COSC 460
Honours Project Report
Department of Computer Science
University of Canterbury

# HDLC

A Simulation Model to study the performance of the HDLC protocol.

Supervisor: R. Hunt.

Bruce J. McAuley
October 1984

## Table of Contents.

## 1. Introduction.

The High-Level-Data-Link Control (HDLC) protocol has been accepted as an international standard for computer communications. The International Standards Organization (ISO) which define HDLC [8,9] contain options and parameters to be selected according to the environment it is to be used in.

The aim of this project was to produce a Simula program that will model the ISO HDLC Asynchronous Balanced Mode (ABM) protocol. This program can be used firstly to provide a tool for performance evaluation of the HDLC controlled data links and secondly, to provide more insight into how the various parameters of an HDLC implementation and the data channel characteristics influence the performance.

To understand as well as to design information networks it is common practice to distinguish between several layers where specific functions are implemented by means of services offered by the layer below. Here we are interested in the data link layer which aims at controlling the exchange of data between two distant stations.

It is assumed the two stations are connected by some physical means that enables the exchange of data bits at a fixed rate. The exact details of the physical layer(1) are unimportant to the data layer(2).

What is important is that when observed from a higher layer the data link appears to be capable of transmitting data blocks in an orderly and secure way with no loss, duplication or missequencing occuring in spite of occasional transmission errors at the physical level.

The presence of transmission errors as well as the fact that a single data circuit is often time multiplexed on demand between several data streams, requires both stations to apply the same rules to delimit bit sequences, to check validity of data, to organize error recovery mechanisms and so on. Such a set of rules is called a link control protocol.

A number of data link control procedures have been defined, for example, ADCCP (Advanced Data Communication Control Procedure), the national American standard or SDLC (Synchronous Data Link Control) of IBM. Among the existing link protocols the one which is gaining attention and the one chosen for this project is the ISO defined standard called HDLC. HDLC was developed by ISO and accepted by CCITT and as such the balanced mode is fully compatible with LAP-B, the second level of CCITT Recommendation X.25.

Of the three modes of operation available, the one modelled here is the balanced class of procedure, applicable to point-to-point configurations. This procedure is intended for situations which require equal control capability at both ends of the link and for use on links carrying heavy traffic. A data station in this context may be a terminal, a host computer or a switching node within a packet switching network.

What follows is a description of the HDLC protocol followed by a discussion of how the model is implemented, how it may be used and the results obtained from it. Possibilities for further experiments and a discussion on the use of this model as a simulation tool for HDLC are also presented.

## 2. Development and Design of the Model.

### 2.1 Background Information on the Procedure.

For balanced operation the stations at both sides of the link are called combined stations, which means that information transfer, commands and responses may take place in either direction.

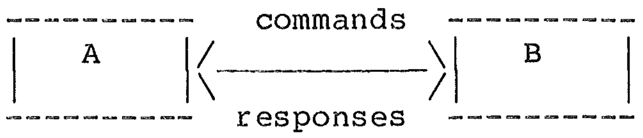


Fig 1.

At Layer 2 where HDLC is implemented all transmissions are in frames and each frame appears as either Fig 2 or Fig 3.

As we are only interested in studying parameters involved with the transfer of data then it is safe to assume that the link has already been set up and is available for the time necessary. Therefore of the usual frames (I->information, S->supervisory, U->unnumbered) the U frame which is responsible for link initialisation and clearing down is not implemented. This also means that frames to handle unknown commands, invalid information fields, unknown responses and so on will also be unavailable.

Therefore the following commands/responses are used:

I: Information.
Send information between nodes.

RR: Receive Ready.
Indicate a station has gone from busy to ready.
Query a station on its status.
Return an acknowledgement.

RNR: Receive Not Ready.
Indicate a station has gone from ready to busy.

REJ: Reject.
Reject this information frame.

I- frame ( I )

| flag | address | control | information | fcs | flag |
|---|---|---|---|---|---|
| 01111110 | 8(bits) | 8(bits) | x bits | 16(bits) | 01111110 |

Fig 2.

S- frame ( RR, RNR, REJ )

| flag | address | control | fcs | flag |
|---|---|---|---|---|
| 01111110 | 8(bits) | 8(bits) | 16(bits) | 01111110 |

Fig 3.

Control

| Contol field bits | 1 | 2 3 4 | 5 | 6 7 8 |
|---|---|---|---|---|
| I -frame | 0 | N(S) | 1 | N(R) |
| S -frame | 1 | 0 S | P/F | N(R) |

Fig 4.

(P/F) is the Poll/Final bit. Whenever this is set to "1" indicates that an immediate response is required. When a frame with a P bit set is sent from a station then no other frame can be sent from that station until the response (F bit set) is returned.

N(S) is the sequence number inserted in a frame by the transmitter to uniquely identify the frame sent.

N(R) indicates the sequence number of the next frame expected by the station transmitting this frame and serves to acknowledge correct receipt of all frames up to N(R)-1.

| | |
|---|---|
| S | 00=RR, 10=RNR, 01=REJ |
| address | always of the opposite station. |
| flag | defines the frame boundary. |
| fcs | considers address, control and information fields. |

The FCS is used to detect corruption due to random or burst errors. It is initialised before transmission and checked at the end of transmission to determine if an error has occured.

Each station has two variables that store the state of the frames sent and received. V(S) the most recent I frame sent and V(R) the I frame this station expects to receive next.

The sequence numbering scheme is based on the modulo of a maximum number. So that all the numbers cycle between zero and a maximum sequence number.

To send an I frame then it is transmitted with N(S)=V(S) and N(R)=V(R). After transmitting V(S) is incremented to V(S)+1 to indicate another frame has been sent. If the number of frames not yet acknowledged by the other station becomes equal to the window size (parameter-WindowSize) then sending further I-frames is held until an acknowledgement arrives.

When a station has no more I-frames to send, it can acknowledge incoming frames by sending supervisory frames either with the response RR if further I-frames can be accepted, or with the command RNR if no further frames can be accepted.

If the receiving station has not been blocked for incoming I-frames, and the FCS is ok, indicating the frame has arrived free of bit or block errors and N(S)=V(R), indicating the frame is the next frame this station expects and is not out of sequence then transmission has been successful, the frame can be accepted and V(R) can be incremented to V(R)+1.

I-frames can be used to acknowledge reception. If there is not an I-frame immediately available, the acknowledgement will wait for a reasonable period of time (parameter-T2) and either "piggyback" on an incoming frame or leave via a Supervisory frame (RR) created specifically for that purpose.

Any frames arriving with bad FCS's are discarded immediately, the error being realized by the presence of the next frame to arrive with FCS intact but sequence number out of order.

Should an out of order sequence number be discovered indicating that somewhere previously an error occured, then a REJ command is sent indicating to the sending station which frames should be transmitted. Retransmission starts with the I-frame indicated by the N(R) count contained in the REJ frame, with the oldest frame first.

All frames received after the REJ has been issued are discarded until the frame expected is received. A REJ frame is sent only once for every group of frames out of sequence. For example suppose frames 1-7 are sent but frame 1 is an error and discarded then frames 2-7 are out of order and must also be retransmitted. The REJ is issued when frame 2 is received correctly with FCS ok but out of order sequence number indicating to the sending station that frames 1-7 must be retransmitted. But frames 3-7 do not trigger a REJ frame they are only discarded.

All frames received with FCS ok have their N(R) fields examined for acknowledgements of previously transmitted I-frames.

If a station should become blocked to I-frames then it sends an RNR command which indicates receive not ready. The station which receives the RNR stops sending information frames until it receives an RR with P bit set which indicates the station is once again ready to receive. The station sending the RNR discards any incoming I-frames which are resent latter.

A single I-frame or the last I-frame in a sequence of I-frames cannot be recovered by REJ. Also, a frame with the P bit set may be lost. Therefore most implementations introduce a timer as a more robust method for dealing with errors. Any frames that are sent restart a timer for a period (parameter-T1). When a timeout occurs all unacknowledged frames are retransmitted, similarly to receiving a REJ. When the last frame sent is acknowledged the timer is stopped.

## 2.2 The Model.

### 2.2.1 Why Simulate?

Simulation is the technique of representing a dynamic system by a model in order to gain information about the system through experiments with the model. It is used to find optimum rules concerning a systems configuration, to provide a basis for long-range decision making or to study the behaviour of a system.

The model must be simplified to the point where it has all relevant details and no irrelevant details. In this sense simulation is an art as there are no definite rules describing what is or is not important. After a model has been decided then it must be verified that the program suitably represents the model and validated that the model is a true representation of the real world.

The approach of this study has been different to others made in this field. In particular to studies found in references [1] to [6] where the usual approach has been to undertake a considerable mathematical analysis and to build a model from this, using simulation to validate both model and results. The difficulty with this procedure apart from the rather involved mathematics is that analytical models tend to be purposeful, resulting in the study of an aspect or component of the system. As that is not the intent of this project a simulation model with the aid of Simula as a programming language has been built. The model is validated by comparing results from the simulation to the mathematical results obtained from papers [1] to [6]. The advantage of this method lies in having a general purpose model for studying the interaction of numerous paramters of the HDLC protocol. In addition there is the capability of observing a display of the relevant parts of the model interacting with frames and packets. This feature is provided with the aim towards improving understanding of the HDLC protocol.

### 2.2.2 Configuration.

A diagram of the model is shown in Fig 5. The two stations are connected by a full-duplex transmission link which is characterized by a bit error probability, propagation delay and transmission speed. The bit errors are assumed to be statistically independent however any other error model can be implemented. It is assumed that all operations occur in zero time and are not slowed by the processor in which the algorithms are implemented.

To model error transmissions a single parameter model has been selected, in which errors are generated according to a Poisson process. More precisely:

$$Pr\{ \text{ one or more errors in a frame of length } L \} = 1 - e^{-L/LE}$$

where LE is the mean number of good bits between two erroneous ones. As pointed out in [2] for suitable paramters, the above formulae can be simplified to 1-(1-BER)**L which is used in the model. From the typical calculations displayed in Fig. 6 it can be seen, as expected, that longer frames are more exposed to transmission errors.

Pr of one or more bit errors.

| L | BER $10^{-3}$ | $10^{-7}$ |
|---|---|---|
| 1064 | 0.641 | $1.06*10^{-4}$ |
| 104 | 0.070 | $1.04*10^{-5}$ |

Fig 6.

As pointed out by J.George and D.Wybux in [1] if the maximum packet size is 512 and OVERHEAD=40 then the maximum link usage is 512/(512+OVERHEAD)=0.930%. Thus about 7% of the link capacity is required just to be able to recover from presumably rare transmission errors.

Errors such as those causing flag destructions have not been considered. The bits '0' which are inserted into HDLC frames in order to avoid undesirable flag detection have not been taken into account.

Transmission is not detailed in the sense that the cyclic redundancy check is replaced by the boolean FCS and a Simula function DRAW is used to determine if an error will occur in the transmission of the frame. DRAW accepts as paramters the probability of returning true and a random number then returns true or false depending on these values. The probability that a frame will be in error has been discussed above and can now be detailed as

```
       FCS := DRAW ( (1-BER)**L, Seed );
```

where BER is the bit error rate, L the frame length and Seed the random number.

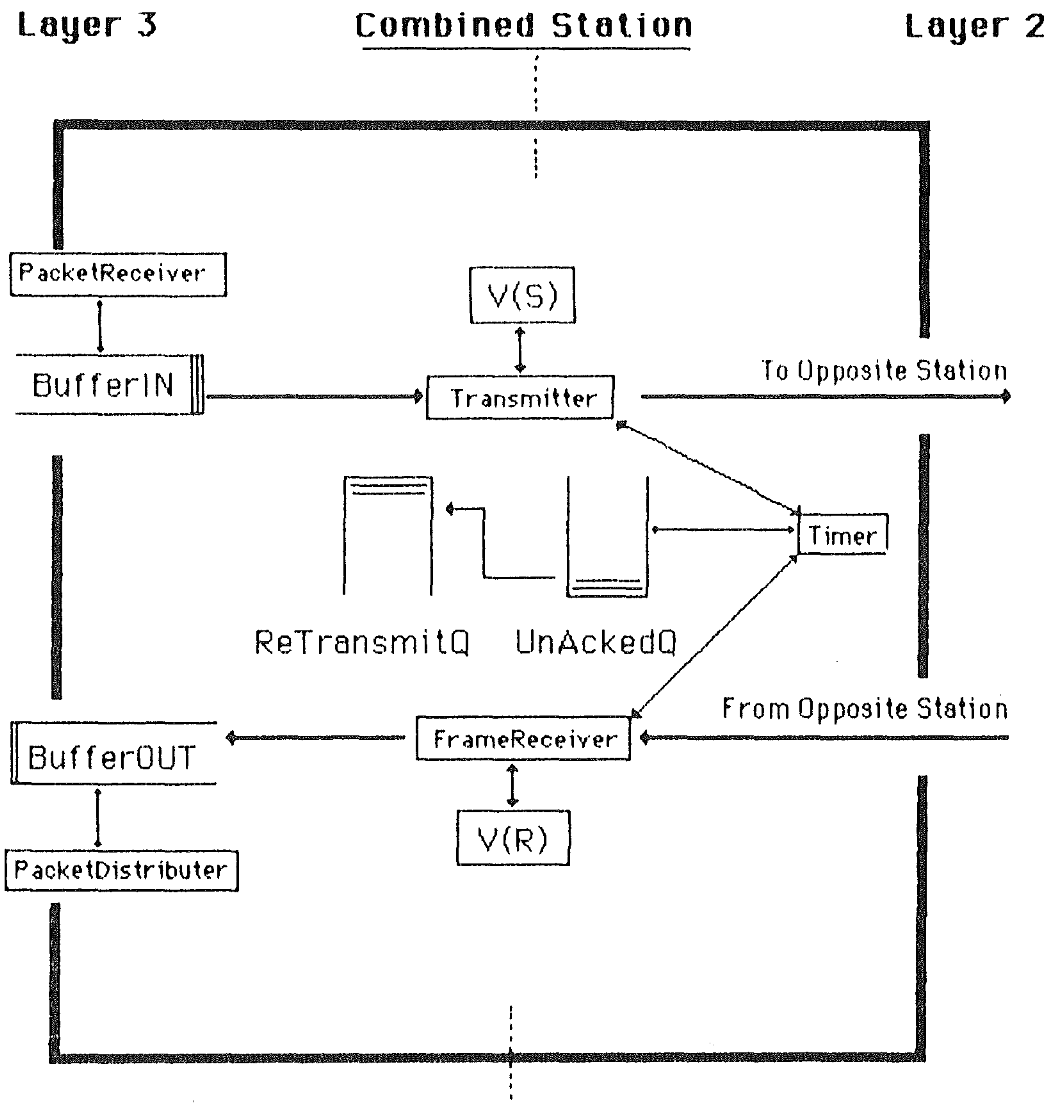

Layer 3
Combined Station
Layer 2
PacketReceiver
V(S)
BufferIN
Transmitter
To Opposite Station
Timer
ReTransmitQ
UnAckedQ
From Opposite Station
BufferOUT
FrameReceiver
V(R)
PacketDistributer

Fig 5

## 2.2.3 Flow of Data and Control.

The flow of data and control at layer 2 is under the supervision of two processes the **framereceiver** and **transmitter**. The interface between layer 2 and layer 3 is carried out by processes **packetreceiver** and **packetdistributer**.

The **packetreceiver** is responsible for placing incoming packets into BufferIN shown in Fig. 5, and notifying the stations' **transmitter** of the packets arrival. It continues placing packets into BufferIN and notifying the **transmitter** until it reaches the maximum buffer space allocated by the user. For most modelling purposes the upper limit placed on this buffer should be large enough so that BufferIN will never become blocked. However if it does block then **packetreceiver** will not accept any packets until notified by the **transmitter**. The **transmitter** will indicate to **packetreceiver** when a new I-frame has been transmitted and therefore BufferIN has space available to receive more packets.

The **packetdistributer** is responsible for placing incoming frames that have been stripped of their overhead bits into BufferOUT, where it takes care of dispersing these packets to the layers above. Similarly to **packetreceiver**, the **packetdistributer** process continues to place packets into BufferOUT until it reaches the maximum buffer space allocated by the user. As described in section 2.1 when an I-frame arrives but BufferOUT is blocked then an RNR-frame is transmitted and the I-frame discarded. As soon as BufferOUT becomes unblocked **packetreceiver** notifies the **transmitter** to send a clear buffer signal which in turn resumes the flow of incoming frames.

The **framereceiver** process is responsible for receiving all frames, removing acknowledged frames from the UnAckedQ, placing frames into RetransmitQ if a REJ or RNR frame is received and notifying the **transmitter** if a reply must be sent. Further responsibilities involve incrementing the V(R) counter which indicates the number of frames received and deactivating the timer when a REJ frame is received, to avoid a double retransmission. There is no other provision to avoid a double retransmission when a REJ frame arrives after timeout when both mechanisms are trying to recover from the same frame loss.

The **transmitter** process is responsible for sending all frames. The emission of a frame lasts a simulated time dependent upon the link characteristics and the length of the frame. Emission also includes the determination of whether one or more bit errors has occured.

A copy of each I-frame transmitted is kept in the originating station until its acknowledgement is received. The buffer or logical queue these frames are kept in is called the UnAckedQ. The **transmitter** ensures that at any time the number of outstanding frames, which is equivalent to the number of frames resident in UnAckedQ, does not exceed the window size. When a timeout occurs or a REJ-frame is received then all frames in the

UnAckedQ are placed into the RetransmitQ in such a way that the oldest frame is transmitted first. Frames in the RetransmitQ are always given priority by the **transmitter** over packets waiting to be transmitted from BufferIN.

A typical life cycle of a packet being transmitted between two stations of this model consists of the following. A packet arrives at BufferIN, if there are no other packets waiting then it is transmitted immediately, with a copy being kept in UnAckedQ for purposes of possible retransmission. If the frame arrives free of error then it is checked that it is in sequence. If the frame is carrying an acknowledgement or group acknowledgement then the relevant frames are removed from UnAckedQ and the **transmitter** notified of the frames correct arrival. The overhead bits are removed from the frame and the packet placed into BufferOUT for dispersal to the upper layers. If there are no packets being dispersed and there are no other packets waiting in BufferOUT then it is distributed immediately. If a bit error has occured or the error-free frame indicates a previous error then the frame is discarded and the copy eventually retransmitted.

### 2.2.4 Why Simula?

Simula is a general purpose language with ALGOL60 as a subset. The main addition, the class construction in Simula, is used to define the data structures and operations of independent objects. Many objects may be generated, co-existing and executing in quasiparallel,i.e as coroutines.

In Simula a system class SIMULATION establishes a context in which it is easy to describe processes (both data structure and operations) and the interaction between processes (both data interaction and sequencing). Thus Simula is well suited for modelling the four processes **packetreceiver**, **packetdistributer**, **transmitter** and **framereceiver** as described in section 2.2.3. Two of the system defined classes are HEAD and LINK classes. Any class prefixed with HEAD has the additional property that it may be used as a list store in which objects of a class prefixed with LINK may be stored. There are a number of procedures available as attributes of HEAD and LINK objects. These two classes enable the simple implementation of buffers BufferIn, BufferOUT and queues UnAckedQ, RetransmitQ. A complex program involving a large number of interactions makes the debugger a useful and efficient tool for debugging.

The simulation was carried out on the University of Canterbury Computer Science Departments D.G. Eclipse S/130 computer. The Simula program is designed to be portable to other simula implementations, however there is one feature that has been used for readability that is non-standard. The feature ANDTHEN has been included as part of the Simula compiler provided by the Lund Institute of Technology. If the expression

```
          WHILE ( el ANDTHEN e2 ) DO
```

is to be evaluated, then e2 is evaluated if and only if el has evaluated to true.

### 2.2.5 Simulation Parameters.

The model has been implemented in Simula and employs event-by-event simulation. It can handle symmetrical and asymmetrical traffic flows and equal or different transmission speeds in both directions of the full-duplex link. The link is assumed to be available and operating during the length of the simulation. The type of transmission is non-selective, that is, all retransmissions are made in the sequence they were originally sent in.

The user determines the following parameters:

.Transmission rate in each direction.
.Propagation delay in each direction.
.Max sequence number of the frame-numbering scheme.
.The number of overhead bits carried in each frame.
.Bit-error probabilities.
.Window Size.
.Length of packets to be transmitted in each I-frame.
.Interarrival rate of packets to be transmitted.
.Interdistribution rate of packets to be distributed.
.Primary timer.
.Secondary timer.
.Size of the buffer, that packets arrive to.
.Size of the buffer, that packets leave from.

The transmission time of a frame is the time between the start and end of the same transmission. How fast a frame can be transmitted is dependent upon the transmission rate. A typical example of terrestrial links is a transmission rate of 9600 bps which enables a frame of 1024 bits plus overhead bits to take a transmission time of 1024+OVERHEAD/9600.

The propagation delay is the time between the start of transmission to the start of receipt of that frame at the opposite station. Typically terrestrial links have propagation delays small enough for zero to be a reasonable approximation, whereas satellite links usually have more significant delays.

The maximum sequence number of the frame-numbering scheme is dependent upon the number of bits available in the contol byte for NR and NS fields. Usually there are three bits which allow a number to range between 0 and 7, the maximum sequence number in this case being 7. It has been left as an option to be specified by the user in the event that extended mode is used which allows a numbering scheme between 0 and 127.

As described in section 2.1 a supplement of information to each frame is needed in order to make possible the detection of garbled frames. This supplement of information including flags, address, control and fcs fields is called the link overhead and consists of at least forty bits. It is assumed the size of the supervisory frames and the size of the overhead is the same. Similarly to max sequence number this has been left as an option to the user in the event the extended numbering system is experimented with, which raises the number of overhead bits required to at least 56.

The bit error rate is an indication of the number of errors the user expects from a transmission link. A BER of 0.001 indicates the user expects a single bit error from the physical layer every thousand bits transmitted. It has been discussed above how BER may be applied to determine the value of FCS.

The window size of each station is the number of frames that can be transmitted before the station is blocked and forced to wait for acknowledgements.

Packets, and therefore the data field of Information frames, are assumed to be of constant length. An I-frame in this model can only contain one packet. They are assumed to arrive at each station according to two independent Poisson processes,i.e, The mean time between arrivals can be different for traffic in either direction. The system is in a stable state if and only if the rate of arriving packets at each node is less than the maximum throughput. If that were not the case the number of packets waiting to be transmitted would grow indefinitely. Similarly packets are assumed to be distributed from each station according to independent poisson processes. If the distribution rate is less than the maximum throughput and the packets are arriving sufficiently fast, the buffer receiving packets to be distributed soon becomes blocked.

The primary timer is restarted with each frame transmitted. It is the time the sending station will wait for a reply from the other station before assuming the transmitted frame or its acknowledgement has been lost. When this timer times out all frames waiting to be acknowledged are retransmitted.

The secondary timer is restarted when an acknowledgement has to be transmitted. If the acknowledgement has not piggybacked a ride to the opposite station by this time then an RR frame is created to specifically transport the response.

### 2.2.6 A Secondary Aim.

A secondary goal of this project, apart from being able to evaluate performance of the HDLC protocol is to improve understanding of the system. An option has been included that allows the user to observe individual packets arriving, being transmitted between stations and leaving the system.

A display is triggered at each significant event, i.e., the arrival of a packet in the system, the distribution of a packet from the system, the transmission and receipt of a frame. The display includes the time the event was triggered, the time the simulation is due to finish, the contents of all buffers from each station, the event that triggered the display the send and receive counters of each station. This display can be very practical for observing the interaction of commands and frames, and a useful device for finding likely bottlenecks.

If the display option is being used and the interaction of frames becomes difficult to follow then another option can be used which allows the user to specify a source and sink file for each station. If these files are specified then arriving packets are simulated by the next character from each stations source file, which is transmitted as a frame and distributed to the upper layers by writing the character to the receiving stations specified sink file. Hence the sending of a packet/frame is simulated by sending a character. If they are not specified the character assumes the default value of "*". An advantage in using this option lies in assuming the user to be familiar with the words or characters in the source files and hence more able to follow the sequence of complicated interactions. This method also acts as a secondary source of validation. StationA's source file by the end of the simulation should appear as StationB's sink file with minor changes. It should be apparent to the user that the HDLC protocol has worked correctly and despite occasional transmission errors all the packets/characters have arrived at the opposite station in the correct sequence and without loss.

The minor changes in the two files are due to an inherent feature of Simula that makes it difficult to implement a suitable procedure to read a line from a file without padding the rest of the line with blanks. Instead of having extra blanks in the file it has been decided to have no blanks at all. Thus the sink file should be a copy of the opposite stations source file without blanks. This leaves the user with the option of being able to observe the flow of a single frame through the system.

This option may be useful to further understanding of the complex interactions involved with HDLC but is usually not used as a tool for performance evaluation. For most evaluation purposes the option of specifying a source and sink file and the option of observing the interactions are left out because of the extra time needed to run a simulation caused by the significant increase in I/O. As this option is not used for performance evaluation the problem of missing blanks in the sink file is of minor consequence. The usual aim of a performance evaluation run is to produce the final report summary which contains information concerning the mean waiting time of frames, the mean transfer time of frames, the number of I-frames transmitted, received and retransmitted, the number of supervisory frames transmitted and a copy of all the relevant parameters specified by the user.

## 3. To Use The Model.

### 3.1 Initialising Paramters.

To run the program, type **HDLC** and then press the carriage return. The user is then presented with a series of questions relating to the configuration of the model. Figure 5 7 presents a typical display.

```
Initialise variables for both stations.
Type '0' to obtain the default value '<>':


Max Sequence Number<7>   : >0
Simulation Length<1sec> : >100
Do you want a display of the interactions  <NO>: >0
Do you want a hardcopy of the interactions <NO>: >1
What do you want to call this file ? : >TRACE.

What do you want to call the report file ? : >REPORT.
```

Fig 7.

All parameters have an indication of what value should be provided by the user within the "<>". Whenever a "<>" appears then this default can be obtained by typing "0". Only queries that do not have a default option should be answered by typing anything other than a number.

A number of these parameters have been previously discussed and will not be described again in this chapter. The reader is refered to section 2.2.5 on the simulation parameters for the necessary information. In the above example the maximum sequence number has been given the default value 7 and the simulation period is for 100 simulated seconds. The simulation period determines the length of the simulation run. When source and sink files are not specified then the simulation period and link characteristics determine the number of frames that will be transmitted.

The user has refused the option of observing the display of interactions. Choosing the default parameter is obtained by typing a "0", choosing any other value for this parameter is achieved by typing any other number. In response to a boolean yes or no query then the default "NO" is obtained by typing "0", any other number typed will have the effect of returning yes or true to the question. When the question

```
Do you want a hardcopy of the interactions <NO> :
```

appeared in the example, the user replied with "YES" by typing a number other than zero. In response to the users reply the next question is for the name of the file the trace of events is to be stored in. Note there is no default value for this.

The aim of each run is usually to gain the final breakdown or report of what happened during the run, thus the final query before initialising the parameters of each station is for the name of the file the report is to be written to.

Next the parameters for each station are initialised. The format of StationA is exactly the same as the format for StationB. A typical example of initialising StationA follows.

```
Initialise Station A.


Transmission Rate<9600.0bps> : >0
Propagation Delay<0.0secs>   : >0
No. of Overhead Bits<40>     : >0
Seed<12345>          : >123
Bit Error Rate <0.0> : >0.0001
Window Size<7>       : >0
Max Information field size<1024>          : >0
Inter-Arrival rate of packets<0.1>        : >0.01
Inter-Distribution rate of packets<0.1> : >0.01
The primary timer T1<1.0sec>     : >0.5
The secondary timer T2<0.0sec> : >0
Max number of packets in BufferIN<15>   : >0
Max number of packets in BufferOUT<15> : >0
Do you want a Source and Sink for Packets ? <NO> : >1
Enter the source file name : >SOURCEA.
Enter the sink file name   : >SINKA.
```

Fig 8.

The Seed is used as an input variable for the distribution functions of Simula. A different random number seed should be supplied for both stations however the option of repeating the same default seed for two simulation runs is provided as it is sometimes useful to recreate a run using exactly the same parameters. Given the same initial seed and the same configuration of a previous run the Simula functions will provide the same outcome.

By using the default values to all questions and providing a file name for the final report the program will run on a default configuration and provide a sample report.

## 3.2 Interaction Display

In fig 7 if the user had replied

**Do you want a display of the interactions** <NO>: >1

then the display of interactions is sent to the terminal. The display is triggered by the arrival of a packet, the distribution of a packet and the transmission or receipt of a frame. The display includes the time the event was triggered, the time the simulation is due to finish, the contents of all buffers from each station, the event that triggered the display, the send and receive counters for each station. As well as the event indication there is a direction indicator that describes the direction a frame is travelling, or which station a packet is leaving from or arriving to.

```
Time Now (secs):     0.355  Finish At (secs):    1.000
Event:              Frame Transmitted Frame Type = I

           StationA            A -> B         StationB
           ========                           ========


BufferIN    [WAITING                 BufferIN    [**

BufferOUT   [*****                   BufferOUT   [r

UnAckedQ    [123                     UnAckedQ    [2
             re-                                  *

ReTransQ    [4567                    ReTransQ    [
             send


VS=  4   VR=  3                      VS=  3   VR=  2
```

Fig 9.

In Fig. 9 the frame with sequence number 3 ("-") has just been retransmitted. The direction indicator "A -> B" indicates the frame is travelling from StationA to StationB. RetransQ and UnAckedQ also display the uniquely identifiable sequence number of each frame transmitted.

Other events that may be displayed include the following:

```
Packet Arrived,  BufferIN
Packet Distributed, BufferOUT

Frame Transmitted Frame Type =  I
  "         "       "    "      RR
  "         "       "    "      RNR
  "         "       "    "      REJ

Frame Received    Frame Type =  I
  "         "       "    "      RR
  "         "       "    "      RNR
  "         "       "    "      REJ
```

StationA is reading packets from a source file whereas StationB is using the default "*" to represent packets. StationA has 7 packets in BufferIN waiting to be transmitted, 3 frames in UnAckedQ waiting to be acknowledged and 5 frames in RetransmitQ waiting to be retransmitted. Frames in RetransmitQ have higher priority to be transmitted over packets waiting in BufferIN. StationA also has 5 packets, that have been correctly received from StationB and are waiting to be dispersed to the upper layers. StationB has two packets waiting in BufferIN, 1 frame in UnAckedQ and no frames being retransmitted.

In fig 7 when the dialogue

```
Do you want a hardcopy of the transactions  <NO> : >1
What do you want to call this file ? : >TRACE.
```

took place, the user indicated a trace of events should be placed in a file called "TRACE". The file will appear exactly as it would on the screen but each event will be one after the other. This enables the user to review the interactions after the simulation run at a more leisurely pace.

## 3.3 Report Summary.

At the end of each simulation run a report summary is provided which contains the value of parameters provided by the user including the number of supervisory frames, the number of information frames transmitted and retransmitted, the number of I-frames received, mean waiting and transfer times of each station.

```
Simulation Length(secs)        100.0
Max Sequence Number                7


                                                   A -> B        A <- B
                                                   ======        ======
Transmission Rate                                  9600.0        9600.0
Propagation Delay                                   0.005         0.005
Overhead                                               40            40
Bit Error Rate                                  0.0001000     0.0001000
Window Size                                             7             7
Max Information Field Size                           1000          1024
Arrival Rate of Packets                             0.130         0.100
Distribution Rate of Packets                        0.100         0.130
Primary Timer (secs)                                1.000         1.000
Secondary Timer(secs)                               0.000         0.000
Max Size of BufferIN                                   15            15
Max Size of BufferOUT                                1000          1000

No. of Supervisory Frames Transmitted                   0           707
No. of Information Frames Transmitted                 646             0
No. of Information Frames ReTransmitted               216             0
No. of Information Frames Received                      0           645
Mean Waiting Time                                   1.336         0.000
Mean Transfer Time                                  1.619         0.000
```

Fig 10.

# 4. Numerical Results.

## 4.1 The Measurements.

Two different and important traffic situations are treated. Firstly the saturated operation where a station always has a packet waiting to be sent, in this case throughput is the measure of performance which most appropriately characterizes this mode of operation. Secondly the non-saturated operation where the amount of information varies statistically, in this case the most relevant performance measure is average delay. This is the more realistic case, it means that transmission channels are not fully loaded because of delays and throughput is less than the maximum achievable.

Bux, Kummerele and Troung in [2] have made a similar study using this concept of monitoring two cases, the saturated and non-saturated operation. In [1] Georges and Wybux use this idea to narrow the range of parameters to study by selecting the best values for the saturated case then applying these values to the non-saturated case.

In the non-saturated operation Bux, Kummerele and Troung distinguish average delays into two categories, the mean waiting time and the mean transfer time. The waiting time is the time interval from the arrival of a packet in BufferIN to the beginning of its first transmission. The transfer time is the time interval between the arrival of a packet in BufferIN and its correct reception, including potential retransmissions, at the other station.

## 4.2 A Discussion of Results.

### 4.2.1 Saturated Case.

Fig. 11 shows the maximum throughput of information bits relative to the transmission rate as a function of the length of the information field in I-frames. The maximum ratio of information throughput to transmission rate when the packet size is 10 and the overhead is 40 is 10/(10+40), which equals 0.2, i.e. the best information throughput obtainable occurs in the presence of zero errors and can be found by the ratio I/(I+OVERHEAD).

Given 10 seconds to transmit the data at 9600bps then the station should be capable of transmitting 10*9600 bits. From one of the simulation runs with these parameters and a bit error rate of 0.001, a total of 1596 frames were transmitted of which there were only 10 information bits in a frame therefore the information throughput was (1596*10bits)/(9600*10seconds) which is approximately 0.166 but significantly less than 0.2. From the graph we can see that bit error probabilities equal to or less than 10**-7 start to follow the maximum throughput line found by I/(I+OVERHEAD).

The curves of the graph are typical of systems depending upon retransmissions, and demonstrate how much throughput depends upon the likely error rate. For large bit error rates there is usually a definite maxima, i.e. for this error rate there is a best information field length to ensure maximum information throughput. This is because with short I-fields there is a low probability a frame gets disturbed but the number of overhead bits is relatively high. With a large I-field, the number of overhead bits is relatively lower but the probability of a frame error is higher. These sort of graphs are useful for cost assesments, i.e. it may be worthwhile to improve the current system and to reduce the bit error rate to gain a significant increase in throughput.

Fig. 12 demonstrates the impact of the window size and propagation delay by comparing terrestrial and satellite links. First, the terrestrial case, when the bit error rate is zero then there is no significant difference in throughput. Secondly the satellite case shows how a window size of 7 greatly reduces the amount of information throughput compared to the window size of 127. A small window size significantly reduces the maximum information throughput because transmission of I-frames has to be discontinued if frames are not acknowledged. For higher window sizes, the situation should improve possibly coinciding with values of a terrestrial link.

**Transmssion Rate = 9600 bps**
**Propagation Delay = 0 secs**
**Window Size = 7**

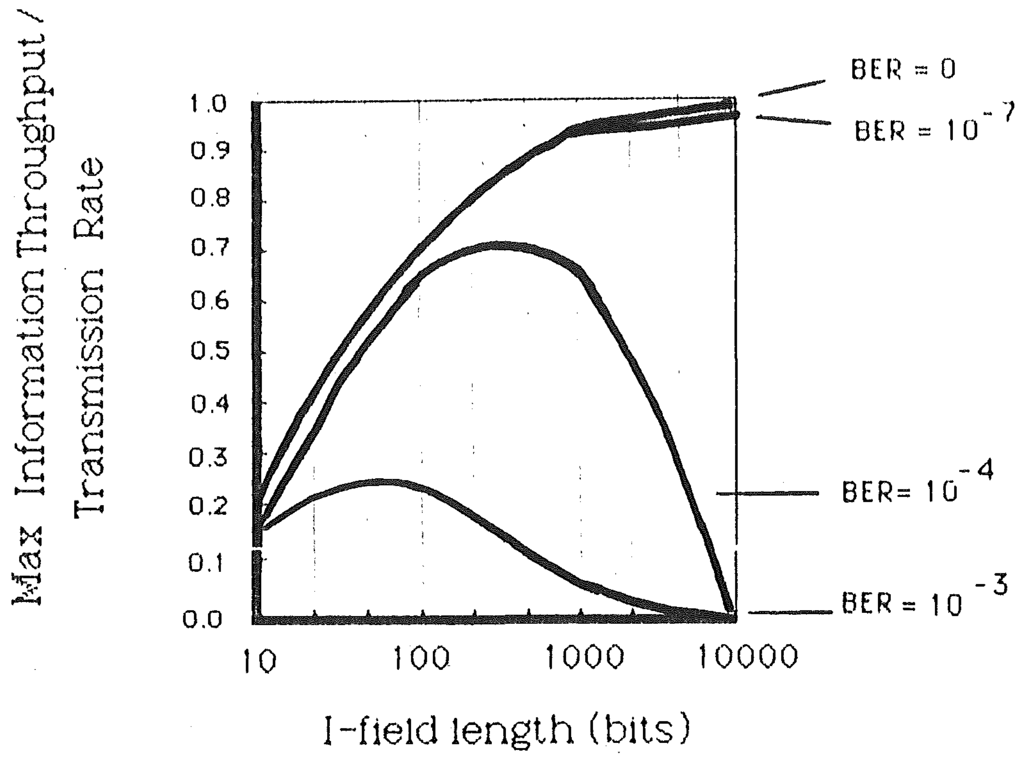


**Fig 11**

**Transmssion Rate = 51200 bps**
**BER = 0 secs**

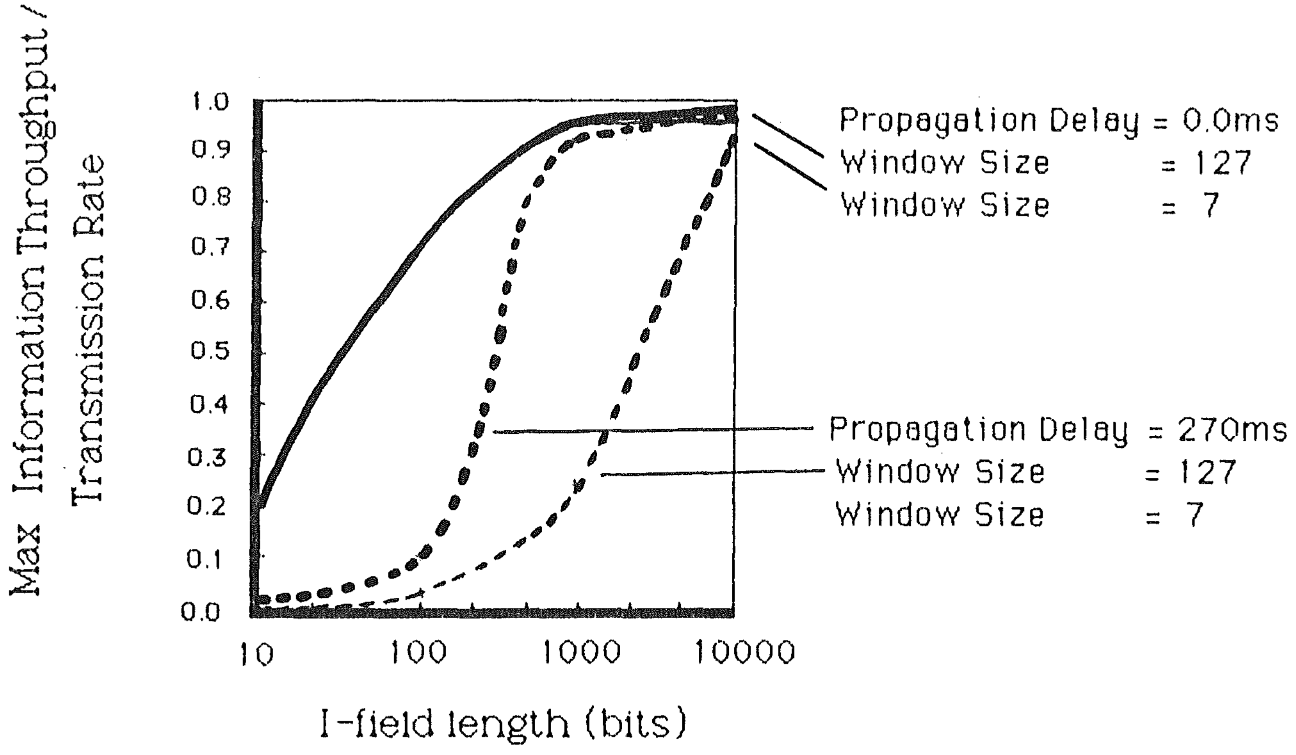


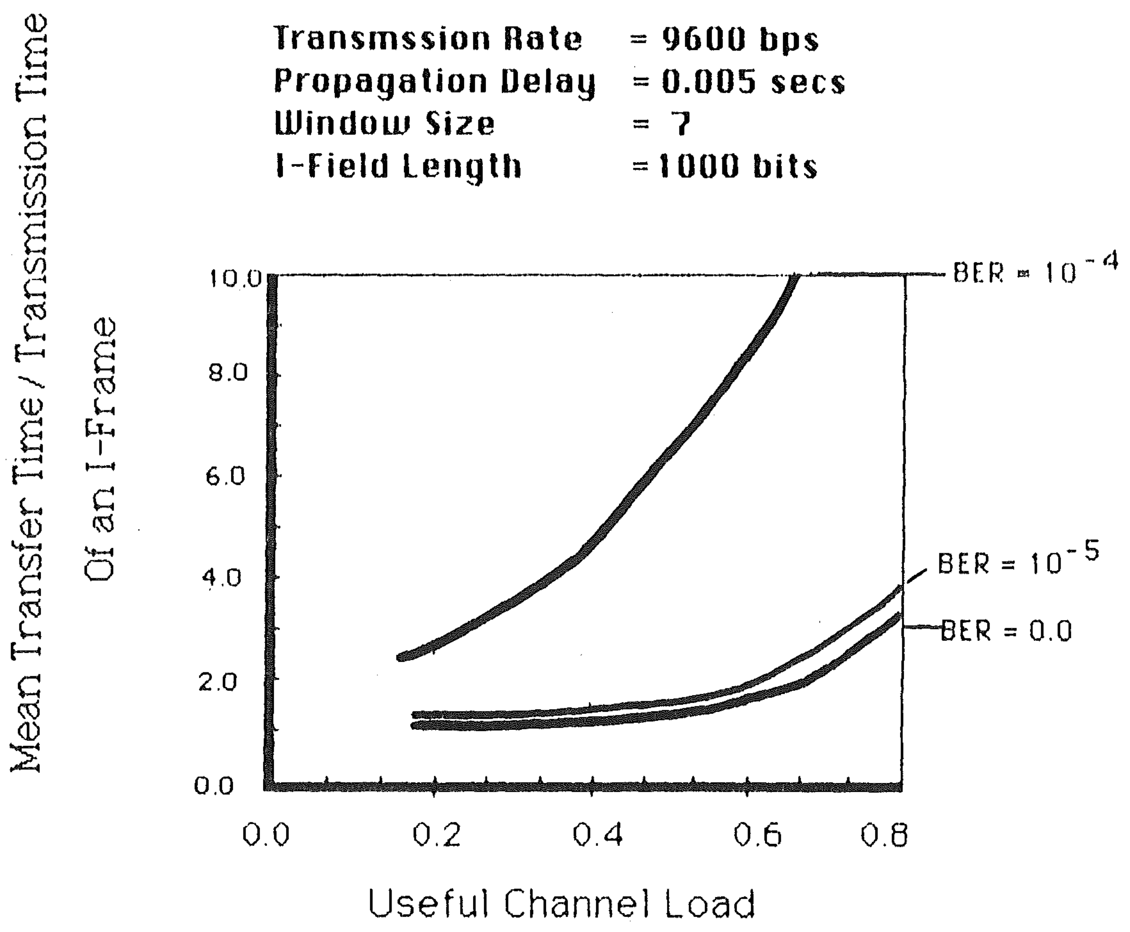
**Fig 12**

### 4.2.2 Non-Saturated Case.

Fig. 13 demonstrates the effect error rate has on the mean transfer time. The axis are, -the mean transfer time/ transmission time of an I-frame versus the useful channel load. The useful channel load corresponds to the portion of total channel load caused by the successful transmission of information bits. Transfer time is the time a packet enters the system until it is correctly received by the other station. Because the non-saturated case is being considered the channels are only loaded corresponding to a fraction of their full capacity. The useful channel load is equivalent to Arrival Rate * I-frame length/Transmission rate.

A distinct minima is shown in fig. 14. For small I-field lengths the number of bits used for overhead is relatively large and since a large number of overhead bits is necessary to achieve this throughput. As the I-fields grow larger the relative overhead size decreases but the block error probability grows, error recovery takes longer and waiting and transfer times grow proportionally with the length of I-frames. A comparison with fig. 11 shows that the shortest delays occur at significantly smaller I-field lengths than the maximum throughput values. Presumably the increase of transmission time and block error probability with growing I-frame lengths outweighs the impact of the decreasing overhead at smaller I-field lengths.

**Transmssion Rate = 9600 bps**
**Propagation Delay = 0.005 secs**
**Window Size = 7**
**I-Field Length = 1000 bits**



**Fiq 13**

| | |
|---|---|
| Transmssion Rate | = 9600 bps |
| Propagation Delay | = 0.0 secs |
| BER | = $10^{-5}$ |
| Window Size | = 7 |
| Channel Load | = 6000 bps |

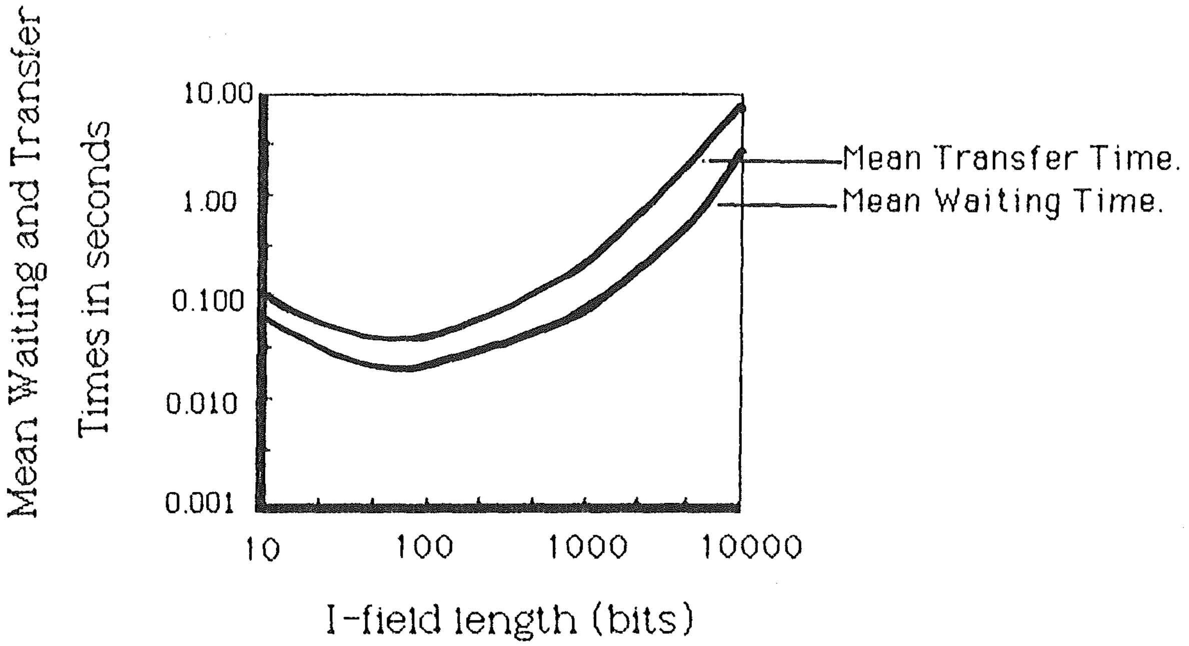


Fig 14

## 4.3 Results Conclusion.

From the results, it is apparent that if the relevant parameters are adjusted to meet the specific needs of the users then the HDLC balanced class of procedures provides efficient utilization of the transmission lines with low delays and transfer times.

A study of the saturated and non-saturated case is the key for performance considerations. In the saturated case where throughput is the most relevant consideration, then it can be shown to have a distinct maxima determined by the length of I-field and the bit error rate. The window size of satellite links has a significant impact on throughput as opposed to terrestrial links where window size has very little impact. With the non-saturated option, waiting and transfer times show definite minima determined by the length of I-field and by the information throughput [4,6].

## 5. Further Experiments.

### 5.1 Experiments.

Only a portion of the performance evaluations that can be carried out with this model have been demonstrated. What follows are some ideas for further experimentation with the existing model. These experiments have not been attempted but the model has been designed, built and tested to the extent that all of these experiments are possible.

A detailed experiment could include a study on the performance of traffic in both directions, and then a study of disymmetrical traffic where the paramters for traffic in each direction are different. It may be shown how a window size of 7 is suitable for symmetrical traffic but not appropriate under disymmetric conditions, or how the largest window width of 127 may be sufficient for packets of reasonable length but short packets might require a further increase in window size to produce a more efficient throughput.

It has not been discussed what effect changes to the primary and secondary timers would have. A study on the values of T1 and T2 is useful in the sense that if T1 is too small then some frames will be retransmitted for no reason other than a timeout, creating a deadloop, or if T1 is not small enough this will result in some inefficiency while frames wait to be retransmitted. If T2 is too small then an RR-frame will be created to piggyback the acknowledgement for every frame received. If it is too large then there is some loss in efficiency when the opposite station becomes unable to transmit until some of the transmitted frames are acknowledged.

## 5.2 Extensions To Model.

Any small changes to the model can be made efficiently because of the modularity provided by Simula and the relatively simple queues used to implement the model.

Another possibility for experimentation lies in having a number of packets being transmitted by a single I-frame. It is assumed in the model that all packets are of the same length and that only one packet is transmitted per I-frame. It was decided to model I-frames in this manner after a number of discussions. The standards [8] and [9] leave these parameters optional but it appears that most implementations follow the procedure of padding packets that are not full with blanks or other such characters. If a study were carried out on this parameter only, it may provide useful results. For example a user may discover that a newly configured model that accepts exponentially distributed packet lengths and fits any number of packets into the information field of a frame may find that throughput increases significantly enough to justify a change in the actual implementation.

Inter-arrival and inter-distribution times have been modelled with an exponential distribution. In reality the distribution to use depends upon the implementation being simulated. Other distributions that may be used are hyper-exponential and constant distributions.

## 6. Summary.

A powerful simulation tool for studying the effects and interaction of a number of paramters of the HDLC protocol is provided. It has been validated by two methods, firstly by comparison with results found in other studies that have used analytical and simulation methods, and secondly by directly observing the flow of frames and outcome of the interactions. An indication of how it may be used in practice has been demonstrated by performing a study of a general configuration.

A weakness of the model is the algorithm used to detect blocked buffers. When the buffer becomes full an RNR frame is sent and further received frames discarded until the station sends an RR indicating it is once again ready to receive frames. It is possible for the user to initialise a configuration such that every second frame is retransmitted because of a blocked buffer. This happens because a station is considered blocked when BufferOUT is full and unblocked after **packetdistributer** has released a packet. Thus it is possible for a station to alternate between blocked and unblocked. For performance evaluation of other areas this problem is usually ignored by making the buffer space large enough so that BufferOUT is never likely to become blocked. However if a study were to be made in this area then it would be useful to develop a better algorithm for handling this situation. Possibly an algorithm that "anticipates" the buffer filling and delays sending an RR frame until a sufficient amount of space is available can be implemented.

It has not been possible to demonstrate every use of the model nor has it been possible to cater for every implementation the model may need to simulate. Section 5 describes some of the experiments that can be performed by the model and suggests possibilities for extensions to the model.

## References.

[1] Bux W., Kummerele K. and Troung H.L.
"Results on Performance of Balanced HDLC Procedures."
National Telecommunications Conference.
Birmingham Alabama U.S.A. December 1978.

[2] Georges J. and Wybux D.
"A Simulation Study of the Performance
of HDLC Controlled Links."
Performance Evaluation 1 (1981) pp 126-138.

[3] Bux W. and Troung H.L.
"High Level Data Link Control Traffic Considerations."
9th International Telecommunications Conference.
Torrenolnos Spain. October 1979.

[4] Labetoulle J. and Guy Pujolle.
"HDLC Throughput and Response Time for Bidirectional
Data Flow with NonUniform Frame Sizes."
IEEE Transactions on Computers.
Vol C-30, No.6 June 1981.

[5] Bux W. and Troung H.L.
"A Queuing Model for HDLC Controlled Data Links."
International Symposium on Flow Control
in Computer Networks. Versailles. 1979.

[6] Labetoulle J. and Pujolle G.
"Modelling and Performance Evaluation of the Protocol HDLC."
Ibid. pp307-320.

[7] Belsnes D.and Bringsrud K.
"X.25 DTE Implemented in Simula."
Proceedings Eurocomp. London. 1978.

[8] ISO International Standard IS-3309.
"Data Communications-High Level Data Link Control Procedures-
Frame Structure."
ISO-TC97/SC6 Document N2293 July 1981.

[9] ISO Draft International Standard (Dis)4335.
"Data Communications-High Level Data Link Control Procedures-
Elements of Procedure."
ISO-TC97/SC6 Document N2290 July 1981.

[10] Tanenbaum A.S.
"Computer Networks"
Prentice Hall, 1981.

```
BEGIN
SIMULATION BEGIN
Comment           ++++++++++++++++++++++++++++++++++++++++++++++++
                  +                                              +
                  +                      HDLC                    +
                  +                                              +
                  +     A Simulation Model to study  the         +
                  +     performance of the HDLC protocol.        +
                  +                                              +
                  +                   Bruce J. McAuley           +
                  +                   October 1984               +
                  +                                              +
                  ++++++++++++++++++++++++++++++++++++++++++++++++

    The following is a simulation  program written in Simula that models
  an HDLC link. The link is assumed to be available and operating during
  the length of the simulation. Therefore U-frames which are responsible
  for link initialising and clearing down are not implemented.The frames
  involved are I-information, RR-ready to receive, RNR-reaoy not receive
  and  REJ-reject.  The type  of transmission is non-selective, that is,
  all retransmissions are maoe in the sequence they were originally sent
  in. Link parameters that can be altered include the transmission rate,
  propagation delay, and the  maximum sequence number.

    The   parameters   which   can   be   altered   for   each   station are
  bit error  rates,  window  size,  maximum size of information field,
  arrival   rate   of   packets   to   be   transmitted, distribution rate of
  packets to be distributed to layer 3,  primary  and  secondary timers,
  the maximum number of packets allowed in the input/output buffers, and
  the number of overhead bits carried per frame.

    Each   run   of  the  program  produces a file specified by the user
  containing a summary of the simulation providing information including
  the number of frames transmitted  and  received,  the number of frames
  retransmitted, a copy of the parameters provided  and the mean waiting
  and transfer times.

    For further details refer to the honours report and appendices.    ;


 REAL SimTime             ;        Comment Length of simulation       ;
 INTEGER SeqNoMax         ;        Comment Max Sequence Number        ;
 TEXT ARRAY Screen(0:19)  ;        Comment Terminal and debug info    ;
 INTEGER ScreenSize       ;        Comment Terminal depth             ;
 INTEGER A, B             ;        Comment Global station addresses   ;

 INTEGER I   ,                     Comment Information             ;
         RR  ,                     Comment Reaoy-to-Receive        ;
         REJ ,                     Comment Reject                  ;
         RNR ;                     Comment Not Reaoy-to-Receive;

 BOOLEAN PrettyPics         ;      Comment True displays interactions;
 BOOLEAN DisplayPictures    ;      Comment True displays to user     ;
 REF ( OUTFILE ) ScreenEcho;       Comment File of interactions      ;
 BOOLEAN HardCopy           ;      Comment True recoros interactions ;
 REF ( OUTFILE ) ReportFile;       Comment File for the final summary;

 REF ( Transmitter        ) ATransmitterRef,        BTransmitterRef       ;
 REF ( PacketDistributer  ) APacketdistributerRef, BPacketDistributerRef;
REF  ( TimeOut1             ) ATimeOut1Ref,          BTimeOut1Ref          ;
REF  ( TimeOut2             ) ATimeOut2Ref,          BTimeOut2Ref          ;
REF  ( PacketReceiver       ) APacketReceiverRef,    BPacketReceiverRef    ;

REF  ( HDLCStation) StationA, StationB;

Comment:******** CLASS HDLCStation ********************************;

  CLASS HDLCStation ( Address ); INTEGER Address;
  BEGIN
  Comment  An HDLC station is responsible for receiving data  in
  packets and transmitting information in frames to the opposite
  station.  The station ensures that when observed from a higher
  layer the data link  appears  to be  capable  of  transmitting
  frames in an orderly and secure way with no loss,  duplication
  or missequencing occuring in spite of  occasional transmission
  errors at the physical level.;

  REF  ( HEAD ) BufferIN    ;     Comment Arriving packets in this Q;
  REF  ( HEAD ) BufferOUT   ;     Comment Leaving packets in this Q ;
  REF  ( HEAD ) ReTransmitQ;      Comment Frames to be retransmitted;
  REF  ( HEAD ) UnAckedQ    ;     Comment Frames to be acknowledged ;

  REAL TransmissionRate;          Comment Line transmission speed   ;
  REAL PropagationDelay;          Comment One way delay             ;
  INTEGER Overhead      ;         Comment flags,fcs,address,control;
  INTEGER Seed          ;         Comment Random number seed        ;

  REAL ArrivalRate   ;            Comment Mean interarrival rate of packets;
  REAL DistributeRate;            Comment Mean rate of packet distibution  ;
  REAL BER           ;            Comment Bit Error Rate ;
  REAL T1            ;            Comment Timer          ;
  REAL T2            ;            Comment Secondary Timer;
  INTEGER WindowSize ;            Comment Window Size    ;

  INTEGER VS,                     Comment Send state variable          ;
          VR;                     Comment Receive state variable       ;
  INTEGER NextToBeAcked;

  BOOLEAN TransmitRej     ,       Comment A REJ needs to be transmitted;
          RejTransmitted  ,       Comment A REJ has been transmitted   ;
          TransAck        ,       Comment Transmit an Acknowledgement  ;
          SrcnSink        ;       Comment Source and Sink specified    ;

  BOOLEAN TransmitRnr     ,       Comment A RNR needs to be transmitted;
          RnrTransmitted  ,       Comment A RNR has been transmitted   ;
          SendBufferFull  ;       Comment Output is temporarily blocked;

  INTEGER MaxFrameSize,           Comment Max size of the frame data field;
          BufferINMax ,           Comment Max size of BufferIN Q          ;
          BufferOUTMax;           Comment Max size of BufferOUT Q         ;

  REF  ( INFILE   ) SourceFile;   Comment File of and data to be sent;
  REF  ( OUTFILE  ) SinkFile   ;  Comment File of received data      ;

  INTEGER SFramesTransmitted,     Comment No. of supervisory frames sent;
          IFramesTransmitted,     Comment No. of information frames sent;
          FramesRetransmitted,    Comment No. of frames retransmitted   ;
          IFramesReceived    ;    Comment No. of frames received        ;
  REAL    WaitingTimes,          Comment Total time BufferIn->UnAckedQ ;
          TransferTimes;         Comment Total time UnAckedQ->BufferOUT;


  BEGIN
  TEXT Source, Sink;

  BufferIN     :- NEW HEAD;
  BufferOUT    :- NEW HEAD;
  UnAckedQ     :- NEW HEAD;
  ReTransmitQ  :- NEW HEAD;

  VR :=  VS :=  0;
  NextToBeAcked := 0;
  TransmitRej :=  TransAck := RejTransmitted := FALSE;
  TransmitRnr :=  RnrTransmitted := SendBufferFull := FALSE;
  SFramesTransmitted  := IFramesTransmitted := 0;
  FramesRetransmitted := IFramesReceived    := 0;
  WaitingTimes := TransferTimes := 0.0;

  QueryUser( Address, TransmissionRate, PropagationDelay, OverHead, Seed,
             BER,  WindowSize, MaxFrameSize, ArrivalRate, DistributeRate,
             T1, T2, BufferINMax, BufferOUTMax, SrcnSink, Source, Sink );

  HomeCursor;
  IF SrcnSink THEN
    BEGIN
    SourceFile :- NEW INFILE  ( Source );
    SinkFile   :- NEW OUTFILE ( Sink   );
    SourceFile.OPEN ( BLANKS (80) );
    SinkFile.OPEN   ( BLANKS (80) );
    END;
  DETACH;
  END;
END process HDLCStation;

Comment:******** LINK CLASS WaitingFrame ************************;

LINK CLASS WaitingFrame;
  BEGIN
  Comment A copy of the essential information for each frame transmitted
          waits in UnAckedQ and RetansmitQ.;
  REAL    ArrivalTime;         Comment The time frame first arrives in Q ;
  INTEGER SendVar    ;         Comment SV or V(S)- uniquely defines frame;
  TEXT    SendMessage;         Comment The data being transmitted        ;
  END waitingframe   ;

Comment:******** LINK CLASS Packet ******************************;

  LINK CLASS Packet;
  BEGIN
  Comment Packets arrive from Layer 3 are enclosed with overhead bits
          to produce frames.  Transmitted as frames and   distributed
          to Layer 3 as packets. ;
  REAL ArrivalTime;            Comment Time packet first arrives in Q ;
  TEXT Message    ;            Comment Data to send/receive           ;
  END packet      ;
```

```
Comment:******** PROCESS CLASS TimeOut1 ****************************;

PROCESS CLASS TimeOut1( TransmitterRef , UnAckedQ, RetransmitQ );
  REF ( Transmitter ) TransmitterRef ;
  REF ( HEAD        ) UnAckedQ, RetransmitQ ;
  BEGIN
  Comment  This is a more robust method of protecting frames. Each
           time a frame is transmitted a timer is restarted.    If
           the timer expires then this process is activated  which
           ensures the frame is retransmitted.;
  WHILE TRUE DO
    BEGIN
    Merge ( UnAckedQ, RetransmitQ );
    IF TransmitterRef.IDLE THEN ACTIVATE TransmitterRef DELAY(0.0);
    PASSIVATE;
    END;
  END timeout;

Comment:******** PROCESS CLASS TimeOut2 *************************;

PROCESS CLASS TimeOut2( S );
  REF ( HDLCStation ) S;
  BEGIN
  Comment  The timeout associated with this timer is started when an
           acknowledgement needs to be sent. If a packet arrives and
           is transmitted the timer is  cancelled  and   the   reply
           is  piggybacked  otherwise  this timeout is activated and
           the ack sent via an RR-frame.;
  WHILE TRUE DO
    BEGIN
    S.Transack := TRUE;
    IF ( S.Address = A ) THEN
      BEGIN
      IF ATransmitterRef.IDLE THEN ACTIVATE ATransmitterRef;
      END
      ELSE
      IF BTransmitterRef.IDLE THEN ACTIVATE BTransmitterRef;
    PASSIVATE;
    END;
  END timeout2;

Comment:******** PROCESS CLASS Merge *****************************;

PROCEDURE Merge ( UnAckedQ, RetransmitQ );
  REF ( HEAD ) UnAckedQ, RetransmitQ;
  Comment   A timeout or receiving a REJ-frame causes all frames
            waiting to be acknowledged to be retransmitted.  The
            RetransmitQ becomes the UnAckedQ concatenated   with
            the RetransmitQ.;
  BEGIN
  REF ( WaitingFrame ) WF1, WF2;
  WF1 :- UnAckedQ.LAST ;
  WHILE WF1 =/= NONE DO
    BEGIN
    WF1.OUT;
    WF2 :- ReTransmitQ.FIRST;
    IF WF2 == NONE
      THEN WF1.INTO ( ReTransmitQ )
      ELSE WF1.PRECEDE( WF2 );
    WF1 :- UnAckedQ.LAST ;
    END while;
  END merge;

Comment:******** PROCEDURE StartTimer1 ***************************;

PROCEDURE StartTimer1 ( S );
  REF ( HDLCStation ) S;
  Comment  Start primary timer, by setting a timeout in the future;
  BEGIN
  IF ( S.Address = A )
    THEN REACTIVATE ATimeOut1Ref DELAY S.T1
    ELSE REACTIVATE BTimeOut1Ref DELAY S.T1;
  END starttimer1;

Comment:******** PROCEDURE StartTimer2 ***************************;

PROCEDURE StartTimer2 ( S );
  REF ( HDLCStation ) S;
  Comment  Start secondary timer, by setting a timeout in the future;
  BEGIN
  IF ( S.Address = A )
    THEN ACTIVATE ATimeOut2Ref DELAY S.T2
    ELSE ACTIVATE BTimeOut2Ref DELAY S.T2;
  END starttimer2;

Comment:******** PROCEDURE HomeCursor ****************************;

PROCEDURE HomeCursor;
  Comment  Clear screen, home cursor;
  BEGIN
  OUTCHAR( CHAR( 26 ) );
  OUTIMAGE;
  END homecursor;

Comment:******* PROCEDURE Increment ******************************;

PROCEDURE Increment ( V ); NAME V;
  INTEGER V;
  BEGIN
  Comment  Increment V in the range 0..SeqNoMax;
  IF V < SeqNoMax THEN V := V + 1
                  ELSE V := 0;
  END increment;

Comment:**********************************************************
*                                                                *
*                    PROCESS CLASS Transmitter                   *
*                                                                *
*****************************************************************;

PROCESS CLASS Transmitter( BufferIN, UnAckedQ, ReTransmitQ, TimeOut2Ref,
                           PacketDistributerRef, TransmitterRef,     S );

  REF ( HEAD               ) BufferIN, UnAckedQ, RetransmitQ  ;
  REF ( TimeOut2           ) TimeOut2Ref                      ;
  REF ( PacketDistributer  ) PacketDistributerRef;
  REF ( Transmitter        ) TransmitterRef                   ;
  REF ( HDLCStation        ) S                                ;
  BEGIN
  INTEGER Seed;
  Comment  This process is responsible for sending all frames.  The
           emission of frame lasts a simulated time dependent  upon
           the link characteristics and the length of the frame.
           Emission also determines whether one or more bit  errors
           has occured.  Frames waiting to  be  retransmitted   are
           given priority over packets waiting to be transmitted in
           UnAckedQ;

BOOLEAN PROCEDURE PrOfErrorFree( L, BER );
  INTEGER L;  Comment Length of frame being transmitted;
  REAL BER ;  Comment Probability of a bit error        ;
  Comment  This procedure simulates the use of a cyclic redundancy
           check, by returning false if one or more bit errors will
           occur during transmission.  For reasonable    BER and L
           the   poisson   distribution  can  be  approximated   by
           ( 1 - BER ) ** L;
  PrOfErrorFree := DRAW( ( 1-BER )**L, Seed );

PROCEDURE HoldIN ( SV, Txt, ATime );
  INTEGER SV; TEXT Txt; REAL ATime;
  BEGIN
  Comment  Hold the essential parts of transmitted frame in UnAckedQ;
  REF ( WaitingFrame ) WF;
  WF :- NEW WaitingFrame;
  WF.SendVar      := SV;
  WF.ArrivalTime  := ATime;
  WF.SendMessage  :- BLANKS( 1 );
  WF.SendMessage  := Txt;
  WF.INTO ( UnAckedQ );
  END holdin;

PROCEDURE Emission ( FrameType,SendVar,ReceiveVar,Data,FCS,INTime );
  INTEGER FrameType, SendVar, ReceiveVar;
  TEXT Data;
  Boolean FCS;
  REAL INTime;
  BEGIN
  Comment  Send a frame from this station to the other.  The time
           to send the frame is dependent upon the length of  the
           frame and the transmission rate.  The station can only
           transmit one frame at a time, but may transmit  whilst
           a frame is being propagated between stations.
           A supervisory frame is assumed to be of size overhead.;
  REAL EmissTime;
  EmissTime := S.Overhead/S.TransmissionRate;
  IF FrameType = I THEN
    EmissTime := EmissTime + S.MaxFrameSize/S.TransmissionRate;

  IF ( S.Address = A ) THEN
    INSPECT StationB DO
        BEGIN
        HOLD( EmissTime );
        ACTIVATE NEW
          FrameReceiver ( FrameType,SendVar,ReceiveVar,Data,
          FCS,    INTime,   BufferOUT,   BPacketDistributerRef,
          UnackedQ,    BPacketReceiverRef,        BTimeOut1Ref,
          BTransmitterRef,    RetransmitQ,            StationB)
          DELAY  S.PropagationDelay;
          END;

   IF ( S.Address = B ) THEN
     INSPECT StationA DO
          BEGIN
          HOLD( EmissTime );
          ACTIVATE NEW
            FrameReceiver ( FrameType,SendVar,ReceiveVar,Data,
            FCS,    INTime,    BufferOUT, APacketDistributerRef,
            UnAckedQ,    APacketReceiverRef,       ATimeOut1Ref,
            ATransmitterRef,      RetransmitQ,         StationA)
          DELAY S.PropagationDelay;
          END;
   END emission;

 PROCEDURE Transmit;
   BEGIN
   Comment  Take the packet from BufferIN, start the  primary  timer,
            place a copy of the frame to be transmitted into UnAckedQ
            and transmit.;
   REF ( Packet ) Txt;
   INTEGER SVS, SVR;
   SVS := S.VS; SVR := S.VR;
   IF S.TransAck THEN S.TransAck := FALSE;
   IF NOT TimeOut2Ref.IDLE THEN CANCEL( TimeOut2Ref );
   Txt :- BufferIN.FIRST;
   Txt.OUT;
   StartTimer1( S );
   HoldIn (   SVS, Txt.Message, Txt.ArrivalTime );
   Increment  ( S.VS );
   S.WaitingTimes := S.WaitingTimes + ( TIME - Txt.ArrivalTime );
   S.IFramesTransmitted := S.IFramesTransmitted + 1;
   IF PrettyPics THEN PrettyPic( "Transmitter" ,S.Address , I );
   Emission ( I ,   SVS,   SVR, Txt.Message,
     PrOfErrorFree( S.MaxFrameSize+S.Overhead, S.BER ),Txt.ArrivalTime);
   END transmit;

 PROCEDURE Retransmit;
   BEGIN
   Comment  Take the packet from BufferIN, start the primary timer,
            place a copy of the frame  to  be  retransmitted   into
            UnAckedQ and retransmit.;
   REF ( WaitingFrame ) WF;
   IF S.TransAck THEN S.TransAck := FALSE;
   IF NOT TimeOut2Ref.IDLE THEN CANCEL( TimeOut2Ref );
   WF :- ReTransmitQ.FIRST;
   WF.OUT;
   StartTimer1( S );
   HoldIn ( WF.SendVar, WF.SendMessage, WF.ArrivalTime );
   S.FramesRetransmitted := S.FramesRetransmitted + 1;
   IF PrettyPics THEN PrettyPic( "Transmitter" ,S.Address , I );
   Emission ( I, WF.SendVar, S.VR, WF.SendMessage,
     PrOfErrorFree( S.MaxFrameSize+S.Overhead, S.BER ),WF.ArrivalTime);
   END of retransmit;

 PROCEDURE Transsup ( FrameType, Condition );
   NAME Condition; INTEGER FrameType; BOOLEAN Condition;
   BEGIN
```

```
Comment  Start the primary timer and transmit a supervisory frame;
IF FrameType = REJ THEN S.RejTransmitted := TRUE;
IF FrameType = RNR THEN S.RnrTransmitted := TRUE;
StartTimer1( S );
Condition := NOT Condition;
S.SFramesTransmitted := S.SFramesTransmitted + 1;
IF PrettyPics THEN PrettyPic( "Transmitter" ,S.Address ,FrameType);
Emission ( FrameType, 0, S.VR, NOTEXT,
  PrOfErrorFree( S.Overhead, S.BER ), 0.0 );
END;

Seed := S.Seed;
WHILE TRUE DO
  BEGIN

  IF ( S.TransmitRej ) AND ( NOT S.RejTransmitted )
    AND ( NOT S.Transmitrnr )
  Comment { if we must send a reject frame and we haven't yet then..};
    THEN Transsup ( REJ, S.TransmitRej )

  ELSE
  IF ( S.TransmitRnr ) AND ( NOT S.RnrTransmitted )
  Comment { if we must send an rnr frame and we haven't yet then.. };
    THEN Transsup ( RNR, S.TransmitRnr )

  ELSE
  IF ( ReTransmitQ.CARDINAL > 0 ) AND ( NOT S.SendBufferFull )
  Comment { if there is a frame in the retransmission Q and    }
          { the station we're sending to is not blocked then..};
    THEN Retransmit

  ELSE
  IF ( BufferIN.CARDINAL > 0 ) AND ( UnAckedQ.CARDINAL < S.WindowSize
    AND ( NOT S.SendBufferFull )
  Comment { if there is a frame to transmit and we }
          { havent got "window size" outstanding   }
          { frames and the station we're sending   }
          { to is not blocked then...              };
    THEN Transmit

  ELSE
  IF ( S.TransAck )
  Comment { if we have to send an acknowledgement and }
          { all other tests have failed wait for   T2 }
          { to piggyback the reply if it still hasn't }
          { been sent then send a supervisory frame   };
    THEN Transsup ( RR, S.TransAck );

  IF NOT
    ((( RetransmitQ.CARDINAL > 0 ) AND (NOT S.SendBufferFull) )
    OR
    ( ( BufferIN.CARDINAL > 0 ) AND ( UnAckedQ.CARDINAL < S.WindowSize
        AND ( NOT S.SendBufferFull ) )
    OR ( ( S.TransmitRej) AND ( NOT S.RejTransmitted ) AND
         ( NOT S.TransmitRnr ) )
    OR ( ( S.TransmitRnr) AND ( NOT S.RnrTransmitted ))
    OR S.TransAck )
    THEN PASSIVATE;
```

```
   END while true;
  END transmitter;

Comment:*****************************************************************
*                                                               *
*                 PROCESS CLASS FrameReceiver                   *
*                                                               *
*****************************************************************;

PROCESS CLASS FrameReceiver ( FrameType, NS, NR, Txt, FCS  ,
  ArrivalTime,  BufferOUT,  PacketDistributerRef,  UnackedQ,
  PacketReceiverRef, TimeOut1Ref, TransmitterRef, RetransmitQ, S);

  INTEGER FrameType, NS, NR; REAL ArrivalTime; TEXT Txt; BOOLEAN FCS;
  REF ( TimeOut1          ) TimeOut1Ref                         ;
  REF ( Transmitter       ) TransmitterRef                      ;
  REF ( HEAD              ) BufferOUT, UnackedQ, RetransmitQ;
  REF ( PacketReceiver    ) PacketReceiverRef                   ;
  REF ( PacketDistributer ) PacketDistributerRef                ;
  REF ( HDLCStation       ) S                                   ;
  BEGIN
  Comment  This process is responsible for  receiving   all  frames,
           removing acknowledged frames from the  UnAckedQ,  placing
           frames into RetransmitQ if a REJ or RNR frame is received
           and notifying the transmitter if a reply must be sent.;
  INTEGER ARRAY Seq( 0:SeqNoMax );

  PROCEDURE ReceiveFrame ( Txt ); TEXT Txt;
    BEGIN
    Comment  Accept the frame and place it into BufferOUT as a
             packet ready to be dispersed to Layer 3.;
    IF ( BufferOUT.CARDINAL < S.BufferOUTMax ) THEN
      BEGIN
      REF ( Packet ) TxtReceived;
      S.TransmitRnr := S.RnrTransmitted := FALSE;
      Increment( S.VR );
      StartTimer2( S );
      TxtReceived :- NEW Packet ;
      TxtReceived.Message :- BLANKS ( 1 );
      TxtReceived.Message := Txt;
      TxtReceived.INTO ( BufferOUT );
      IF ( S.Address = A ) THEN
        StationB.TransferTimes := StationB.TransferTimes
          + ( TIME - ArrivalTime )
        ELSE
        StationA.TransferTimes := StationA.TransferTimes
          + ( TIME - ArrivalTime );
      S.IFramesReceived := S.IFramesReceived + 1;
      IF PacketDistributerRef.IDLE THEN  ACTIVATE PacketDistributerRef;
      END
      ELSE S.TransmitRnr := TRUE;
    END receiveframe;

   PROCEDURE RemoveAcknowledgedFrames ( QRef, NR );
    REF ( HEAD ) QRef; INTEGER NR;
    BEGIN
    Comment  Remove all frames that have been acknowledged from the
             UnAckedQ.;
    REF ( WaitingFrame ) WF;
   WF :- QRef.FIRST;
   WHILE ( ( WF=/=NONE ) ANDTHEN ( WF.SendVar<>NR ) ) DO
     BEGIN
     WF.OUT;
     WF :- QRef.FIRST;
     END;
   END  removeacknowledgedframes;

 PROCEDURE InitSeq;
   BEGIN
   Comment  Set the array seq to represent the order of the frames
            as they appear in UnAckedQ.;
   INTEGER WindowOrder, WindowVar;
   REF ( WaitingFrame ) WF;
   WF :- UnAckedQ.FIRST;
   IF WF == NONE
     THEN WindowVar := 0
     ELSE WindowVar := WF.SendVar;
   FOR WindowOrder := 0 STEP 1 UNTIL SeqNoMax DO
     BEGIN
     Seq( WindowVar ) := WindowOrder;
     Increment( WindowVar );
     END;
   END;

 IF FCS THEN
   BEGIN
   IF FrameType = I THEN
     BEGIN
     IF NS = S.VR THEN
       BEGIN
       ReceiveFrame ( Txt );
       S.TransmitRej := S.RejTransmitted :=  FALSE;
       END
       ELSE S.TransmitRej := TRUE;
     END if then if then;

   Comment { Make use of any acknowledgement };
   InitSeq;

   IF ( Seq( S.NextToBeAcked ) < Seq( NR ) ) AND
      ( Seq( NR )                  <= Seq( S.VS ) )
     THEN BEGIN
     RemoveAcknowledgedFrames ( UnAckedQ, NR );
     S.NextToBeAcked := NR;
     IF PacketReceiverRef.IDLE THEN ACTIVATE PacketReceiverRef;
     IF ( Seq( S.NextToBeAcked ) < Seq( S.VS ) )
       THEN StartTimer1 ( S )
       ELSE CANCEL ( TimeOut1Ref );
     END;

   IF ( FrameType = REJ ) AND
     ( Seq( S.VS ) > Seq( S.NextToBeAcked ))
     THEN  BEGIN
     CANCEL (TimeOut1Ref);
     Merge( UnAckedQ, RetransmitQ );
     END ;

   IF ( FrameType = RNR ) THEN
```

```
      BEGIN
      CANCEL ( TimeOUT1Ref );
      Merge( UnAckedQ, RetransmitQ );
      S.SendBufferFull := TRUE;
      END ;

    IF ( FrameType = RR ) AND ( S.SendBufferFull )
      THEN S.SendBufferFull := FALSE;

    IF PrettyPics AND ( FrameType <> I ) THEN
      PrettyPic( "FrameReceiver" ,S.Address, FrameType );

    IF TransmitterRef.IDLE THEN ACTIVATE TransmitterRef;

    END of FCS if FCS indicates an error then ignoring
        the incoming frame effectively discards    it;

  END  framereceiver;

Comment:***************************************************************
*                                                                     *
*                  PROCESS CLASS PacketReceiver                       *
*                                                                     *
***************************************************************;

  PROCESS CLASS PacketReceiver ( BufferIN, TransmitterRef, S );
    REF ( HEAD          ) BufferIN       ;
    REF ( Transmitter ) TransmitterRef;
    REF ( HDLCStation ) S                ;

    BEGIN
    INTEGER Seed;
    Comment  This process  is  responsible  for placing incoming
             packets into BufferIN and notifying the transmitter
             of a  packets   arrival.  It  continues   receiving
             packets from Layer 3 and placing them into BufferIN
             until the max  buffer  space  allocated by the user
             is reached.;

    PROCEDURE ReceivePacket;
      BEGIN
      Comment  Place a new packet into BufferIN. ;
      REF ( Packet ) Txt;
      Txt :- NEW Packet;
      Txt.Message :- BLANKS (1);
      Txt.ArrivalTime := TIME;
      IF S.SrcnSink
        THEN Txt.Message := S.SourceFile.INTEXT (1)
        ELSE Txt.Message := "*";
      Txt.INTO ( BufferIN );
      IF PrettyPics THEN PrettyPic( "ReceivePacket" ,S.Address , 0 );
      END receivepacket;

    Seed := S.Seed;
    WHILE ( IF S.SrcnSink THEN NOT S.SourceFile.LASTITEM ELSE TRUE ) DO
      BEGIN
      IF ( BufferIN.CARDINAL < S.BufferINMax ) THEN
        BEGIN
        HOLD( NEGEXP( 1/S.Arrivalrate, Seed ) );
        ReceivePacket;
        IF TransmitterRef.IDLE THEN  ACTIVATE TransmitterRef;
        END
        ELSE PASSIVATE;
      END ;
    END packetreceiver;

Comment:*************************************************************
*                                                                   *
*                PROCESS CLASS PacketDistributer                    *
*                                                                   *
*******************************************************************;

PROCESS CLASS PacketDistributer ( S );
  REF ( HDLCStation ) S;
  BEGIN
  INTEGER Seed;
  Comment  This process is responsible for placing incoming frames that
         have been stripped of  their overhead  bits  into  BufferOUT,
         where  it takes care of dispersing these packets to the layers
         above.   Packets are placed   BufferOUT   until the max buffer
         space  allocated  by  the  user is  reached.   An RNR frame is
         sent to the opposite station when BufferOUT is blocked.;

  PROCEDURE DistributePacket;
   BEGIN
   Comment Dispose of a packet  from   BufferOUT,   either from
          a specified source file or to the garbage collector.;
   REF ( Packet ) Txt;
   Txt :- S.BufferOUT.FIRST;
   Txt.OUT;
   IF S.SrcnSink THEN S.SinkFile.OUTTEXT ( Txt.Message );
   IF ( S.BufferOUT.CARDINAL = ( S.BufferOUTMax - 1 ) )
     THEN StartTimer2( S );
   IF PrettyPics THEN PrettyPic( "DistributePacket" ,S.Address , 0 );
   END  distributepacket;

  Seed := S.Seed;
  WHILE TRUE DO
    BEGIN
    IF ( S.BufferOUT.CARDINAL > 0 ) THEN
      BEGIN
      DistributePacket;
      HOLD( NEGEXP( 1/S.DistributeRate, Seed ) );
      END
      ELSE PASSIVATE;
    END while ..do;
  END  of packetdistributer;

Comment:******* PROCEDURE PrettyPic ***************************;

PROCEDURE PrettyPic( EventText, Address, FrameType );
  NAME EventText; TEXT EventText; INTEGER Address, FrameType;
  BEGIN
  Comment  This procedure displays the current state of the major
          variables  belonging  to StationA and StationB.    The
          display can  be sent  to the terminal to give the user
          a step by  step account of what   is happening, and/or
          sent to a  file specified   by   the user which can be
```

```
              reviewed later.
              Significant events such as a  transmission   receiving
              frames, the arrival and  distribution of packets, only
              will trigger the display.;

    REF ( WaitingFrame ) UQRef, RQRef  ;
    REF ( Packet       ) INRef, OUTRef ;
    INTEGER J;

    PROCEDURE InitScreen;
      BEGIN
Screen(00):="Time Now (secs):             Finish At (secs):                   ";
Screen(01):="Event:                                                           ";
Screen(02):="                                                                 ";
Screen(03):="         Station A         A - B          Station B              ";
Screen(04):="         =========                        =========              ";
Screen(05):="                                                                 ";
Screen(06):="BufferIN     [                    BufferIN    [                  ";
Screen(07):="                                                                 ";
Screen(08):="BufferOUT    [                    BufferOUT   [                  ";
Screen(09):="                                                                 ";
Screen(10):="UnAckedQ     [                    UnAckedQ    [                  ";
Screen(11):="                                                                 ";
Screen(12):="                                                                 ";
Screen(13):="ReTransQ     [                    ReTransQ    [                  ";
Screen(14):="                                                                 ";
Screen(15):="                                                                 ";
Screen(16):="VS=       VR=                     VS=      VR=                   ";
Screen(17):="                                                                 ";
Screen(18):="                                                                 ";
Screen(19):="                                                                 ";
      END of initscreen;

    PROCEDURE GlobalStats;
      BEGIN
      Comment  Display the simulation time now and when due to finish.;
      PlotReal( 17, 00, 9, TIME );
      PlotReal( 45, 00, 9, SimTime );
      IndicateEvent;
      END;

    PROCEDURE LocalStats( VS, VR, Xpos ); INTEGER VS, VR, Xpos;
      BEGIN
      Comment  Display VS and VR of each station.;
      PlotInt( Xpos+4, 16, 2, VS );
      PlotInt( Xpos+14,16, 2, VR );
      END;

    Comment { x,y are coordinates, w is width for the text area };

    PROCEDURE PlotChar ( x, y, c ); VALUE c; INTEGER x, y; TEXT c;
      Screen( y ).SUB( x, 1 ) := c;

    PROCEDURE PlotInt  ( x, y, w, n ); INTEGER x, y, w, n ;
      Screen( y ).SUB( x, w ).PUTINT( n );

    PROCEDURE PlotReal  ( x, y, w, n ); REAL n; INTEGER x, y, w;
      Screen( y ).SUB( x, w ).PUTFIX( n, 3);
```

```
PROCEDURE PlotText  ( x, y, w, t );VALUE t; INTEGER x, y, w; TEXT t;
  Screen( y ).SUB( x, w ) := t;

PROCEDURE PlotFrameType( x, y, FrameType );
  INTEGER x, y, FrameType;
  BEGIN
  PlotText( x,  y, 20, "Frame Type = " );
  IF ( FrameType =I  ) THEN PlotText(x+13,  y, 8, "I"  );
  IF ( FrameType =RR ) THEN PlotText(x+13,  y, 8, "RR" );
  IF ( FrameType =RNR) THEN PlotText(x+13,  y, 8, "RNR");
  IF ( FrameType =REJ) THEN PlotText(x+13,  y, 8, "REJ");
  END of plotframetype;

PROCEDURE IndicateEvent;
  BEGIN
  Comment  Indicate which event triggered the display.;
  IF ( EventText = "ReceivePacket" )
    THEN PlotText( 20, 1, 30, "Packet Arrived, BufferIN" );
  IF ( EventText = "DistributePacket" )
    THEN PlotText( 20, 1, 34, "Packet Distributed, BufferOUT");
  IF ( EventText = "Transmitter" ) THEN
    BEGIN
    PlotText( 20, 1, 30, "Frame Transmitted" );
    PlotFrameType( 38, 01, FrameType );
    END;
  IF ( EventText = "FrameReceiver" )
    THEN PlotText( 20, 1, 30, "Frame Received" );
  IF ( EventText = "Finished" )
    THEN PlotText( 20, 1, 30, "Finished" );

  IF ( EventText = "ReceivePacket"    ) OR
     ( EventText = "DistributePacket" ) THEN
    BEGIN
    IF ( Address = A ) THEN PlotChar( 27, 3, "<" )
                       ELSE PlotChar( 29, 3, ">" );
    END;
  IF ( ( EventText = "Transmitter"   ) AND ( Address = A ) ) OR
     ( ( EventText = "FrameReceiver" ) AND ( Address = B ) )
     THEN PlotChar( 29, 3, ">" );
  IF ( ( EventText = "Transmitter"   ) AND ( Address = B ) ) OR
     ( ( EventText = "FrameReceiver" ) AND ( Address = A ) )
     THEN PlotChar( 27, 3, "<" );
  IF ( EventText = "FrameReceiver" )  THEN
     PlotFrameType( 36, 01, FrameType );
  END;

PROCEDURE PlotQs( Xpos ); INTEGER Xpos;
  BEGIN
  Comment  Display the contents of all the queues.;
  INTEGER x;
  x := Xpos;
  WHILE INRef =/= NONE DO
    BEGIN
    PlotChar( x, 06, INRef.Message );
    INRef :- INRef.SUC; x := x+1;
    END;
  x := Xpos;
  WHILE OUTRef =/= NONE DO
    BEGIN
```

```
      PlotChar( x, 08, OUTRef.Message );
      OUTRef :- OUTRef.SUC; x := x+1;
      END;
   x := Xpos;
   WHILE UQRef =/= NONE DO
      BEGIN
      PlotInt( x, 10, 1, UQRef.SendVar );
      PlotChar(x, 11, UQRef.SendMessage );
      UQRef :- UQRef.SUC; x := x +1;
      END;
   x := Xpos;
   WHILE RQRef =/= NONE DO
      BEGIN
      PlotInt( x, 13, 1, RQRef.SendVar );
      PlotChar(x, 14, RQRef.SendMessage );
      RQRef :- RQRef.SUC; x := x +1;
      END;
   END of plotqs;

PROCEDURE OutDisplayPictures;
   BEGIN
   Comment Display information to the user.;
   Comment: Home cursor, leave screen;
   OUTCHAR ( CHAR(30) );
   FOR J := 0 STEP 1 UNTIL ScreenSize DO
      BEGIN
      OUTTEXT( Screen(J) );
      OUTCHAR( CHAR(27 )) ;  Comment: ESC T                    ;
      OUTCHAR( CHAR(84) ) ;  Comment: Pad  the line with blanks ;
      OUTIMAGE;
      END;
   END outdisplaypictures;

PROCEDURE OutHardCopy;
  BEGIN
  Comment Write information to a trace file.;
  ScreenEcho.OUTIMAGE;
  FOR J := 0 STEP 1 UNTIL ScreenSize DO
     BEGIN
     ScreenEcho.OUTTEXT( Screen(J) );
     ScreenEcho.OUTIMAGE;
     END;
  END outhardcopy;

 InitScreen;
 GlobalStats;

 INSPECT StationA DO
   BEGIN
   INRef :- BufferIn.FIRST; OUTRef :- BufferOUT.FIRST  ;
   UQRef :- UnAckedQ.FIRST; RQRef  :- ReTransmitQ.FIRST;
   PlotQs( 13 );
   LocalStats( VS, VR, 0 );
   END;

 INSPECT StationB DO
   BEGIN
   INRef :- BufferIn.FIRST; OUTRef :- BufferOUT.FIRST  ;
   UQRef :- UnAckedQ.FIRST; RQRef  :- ReTransmitQ.FIRST;
```

```
    PlotQs( 44 );
    LocalStats( VS, VR, 31 );
    END;

    IF DisplayPictures THEN OUTDisplayPictures;
    IF HardCopy THEN OutHardCopy;

  END of prettypic;


Comment********* PROCEDURE Report ********************************;

PROCEDURE Report;
  BEGIN
  Comment  Write the final summary to a file defined by the user.;

  INTEGER LMarge,     Comment Left margin ;
          TitleLen,   Comment Title Length;
          AStart,     Comment StationA data starts in this coloumn;
          BStart ;    Comment StationB data starts in this coloumn;

  PROCEDURE EnterReportInt( Title, Aval, Bval );
    NAME Title; Text Title; INTEGER Aval, Bval;
    BEGIN
    ReportFile.IMAGE.SUB ( LMarge, TitleLen ) := Title;
    ReportFile.IMAGE.SUB ( AStart, 10       ).PUTINT( Aval );
    ReportFile.IMAGE.SUB ( BStart, 10       ).PUTINT( Bval );
    ReportFile.OUTIMAGE;
    END;

  PROCEDURE EnterReportRel( Title, Aval, Bval, Dec );
    NAME Title; Text Title; REAL Aval, Bval; REAL Dec;
    BEGIN
    ReportFile.IMAGE.SUB ( LMarge, TitleLen ) := Title;
    ReportFile.IMAGE.SUB ( AStart, 10       ).PUTFIX( Aval , Dec );
    ReportFile.IMAGE.SUB ( BStart, 10       ).PUTFIX( Bval , Dec );
    ReportFile.OUTIMAGE;
    END;

  ReportFile.OPEN ( BLANKS(80) );
  LMarge := 5;
  TitleLen := 40;
  AStart := 50;
  BStart := 63;

  ReportFile.OUTIMAGE;
  ReportFile.IMAGE.SUB( 30, 50 ) := "SUMMARY";
  ReportFile.OUTIMAGE;
  ReportFile.IMAGE.SUB( 30, 50 ) := "=======";
  ReportFile.OUTIMAGE;
  ReportFile.OUTIMAGE; ReportFile.OUTIMAGE;

  ReportFile.IMAGE.SUB( 5, 25 )  := "Simulation Length(secs)";
  ReportFile.IMAGE.SUB( 30,10 ).PUTFIX( SimTime, 1);
  ReportFile.OUTIMAGE;

  ReportFile.IMAGE.SUB( 5, 25 ) := "Max Sequence Number";
  ReportFIle.IMAGE.SUB( 30,10 ).PUTINT( SeqNoMax );
  ReportFile.OUTIMAGE;
ReportFile.OUTIMAGE; ReportFile.OUTIMAGE;

ReportFile.IMAGE.SUB( 55,10 ) := "A -> B";
ReportFile.IMAGE.SUB( 68,10 ) := "A <- B";
ReportFile.OUTIMAGE;
ReportFile.IMAGE.SUB( 55,10 ) := "======";
ReportFile.IMAGE.SUB( 68,10 ) := "======";
ReportFile.OUTIMAGE;

EnterReportRel( "Transmission Rate",
                    StationA.TransmissionRate,
                    StationB.TransmissionRate, 1    );
EnterReportRel( "Propagation Delay",
                    StationA.PropagationDelay,
                    StationB.PropagationDelay, 3    );
EnterReportInt( "Overhead",
                    StationA.Overhead,
                    StationB.Overhead               );
EnterReportRel( "Bit Error Rate",
                    StationA.BER,
                    StationB.BER, 7                 );
EnterReportInt( "Window Size",
                    StationA.WindowSize,
                    StationB.WindowSize );
EnterReportInt( "Max Information Field Size",
                    StationA.MaxFrameSize,
                    StationB.MaxFrameSize           );
EnterReportRel( "Arrival Rate of Packets ",
                    StationA.ArrivalRate,
                    StationB.ArrivalRate, 3         );
EnterReportRel( "Distribution Rate of Packets",
                    StationA.DistributeRate,
                    StationB.DistributeRate, 3      );
EnterReportRel( "Primary Timer (secs)",
                    StationA.T1,
                    StationB.T1, 3                  );
EnterReportRel( "Secondary Timer(secs)",
                    StationA.T2,
                    StationB.T2, 3                  );
EnterReportInt( "Max Size of BufferIN",
                    StationA.BufferINMax,
                    StationB.BufferINMax            );
EnterReportInt( "Max Size of BufferOUT",
                    StationA.BufferOUTMax,
                    StationB.BufferOUTMax           );

ReportFile.OUTIMAGE; ReportFile.OUTIMAGE;
EnterReportInt( "No. of Supervisory Frames Transmitted",
                    StationA.SFramesTransmitted,
                    StationB.SFramesTransmitted     );
EnterReportInt( "No. of Information Frames Transmitted",
                    StationA.IFramesTransmitted,
                    StationB.IFramesTransmitted     );
EnterReportInt( "No. of Information Frames ReTransmitted",
                    StationA.FramesRetransmitted,
                    StationB.FramesRetransmitted    );
EnterReportInt( "No. of Information Frames Received",
                    StationA.IFramesReceived,
                    StationB.IFramesReceived        );
```

```
  EnterReportRel( "Mean Waiting Time",
    IF ( StationA.IFramesTransmitted <> 0 ) THEN
      StationA.WaitingTimes / StationA.IFramesTransmitted ELSE 0,
    IF ( StationB.IFramesTransmitted <> 0 ) THEN
      StationB.WaitingTimes / StationB.IFramesTransmitted ELSE 0, 3 );
  EnterReportRel( "Mean Transfer Time",
    IF ( StationB.IFramesReceived <> 0 ) THEN
      StationA.TransferTimes / StationB.IFramesReceived ELSE 0,
    IF ( StationA.IFramesReceived <> 0 ) THEN
      StationB.TransferTimes / StationA.IFramesReceived ELSE 0, 3   );

  ReportFile.CLOSE;

  END report;

Comment:********* PROCEDURE InitProcesses **********************;

PROCEDURE InitProcesses;
  BEGIN

  INSPECT StationA DO
    BEGIN
    ATimeOut2Ref          :- NEW TimeOut2( StationA );
    APacketDistributerRef :- NEW PacketDistributer( StationA );
    ATransmitterRef       :- NEW Transmitter
      ( BufferIn,   UnAckedQ,   ReTransmitQ,
        ATimeOut2Ref, APacketDistributerRef,
        ATransmitterRef, StationA                   );
    ATimeOut1Ref          :- NEW TimeOut1
      ( ATransmitterRef, UnackedQ, RetransmitQ   );
    APacketReceiverRef    :- NEW PacketReceiver
      ( BufferIN, ATransmitterRef, StationA      );
    ACTIVATE APacketReceiverRef;
    END;

  INSPECT StationB DO
    BEGIN
    BTimeOut2Ref          :- NEW TimeOut2( StationB );
    BPacketDistributerRef :- NEW PacketDistributer ( StationB );
    BTransmitterRef       :- NEW Transmitter
      ( BufferIn,   UnAckedQ,   ReTransmitQ,
        BTimeOut2Ref, BPacketDistributerRef,
        BTransmitterRef, StationB                   );
    BTimeOut1Ref   :- NEW TimeOut1
      ( BTransmitterRef, UnackedQ, RetransmitQ   );
    BPacketReceiverRef    :- NEW PacketReceiver
      ( BufferIN, BTransmitterRef, StationB      );
    ACTIVATE BPacketReceiverRef;
  END ;
  END initrefs;

Comment:******** PROCEDURE QueryUser ****************************;

PROCEDURE QueryUser(  Ad, Tr, Pd, Oh, Sd, Br,
                      Ws, Mf, Ar, Dr, T1,
                      T2, Bn, Bo, SS, So, Sk );
  NAME Tr, Pd, Oh, Sd, Br, Ws, Mf, Ar,
       Dr, T1, T2, Bn, Bo, SS, So, Sk;
  TEXT So, Sk;
  REAL Tr, Pd, Br, Ws, Mf, Ar, Dr, T1, T2, Bn, Bo;
  INTEGER Ad, Oh, Sd;
  BOOLEAN  Ss;
  BEGIN
  Comment  Ask the user for the values of these paramters.;
  OUTTEXT( "Initialise Station " );
  IF ( Ad = A ) THEN OUTTEXT( "A." )
                     ELSE OUTTEXT( "B." );
  OUTIMAGE; OUTIMAGE;
  OUTTEXT( "Transmission Rate(9600.0bps) : !0!" ); OUTIMAGE;
  INIMAGE; Tr := INREAL; IF ( Tr=0.0 ) THEN Tr := 9600.0;
  OUTTEXT( "Propagation Delay(0.0secs)    : !0!" ); OUTIMAGE;
  INIMAGE; Pd := INREAL; IF ( Pd=0.0 ) THEN Pd := 0.0;
  OUTTEXT( "No. of Overhead Bits(40)      : !0!" ); OUTIMAGE;
  INIMAGE; Oh := ININT; IF ( Oh=0 ) THEN Oh := 40;
  OUTTEXT( "Seed(12345)              : !0!" ); OUTIMAGE;
  INIMAGE; Sd := ININT; IF ( Sd=0 ) THEN Sd := 12345 ;
  OUTTEXT( "Bit Error Rate (0.0) : !0!" ); OUTIMAGE;
  INIMAGE; Br := INREAL; IF (Br=0.0) THEN Br := 0.0;
  OUTTEXT( "Window Size(7)           : !0!" ); OUTIMAGE;
  INIMAGE; Ws := ININT; IF (Ws=0) THEN Ws := 7;
  OUTTEXT( "Max Information field size(1024)   : !0!" ); OUTIMAGE;
  INIMAGE; Mf := ININT; IF (Mf=0) THEN Mf := 1024;
  OUTTEXT( "Inter-Arrival rate of packets(0.1)       : !0!" );
  OUTIMAGE; INIMAGE; Ar := INREAL; IF (Ar=0.0) THEN Ar := 0.1;
  OUTTEXT( "Inter-Distribution rate of packets(0.1) : !0!" );
  OUTIMAGE;  INIMAGE; Dr := INREAL; IF (Dr=0.0) THEN Dr := 0.1;
  OUTTEXT( "The primary timer T1(1.0sec)    : !0!" ); OUTIMAGE;
  INIMAGE; T1:= INREAL; IF (T1=0.0) THEN T1:= 1.0;
  OUTTEXT( "The secondary timer T2(0.0sec) : !0!" ); OUTIMAGE;
  INIMAGE; T2:= INREAL; IF (T2=0.0) THEN T2:= 0.0;
  OUTTEXT( "Max number of  packets in BufferIN(15)   : !0!" );
  OUTIMAGE; INIMAGE; Bn := ININT; IF (Bn=0) THEN Bn := 15;
  OUTTEXT( "Max number of  packets in BufferOUT(15) : !0!" );
  OUTIMAGE; INIMAGE; Bo := ININT; IF (Bo=0) THEN Bo := 15;
  OUTTEXT( "Do you want a Source and Sink for Packets ? (NO) : !0!");
  OUTIMAGE; INIMAGE; IF ( ININT=0 ) THEN Ss := FALSE ELSE Ss := TRUE;
  IF Ss THEN
    BEGIN
    So :- BLANKS( 16 ); Sk :- BLANKS( 16 );
    OUTTEXT( "Enter the source file name : !0!" ); OUTIMAGE;
    INIMAGE; So := INTEXT( 16 );
    OUTTEXT( "Enter the sink file name   : !0!"    ); OUTIMAGE;
    INIMAGE; Sk := INTEXT( 16 );
    END
  END of queryuser;

Comment:******** PROCEDURE Initialise ***************************;

PROCEDURE InitGlobalVars;
  BEGIN
  Comment  Initialise the variable common to both stations.;
  INTEGER J;
  TEXT FileName;
  I :=   1;
  RR :=  2;
  RNR := 3;
  REJ := 4;
  A := 0;
```

```
   B := 1;
   ScreenSize := 19;
   HomeCursor;
   FOR J := 0 STEP 1 UNTIL ScreenSize DO Screen(J):-BLANKS( 60 );
   OUTTEXT( "Initialise variables for both stations." ); OUTIMAGE;
   OUTTEXT( "Type '0' to obtain the default value '( )':" ); OUTIMAGE;
   OUTIMAGE;  OUTIMAGE;
   OUTTEXT( "Max Sequence Number(7)   : !0!" ); OUTIMAGE;
   INIMAGE; SeqNoMax := ININT; IF ( SeqNoMax=0 ) THEN SeqNoMax := 7;
   OUTTEXT( "Simulation Length(1sec) : !0!" ); OUTIMAGE;
   INIMAGE; SimTime := INREAL; IF ( SimTime=0.0 ) THEN SimTime := 1.0;
   OUTTEXT( "Do you want a display of the interactions (NO)   : !0!");
   OUTIMAGE; INIMAGE;
   IF ( ININT = 0)
     THEN Displaypictures := FALSE
     ELSE DisplayPictures :=TRUE;
   OUTTEXT( "Do you want a hardcopy of the interactions (NO) : !0!" );
   OUTIMAGE; INIMAGE;
   IF ( ININT = 0 ) THEN HardCopy := FALSE ELSE HardCopy := TRUE;
   IF HardCopy THEN
     BEGIN
     FileName :- BLANKS( 16 );
     OUTTEXT( "What do you want to call this file ? : !0!"); OUTIMAGE;
     INIMAGE; FileName := INTEXT( 16 );
     ScreenEcho :- NEW OUTFILE( FileName );
     ScreenEcho.OPEN ( BLANKS( 80 ) );
     END;
   OUTIMAGE;
   FileName :- BLANKS( 16 );
   OUTTEXT( "What do you want to call the report file ? : !0!"); OUTIMAGE;
   INIMAGE; FileName := INTEXT( 16 );
   ReportFile :- NEW OUTFILE ( FileName );
   HomeCursor;
   PrettyPics := DisplayPictures OR HardCopy;
   END initialise;

Comment:******** PROCEDURE CloseFiles ***************************;

PROCEDURE CloseFiles;
   BEGIN

   IF StationA.SrcnSink THEN
   INSPECT StationA DO
     BEGIN
     SourceFile.CLOSE; SinkFile.CLOSE;
     END;

   IF StationB.SrcnSink THEN
   INSPECT StationB DO
     BEGIN
     SourceFile.CLOSE; SinkFile.CLOSE;
     END;

   IF PrettyPics THEN PrettyPic( "Finished", 0, 0 );
   IF HardCopy THEN ScreenEcho.CLOSE;

   END of tidy;
```

```
Comment:*****  MAIN BODY  ****************************************;

InitGlobalVars;

StationA :- NEW HDLCStation( A );
StationB :- NEW HDLCStation( B );

InitProcesses;

HOLD ( SimTime );

Report;

CloseFiles;

END of simulation;
END of model;
```