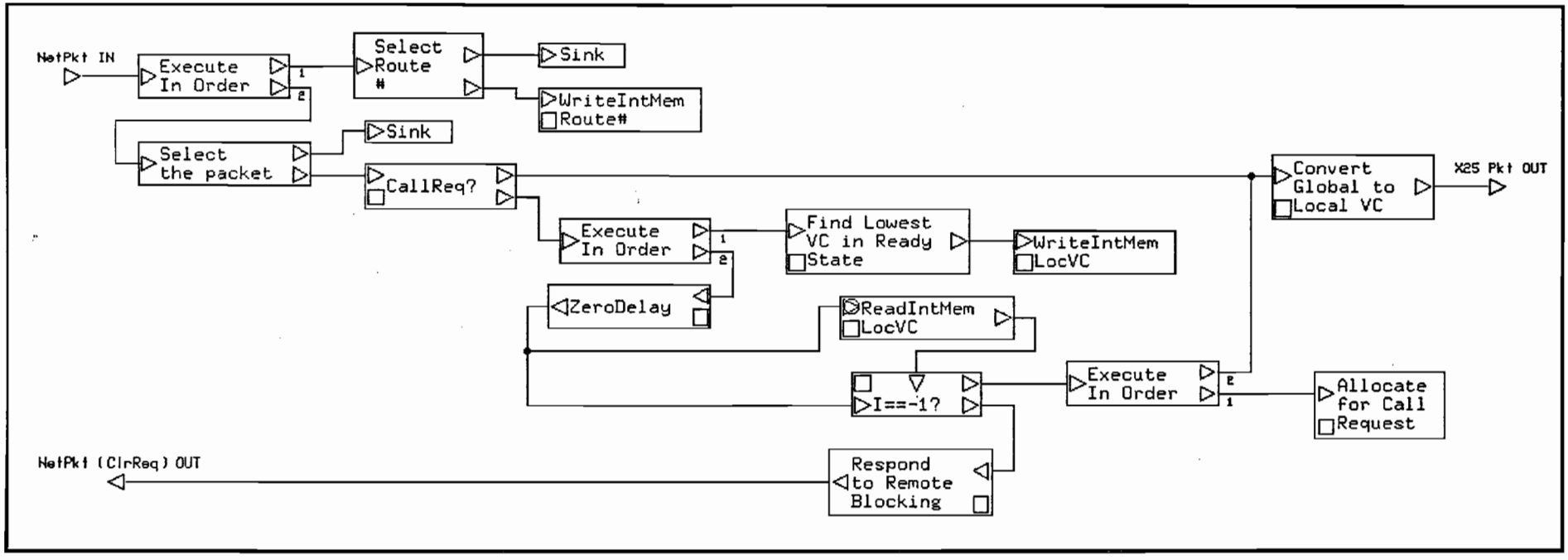



Figure 3.14 The "Incoming Packet Processor (Tel)" Module.

of incoming-call processor's input data structures is explained next.

(i) Save the input call's route #

A dedicated integer storage is used to hold the value of the input call's route # for later use.

(ii) Is the signal a CallReq ?

If the signal is a CallReq, two actions must be taken -- a new local trunk must be allocated for the incoming call, and routing information about the call must be recorded.

A trunk is allocated by finding the lowest trunk # in the "Idle" (i.e., "Ready" in X.25 terms) state. Trunk states are obtained from the VC state vector, physically located at the local-office level (see figure 3.4).

Module "Find Lowest VC in Ready State" is shown in Figure 3.15. If a trunk in the "Idle" state is found, this module returns the trunk #; otherwise a "-1" is returned.

Note: Incoming calls are allocated by the CP-Net interface by choosing the lowest trunk # in the "Idle" state; conversely, outgoing calls are allocated by the call processor by choosing the highest trunk # in the "Idle" state. This algorithm, adopted in most telephone systems, minimizes the potential for call collision and maximizes resource utilization [1].

If a trunk cannot be allocated, i.e., "Find Lowest VC" returns a "-1", module "Respond to Remote Blocking" is activated; the latter is expanded in Figure 3.16. As seen earlier in Figure 3.14, "Respond to Remote Blocking" takes as input a call request and returns to the network a clear request (this corresponds to the call processing specification of a remotely blocked call, see Figure 3.6).

If a trunk in the "Idle" state can be found, module "Allocate for Call Request" is triggered. This module is expanded in Figure 3.17. The route of the inbound signal is read from the local memory mentioned in (i), and is entered into the route # vector; also, that signal's network-unique VC # is entered into the ConvTable vector. Note that both vectors are indexed by the newly allocated trunk #.

(iii) Get local VC

The last of incoming-call processor's operations is to convert the input signal's global circuit number to a local trunk identifier. This is performed by the "Convert Global to Local VC" module, appearing on the far right of Figure 3.14. This module is expanded in Figure 3.18.

"Convert Global to Local VC" takes the input signal's network-unique VC # field,

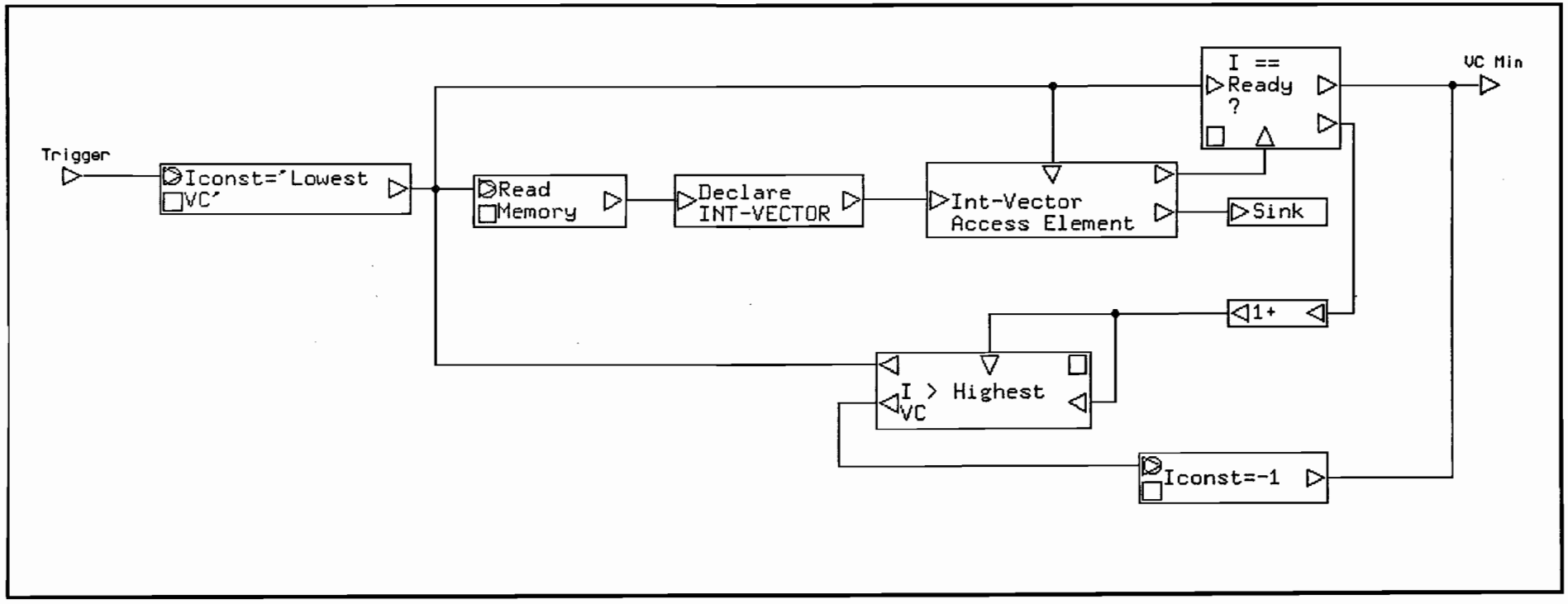



Figure 3.15 The "Find Lowest VC in Ready State" Module.

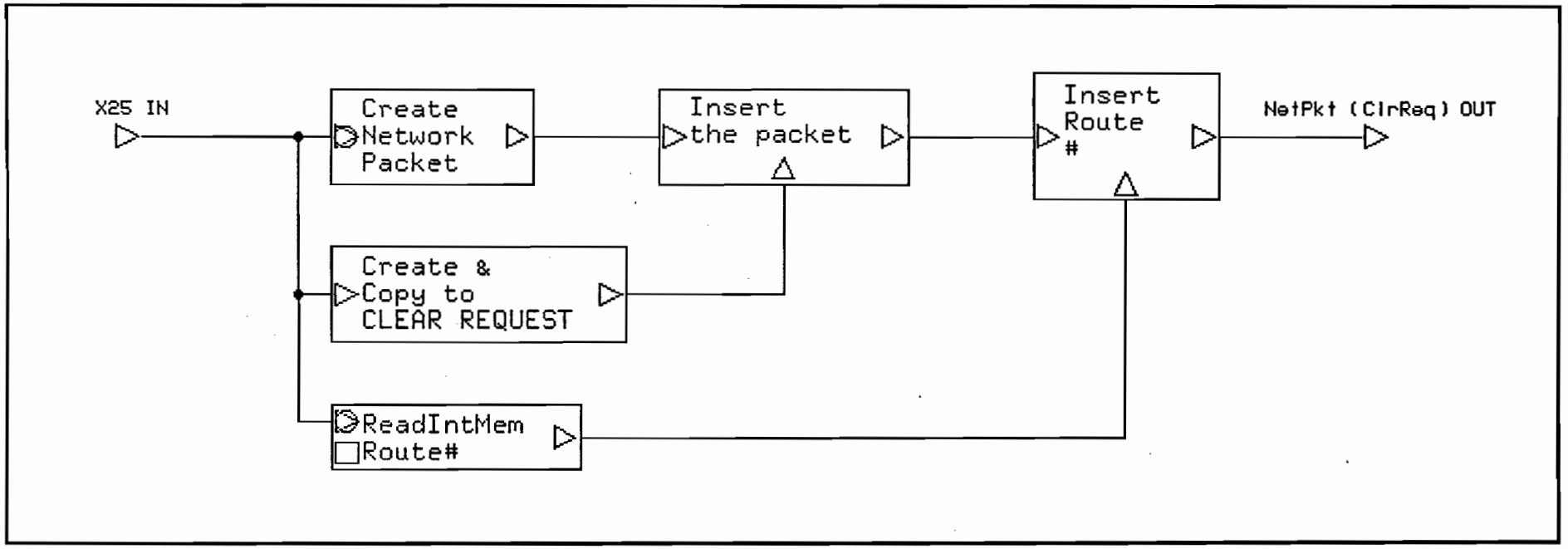



Figure 3.16 The "Respond to Remote Blocking" Module.

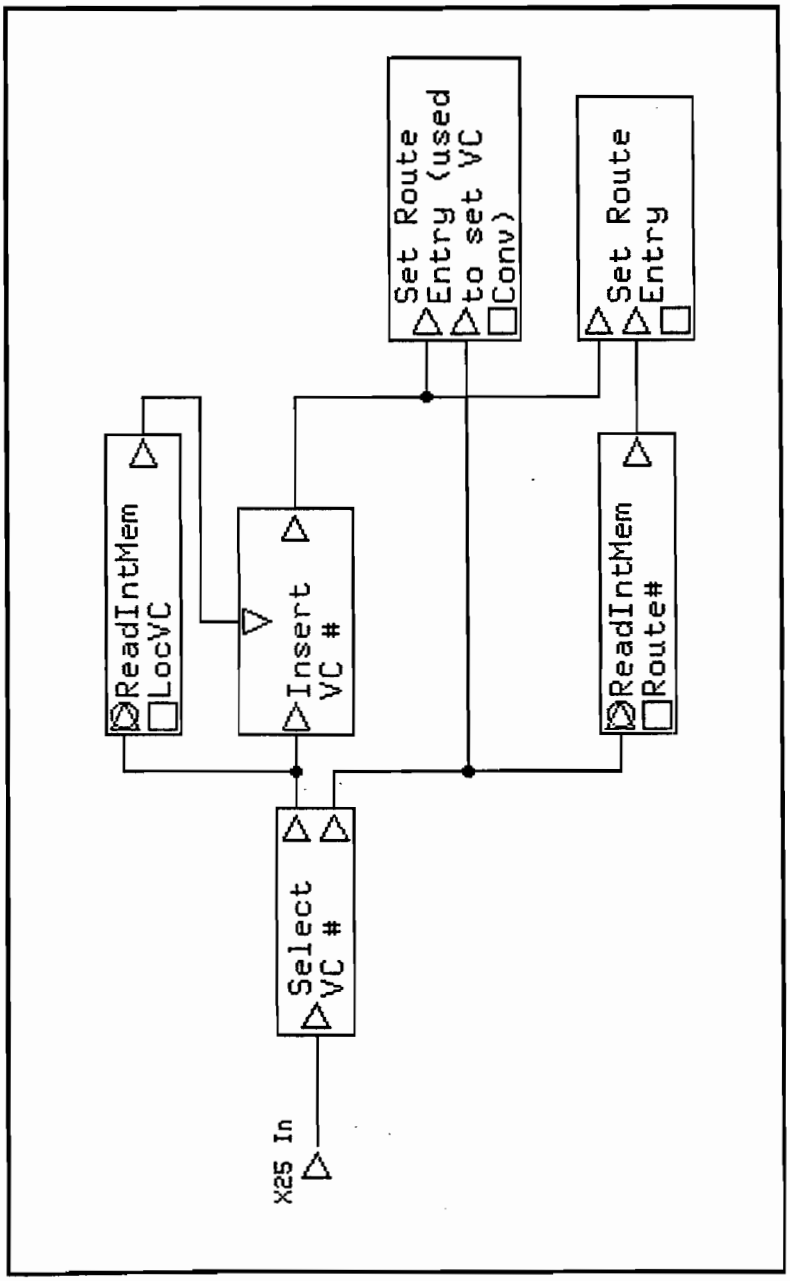



Figure 3.17 The "Allocate for Call Request" Module.

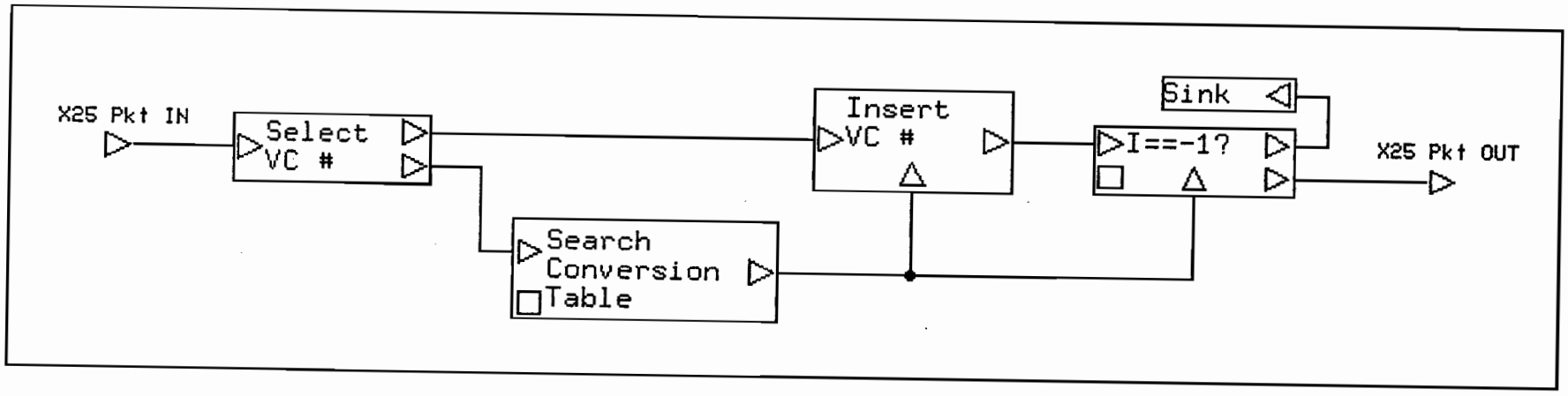



Figure 3.18 The "Convert Global to Local VC" Module.



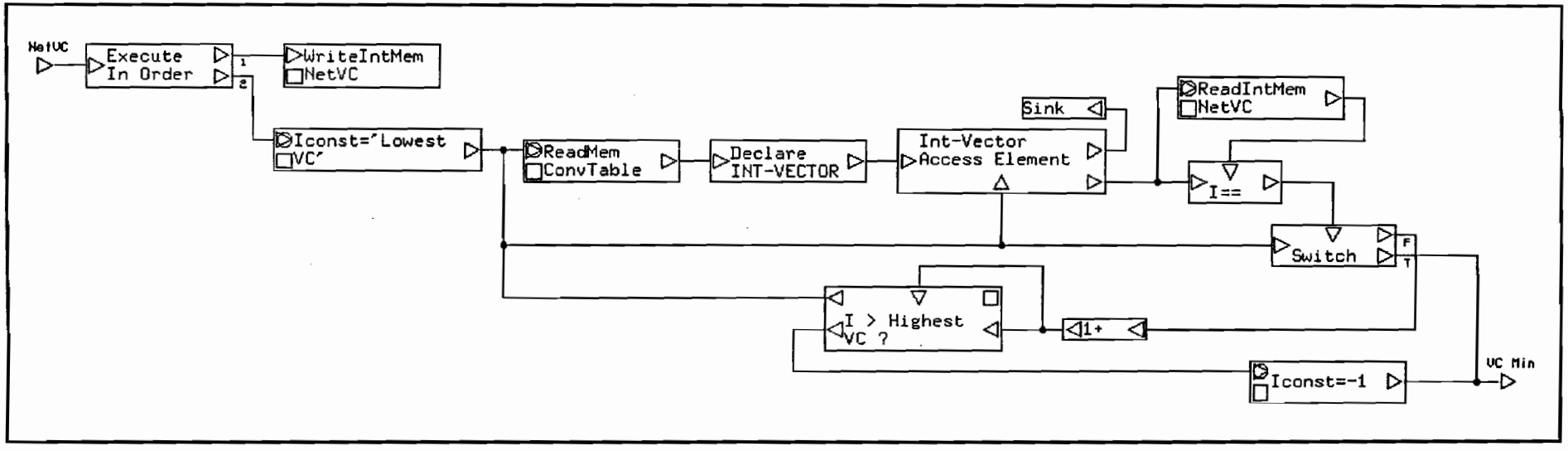



Figure 3.19 The "Search Conversion Table" Module.

and tries to match it with an entry on the ConvTable vector. Module "Search Conversion Table", expanded in Figure 3.19 is used to perform a linear search on ConvTable. This module returns the index at which the match occurred; it may also return a "-1" indicating that that global circuit number does not correspond to any call currently in progress at the particular local office.

Assuming that the search is successful, the resulting trunk number is inserted into the flowing DS's VC #. This field will now carry a locally significant trunk number that can be properly handled by the call processor.

### 3.7 Summary of the BONeS-Implementation of the Telephone System

The key functional units and BONeS objects in the telephone model have been described in this section.

A large number of concepts and objects in the present model have been derived from a previous one, namely, the BONeS model for an X.25 network [2]. X.25 objects have been re-used in the telephone system either unaltered or with small functional enhancements.

# 4. Simulation Systems & Results

## 4.1 Introduction

The integration of section 3's objects leads to the top-level simulation system shown in Figure 4.1. Major functional units in the system include: call traffic sources (CTSs), local offices (L.O's, with addresses 0-5), and tandems (labeled 0-3).

This section shall describe the simulation aspects of the telephone system in the following order: simulation parameters, functional validation, and simulation results.

## 4.2 Simulation Parameters

The system's level simulation diagram (Figure 4.1) has been configured to offer maximum simulation flexibility. Namely, parameters exported by the main functional units have been renamed, altered, and combined to allow most of the results to be extracted from a single, general system.

Figure 4.2 shows the parameter form used for telephone system simulations. The parameters in this form have been divided into six main groups, explained next.

### 4.2.1 Call Directivity

Parameters of the form "L.O # 'N' - Min/Max Called Addr.", where 'N' is a number between 0 and 5, control the "directivity" of outgoing calls. Their Min/Max values indicate to the CTS connected to local office 'N' the range of remote addresses to be used for outgoing calls.

Note: every call generated by a CTS contains a random remote address; addresses are uniformly distributed over the specified Min/Max range; the address generated is always remote, i.e., it is never equal to the local address.

For example, consider the following settings:

"L.O # 3 - Min Call Addr." = 1
"L.O # 3 - Max Call Addr." = 4

The above causes the CTS connected to L.O # 3 to pick random remote addresses from the { 1, 2, 4 } set, i.e., in the 1-4 range, except for the particular local address, which is 3.

Control over call directivity is an important capability in the simulation of unbalanced networks.

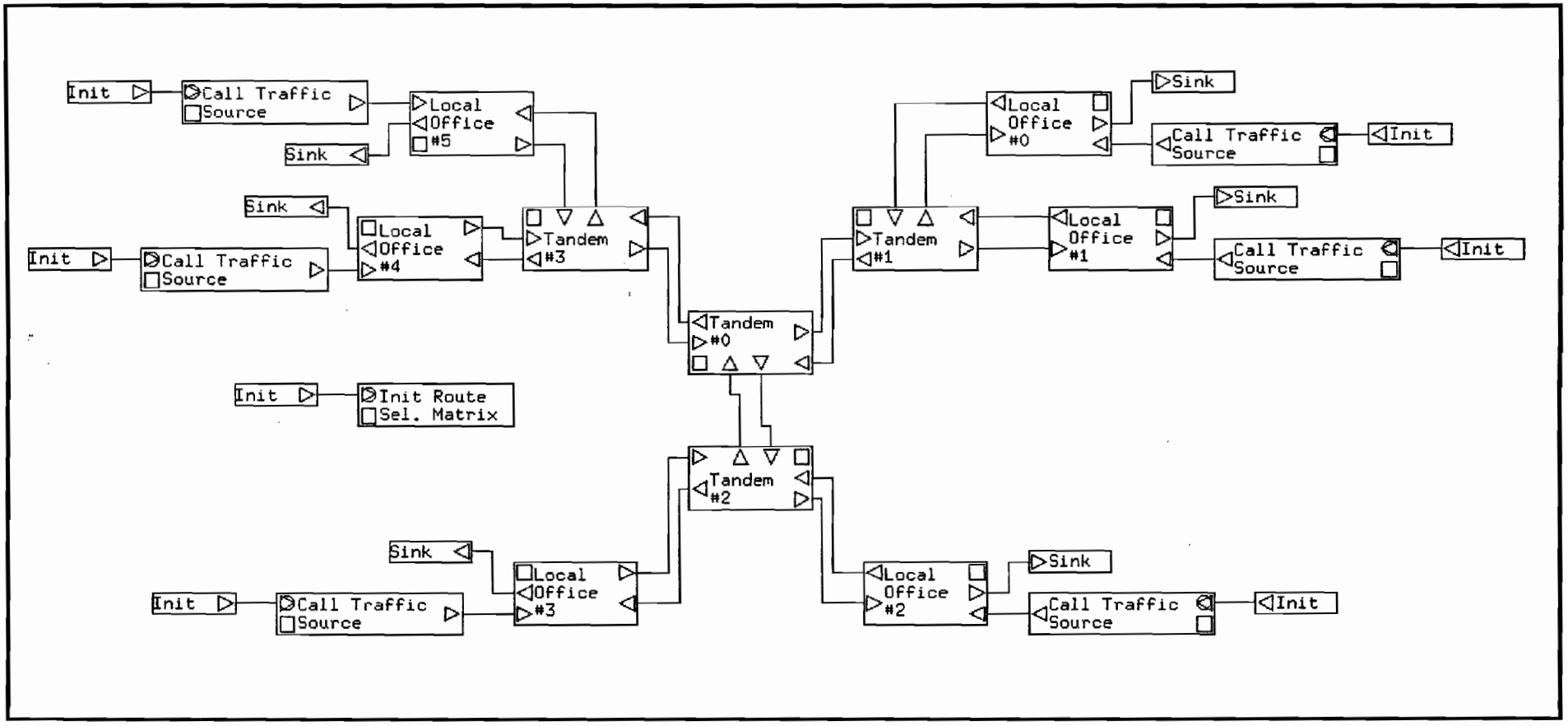



Figure 4.1 Top Level Simulation System for the Telephone Model.

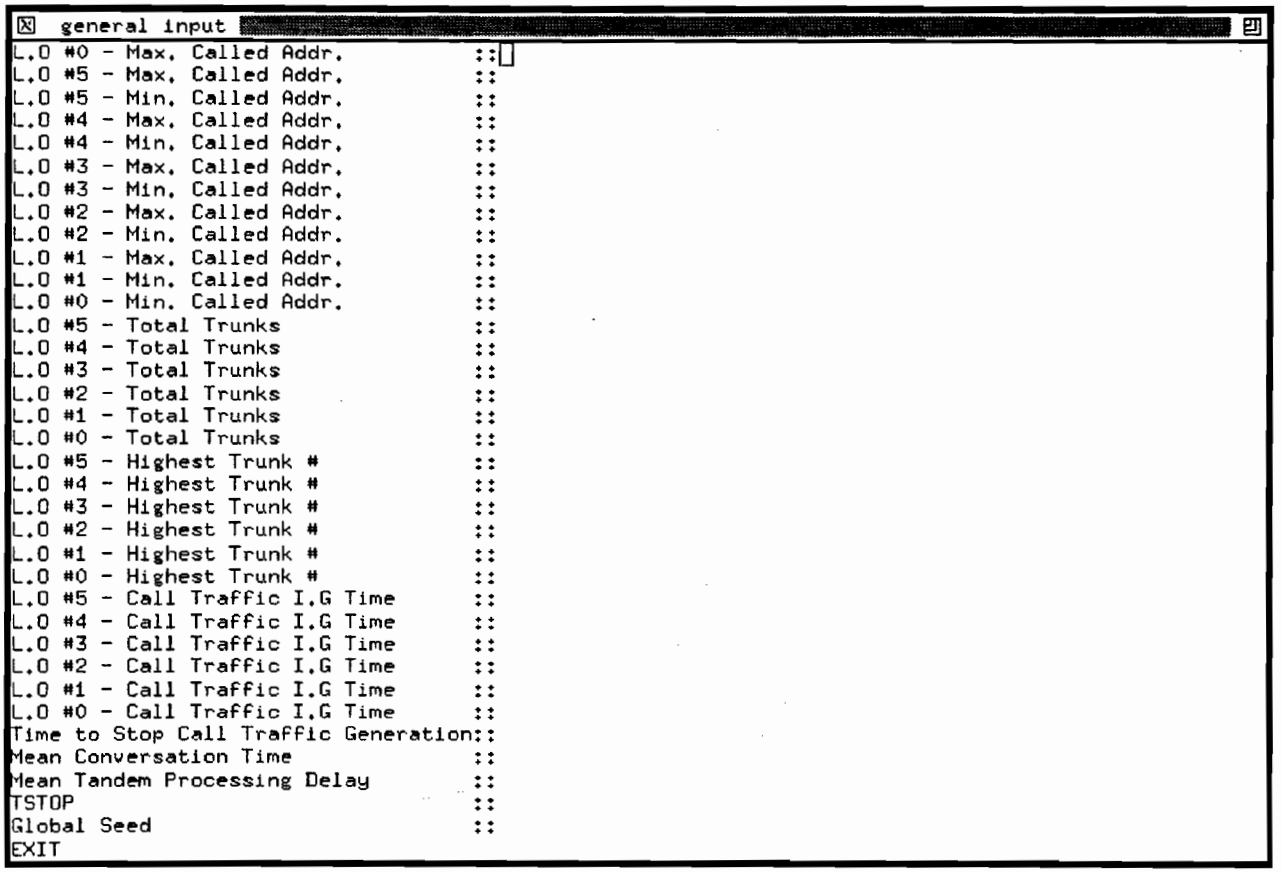



Figure 4.2 Parameter Form for Simulations of the Telephone System.

### 4.2.2 Number of Local Office Trunks

Parameters of the form "L.O # 'N' - Total Trunks", where 'N' is a L.O address, define the total number of trunks available to local office 'N'. In implementation terms, this parameter controls the length of various vectors associated with the "Local Office" module (see section 3).

Trunk availability affects many network performance variables, including blocking ratios, intensity of network traffic, and connection delays.

Note: Parameters labeled "L.O # 'N' - Highest Trunk #" must be set under the assumption that trunk numbers range from 0 to "Total Trunks" - 1. For example, if "L.O # 1 - Total Trunks" is set to 24, "L.O # 1 - Highest Trunk #" must be set to 24 - 1 = 23.

### 4.2.3 Traffic Intergeneration Time

The "L.O # 'N' - Call Traffic I.G Time" parameters control the mean intergeneration (I.G) time of calls generated by a CTS and sent to L.O 'N'.

Traffic I.G time is important for controlling the load offered to a particular local office; this parameter also affects various network performance variables.

### 4.2.4 Conversation Time

Parameter "Mean Conversation Time" defines the average time between the receipt of a "call connected" and the issuing of a "clear request" signal by local offices in the system. In other words, this is the mean resource-holding-time of successfully established connections.

### 4.2.5 Tandem Processing Delay

The "Mean Tandem Processing Delay" parameter defines the delay imposed by a single tandem when forwarding an incoming signal to the appropriate output. Elements of this delay include: queuing, routing decisions, and switch commutation times.

### 4.2.6 Miscellaneous Settings

"TSTOP" is the usual BONeS parameter defining the length of the simulation.

"Time to Stop Call Traffic Generation" is used to interrupt CTS traffic before a simulation is over. To accomplish that, "Time to Stop" must be made less that "TSTOP". This permits the visualization of a "reverse" transient effect on network traffic.

"Global Seed" is used by various BONeS random number generators.

### 4.3 Functional Validation

In order to verify the functionality of the implementation and validate its results, simplified simulations systems were devised. These are described next.

#### 4.3.1 Call Processing

The system shown in Figure 4.1 was configured for the testing of call processing functions. Namely, probes were placed at local-office-0's call-processor-to-CP-Net interface, as shown in Figure 4.3. Traffic levels were kept low for reduced network congestion and low blocking ratios. Local offices were given 24 trunks each.

The call processing signals for the above situation are shown in the plot of Figure 4.4. The X-axis in this plot corresponds to simulation time. However, the Y-axis uses a special formula to convey both trunk # and signal type information. This formula is shown below:

$$\text{Y-value} = \text{"Trunk \#"} * 10 + \text{"Signal Type"} \quad \text{[eq. 4.1]}$$

Note: signals in the telephone module correspond to packets in the X.25 model; each of these packets carry both a "VC #" (used as the "Trunk #") and a "Type" field. Recall that integers are used to represent different signal types [2] as indicated by Table 4.1.

**Table 4.1 - Signal Type Codes [2]**

| Code | Type | Acronym |
|---|---|---|
| 1 | Call Request | CallReq |
| 2 | Call Connected | CallConn |
| 3 | Clear Request | ClrReq |
| 4 | Clear Confirmation | ClrConf |

Figure 4.4 exhibits activity in all of L.O # 0's trunks, i.e., calls are distributed over the full 0*10 to 23*10 Y-value range (trunks 0-23). An interesting "transient" effect can be observed on the left region of this plot (i.e., within the 0.0 to 2.0 units of simulation time). This is an effect of the algorithm for allocating local trunks. As pointed out in section 3, outgoing calls allocate trunks from top to bottom in the trunk-number range, whereas incoming calls allocate trunks in the opposite direction. Therefore, the descending diagonal corresponds to recently initiated outgoing calls; conversely, the ascending diagonal corresponds to trunks allocated to incoming calls. Note that the trend is for these two lines to meet; when they do, blocking starts to occur.

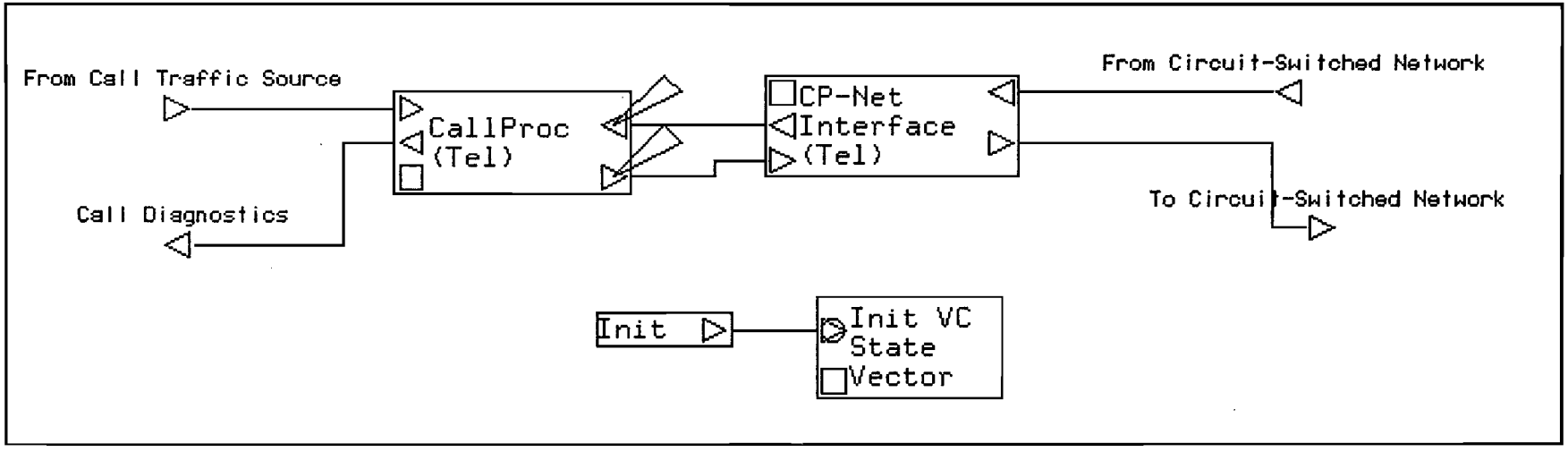



Figure 4.3 Probe Placement for Call Processing Visualization.

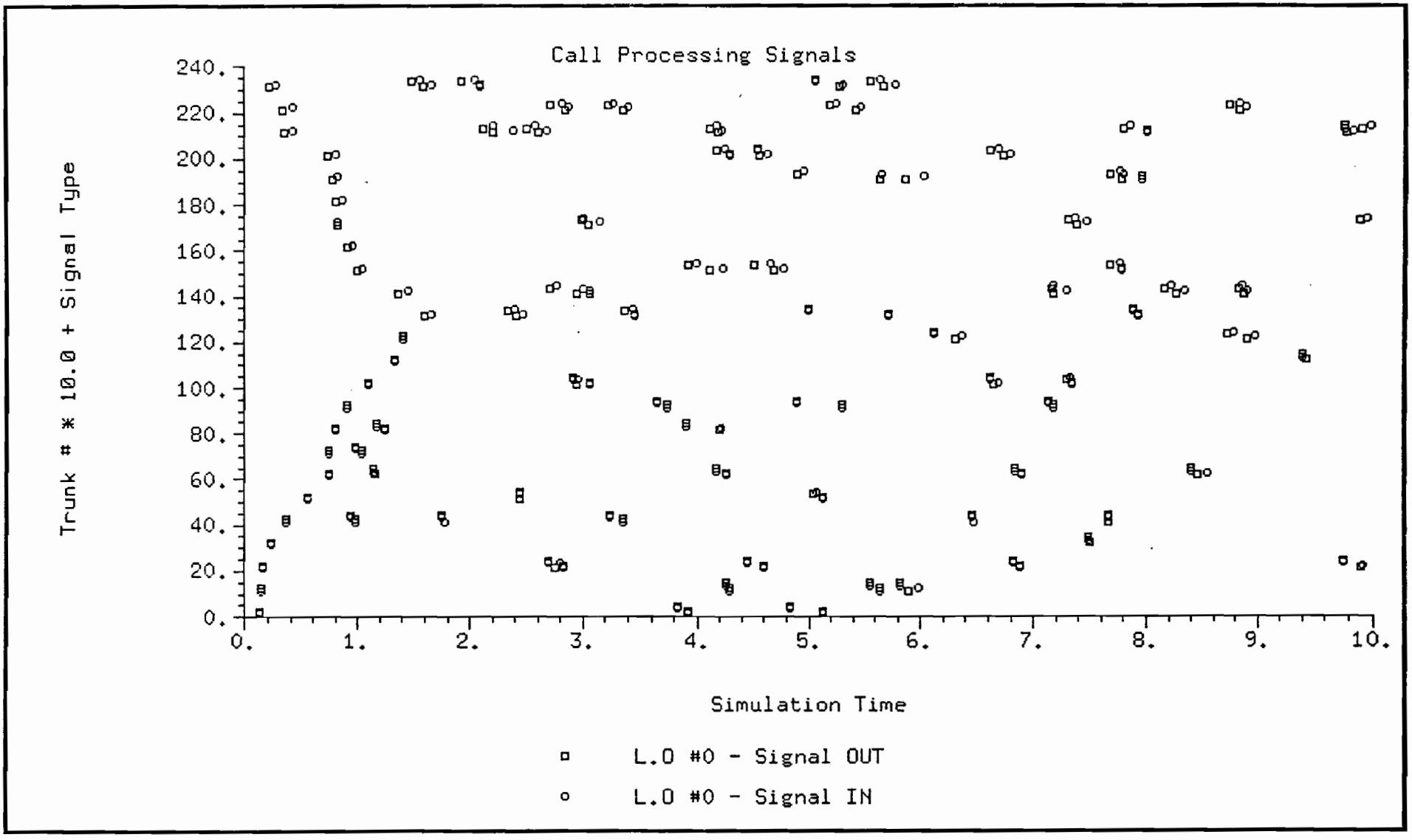



Figure 4.4 Call Processing Signals

For detailed visualization of the call processing steps, a small region of Figure 4.4 is expanded. The expansion is shown in the plot of Figure 4.5. Important features of this plot have been labeled for easy reference throughout this discussion.

Figure 4.5's calls are surrounded by boxes and labeled alphabetically. The first of these calls, "A", appearing on the top-left corner of Figure 4.5, is an outgoing call. The call processing signals for this call appear in the 141-144 Y-value range, i.e., this call is utilizing trunk # 14. The call setup, resource holding time, and call teardown phases for call "A" are illustrated in the figure.

Call "C" is also an outgoing call. The call processing signals for this call are indicated. As shown, these signals flow in the following order: output CallReq, input CallConn, some delay, output ClrReq, and input ClrConf.

On the other hand, Call "D" is incoming, since the CallReq initiating this call is inbound (the plotting symbol is a circle). Note the almost immediate response by the local call processor -- there is virtually no delay between the receipt of the CallReq and the generation of a CallAcc signal. Unlike incoming calls, outgoing calls (e.g., calls "A" and "B"), present some delay between the CallReq and CallAcc packets. This is associated with the time it takes for these signals to traverse the network.

Call "E" is a similar example of an incoming call. This call utilizes trunk # 12. Note the much longer resource holding time for this call (this is set randomly for each call).

Note that there are effectively multiple calls (both incoming and outgoing) taking place concurrently in the system. This is illustrated by, say, calls "B" and "E", which overlap in time. However, for the same physical resource, e.g., trunk # 13, calls cannot overlap, as shown by the "B", "C", and "D" call sequence.

The last commentary on call processing functionality has to do with Figure 4.5's call "F". As shown, this call starts with an output CallReq and terminates almost immediately with an inbound ClrReq. In call processing terms, this corresponds to a remotely blocked call. Note that the resource used by this call, namely, trunk # 14, is re-allocated by another call right after the aforementioned ClrReq is received.

Note: Calls in Figure 4.5 which are surrounded by dashed-line boxes are not shown in their full extension. Typically, only the call setup phase and a portion of the connection time is shown for these calls.

### 4.3.2 Blocking Probability Validation

A second simulation system was devised for validation purposes, as shown in Figure 4.6. Differences between this system and the one in Figure 4.1 include: local offices 3 and 4, as well as their CTSs, have been removed; local offices 2 and 5 are

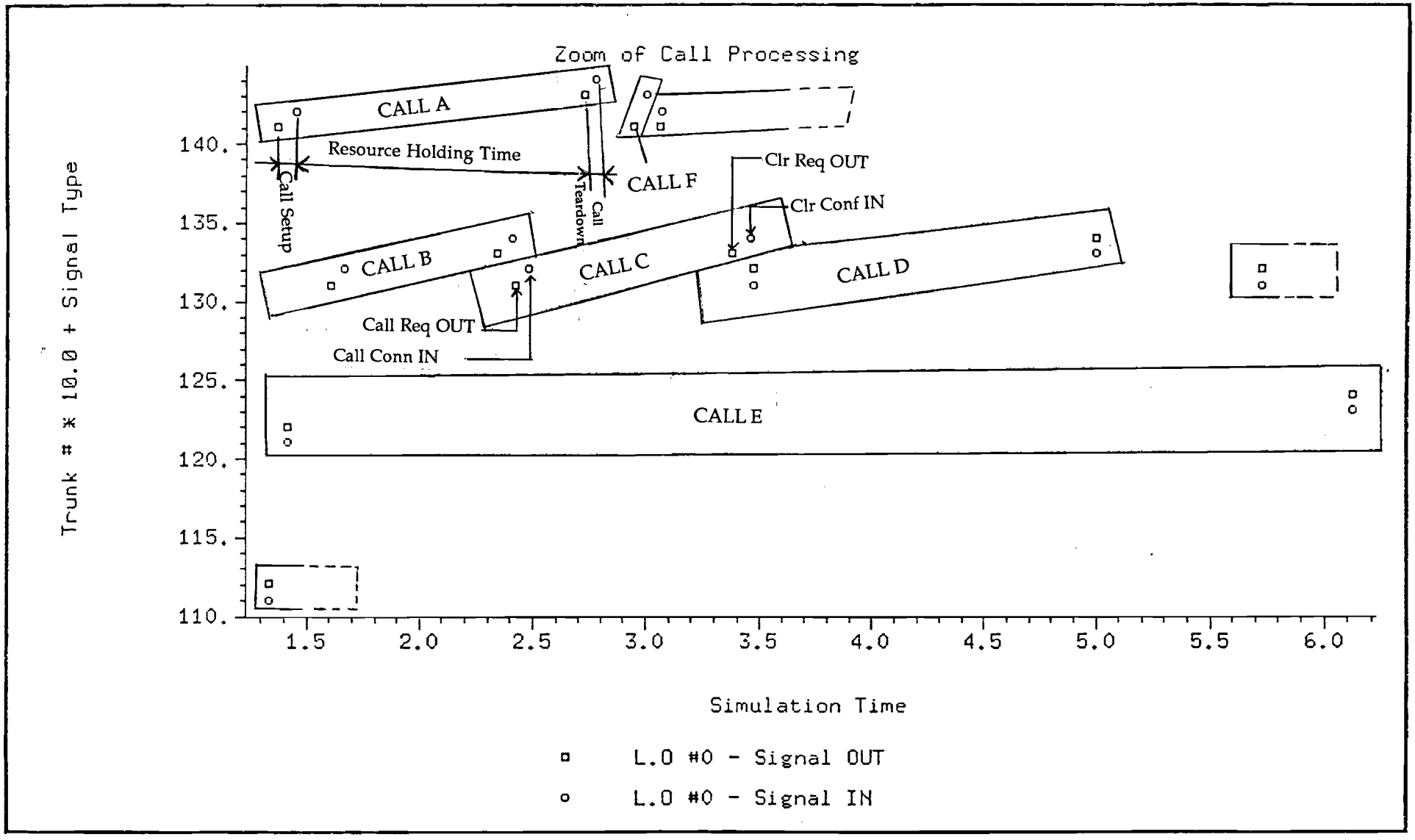



Figure 4.5 Detail of Call Processing Signals.

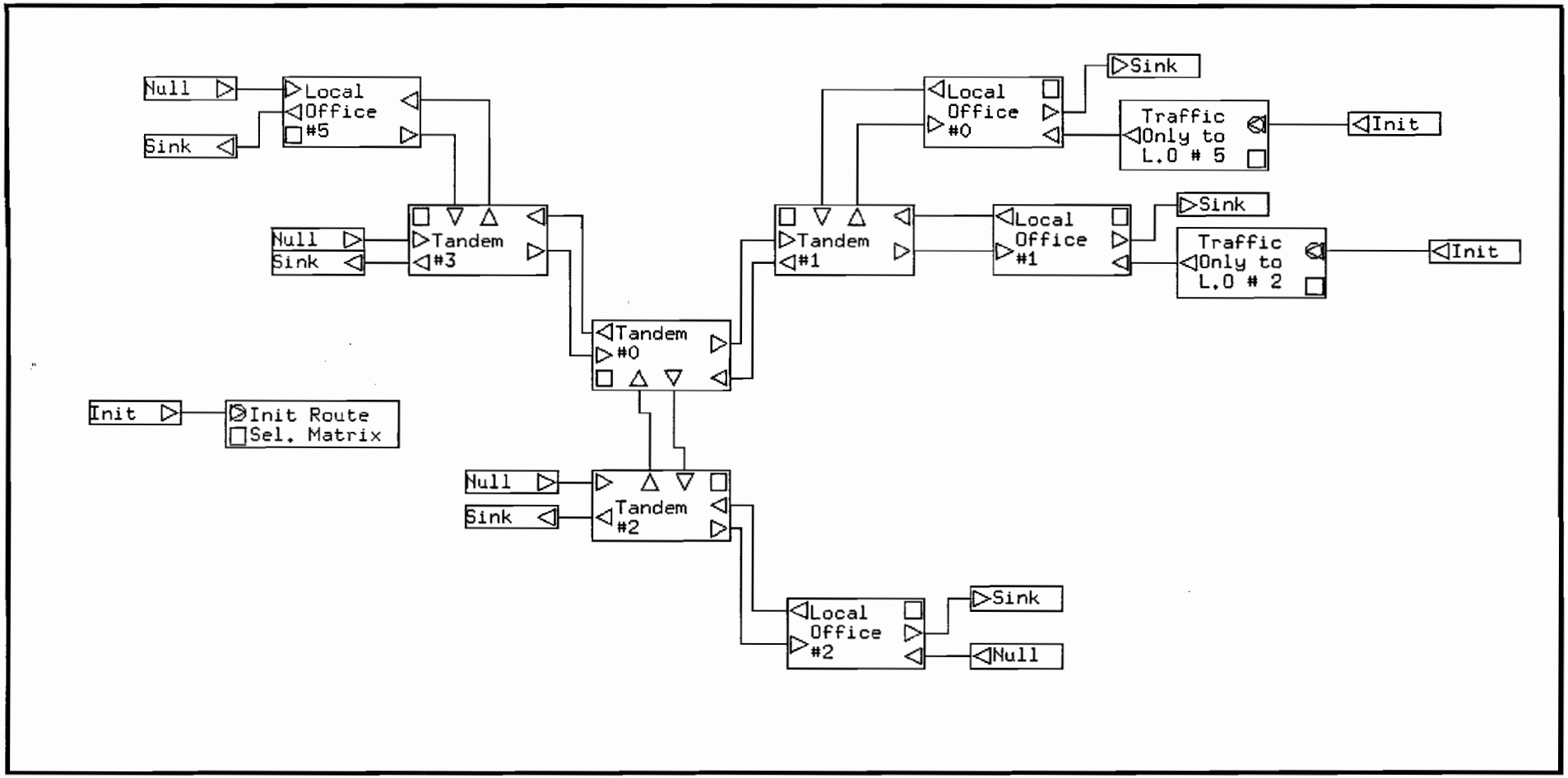



Figure 4.6 Simulation System for the Erlang-B Validation Study.

made "passive", i.e., no outgoing calls are effectively initiated by these L.O's; L.O # 0 is made to generate calls ONLY to L.O # 5; L.O # 1 is made to generate calls ONLY to L.O # 2.

The above configuration permits measurement of blocking ratios for both the 0-5 and 1-2 L.O pairs. Namely, for the 0-5 pair, we are interested in local blocking at L.O # 0; for the 1-2 pair, we want to evaluate remote blocking produced by L.O # 2, but measured at L.O # 1. The above system configuration allows for direct comparison with standard traffic theory results, namely, this study intends to validate its measured data using the Erlang-B blocking formula [1].

The parameter form for this study is shown in Figure 4.7. Note that a few new entries have been added to this form, with respect to Figure 4.2. These include:

- "Subscribers per Local Office";
- "% Calls Needing Outgoing Trunks";
- "# Calls per Hour per Subscriber".

Note that "Traffic I.G Time", described earlier in 4.2.3, can in fact be computed as follows:

"Traffic Rate" = "Subscr. per L.O" * "% needing trunks" * "# calls per hour" / 60 [eq. 4.2]

"Traffic I.G Time" = 1 / "Traffic Rate" [eq. 4.3]

Note: "Traffic I.G Time" is given in minutes.

Comments on Figure 4.7's parameters:

(i) "Tandem Processing Time" is made quite small: this causes the network to introduce negligible delays for the call processing signals, regardless of the amount of network traffic.

(ii) L.O # 5's number of trunks is made twice as large as L.O # 0's. This avoids remote blocking for the 0-5 pair (remote blocking would cause the system to deviate from the Erlang-B model). Similarly, L.O # 1's number of trunks is made twice as large as L.O # 2's. This discards the possibility of local blocking for the 1-2 L.O pair, again allowing the Erlang-B formula to be used.

(iii) The call traffic load offered to both of the "active" L.O's in the system (i.e., L.O's 0 & 1) is made equal to 20 erlangs. This is accomplished by having a "Traffic Rate" of 6.67 calls/min and a "Mean Conversation Time" equal to 3 minutes.

Blocking measurements are gathered in a three-step iteration of the number of

trunks available to L.O's 0 & 2. Results are shown in Figure 4.8. Table 4.2 shows a comparison between the measured blocking ratios and the Erlang-B model.

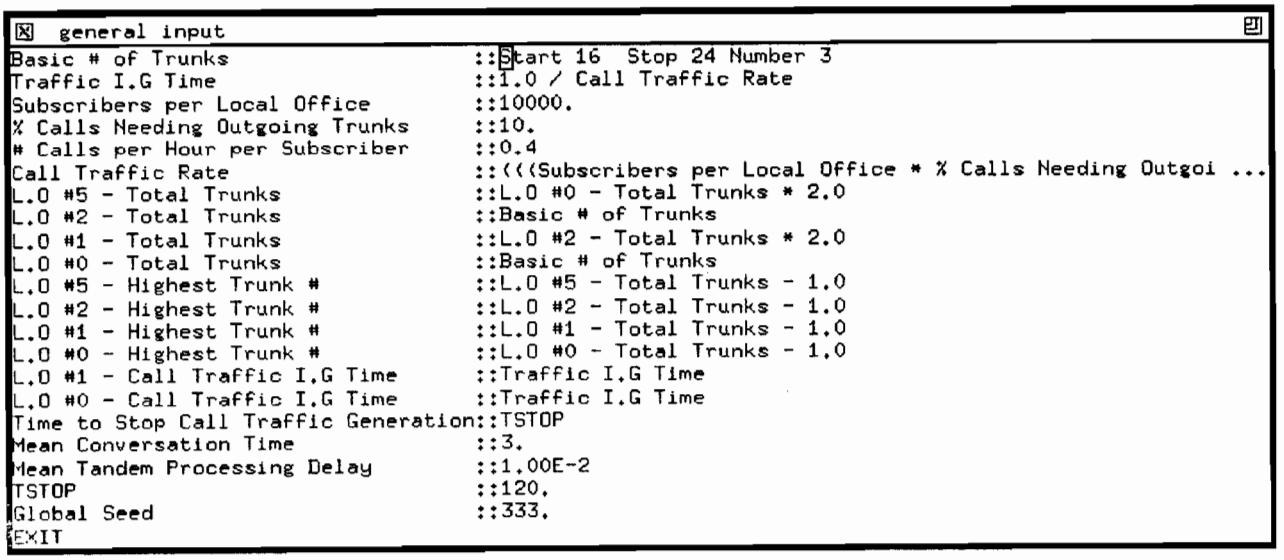



Figure 4.7 Parameter Input Form for the Erlang-B Validation Study

**Table 4.2 - Comparison Between Measured Blocking Ratios and the Erlang-B Formula [1]**

| Number of Trunks | L.O # 0 Local blocking | Erlang-B | % Deviation |
|---|---|---|---|
| 16 | .302 | .293 | 3 |
| 20 | .166 | .160 | 4 |
| 24 | .071 | .068 | 4 |

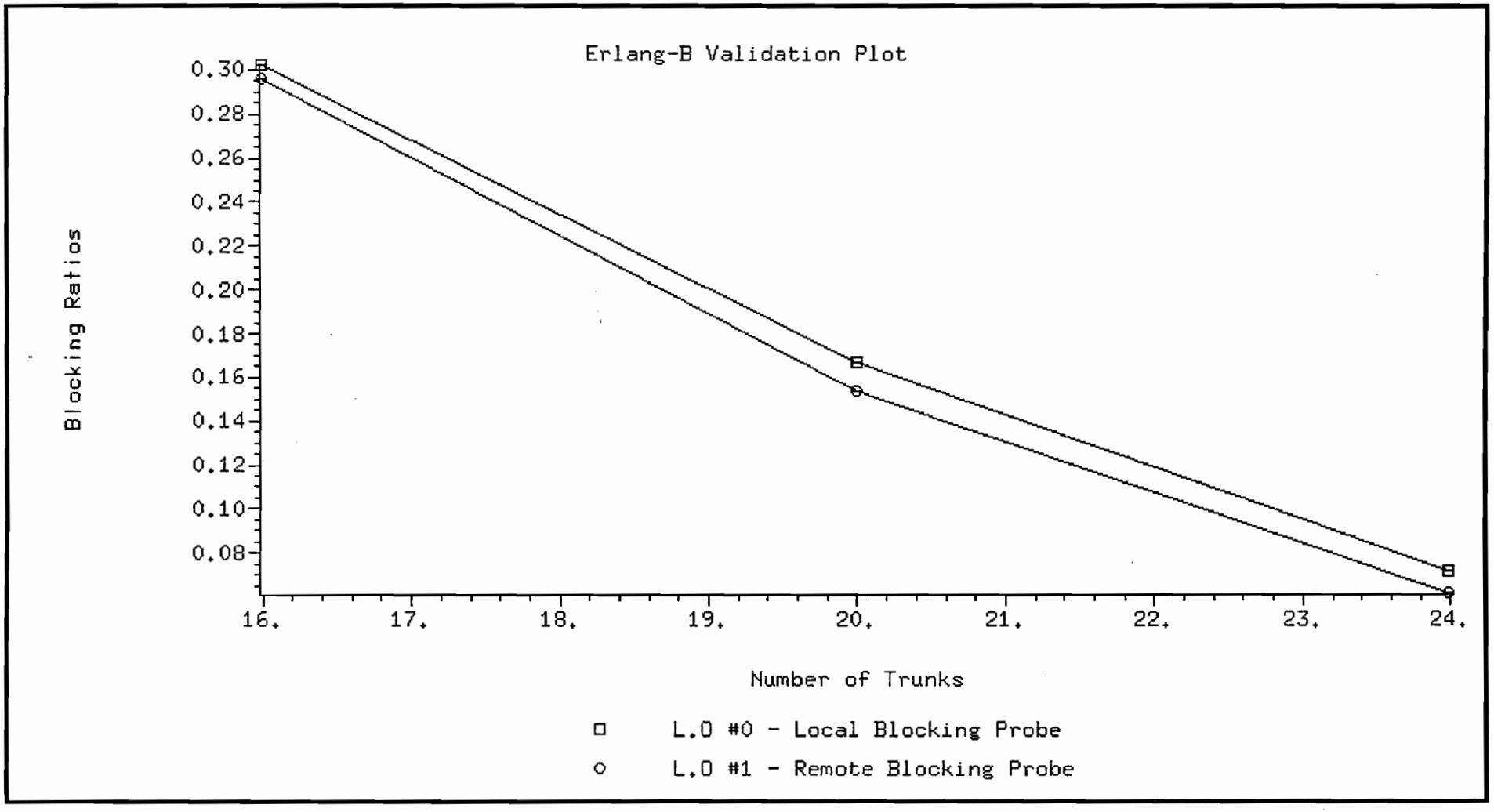



Figure 4.8 Plot of Measured Blocking Ratios in the Erlang-B Validation Study.

## 4.4 Simulation Results

A total of three studies were conducted with the top-level simulation system of Figure 4.1. These studies evaluate the blocking performance of local offices with respect to important network parameters such as traffic intensity, switching rate, and load balancing. The simulation details for these studies are discussed next.

### 4.4.1 Blocking vs. Traffic Intensity

A simulation study of blocking as a function of traffic intensity is configured with the parameter settings of Figure 4.9. In this simulation, L.O's attempt to setup calls with all the other L.O's in the network; all the L.O's in the system are given 48 trunks; the iteration of parameter "# Calls per Hour per Subscriber" causes the load offered to local offices to vary as shown on Table 4.3.

**Table 4.3 - Load Offered to Local Offices**

| Calls/Subscr./Hr. | Offered Load (Erlangs) |
|---|---|
| .2 | 10 |
| .3 | 15 |
| .4 | 20 |
| .5 | 25 |
| .6 | 30 |

A plot showing blocking vs. call traffic rate is shown in Figure 4.10. Note that blocking figures are shown as ratios (i.e., 0 to 1 quantities) rather than as percentages. This plot shows three distinct curves: local blocking, remote blocking, and remote blocking "relative". Both local and remote blocking are computed in the usual way, i.e., number of blocked calls divided by total number of calls. However, the "relative" figure for remote blocking is computed as a conditional probability, namely, it is obtained by dividing the total number of remotely blocked calls by the number of successfully initiated calls only (locally blocked calls do not contribute to the ratio's denominator).

The probes used to compute standard blocking figures were described in [2]. The probe used to compute relative remote blocking is expanded in Figure 4.11.

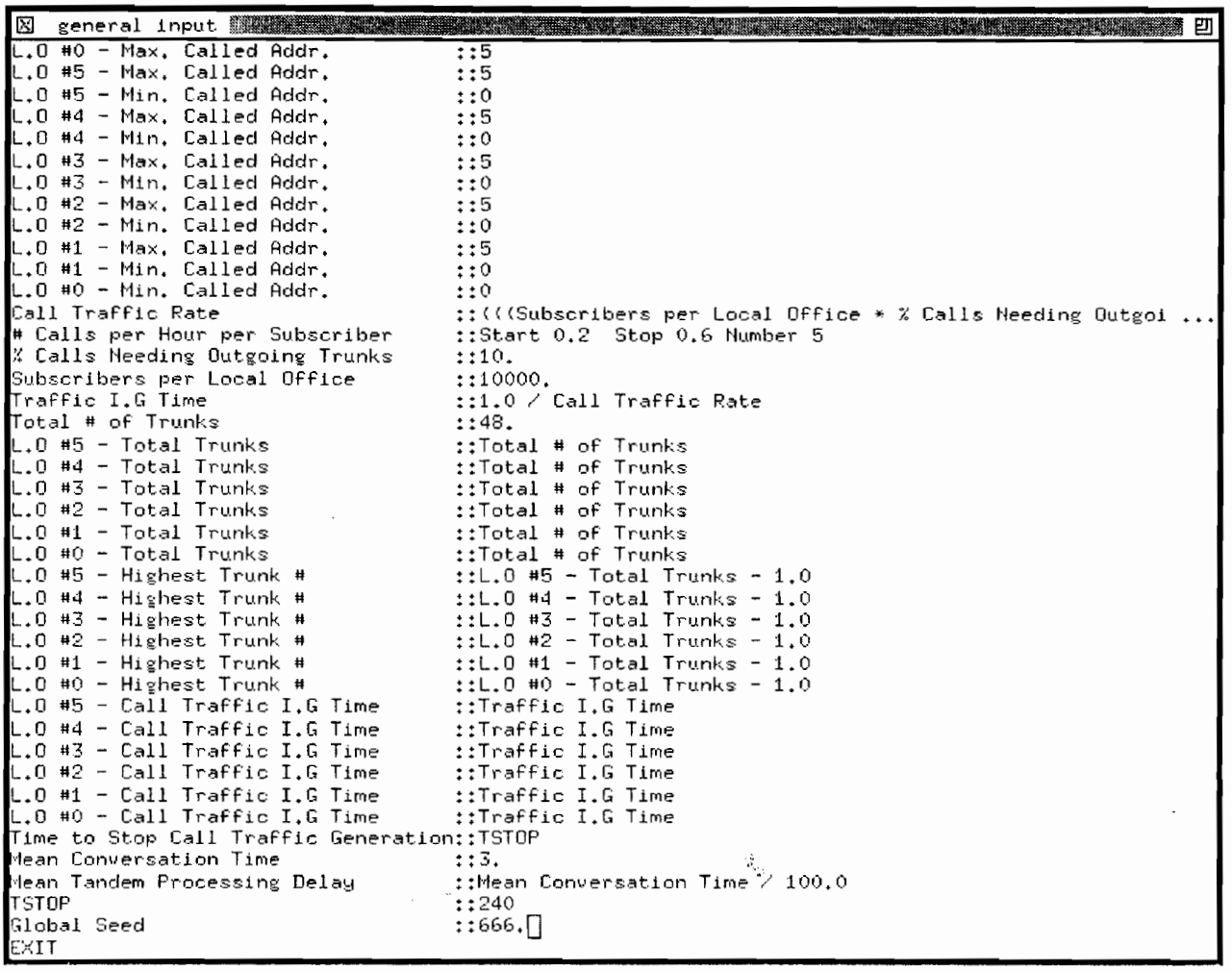



Figure 4.9 Simulation Input Form for the Blocking vs. Traffic Intensity Study.

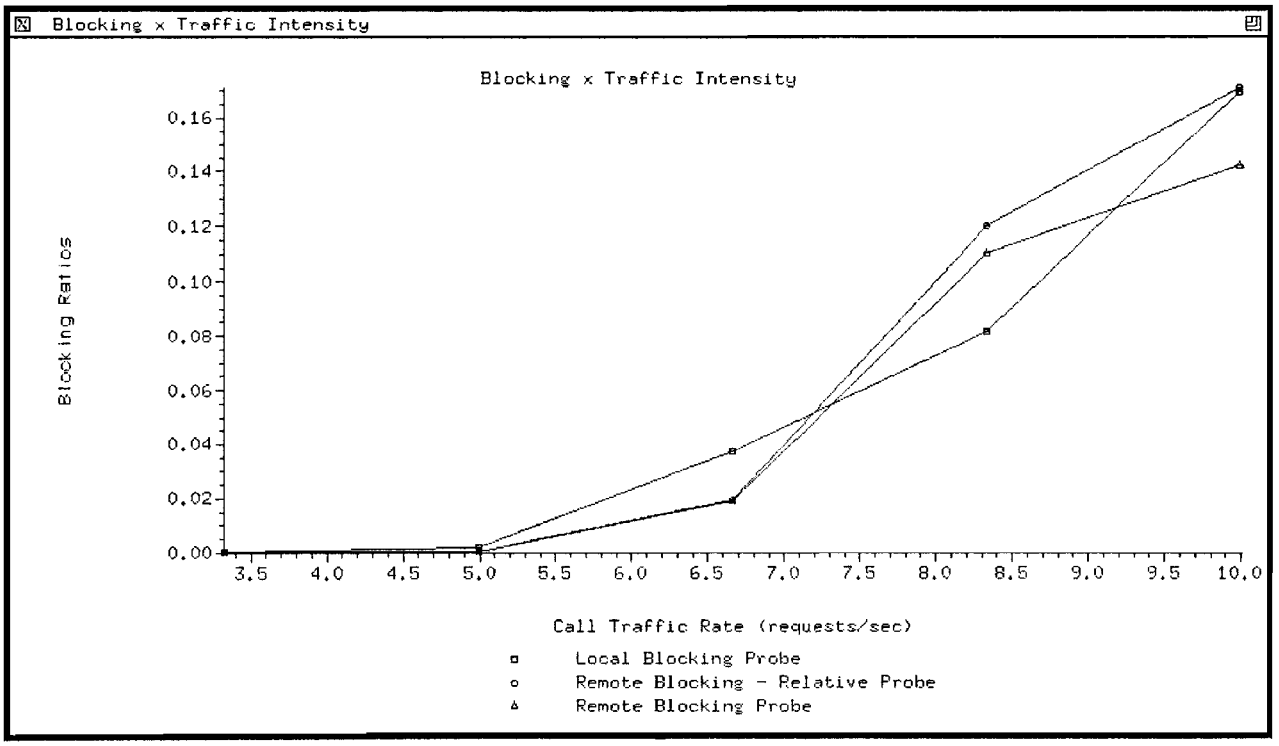



Figure 4.10 Blocking vs. Traffic Intensity Plot.

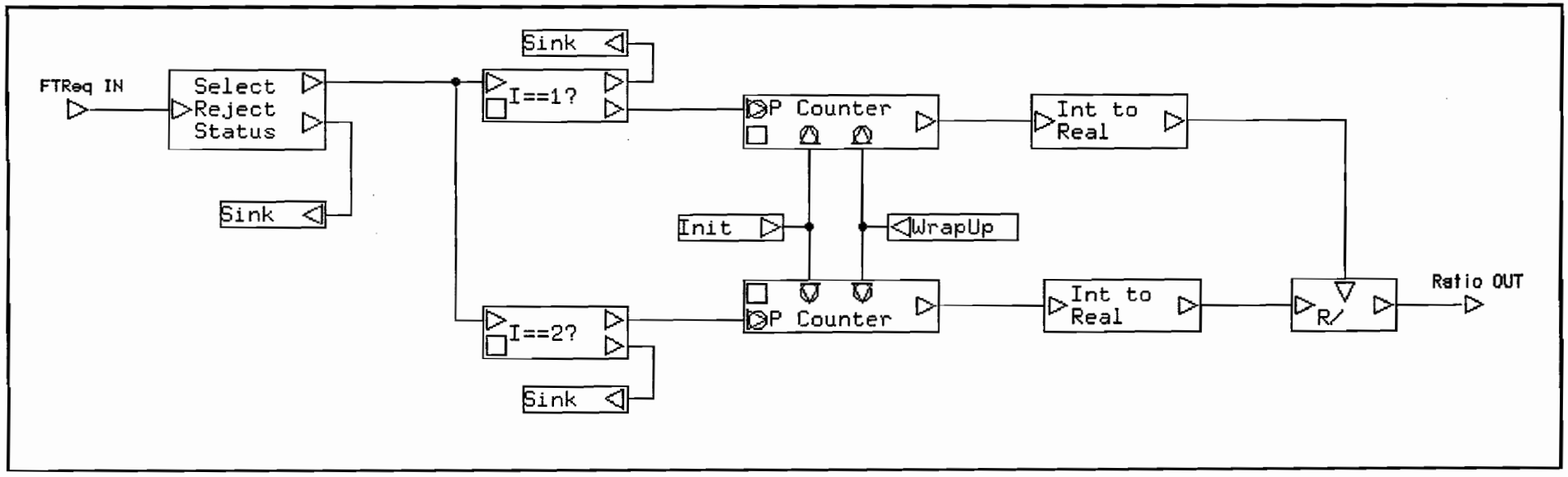



Figure 4.11 The "Remote Blocking-Relative" Probe.

### 4.4.2 Blocking vs. Switching Rate

The next system's level simulation performed evaluates blocking ratios with respect to the switching rate of individual tandems. The corresponding input form is shown in Figure 4.12. As in Section 4.4.1, network traffic is evenly distributed across the network: calls are setup between all L.O pairs; all L.O's are given the same number of trunks, 48; all L.O's are offered the same call traffic load, namely, 20 erlangs.

The parameter of iteration for this simulation is "Mean Tandem Processing Delay"; this is made to vary as follows: .05, .1, .15, .2, and .25 minutes. The blocking curves (local, remote, and remote relative) resulting from this study are shown in Figure 4.13.

### 4.4.3 Blocking in an Unbalanced Network

The last study conducted with the system's level model of the telephone network attempts to evaluate blocking performance with respect to switching rate for an unbalanced network. By unbalanced, it is meant that the call traffic load offered to different L.O's in the system is not the same. Also, call directivity is not symmetrical. An arbitrary directivity/load distribution is devised, as shown in Figure 4.14. The directivity/load characteristics in that figure are tabulated on Table 4.4.

**Table 4.4 - Call Directivity & Load for the Unbalanced Network**

| L.O | Dest. L.O's | Outgoing Load (Erlangs) | Incoming Load (Erlangs) |
|---|---|---|---|
| 0 | all | 20 | 23 |
| 1 | 0 | 5 | 12 |
| 2 | all | 20 | 8 |
| 3 | 0 | 5 | 12 |
| 4 | all | 20 | 8 |
| 5 | 0 | 5 | 12 |

The input form for the unbalanced network simulation is shown in Figure 4.15. Note that "Mean Tandem Processing Delay" is iterated similarly to 4.4.2.

Figure 4.16 shows local blocking measurements performed at local offices 0, 1, and 2. Figure 4.17 shows the relative remote blocking measurements performed at the same L.O's.

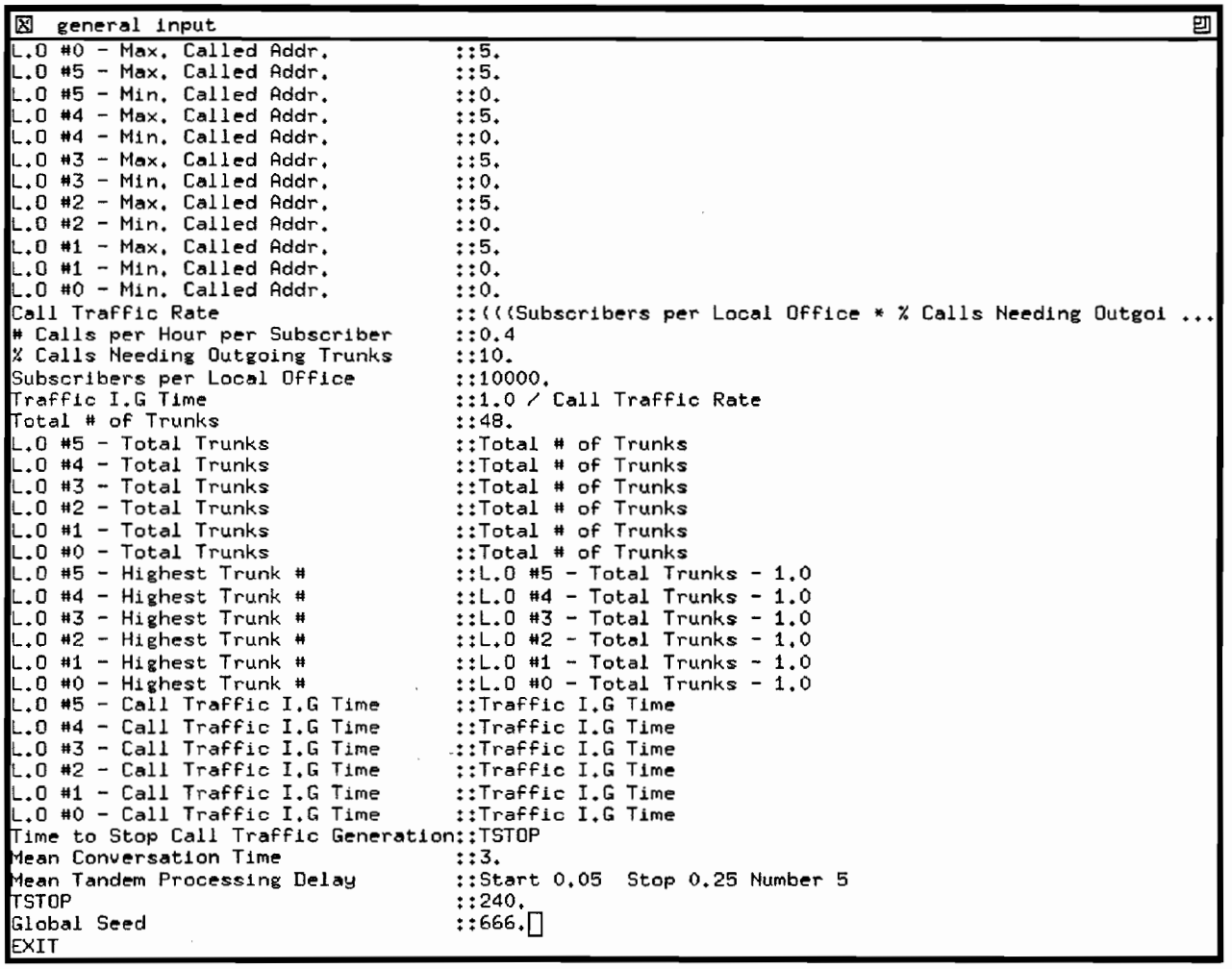



Figure 4.12 Simulation Input Form for the Blocking vs. Switching Rate Study.

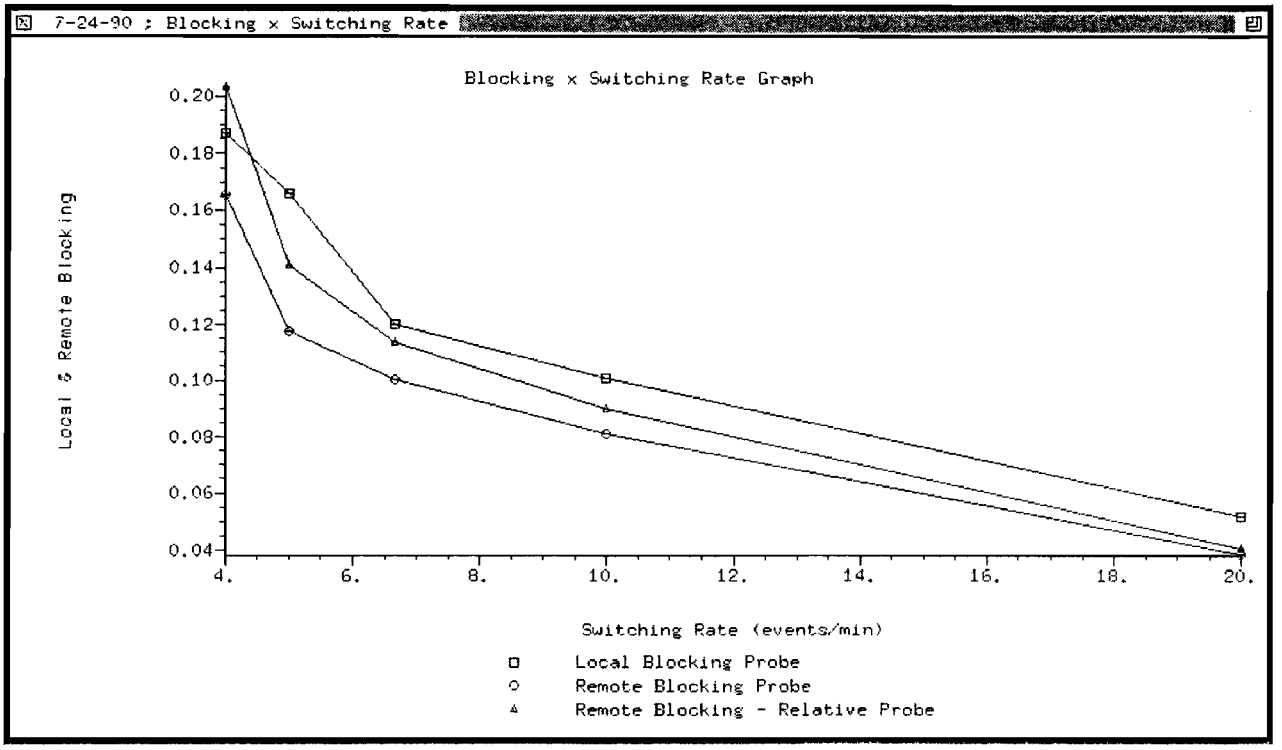



Figure 4.13 Blocking vs. Switching Rate Plot

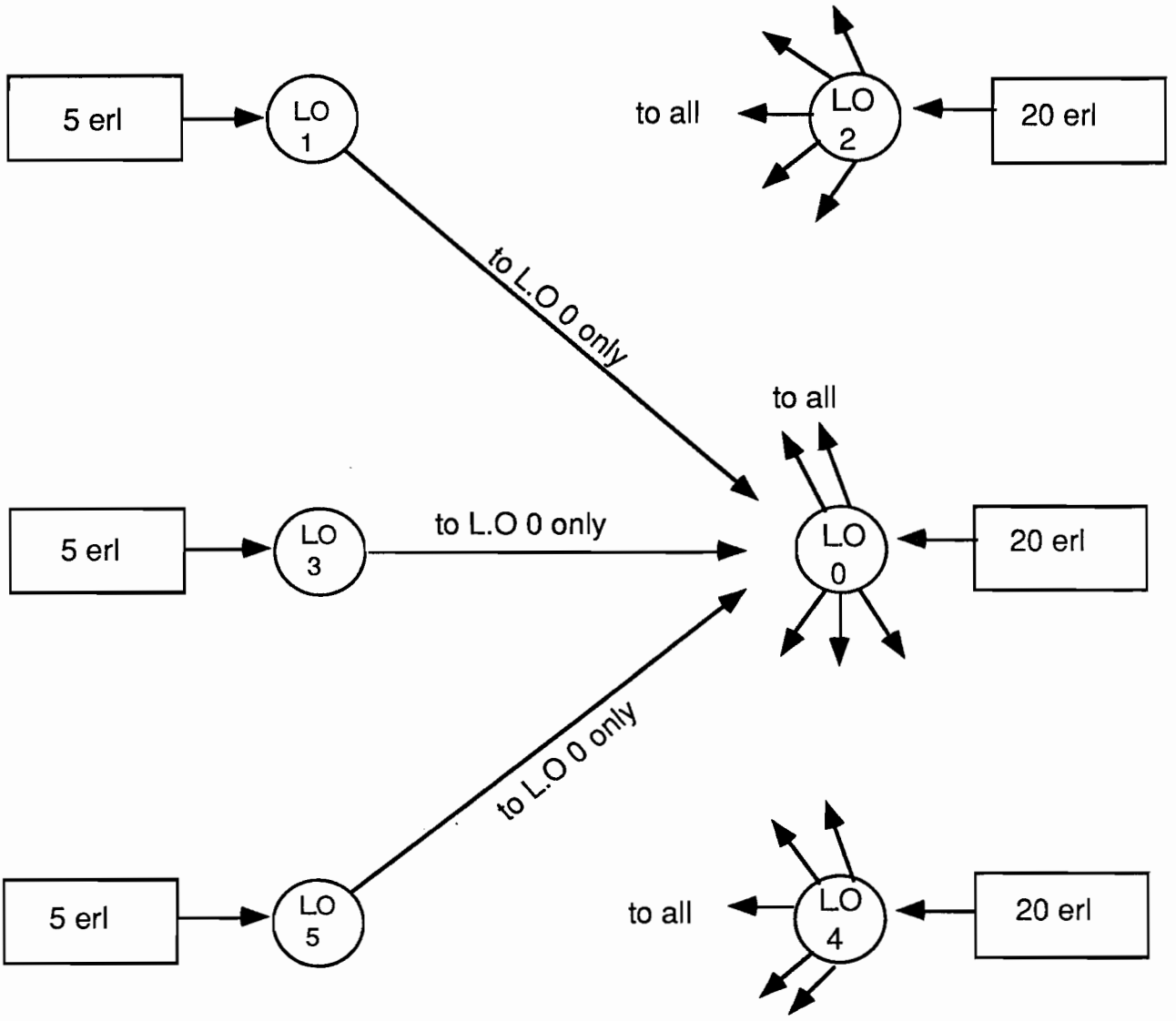



Figure 4.14 Network Traffic Distribution for the Unbalanced Network Study.

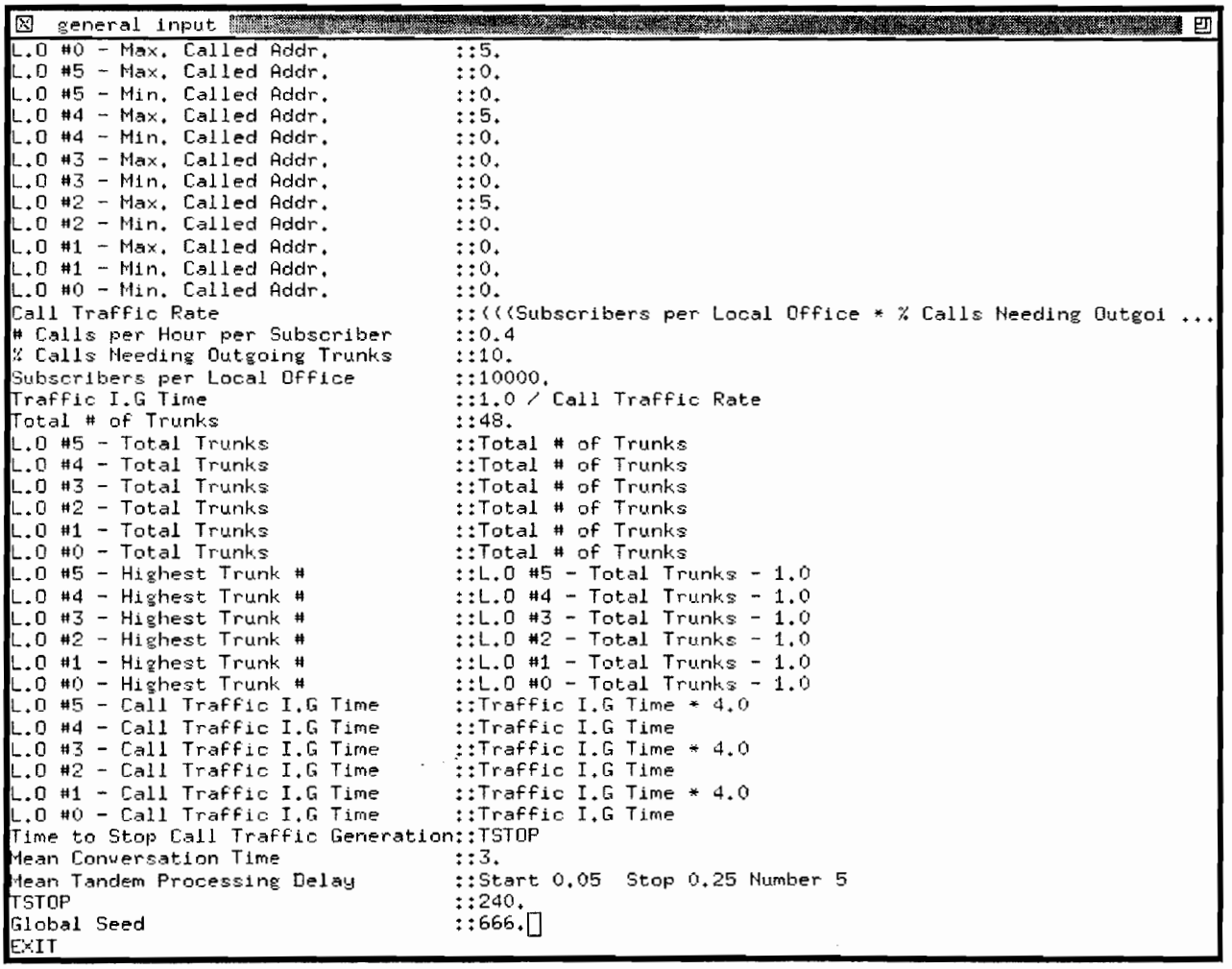



Figure 4.15 Simulation Input Form for the Unbalanced Network Blocking Study.

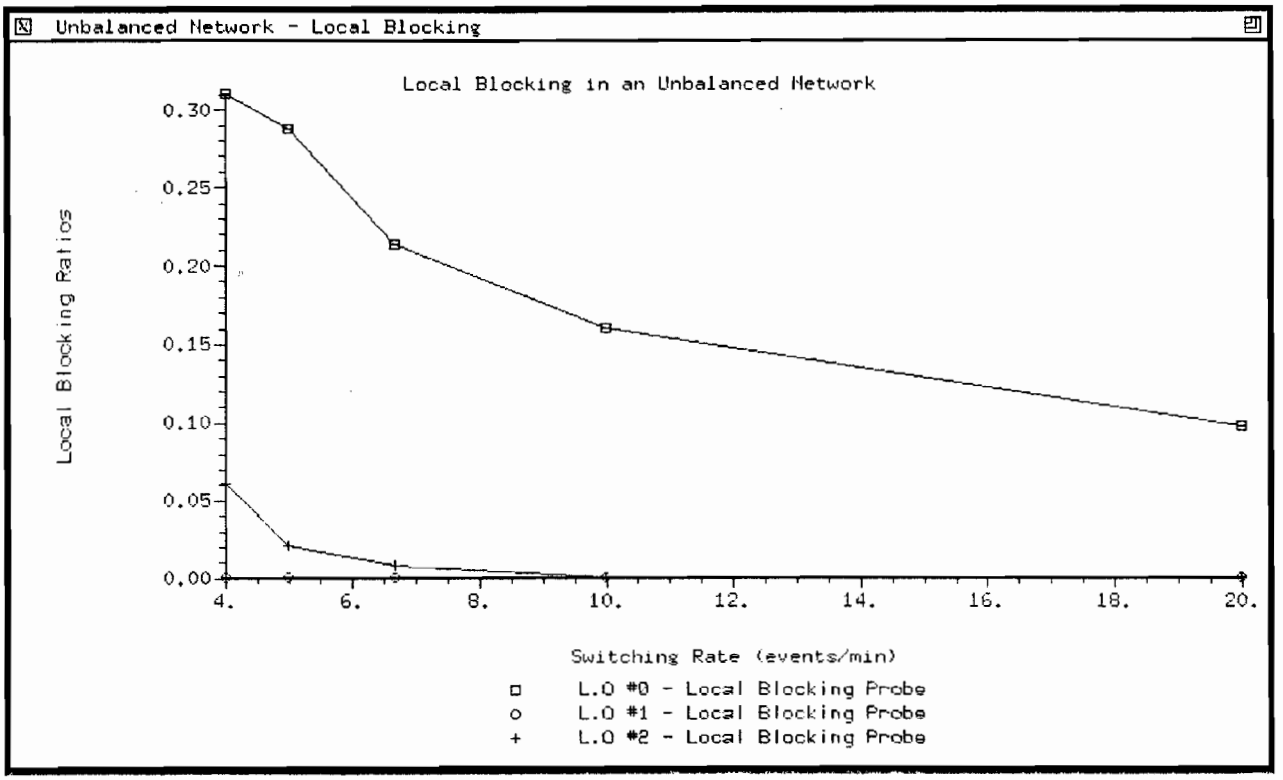



Figure 4.16 Plot of Local Blocking vs. Switching Rate in an Unbalanced Network.

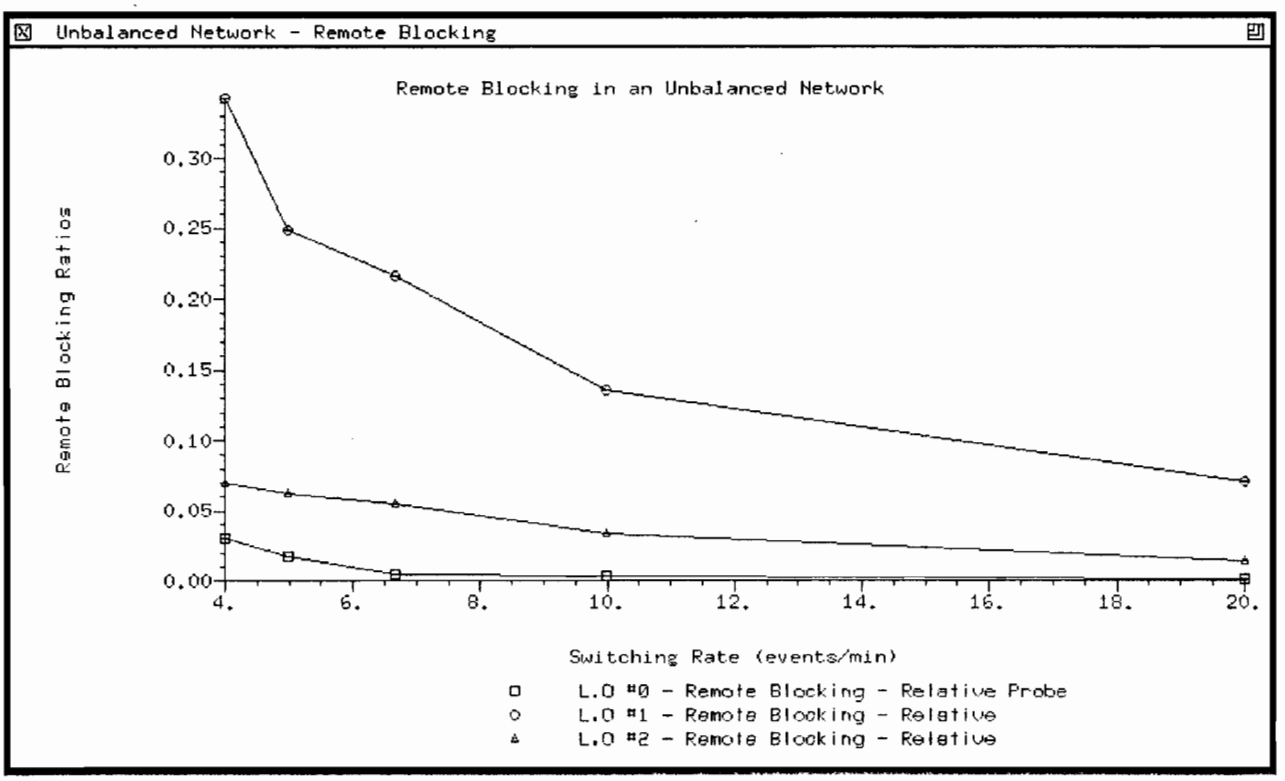



Figure 4.17 Plot of Remote Blocking vs. Switching Rate in an Unbalanced Network.

## 5. Conclusions

Major functional units of a telephone network have been constructed and modeled successfully. The implementation phase has been characterized by a high level of object re-usability. Validation tests show that the implementation is sound. Important network performance measurements such as blocking and call processing visualization have been obtained from system's level simulations. The system demonstrates enough flexibility to be configured to study a large variety of telephone system issues. Further, the model described here could provide the basis for the performance evaluation of CCITT Signaling Systems 7 and Intelligent Networks.

## 6. References:

[1] Bellamy, J. "Digital Telephony", Appendix D, John Wiley & Sons, New York, 1982.

[2] Reznik, D. and Frost, V. "An X.25 Network Model Using BONeS", Technical Report TISL-8670-2, Telecommunications & Information Sciences Laboratory (TISL), University of Kansas, Lawrence, Kansas, May 1990.

[3] Schwartz, M. "Telecommunication Networks - Protocols, Modeling and Analysis", Chapters 1, 10, and 11, Addison-Wesley Publishing Company, Massachusetts, 1987.

## Appendix A - Routing Matrices Used by the Telephone Network:

Switch: S0
Filename: s0.fwd.tel

45
-1 -1 -1 2 2 -1 -1 2 2 -1 3 3 3 3 -1
-1 3 3 1 1 3 3 1 1 -1 -1 -1 -1 -1 -1
-1 2 2 -1 -1 2 2 -1 -1 -1 1 1 1 1 -1

Switch: S1
Filename: s1.fwd.tel

45
-1 3 3 3 3 2 2 2 2 -1 -1 -1 -1 -1 -1
3 -1 -1 -1 -1 1 1 1 1 -1 -1 -1 -1 -1 -1
2 1 1 1 1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1

Switch: S2
Filename: s2.fwd.tel

45
-1 3 -1 -1 -1 3 -1 -1 -1 2 3 3 -1 -1 -1
-1 -1 3 -1 -1 -1 3 -1 -1 1 -1 -1 3 3 -1
-1 1 2 -1 -1 1 2 -1 -1 -1 1 1 2 2 -1

Switch: S3
Filename: s3.fwd.tel

45
-1 -1 -1 2 -1 -1 -1 2 -1 -1 2 -1 2 -1 3
-1 -1 -1 1 3 -1 -1 1 3 -1 1 3 1 3 -1
-1 -1 -1 -1 2 -1 -1 -1 2 -1 -1 2 -1 2 1

## Appendix B - Route Selection Matrix used by the Telephone Model

Filename: route6.mtx

36
-1 0 1 2 3 4
0 -1 5 6 7 8
1 5 -1 9 10 11
2 6 9 -1 12 13
3 7 10 12 -1 14
4 8 11 13 14 -1

## Appendix C - Index to Module Figures