RELIABLE SATELLITE BROADCAST TRANSMISSION: PROTOCOL DESIGN, SIMULATION AND ANALYSIS

Ivan Zvonko Jutrisa

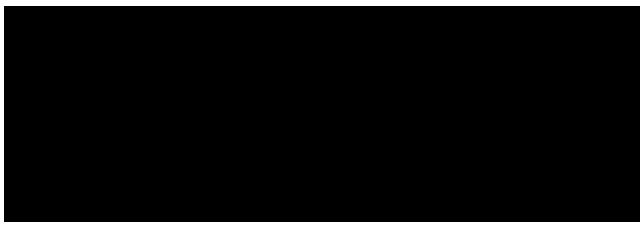
This thesis is presented in fulfilment of
the requirements of the degree of
Doctor of Philosophy

School of Communications and Informatics
Faculty of Engineering and Science
Victoria University of Technology
Melbourne

August 1998

DECLARATION

I declare that, to the best of my knowledge, the research described herein is the result of my own work, except where otherwise stated in the text. This thesis is submitted in fulfilment of the candidature of the degree of Doctor of Philosophy of Victoria University of Technology, Australia. No part of this thesis has been submitted previously for this or any other degree.



Ivan Zvonko Jutrisa

August 1998

ABSTRACT

This dissertation presents research carried out in the development of the Reliable Satellite Broadcast Transmission (RSBT) protocol. The RSBT design was largely based on the data dissemination requirements of Australia's Bureau of Meteorology, and on Optus Communications' Omnicast Digital satellite service.

A literature survey did not uncover a reliable broadcasting/Multicasting protocol for either satellite or terrestrial transmission. Thus, protocols commonly used for point-to-point satellites transmission were analysed—mainly to identify protocol procedures that inhibit good performance over a satellite link.

The RSBT protocol, which was formally specified and verified with SDL (Specification and Description Language), differs greatly to protocols such as TCP and HDLC. It does not use addressing—the target satellite service provides a permanent connection. It does not use flow control—all frames are transmitted contiguously. The frames have two Frame Check Sequence (FCS) fields—header FCS and data FCS. Receivers send minimal acknowledgments, which is particularly important in broadcast transmission. And retransmission, which is based on Selective Repeat, is done after all frames have been sent once.

RSBT experiments, which were largely based on three network sizes coupled with five error rates (15 combinations), were carried out with mathematical and simulation models. The performance results (broadcast) clearly surpassed the results of all other protocols (in point-to-point satellite transmission) found in the literature. This vindicates the decision to tailor the RSBT protocol specifically for satellite transmission.

In conclusion, the RSBT protocol was successfully developed. More importantly though, the protocol is very efficient.

DEDICATION

This thesis is dedicated to my wife **Til (Matilda)**. From start to finish, Til was supportive and understanding; she continuously encouraged me; and she assisted me in whatever way she could. Til's support made life a great deal easier and alleviated many problems. Her support was most appreciated, and I would thus like to say **Thank You**.

ACKNOWLEDGMENTS

It is with great relief that I write this part of the thesis—it has been a long road home, with many highs and lows. The difficulties—and there were many—were greatly alleviated by many people. I am very grateful for the assistance I received, and thus, I would like to say **Thank You** to the following people:

Assoc. Prof. Dr. Nalin Sharda, my Supervisor, for all of his time and effort. Paradoxically, I found the advice on relatively simple issues the most useful; in particular, advice on the project scope provided a good focus, and advice on writing style was most useful for obvious reasons.

Assoc. Prof. Dr. Peter Cerone, my co-supervisor, for his invaluable assistance with the mathematical aspects of my thesis, and for the constructive criticism and advice throughout my candidature. Peter also put me in contact with several people who are very knowledgeable in my field of research, and thus were able to provide me with valuable information.

Mike Hassett, Superintendent Communications at the Bureau of Meteorolgy, for initiating the interesting project, and for his and effort.

Dr. Neil Diamond, Dr. Alasdair McAndrew, Mr. Ian Gomm, Mr. Mehmet Tat and **Mr. Fernando Scarmozzino** (mathematics/statistics staff in the department) for the help with mathematical problems and proof reading of various works. The assistance was appreciated; however, I am most grateful for the warm and welcoming manner in which these people offered their time. This was epitomised by **Dr. Neil Diamond** who, without hesitation, offered to gloss over my entire thesis as a final check.

Assoc. Prof. Dr. Neil Barnett, Head of School, for supporting me in other ways. I would also like to extend my gratitude to all staff members and fellow research students for the warm and friendly atmosphere they created.

Josee Fortin and **Alfred Gagne**, Canadian Meteorological Centre, for generously providing me with valuable information about their systems.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AAL5	ATM Adaption Layer protocol 5
ACK	Positive Acknowledgment
ARQ	Automatic Repeat Request
ANSI	American National Standards Institute
ASCII	American Standard Code for Information Interchange
ATM	Asynchronous Transmission Mode
BER	Bit Error Rate
Bps	Bits Per Second
CASE	Computer Aided Software Engineering
CMC	Canadian Meteorological Centre
CODE	Co-operative Data Experiment
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
DFD	Data Flow Diagram
EBCDIC	Extended Binary Coded Decimal Interchange Code
FCS	Frame Check Sequence
FDT	Formal Description Technique
FEC	Forward Error Correction
FER	Frame Error Rate
FTP	File Transfer Protocol
HDLC	High-level Data Link Control
ID	Identification
I/O	Input/Output
IP	Internet Protocol
IPX	Internet Packet Exchange
I-TCP	Indirect TCP
Kbits	Kilobits
Kbps	Kilobits Per Second
LAN	Local Area Network
LAPB	Link Access Procedure, Balanced

Mbits	Megabits
Mbps	Megabits Per Second
Mbytes	Megabytes
METSIS	Meteorological Satellite Information System
MHz	Megahertz (frequency—cycles per second)
MIPS	Million Instructions Per Second
NAK	Negative Acknowledgment
NCP	Netware Core Protocol
NETBLT	Network Bulk Transfer
OSI	Open Systems Interconnection
PE	Processing Element
POP	Packet Oriented Protocol
QFTP	Quick File Transfer Protocol
QPSK	Quadrature Phase Shift Keying
RACE	Research in Advanced Communications in Europe
RAM	Random-access Memory
RFC	Request For Comments
RPC	Remote Procedure Call
RSBT	Reliable Satellite Broadcast Transmission
SD	Storage Device
SDL	Specification and Description Language
SNR	A high-speed transport protocol named after its authors
SSCOP	Service Specific Connection Oriented Protocol
TCP	Transmission Control Protocol
TD	Transfer Device
UDP	User Datagram Protocol
VMTP	Versatile Message Transaction Protocol
WAN	Wide Area Network
XTP	Express Transfer Protocol

1. INTRODUCTION

Many large organisations disseminate information from their head office to regional offices. Existing transmission media and services provide these organisations with options that best suit their needs. In most cases, the selected media or service is sufficient for the requirements. In some cases though, tight constraints (time, quality, etc.), and variables such as file size, may render existing systems inefficient.

The need for Reliable Satellite Broadcast Transmission (RSBT) was first mooted by the Australian Bureau of Meteorology. The Bureau collects weather data from all over the country and processes it at its head office in Melbourne. The resulting data (several types) are then compressed, and disseminated to all six Regional Offices (one in each state capital city) via terrestrial links [1]. Currently, the Bureau is using leased lines with HDLC (High-level Data Link Control) connections between Melbourne and the six Regional Offices; data is disseminated by a series of unicast transmissions.

The usefulness of weather data and its derivatives deteriorates rapidly with time. Therefore, data processed centrally must be transmitted without delay for the Regions to take full advantage of the information. For example, satellite data, which is given high priority, should be available in Regional Offices within two to three minutes of the end of central processing. Delays in transmission are inevitable due to the large amount of data currently available and the projected increase in weather data over the next five years [2]. Hence, a new method of disseminating data is required.

A potential alternative method for broadcasting data to Regional Offices is a new satellite service offered by Optus Communications called Omnicast Digital. Omnicast Digital is a one-way broadcasting service, which transmits data at rates

between 64 Kbps and 2 Mbps in a continuous flow [3]. Preliminary analysis based on information from the Bureau of Meteorology and Optus Communications, suggest that Omnicast Digital could prove to be more efficient, reliable and cheaper than the currently used terrestrial links.

A communication protocol for the effective use of the Omnicast Digital service is not available from Optus or any other vendor. To explore the possibilities of using the Omnicast Digital service, the Bureau of Meteorology needs to develop a suitable protocol that includes return links for messages such as acknowledgments. The successful development of RSBT would satisfy the Bureau's requirements, and provide it with a viable alternative for disseminating data.

1.1 Aim

The aim of this research project is to develop a communication protocol for reliable satellite broadcast transmission—RSBT. The main objectives are to:

- develop a viable and efficient alternative for disseminating data by tailoring the protocol for Omicast Digital transmission or a similar satellite service, avoiding the shortcomings of generic protocols when used for satellite transmission;
- develop a protocol conversion mechanism for FTP (File Transfer Protocol) , which will interface with RSBT;
- derive various implementation options;
- compare the options;
- formally specify at least one implementation option: the most realistic or suitable option;
- develop a mathematical model and a simulation model, and carry out experiments;
- compare the mathematical model results and the simulation results against each other and against results given in the literature for other protocols.

1.2 Research Approach

The first step in the development of RSBT was to study several other protocols to obtain a clear understanding of their operation. The aspects of these protocols that are suitable for Omnicast Digital transmission were noted, as were the shortcomings. The RSBT protocol was then designed and formally specified. The formal specifications served as a basis and a verification aid for a mathematical model and a simulation model. The mathematical model results and the simulation results were analysed and the RSBT transmission performance was compared to the performance of other protocols. Finally, a discussion was carried out and conclusions were drawn. Details of the research procedure are described in the following chapters, as outlined below.

Chapter 2, "Literature Review", presents a summary of the literature on protocols and methods used in reliable satellite and terrestrial transmission. A specific example of a similar data dissemination operation performed by the Canadian Meteorological Centre is described. Papers on various protocols are outlined, focusing on the procedures used for data transmission; the suitability and shortcomings of these procedures for satellite transmission are discussed.

Chapter 3, "Protocol Description", describes the RSBT protocol and its design fundamentals. The description includes the purpose of the protocol, the environment in which it will operate, the frame format, and the transmission procedures including error control. Error control makes up the bulk of this chapter, describing and comparing four alternatives for the acknowledgment process.

Chapter 4, "SDL Formal Specification", provides an introduction to SDL (Specification and Description Language), and then describes the SDL specification of the RSBT protocol. Only the higher level (abstract) specifications are given in this chapter; the complete specifications are given in Appendix A.

Chapter 5, "Mathematical Modelling", describes the mathematical model of the transmission process step-by-step. The mathematical model provides the expected values for, inter alia, the number of retransmissions, the time required for the transmission, and the resulting throughput.

Chapter 6, "Network II.5 Modelling", describes the development of the Network II.5 simulation model. First, the communication hardware is modelled to set up the environment in which the RSBT protocol will operate. Then the instructions, and the software modules that execute them, are modelled in accordance to the formal specifications given in Appendix A.

Chapter 7, "Experimentation and Results Analysis", presents a plan for the mathematical model and simulation experiments. The plan discusses the parameters that were used by in the experiments and the effects of the assumptions that were made. A summary of the results is presented; the full mathematical model results, and simulation results are given in Appendix B and Appendix C respectively. The results are analysed and compared with satellite transmission results of other protocols found in the literature.

Chapter 8, "Discussion", presents a discussion with supporting arguments for the main points and/or methods used in each of the preceding chapters. This chapter also includes a section on future research—extensions to the research presented in this dissertation.

Chapter 9, "Conclusion", draws a conclusion on the overall success of this research project, with respect to the initial aim (see section 1.1). Summaries of Chapter 8 provide support for the conclusion and show that the objectives listed on section 1.1 have been fulfilled.

2. LITERATURE REVIEW

2.1 Introduction

Although there has been much work done in the area of data communications, there has been no serious work done on reliable broadcasting or multicasting [4]. This statement is supported by Aylott [5], who said , "current protocols and equipment for multipoint services are very immature". Hence, no literature was found on broadcast/multicast transmission that uses reverse error control and retransmits lost frames. Some research on transmission in a continuous flow was found. Continuous transmission avoids the delays incurred by waiting for acknowledgments; these delays are prevalent in satellite transmission. Komisarczuk [6], and Clark [7] discussed two protocols that can transmit data (point-to-point) in a virtually continuous flow; they also mention possibilities for broadcasting, but without feedback from receivers. Similarly, Irani [8], like many others, considers multicasting (broadcasting to selected receivers), but only for video services and teleconferencing which do not require retransmission of lost packets. Most significantly though, Deering [9] presents the Multicasting Backbone, but again, reliability is not provided. The first step in filling this void was to analyse some of the current methods used in reliable satellite transmission. The analysis was used to identify deficiencies, extract transmission aspects suitable for the proposed RSBT protocol requirements, and to note test results for comparisons with RSBT.

2.2 The Canadian Meteorological Centre

The Canadian Meteorological Centre (CMC) [10],[11] is an organisation similar to the Australian Bureau of Meteorology in its functions and the problems faced in executing these functions. CMC is currently using two satellite systems to disseminate weather data: METSIS and ANIKOM 100. METSIS is used to deliver meteorological products to the CMC regional offices over four 56 Kbps

channels. Much of the information transmitted over the individual channels is specific for a particular region of the country; there is some overlap in the information required by the regions, including some information that is required by all receivers. Receivers monitor only one or two channels and capture the information selectively; obviously, weather products that are required by all receivers must be broadcast over more than one channel. ANIKOM 100 is a much slower (4.8 Kbps) and cheaper system used to cater for the aviation community and to provide third party users with general information. In addition, the ANIKOM 100 system is used as a backup for the METSIS system.

METSIS has three uplink sites and about 70 receiving sites. All earth station LANs, uplinks and receiving sites, run TCP/IP (Transmission Control Protocol/Internet Protocol) on an Ethernet backbone, and are linked together on a terrestrial WAN. A weather product that needs to be disseminated is transferred by FTP over the WAN to a METSIS uplink, where it is converted to the METSIS protocol format and broadcast over the satellite. The product is also stored on disk for possible retransmission. Receivers convert the data back to FTP/TCP format and transmit it to the appropriate workstation on the LAN. Retransmission requests are received through the Network Management System and are given a lower priority than regular product broadcasts; in fact retransmissions may be sent over the terrestrial WAN. The METSIS protocol frame headers include:

- **Flag.** This indicates whether a Frame Check Sequence (FCS) field is used.
- **Frame Length.**
- **Frame Type.** This may be either control or data, and includes:
  - **Product Identification.** This provides information about the product originator, product ID, retention time, timestamps, etc.
  - **Product Definition.** This provides information about the product format, resolution of graphical products, etc.
  - **Data.**
  - **End of Product.** This indicates the end of transmission.

The control information is transmitted in control frames preceding data transmission. In the event that a control frame is lost, recovery may not be possible, and the entire file may be lost. When the loss of an important product becomes obvious to a meteorologist at a receiving site, a manual request is made for a retransmission.

The main problem with the transmission method employed by CMC is that error control is not an integral part of the broadcast transmission scheme. If errors are detected by the system, the retransmission request and the retransmission are treated as separate functions. In the scenario where the control frames are lost—and thus the complete product—additional delays are incurred in waiting for the meteorologist to spot the non-reception of a product. The RSBT design will aim to overcome shortfalls of the METSIS system.

2.3 Network Block Transfer protocol

Clark [7] presents a revised version of the Network Block Transfer (NETBLT) protocol, which was originally developed in RFC 969 [12]. NETBLT is a reliable transport layer protocol intended for rapid transfer of large volumes of data. The revision was aimed at improving performance over long delay, high bandwidth satellite channels, and to provide a higher throughput than that of other protocols. To achieve these goals, the revision focused on minimising the effects of the satellite propagation delay, network congestion, and packet loss; most of the changes relate to timers used with multiple data buffers

The NETBLT protocol file transfer is done as a series of buffer transmissions, with each buffer containing a number of data packets. Once the sending client provides the NETBLT module with enough data to fill one buffer, NETBLT breaks up the buffer into packets and transmits it to the receiver, which stores it in a matching buffer. When all packets belonging to the buffer have been received, the receiver transmits a control packet containing a list of corrupted packets that need to be retransmitted. Once the retransmissions have been received correctly, the data is passed on to the higher layers and the sender is

notified that a new buffer is ready to receive data. The sender then sends the next buffer in the same way. When the whole file has been transmitted, with no outstanding retransmission or acknowledgments, the connection is closed. The transmission procedure thus far is similar to stop-and-wait automatic repeat request (ARQ), except that it is done one-buffer-at-a-time instead of one-packet-at-a-time. The transmission efficiency is improved by the use of multiple buffers. Thus, when the sender has completed the transmission of one buffer and is awaiting acknowledgment, it begins the transmission of another. If NAKs for the previous buffer are received, the sender retransmits the individual packets when the current buffer has been transmitted. The use of buffers in this way, makes the packet flow essentially continuous. During transmission the NETBLT module has only one buffer, and copies of the transmitted buffers are not kept for retransmission purposes, if needed, they are recopied from the client's main buffer. Although the client can not release the buffer storage as the data is given to the NETBLT buffers, this has apparently not been a problem in preliminary implementations.

NETBLT uses two flow control methods simultaneously. The first method is used at the packet level. The sender transmits a burst of packets over a negotiated time interval, which ensures that there are no packet overflow losses at the receiving site or at any intermediate node. At a higher level, NETBLT uses a modified sliding window to control the flow of buffer transmissions, allowing only the negotiated number of buffers to be open for transmission at any one time.

The buffer size and the packet size are negotiated at connection set up. All packets must be the same size except for the last packet in the buffer, which is identified by a flag in the header. The packet size should be large so that the overheads are minimised. However, the packet size should be kept small enough to avoid internetwork fragmentation. As with the packets, all buffers must be the same size except for the last buffer, which is identified by a "last-buffer" flag in the header of each packet belonging to the buffer. The buffer size too, should be

as large as possible to minimise the number of buffer transmissions, and thus improve the performance of the protocol.

The NETBLT transmission process is not that much different from that of the TCP or HDLC protocols, except that the sliding window works at the buffer level and Selective Repeat retransmission is used instead of Go-back-N. The use of multiple buffers allows an almost break-free transmission because a much greater volume of data may be awaiting acknowledgment, and thus, the effect of the satellite propagation delay is minimised. However, the use of multiple buffers is not much different to the expansion of the TCP sliding window, except that the NAKs are sent in blocks (one block per buffer). The question must be asked: why do the NAKs have to be sent in blocks. Certainly, NAKs sent in blocks would result in less return traffic and less processing. However, it is questionable whether or not these gains outweigh the NAK delays, which would be prominent on high error rate links. If the NAKs were sent individually, the sender could complete the buffer retransmissions sooner; the situation where each buffer needs only a few packets retransmitted, would not arise as often. Another aspect of the NETBLT transmission process that is questionable, is the way in which the multiple buffers are used at the sending site. It would seem more appropriate for the NETBLT module to maintain the multiple buffers, and thereby minimise the need to transfer data from the main buffer to the NETBLT buffer.

NETBLT is a lightweight protocol, which was designed to make best use of new high-speed networks, and thus to produce high throughput. Examples of other lightweight protocols with the same objective as NETBLT, include: SNR (named after its designers) [13],[14],[15], VMTP (Versatile Message Transaction Protocol) [16], and XTP (Express Transfer Protocol) [17],[18].

2.4 Novell's Burst Mode Protocol

Morgan [19] and Thomas [20] describe Novell Netware's Burst Mode protocol—an application layer protocol developed to improve transmission efficiency over WANs. The Burst Mode protocol improvements are based on

multiple buffering and a dynamic sliding window. The improvements in transmission efficiency are generally greater with complex and faster networks, and with larger transfer volumes. In low volume transmissions, where the data can fit into a single packet, the Burst Mode protocol can actually have a negative effect on transmission efficiency.

In Netware's conventional protocol stack, application level requests are carried out by the Netware Core Protocol (NCP). The NCP packets are encapsulated within the Internet Packet Exchange (IPX) packets for transmissions between a workstation and server. IPX is unreliable, and therefore, NCP performs its own error control, which is based on stop-and-wait Automatic Repeat Request (ARQ), i.e. a packet must be acknowledged before the next packet is sent. This transmission method is acceptable in LANs because of the transmission speed and low error rates. However, stop-and-wait ARQ is not well suited to WAN transmissions because of the inherent delays, particularly over a satellite link.

The Burst Mode protocol overcomes Netware's shortfalls in WAN transmission by providing a dialogue for IPX/NCP via an adaptive, self-tuning flow control mechanism. To transmit a file, the client can read blocks of up to 64 Kbytes. The Burst Mode protocol partitions the buffered data into IPX packets and transmits these packets contiguously (in a burst). The IPX WAN transmission does not require any acknowledgments until after the last packet in the burst has been transmitted. Unlike the TCP sliding window protocol, which uses Go-back-N retransmission, the Burst Mode protocol resends only those packets that were lost in the transmission. The Selective Repeat retransmission substantially improves transmission efficiency over long delay links. Although the burst size is negotiated during connection set up, the number of packets transmitted in a burst is varied dynamically. If one or more packets are lost in a transmission burst, the sender reduces the following buffer (burst size) by 1/8 at a time. Similarly, if no packets are being lost the buffer is progressively increased by 100 bytes up to a maximum of 64 Kbytes. The amount of buffer memory available on the sender, receiver, and any intermediate nodes dictates the actual maximum burst size.

Tests performed by INFOLAN [19] with and without the Burst Mode protocol, produced throughput results seven times higher with the Burst Mode protocol. The effective throughput was as high as 85% (47.6 Kbps on a 56 Kbps channel). Transmission of a 500 Kbyte file between Brussels and Los Angeles took 104 seconds with the Burst Mode protocol and 832 seconds without it. Generally, a 300% improvement was expected, but the particular INFOLAN configuration used in this case produced a much better result. INFOLAN tests carried out on a LAN also produced throughput between 86% and 95% better with the Burst Mode protocol. Tests in [20] produced up to 350% better throughput with the Burst Mode protocol. The best results were achieved with a large packet size, large window size, and faster channels.

The Burst Mode protocol is very similar to the NETBLT protocol in terms of the flow control mechanism and the use of multiple buffers; both protocols reduce the sender's idle periods, i.e. awaiting acknowledgments over long delay links. However, the protocol description in [6],[19],[20] provided only general information on the Burst Mode protocol, and more detailed information could not be obtained from Novell. Thus, it is not clear how the buffers are used when retransmissions are required. That is, does the Burst Mode protocol retransmit only those packets that were lost in the previous transmission, or, is the buffer topped up with fresh data packets to maintain a consistent burst size? Nevertheless, the Burst Mode protocol with its use of multiple buffers, should in most cases improve transmission efficiency. However, the efficiency gains are minimal when channels slower than 64 Kbps are used. This is because the transmission serialisation delay and Burst Mode's high processing overheads become the dominant factors. A shortcoming of the Burst Mode protocol is its eagerness to change the burst size; if a burst has just one error the burst size is still reduced, and if the following burst has no errors the burst is increased again. These constant changes contribute to the high overheads and promote small burst sizes, even with reasonable error rates.

2.5 NetX Satellite Transmission

Fairhurst [21] describes some aspects of the NetX simulation package. NetX was developed to investigate the real-time performance of low capacity satellite circuits when used with the X.25 LAPB protocol. POP (Packet-Oriented Protocol), which is an extension to the NetX package, allows real-time transmission of HDLC frames over an actual satellite link, or over a hardware simulated satellite link.

The system configuration consists of a Sun workstation (NetX host), a Macintosh PC (satellite interface), and two modems. The Sun transmits POP packets to the Macintosh, which converts the packets to HDLC frames and sends them to a modem, which in turn transmits the frames to the satellite. Another modem receives the satellite transmission and sends it to the same Macintosh, which converts the HDLC frames back to POP packets and transmits them to the Sun workstation. It was found that for many low capacity satellite circuits, the link performance is dominated by the distribution of errors. This lead to the development of a number of alternative error control schemes, which have improved transmission efficiency.

This paper focused on the NetX simulation package and the model building process rather than the protocols used. Thus, the main point of interest is the system configuration, which includes a satellite interface for the purpose of controlling the satellite transmission and performing the protocol conversion. A similar configuration is envisaged for the RSBT implementation.

2.6 TCP

TCP [22] was the first transport layer protocol available for the connectivity of heterogeneous LANs. Moreover, it is still the most popular protocol used for internetworking [23]; commercial users did not migrate to OSI protocols as much as expected [24].

Although TCP is very widely implemented and is generally a very good protocol for LAN transmission, it is not well suited to satellite transmission for two main reasons. First, the TCP sliding window size is limited, and when it is used over a satellite link it results in a stop-start transmission[6],[24]. That is, when the sender has transmitted the allowable amount of data, it must wait for an acknowledgment before it can transmit any more data. The problem is the propagation delay associated with satellite transmission. And second, TCP is not suitable for satellite transmission because of the Go-back-N retransmission strategy [25]. Go-back-N works well on LANs because the processing requirements are simpler and the data throughput rates are high. However, retransmitting data that may have been received correctly the first time is very inefficient over long delay links. In addition to the above mentioned shortcomings, the TCP performance is adversely affected by the high processing overheads [26], and the high frame header overheads (20 bytes minimum) [22]. Moreover, the TCP checksum is not very reliable, although this is not a serious problem if a protocol with strong error detection such as HDLC is used at the data link layer [23].

Due to the fact that TCP is widely used, and yet very inefficient over a satellite link [6],[27],[28], much research has been carried out on strategies for improving TCP over wireless networks [24],[28],[29],[30],[31],[32].

2.7 Reliable Broadcasting Across ATM Networks

Although a protocol for reliable broadcasting/multicasting has not been found in the literature, alternative methods for delivering data to multiple receivers have been found. Research on communications issues in parallel computing on ATM networks [33], present three methods of multicasting. These are: 'Separate Addressing', 'Multicast Tree', and 'Hardware Delivery'. Separate addressing and multicast tree methods use the AAL5 protocol [34] and are thus reliable. Hardware delivery however, requires a higher layer protocol to implement reliability.

Separate addressing is the simplest method, where the sender transmits packets to each receiver individually. Parallel processing enables the sender to transmit to several receivers simultaneously, and the method works well with small networks. With large networks however, bottlenecks build up at the sender's network interface and the system becomes inefficient. The multicast tree approach is more efficient than separate addressing and reduces the network traffic. With a multicast tree, the sender transmits packets only to a subset of receivers, who in turn transmit to another subset, and so on until all receivers have been reached.

The best performance though, was achieved by hardware delivery. With this method, the sender transmits packets to a switch that in turn delivers the packets to all receivers simultaneously. Test results show hardware delivery completing a 32 Kbyte multicast to two receivers in less than half the time used by the multicast tree method, and in about one fifth of the time for a multicast to eight receivers. Hardware delivery however does not provide reliability; receivers can not request retransmission of packets from the hardware switch. To overcome this shortfall, the authors developed a minimal user-level reliable multicast transport service. Although the details of the multicast service were not provided, a diagram indicated that retransmission followed the same path. However, the reliability provided in this scheme is questionable, as transmission times were almost the same for varying number of receivers—two to eight. Perhaps the network error rate was too small to show any distinct difference with a 32 Kbyte transmission. Also, the cyclic redundancy check (CRC) used by the ATM protocols provides some forward error correction, which would further reduce the retransmission requirements.

The system configuration used with the hardware delivery is similar to the configuration envisaged for RSBT, except that a dedicated node serving as a satellite interface would be used for broadcasting rather than switching. In addition, the satellite interface would deal with the retransmission requests, instead of the transmission originator.

2.8 Satellite Transmission Experiments

The growing popularity of satellite based networks indicates that wireless links will play an important role in future internetworks [28]. Hence, many research reports can be found on analysis of protocol performance over satellite links, along with strategies for improving the performance of commonly used protocols such as TCP. The following subsections summarise just a few of these research reports.

2.8.1 TCP in Satellite Transmission

Kruze [24] presents a number of experimental findings regarding the efficiency of the TCP/IP protocol when used over a satellite link at transmission speeds of up to 1.5 Mbps. The results show that even with the maximum TCP window size of 24 Kbyte, the throughput tops out at about 320 Kbps regardless of the channel speed. The experiments are supported by a theoretical model, which also produced a throughput of 1,343 Kbps with a window size of 95 Kbytes. TCP's 'Slow-start' congestion control mechanism and windowing flow control were shown to be the main reasons for the poor performance over a satellite link. The Slow-start algorithm only allows gradual expansion of the window, and is thus the cause of the very low throughput in the early stages of the transmission (first 100 Kbytes or so).

2.8.2 TCP and Burst Mode in RACE Experiments

Experiments in [6] analyse the performance of various protocols in connecting LANs over an ATM/satellite link. These experiments are part of the RACE (Research in Advanced Communications in Europe) Catalyst project. The simulated network used a 10.4 Mbps satellite channel and a cell error rate better than 10^{-6}. Five protocols were simulated; Novell's Burst Mode protocol produced the best throughput result—2.4 Mbps. TCP with an expanded window—64 Kbytes—achieved the second best throughput—560 Kbps. The throughput that was achieved with these two protocols, was possible mainly because the protocols permitted large volumes of data to be transmitted before any acknowledgments were required by the sender. Although the Burst Mode protocol performed some

four times better than TCP, its transmission rate was still only at 25% of the link capacity. This low efficiency is due mainly to the Burst Mode protocol high processing overheads, which counter some of the efficiency improvements gained with the use of multiple buffers. Furthermore, the Burst Mode protocol was far superior to the other protocols only when large transmission volumes were involved; there was not much difference between the five protocols when small data volumes were transmitted.

2.8.3 TCP in CODE Experiments

Fairhurst [25] analyses the TCP/IP (including UDP—User Datagram Protocol) protocol experiments in the CODE (Co-Operative Data Experiments) system. CODE is a VSAT system used for the interconnection of LANs, and is part of the European Space Agency's Olympus Utilisation Programme. Table 2.1 is an excerpt from one set of experimentation results presented in [25], which compare various data link protocols, error recovery strategies and fragmentation methods. The results show that Selective Repeat retransmission is far more suitable for satellite transmission than Go-back-N retransmission. On average, Selective Repeat retransmission produced throughput 37 % better than Go-back-N with a Bit Error Rate (BER) of 10^{-5}, and 67% better with a BER of 10^{-4}.

Fairhurst states that a typical buffer size of 4 Kbytes per TCP session limits the maximum throughput to approximately 50 Kbps for the lowest delay satellite link. Experiments with 8 Kbyte buffers, which facilitate larger window sizes, over 64 Kbps and 2 Mbps error free channels produced the results shown in table 2.2. Default socket settings limit throughput to 50 packets per second. Correcting these settings may improve the throughput as shown in brackets in table 2.2. Another set of experiments (not shown here) involving various error rates over a 64 Kbps channel produced results lower than those for an error free link (obviously). However, the interesting point is that the throughput rates for BERs of 10^{-5} and lower, peaked with the larger packet sizes used—between 512 bytes and 1024 bytes.

Table 2.1. *TCP throughput results for various system configurations and BERs (excerpt from table 2 in [25]).*

Error Recovery	Fragmentation Method	Throughput	
		BER=10^{-5}	BER=10^{-4}
GBN	None	47.8	5.7
GBN	IP	28.6	14.6
GBN	Transparent	36.9	21.2
SR	None	58.6	9.0
SR	IP	43.6	28.1
SR	Transparent	49.7	31.7

Table 2.2. *TCP throughput results over error free channels, results with optimal socket settings are in brackets (excerpt from table 3 in [25]).*

Channel Speed	Packet Size	Throughput—Kbps	
		TCP	UDP
64 Kbps	128	50.8	46.5
64 Kbps	512	58.4	58.5
2 Mbps	128	51 (87.2)	51.6 (> 1000)
2 Mbps	512	87.2	204.6 (> 1000)

According to the authors of this report, because of to the satellite propagation delay, the maximum achievable throughput for TCP over the 2 Mbps link is 320 Kbps. A Quick File Transfer Protocol (QFTP) written by the authors, achieved a throughput of 910 Kbps over a predominantly error free link. QFTP uses the User Datagram Protocol (UDP) with a large retransmission window, and a Go-back-N recovery protocol. One conclusion drawn by the authors, is that TCP/IP provides poor performance over long delay links, particularly if the BER is greater than 10^{-6}.

2.8.4 Strategies For Improving TCP over Satellite Links

As mentioned in section 2.6, a great deal of research has been carried out in search of strategies to improve TCP performance over a satellite link. An example of such research is presented in [28], where 11 schemes, designed to improve TCP performance over a satellite link, are compared. The schemes are categorised into three basic groups:

- end-to-end proposals
- link layer proposals
- split-connection proposals

End-to-end proposals are based on TCP-Reno [35] with enhancing features such as selective acknowledgment and Explicit Loss Notification, which enables the sender to distinguish between congestion losses and other losses. The highest throughput achieved with this method was 760 Kbps.

Link-layer proposals are based on the Snoop protocol [30] and use features such as selective acknowledgment and Duplicate Acknowledgment Suppression. The link-layer protocols attempt to hide link related losses from the TCP sender by retransmitting lost frames if they are cached. The best throughput achieved with this method was 1.22 Mbps.

Split-connection proposals, like I-TCP (Indirect-TCP) [29], use a separate TCP connection over the satellite link; the initial TCP connection is terminated at the base station. Therefore, the satellite link is completely hidden from the sender. The main enhancement feature used with this method was Selective Acknowledgment, which produced a throughput of 1.1 Mbps.

Although the best performance was produced by a Link-layer proposal, a Split-connection method may be a better option if a link layer protocol is used over the satellite link rather than TCP. A Split-connection setup is the envisaged environment for the RSBT protocol.

2.9 Conclusion

The research reported in the literature did not provide any proven protocol for reliable satellite broadcasting/multicasting. However, several broadcasting methods that are not completely reliable or suitable for satellite broadcasting have been presented. This chapter has reviewed these methods. The review has also highlighted the protocol procedures that have a detrimental effect on satellite transmission performance, as well as the procedures and countermeasures that enhance performance.

The ATM methods of broadcasting/multicasting—separate addressing and multicast tree—do provide reliability, but the multicast is in fact made up of many unicast transmissions [33]. The ATM hardware delivery approach [33] and the CMC METSIS system [10] are true broadcasting systems, in that they transmit packets to all receivers simultaneously. Furthermore, the significantly superior performance results of hardware delivery broadcasting over ATM networks, as compared to the separate addressing and multicast tree methods, are very encouraging. However, the reliability of both hardware delivery and the METSIS systems is limited because a truly reliable broadcasting protocol is not available.

The literature on reliable protocols used for point-to-point transmission across a satellite link revealed some of the obstacles to greater efficiency and some remedies. First, the sliding window flow control mechanism is a cause of major delays. The problem is not so much with the sliding window, but more so with the amount of buffer space available. The delays caused by the sliding window protocol are far more critical with faster links, resulting in very low link utilisation, which reflects as reduced efficiency. The delays are compounded for TCP because it uses the Go-back-N retransmission protocol, which was shown to be inefficient over a satellite link [6],[24],[25]. Due to the larger amount of data in a satellite channel (due to the propagation delay), performance is greatly enhanced by using Selective Repeat retransmission. Optimal packet sizes also improve efficiency. If the packets are too small, the header overheads can become significant. On the other hand, if the packets are too large, the probability of

packet loss increases and much larger volumes of data are retransmitted. Perhaps the best performance improvement for TCP based satellite transmission can be achieved by using the methods described in [28]. That is, either using an appropriate link layer protocol, or splitting the TCP connection and using a more suitable protocol over the satellite link.

NETBLT and the Burst Mode protocols have addressed the above mentioned problems with satellite communication. Multiple buffers are used by both to overcome the flow control delays, and both protocols employ Selective Repeat retransmission. Furthermore, NETBLT negotiates the packet size at connection setup, while Burst Mode uses a dynamic packet size, which is adjusted during transmission.

The design and development of the RSBT protocol follows on from the improvements made by the NETBLT protocol and the Burst Mode protocol. That is, improving on some aspects of satellite transmission and extending the functionality to serve as a truly reliable point-to-multipoint data communication protocol.

3. PROTOCOL DESCRIPTION

3.1 Introduction

In the second century, the Greek historian "Polybius" described how fire signals could be used to communicate; the method was apparently used by Athens in 458 B.C. to conquer Troy [36]. Thus, communication protocols have existed for a very long time, and currently there are many protocols that are considered as standards for providing specific solutions. The reliability of a protocol is an important issue, particularly with data communications. Hence, many Reverse Error Control techniques have been developed and incorporated in protocols for reliable point-to-point data communication. However, no genuine Reverse Error Control scheme has been developed for reliable broadcast/multicast transmission.

Three methods of broadcasting in an ATM environment were briefly described in section 2.7. Two of the methods: Separate Addressing and Multicast Tree, use the AAL5 protocol [34] and are thus reliable. However, Separate Addressing is nothing more than a series of unicast transmissions. The Multicast Tree is the same, except that the sender transmits data only to its children in the tree. Each child node then transmits the data to its children, and so on. The third method, Hardware Delivery, uses a switch to broadcast the data to all receivers simultaneously. However, the switch can not provide error control, and therefore, any error checking must be done at a higher level.

None of the three broadcasting methods mentioned above provide true reliable broadcasting suitable for satellite transmission. Hence, the need for developing the RSBT protocol. More than just providing reliability to a broadcast transmission, RSBT also provides better performance over a satellite link. All of the common reliable protocols, such as TCP and HDLC, were developed for use on LANs and terrestrial WANs. Thus, for various reasons, these protocols

perform poorly over satellite links [37]. RSBT on the other hand was developed specifically for satellite transmission, avoiding the shortcomings of generic protocols when used over a satellite link.

3.2 Purpose

The RSBT protocol was developed primarily to satisfy the Australian Bureau of Meteorology's requirements in reliably broadcasting data over a permanent satellite link. Data is to be transmitted at a speed of 2 Mbps in a continuous flow; all frames are to be transmitted contiguously to minimise delays. The protocol must broadcast all types of files, from plain text to compressed image data. Only one site will be used to broadcast data; receiving sites will only ever transmit acknowledgments (hereinafter meaning ACK/NAK) and possibly control messages (implementation dependent). Receivers will transmit acknowledgments and messages across separate satellite or terrestrial return links. Reliable return links would be preferred because possible delays in retransmission would be avoided, though RSBT should function with unreliable return links as well. RSBT minimises the volume of return traffic. Thus, the number of receivers on the network need not be limited. However, further analysis of the return traffic should be carried out if a very large network is involved.

In regards to the Bureau of Meteorology, RSBT will be used in an environment where files are automatically generated by the sender and disseminated to all receivers for further processing. The process begins with automatic FTP/TCP transmission of centrally processed data to the satellite interface (described in section 3.3.1). The RSBT protocol, which will reside in the satellite interface, will then broadcast the data across a permanent satellite link to each receiving satellite interface. The RSBT module on each of the receiving satellite interfaces will temporarily store the received data on disk, if it is not corrupted; all error control requirements are performed by the RSBT modules on the sending and receiving satellite interfaces. Finally, the data is transmitted to its final destination on the LAN using FTP/TCP. This three-step transmission was also proposed in [21] and [28] (see sections 2.5 and 2.8.4).

3.3 Assumptions About the Environment

The RSBT protocol was designed to function in WANs where all information is disseminated from one site, e.g. the head office. The end-to-end file transfer involves several media, and RSBT must interface with the application protocols used on the LANs, e.g. FTP. The assumptions made here are based on the Australian Bureau of Meteorology's requirements, and the use of the Omnicast Digital service provided by Optus Communications. Nonetheless, it would be possible to fine-tune the protocol for other applications and environments as well.

3.3.1 System Configuration

Figure 3.1 shows a generic system configuration, which includes a satellite interface at each site. The use of TCP on the LANs and HDLC on the WAN, reflect the current status of communications at the Bureau of Meteorology. However, RSBT can be interfaced to other protocols as well.

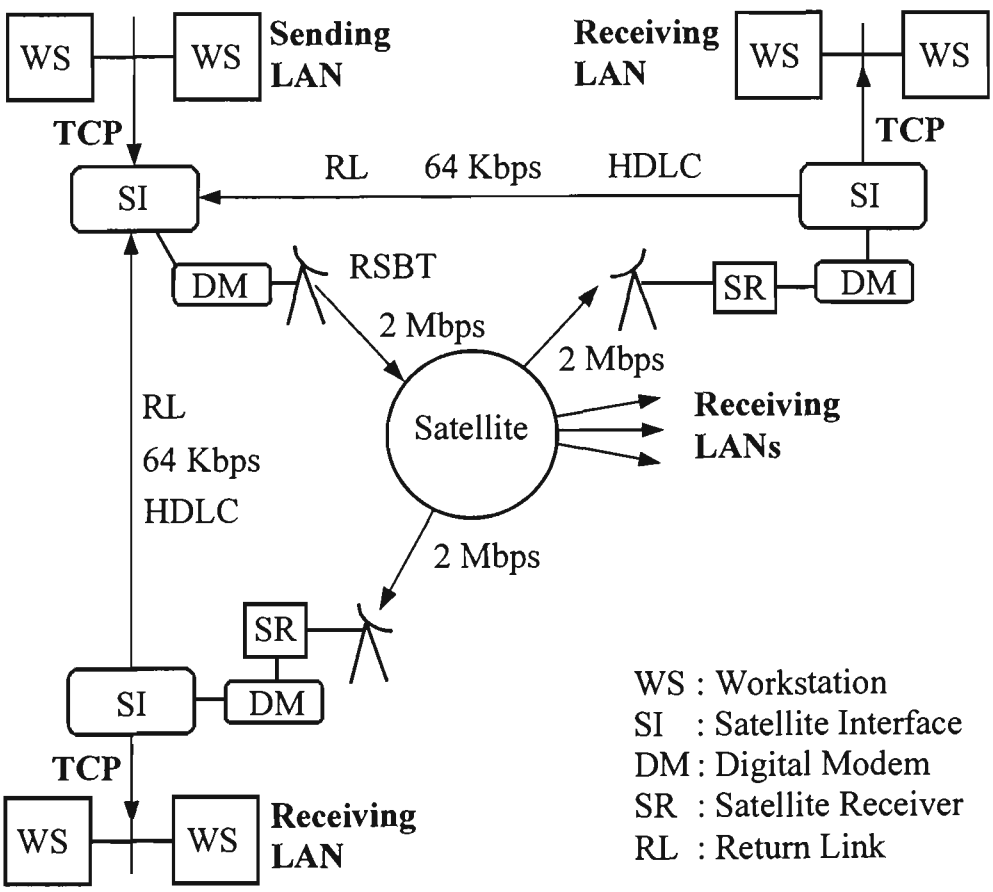


Figure 3.1. *System configuration, links, and protocols used.*

The RSBT protocol will reside in each satellite interface—a dedicated UNIX box with its own hard disk for temporary storage of data. As with the NETBLT protocol [7], each receiving satellite interface will need multiple buffers to cope with the continuous incoming data stream (see section 3.3.2). The buffers will be used in a cyclic manner, ie. when the first buffer becomes full, a second buffer is used to store the incoming data, and at the same time, the data in the first buffer is processed, and so on. Each satellite interface is to be connected to its respective LAN in the same way as any other workstation. Thus, all satellite interfaces will have their own IP address, allowing them the same network connections. These network connections will be used by the terrestrial return links from the receivers to provide reliability to the broadcast transmission.

3.3.2 Service Used

RSBT is being developed for the Omnicast Digital service, provided by Optus Communications. The major features of the service are [3]:

- One-way digital transmission: return links are established separately.
- Continuous transmission: frames may be transmitted contiguously.
- Transmission speeds between 64 Kbps and 2 Mbps in 64 Kbps steps.
- QPSK modulation (Quadrature Phase Shift Keying).
- FEC: Forward Error Correction with a code rate of ½ is used.
- Permanent connection: if the connection is lost, it may take up to one minute to regain it.

To establish a connection, each receiver progressively seeks a digital carrier, looking for a unique carrier indent pattern. Once locked-on it maintains a lock over ± 3 MHz, which provides a permanent connection. If the connection is lost, a reconnection process is restarted automatically—it takes up to one minute to re-establish a connection [3]. As the overall performance depends on the customer's receiving equipment, only the minimum signal level received from the satellite is specified, the performance parameters such as BER, are not [3].

3.3.3 Protocol Conversion

As depicted in figure 3.1, the sending satellite interface will use the RSBT protocol to communicate with each receiving satellite interface. While FTP/TCP will provide the communication between workstations and the satellite interface on each LAN. Hence, the satellite interfaces must perform the protocol conversion between FTP and RSBT.

The FTP file transfer from the sending LAN to its satellite interface will use normal FTP Data Transfer Commands. RSBT must pass these commands to the receiving satellite interfaces so that they can revert back to FTP and transmit the file to a workstation on their respective LANs more quickly. The FTP Data Transfer Commands available are [38]:

Mode	Stream, Block, and Compressed;
File Structure	Structured, Unstructured, and Random Access;
Data Type	Binary, Text (ASCII and EBCDIC, with vertical format control: Non-print, Telnet Format Effectors, Carriage Control (ANSI)) and Local Type.

The FTP Data Transfer Commands may be encapsulated into each RSBT frame header. Alternatively, they may be sent in a control frame preceding data transmission. However, the control frame, which may also be used to check how many receivers are on line, would require priority in the retransmission queue.

The RSBT frame length may be independent of the FTP packet length, or it may contain an exact number of FTP packets. If the RSBT frame length is independent of FTP packets, then any FTP data headers [38], which provide further information about the data, would have to remain as part of the data stream. The existence of these FTP data headers depends on the transmission type, e.g. *Block Mode* includes the *Block Length* and a I [38]. The *Block Length* and the *Descriptor* are required by receivers to segment the data into the original FTP packets for transmission on the LAN. Having an exact number of FTP

packets within a RSBT frame allows the sender to remove or curtail any FTP data headers in the RSBT transmission. It would also require less processing by receivers, and allow them to transmit some FTP packets to the LAN sooner.

3.4 Frame Format

The RSBT protocol uses only two frame formats: the format that is transmitted by the sender (data frames), and the format that is transmitted by receivers (acknowledgments). The details of these two frame formats are shown in figure 3.2 and figure 3.3 respectively. The frames transmitted by the sender use a *Flag* as a start-of-frame delimiter. To ensure that the bit pattern used for the flag does not appear in the data, the flag must be complemented by zero bit insertion, as with HDLC transmission [39]. The acknowledgment frames that are transmitted by receivers are enclosed within the frames of the protocol used over the return links.

Flag	File ID	Version	Sequence Number	Length	More Data	FCS Header	Data	FCS DATA
8	16	4	8 to 32	16	1	32		32

Figure 3.2. *Format of frames sent by the sender; field lengths are in bits.*

File ID	Sequence Number	ACK/ NAK
16	8 to 32	1

Figure 3.3. *Format of frames sent by receivers; field lengths are in bits.*

As the Omnicast Digital service provides a permanent satellite connection, frames that are transmitted by the sender do not include any addressing. Instead of carrying an address, the RSBT frames carry a *File ID*; a similar ID is used by the NETBLT protocol during the connection process [7]. One half of the *File ID* identifies the file type and release (concerning data disseminated at regular time intervals), while the other half carries the encapsulated FTP Data Transfer

Commands [38]. In fact, only seven bits are required for encapsulating the FTP commands—there are 72 FTP command combinations. The *File ID* is also used by receivers to recognise the transmission of a new file. This would occur in the event of major transmission errors, where the sender has given up retransmitting corrupted frames, and commenced transmitting a new file.

The RSBT protocol does not recycle frame sequence numbers; each sequence number identifies only one frame in the file. Therefore, a large set of sequence numbers is required. To accommodate large files, and at the same time minimise overheads, the version field sets the sequence number length between 8 and 32 bits, in 4-bit steps. The additional overhead, created by not reusing sequence numbers, can not be avoided because RSBT transmits the complete file before it begins retransmission of lost frames. The alternative would be to retransmit frames as they are requested, and to use smaller sequence numbers that are recycled. However, because recycling of a particular sequence number can not be done until all receivers have received that frame, a receiver with bad reception would delay all other receivers.

The RSBT error control scheme is enhanced by the use of two Frame Check Sequence (FCS) fields: *FCS Header*, and *FCS Data*. NETBLT [7] and ATM [33] also use two FCS fields. The additional FCS allows negative acknowledgment of corrupted frames, whose header integrity was maintained. The FCS is derived by a CRC-32 (Cyclic Redundancy Check) generator polynomial, which is capable of capturing all single bit errors, all burst errors of up to 31 bits, and most burst errors of 32 bits or more [39].

The main purpose of the frames transmitted by receivers, is to deliver acknowledgments to the sender. However, the same frame format can also be used to inform the sender that a particular receiver is either coming on-line or going off-line (see section 3.5.1). To identify that a frame is being used for this purpose, the File ID would carry a unique bit pattern rather than the actual file identification, and the ACK/NAK field would be used to indicate on-line/off-line.

3.5 Transmission Procedures

RSBT is to provide reliable transmission to multiple receivers over a permanent satellite link. RSBT is responsible only for delivering the data to each receiver; the receivers send the data to the appropriate workstation on the LAN. Therefore, RSBT frames do not need to include any addressing, as they do not need to be routed. Furthermore, because the communication channels are permanent, control frames for opening and closing session are not required. In addition, the use of a satellite interface on each LAN allows RSBT to dispense with flow control, as is done with the Remote Procedure Call protocol [40]. Thus, all frames are transmitted contiguously, and retransmissions are done after the whole file has been transmitted once. The main functions of the RSBT protocol—session indication, error control and retransmission—are described in the following subsections.

3.5.1 Session Indication

The Omnicast Digital service provides a permanent link between the sender and the receivers. Thus, the communication channels are assumed to be established before the RSBT protocol comes into play. Therefore, RSBT does not need to use addition control frames to open and close communication sessions. However, receivers that are going off-line for long periods, transmit a message to inform the sender of their status.

A receiver may temporarily lose the connection, but when this happens, the Omnicast Digital service automatically begins a reconnection process as described in section 3.3.2. If a receiver is off-line when the transmission is about to begin, it would be pointless delaying the other receivers until that receiver regains the connection. The sender should know how many receivers there are, and each of the error control methods described in section 3.5.2, can handle the temporary loss of reception by one or more receivers. When a receiver's connection is re-established during transmission, the receiver simply transmits the appropriate list of NAKs and continues as per the normal procedure.

There is one scenario that could cause unnecessary delays. That is, if a receiver goes off line for reasons other than loss of reception: LAN maintenance, satellite receiving equipment failure, etc. In this type of situation, the receiver could be off-line for much longer periods than if only the carrier is temporarily lost. Therefore, to minimise delays for the sender, the affected receiving satellite interface, or any workstation on the LAN, transmits a control message to the sender, indicating that it is off-line. When the problem is corrected, another message is sent to the sender, indicating that the receiver is back on-line. Excessive delays are avoided because under normal circumstances, the sender would continue retransmission (within limit) until each receiver has acknowledged the successful reception of the whole file (see section 3.5.2). The on-line/off-line messages allow the sender to halt futile retransmission, and perhaps start another broadcast. Note that the aim of these control messages is to avoid unnecessary delays for the sender. If these messages are not used, RSBT will not crash or become deadlocked because retransmission would stop if an acknowledgement is not received within a set time period (see section 3.5.2).

3.5.2 Error Control

The Omnicast Digital service uses FEC to minimise errors [3]. This is sufficient for most random errors, but not good enough to trap and correct burst errors; most errors occur in bursts (see section 6.2). Therefore, the RSBT transmission was enhanced by reverse error control procedures. Unlike most protocols though, RSBT does the required retransmission once it has transmitted all the frames once—retransmissions do not have priority.

To facilitate reverse error control, RSBT frames include two FCS fields. If only one FCS is used to cover the complete frame, corrupted frames could not be negatively acknowledged because the integrity of the sequence number could not be guaranteed. With two FCS fields, one FCS covers the header, and the second FCS covers the data. Therefore, RSBT can negatively acknowledge frames where the data is corrupted, but the header is received intact. It should be noted that the frame header is much smaller than the data. Thus, the frame header has a

correspondingly lower error rate (or loss rate). The additional overheads incurred by the use of a second FCS field will not have a significant effect on transmission efficiency. This is because RSBT frames are expected to be quite large compared to frames of other protocols; RSBT frame sizes are not restricted by buffer sizes of intermediate nodes and traffic, which compromise protocols used over terrestrial links. Furthermore, the Omnicast Digital service uses forward error correction, which reduces errors, and thus, promotes larger frame sizes. Another advantage of having two FCS fields, is the early transmission of NAKs, which avoids delays when several frames in the tail end of the transmission are corrupted. However, there is no efficiency improvement if all frames are acknowledged (section 3.5.2.1), because the sender will retransmit frames anyway if an ACK has not been received from each receiver. In fact, the additional overhead caused by the second FCS field will reduce the transmission performance if all frames are acknowledged.

In providing reliability to a broadcast transmission, the RSBT acknowledgment process must be streamlined. Otherwise, the sender may be overwhelmed by return traffic if the RSBT protocol is implemented on a large network. Hence, four acknowledgment procedures have been devised and investigated. They are:

1. *Acknowledge All Frames*
2. *Acknowledge Last Frame Only*
3. *Acknowledge First and Last Frames*
4. *Acknowledge Lapse in Transmission*

3.5.2.1 Acknowledge All Frames

This is the most common approach, and is used by protocols such as TCP [22] and HDLC [39],[41] in point-to-point data transfer. This method of acknowledgment requires receivers to send an ACK for each frame received successfully. RSBT differs from TCP and HDLC, in that it is a point-to-multipoint protocol. Therefore, RSBT will receive acknowledgments from

multiple receivers, and thus needs to maintain a more complex array of variables during transmission, which include:

- **Retransmission List.** The *Retransmission List*, as shown in figure 3.4, is a list that contains both the sequence numbers of frames that were transmitted and the number of ACKs received for each frame. These frames are retransmitted in a round-robin manner, and are removed from the list when an ACK has been received from each receiver. Figure 3.5 shows a more detailed *Retransmission List*, which is used if the return links are not reliable. This *Retransmission List* uses flags to keep track of which receivers have acknowledged which frames; the flag is set to **1** if an ACK is received and it is left as **0** if a NAK is received. Although a flag set to **0** could also mean that an acknowledgment was not received, the *Receiver List*, described below, indicates whether a NAK was received or in fact, an acknowledgment was not received.
- **Receiver List** (used if return links are not reliable). The *Receiver List*, as shown in figure 3.6, is a matrix that keeps track of which frames have and have not been acknowledged (positively or negatively) by which receivers; a flag (**1** or **0**) indicates whether or not an acknowledgment has been received. The sender uses the return links to request the missing acknowledgments after a time-out period. Recovery of acknowledgments can be delayed if a frame has been negatively acknowledged by another receiver, because the missing acknowledgment may well be a NAK. Thus, the receiver will send another acknowledgment when it receives the retransmission. A frame sequence number is removed from the *Receiver List* when it is removed from the *Retransmission List*.
- **Not Received List.** Receivers use this list to store the sequence numbers of frames not received successfully. During retransmission, the list is used to select the frames that are required by the receiver, and to disregard those that are not required.

- **Greatest.** Receivers use this variable to store the largest frame sequence number received. *Greatest* allows receivers to negatively acknowledge frames that are missing in the sequence. However, it becomes redundant once the last frame in the file has been received.
- **Last.** This variable is used by receivers to store the sequence number of the last frame in the file. If the last frame has been received and the *Not Received List* is empty, a receiver knows that it has received the complete file.

Frames awaiting ACKs	2	3	4	8	12	13	20	21	30
No. of ACKs Received	2	2	0	1	2	2	1	2	2

Figure 3.4. *Retransmission List used with acknowledgment of all frames when the return links are reliable.*

Frames Awaiting ACKs	2	3	4	8	12	13	20	21	30
Receiver 1	0	1	0	1	1	0	0	1	1
Receiver 2	1	1	0	0	0	1	0	1	0
Receiver 3	1	0	0	0	1	1	1	0	1

Figure 3.5. *Retransmission List used with acknowledgment of all frames when the return links are not reliable.*

Frames Awaiting ACKs	2	3	4	8	12	13	20	21	30
Receiver 1	1	1	1	1	1	1	1	1	1
Receiver 2	1	1	0	1	1	1	1	1	1
Receiver 3	1	0	1	1	1	1	1	1	1

Figure 3.6. *Receiver List used with acknowledgment of all frames when return links are not reliable.*

The high-level pseudocode shown in figure 3.7, provides an overview of the *Acknowledge All Frames* protocol (the pseudocode is based on C++ notation).

Note that the protocol description given in figure 3.7 is not intended to be absolutely correct or complete, it is there to assist the description that follows.

```
SENDER
Transmit
        While (more frames)
                Transmit next frame
                Put sequence number in Retransmission List
                If ACK received
                        Call Process ACK
        While (Retransmission List not empty)
                Retransmit next frame
                If ACK received
                        Call Process ACK
Process ACK
        Add 1 to Retransmission List frame counter
        If (frame ACKs == number of receivers)
                Update Retransmission List
                If (Retransmission List empty)
                        Stop transmission
RECEIVER
Process Frame
        Last = -1, Greatest = -1
        While (Last == -1 || Not Received List not empty)
                Wait for frame
                If (sequence number > Greatest)    /*New Frame*/
                        If (data ok)
                                Store frame
                                Send ACK
                                If (sequence number > Greatest + 1)
                                        Update Not Received List
                        Else
                                Update Not Received List    /*For 1 frame*/
                                If (sequence number > Greatest + 1)
                                        Update Not Received List
                        Greatest = sequence number
                        If (last frame)
                                Last = sequence number
                Else if (sequence number in Retransmission List && data ok)
                        Store frame
                        Send ACK
                        Update Not Received List
```

Figure 3.7. *Overview of the Acknowledge All Frames protocol.*

When the sender begins transmission, it places the sequence number of each transmitted frame in the *Retransmission List*. The *Retransmission List* is used not only to store the sequence of frames that may require retransmission, but also to count of the number of ACKs received for each frame (see figure 3.4). A frame sequence number is removed from the *Retransmission List* when the number of ACKs for the frame tallies with the number of receivers. The frames listed in the *Retransmission List* are retransmitted in a round robin manner. Even when the *Retransmission List* becomes small and all frames are transmitted in a time shorter than the round trip time, the sender keeps retransmitting the frames in turn, instead of sitting idle waiting for acknowledgments. This pre-emptive retransmission is similar to the retransmission procedure used by the Remote Procedure Call protocol [40], and is quite appropriate for RSBT because the Omnicast Digital provides a dedicated channel which would otherwise sit idle during lapses in transmission. If no acknowledgments are received within a set time period, the pre-emptive retransmission would be stopped. This would avoid deadlocks in a situation where a receiver loses reception and does not recover within an acceptable time period, or if a return link is severed.

Each receiver sends an ACK for each frame that is received correctly. The sequence number of each corrupted or lost frame is stored in the *Not Received List*. As these frames are received in the retransmission phase, appropriate ACKs are transmitted to the sender and the sequence numbers are removed from the *Not Received List*. Receivers also maintain two other essential variables: *Greatest* and *Last*. *Greatest* keeps the largest sequence number received. Its main purpose is to identify lost frames. For example, if frame 12 is received, and *Greatest* has the value 10, then obviously frame 11 was lost and so the number 11 is added to the *Not Received List*. If a frame sequence number is smaller than *Greatest*, then a simple check of the *Not Received List* is used to determine whether the frame is required. *Last* is initialised with a negative value, and set to the sequence number of the last frame in the file when it is received (the *More Data* field in the frame header identifies the last frame). Thus, if the *Not Received List* is empty and *Last* is not negative, the receiver knows that it has received the whole file successfully.

The transmission process may get into an infinite loop if the return links are not reliable. This is because frames that have been received correctly, and acknowledged once, are not acknowledged in later retransmissions. Thus, if an ACK is lost, the sender will keep retransmitting the frame because it has not received an ACK, and the receiver will keep disregarding the retransmissions because it already has the frame. Therefore, if the return links are unreliable, receivers will also transmit NAKs for frames not received correctly. The sender will use the *Receiver List* shown in figure 3.6, to record the receipt of each acknowledgment (ACK or NAK) from each receiver. In addition, the sender will use the more detailed *Retransmission List* shown in figure 3.5. With the altered procedure, the sender is aware of any lost acknowledgment, and although it does not know whether a lost acknowledgment is an ACK or a NAK, it does know which receiver is involved. Thus, the sender can use the return links to request another acknowledgment for the particular frame. However, the frame will remain in the *Retransmission List* as normal until the acknowledgment is received.

This protocol would not be suitable for large networks, because the return traffic volume could cause congestion and overflow losses. However, it would be suitable if the return links are unreliable and the network is small.

3.5.2.2 Acknowledge Last Frame Only

Acknowledging each frame in a point-to-multipoint transmission could overwhelm the sender with acknowledgments if there are many receivers. Thus, the streamlined *Acknowledge Last Frame Only* protocol works on the basic philosophy that no news is good news. Receivers use selective reject acknowledgment, and retransmit NAKs if a frame retransmission is not received within a pre-defined time-out period. Each receiver sends only one ACK—when the complete file is received. The variables maintained with this method include:

- **Retransmission List.** This list contains the frame sequence numbers of all negatively acknowledged frames.
- **ACK Counter.** A counter for the ACKs sent by receivers.

- **Receiver List** (optional—used instead of the ACK Counter). This is a list of receiver flags indicating from which receivers an ACK has been received and from which an ACK has not been received. The use of an ACK Counter would inform the sender as to which receiver(s) is having problems receiving, and thus, appropriate action could be taken. This enables the sender to transmit the file to a receiver with reception difficulties across the return links or to simply stop retransmission when an ACK has been received from all other receivers.
- **Not Received List.** Receivers use this list to store the sequence numbers of frames not received successfully. During retransmission, this list is used to select the frames that are required, and to disregard those that are not required (same as for *Acknowledge All Frames*).
- **Greatest.** This is used by receivers to store the largest frame sequence number received (same as with *Acknowledge All Frames*).
- **Last.** This is used by receivers to store the sequence number of the last frame in the file (same as with *Acknowledge All Frames*).
- **Timer.** Receivers use the *Timer* to retransmit NAKs if retransmissions of frames are not received in a specified time-out period. The time-out period is dynamic and takes into account retransmission requirements of other receivers. First, the time-out period is set equal to the round trip delay. Then, if the sequence numbers of the incoming frames skip the desired frame, the NAK is resent. NAKs may be retransmitted earlier if after the initial round trip time multiple copies of the last frame are being received, because this is an indication that the senders retransmission queue is empty.

An overview of the *Acknowledge Last Frame Only* method is given in figure 3.8 in the form of a high level pseudocode; the pseudocode is based on C++ notation. Note that, as with figure 3.7, the operation description in figure 3.8 is not intended to be absolutely correct or complete.

```
SENDER
Transmit
        While (more frames)
                Transmit next frame
                If (NAK received)
                        Call Process Acknowledgment
        While (more ACKs to arrive)
                If (Retransmission List not empty)
                        Retransmit next frame
                Else    Retransmit last frame
                If acknowledgment received
                        Call Process Acknowledgment
Process Acknowledgment
        If (NAK received)
                If (sequence number not in Retransmission List)
                        Put sequence number in Retransmission List
                Else    Discard NAK
        If (ACK received)
                Add 1 to ACK Counter
                If (ACK Counter == number of receivers)
                        Stop transmission
RECEIVER
Process Frame
        Last = -1, Greatest = -1
        While (Last == -1 || Not Received List not empty)
                Wait for frame
                If (sequence number > Greatest)
                        If (data ok)
                                Store frame
                                If (sequence number > Greatest + 1)
                                        Update Not Received List
                                        Send NAK(s)
                        Else if (sequence number > Greatest + 1)
                                Update Not Received List
                                Send NAK(s)
                        Else    Send NAK
                                Update Not Received List
                        Greatest = sequence number
                        If (last frame)
                                Last = sequence number
                Else if (data ok)
                        Store frame
                        Update Not Received List
        Send ACK
```

Figure 3.8. *Overview of the Acknowledge Last Frame Only protocol.*

The transmission process begins with the sender transmitting all of the frames in the file contiguously. As NAKs are received, their sequence numbers are placed in the *Retransmission List*; duplicate NAKs are disregarded. When all of the frames have been transmitted once, the sender begins retransmission of frames that were negatively acknowledged; the sequence numbers of the retransmitted frames are removed from the *Retransmission List*. The sender re-orders the *Retransmission List* if NAKs are received out of sequence, which can occur for the following reasons:

- The transmission process works in a Round-Robin manner. First, the sender transmits all of the frames once. Then it retransmits all of the frames that were negatively acknowledged. Then it retransmits the frames that were negatively acknowledged for the second time, and so on. Thus, when retransmitted frames are negatively acknowledged, the NAKs may have a sequence number lower than some or all of the sequence numbers in the *Retransmission List*. Therefore, the sender must decide if a sequence number that appears out of sequence is out of sequence or if in fact, it is the start of the next wave of NAKs. If the NAK is deemed to be out of sequence, it is pushed towards the front of the *Retransmission List*. Otherwise, the sequence number is placed at the back of the *Retransmission List* and becomes the head of the next wave of retransmissions.
- If a receiver loses a string of frames, it sends NAKs when it receives a frame header intact. Hence, all but the last NAK are delayed. Therefore, if other receivers have sent NAKs within that delay time, the *Retransmission List* will be out of order and must be reordered,
- If a receiver experiences communication difficulty on the return link, the delays may result in a NAK arriving out of order.
- If one receiver completely loses a frame, while a second receiver loses only the data of the following frame, both receivers will actually send a NAK at the same time. Hence, the NAK from the second receiver could arrive first, putting the *Retransmission List* out of order.

When the sender has transmitted all of the frames in the *Retransmission List*, it must wait the round trip time to see if all remaining receivers send ACKs. If receivers respond with ACKs, and those ACKs make up the full complement, then the transmission is over. However, If a receiver did not receive any frames at all, or if it did not receive the last one or more frames in the sequence, then obviously the number of ACKs received will not tally with the number of receivers. To overcome the problem of missing ACKs, considering that the sender does not know the nature of the transmission error, the sender could retransmit the file from the beginning. However, this would cause unnecessary delay, if in fact the receiver were only missing the last frame. Furthermore, the receiver could fail to receive the last frame again, leaving the sender no wiser. Therefore, the sender repeatedly retransmits (within limit) the last frame until the receiver eventually gets it. Then, if it was only the last frame that was missing, the receiver sends an ACK, otherwise it sends NAKs for all other frames missing in the sequence.

Receivers transmit a NAK for each frame corrupted or missing in the sequence. In addition, as with NETBLT [7], receivers will retransmit a NAK if the requested frame is not received in a time-out period. The sequence number of each frame that is negatively acknowledged is placed in the *Not Received List*, and is removed from the list only after a retransmission of the frame has been received. Therefore, during retransmission, each receiver will require only a fraction of the frames being received—a frame is ignored if its sequence number is not in the *Not Received List*. Frames received error-free are not acknowledged. However, when a receiver has received all frames, it sends an ACK for the last frame, which actually acknowledges the complete file. Receivers know that they have all of the frames if they have received the last frame in the sequence, and if the *Not Received List* is empty.

Acknowledging the last frame only minimises the processing required by the sender. More importantly though, the return traffic is greatly reduced.

3.5.2.3 *Acknowledge First and Last Frames*

This method of acknowledgment is an enhancement to the *Acknowledge Last Frame Only* protocol (section 3.5.2.2). With this method, receivers also transmit an ACK for the first frame that they receive. The additional ACK informs the sender as to how many receivers are on-line and how many are off-line. This information allows the sender to carry out the retransmissions more efficiently in cases where some receivers are off-line. The variables that are needed with this approach include:

- **Session Counter.** The sender uses this counter to count the initial ACKs; the count informs the sender as to how many receivers have open sessions.
- **ACK Counter.** The sender uses this counter to count the ACKs sent by receivers when they have received all of the frames (same as for *Acknowledge Last Frame Only*).
- **Receiver List** (optional—replaces *Session Counter* and *ACK Counter*). The *Receiver List*, as shown in figure 3.9, uses two flags to indicate which ACKs have been received from which receivers. The *Initial ACK* is the session check and the *Final ACK* is the acknowledgment for the whole file.
- **Retransmission List.** The sender uses this list to stores the sequence numbers of frames that require retransmission (same as for *Acknowledge Last Frame Only*).
- **Not Received List.** Receivers use this list to store the sequence numbers of lost frames (same as for *Acknowledge All Frames*).
- **Greatest.** This is used by receivers to store the largest frame sequence number received (same as for *Acknowledge All Frames*).
- **Last.** This is used by receivers to store the sequence number of the last frame in the file (same as for *Acknowledge All Frames*).
- **Timer.** The *Timer* is used by receivers to retransmit NAKs if retransmissions of frames are not received in a specified time-out period (same as for *Acknowledge Last Frame Only*).

Each receiver sends an ACK as soon as it receives a frame header intact. This ACK always carries the frame sequence number zero, regardless of whether the frame was actually frame zero, and whether the data was intact. If the frame was not frame zero, or if the data was corrupted, the receivers also transmit the appropriate NAK(s). If the sender uses a *Receiver List*, a NAK can also serve as the initial acknowledgment. That is, if the first frame that is received by a receiver is corrupted, the NAK serves as a request for a retransmission and as the initial ACK, which eliminates duplication of return traffic. Moreover, the initial acknowledgment (ACK or NAK) also serves as NAK for all of the preceding frames. The initial ACK informs the sender as to how many receivers are on line. If some receivers are still off line at the end of the transmission, the sender knows that the complete file must be retransmitted. Obviously, if the transmission consists of only one frame, the initial ACK is not used.

Receivers	**R1**	**R2**	**R3**
Initial ACK	1	1	0
Final ACK	1	0	0

Figure 3.9. *Receiver List used with the Acknowledge First and Last Frames protocol; R1 has received all of the frames, R2 has received some frames, and R3 has not received any frames.*

Transmission of frames with the *Acknowledge First and Last Frames* protocol and the *Acknowledge Last Frame Only* protocol, does not differ prior to all receivers with open sessions successfully receiving the complete file, and confirming this with the final ACK. At this point, with the *Acknowledge Last Frame Only* protocol, the sender retransmits the last frame repeatedly, until a response is obtained from a receiver that has regained its session. While with the *Acknowledge First and Last Frames* protocol, the sender retransmits the whole file repeatedly. The advantage of this method is that a receiver that has just regained its session will commence normal reception immediately. Instead of waiting for its NAKs to trigger the retransmission of all other frames by the sender. This difference is illustrated in figure 3.10.

Figure 3.10 depicts a scenario where all receivers with open sessions have received the whole file error free. While the receiver with the lost session was reconnected and began reception just before the retransmission of frame 1 was about to arrive. With acknowledgment of the first and last frames, the receiver receives the remaining frames without requesting them. However, with *Acknowledge Last Frame Only*, the receiver has to first send NAKs for the missing frames, which is less efficient. The actual transmission time difference between the two protocols is equivalent to the round trip time. This transmission time difference will always exist in situations where one or more receivers regain reception after all other receivers have transmitted ACKs.

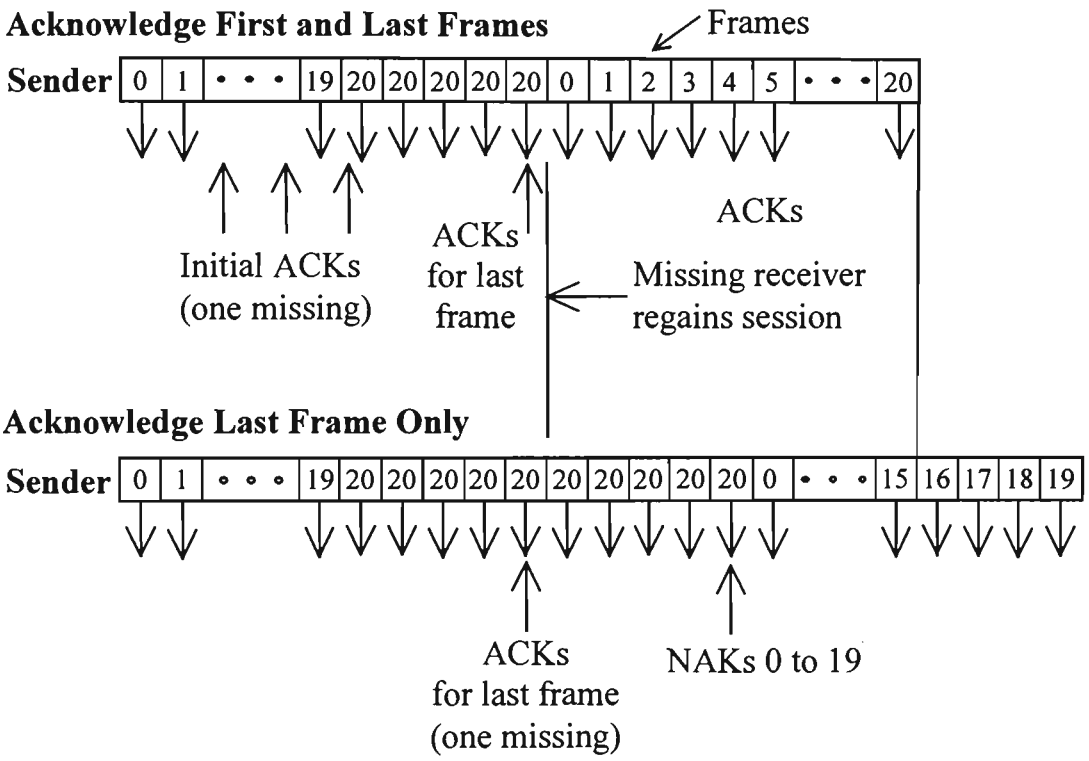


Figure 3.10. *The transmission efficiency of Acknowledge First and Last Frames compared to Acknowledge Last Frame Only. The assumptions are that the round trip time is equivalent to the transmission of four frames, and that there are 21 frames in the file. The repeated retransmission of frame 20 at the end of transmission and the final ACK are not shown for brevity.*

The *Acknowledge First and Last Frames* and the *Acknowledge Last Frame Only* protocols are very similar in both, the performance and the processing

requirements. The additional processing done by *Acknowledge First and Last Frames* protocol is minor. However, the performance improvement is only marginal, and only in certain transmission scenarios.

3.5.2.4 Acknowledge Lapse in Transmission

The *Acknowledge Lapse in Transmission* protocol is also an enhancement to the *Acknowledge Last Frame Only* protocol (section 3.5.2.2). This protocol provides three features that improve the protocol's acknowledgment process, and thus the efficiency. These features are as follows:

1. Receivers send an additional NAK for the frame expected in a transmission lapse (reception lapse). This additional NAK is also used in the NETBLT protocol [7] for expected buffers (groups of packets).
2. Once retransmission begins, receivers that have lost the tail of the transmission send an ACK for the last frame that they received; this ACK serves as a NAK for the lost frames. For example, if a file has 100 frames, and the sender receives an ACK for frame 90, then sequence numbers 91 to 100 are added to the *Retransmission List*. Without this additional ACK, receivers could not send NAKs for frames lost in the tail of the transmission until the *Retransmission List* becomes empty and the sender starts sending the last frame repeatedly.
3. After the last frame in the *Retransmission List* has been transmitted (say 16), the remaining frames in the sequence (17, 18, 19,....), which were not requested, are also retransmitted. Only then is the last frame retransmitted repeatedly. This pre-emptive retransmission improves efficiency because receivers that have lost the tail of the transmission receive the retransmissions sooner. However, this procedure is only useful if the collective size of the retransmissions is smaller than the amount of bits that can be in the stream—0.5 Mbits for a 2 Mbps channel, plus a bit more for the return channels. With large files, the additional ACK described in point 2 would place the lost frames into the *Retransmission List* and no delay would be incurred.

During the first transmission, if a receiver experiences a lapse in reception it sends a NAK for the frame that was expected, unless the previous frame was the last in frame in the file. Moreover, a NAK for the expected frame may be sent even if the previous frame was negatively acknowledged, as long as the previous frame header was intact. Transmitting NAKs early reduces the number of NAKs the sender receives out of order, and it may result in earlier frame retransmission. However, the additional NAK is most useful when a receiver loses all of the frames in the tail end of the transmission. Figure 3.11 depicts this scenario, where a receiver loses the last five frames of the transmission (16, 17, 18, 19 and 20). The sender receives the NAK for frame 16 while it is transmitting the last frame (20). Thus, frames 16 to 20 are retransmitted without any delay. The time saved in this situation is equivalent to the round trip time.

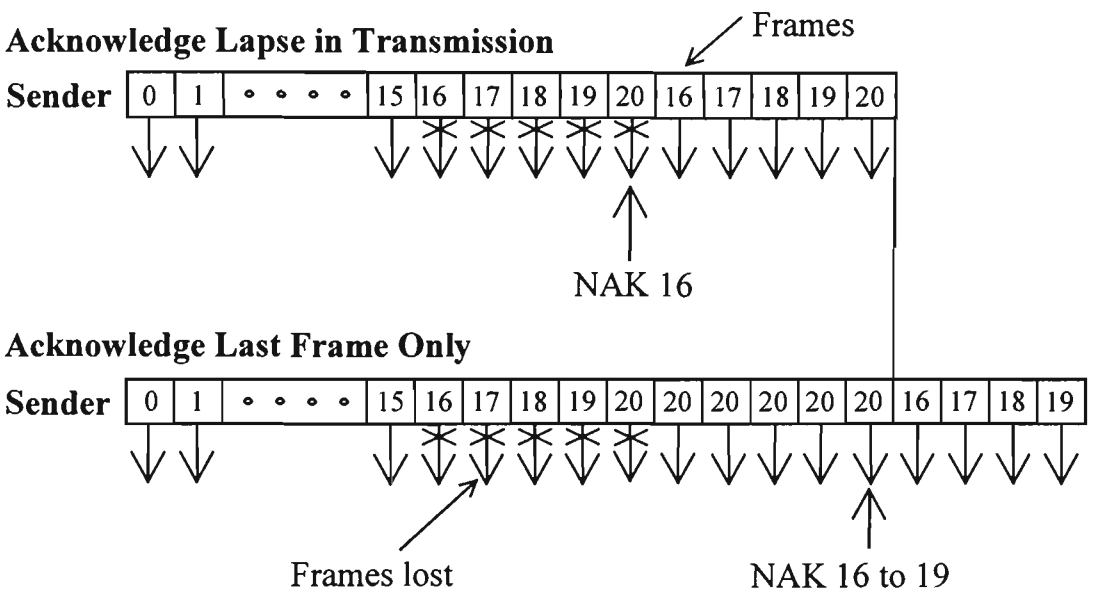


Figure 3.11. *The transmission efficiency of Acknowledge Lapse in Transmission compared to Acknowledge Last Frame Only, when a receiver loses five frames in the tail of the transmission. The assumptions are that the round trip time is equivalent to the transmission of four frames, and that there are 21 frames in the file. The repeated retransmission of frame 20 at the end of transmission and the final ACK are not shown for brevity.*

The maximum amount of time that can be saved using this protocol rather than *Acknowledge Last Frame Only*, is the equivalent to the round trip time. This is the case if the serialisation delay in retransmitting the lost frames is equal to or

greater than the round trip time (as in figure 3.11). However, if the serialisation delay of the retransmissions is less than the round trip time, then the time saved is proportionally less. Figure 3.12 depicts a scenario where a receiver has lost frames 19 and 20. With both methods, frame 20 is received immediately because the sender has retransmitted it several times (last frame). While frame 19 is received two time slots earlier with the *Acknowledge Lapse in Transmission* protocol because the NAK was sent earlier.

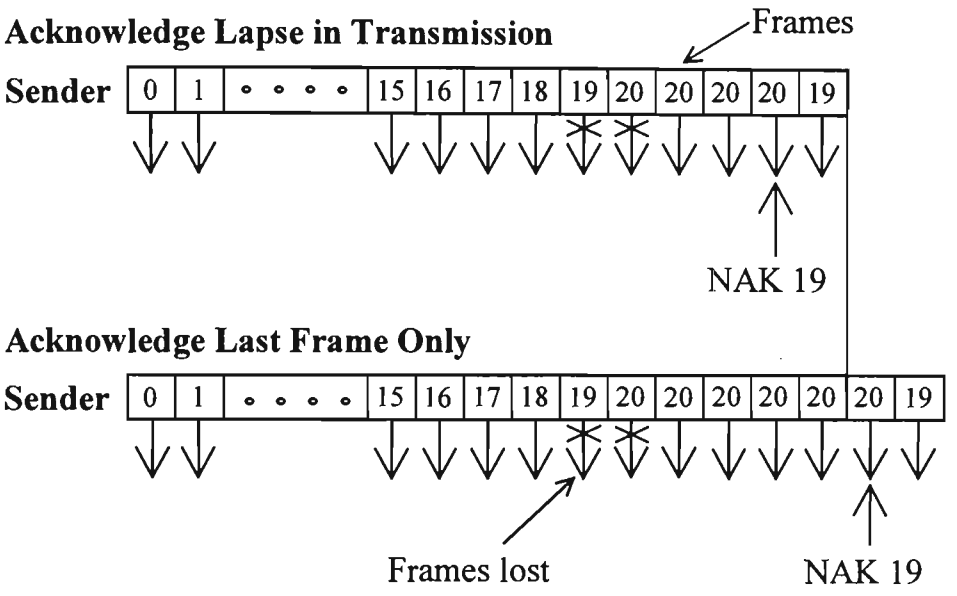


Figure 3.12. *The transmission efficiency of Acknowledge Lapse in Transmission compared to Acknowledge Last Frame Only, when a receiver loses only two frames in the tail of the transmission. The assumptions are that the round trip time is equivalent to the transmission of four frames, and that there are 21 frames in the file. The repeated retransmission of frame 20 at the end of transmission and the final ACK are not shown for brevity.*

In scenarios where two receivers lose the tail of the transmission, the time saved with this method of acknowledgment may be minimal. This can be seen in figure 3.13, where **receiver 1** loses frames 14 to 20 and **receiver 2** loses the last two frames (19 and 20). Before the **sender** has sent all 21 frames, it has received a NAK for frame 14 from **receiver 1**. So with the intention of retransmitting frames 14 through 20, it **sender** retransmits frame 14, then frame 15 and then 16. While it is retransmitting frame 16 though, it receives a NAK for frame 19 from

receiver 2. Therefore, frame 19 is retransmitted next, followed by frame 20. At this point, the *Retransmission List* is empty, so the **sender** begins repeated retransmission of the last frame (20). During the fourth retransmission of frame 20, the **sender** receives NAKs for frames 17 and 18 from **receiver 1**, which would have sent the NAKs on receipt of frame 19. The **sender** promptly retransmits frames 17 and 18, and then recommences repeated retransmission of the last frame until the last ACK arrives, which is not shown in figure 3.13. The *Acknowledge Lapse in Transmission* protocol, in this scenario, completes the transmission only one time slot earlier than with *Acknowledge Last Frame Only*. When a receiver loses the tail of the transmission, the minimum performance improvement with this method is equal to the serialisation delay of transmitting one frame. This is because a NAK is transmitted at least one time slot earlier.

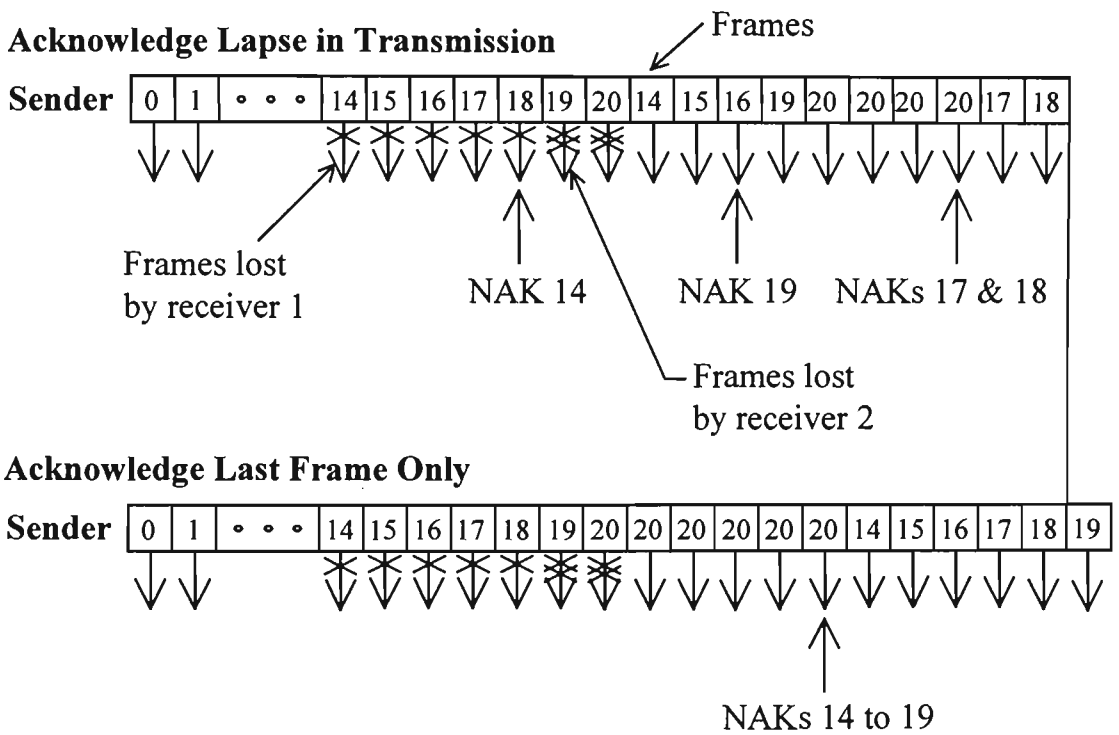


Figure 3.13. *The transmission efficiency of Acknowledge Lapse in Transmission compared to Acknowledge Last Frame Only, when two receivers lose the tail of the transmission. The assumptions are that the round trip time is equivalent to the transmission of four frames, and that there are 21 frames in the file. The repeated retransmission of frame 20 at the end of transmission and the final ACK are not shown for brevity.*

Recall from point 2 in the list of *Acknowledge Lapse in Transmission* protocol features, that receivers also use an additional ACK for the last frame that they have received in the first transmission. A receiver sends this ACK only after the sender has retransmitted a frame whose sequence number is smaller than the greatest sequence number that a receiver has already received. The arrival of a frame whose sequence number is not the greatest received, indicates to a receiver that retransmission has commenced. If retransmission starts before the last frame in the file is received, then a receiver knows that one or more frames in the tail of the transmission have been lost. Thus, an ACK is sent for the last frame received in the first transmission. Although this ACK may actually represent a frame that was originally negatively acknowledged, it can only be sent for a frame that was received with the header intact. An ACK that serves as a NAK for frames lost in the tail of the transmission can not be sent for a frame that was expected in a transmission lapse. This is because the expected frame may have been the last frame in the file, in which case the ACK would be mistaken for the final ACK. When the sender receives this ACK, it places all of the remaining sequence numbers in to the *Retransmission List*.

The addition ACK used by the *Acknowledge Lapse in Transmission* protocol was redundant in the scenarios depicted in figures 3.11, 3.12 and 3.13, and thus it was not shown. Figure 3.14 shows the performance improvement with the additional ACK for a similar transmission error scenario that was depicted in figure 3.13. The only difference in figure 3.14 is that the **sender** receives a NAK for frame 5 early in the transmission; it is irrelevant who sent it. The NAK for frame 5 was added in because in most cases there would be more than 21 frames and a NAK would usually be sent for an earlier frame. In the scenario depicted in figure 3.14, using the *Acknowledge Lapse in Transmission* protocol, **receiver 1** sends an ACK for frame 14 as soon as it has received the frame 5 retransmissions. The **sender** receives this ACK as it is retransmitting frame 20. When frame 20 has been retransmitted, the **sender** retransmits frames 16, 17 and 18; frames 14, 15 and 19 are already in the stream. Thus, the transmission is completed four time slots earlier than with the *Acknowledge Last Frame Only* protocol.

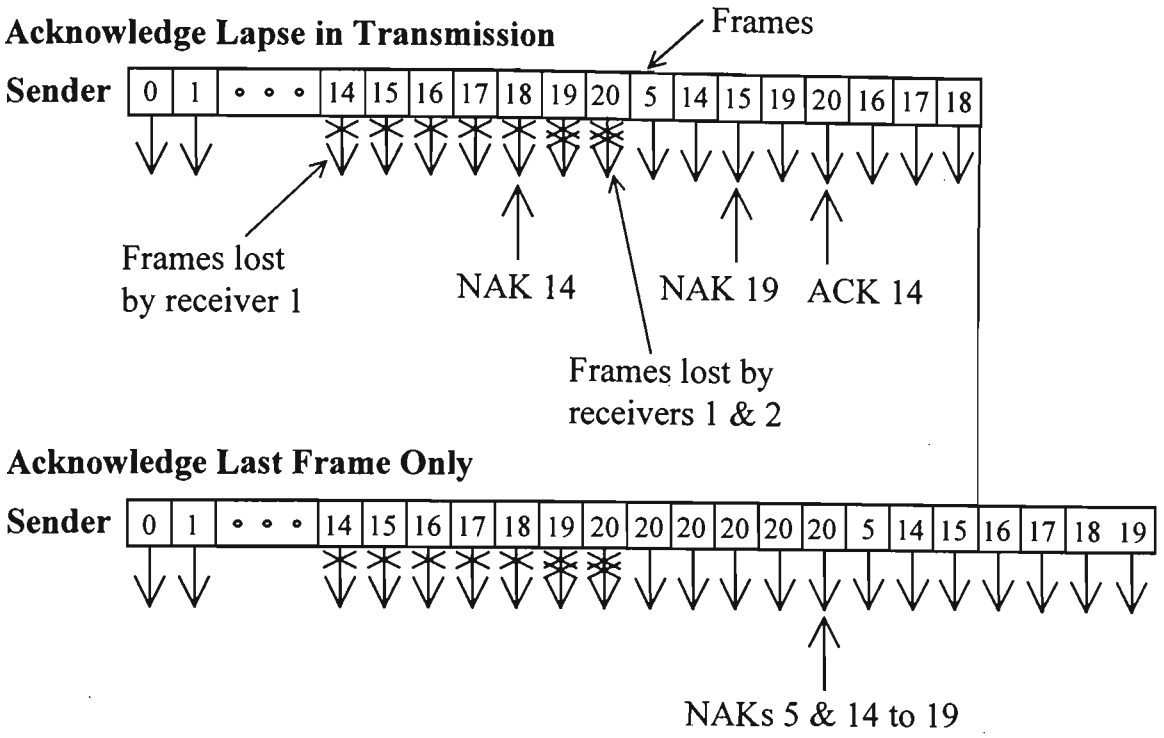


Figure 3.14. *The transmission efficiency comparison of using and not using an ACK for the last frame in the file received, when two receivers lose the tail of the transmission; same as figure 8, except that the NAK for frame 5 was added. The assumptions are that the round trip time is equivalent to the transmission of four frames, and that there are 21 frames in the file. The repeated retransmission of frame 20 at the end of transmission and the final ACK are not shown for brevity.*

The additional ACK may be redundant in some scenarios. If for example, **receiver 2** in figure 3.14 did not lose any frames, the ACK sent by **receiver 1** would be superfluous. This is because the NAK for frame 14 would result in the retransmission of all remaining frames as well. However, the additional ACK is most useful if the network is large, if the error rate is high, or if very large files are being transmitted. In each of these cases, the time taken for the *Retransmission List* to become empty can be substantial. Thus, receivers with good reception that happen to lose the tail of the transmission would incur significant delays.

Acknowledging a lapse in transmission improves performance in situations where receivers lose the frames in the tail of the transmission. In addition, receivers that do lose the frames in the transmission tail, receive the retransmissions earlier. The additional processing required by this method,

compared to acknowledgment of the last frame only, is insignificant. Moreover, the return traffic is actually reduced, because the ACK for the last frame received in the first transmission actually substitutes multiple NAKs.

The *Acknowledge Lapse in Transmission* and *Acknowledge First and Last Frames* protocols can be combined. The only condition is that an ACK for frame 1 can only be used as a session indicator, and not as a NAK for lost frames.

3.5.3 Retransmission

RSBT uses Selective Repeat retransmission rather than Go-back-N. The Go-back-N protocol is not suitable for high-speed and long-delay links [15],[37]. Hence, most new protocols use Selective Repeat—SNR [13] and SSCOP (Service Specific Connection Oriented Protocol) [42] for example. The RSBT protocol goes one step further in that the retransmission is done after all frames have been sent once. Therefore, as with NETBLT [7], transmitted frames are not kept in a buffer for retransmission purposes. They are recopied from the disk if needed.

If the transmission buffer is not large enough to hold all of the frames, then, as frames are transmitted, they are removed from the buffer so that other frames can be added in. After the last frame of the file has been copied from the disk into the buffer and as more room becomes available, RSBT starts recopying frames that need to be retransmitted. When the *Retransmission List* becomes short enough, RSBT recopies frames from the disk that will be sent when the *Retransmission List* becomes empty (as explained in section 3.5.2 for the four acknowledgment protocols). For example, with *Acknowledge Last Frame Only*, when the *Retransmission List* becomes empty the sender repeatedly retransmits the last frame. While with *Acknowledge Lapse in Transmission*, when the last frame in the *Retransmission List* has been sent, all frames that follow in the sequence are also retransmitted. In cases where the file to be broadcast is approximately 500 Kbytes or smaller, the sender may not receive any acknowledgments before all of the frames have been transmitted. In such situations, when room in the buffer becomes available, frame 1 is recopied from

disk, then frame 2 is recopied, and so on. Thus, when acknowledgments are received, frames may be retransmitted without delay, unless a very small buffer is used. When a frame is retransmitted, all of the frames with lower sequence numbers that were not requested for retransmission are removed from the buffer.

There are two common strategies used for the retransmission of lost or corrupted frames: Selective Repeat, or Go-back-N. The Go-back-N strategy works fine in point to point data transfer. However, it has one major disadvantage when broadcasting, particularly if some receivers are experiencing very bad reception (due to bad weather, inferior receiving equipment, etc.). This scenario is depicted in figure 3.15, where **Receiver 1** loses the first frame, and therefore, retransmission begins from frame 1. Then, **Receiver 2** loses frame 2 for the second time, so retransmission begins from frame 2, and so on. This could cause major delays for receivers not experiencing reception difficulties, such as **Receiver 3** in figure 3.15. In addition, using Go-back-N over a satellite link results in low throughput because of the large propagation delay. This is supported by experiments in [25], where Selective Repeat produced far better throughput results in point-to-point satellite transmission. Thus, Selective Repeat is the more appropriate retransmission strategy for RSBT.

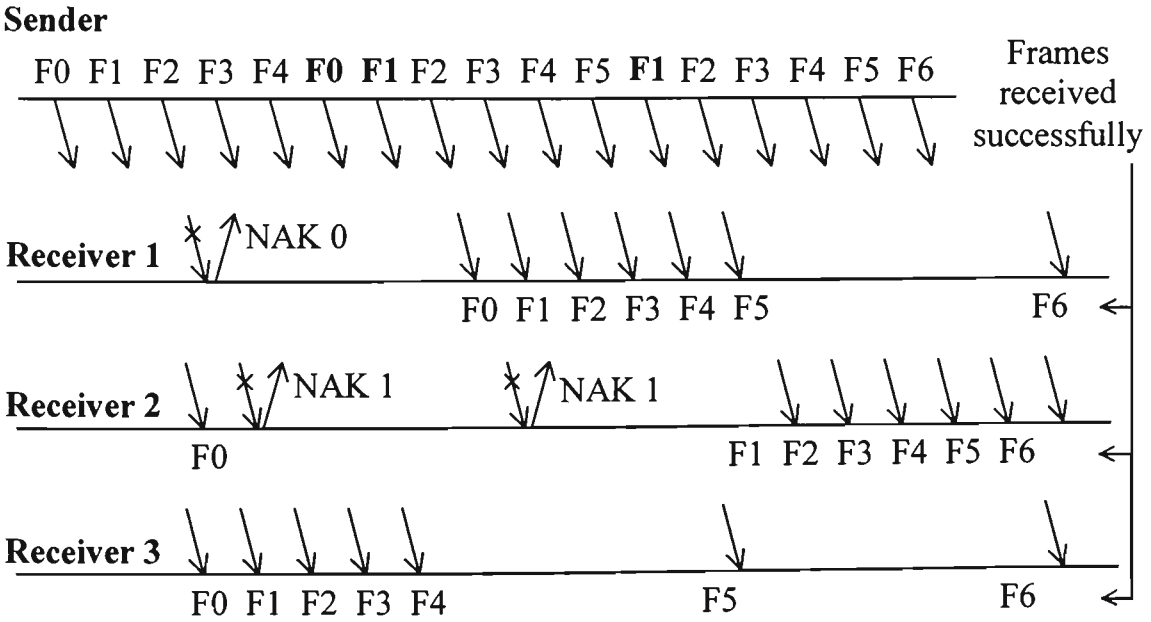


Figure 3.15. *A broadcast transmission scenario with Go-back-N retransmission.*

3.6 Summary of RSBT Implementation Options

This chapter has described the operation of the RSBT protocol and the options that it may use at the various stages of transmission. These options are shown in figure 3.16, and a textual summary is as follows:

1. Session Checking
    - 1.1. Test the existence of a session with each receiver.
        - 1.1.1. Wait for the sessions to be created (delay).
        - 1.1.2. Proceed with disregard to sessions not open.
    - 1.2. Proceed without checking for sessions with the receivers.
2. Frame Preparation
    - 2.1. Strip FTP header and encode critical header information into the RSBT "Information ID". Pack the RSBT frame with a suitable amount of data, disregarding FTP packet size.
    - 2.2. Strip FTP header and encode critical header information into the RSBT "Information ID". Include an exact multiple of FTP packets into the RSBT frame.
    - 2.3. Insert one complete FTP packet into the RSBT frame.
3. Broadcasting
    - 3.1. Broadcast as soon as enough data for one frame has been received.
    - 3.2. Broadcast only when the complete file has been received.
4. Error Control
    - 4.1. *Acknowledge All Frames.*
    - 4.2. *Acknowledge Last Frame Only.*
    - 4.3. *Acknowledge First and Last Frames.*
    - 4.4. *Acknowledge Lapse in Transmission*
5. Retransmission
6. LAN Transmission
    - 6.1. Transmit to the LAN as data arrives.
    - 6.2. Transmit to the LAN when the whole file has been received.

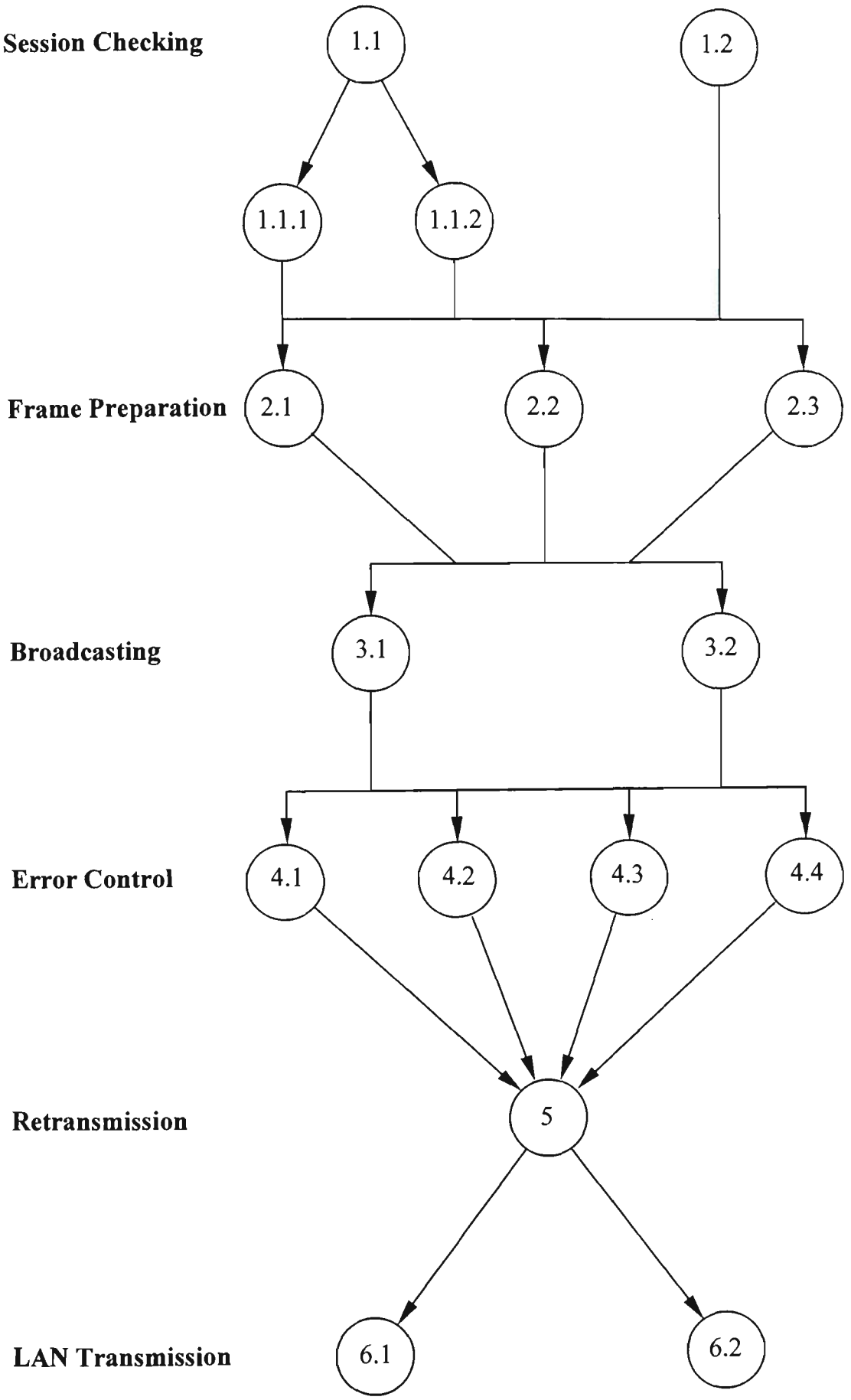


Figure 3.16. *Summary of RSBT implementation options.*

3.7 Conclusion

RSBT was developed to provide reliable data transfer to multiple receivers over a satellite link. Specifically, RSBT was designed to cater for the Bureau of Meteorology's requirements in broadcasting data to receivers using Optus Communications' Omnicast Digital satellite service. Tailoring RSBT for satellite broadcasting should result in much better transmission performance than that achievable by other protocols when used over a satellite link.

The inclusion of a satellite interface at each site facilitates uninterrupted transmission, and therefore maximises transmission efficiency. If RSBT were to operate from a workstation, it would have to compete with other processes for buffer space and CPU time, etc. The competition for the computer's resources could cause additional delays, which would not allow RSBT to take full advantage of the satellite channel capacity. The FTP/TCP transmission over the LAN should not cause any real delay, because as soon as enough data for one RSBT frame is received, the satellite transmission can begin. Furthermore, the FTP/TCP transmission throughput is approximately 7 Mbps, while RSBT transmits at 2 Mbps. Therefore, FTP provides data much faster than RSBT can send it over the satellite.

RSBT transmits all of the frames in the file once, before it starts any retransmission. Moreover, all frames are transmitted contiguously, which optimises the satellite link utilisation. However, the frame sequence numbers can not be recycled, and thus, the frame sequence number field must be large enough to hold the largest possible number. The problem is alleviated to a large degree by varying the size of the frame sequence number field to suit each new file transmission. However, the additional overhead created by the larger sequence number field should be insignificant, because RSBT is expected to use large frame sizes. The RSBT frame size can be large—it is not restricted—so it can be optimised to produce the best achievable throughput.

To improve the acknowledgment process, RSBT frames include two FCS fields: one for the header, and one for the data. The header loss rate should be much smaller than the data loss rate, so in most cases, when the data is received corrupted, the header will be intact, and a NAK can be sent earlier. However, the sender still has to wait for a long time for the acknowledgments to arrive, due to the propagation delay. For RSBT, the problem is compounded by the return traffic volume, because acknowledgments are sent by multiple receivers. Hence, the development of four acknowledgment methods to deal with the inherent difficulties. The pros and cons of each of these acknowledgment methods are as follows:

- **Acknowledge All Frames.** With this protocol, the sender completes a transmission quicker than with any of the other three protocols listed below. However, in scenarios where one receiver has temporarily lost its session, retransmission includes the whole file, and other receivers have to wait a little longer for their particular frames. The main problem though, is the return traffic volume, which could overwhelm the sender. Thus, this method is only suitable for small networks.
- **Acknowledge Last Frame Only.** This protocol produces the smallest amount of return traffic and requires the least amount of processing by the sender. Overall, this protocol is very simple, yet it is only marginally less efficient than the *Acknowledge All Frames* protocol. Therefore, it would be suitable for most implementations.
- **Acknowledge First and Last Frames.** This protocol has an advantage in scenarios where a receiver regains its session after all other receivers have finished; the session checking acknowledgment enables the sender to retransmit the file sooner. However, the maximum time saved is equal to the round trip time, and the gains are only realised in lost session scenarios. Moreover, the return traffic is increased during the peak time (during the first transmission). Therefore, this method is not warranted unless absolute efficiency is desired and the return traffic volume is not approaching its limits

- **Acknowledge Lapse in Transmission.** This protocol is a good enhancement of the *Acknowledge Last Frame Only* protocol. The retransmission of frames following the last frame in the *Retransmission List* can in itself save time equivalent to the round trip time. The early transmission of acknowledgments can also save time. Most significantly though, the ACK that is sent for the last frame that is received in the first transmission, can save time and actually reduce the return traffic because it also serves as a NAK for the remaining frames. So overall, this protocol is perhaps the most suitable solution for most implementations; there is virtually no additional processing required by the sender, yet, it is (marginally) more efficient than the *Acknowledge Last Frame Only* protocol.

4. SDL FORMAL SPECIFICATION

4.1 Introduction

With continuous developments in communications hardware, modern protocols demand strong, sophisticated, and reliable media standards to cater for the complex nature of communications today. Hence, Formal Description Techniques (FDTs) are used to develop protocols for systems that are complex, concurrent, quality-critical, safety-critical, security-critical or standardised [43].

FDTs share a common basis for specifying behaviour, namely labelled transition systems—systems whose state transitions are labelled with associated actions. SDL (Specification and Description Language) [43],[44],[45] is an example of an FDT. Two other examples of FDTs that are commonly used in the telecommunications industry are Estelle [43],[46] and Lotos [43],[47]. SDL is based on an extended state machine model, supplemented by features for specifying Abstract Data Types, and supported by complete formal semantics. SDL provides constructs for representing structures, behaviours, interfaces and communication links. It also provides constructs for abstraction, module encapsulation and refinement. All these constructs were designed to assist the representation of telecommunications system specifications, including aspects of services and protocols. SDL is widely used in the telecommunication community and is well supported by a variety of tools [43].

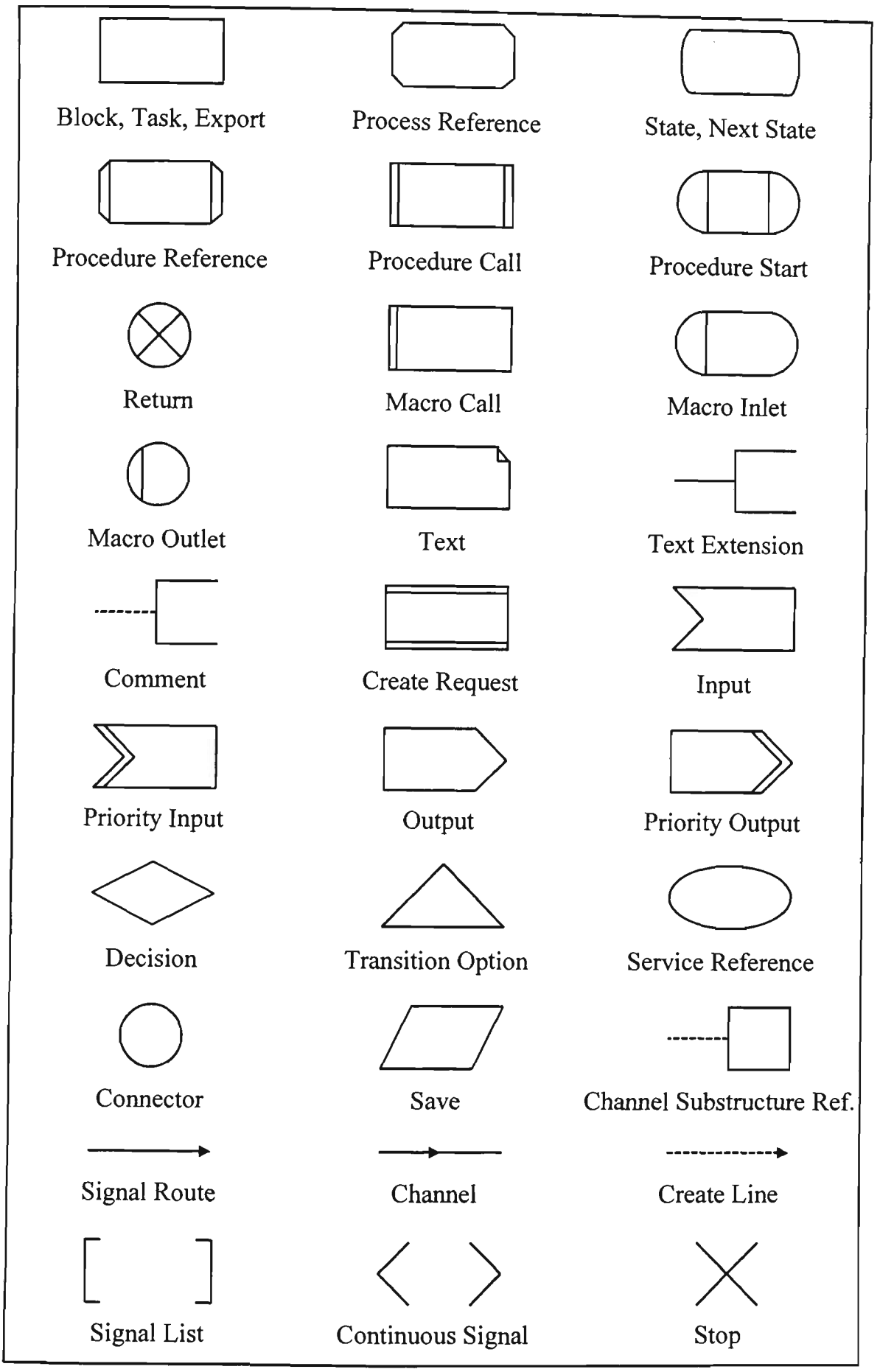
Protocol specifications play a major part in the development of a protocol. Software tools that support FDTs ensure that the specifications are unambiguous, clear, concise, complete, consistent, tractable, and conformed to [43]. These software tools generally use the specifications to model the system, and then simulate its operation. The simulations are used for analysis, testing, verification, and comparisons against other protocols. RSBT was formally specified with the

XMelba CASE tool, which is based on SDL. The XMelba software runs under the X-Windows UNIX system, and like PROSPEC [48], it provides a GUI, which enables very easy entry and modification of SDL specifications. Another feature that XMelba has in common with PROSPEC, is the provision for abstraction and modular construction, which simplify complex designs.

4.2 SDL Overview

A detailed description of SDL can be found in [43] and [44]. Briefly though, SDL specifications can be developed in two ways: *Phrase Representation* and *Graphical Representation*. The *Graphical Representation* is similar to a Flow Chart in appearance, and is much easier to develop and read. Thus, SDL's *Graphical Representation* was used to specify the RSBT protocol. SDL specification is basically a top-down design procedure, similar to the development of a Data Flow Diagram (DFD) in Structured Analysis and Design of software. With SDL, the system is specified at various levels of abstraction, using a range of *diagrams*: *system diagram, block substructures, block diagrams* and *process diagrams*. First of all, the *system diagram* specifies the system at the highest level of abstraction. This includes at least one *block* (a black box) and one or more *channels*, which are used for communication between two *blocks* and for communication between a *block* and the *Environment*. Then, the *block(s)* representing the system may be decomposed into a *block substructure*, and each *block* in the *block substructure* may be further decomposed into separate *block substructures*. Eventually, *blocks* are decomposed into *block diagrams*, which show the *processes* involved. Finally, the *process diagrams* specify the *processes* in detail. *Processes* may also include *procedure calls* and *macro calls*, which simplify the *process diagrams*. As with DFDs, the designer may determine the amount of decomposition in each step, but the implementation environment largely determines the level to which the specifications should be decomposed. For example, a system would have to be decomposed to a much lower level if it was to be implemented in Assembler rather than a fifth generation language. Table 4.1 shows the graphical symbols used by SDL [43].

Table 4.1. *SDL symbols.*



SDL provides a set of predefined *sorts*, *syntypes*, *newtypes* and *synonyms*. *Syntypes* are sorts with restricted set of values. A *Natural* is an example of a *syntype*, as it only allows *integers* greater than or equal to zero. *Newtypes* are user-defined structures that hold multiple variables of different *sorts*. And *synonyms* are names for constant values. For example, RSBT was specified with three receivers and so a variable was declared as: **SYNONYM recs = 3** (see **PROCESS PROC_ACK** *diagram* in Appendix A). Table 4.2 shows the predefined *sorts*, their *literals* and their operators. Further explanation of table 4.2 is as follows:

- *Integers* and *reals* use the '-' for negation and binary subtraction;
- *Float* converts an *integer* to a *real*, and *fix* does the opposite;
- *PID* is an acronym for Process Identification;
- *MkString* creates a one character string;
- *(index)* is used for traversing a *charstring*;
- '//' is a concatenation operator.

Table 4.2. *SDL predefined sorts with their literals and operators.*

Sort	**Literals**	**Operators**
Boolean	True, False	not, and, or, xor, =, /=, =>
Char	Character enclosed by ' '	=, /=, <, <=, >, >=
Integer	..., -1,0,1, ...	-, -, +, *, /, =, /=, <, <=, >, >=, Float, Fix
Natural	0, 1, 2, ...	as integer
Real	..., -1, 0, 1, ..., and ..., -9.99, ..., 1.11, ...	-, -, +, *, /, =, /=, <, <=, >, >=
PID	Null	=, /=
Duration	as Real	+, -, *, /, =, /=, >
Time	as Real	-, -, +, =, /=, <, <=, >, >=
Charstring	Character string enclosed by ' '	=, /=, //, MkString, Length, (index), First, Last, SubString(string, start position, length)

4.3 RSBT Specification

The SDL specification of the RSBT protocol was based on the *Acknowledge Last Frame Only* method of acknowledgment (see section 3.5.2.2). The specification was done at three levels of abstraction: *system*, *block* and *process*. The *system diagram* and the *block diagram* are shown figures 4.1 and 4.2, and the complete specifications are given in Appendix A.

The **SYSTEM RSBT** *diagram* in figure 4.1 shows the RSBT protocol as one *block*—**BLOCK BROADCAST**—a black box interacting with the *Environment*. The communication between the *block* and the *Environment* represents the communication between RSBT and the protocol used on the LANs. The **BLOCK BROADCAST** *diagram* shows the *processes* that compose the *block*: **INITIALISE**, **TRANSMIT**, **PROC_ACK**, **REC1**, **REC2**, and **REC3**. **PROCESS INITIALISE** is used only to initialises the other *processes* -an SDL requirement—so it is not discussed hereinafter. Finally, the *process diagrams* provide lowest level details of each *process*. The *process diagrams* are simplified by calls to common *procedures*, which are specified in *procedure diagrams*.

In the *system diagram*, the transmission process begins when the *Environment* sends the **IdNum** (file ID) and **NumFrames** (number of frames) *signals* to **PROCESS TRANSMIT**. If the transmission is to commence before the LAN transmission is completed, the **NumFrames** *signal* can be used only if a protocol such as FTAM is used on the LAN; FTAM uses control frame that includes the file size [40]. To get the data for the first frame, **BLOCK BROADCAST** uses a **Give** *signal* to send the sequence number zero to the *Environment*; the *Environment* sends the data in the **SendThis** *signal*. **BLOCK BROADCAST** uses the **Give** *signal* to acquire the data for all frames, including retransmissions, in the same way. At the receiving sites, **BLOCK BROADCAST** sends the **IdNum** to the *Environment* as soon as it is available; **RECEIVER1** uses *channel* **C2**, **RECEIVER2** uses *channel* **C3**, and **RECEIVER3** uses *channel* **C4**. When data packets arrive, **BLOCK BROADCAST** sends them to the *Environment* in the **Receive** *signal*.

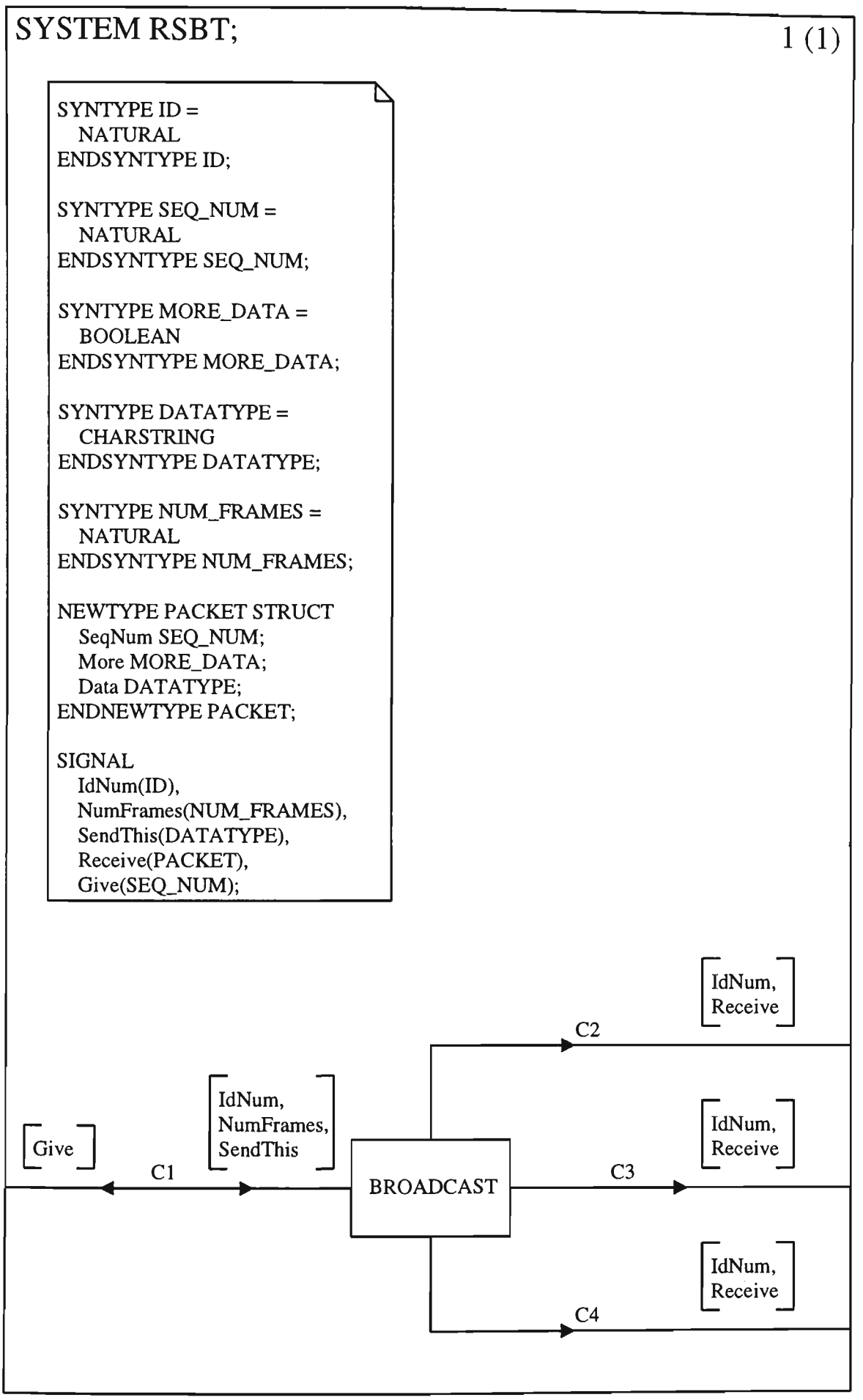


Figure 4.1. *SYSTEM RSBT diagram.*

In the **BLOCK BROADCAST** *diagram*, shown in figure 4.2, all communication with the *Environment* is the same as in the **SYSTEM RSBT** *diagram*, except that the *block diagram* shows the *processes* sending and receiving *signals*. Also, *signal routes* (**ET**, **R1E**, **R2E**, and **R3E**) are used instead of *channels*; the original *channels* are shown on the *block diagram boundary*.

The *block diagram* shows that **PROCESS TRANSMIT** sends the **IdNum** and **NumFrames** *signals* to **PROCESS PROC_ACK** as soon as the information is available. **PROCESS PROC_ACK** uses this information to validate acknowledgments (**Ack** *signals*). **PROCESS TRANSMIT** sends frames to the three receivers using the **Frame** *signal*. When the whole file has been sent, **PROCESS TRANSMIT** sends the **Next** *signal* to **PROCESS PROC_ACK** as a request for the next frame sequence number in the retransmission list. **PROCESS PROC_ACK** returns the frame sequence number in the **Resend** *signal*. As in the explanation of the *system diagram*, **PROCESS TRANSMIT** sends this frame sequence number to the *Environment* (**C1**) using the **Give** *signal*, and the *environment* sends the data in the **SendThis** *signal*. **PROCESS TRANSMIT** relays the data to the receivers in the **Frame** *signal*. This process continuous until **PROCESS PROC_ACK** has received an **Ack** *signal* from each receiver.

When **PROCESS PROC_ACK** receives three **Ack** *signals*, it sends the **Done** *signal* to **PROCESS TRANSMIT**, and in real terms the transmission is over. However, SDL requires all *processes* to be informed that all communications have concluded. Therefore, when **PROCESS TRANSMIT** has received the **Done** *signal*, it sends the same signal to the receivers, and then Stops. When the receivers get the **Done** *signal*, they pass it on to **PROCESS PROC_ACK**, and also stop. And finally, when **PROCESS PROC_ACK** receives the **Done** *signal* from all receivers, it too stops, and the transmission is complete.

The *process* and *procedure diagrams* are not explained because they are detailed at the lowest level and resemble program code.

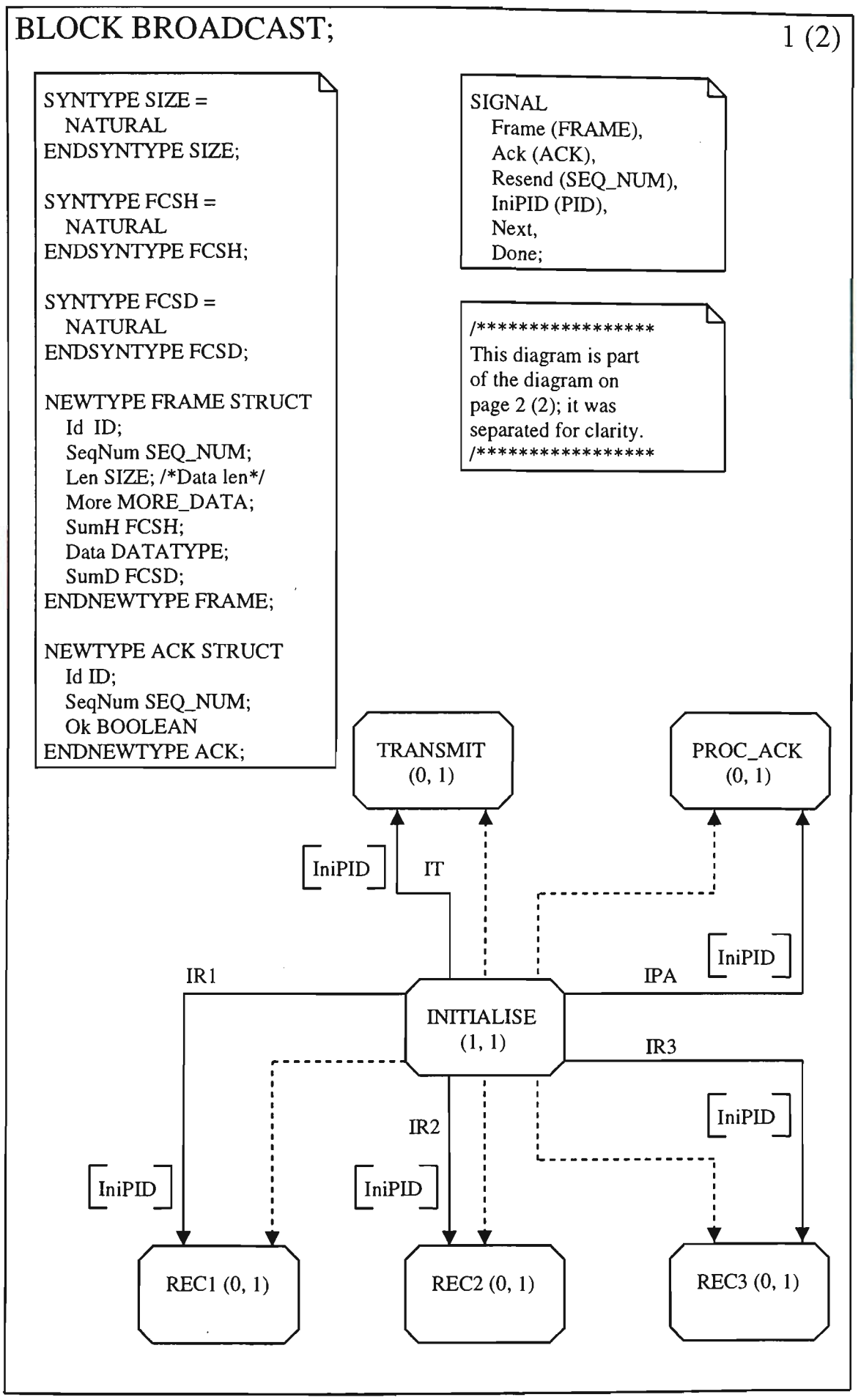


Figure 4.2. *BLOCK BROADCAST diagram (cont. on next page).*

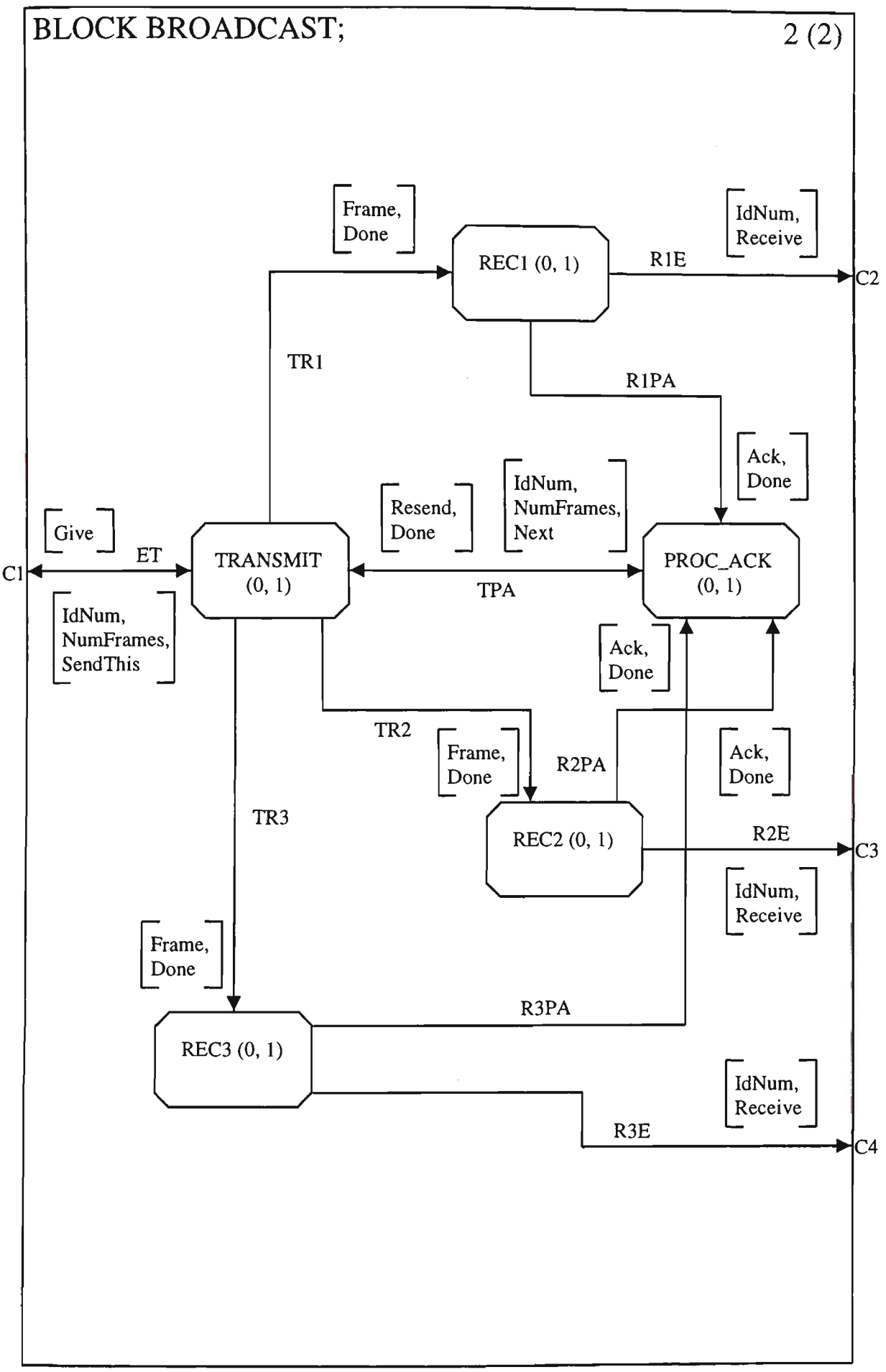


Figure 4.2. *BLOCK BROADCAST diagram (cont. from previous page).*

The purpose of the SDL specifications is to verify the RSBT semantics and to provide a basis for the protocol implementation. Therefore, the SDL specification must present an accurate description of the protocol. However, to avoid complex specifications detracting from the overall flow, some of the isolated functions were simplified. The main simplifications are as follows:

- **PROCESS TRANSMIT** gets data from the *Environment* for one frame at a time, instead of filling a buffer. However, since the SDL specification do not evaluate transmission efficiency, it is irrelevant.
- **PROCESS PROC_ACK** does not remove duplicate NAKs. Again, this is irrelevant because efficiency is not being measured.
- Receivers use a Received List (**RecList**) instead of the *Not Received List* (described in section 3.5.2.2) to select the required frames during retransmission. With the *Not Received List*, as retransmitted frames arrive, receivers would have to traverse the list and compare frame sequence numbers to determine whether the frame is required. In contrast to the *Not Received List* (integer array), **RecList** is an array of Boolean values; the array size is greater than the largest frame sequence number. Thus, as the retransmitted frames arrive, they are accepted if the **RecList** cell number corresponding to the frame sequence number has a value of zero (**RecList (SeqNum) = 0**).
- The **TimeOut** period used for the retransmission of NAKs is specific, not dynamic; receivers do not take into account retransmission requests made by other receivers (see section3.5.2.2). Specifying the full details would only distract from the logic flow of the protocol.
- The CRC procedure details are not specified because they are readily available and may be done by hardware. However, the *procedure calls* (**Calc_FCSH**, **Calc_FCSD**, **Test_FCSH** and **Test_FCSD**) were included in the specifications and simple Check Sum *procedures* were specified instead, so that XMelba could compile and analyse the specifications. These *procedures* are not shown in the specifications because they serve only as fill-ins for CRC procedures.

Figure 4.1 shows the *diagram* structure of RSBT specifications. The first three levels represent the protocol specification at the three levels of abstraction described above. The arrows in figure 4.3 indicate the procedure calls made by the *Processes*. The *Procedures* marked with an '*' are not shown in the specifications because the are common CRC procedures which were simplified in the SDL specifications. **RECEIVER2** and **RECEIVER3** are the same as **RECEIVER1**, so they are not shown in the figure 4.1 or in the specifications.

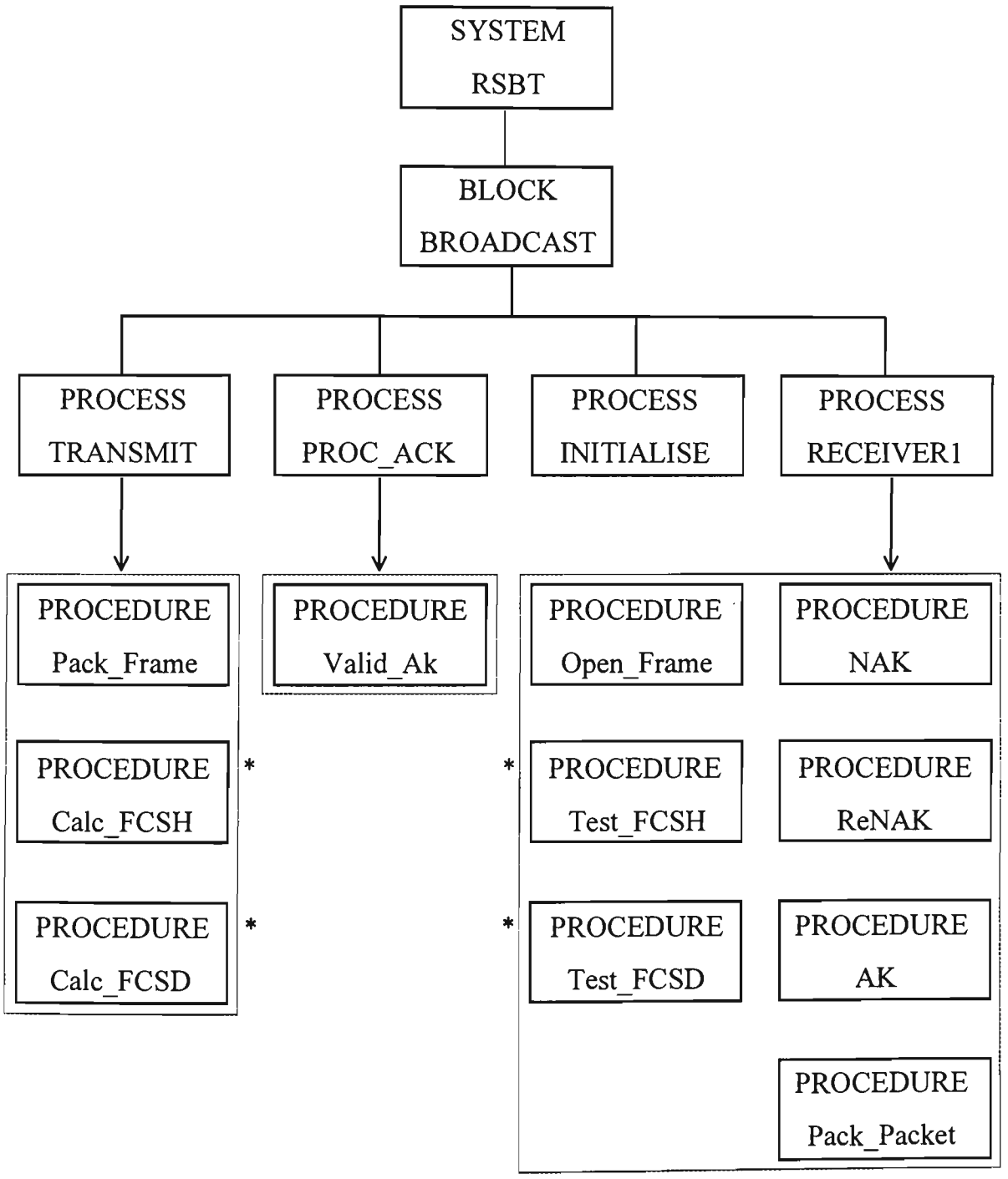


Figure 4.3. *Diagram structure of the SDL specification for RSBT.*

5. MATHEMATICAL MODELLING

5.1 Introduction

The RSBT protocol was described in Chapter 3 and formally specified in Chapter 4. This chapter describes the mathematical model that is used to evaluate the RSBT protocol. The mathematical model predicts the expected total transmission volume, including retransmission, and the overall transmission time for a given set of parameters. A simulation model of the RSBT protocol was also developed, and is described in Chapter 6. The simulation model provides much the same information as the mathematical model, as well as some additional information. The common information obtained by the two models is used as a means of validating each other (see section 6.8).

To derive the expected values for the transmission volume and the transmission time, the mathematical model produces the expected value for each step of a broadcast transmission. Specifically, the mathematical model provides the following information:

1. **Number of NAKs.** The number of NAKs received by the sender for the first transmission of all frames, the number of NAKs received for the second transmission (retransmission), etc.
2. **Number of duplicated NAKs.** The number of NAKs that are duplicated for the first transmission, and for each successive retransmissions. Duplication of a NAK occurs if multiple receivers send a NAK for one frame; because the transmission is broadcast, a frame is resent only once, not once for each NAK received.
3. **Number of retransmissions.** The number of frames that are sent in each successive transmission, which is equal to the number of NAKs received minus the number of duplicated NAKs.

4. **Proportion of retransmissions accepted/ignored.** The proportion of retransmitted frames that each receiver should accept. The remaining frames are disregarded—they represent frames that are required by other receivers. The accepted frames are treated as before. That is, a percentage equal to the frame error rate is regarded as being corrupted again, and thus, negatively acknowledged again.
5. **Total transmission volume.** The total number of bits transmitted. This includes frame headers and retransmissions.
6. **Last frame transmitted.** The average sequence number of the last frame to be sent in the final retransmission.
7. **Times to retransmit last frame.** The time between each retransmission of the last frame (see point 6). That is, the time to transmit that frame for the first time, second time, etc.
8. **Total transmission time.** The sum of all of the times described in point 7 plus the round trip time needed for the final acknowledgments to arrive at the sending site.

5.2 Transmission Assumptions

Although a model should represent the real system accurately, some assumptions and simplification are generally required to keep the model manageable, and to avoid insignificant details complicating and distracting from the main goal. The assumptions made for the mathematical model are as follow:

- **Consistent error rate.** The error rate is the same for all receivers. The mathematical model is not concerned with results of individual receivers. Therefore, using different error rates for each receiver would not enhance the model, yet, it would greatly complicate it.
- **No lost sessions.** Receivers do not lose their sessions. Lost sessions, which were explained in sections 3.5.1 and 3.5.2, were not included in the model for two reasons. First, they would greatly complicate the model. And second, detailed statistical information regarding the frequency and duration of lost sessions is not available.

- **No lost acknowledgments.** Reliable returns links are assumed.
- **No loss of frame headers.** Frame headers are never lost, and thus NAKs are never delayed. The inaccuracy that this assumption causes is insignificant. This is because the RSBT frame size is not restricted, and in most cases, the frame header loss rate will be far smaller than the frame data loss rate. Moreover, the transmission time is affected only if a NAK in the tail of the overall transmission is delayed.

The assumptions mentioned above do not affect the results in any significant way. Moreover, due to these assumptions, the mathematical model is valid for three (of the four) acknowledgment protocols: *Acknowledge Last Frame Only* (described in section 3.5.2.2), *Acknowledge First and Last Frames* (described in section 3.5.2.3), and *Acknowledge Lapse in Transmission* (described in section 3.5.2.4). The mathematical model is valid for all three of these acknowledgment protocols because the protocols differ only slightly, and only in scenarios where a receiver loses its session, or when the frame header is lost.

5.3 Parameters and Variables

The first step in deriving all of the information that was detailed in the previous section, is to provide a detailed list of parameters and variables that are required by the mathematical model. As a starting point, the mathematical model requires the following parameters:

R The number of receivers on the network.

E The frame error rate.

$\mathbf{F_1}$ The number of frames in the file (sent in the first transmission).

S The frame size in bits (including the frame header).

$\mathbf{t_r}$ The round trip time required for a frame to reach the receivers and an acknowledgment to arrive back.

C The channel speed of the satellite link in bps (used by the sender to transmit frames).

The complementary set of variables that is required by the model is as follows:

F_i The number of frames sent in the *i-th* transmission.

F The total number of frames sent in the whole file transmission, including all retransmissions ($F = F_1 + F_2 + F_3 + \cdots$).

N_i The total number of NAKs received by the sender for the *i-th* transmission.

$n_{i,j}$ The number of N_i NAKs that represent frames lost by exactly j receivers ($j = 1, 2, 3, \cdots, R$).

$A_{i,j}$ The proportion of frames in the *i-th* ($i > 1$) transmission that each receiver should accept as being required by it and j - 1 other receiver(s) ($j = 1, 2, 3, \cdots, R$).

A_i The overall proportion of frames in the *i-th* ($i > 1$) transmission that each receiver should accept ($A_i = A_{i,1} + A_{i,2} + A_{i,3} + \cdots + A_{i,R}$)

B_i The proportion of frames in the *i-th* ($i > 1$) transmission that each receiver initially accepts as frames that are required, but then rejects as frames that are corrupted again ($B_i = A_i E$),

m The total number of transmissions required by the sender to successfully deliver all of the frames in the file to all receivers; highest value of i used.

L The average sequence number of the last frame to be transmitted in the *m-th* transmission (last transmission).

Q The proportion of frames from any transmission (F_1, F_2, etc.) that represents frames up to and including frame L.

t_s The serialisation delay in transmitting one frame of size S.

T_i The time delay incurred between the *i-th* and *(i-1)-th* transmission of frame L ($i = 1, 2, \cdots, m$)

T The total transmission time.

$P_s(X)$ The probability that X number of receivers will receive a particular frame corrupted; s represents the number of receivers being considered ($s = 1, 2, 3, \cdots, R$); $X = 0$ to s, and is binomially distributed.

5.4 Transmission Volume Calculations

The RSBT transmission process can be broken down into distinct steps, which form the basis for the mathematical model. The mathematical model provides some initial information and then it loops through a set of equations. Each of these equations provides information for the following equation until the end condition is reached. The process begins with the sender transmitting all of the frames in the file once (F_1 frames). In response, receivers send a NAK for each frame that is received corrupted (N_1 NAKs). Then, the sender retransmits F_2 frames, which is derived by removing the duplication from the N_1 NAKs. On receipt of the frames in the second transmission, each receiver sends a NAK for each frame that is required but corrupted again (N_2 NAKs). For the third transmission, the sender retransmits F_3 frames; again the NAK duplicates are removed. The process continues in this manner until no further retransmission is needed. Then, the total transmission volume (F) is derived by summing up the number of frames in each of the transmissions ($F = F_1 + F_2 + F_3 + \cdots + F_m$).

Each frame that the sender transmits has a probability E of being received corrupted by each receiver. Receivers send a NAK for each frame that is received corrupted in the first transmission. Thus, the total number of NAKs that the sender receives for the first transmission is

$$N_1 = F_1 R E . \tag{5.1}$$

The N_1 NAKs that are received by the sender for the first transmission, are duplicated in cases where two or more receivers have lost the same frame. To determine the amount of duplication in the N_1 NAKs, the transmission error details must first be derived by the binomial distribution

$$\begin{aligned} P_s(X = r) &= \binom{s}{r} E^r (1-E)^{s-r} \\ \text{where } r &= 0, 1, 2, 3, \cdots, s . \end{aligned} \tag{5.2}$$

Letting $s = R$, equation (5.2) provides the following information about the first transmission:

- The proportion of frames that is received correctly by all of the receivers: $P_s\ (X = 0)$.
- The proportion of frames that is not received correctly by only one receiver: $P_s\ (X = 1)$.
- The proportion of frames that is not received correctly by two receivers: $P_s\ (X = 2)$.
- etc. up to $P_s\ (X = s)$.

The error details of the first transmission that were obtained by the binomial distribution equation (5.2), are used by

$$n_{1,j} = j\, F_1\ P_{s=R}(X = j)$$
$$\text{where } j = 1,\ 2,\ 3,\ \cdots,\ s \qquad (5.3)$$

to provide a complete breakdown of the N_1, which was derived by equation (5.1). Specifically, equation (5.3) provides the following information about the duplication of NAKs for the first transmission (N_1):

- The number of NAKs that represent frames lost by only one receiver—not duplicated ($j = 1$).
- The number of NAKs that represent frames lost by any two receivers—duplicated once ($j = 2$).
- The number of NAKs that represent frames lost by any three receivers—duplicated twice ($j = 3$).
- etc. up to $j = s = R$.

The duplicated NAKs must be removed because the RSBT transmission is broadcast to all receivers, so a frame needs to be retransmitted only once even if it was lost by multiple receivers. Hence, the number of frames to be sent in the

second transmission (F_2), and all transmissions that follow, is derived by the equation

$$F_i = \sum_{j=1}^{R} \frac{n_{i-1,j}}{j} \qquad (5.4)$$

where $i > 1$.

Receivers always receive all frames sent in any *i-th* transmission. However, with regard to retransmissions, each receiver only needs the frames that it received corrupted in the *(i-1)-th* transmission. Therefore, as a first step in deriving the proportion of frames in the *i-th* transmission that each receiver should accept, the equation

$$A_{i,j} = \frac{n_{i-1,j}}{F_i R} \qquad (5.5)$$

where $j = 1, 2, 3, \cdots, R,$
and $i > 1$

gives the proportion of frames that each receiver should accept as:

- frames not required by any other receiver ($j = 1$);
- frames required by only one other receiver ($j = 2$);
- frames required by two other receivers ($j = 3$);
- etc. up to $j = R$.

Thus, the overall proportion of frames in the *i-th* transmission that each receiver should accept is given by

$$A_i = \sum_{j=1}^{R} A_{i,j} \qquad (5.6)$$

where $i > 1$.

The remaining portion of frames in the *i-th* transmission (1 - A_i) is disregarded as frames required by other receivers. The frames that are accepted are treated as before. That is, each frame again has the probability E of being corrupted. Thus, the proportion of frames in the *i-th* transmission that were accepted by (5.6), that are rejected as frames corrupted again, is given by

$$B_i = A_i E \qquad (5.7)$$
where $i > 1$.

Each receiver sends a NAK for each frame in the *i-th* transmission that was rejected by equation (5.7) as a corrupted frame. Therefore, the number of NAKs that the sender receives for frames in the *i-th* transmission is given by

$$N_i = F_i B_i R \qquad (5.8)$$
where $i > 1$.

Depending on the transmission error rate (E), the number of frames in the file (F_1), and the number of receivers on the network (R), duplication of NAKs may occur beyond the NAKs for frames in the first transmission. Therefore, just as equation (5.3) provided a breakdown of the N_1 NAKs, the equation

$$n_{i,j} = \sum_{r=0}^{R-j} \frac{n_{i-1,j+r}}{j+r} \, j \, P_{s=j+r}(X = j) \qquad (5.9)$$
where $j = 1,\ 2,\ 3,\ \cdots,\ R$

provides a breakdown of any N_i. Here, $s = j + r$ represents the number of combinations in the binomial distribution equation (5.2). Equation (5.9), like equation (5.3), gives the number of NAKs that are:

- not duplicated ($j = 1$),
- duplicated once ($j = 2$),
- etc. up to $j = R$.

From this point on, equations (5.4) to (5.9) are repeated to obtain the number of frames in each successive transmission (F_i) and the corresponding number of NAKs (N_i).

The number of NAKs that are duplicated diminishes dramatically with each retransmission. When the amount of NAK duplication approaches zero—say less than 0.1—the duplication is disregarded. Therefore, each frame in any further retransmission is required by only one receiver. Thus, simplified forms of equations (5.4) and (5.7) may be used, and equations (5.5), (5.6) and (5.9) become redundant; equations (5.8) is used as before. First, equation (5.4) may be replaced with

$$F_i = N_{i-1}$$
$$\text{where } i > 1$$

to derive the number of frames in the *i-th* transmission. Then, equation (5.7) may be replaced with

$$B_i = \frac{E}{R} \tag{5.10}$$

to determine the proportion of frames in the *i-th* transmission that each receiver should reject as frames that are required, but corrupted again.

When the number of NAKs (N_i) becomes insignificant—say 0.1 or lower—the transmission is considered complete. Then, the total number of frames transmitted, including all retransmission, is derived by

$$F = \sum_{i=1}^{m} F_i \,, \tag{5.11}$$

where m represents the number of transmissions required to broadcast the file successfully.

5.5 Transmission Time Calculations

This section completes the RSBT mathematical model by deriving the expected total transmission time that is required by the sender to complete a file broadcast. The total transmission time is derived by a step-wise procedure based on the results obtained from the transmission volume calculations, which were described in section 5.4. First, the mathematical model gives an expression for the expected (average) sequence number of the last frame to be transmitted (L) by the sender in the last transmission. Then, the mathematical model gives the expected time required to transmit frame L in the first transmission (T_1), then the expected time required to transmit frame L in the second transmission (T_2), and so on. The expected total transmission time (T) is obtained by simply summing up all T_i values (T_1, T_2, $\cdots$, T_m) and adding the round trip time (t_r) that is required for the final acknowledgment to arrive at the sending site.

The average sequence number of the last frame to be transmitted by the sender in the *m-th* transmission (the last transmission) is given by

$$L = \frac{F_m}{F_m + 1}(F_1 + 1) \tag{5.12}$$

if F_m is an integer. If F_m is a real number greater than 1, then equation (5.12) is modified to

$$L = \frac{\lfloor F_m \rfloor}{\lfloor F_m \rfloor + 1}(F_1 + 1)(\lceil F_m \rceil - F_m) + \frac{\lceil F_m \rceil}{\lceil F_m \rceil + 1}(F_1 + 1)(F_m - \lfloor F_m \rfloor), \tag{5.13}$$

where $\lfloor x \rfloor$ is the biggest integer less than x, and $\lceil x \rceil$ is the smallest integer greater than x. If F_m is a real number less than 1, then equation (5.13) can be simplified to

$$L = \frac{F_1 + 1}{2}. \tag{5.14}$$

Equation (5.14) produces the same result as equation (5.12) when the latter uses an F_m value of 1. Therefore, equation (5.14) always results in L having the same value as if one frame is to be sent in the last transmission. This is because on average, the *m-th* transmission will contain only one frame in every $1/F_m$ file transmissions. If for example, $F_m = 0.25$, then on average, the sender will transmit one frame in the F_m transmission in every fourth file transmission, the file transmissions in between will conclude with the *(m-1)-th* transmission. Therefore, in the scenario described above, when the *m-th* transmission is used, it contains exactly one frame. The fact that the *m-th* transmission in the above example is used in every fourth file transmission is compensated by equation (5.19), which is described later.

The proportion of any transmission (*1, 2, 3,* $\cdots$, *m*), that represents frames up to and including frame L (obtained from equation (5.12), (5.13), or (5.14)), is given by

$$Q = \frac{L}{F_1 + 1} \,. \tag{5.15}$$

The transmission serialisation delay in transmitting each frame, t_s, is given by

$$t_s = \frac{S}{C} \,, \tag{5.16}$$

where S is the frame size in bits, and C is the channel speed in bps. Then, using equations (5.15) and (5.16), the equation

$$T_1 = (F_1 + 1)\, Q\, t_s \tag{5.17}$$

represents the time that is required to transmit all frames up to and including frame L in the first transmission. An example of equation (5.17) is shown in figure 5.1—a graphical explanation of how the values for T_1 and T_2 are obtained.

The time that is required between each successive transmission of frame L (between the first and second transmission, the second and third transmission, etc.) is obtained from the expression

$$T_i = \max \begin{Bmatrix} t_s\left(\left(F_{i-1} - \left(F_{i-1}+1\right)Q\right) + \left(F_i+1\right)Q\right) \\ \\ t_r + t_s \end{Bmatrix} \qquad (5.18)$$

where $i = 2, 3, 4, \cdots, m$.

Equation (5.18) is only valid while F_i is greater than or equal to 1. When F_i becomes less than 1, the value of T_i is obtained from the expression

$$T_i = \max \begin{Bmatrix} t_s\left(F_{i-1} - \left(F_{i-1}+1\right)Q + F_i\right) \\ \\ F_i\left(t_r + t_s\right) \end{Bmatrix} \qquad (5.19)$$

where $i = 2, 3, 4, \cdots, m$.

Figure 5.1 also shows an example of the composition of the first part of equation (5.18): t_s ((F_1 - (F_1 + 1) Q) + (F_2 + 1) Q). As shown in figure 5.1, to derive T_2, the frames transmitted between the first and second transmission of frame L are split into two groups. First, F_1 - (F_1 + 1) Q represents the frames in the first transmission that are sent after frame L. Second, (F_2 + 1) Q represents frames up to and including frame L in the second transmission. These frames are shown in figure 5.1 as frames up to and including frame L in the first transmission that are corrupted. The total number of frames that are transmitted between the first and second transmission of frame L is the sum of F_1 - (F_1 + 1) Q and (F_2 + 1) Q. Then, T_2 is the product of t_s, and (F_1 - (F_1 + 1) Q) + (F_2 + 1) Q.

Finally, the total time required to successfully deliver all of the frames in the file to all receivers, is given by

$$T = t_r + \sum_{i=1}^{m} T_i \,. \tag{5.20}$$

If small files are broadcast, and no retransmission is required, then instead of using equation (5.20), the total transmission time may be obtained from

$$T = F_1 \, t_s + t_r \,. \tag{5.21}$$

If equation (5.21) is used, then obviously equations (5.12) to (5.20) are not required.

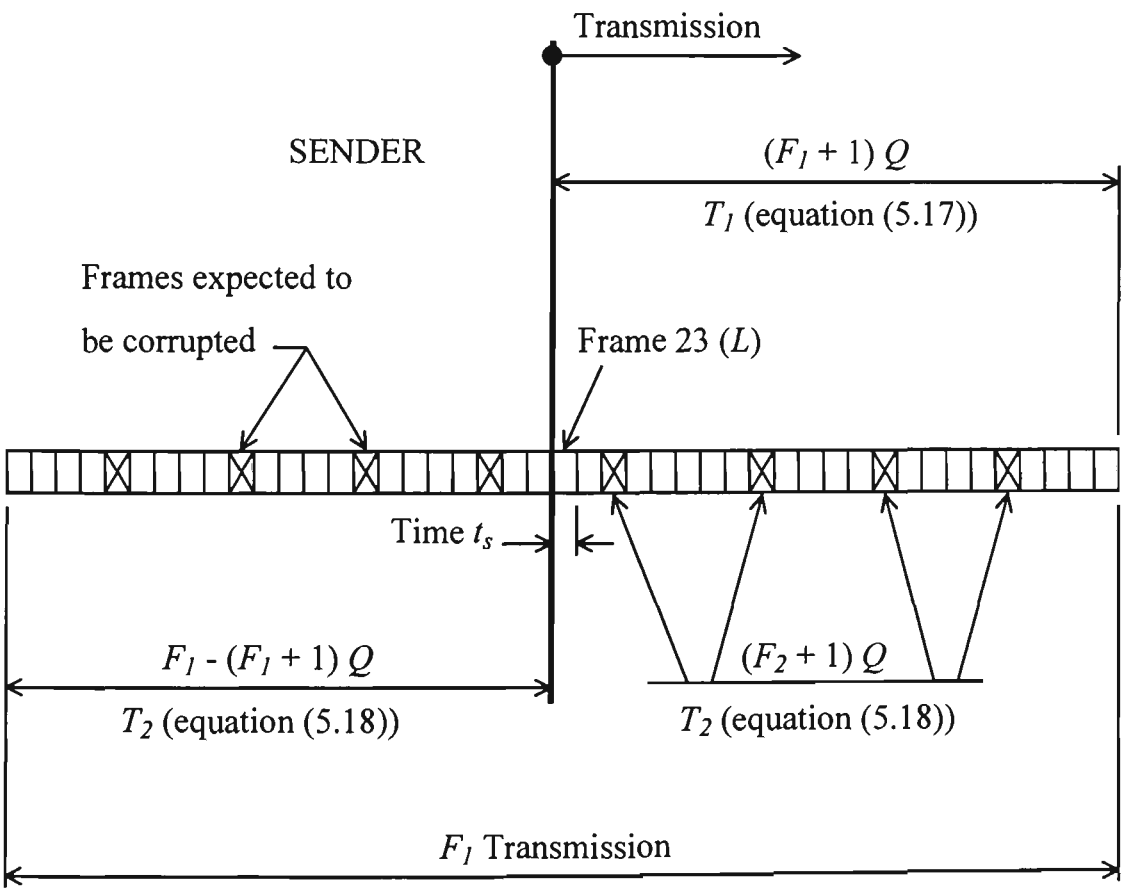


Figure 5.1. *Transmission segmentation used by (5.17) and (5.18) to calculate "T_1" and "T_2". $F_1 = 45$, $F_2 = 8$, $F_3 = 1$, $L = 23$, and $Q = 0.5$.*

5.6 Conclusion

The mathematical model was developed to obtain the total transmission volume and the total transmission time, for a given set of parameters. The transmission volume calculations and the transmission time calculations progress in a step-wise manner, with each step providing information for the next. One set of details that is obtained by the step-by-step procedure is the number of frames in each successive retransmission. The simulation model, which is described in Chapter 6, also reports on the number of frames that are sent in each retransmission. Therefore, the step-by-step details allow the results of the two models to be validated in detail, rather than just comparing the end results.

The simulation model parameters are limited because of the demand made on the computer's resources, and because of the time required to complete the simulation. The mathematical model on the other hand, is not limited in any way, and can be used with any extreme set of parameters.

The assumptions (or simplifications) that were made greatly simplified the mathematical model, yet, they do not compromise the results in any significant way. There are no shortcuts made in the step-wise progression of the model, and the results of the individual equations are not rounded off to make the following equations easier. Therefore, the mathematical model provides accurate results for any given set of parameters.

6. Network II.5 Modelling

6.1 Introduction

Modelling and simulation is an effective and widely used method of evaluating real-time systems. Potential designs may be modeled separately, and the simulation results compared to determine which design is better for a particular set of requirements. Simulation can also be used to determine optimal parameter settings for a given set of variables, so that efficiency of a system can be maximised.

RSBT was modeled and simulated with the Network II.5 design tool. An overview of Network II.5 is given in section 6.2. Like the mathematical model, which was described in Chapter 5, the simulation model is also used to evaluate the RSBT protocol. In addition, as stated in section 5.1, the mathematical model results and the simulation model results are used to validate each other.

The development of software generally follows some formal procedure, such as that described by the Water Fall Model or the Spiral model. Similarly, the Network II.5 modelling and simulation process was carried out according to the following 10 steps [49]:

Step 1. Problem Formulation

This step gives a clear and concise description of the problem, in regards to simulation modelling requirements. This includes the modelling and simulation objectives, and any assumptions that are made. This step is described in section 6.3.

Step 2. Model Building

This step provides a description of a model or method that is used to verify the simulation model. This may be a prototype system,

a mathematical model, etc. The limitations of the model used as a basis for the simulation model must be clarified, including any assumptions. This step is described in section 6.4.

Step 3. Data Collection

This step involves the identification and compilation of all data that is required for the simulation model. Examples of such data include error rate, communication channel speed, processing speed, etc. This step is described in section 6.5.

Step 4. Model Translation

This step describes the translation of the conceptual model to the simulation model, using the chosen modelling tool. This involves mapping the specifications of the model developed in Step 2, to specifications that are suitable for the simulation model. In general, the mapping should be precise. However, estimations or simplifications may be made for trivial features of the simulation model that do no have a significant affect on the simulation results. In situations where the simulation tool can not model specific requirements that are critical, an alternative method must be devised so that the simulations produces correct results. This step is described in section 6.6.

Step 5. Model Verification

This is the process of verifying that the simulation model behaves as it was intended to. This may be achieved by tracing a simulation and comparing the actions with the actions of the model developed in step 2. This step is described in section 6.7.

Step 6. Model Validation

This is the process of validating the simulation results. This involves comparing the simulation results with results obtained by another method, to be certain (or at least confident) that the simulation results are valid. Other methods that can be used to obtain results include prototype implementations, mathematical or statistical models, etc. This step is described in section 6.8.

Step 7. Experiment Planning

This is the design of a strategic plan that details the simulations that need to be run to ensure that the simulation objectives are met. This involves selecting appropriate value sets for each parameter, and matching up parameter settings in a systematic way. The aim of the plan is to obtain the desired results and at the same time minimise the number of simulations to be run. This step is described in Chapter 7.

Step 8. Experimentation

This step focuses on the execution of the simulation experiments that were planned in step 7. The raw simulation results are given in Appendix C; the raw mathematical model results are given in Appendix B.

Step 9. Analysis of Results

This is the analysis of all simulation results that were obtained in step 8. This involves explaining the results, drawing inferences, and making recommendations. The simulation results and the mathematical model results are both analysed in Chapter 7,

Step 10. Documentation and Conclusion

This step includes the derivation of conclusions about the system that was simulated, and the documentation of the previous nine steps. The document can then be used to assist implementation of the system, and for the maintenance or reuse of the simulation model. The conclusions derived from both the simulation results and the mathematical model results are given in section 7.4.

The above 10 steps are performed sequentially, with one step building on to the previous. Enhancements or corrections at any step require a review of the previous steps where consequential alterations may be required. When alterations are made to preceding steps, care must be taken to walk through the steps in turn again, to ensure that all the required changes have been made at each of the following steps.

6.2 Network II.5 Overview

Network II.5 is a design tool used for modelling and simulating computer systems; it can be used to resolve whether a proposed system configuration can meet the given criteria, and it can be used to evaluate competing designs. The Network II.5 package provides three main functions: system description, system simulation, and system analysis. The system description, both hardware and software, is specified via a graphical user interface; the model is verified and debugged using tools provided by Network II.5. Then, various simulation runs are executed with different parameter values. Each simulation produces a data set, which is used by the various analysis tools capable of generating animations, plots, and reports [50].

6.2.1 Hardware Specification

The modelling process begins with the hardware specification. The hardware devices that are available in Network II.5 are: *processing elements (PEs)*, *storage devices (SDs)*, *transfer devices (TDs)*, *gateways, and LANs*. A *gateway* is actually a special case of a *PE*, and a *LAN* is a special case of a *TD*. To build the system architecture, first, all of the hardware devices that are required for the system are created with the buttons provided in the tool bar on the left side of the display (see figure 6.1). Then, all of the *PEs*, *SDs*, and *gateways* are connected to the appropriate *TDs* and *LANs*; the connections are created with the **connection** button in the tool bar (the **connection** button is the one with the diagonal line). An example of a simple system is shown in figure 6.1, where the **PE 1** *PE* can get data from the **SD 1** *SD* over the **TD 2** *TD*, and then transfer the data to **PE 2** *PE* over the **TD 1***TD*.

When the model's hardware architecture has been completed, all of the hardware components must be defined. A component is defined by first double clicking on it with the mouse (or selecting it through the menus) and then entering the details on a form displayed on the screen. Figure 6.2 shows the form that is used to define an *SD*. The forms that are used to define *PEs* and *TDs* can be seen in section 6.6.2, where they are used to specify the RSBT simulation model.

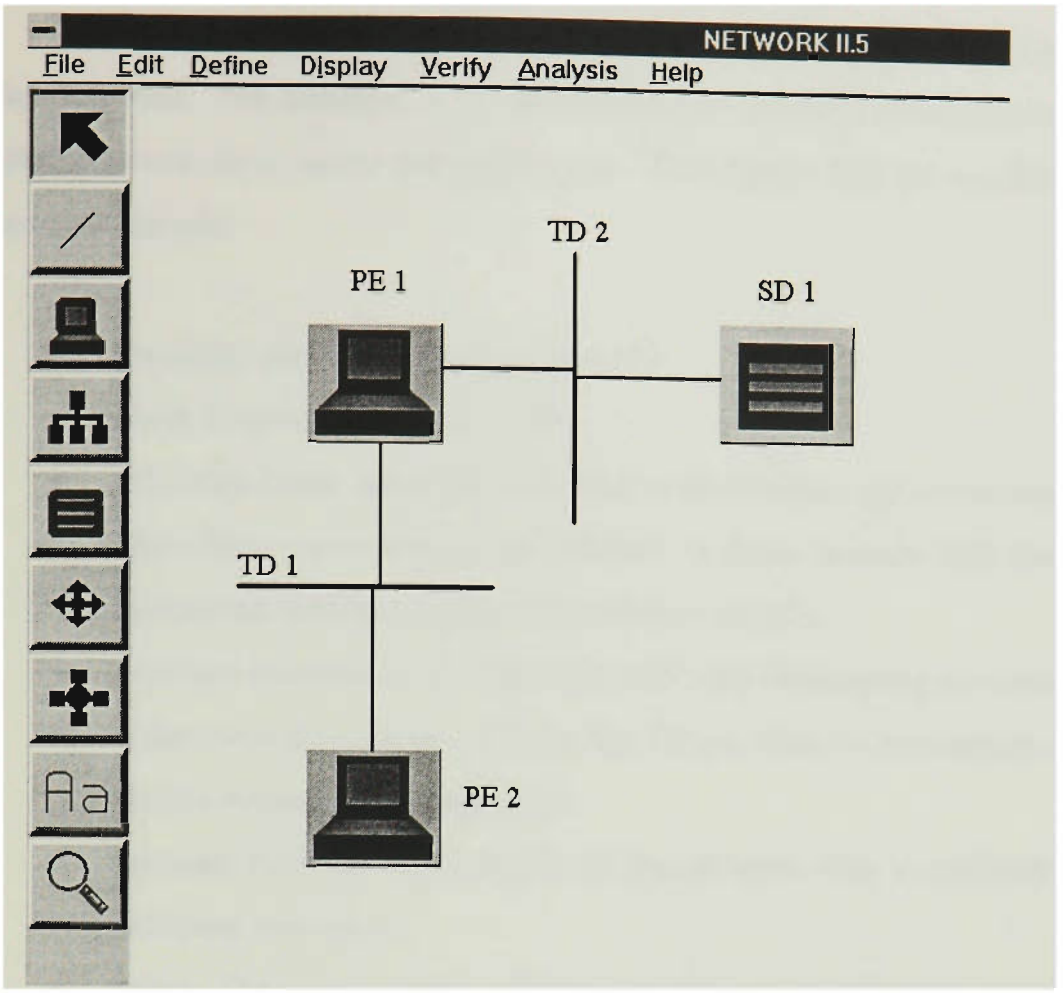


Figure 6.1. *A simple Network II.5 Model.*

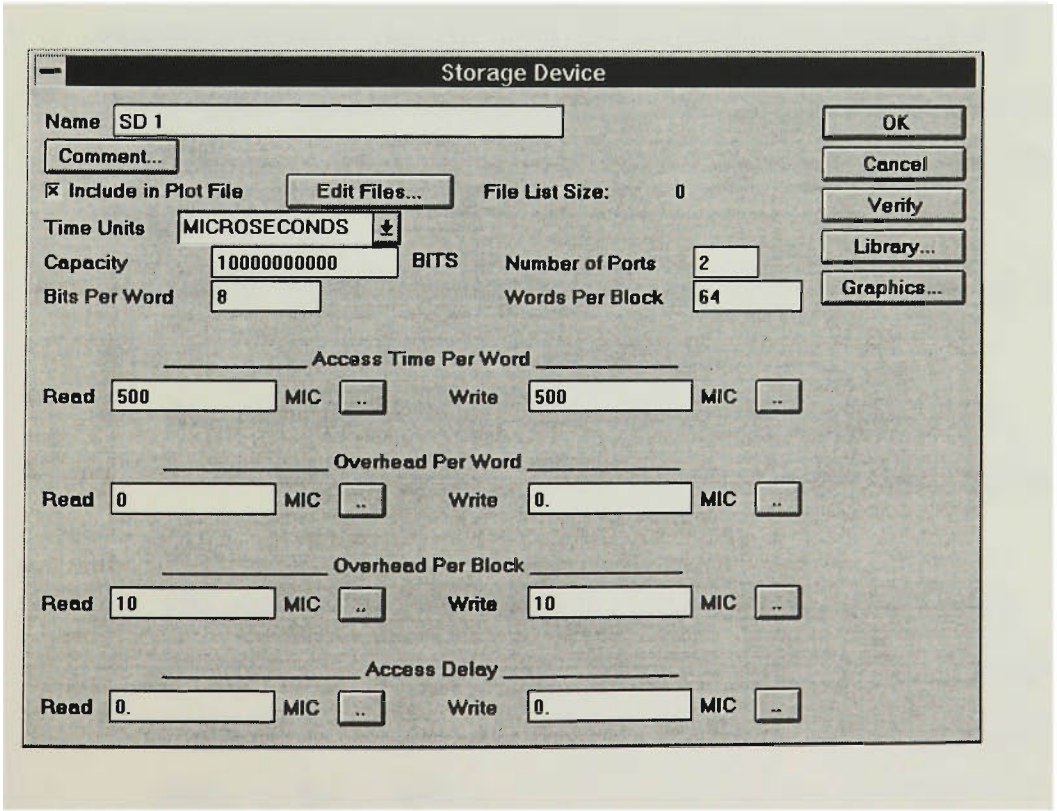


Figure 6.2. *SD specification form.*

A *PE* is used to model any hardware device that executes instructions or makes decisions. For example, a *PE* may be used to model a bus controller, a display, a sensor, or a central processing unit. The details that are required to define a *PE* include:

- *Quantity*—the number of identical *PEs*.
- *Cycle Time*—clock cycle speed.
- *I/O Setup Time*—setup time for *read*, *write* and *message instructions*.
- *Time Slice*—processing time allocated to those *modules* that can be interrupted, which is indicated by an *interrupt flag*.
- *Interrupt Overhead*—the delay incurred when interrupting a *module*.
- *Input Controller*—if set to **True**, the *PE* can execute *instructions* and receive *messages* simultaneously.
- *Message List Size*—the amount of storage space that is available for incoming *messages*.
- *Queue Flag*—determines whether the *PE* stores received messages that are duplicates.
- *Lose Overflow Messages*—determines whether or not received *messages* are lost when the buffer (*message list size*) becomes full.
- *Keep Blocks Separate*—if set to **False**, each *block* that is received in a transmission is combined to form a single *message*, otherwise *blocks* are not combined.
- *Instruction Repertoire*—the list of *instructions* that the *PE* can execute. The available *instruction* types are:
  - *Processing Instructions*—used to execute a process on a *PE*.
  - *Read/Write Instructions*—used to move data between an *SD* and a *PE*.
  - *Message instructions*—used for communication between *PEs*.
  - *Semaphore Instructions*—used for *setting* and *resetting* *semaphores*, and for *incrementing*, *decrementing* and *assigning* a value to the *semaphore counter*; *semaphores* are explained in section 6.2.2.

A *PE's instruction repertoire* is listed on the *PE's* specification form (see figure 6.5 in section 6.6.2.1). There are five type of *instructions*: *message, processing, semaphore, read and write*. An *instruction* is added to a *PE's instruction repertoire* by simply clicking on the *Add* button provided on the *PE* form, and then filling in the details. The details of the five *instruction* types are not shown for brevity. However, the details of *message, processing* and *semaphore instructions* can be seen in figures 6.6, 6.7 and 6.8 (in section 6.6.2.1).

An *SD* is used to model anything that stores data. As shown in figure 6.2, the details that are required to define an *SD* include:

- *Capacity.*
- *Bits Per Word.*
- *Words Per Block.*
- *Number of Ports*—the number of *PEs* that can access the *SD* simultaneously.
- *Read Word Access Time*—the time that is required to read one word.
- *Write Word Access Time.*
- *Read Word Overhead Time.*
- *Write Word Overhead Time.*
- *Read Block Overhead Time.*
- *Write Block Overhead Time.*
- *Read Access Delay*—the time delay before the *SD* can be read.
- *Write Access Delay.*

TDs are used to model communication links between *PEs*, *gateways* and *SDs*. The details that are required to define a *TD* include:

- *Protocol*—used for solving contention between *PEs* for a *TD*.
- *Cycle Time*—the *TD* clock speed; if *bits per cycle* is one, then the *cycle time* is the reciprocal of the channel speed.

- *Bits Per Cycle*—the number of bits transmitted in one *cycle* (values greater than 1 are used for parallel transmission).
- *Cycles Per Word.*
- *Words Per Block.*
- *Word Overhead Time.*
- *Block Overhead Time.*
- *Minimum Bits to Send*—the smallest *block* that can be transmitted; the system adds padding if required.
- *Block Error Probability.*
- *Scale Error Probability Flag*—if a *block* is padded (if it is not full with data), this flag can be used to scale the *block error probability.*
- *Separate Blocks Flag*—if set to true, the each *block* in the *message* is treated as a separate transmission.
- *Connection List*—a list *PEs*, *gateways* and *SDs* connected to the *TD.*

6.2.2 Software Specification

Section 6.2.1 described the specification of the model's hardware components and the *instructions* that can be executed by each *PE.* The simulation model is completed by specifying the software components—*modules*, *semaphores*, *instruction mixes*, *macro instructions* and *files.*

Network II.5 *modules* are specifications for functions that a *PE* must perform. A *module* resembles high-level pseudocode, and its *instruction list* contains a set of *instructions*, which are selected from the *PE's instruction repertoire.* A *module's instruction list* may also include *instruction mixes*, and *macro instructions*, which are explained later. A *module* has four stages in its life cycle, and *module* specifications have four corresponding parts, which are:

- *Scheduling Conditions*—there are four possible *scheduling conditions*, and each *module* must have at least one:
  - *Time-Based—start time*, *stop time* and *iteration period.*
  - *Awaited*—waits for a *predecessor module* to invoke it.

  - *Checked Once*—also waits for a *predecessor module* to invoke it, but also requires either an appropriate *semaphore* setting, the existence of a *file*, or a particular *message* to be received.
  - *System State*—waits for an appropriate *semaphore* setting, the existence of a *file*, a particular *message* to be received, or a particular *hardware status*.
- *Host Processing Element Options*—the *module* must queue up for a *PE*, which is selected from the *module's allowed/resident processor list*. In addition, options for *delays* and *interrupts* are available in the *module controls*.
- *Instructions*—the list of *instructions* that are executed during a simulation.
- *Module Completion*—when the execution of instructions has been completed, the *module* must select a *successor module* (if one exists). If there are more than one *successor modules*, the *module* may choose a particular *module* as a successor according to the *scheduling conditions* of the candidates, or it may select a *statistical successor*. With *statistical successors*, each candidate *successor module* has an associated probability of being selected; the sum of the probabilities must equal to 100%, and the *successor module* is randomly selected from a uniform distribution.

As mentioned above, *modules* may include *instruction mixes* and *macro instructions* in the *instruction list*. An *instruction mix* is a list of *instructions*, *instruction mixes* and *macro instructions*, with a probability associated with each item; the probabilities must add up to 100%. When an *instruction mix* is invoked, an item from the list is randomly selected with the associated probability using a uniform distribution. A *macro instruction* is an *instruction list* that is not linked to any particular *module*. However, a *macro instruction* can be executed by any *module* as long as the host *PE's instruction Repertoire* includes all of the *instructions* in the *macro instruction*.

Instructions that are executed by a *module* may involve *semaphores* and *files*. *Semaphores* are global flags with both a binary value and an integer *count*. *Semaphores* have an initial *semaphore status* (**Set** or **Reset**) and an initial *count*. During a simulation, a *semaphore instruction* can *Toggle* the *semaphore status*, *increment* or *decrement* the *semaphore count* by any integer value, or *assign* any integer value to the *semaphore count*. *Files* may be created during the modelling procedure, or they may be created dynamically by a *write instruction* during a simulation.

6.2.3 Simulation and Analysis

When the all of the model's hardware and software components have been defined and verified, the model is ready for simulation. The simulation process may be started by first selecting the *Analysis* option from the menu-bar, and then *Start Simulation* from the submenu, which displays the *Run Parameters* form on the screen as shown in figure 6.3. The simulation may also be launched from the SIMWORK or TEXTWORK programs. The *Run Parameters* form allows the user to customize the way in which the simulation is executed and the information it provides. However, the only simulation *run parameter* that must be set is the *Run Length*, in which case the simulation will produce a *progress graph*, a simulation *plot file*, and a simulation *report file* that contains every applicable report.

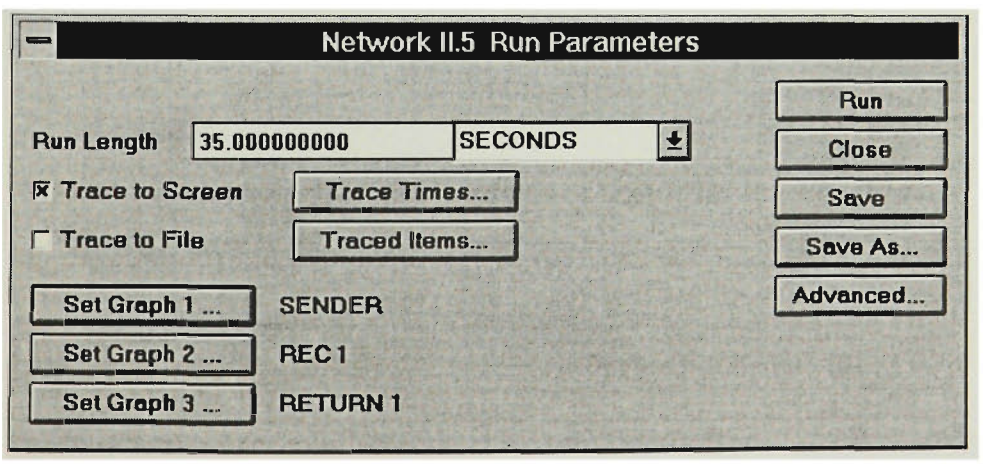


Figure 6.3. *Simulation Run Parameters form.*

The additional options may be used to provide graphs of particular hardware components, and to *trace* selected hardware and software components of the model. The *trace* feature can be set for the entire simulation or a specified period. The *Advanced* button on the *Run Parameters* form may be used to set more specific parameters for the plots and reports, and to direct the simulation *runtime warnings* to the screen, to a file or to both.

When the simulation concludes, a *simulation complete* message form is displayed. The message form states the time taken to run the simulation and the number of *runtime warnings* that were encountered. The *runtime utilisation graphs* and the final *summary report* can be viewed at this time, or they can be viewed later with all of the other reports, plots and graphs.

After a simulation has been executed, the model can be animated. The *animation* is started by selecting *Analysis*, *Animation* from the menubar. An *animation* may be set up to run continuously or step-by-step, and *trace messages* may be displayed concurrently.

6.3 Problem Formulation

The RSBT protocol, which was described in Chapter 3, was simulated so that its performance may be evaluated. RSBT was designed with four options for the acknowledgment process (see section 3.5.2):

1. Acknowledge All Frames
2. Acknowledge Last Frame Only
3. Acknowledge First and Last Frames
4. Acknowledge Lapse in Transmission

It was concluded that although Acknowledge All Frames is the most efficient acknowledgment method, it is not suitable form most implementations because of the amount of return traffic it generates. Therefore, the simulations were focussed on the other three methods of acknowledgment. Specifically, the

simulation model is based on the simplest method: Acknowledge Last Frame Only. However, because of the assumption that were made (outlined below), the simulation model is also valid for the Acknowledge First and Last Frames and Acknowledge Lapse in Transmission methods. Acknowledge First and Last Frames and Acknowledge Lapse in Transmission methods are slightly more efficient than Acknowledge Last Frame Only in some scenarios. That is, when a receiver loses frames in the tail of the transmission, and when a receiver loses its communication session and does not recover before the other receivers have successfully received all of the frames in the file. These efficiency gains are minor and can be calculated manually as shown in sections 3.5.2.3 and 3.5.2.4.

The assumptions that were made in the simulation model are the same as those described in section 5.2 for the mathematical model, which are:

- the error rate is the same for all receivers,
- receivers do not lose their communication sessions,
- acknowledgements are never lost, and
- frame headers are never lost.

As with the mathematical model, the assumptions listed above were made to simplify the simulation model and to keep the model to a manageable size. These assumptions were also required by the simulation model to accommodate the Network II.5 limitations. Network II.5 is predominantly hardware oriented and capable of modelling complex hardware configurations. However, the Network II.5 package is somewhat limited and inflexible when it comes to software modelling. In fact, it would be very difficult, if not impossible, to model the subtle differences between the three selective acknowledgment methods mentioned above. Therefore, the model neglected the protocol's processing requirements that do not influence the transmission time, and whose performance measurements are not required. The simplified processing requirements are outlined in section 6.6, where the Network II.5 model is described in detail.

The objectives of the Network II.5 modelling and simulation of the RSBT protocol are as follows:

- to derive the optimal frame sizes, and the corresponding effective throughput details, for various transmission error rate and network size (number of receivers) combinations;
- to obtain the utilisation rates of processes at the sending site and at the receiving sites;
- to obtain satellite link utilisation rate;
- to determine the total return traffic volume, or rate, produced by receivers transmitting acknowledgements; and
- to reveal possible bottlenecks that either the sender or the receivers may encounter under certain conditions.

6.4 Model Building

The Network II.5 simulation model was based on the SDL (Specification and Description Language) specifications of the RSBT protocol. The SDL specifications were developed with the XMelba CASE tool and were based on acknowledgment of the last frame only with three receivers on the network. SDL is an FDT (Formal Description Technique). An introduction to FDTs, an overview of SDL and the high level SDL specifications of the RSBT protocol are given in Chapter 4; the complete specifications of the RSBT protocol are given in Appendix A. Succinctly, the SDL model of RSBT was developed at three levels of abstraction:

1. a *system diagram* that shows the RSBT protocol as a black box, an SDL *block*, interacting with its environment;
2. a *block diagram* that shows all of the *processes* that make up the protocol; and
3. the *process diagrams*, which shows the details of each of the protocol's processes.

6.5 Data Collection

This section provides all the date that is required to build the Network II.5 simulation model of the RSBT protocol. The required data are as follows:

- speed of satellite link: **2.048 Mbps**—based on the Omnicast Digital service.
- speed of terrestrial return links: **64 Kbps**—based on the links currently used by the Bureau of Meteorology.
- processing speed of each satellite interface: **100 MIPS**—for the Network II.5 modelling purposes, it is assumed that each instruction requires one clock cycle, and thus, each satellite interface has a cycle time of **10 nanoseconds**.
- file size to be transmitted: **5 MB**.
- frame header size: **128 bits**—for the simulation model the frame header size was rounded up to the next byte; for a 5 MB file transfer the frame header should contain at most 125 bits.
- acknowledgment frame size: **80 bits**—based on acknowledgments being enclosed within HDLC control frames.
- satellite propagation delay: **0.25 seconds**.
- processing time estimations:
  - the sender's processing time for preparing the first frame for transmission: **3000 cycles**.
  - the senders acknowledgment processing time: **3000 cycles**.
  - the receivers' frame processing time: **100 cycles**.
  - the receivers' processing time for storing frames that are not corrupted, acknowledging (ACK/NAK) a frame or disregarding a frame because it had been received successfully before: **100 cycles**.

6.6 Model Translation

RSBT has already been modeled with XMelba (SDL) to formally specify the protocol and to verify the protocol's procedures. Therefore, the SDL model, which was described in Chapter 4, was used as a basis for the simulation model. The SDL model does not include any hardware devices and so the simulation model's hardware components were based on the system configuration shown in figure 3.1. The software elements of the simulation model were mapped from the SDL model as closely as possible to add credence to the simulation model. However, the SDL *process* specifications could not be mapped directly to the Network II.5 model in detail, and some simplifications were required.

SDL and Network II.5 are based on entirely different metaphors. SDL uses the metaphor of a programming language, and it is concerned mainly with software elements (variables, signals, etc.). Network II.5 on the other hand, is predominantly hardware based (network nodes, transfer devices, storage devices, *gateways*, etc.). In fact, semaphores are the only variables available in Network II.5. One example that highlights the Network II.5 inability to model detailed SDL specifications is the numbering of frames during transmission. SDL simply numbers each frame in the sequence; a frame is declared as a structure, and one of the structure's variables stores the frame sequence number. With Network II.5 on the other hand, a separate message must be created for each frame transmitted. In addition, receivers require a separate *module* for each frame that it is expected to receive, if they are required to acknowledge each frame individually. This could be done for very small files, but for files involving more than 100 or so frames, it would be too time consuming and the system resources would soon be exhausted.

Due to the incompatibility problems between SDL and Network II.5, the SDL model was mapped across to Network II.5 at the *block diagram* level, which shows all of the processes that are required, but not their details. Although most of the processing is done concurrently with the transmission of frames and is therefore insignificant, time estimations for some of the processing requirements had to be made.

Sections 6.6.1, 6.6.2 and 6.6.3 describe the entire Network II.5 simulation model development process and how it relates to the SDL model. The description of the simulation model, like the SDL model, was based on a network with three receivers. However, the Network II.5 model was modified to simulate broadcast transmissions to six and 12 receivers as well.

6.6.1 System Architecture

The first step in developing a simulation model with Network II.5 was to create the system's hardware components: *PEs*, *SDs* and *gateways*. Then, appropriate communication channels, *TDs* and *LANs*, were created to link the hardware components. The simulation model's hardware configuration, which is shown in figure 6.4, was derived directly from the system configuration illustrated in figure 3.1. However, the FTP transmissions over the LANs at both the sending and receiving sites, were not in the simulation model's scope. Therefore, the simulation model does not include the LANs and the additional workstations.

The modelling process began with the creation of all *PEs*: **SENDER**, **SATELLITE**, **REC 1** (Receiver 1), **REC 2** and **REC 3**. The **SENDER** *PE* represents the node that executes the software *modules* which perform the same functions as the **TRANSMIT** and **PROC_ACK** (process acknowledgment) *processes* in the SDL model's **BLOCK BROADCAST**. The **REC 1**, **REC 2** and **REC 3** *PEs* represent the receivers and execute software *modules* which have the same purpose as the **REC1**, **REC2** and **REC3** *processes* in the SDL model's **BLOCK BROADCAST**. The **SATELLITE** *PE* was added to the Network II.5 model to simulate the satellite propagation delay. Neither the satellite nor the satellite propagation delay was included in the SDL model, because SDL does not measure transmission time or any aspect of a system's performance. Thus far, all but one of the SDL model's *processes* have been allocated a *PE* on which their purpose is to be served. The SDL *process* that has not been catered for is the **INITIALISE** *process*. The **INITIALISE** *process* was not included in the simulation model because it is used by XMelba, or SDL, only to trigger the system at startup, and is therefore not required by the Network II.5 Model.

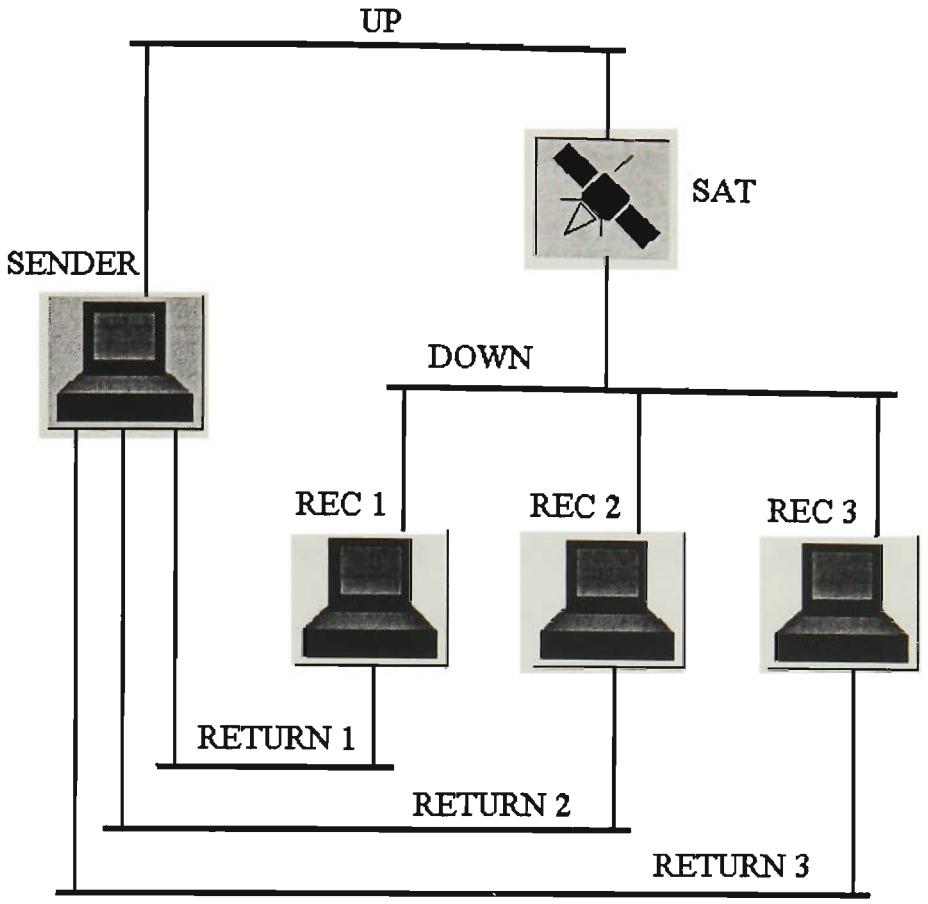


Figure 6.4. *The RSBT system architecture for the simulation model.*

The communication channels (*TDs*) in the simulation model are not quite the same as the communication links in the SDL model. The simulation model's **UP** and **DOWN** *TDs* were not used in the SDL model because SDL can not model a broadcast channel. Hence, the SDL model was developed with three unicast channels which link the **TRANSMIT** *process* to each receiver. The return channels: **RETURN 1**, **RETURN 2** and **RETURN 3**, provide the same links as the SDL model's **R1PA**, **R2PA** and **R3PA** communication *channels*.

The simulation model's hardware configuration was completed by connecting each *PE* to the appropriate *TDs*. The **SENDER** *PE* was connected to the **UP**, **RETURN 1**, **RETURN 2** and **RETURN 3** *TDs*. The **SAT** *PE* was connected to the **UP** and **DOWN** *TDs*. While the receivers—**REC 1**, **REC 2** and **REC 3** *PEs*—were connected to the **RETURN 1**, **RETURN 2** and **RETURN 3** *TDs* respectively; each receiver was also connected to the **DOWN** *TD*.

6.6.2 Hardware Specification

The hardware specifications provide the details of each component of the simulation model, which is shown in figure 6.4. Note that these details do not represent any of the details in the SDL model. This is because SDL is not concerned with time or performance measurement, and it does not model hardware components.

6.6.2.1 The Sender

The **SENDER** *PE* represents the sending satellite interface, which is responsible for broadcasting data reliably to all receivers across the satellite link (**SAT** *PE*). As shown in figure 6.5, the characteristics of the **SENDER** *PE* are as follows:

- *Cycle Time*: **10 nanoseconds**, which is based on a computer that executes 100 MIPS and the assumption that each instruction takes one clock cycle to execute—the *cycle time* is the reciprocal of the processing speed.
- *Message List Size*: **1 Mb**.
- *Queue Flag*: **Yes**.
- *Input Controller*: **Yes**.
- *Instruction List*:
  - **FRAME C1**—a *message instruction* used to represent each frame in the first transmission. As shown in figure 6.6, **FRAME C1** has the following characteristics:
    - *Length*: **1024 bits**, which represents the frame length, including the header, and is varied to simulate a range of frame sizes.
    - *Message Text*: **NR** (no response), which defaults to the *message* name—the receiver of the *message* gets the *message Text*, not the *message* name.
    - *Destination*: **SAT** (satellite).

- *Queue Flag*: **Yes**, however, the destination *PE's Queue Flag* setting takes precedence.
- *Allowed TD*: **UP**, which means that this message can only be sent across the **UP** *TD*.
- **FRAME C2**—a *message instruction* that is used to represent each frame in the second transmission (first retransmission), otherwise it is the same as **FRAME C1**.
- **FRAME C3**—a *message instruction* that is used to represent each frame in the third transmission, otherwise it is the same as **FRAME C1**.
- **FRAME C4**—a *message instruction* that is used to represent each frame in the fourth and all following transmissions, otherwise it is the same as **FRAME C1**.
- **INITIALISE**: **3000 cycles**, a *processing instruction* that is used to represent the processing time that is required to prepare the first frame for transmission (see figure 6.7).
- **PROCESSING**: **2 cycles**, which is a token *processing instruction* that is used so that *modules* which do not execute any real instructions have at least one *instruction* in their *instruction list*, e.g. **PROCESS NAK 1**.
- **RUN SET**—as shown in figure 6.8, this is a *semaphore instruction* that is used to *set* the *semaphore* **SEM RUN**, which is used as a timer, when a simulation starts.
- **SENT +** —a *semaphore instruction* that is used to *increment* the *semaphore* **SEM RUN** by an amount equal to the number of receivers each time the **SENDER** *PE* transmits a frame.
- **NAK COUNT -** —a *semaphore instruction* that is used to decrement the *semaphore* **SEM NAK COUNT** each time the **SENDER** *PE* receives a NAK.
- **RUN RESET**—a *semaphore instruction* that is used to *reset* the *semaphore* **SEM RUN** when a simulation should stop.

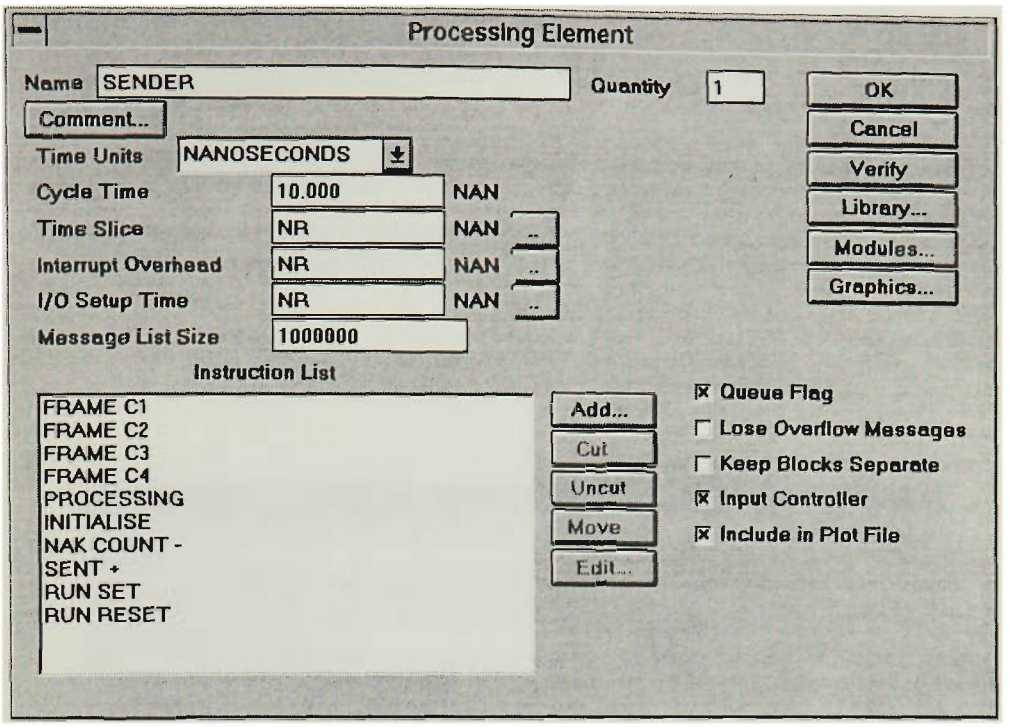


Figure 6.5. *SENDER PE details.*

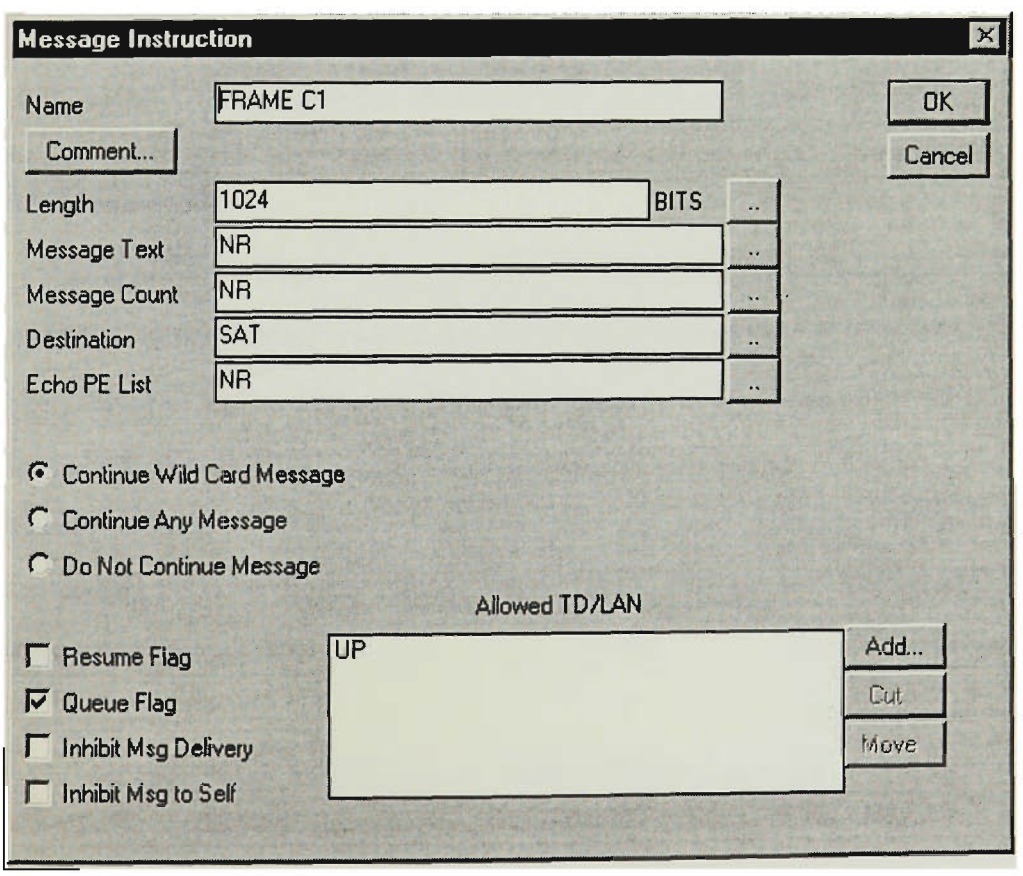


Figure 6.6. *FRAME C1 instruction details.*

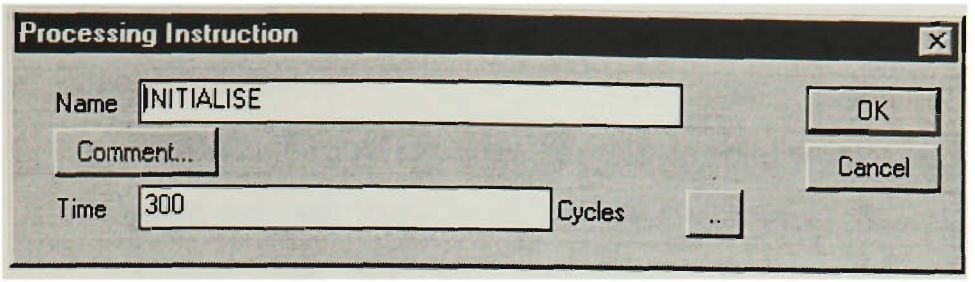


Figure 6.7. *INITIALISE instruction details.*

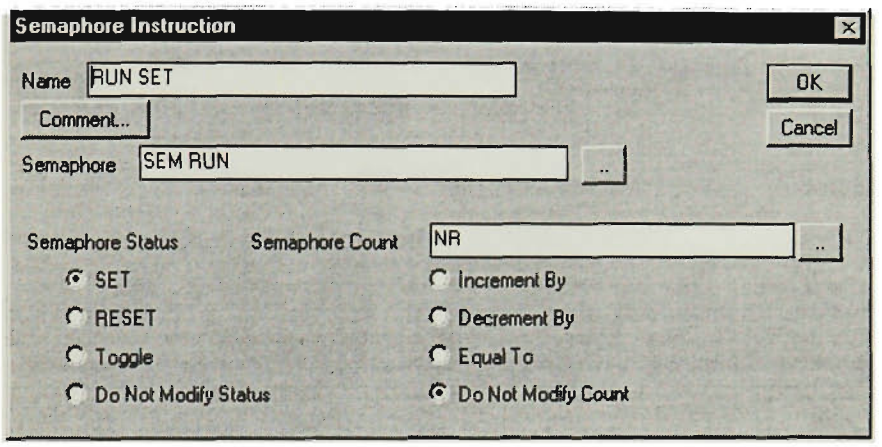


Figure 6.8. *RUN SET instruction details.*

6.6.2.2 The Satellite

The **SAT** *PE*, which represents the satellite, was included in the simulation model for no other reason than to simulate the propagation delay associated with satellite transmission. The *modules* that reside on **SAT** *PE* wait for *messages* (frames) that are sent by the **SENDER** *PE*. When a *message* arrives, the appropriate *module* simply relays (broadcasts) it to the receivers after a delay of 0.25 seconds (the satellite propagation delay). Thus, the **SAT** *PE* has the following characteristics:

- *Cycle Time*: **1,000 nanoseconds**, which is actually irrelevant because the **SAT** *PE* does not invoke any *modules* that execute *processing instructions*. The *modules* that reside on this *PE* merely delay frames for 0.25 seconds before broadcasting them to the receivers.
- *Message List Size*: **1 Mb**.
- *Queue Flag*: **Yes**.

- *Input Controller*: **Yes**.
- *Instruction List*:
  - **RELAY C1**, which is a *message instruction* that is used to represent each frame in the first transmission (same as **FRAME C1**). The **RELAY C1** *instruction* has the following characteristics:
    - *Length*: **768 bits**.
    - *Message Text*: **FRAME C1**.
    - *Destination*: **REC*** (wild card addressing).
    - *Queue Flag*: **Yes**.
    - *Allowed TD*: **DOWN**, which means that this message can only be sent across the **DOWN** *TD*.
  - **RELAY C2**, which is a *message instruction* that is used to represent each frame in the second transmission, otherwise it is the same as **RELAY C1**.
  - **RELAY C3**, which is a *message instruction* that is used to represent each frame in the third transmission, otherwise it is the same as **RELAY C1**.
  - **RELAY C4**, which is a *message instruction* that is used to represent each frame in the fourth and all following transmissions, otherwise it is the same as **RELAY C1**.

6.6.2.3 The Receivers

The **REC 1**, **REC 2** and **REC 3** *PEs* represent the receiving satellite interfaces; simulation models with six and twelve receivers were also developed. The three receivers are identical in every aspect and have the following characteristics:

- *Quantity*: **1**, which means that each receiver is independent even though they are all identical in terms of their characteristics and functionality.
- *Cycle Time*: **10 nanoseconds** (same as the **SENDER** *PE*).

- *Message List Size*: **1 Mb**, which is a size large enough to ensure that there are no overflow losses. In the real world, receivers will use multiple buffers, and thus, they will be able to receive a continuous stream of data.
- *Queue Flag*: **Yes**.
- *Input Controller*: **Yes**.
- *Instruction List*:
  - **NAK 1**—a *message instruction* that is used to represent each NAK that is sent for a **FRAME C1** *message* (frame in the first transmission). The **NAK 1** *instruction* has the following characteristics:
    - *Length*: **80 bits**, which represents the size of an acknowledgment transmitted to the **SENDER** *PE*; RSBT acknowledgements should be enclosed within a HDLC frame, and thus, the *message length* is larger than what was indicated in figure 3.3.
    - *Message Text*: **NR**, which means that no *text* has been provided, and so the **SENDER** *PE* will receive the *message* name by default.
    - *Destination*: **SENDER**.
    - *Queue Flag*: **Yes**, which means that the sender should put the *message* into a *queue* if it is busy. However, as mentioned in section 6.6.2, the destination *PE's Queue Flag* setting takes precedence.
    - *Allowed TD*: **RETURN 1**, which means that this message can only be sent across the **RETURN 1** *TD*; **REC 2** and **REC 3** will have the *allowed TD* set to **RETURN 2** and **RETURN 3** respectively.
  - **NAK 2**—a *message instruction* that is used to represent each NAK that is sent for a **FRAME C2** *message*, otherwise this *instruction* is the same as **NAK 1**.

- **NAK 3**—a *message instruction* that is used to represent each NAK that is sent for **FRAME C3** and **FRAME C4** *messages*, otherwise this *instruction* is the same as **NAK 1**.
- **ACK**—a *message instruction* that represents the positive acknowledgment that is transmitted to the **SENDER** *PE* after all messages have been received successfully, otherwise this *instruction* is identical to the NAK *messages*.
- **PROCESSING**: **100 cycles**, which is a *processing instruction* that is used to represent the processing time for each incoming message and for discarding each frame that is not required.
- **ACCEPT**—a *processing instruction* that is used to represents the processing that is done when a required frame is accepted (not corrupted).
- **NAK COUNT +** —a *semaphore instruction* that is used to *increment* the **SEM NAK COUNT** *semaphore* by one, each time a NAK is sent.
- **SENT –** —a *semaphore instruction* that is used to decrement the **SEM SENT** *semaphore* by one, each time a *message* is received.

6.6.2.4 The Satellite Uplink

The **SENDER** *PE* uses the **UP** *TD*, which represents the Omnicast Digital satellite service uplink, to transmit frames to the **SAT** *PE* (satellite). The **UP** *TD*, as shown in figure 6.9, has the following characteristics:

- *Cycle Time*: **488 nanoseconds**, which is the reciprocal of the satellite channel speed, 2.048 Mbps.
- *Bits Per Cycle*: **1**.
- *Cycles Per Word*: **8**.
- *Words Per Block*: **96**, which is altered for each simulation to match the frame size being used.
- *Connection List*: **SENDER**, **SAT**, which means that the **SENDER** and **SAT** *PEs* are connected to the **UP** *TD*.

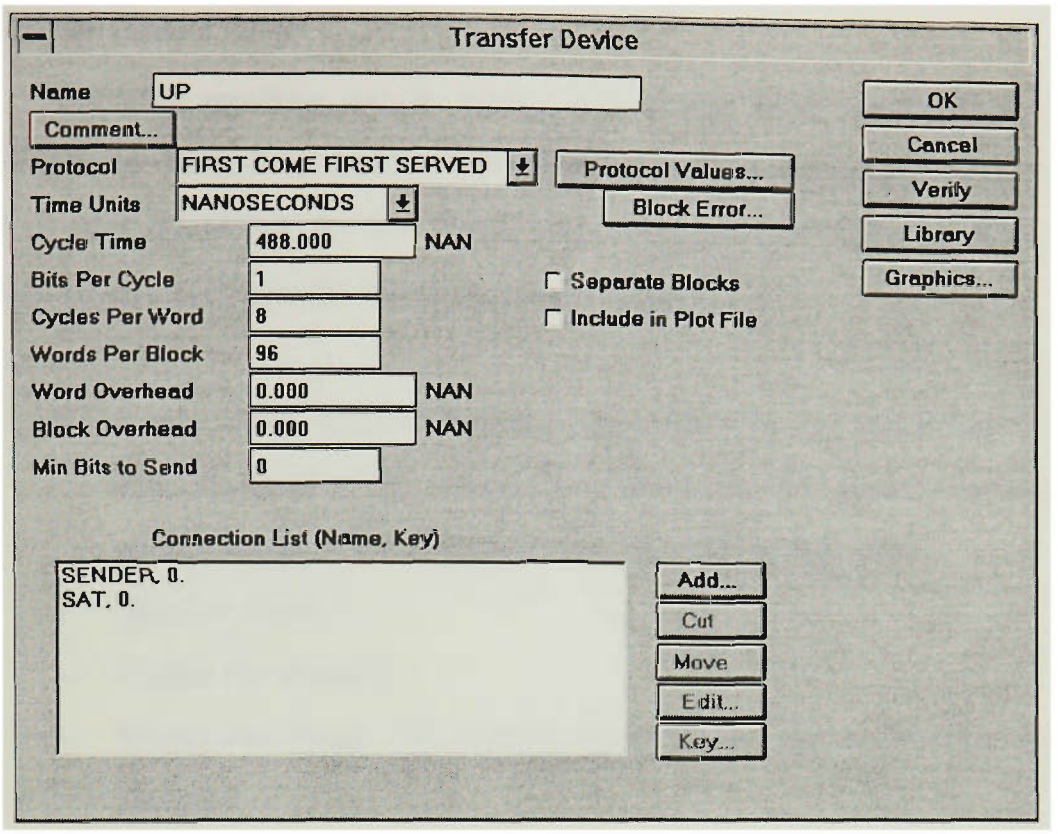


Figure 6.9. *UP TD details.*

6.6.2.5 The Satellite Downlink

The **DOWN** *TD* represents the downlink of the Omnicast Digital satellite service. The **SAT** *PE*, as shown in figure 6.4, uses the **DOWN** *TD* to relay (broadcast) all of the *messages* that are received from the **SENDER** *PE* to the receivers: **REC 1**, **REC 2** and **REC 3**. The **DOWN** *TD* is identical to the **UP** *TD* with two exceptions. The first difference is that the **DOWN** *TD* connects the **SAT** *PE* to the three receivers instead of the **SENDER** *PE*. The second difference is that the *cycle time* was set much lower: **0.001 nanoseconds** instead of the **488 nanoseconds** used on the **UP** *TD*. The *cycle time* was set at a much shorter time because the transmission serialisation delay is simulated by the **UP** *TD*, and the satellite propagation delay is simulated by the **SAT** *PE*. Therefore, the delivery of frames from the **SAT** *PE* to the receivers should be instantaneous, and any additional serialisation delay would create a significant error in the simulation results.

6.6.2.6 The Return Links

The three receivers use the return links to send acknowledgments to the **SENDER** *PE*; **REC 1** *PE* uses the **RETURN 1** *TD*, **REC 2** uses **RETURN 2** and **REC 3** uses **RETURN 3**. The three return links are identical, except for the *connection list*, and have the following characteristics:

- *Cycle Time*: **15258 nanoseconds**, which is the reciprocal of the channel speed of the terrestrial return links. The return channels, as shown in figure 3.1, are assumed to 64 Kbps HDLC links.
- *Bits Per Cycle*: **1**.
- *Cycles Per Word*: **8**.
- *Words Per Block*: **10**, which is based on an acknowledgment size of 80 bits.
- *Connection List*:
  - **RETURN 1: REC 1, SENDER.**
  - **RETURN 2: REC 2, SENDER.**
  - **RETURN 3: REC 3, SENDER.**

6.6.3 Software Specification

As mentioned in section 6.6, the low-level details that are provided by the SDL *process* specifications could not be mapped to the simulation model because Network II.5 does not facilitate detailed processes description. Therefore, the time (*cycles*) that was required for the *processing instructions* had to be estimated. However, only a few estimations had to be made because the processing requirements that do not affect the simulation time were excluded from the simulation model altogether. The inaccuracy in the time estimations does not impact significantly on the simulation time. This is because the RSBT protocol transmits all frames without delay and frames that require retransmission are placed at the back of the queue. Therefore, any inaccuracy affects only the transmission of the first frame and a few frames in the tail of the transmission. The following sections provide the details of the *semaphores* and *modules* that were used in the simulation model.

6.6.3.1 Semaphores

The simulation model uses three *semaphores*: **SEM RUN**, **SEM SENT** and **SEM NAK COUNT**. These *semaphores* are not part of the RSBT protocol description. The **SEM RUN** *semaphore* is used to measure the simulation time. While the **SEM SENT** and **SEM NAK COUNT** *semaphores* are used to keep count of the number of frames that are in the stream and the number of NAKs that the sender has not responded to.

Semaphore Run

The **SEM RUN** *semaphore* does not influence the simulation in any way. Its function is merely to measure the simulation run time (transmission time); the *semaphore* is **Set** as soon as the simulation is launched, and it is **Reset** when the **SENDER** *PE* has received an **ACK** *message* from each receiver. As shown in figure 6.10, the **SEM RUN** *semaphore* has an *initial semaphore status* of **Reset**, and the *measure response time* check box is set to **Yes**, which means that the time between *setting* and *resetting* the *semaphore* is measured. The remaining check boxes are used to select what is and what is not wanted in the *plot file*. The *initial semaphore count* and the *maximum pending responses* are irrelevant for the **SEM RUN** *semaphore*.

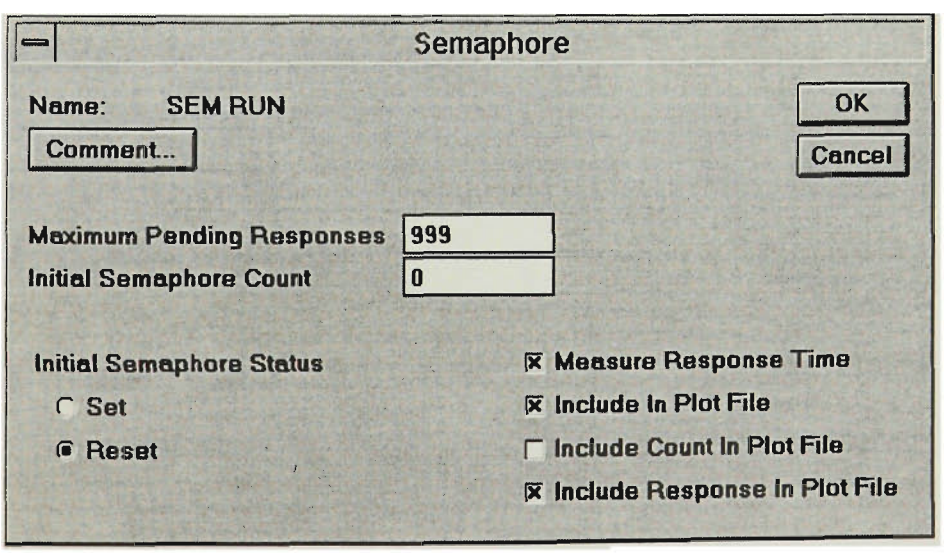


Figure 6.10. *Initial specifications for the* ***SEM RUN*** *semaphore.*

Semaphore Sent

The **SEM SENT** *semaphore* is used to keep track of the number of frames that have been transmitted by the sender but not yet received by the receivers. The **SEM SENT** *semaphore* has an *initial semaphore count* value of **zero**; all other *semaphore* details are irrelevant. The **SENDER** *PE* increments the *semaphore's counter* by the number of receivers in the model each time it transmits a frame. While receivers *decrement* the *semaphore's counter* by one for each frame that is received. The **ACK** *message* that is sent to the sender as an indication that the transmission is complete, is not sent until the **SENDER** *PE* has responded to all NAKs. That is, when the **SEM SENT** and **SEM NAK COUNT** (described below) *semaphores* both have counter values of **zero**.

Semaphore NAK Count

The **SEM NAK COUNT** *semaphore* is much the same as the **SEM SENT** *semaphore*, except that it is used to keep count of the number of NAKs that the sender has not responded to. The **SEM NAK COUNT** *semaphore* too has an *initial semaphore count* value of **zero**; again, all other *semaphore* details are irrelevant. Receivers increment the *semaphore's counter* by one each time a NAK *message* (**NAK 1, NAK 2 or NAK 3**) is sent. While the **SENDER** *PE decrements* the *semaphore's counter* by one each time it receives a NAK.

6.6.3.2 Modules Executed by the Sender

The **TRANSMISSION 1** *module* is used to transmit all of the frames in the file once (contiguously). The relevant **TRANSMISSION 1** details, as shown in figures 6.11 and 6.12, are as follows:

- *Module Controls*:
  - *Priority*: **zero**, which is the highest priority; all *modules* are set with a priority of **zero**, and thus, all *modules* are invoked in a First-Come-First-Serve manner (*priority* is not mentioned for the remaining *modules*).
  - *Concurrent Limit*: **1**, which means that only one **TRANSMISSION 1** *module* can exist at any one time.

- *Delay*: **NR (zero)**, which means that the *module* will request execution on the host *PE* as soon as its *preconditions* are met.

- *Predecessors*: **No Predecessors** (calling *modules*).
- *Preconditions*: **START 0**, which means that the *module* is executed as soon as the simulation starts.
- *Host Processing Element*: **PE: SENDER**.
- *Instruction List (Name, Execution Count)*:
  - **RUN SET, 1**, which means that the **RUN SET** *instruction* is executed once (for the *instruction* details see section 6.6.2).
  - **INITIALISE, 1**.
  - **SENT +, 64000** (for a specific simulation).
  - **FRAME C1, 64000** (for a specific simulation).
- *Successors*: **No Successors** (*successor modules*).

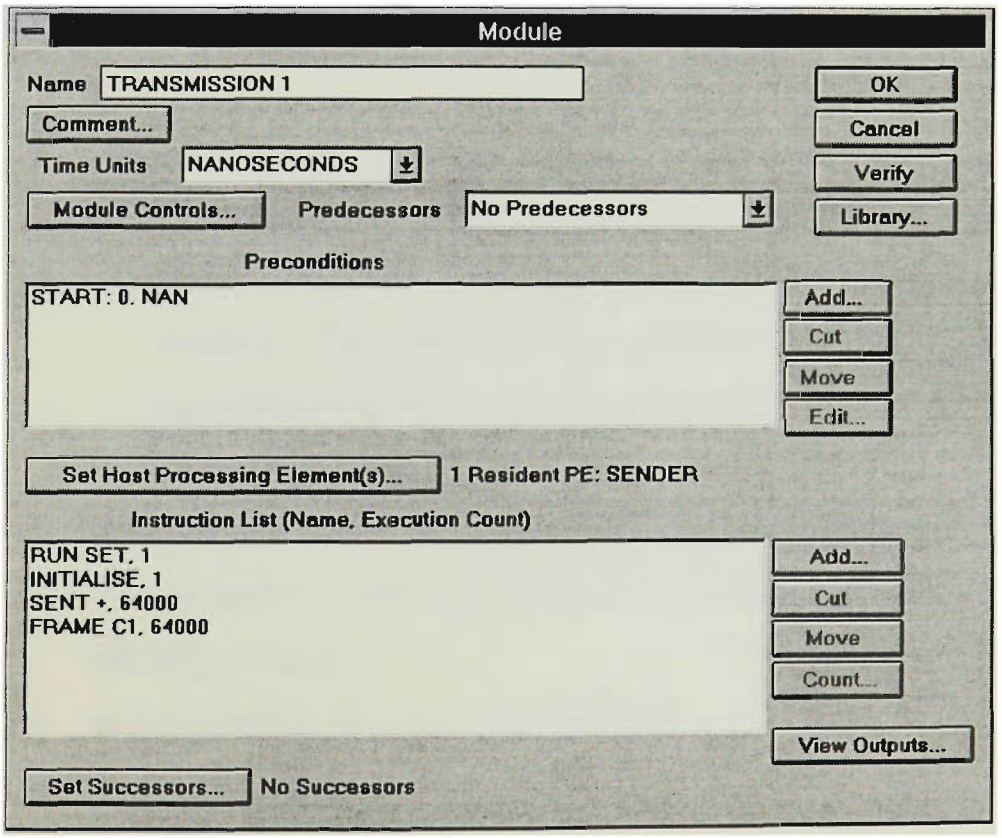


Figure 6.11. ***TRANSMISSION 1*** *module details.*

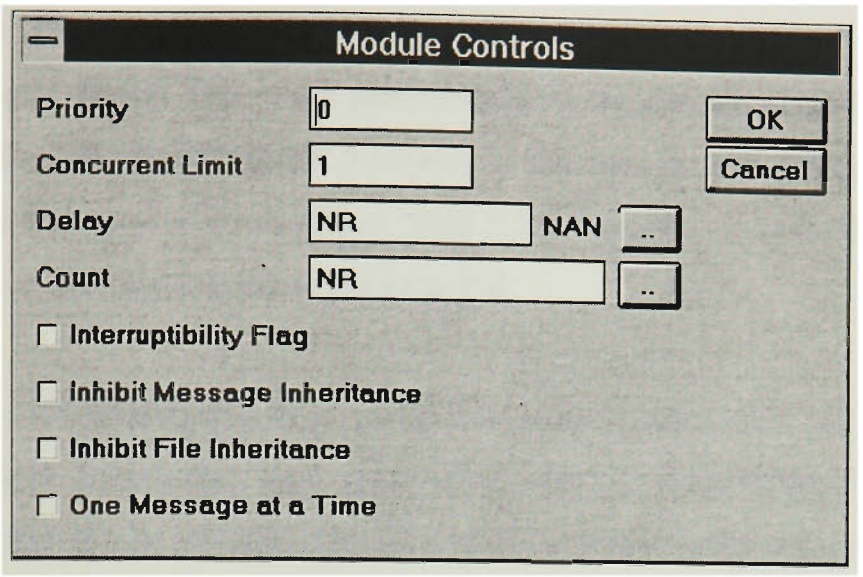


Figure 6.12. *TRANSMISSION 1 module controls.*

The **TRANSMISSION 2** *module* is used to retransmit all of the **TRANSMISSION 1** frames that were negatively acknowledged. The relevant **TRANSMISSION 2** details are as follows:

- *Module Controls*:
  - *Concurrent Limit*: **32000** (max). Frames that need retransmission are put at the back of the queue, thus many **TRANSMISSION 2** *modules*. Therefore, as with all *modules* that may be invoked many times simultaneously, the *concurrent limit* is set at 32,000.
  - *Delay*: **NR (zero)**.
- *Predecessors*: **Ored Predecessors—PROCESS NAK 1** *module*.
- *Preconditions*: **None**, the *module* is invoked by the *module* **PROCESS NAK 1**.
- *Host Processing Element*: **PE: SENDER**.
- *Instruction List*:
  - **FRAME C2, 1**.
  - **SENT +, 1**.
  - **NAK COUNT -, 64000** (for a specific simulation).
- *Successors*: **No Successors**.

The **TRANSMISSION 3** *module* is used to retransmit all negatively acknowledged frames that were previously sent by the **TRANSMISSION 2** *module*. The **TRANSMISSION 3** *module* is the same as **TRANSMISSION 2**, except that it is invoked by the **PROCESS NAK 2** *module* and the **FRAME C3** *instruction* is executed instead of **FRAME C2**.

The **TRANSMISSION 4** *module* is used to retransmit all negatively acknowledged frames that were previously sent by **TRANSMISSION 3** or **TRANSMISSION 4**. Statistically, due to the selected parameters, each frame that was sent by the **TRANSMISSION 4** *module* was needed by only receiver (see mathematical model results in Appendix B). Thus, **TRANSMISSION 4** was used for all further retransmissions. The **TRANSMISSION 4** *module* is basically the same as the **TRANSMISSION 2** and **TRANSMISSION 3** *modules*, except that it is invoked by the **PROCESS NAK 3** *module* and the **FRAME C4** *instruction* is executed instead of the **FRAME C2** or **FRAME C3** *instructions*.

The **PROCESS NAK 1** *module* is used to process NAKs received for frames transmitted by the **TRANSMISSION 1** *module*. The relevant **PROCESS NAK 1** details are as follows:

- *Module Controls*:
  - *Concurrent Limit*: **32000**.
  - *Delay*: **3000 cycles**, which represents the processing done by the sender upon the arrival of a NAK. The processing time is not modeled by a *processing instruction* because the processing may be done while another *module* is executing a *message instruction* (transmitting) and the *module* can not begin execution while another *module* is still in progress.
- *Predecessors*: **None**.
- *Preconditions*: **M: WF: NAK 1**, which means that the *module* waits for (**WF**) a *message* (**M**) called **NAK 1**.
- *Host Processing Element*: **PE: SENDER**.

- *Instruction List*: **PROCESSING**, which is only a token *processing instruction* (**2 cycles**) because the processing is modeled by the *module delay.*
- *Successor Modules*: **2 Statistical Successors**, which means that one of the two *successor modules* is chosen statistically. Each *module*, as shown in figure 6.13, has an associated percentage value that represents its probability of being selected as the *successor*. The *successor module* is chosen statistically because Network II.5 does not recognise frame sequence numbers, and thus, duplicate NAKs must be removed statistically. The portion of NAKs that trigger a retransmission and the portion that is disregarded were derived from results provided by equations (5.4) and (5.1). These equations provide the expected values for the number of frames in the second transmission and the number of NAKs received for frames sent in the first transmission respectively. The two *successor modules*, are:
  - **TRANSMISSION 2, 94.26** (for a specific simulation), which is used to retransmit a frame; and
  - **DISREGARD NAK, 5.74** (for a specific simulation), which is used to discard a duplicate NAK.

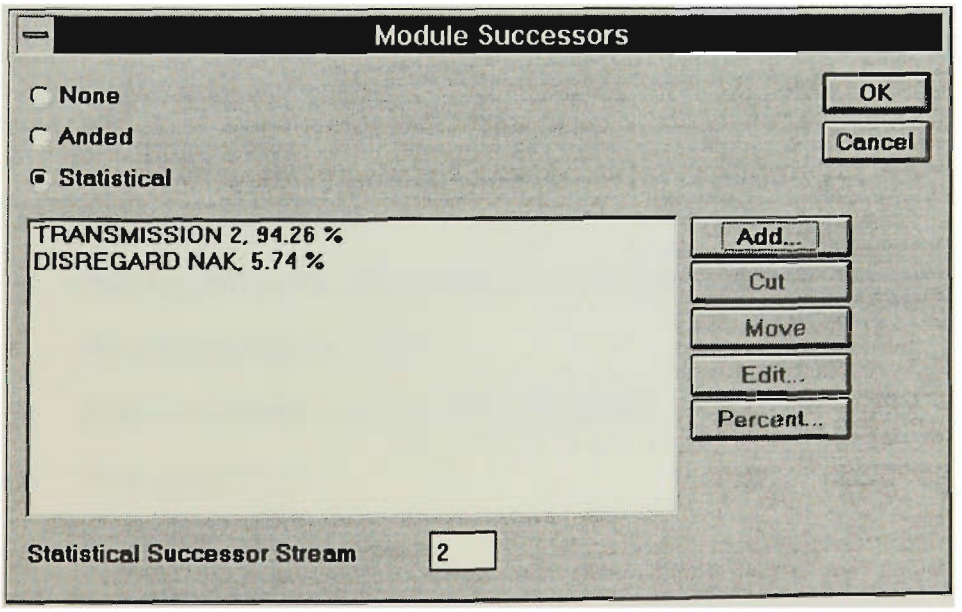


Figure 6.13. ***PROCESS NAK 1*** *Module Successors form.*

The **PROCESS NAK 2** *module* is used to process NAKs received for frames transmitted by the **TRANSMISSION 2** *module*. **PROCESS NAK 2** is the same as **PROCESS NAK 1**, except that it is invoked by the arrival of a **NAK 2** *message*, and the probabilities associated with the *statistical successors* are derived from the result provided by equations (5.8) and (5.4).

The **PROCESS NAK 3** *module* is used to process NAKs received for frames sent by the **TRANSMISSION 3** or **TRANSMISSION 4** *modules*. It was explained in the **TRANSMISSION 4** *module* description, that all **NAK 3** *messages* represent unique frames, and thus, they can also be used as NAKs for frames sent by **TRANSMISSION 4**. The **PROCESS NAK 3** *module* is the same as **PROCESS NAK 1** and **PROCESS NAK 2**, except that it is invoked by the arrival of a **NAK 3** *message*. In addition, because there is no duplication of **NAK 3** *messages*, **PROCESS NAK 3** uses only one *successor*—**TRANSMISSION 4**.

The **SIM DONE** *module* is used to stop the simulation (stop the clock) when the **SENDER** *PE* receives the **ACK** *message*, which indicates that all receivers have received all frames successfully. The relevant **SIM DONE** details are as follows:

- *Module Controls*:
  - *Concurrent Limit*: **1**.
  - *Delay*: **NR**.
- *Predecessors*: **None**.
- *Preconditions*: M**: WF: ACK**, which means that the *module* waits for the *message* called **ACK**.
- *Host Processing Element*: **PE: SENDER**.
- *Instruction List*:
  - **PROCESSING**;
  - **RUN RESET**, which *resets* the **SEM RUN** *semaphore* to record the simulation time (*semaphore response time*).
- *Successor Modules*: **None**.

6.6.3.3 Modules Executed by the Satellite

The **BROADCAST C1** *module* simulates the satellite propagation delay. **BROADCAST C1** receives frames that are sent by the **TRANSMISSION 1** *module*, then delays each frame for 0.25 seconds before broadcasting it to all receivers. The relevant **BROADCAST C1** details are as follows:

- *Module Controls*:
  - *Concurrent Limit*: **1**.
  - *Delay*: **NR**.
- *Predecessors*: **None**.
- *Preconditions*: M**: WF: FRAME C1**, which means that the *module* waits for a *message* called **FRAME C1**.
- *Host Processing Element*: **PE: SAT**.
- *Instruction List*: **RELAY C1**, which relays the **FRAME C1** *message* to all receivers.
- *Successor Modules*: **None**.

The **BROADCAST C2**, **BROADCAST C3** and **BROADCAST C4** *modules*, which also reside on the **SAT** *PE*, are basically the same as the **BROADCAST C1** *module*. The only difference is that they relay (broadcast) the **FRAME C2**, **FRAME C2** and **FRAME C2** *messages* by executing the **RELAY C2**, **RELAY C2** and **RELAY C2** *message instructions* respectively.

6.6.3.4 Modules Executed by Receivers

The **RECEIVE FRAME C1** *module* processes frames sent by the **BROADCAST C1** *module*, and determines whether the frames are corrupted. The **RECEIVE FRAME C1** details are as follows:

- *Module Controls*:
  - *Concurrent Limit*: **100**.
  - *Delay*: **NR**.
- *Predecessors*: **None**.

- *Preconditions*: M: **WF: FRAME C1**, which means that the *module* waits for a *message* called **FRAME C1**.
- *Host Processing Elements*: **3 Allowed PEs: REC 1, REC 2, REC 3,** (may execute on all receivers).
- *Instruction List*: **PROCESSING, (100 cycles)**.
- *Successor Modules*: **2 Statistical Successors**:
  - **ACCEPT FRAME, 94.15%** (for a specific simulation)—the percentage value is the probability that this *module* will be selected as the *successor*. The percentage value actually represents the frame success rate.
  - **REJECT FRAME C1, 5.85%** (for a specific simulation)—again the percentage value is the probability that the *module* will be selected as the *successor*. The percentage value actually represents the frame error rate.

The **RECEIVE FRAME C2** *module* processes frames sent by the **BROADCAST C2** *module*. Then it determines either that the frame is not required because it has been received successfully before, or that the frame is required and received intact or that the frame is required but corrupted. This *module* is the same as **RECEIVE FRAME C1** except that its *precondition* waits for the **FRAME C2** *message* and it has an additional *successor module*. The three *statistical successor modules* to the **RECEIVE FRAME C2** *module* are:

- **FRAME NOT REQUIRED, 64.64%** (for a specific simulation), the percentage value represents the proportion of frames that have been received before. The percentage value is derived from equation (5.6).
- **ACCEPT FRAME, 33.29%** (for a specific simulation), which is the same as for **RECEIVE FRAME C1**, except that the percentage value is derived from equation (5.7).
- **REJECT FRAME C2, 2.07%** (for a specific simulation), which is the same as for **RECEIVE FRAME C1**, except that the percentage value is derived from equation (5.7).

The **RECEIVE FRAME C3** and **RECEIVE FRAME C4** *modules* are the same as **RECEIVE FRAME C2** except that they are used for **FRAME C3** and **FRAME C4** *messages* respectively. In addition, these two *modules* use **REJECT FRAME C3** and **REJECT FRAME C4** as the *successor modules* instead of **REJECT FRAME C2**; the percentage values that are associated with the *statistical successor modules* are obviously different.

The **ACCEPT FRAME** *module* completes the processing that is required for frames that are required and accepted as being intact. The *module* details are as follows:

- *Module Controls*:
  - *Concurrent Limit*: **100**.
  - *Delay*: **NR**.
- *Predecessors*: **Ored**, which are **RECEIVE FRAME C1**, **RECEIVE FRAME C2**, **RECEIVE FRAME C3** and **RECEIVE FRAME C4**.
- *Preconditions*: **None**.
- *Host Processing Elements*: **3 Allowed PEs: REC 1, REC 2, REC 3.**
- *Instruction List*:
  - **ACCEPT**, which is a *processing instruction* of 100 *cycles*;
  - **SENT -**, which decrements the *semaphore* **SEM SENT**.
- *Successor Modules*: **REC DONE**, which executes only if all receivers have received all of the frames in the file successfully.

The **FRAME NOT REQUIRED** *module* does the required processing for frames that are not required by receivers because they have been received successfully before. The **FRAME NOT REQUIRED** *module* is the same as the **ACCEPT FRAME** *module* except that **RECEIVE FRAME C1** is not one of the *predecessors* and it executes the **PROCESSING** *instruction* instead of the **ACCEPT** *instruction*.

The **REJECT FRAME C1** *module* is used to negatively acknowledge frames in the first transmission (**FRAME C1** *messages*) that are deemed to be corrupted by the **RECEIVE FRAME C1** *module*. The *module* details are as follows:

- *Module Controls*:
  - *Concurrent Limit*: **100**.
  - *Delay*: **NR**.
- *Predecessors*: **Ored**, which is **RECEIVE FRAME C1**.
- *Preconditions*: **None**.
- *Host Processing Elements*: **3 Allowed PEs: REC 1, REC 2, REC 3.**
- *Instruction List*:
  - **NAK 1**, which is a *message instruction* used to negative acknowledge a frame;
  - **NAK COUNT +**, which increments the **SEM NAK COUNT** *semaphore* to keep a count of the number of outstanding retransmissions;
  - **SENT -**, which decrements the **SEM SENT** *semaphore*.
- *Successor Modules*: **None**.

The **REJECT FRAME C2**, **REJECT FRAME C3** and **REJECT FRAME C4** *modules* play much the same part as **REJECT FRAME C1**; these *modules* are used to negatively acknowledge **FRAME C2**, **FRAME C3** and **FRAME C4** *messages* respectively. Thus, **REJECT FRAME C2** has the **RECEIVE FRAME C2** *module* as the *predecessor* and executes the **NAK 2** *message instruction* (instead of **NAK 1**). **REJECT FRAME C3** has the **RECEIVE FRAME C3** *module* as the *predecessor* and executes the **NAK 3** *message instruction.* And **REJECT FRAME C4** has the **RECEIVE FRAME C4** *module* as the *predecessor*, and like **REJECT FRAME C3**, it too executes the **NAK 3** *message instruction.* **REJECT FRAME C4** uses the **NAK 3** *message instruction* because there is no duplication with **NAK 3** *messages* (each *message* negatively acknowledges a unique frame). Thus, the **NAK 3** *message instruction*

is used to negatively acknowledge frames that are part of the third and all following transmissions (**TRANSMISSION 4** *module*). Note that the **TRANSMISSION 4** *module* is used to send frames that are part of the fourth and any following transmissions.

The **REC DONE** *module* is used to notify the **SENDER** *PE* that all receivers have successfully received all of the frames in the file, and thus the transmission is over. In a real system, each receiver individually would send an ACK to the sender. However, due to the Network II.5 limitations, only one receiver sends an ACK when there are no outstanding retransmissions. The **REC DONE** *module* details are as follows:

- *Module Controls*:
  - *Concurrent Limit*: **32000**.
  - *Delay*: **NR**.
- *Predecessors*: **Ored**, which are **FRAME NOT REQUIRED** and **ACCEPT FRAME**.
- *Preconditions*: **SSR: CI: SEM NAK COUNT: COUNT = 0 and SSR: CI: SEM SENT: COUNT = 0**, which means that the **Semaphore Status Requirement** (**SSR**) permits this *module* to **Chain If** (**CI**—start) both *semaphores* (**SEM NAK COUNT** and **SEM SENT**) have *counter* values of **zero**.
- *Host Processing Elements*: **3 Allowed PEs: REC 1, REC 2, REC 3.**
- *Instruction List*:
  - **ACK**—a *message instruction* used to positively acknowledge the whole file and indicate to the **SENDER** *PE* that the transmission has been successfully completed;
  - **NAK COUNT +** —a *semaphore instruction* that increments the **SEM NAK COUNT** *semaphore* to ensure that no other receiver executes the **ACK** *message instruction.*
- *Successor Modules*: **None**.

6.7 Simulation Model Verification

The logic of the simulation model was based on the SDL model. However, as mentioned in section 6.6, the SDL processing details could not be modeled fully with the Network II.5 package. Therefore, the simulation model was verified against the SDL model in terms of the I/O operations and the decisions that *modules* make, i.e. the selection of *successor modules*. Thus, the verification was done by comparing the simulation model's structure chart, shown in figure 6.14, and the appropriate SDL diagrams which are given in Appendix A. The comparisons indicate that the simulation model behaves as it was intended to.

The Network II.5 *modules* **TRANSMISSION 1**, **TRANSMISSION 2** **TRANSMISSION 3** and **TRANSMISSION 4** perform the same function as SDL's **PROCESS TRANSMIT**. That is, they transmit frames to the receivers. The SDL model's **PROCESS TRANSMIT** uses two additional output commands and two reciprocal input commands. First, **PROCESS TRANSMIT** requests data by sending the **Give (SEQ_NUM)** *signal* (*message*) to **En** (environment), and in response, the environment sends the data for that frame in the **SendThis (DATATYPE)** *signal*. The simulation model does not use these I/O commands because it is assumed that the data is framed while another frame is being transmitted. The time that is required for preparing the first frame for transmission is simulated by the **INITIALISE** *processing instruction.* In addition, it is assumed that a buffer is used to store a reasonable number of frames. Thus, when the transmission approaches its end and retransmission requests are not delayed, it is expected that the required frames are stored in the buffer, and thus, the framing process is not required. And second, before retransmitting a frame, the SDL model's **PROCESS TRANSMIT** sends the **Next** *signal* to **PROCESS PROC_ACK**, which replies with the **Resend (SEQ_NUM)** *signal*. The simulation model does not use frame sequence numbers. Instead, the **PROCESS NAK 1**, **PROCESS NAK 2** and **PROCESS NAK 3** *modules* simply invoke the appropriate transmission *module*: **TRANSMISSION 2**, **TRANSMISSION 3** or **TRANSMISSION 4**. NAK duplicates are removed statistically.

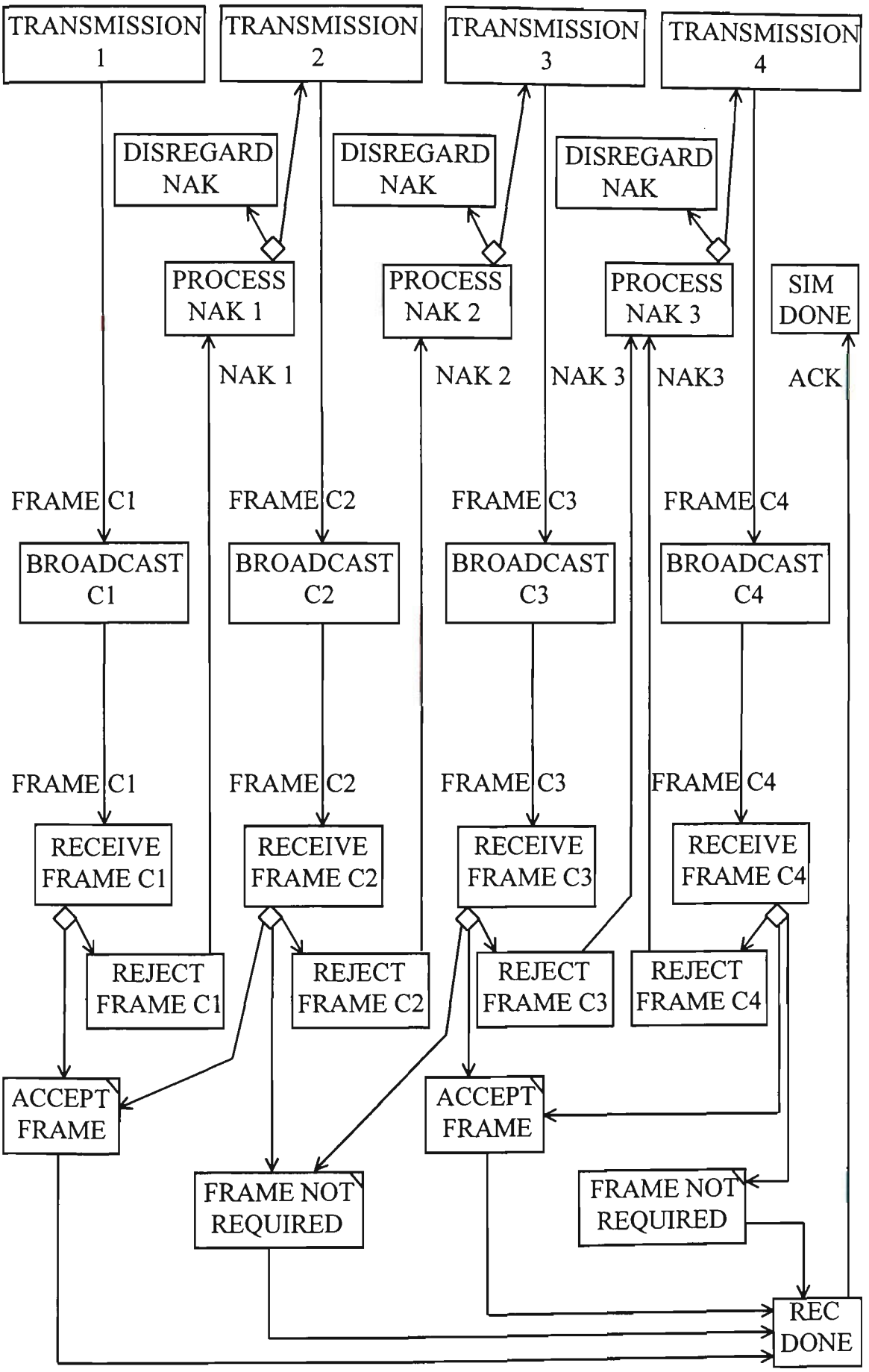


Figure 6.14. *Simulation model structure chart.*

The Network II.5 *modules* **PROCESS NAK 1**, **PROCESS NAK 2** and **PROCESS NAK 3** are the same as the SDL model's **PROCESS PROC_ACK** in that they process NAKs. The **PROCESS NAK 1** and **PROCESS NAK 2** *modules* first determine whether the received NAK is a duplicate. If the NAK is a duplicate the **DISREGARD NAK** *module* is invoked, otherwise, as shown in figure 6.14, the appropriate transmission *module* is invoked. As mentioned above, in contrast to the SDL model, the communication between the NAK processing *modules* and the transmission *modules* is not required. The only other difference between the SDL model and the simulation model in terms of the acknowledgement processing, is that the SDL model's **PROCESS PROC_ACK** also processes and counts the ACKs. When the number of ACKs tallies to the number of receivers on the network, **PROCESS PROC_ACK** sends the **Done** *signal* to **PROCESS TRANSMIT** which begins the session closing procedure. With the simulation model, only one receiver transmits the **ACK** *message*, which is processed by the **SIM DONE** *module*. The **SIM DONE** *module resets* the **SEM RUN** *semaphore*, and thus, records the simulation time (*semaphore response time*) in the simulation report.

The **BROADCAST C1**, **BROADCAST C2**, **BROADCAST C3** and **BROADCAST C4** *modules* are used only to simulate the propagation delay that is associated with satellite transmission, and to broadcast the frames to the receivers; these *modules* do not perform any *processing instructions*. The SDL model does not measure transmission time or performance, and thus, does not have any corresponding processes.

The Network II.5 receiving *modules* **RECEIVE FRAME C1**, **RECEIVE FRAME C2**, **RECEIVE FRAME C3** and **RECEIVE FRAME C4**, are collectively the same as each of the SDL model's receiver *processes* **PROCESS RECEIVER1**, **PROCESS RECEIVER2** and **PROCESS RECEIVER3**. There is a minor difference however. That is, while the SDL model's receiving *processes* control all tasks, the Network II.5 *modules* decide whether or not a frame is corrupted, and then invoke other *modules* to complete the required tasks.

The Network II.5 **RECEIVE FRAME C1** *module* is responsible for processing the frames in the first transmission. If the **RECEIVE FRAME C1** *module* considers a frame to be corrupted, it invokes the **REJECT FRAME C1** *module*. The **REJECT FRAME C1** *module*, like the SDL receiver *processes*, does some processing and then sends a **NAK 1** *message* to the **SENDER** *PE*. If **RECEIVE FRAME C1** considers a frame to be intact, it invokes the **ACCEPT FRAME** *module*. The **ACCEPT FRAME** *module* does some processing and then invokes the **REC DONE** *module*. The **REC DONE** *module* is executed only if the *semaphore status requirement* have been met – if there are no outstanding retransmissions. The **REC DONE** *module*, like the SDL receiver *processes*, sends an ACK to the sender, which indicates that the transmission has been completed. The only difference is that the SDL receivers function independently, and therefore, each receiver sends an ACK when it has received all of the frames successfully.

The **RECEIVE FRAME C2**, **RECEIVE FRAME C3** and **RECEIVE FRAME C4** *modules* have an additional function. That is, they also make a decision (statistically) as to whether a frame has been received successfully before. If this is true, then the **FRAME NOT REQUIRED** *module* is invoked. The SDL model's receiver *processes'* of course make this decision by checking the retransmission list (**RecList (SegNum)**).

There are three SDL communication *signals* for which the simulation model does not have the corresponding *message instructions*. The first two, the **Receive (Packet)** and **IdNum (Id)** *signals* (which are sent to the environment) are not included in the simulation model because they are outside the simulation model's scope; these SDL *signals* represent the FTP/TCP transmission on the LANs. The third, the **TimeOut (SeqNum)** *signal* (which triggers the retransmission of NAKs) was not used in the simulation model because it was assumed that reliable HDLC return links are used—acknowledgements are never lost. Furthermore, Network II.5 does not cater for frame sequence numbers, and thus, a time-out procedure would be extremely difficult to model.

6.8 Simulation Model Validation

To complete the simulation modelling, the model was validated. As stated in section 5.1, the mathematical model results and the simulation model results, were used to validate each other. Both models were based on the same assumptions, and therefore should produce the same, or similar, results.

The simulations corrupt frames randomly, and thus, the results can be misleading. For example, if one simulation run corrupts several frames in the tail of the transmission, while another run does not corrupt any, then the simulation times can differ significantly. Therefore, the simulation model used for validation purposes was run six times to provide a more accurate indication of the expected results. The simulation model and the mathematical model parameters shown below and the results shown in tables 6.1 and 6.2 are based on the mathematical model notation given in section 5.2. The parameters used for the validation were:

R = 3 (number of receivers).

E = 10% (frame error rate).

F_1 = 20,000 (frames in the file).

S = 1024 bits (frame size).

t_r = 0.25125264 seconds (round trip time), which comprises the satellite propagation delay (0.25), the receivers' processing time ($2 * 10^{-6}$), the receivers' serialisation delay (0.00122064) and the senders acknowledgment processing time ($3 * 10^{-5}$).

C = 2.048 Mbps (satellite channel speed).

The mathematical model results, shown in table 6.2, are comparable to the simulation model's average results (of six simulations), shown in table 6.1. Therefore, the two models are assumed to be valid. Recall from section 6.6.2.1 that simulation model's *F_4 Frames* represents the number of frames sent in the fourth and all following transmissions. The complete mathematical model results and simulation results are given in Appendix B and Appendix C respectively. These results can be compared to reinforce the validity of both models.

Table 6.1. *Simulation model validation results.*

Variables	Simulations						
	Run 1	Run 2	Run 3	Run 4	Run 5	Run 6	Average
F_1 Frames	20,000	20,000	20,000	20,000	20,000	20,000	20,000
F_2 Frames	5,404	5,386	5,511	5,392	5,440	5,472	5,434
F_3 Frames	591	599	650	545	590	592	595
F_4 Frames	64	70	81	59	76	67	70
Total Sent	26,059	26,055	26,242	25,996	26,106	26,131	26,098
N_1 NAKs	5,993	5,964	6,120	6,009	5,984	6,040	6,018
N_2 NAKs	597	606	657	550	596	598	601
N_3 NAKs	64	70	81	59	76	67	70
T time (sec.)	14.051	13.717	13.757	13.691	13.761	13.764	13.790

Table 6.2. *Mathematical model validation results.*

Variables	Results
F_1 Frames	20,000.00
F_2 Frames	5,420.00
F_3 Frames	594.02
F_4 Frames	60.00
F_5 Frames	6.00
F_6 Frames	0.60
F_7 Frames	0.06
Total Sent	26,080.68
N_1 NAKs	6,000
N_2 NAKs	600
N_3 NAKs	60
N_4 NAKs	6.00
N_5 NAKs	0.60
N_6 NAKs	0.06
T time (sec.)	13.780

6.9 Conclusion

The RSBT simulation model was developed in formal steps that ensured that the model was an accurate representation of the desired system and that the simulation objectives could be realised. Although the Network II.5 package could not model the RSBT protocol's detailed processing requirements, it was a suitable simulation package because it enables detailed hardware and I/O specification. The protocol processing that is performed during I/O operations was not included in the model. The processing that affects the simulation time was estimated from the SDL specifications. The processing procedures that were modelled, affect the transmission time only before the first frame is transmitted, and in the tail of the transmission when the sender experiences idle periods. Therefore, any inaccuracy in the time estimations is insignificant.

As mentioned above, the RSBT simulation model was developed using a step-by-step procedure. First, the simulation problem was clearly defined. Second, the SDL model, which is described in Chapter 4, was identified as the basis for the simulation model. This was appropriate because the SDL model is a formal specification of the RSBT protocol, and thus provided detailed information, which was used for both developing and verifying the simulation model. Third, all of the data that was required for building the simulation model was gathered from the SDL specifications in Appendix A, and from the RSBT description and system configuration given in Chapter 3. Fourth, the SDL model (at an abstract level) was mapped to the simulation model and complimented by the data provided by the previous step. Fifth, the simulation model was verified against the SDL model by comparing all I/O operation and the major decisions made by the protocol. The simulation model was also verified visually by stepping through a Network II.5 animation of the model. The verification process indicated that the simulation model behaves as it is intended to. And sixth, the model was validated by comparing the results of a set of simulation runs to the results obtained by the mathematical model with the same set of parameters. The results produced by the simulation model and the mathematical were very similar, and thus, attribute to the confidence in both models.

7. Experimentation and Results Analysis

7.1 Introduction

This chapter first describes the experiment plan that was devised for both the mathematical model and the simulation model. Then the simulation and mathematical model results are analysed and compared. The mathematical model results are more accurate (in terms of expected values) than the simulation results because of the random selection of events during a simulation, and also because the mathematical model facilitated much easier fine-tuning. Hence, the mathematical model results, rather than the simulation results, were used for comparison against results found in the literature—using other protocols in point-to-point satellite transmission. The raw mathematical model results and simulation results are given in Appendix B and Appendix C respectively.

The experiment plan was developed to minimise the amount of experimentation that was required to satisfy the objectives outlined in section 6.3. The experiment plan was particularly important for the simulations. This is because some of the simulation took more than 16 hours to complete, so it was not viable to simulate models with all possible parameter combinations. The mathematical model did not have the same time constraints because it was implemented in a spreadsheet, which produced new computational results almost instantly. In fact, the mathematical model results were generated for a larger set of parameter values (see Appendix B and Appendix C), and thus provided greater accuracy.

The main aim of the simulations and the mathematical model computations was to find the optimal frame sizes and the corresponding throughput rates for a set of *Network Size* and *Base Error Rate* combinations; these parameters are explained in section 7.2. An optimal frame size can greatly improve the

throughput: if frames are too small, then the overheads are unnecessarily large; and if frames are too large, then the retransmission volume is unproportionally larger because of the greater frame loss rate.

The simulations and the mathematical model computations were also used for general evaluation of the protocol, with the satellite link utilisation and return traffic being the other main points of interest. The satellite link utilisation rate has a direct effect on the throughput, and thus, a high link utilisation rate is particularly important. In many cases, the procedures used by protocols restrict the satellite link utilisation and thereby have a greater effect on the throughput then the channel error rate. For example, the throughput of the TCP protocol is restricted by the slow-start algorithm and the flow control window [24],[25]. The return traffic (acknowledgments) increases with higher error rates and larger networks. Therefore, it is important to minimise the number of acknowledgments sent by receivers, as it ultimately determines the maximum number of receivers that the sender can service.

Although the largest network that was modeled included only 12 receivers, the results may be used as a guide for larger networks, particularly in regards to the return traffic.

7.2 Experiment Planning

The experiments (simulations and mathematical model computations) were based on 15 *Network Size* and *Base Error Rate* combinations. The *Network Size*, for the purpose of this thesis, represents the number of receivers on the WAN. While the *Base Error Rate* represents the frame error rate (FER) at a basic frame size of 128 bits, which is more meaningful than a bit error rate (BER). Each *Network Size* and *Base Error Rate* combination was run with varying frame sizes until the optimal was found (shortest transmission time); all of the frame sizes used by the experiments were multiples of 128. The mathematical model required two fixed parameters, i.e. the satellite channel speed (2.048 Mbps—as with the simulation model) and the round trip time (0.2512517 sec.). The round trip time

is composed of four components: the satellite propagation delay (0.25 sec.); the receivers' frame processing time (10^{-6} sec.); the receivers' serialisation delay in transmitting an acknowledgment (1.2207×10^{-3} sec.); and the sender's acknowledgment processing time (3×10^{-5} sec.). Note that the round trip time excludes the senders serialisation delay in transmitting a frame. The four round trip time components are either listed or may be derived from the data provided in section 6.5. The main set of experiments were based on a 5 Mbyte file (rounded to the nearest full frame) and combinations of the following parameters:

- **Network sizes**: 3 receivers, 6 receivers, and 12 receivers.
- **Base error rates**: 0.01, 0.005, 0.001, 0.0005 and 0.0001.
- **Frame sizes** (in bits):
  - **Mathematical model**: all multiples of 128.
  - **Simulations**: 256, 512, 768, 1024, 1280, 1536, 2048, 2560, 3072, 3584, 4096, 5120, 6400 and 8192.

Two *Network Size* and *Base Error Rate* combinations where re-simulated and mathematically re-evaluated with a 2 Mbyte file and a 10 Mbyte file. This was done to confirm that the file size does not affect the optimal frame size and to check the performance difference. In addition, four experiments were carried out with the mathematical model using 100 Kbyte files.

The *Base Error Rate* is derived from the bit error rate (BER). Although the BER for satellite links is difficult to estimate, most BERs fall in the 10^{-4} to 10^{-8} range [25] (10^{-3} to 10^{-7} according to [23],[51]). However, bit errors are not independent; they occur in burst typically between 3 and 10 bits. Therefore, burst errors, which are independent, can be conservatively approximated at 1/3 of the BER [23],[51]. With these assumptions, the *Base Error Rate* is given by the following equation [23]:

$$\begin{aligned} &\textit{Base Error Rate} = 1-(1-p)^{k} \\ &\textit{where } k = \textit{no. of bits in the frame} \\ &\textit{and } \quad p = \textit{burst error rate } (BER/3). \end{aligned} \tag{7.1}$$

The BER range suggested by [23] and [51] is based on information and technology available in 1969-70. In contrast, the BER range suggested by [25] is based on information compiled in 1993. Therefore, the BER is assumed to be between 10^{-4} and 10^{-8}. Then, using equation (7.1), the respective *Base Error Rate* range is 4.3×10^{-3} to 4.3×10^{-7}. These calculations show that the selected *Base Error Rates* (0.01 to 0.0001) only cover BERs ranging from worse than expected to very high. In addition, the calculations do not take into account that the Omnicast Digital service uses forward error correction, which would improve the FER. Thus, it may be concluded that the selected *Base Error Rates* produced results that are somewhat pessimistic.

The experiments only used frame sizes that were multiples of 128, and the smallest frame used was 256 bits because the frame header was modeled as 128 bits. Therefore, the FER is a compound *Base Error Rate*, which is derived by

$$FER = 1 - (1 - Base\ Error\ Rate)^{n}$$
$$where\ \ n = Frame\ Size / 128. \qquad (7.2)$$

The mathematical model computations were carried out with frame sizes of all multiples of 128, which facilitated very accurate optimal frame size estimation. The simulations however, were run with only a subset of the frame sizes used by the mathematical model. This was done for two reasons. First, changing the frame size by 128 bits has a diminishing affect on throughput as frame sizes get larger. And second, Network II.5 simulations corrupt frames randomly, and due to the propagation delay, the closer that a corrupted frame is to the tail of the transmission, the greater is the delay. Thus, a simulation may be advantaged or disadvantaged by the positions of the errors. Selecting frame sizes that are significantly different reduces the probability of a non-optimal frame size producing the best simulation time. Obviously, the anomalies that are produced by the random positioning of corrupted frames could have been resolved by re-simulating each model many times with each frame size to get the averages. However, this would involve several hundred simulations, and since a test run of

model S12-01-32 (see table 7.1) took more than 16 hours on a Pentium 120 with 32 Mbytes of ram, this was not viable.

The mathematical model and the simulation model objectives are explained in section 6.3. Succinctly, the objectives are to obtain the following information:

- the total transmission time and the corresponding throughput,
- the optimal frame size for each *Network Size* and *Base Error Rate* pair,
- the satellite link utilisation rate,
- the average return traffic, and
- the peak return traffic.

The mathematical model, which was implemented in a spreadsheet, produced all of the above information for each experiment. With the Network II.5 simulations, the reports provided the simulation transmission time from which the throughput was calculated. The satellite link utilisation and the return traffic results were provided by the plots and verified by using the information in the reports to obtain numerical values.

7.3 Experimentation results

The experiments that were detailed in section 7.2 were carried out as planned; the mathematical model results and the simulation results of the optimal frame size models are shown in tables 7.1 and 7.2 respectively; the complete mathematical model results and simulation results are given in Appendix B and Appendix C respectively. Note that the model numbers have been coded with information about the experiment. For example, S3-01-96 is a simulation model (mathematical models have the prefix "M") with 3 receivers, a *Base Error Rate* of 0.01 and a frame size of 96 bytes. The results in table 7.1 and table 7.2 show that the optimal frame sizes are generally quite large and the respective throughput results are very good: much better than results of other experiments found in the literature (see section 7.3.2).

Table 7.1. *Mathematical model results with optimal frame sizes.*

Model No.	Network Size	Base Error Rate	File Size (Mbytes)	Optimal Frame Size (bits)	Throughput (bps)	Satellite Link Utilisation (%)	Average Return Traffic (bps)	Peak Return Traffic (bps)
M3-01-112	3	0.01	5	896	1,424,650	97.80	32,457	37,086
M3-005-144	3	0.005	5	1,152	1,573,168	97.88	17,024	18,824
M3-001-288	3	0.001	5	2,304	1,809,614	98.57	3,638	3,811
M3-005-416	3	0.0005	5	3,328	1,872,518	98.77	1,849	1,912
M3-0001-992	3	0.0001	5	7,936	1,957,678	98.96	386	387
M6-01-96	6	0.01	5	768	1,265,654	98.27	59,011	74,908
M6-005-112	6	0.005	5	896	1,438,858	98.13	32,131	37,832
M6-001-224	6	0.001	5	1,792	1,732,625	98.56	7,070	7,634
M6-0005-304	6	0.0005	5	2,432	1,815,476	98.81	3,632	3,826
M6-0001-656	6	0.0001	5	5,248	1,931,490	99.03	767	770
M12-01-80	12	0.01	5	640	1,088,652	98.53	105,221	150,561
M12-005-96	12	0.005	5	768	1,277,876	98.38	58,554	75,849
M12-001-160	12	0.001	5	1,280	1,631,343	98.65	13,708	15,294
M12-0005-224	12	0.0005	5	1,792	1,738,712	98.89	7,089	7,659
M12-0001-480	12	0.0001	5	3,840	1,889,692	98.93	1,513	1,538
M3-01-112	3	0.01	2	896	1,370,667	94.09	31,240	37,274
M3-01-112	3	0.01	10	896	1,441,270	98.94	32,832	37,268
M6-001-224	6	0.001	2	1,792	1,705,390	97.01	6,989	7,639
M6-001-224	6	0.001	10	1,792	1,741,898	99.09	7,098	7,632

Table 7.2. *Simulation results with optimal frame sizes.*

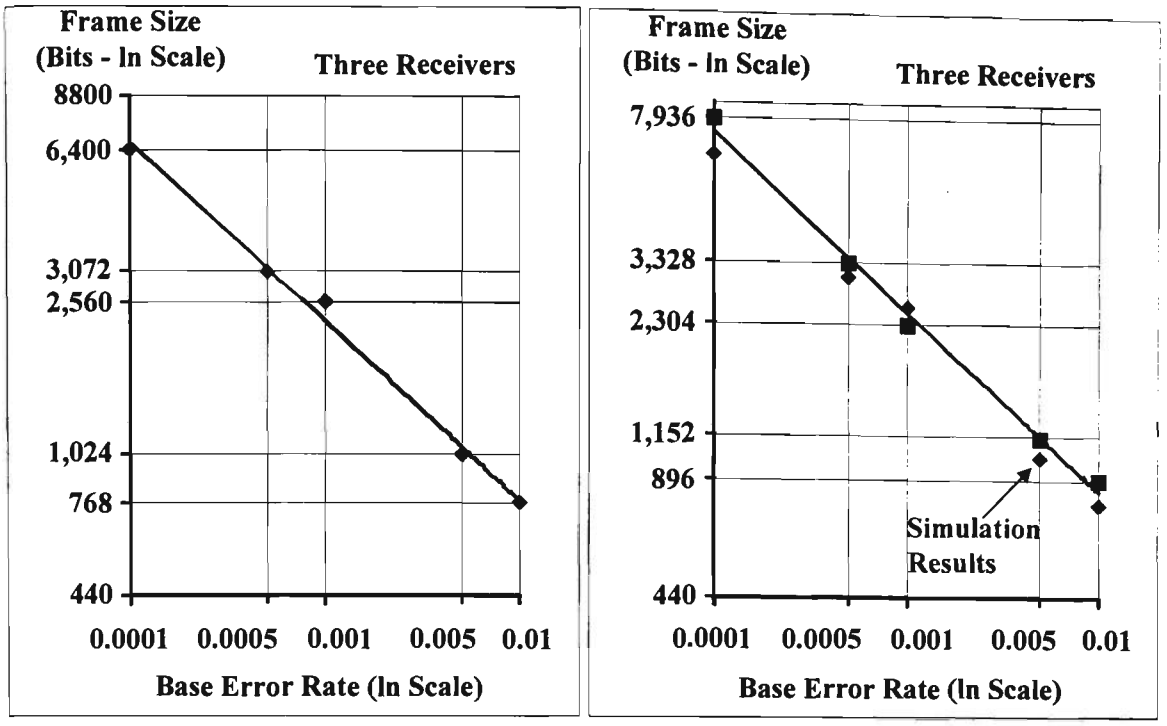
Model No.	Network Size	Base Error Rate	File Size (Mbytes)	Optimal Frame Size (bits)	Throughput (bps)	Satellite Link Utilisation (%)	Average Return Traffic (bps)	Peak Return Traffic (bps)
S3-01-96	3	0.01	5	768	1,424,029	98.37	33,565	37,813
S3-005-128	3	0.005	5	1,024	1,576,171	98.17	16,993	18,494
S3-001-320	3	0.001	5	2,560	1,803,487	97.95	3,438	3,644
S3-005-384	3	0.0005	5	3,072	1,876,240	98.91	1,799	1,871
S3-0001-800	3	0.0001	5	6,400	1,956,874	98.86	363	365
S6-01-096	6	0.01	5	768	1,261,076	97.79	58,633	74,606
S6-005-128	6	0.005	5	1,024	1,442,313	98.35	31,371	37,485
S6-001-160	6	0.001	5	1,280	1,721,504	98.96	7,310	7,736
S6-0005-384	6	0.0005	5	3,072	1,816,127	98.95	3,536	3,783
S6-0001-640	6	0.0001	5	5,120	1,933,077	98.87	695	698
S12-01-064	12	0.01	5	512	1,081,380	98.70	110,959	151,301
S12-005-096	12	0.005	5	768	1,280,497	98.54	58,608	75,686
S12-001-160	12	0.001	5	1,280	1,630,722	98.66	13,832	15,433
S12-0005-256	12	0.0005	5	2,048	1,721,262	97.95	6,949	7,642
S12-0001-512	12	0.0001	5	4,096	1,893,880	98.90	1,450	1,476
S3-01-96	3	0.01	2	768	1,370,866	94.44	31,889	37,424
S3-01-96	3	0.01	10	768	1,437,698	99.06	33,442	37,305
S6-001-192	6	0.001	2	1,536	1,680,636	95.62	6,762	7,397
S6-001-256	6	0.001	10	2,048	1,744,003	99.37	7,088	7,657

The mathematical model results and the simulation results, which are shown in tables 7.1 and 7.2 respectively, exhibit a strong correlation in all aspects: optimal frame sizes, throughput, satellite link utilisation, and the return traffic rates. The optimal frame sizes derived from the simulations and those derived from the mathematical model computations differ for most *Network Size* and *Base Error Rate* combinations (see section 7.3.1). In cases where the two methods produced the same optimal frame size, the throughput and the information about the communication links still differ slightly. This difference in the results comes about from the fact that the simulations used a normal distribution to randomly corrupt frames and discard NAKs that are duplicated. Thus, simulations would sometimes retransmit more frames than was expected statistically, and sometimes less than was expected.

7.3.1 Optimal Frame Sizes

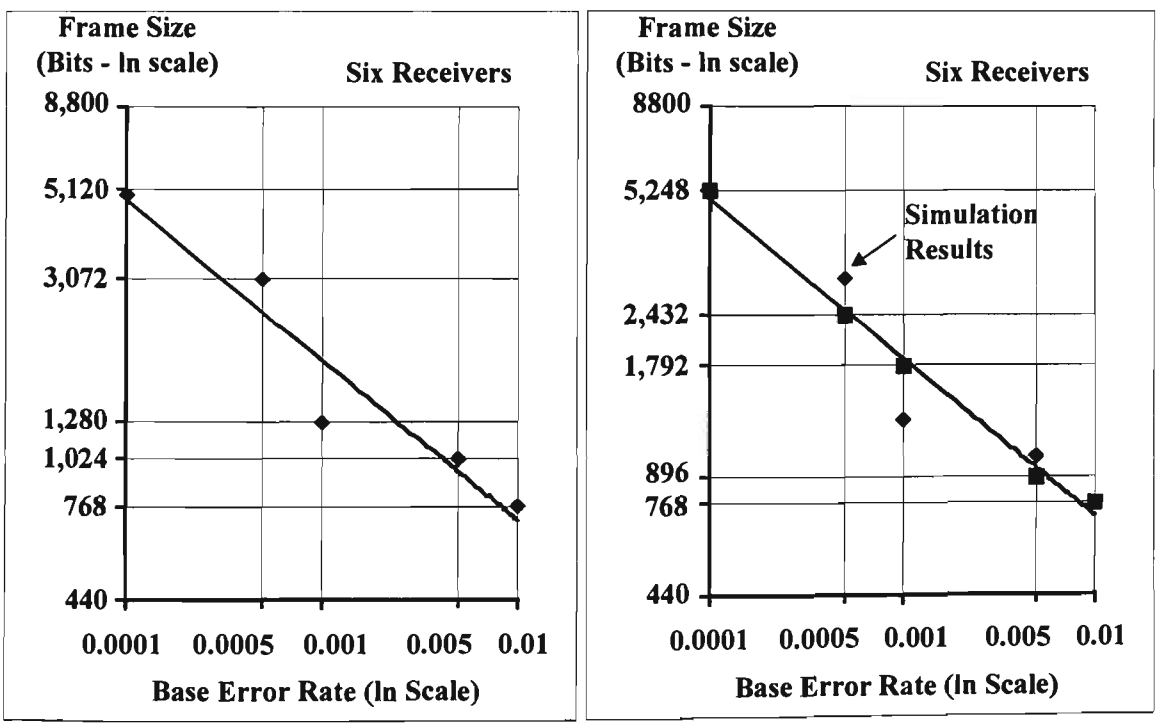
The optimal frame sizes obtained by the mathematical model and the simulations for a 5 Mbyte file transmission to 3, 6 and 12 receivers are shown in figures 7.1, 7.2 and 7.3 respectively. As explained in section 7.2, the simulations were not run with frame sizes of all multiples of 128—the mathematical model was. Therefore, the optimal frame sizes derived by the simulations, in most cases, do not match those derived from the mathematical model. However, the corresponding difference in the throughput and the return traffic are minor. The worst case was the 12 receivers and 0.0005 *Base Error Rate* combination: the simulation model S12-0005-256 produced a throughput of 1,721,262 bps (see table 7.1), and the mathematical model M12-0005-224 produced a throughput of 1,738,712 bps (see table 7.2) which is 1% better.

The optimal frame size trend lines, which are shown in figures 7.1, 7.2 and 7.3, fit quite well with the exception of two outliers, which are discussed later. For comparison purposes all of the trend lines are shown together in figure 7.4. The comparisons show that the trend lines derived from the simulations closely match those obtained from the mathematical model computations.



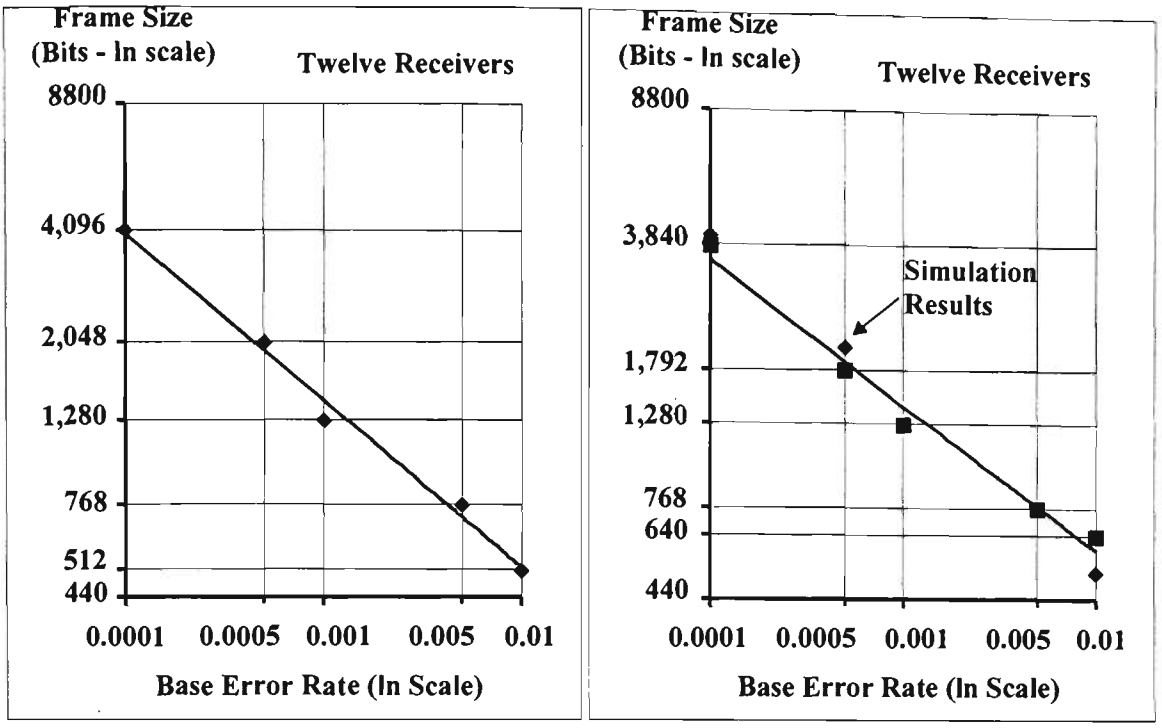
a) Simulation results b) Mathematical model results

Figure 7.1. *Optimal frame sizes for a network with three receivers.*



a) Simulation results b) Mathematical model results

Figure 7.2. *Optimal frame sizes for a network with six receivers.*



a) Simulation results

b) Mathematical model results

Figure 7.3. *Optimal frame sizes for a network with 12 receivers.*

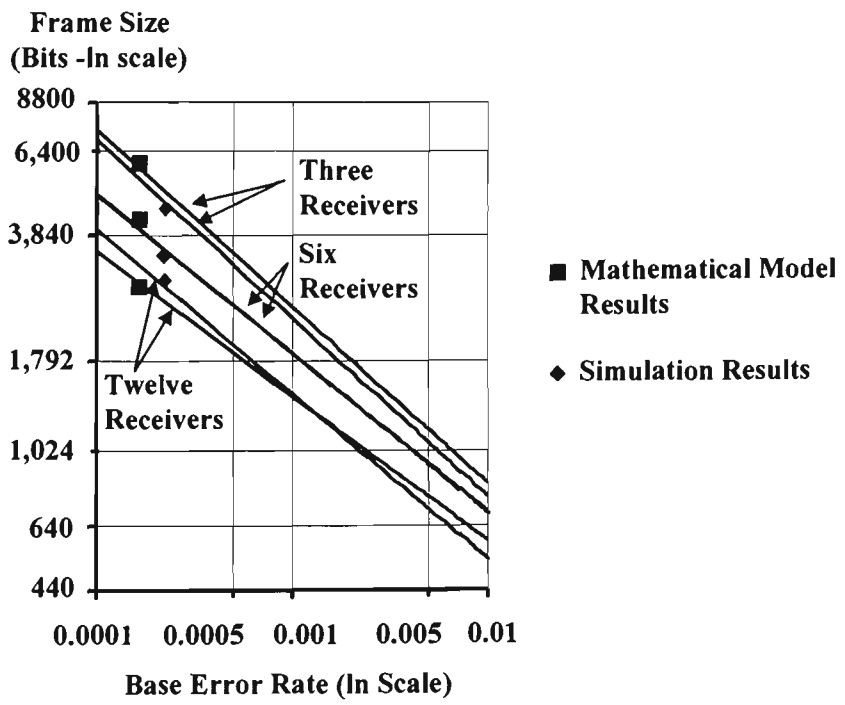


Figure 7.4. *Trend line Comparisons.*

The two optimal frame size outliers mentioned above correspond to the simulation models S6-001-160 and S6-0005-384, and are plotted in figure 7.2a (*Base Error Rates* 0.001 and 0.0005). In viewing figure 7.2a, it can be seen that for the combination of 6 receivers and a 0.001 *Base Error Rate*, the optimal frame size should have been larger. While for the 6 receivers and 0.005 *Base Error Rate* combination, the optimal frame size should have been smaller. The reports that were generated by the Network II.5 simulation package confirmed that the two models in question did have a favourable run with errors in the tail of the transmission. The reports also provided information from which the total number of bits transmitted was derived. The results of the total number of bits transmitted indicate that model S6-001-192 should have produced a better throughput result than model S6-001-160. While model S6-0005-320 should have produced a better throughput result than model S6-0005-384 (see tables C.8 and C.9).

Comparing the total number of bits transmitted is a good guide as to which frame size is closer to the optimal, particularly if the error rate is high. However, this is not always true. First, the file sizes were rounded off to the nearest full frame, and thus, the transmission size was slightly different between most models; the throughput calculations obviously considered this. And second, the frame overheads, the frame error rate and the propagation delay affect on frames in the tail of the transmission may tip the balance. For example, table B.4 shows that model M3-0005-432 transmitted 1329 fewer bits than model M3-0005-416, which transmitted a smaller file. Model M3-0005-432 actually sent fewer bits in the first transmission because of the lower frame overheads (larger frames), but the larger FER resulted in slightly more frames sent in the following transmissions. Thus, the effect of the propagation delay on frames in the tail of the transmission resulted in an inferior performance. Nevertheless, in the case of model S6-001-160, model S6-001-192 transmitted 366 Kbits less, which is a clear indication (because of the volume) that the number of errors in the tail of the transmission played a decisive roll in the transmission time. This argument is supported by the simulation results for the 2 and 10 Mbyte transmissions, which are shown in tables 7.1 and 7.2 and discussed later.

The mathematical model results in table 7.1 show that model M6-001-224 has the optimal frame size (1,792 bits) for the 6 receivers and 0.001 *Base Error Rate* combination. While for the 6 receivers and 0.0005 *Base Error Rate* combination, model M6-0005-304 has the optimal frame size (2,432 bits). These results confirm that the simulations with 6 receivers and a 0.001 *Base Error Rate* combination should have resulted with an optimal frame size of either 1,536 or 2,048 bits (models S6-001-192 or S6-001-256) instead of 1,280 bits (model S6-001-160). Note that models S6-001-192 and S6-001-256 had the nearest frame sizes simulated either side of the mathematical model M6-001-224. Similarly, for the 6 receivers and 0.0005 *Base Error Rate* combination, the simulations should have resulted with an optimal frame size of 2,560 bits (model S6-0005-320) instead of the 3,072 bits (model S6-0005-384).

In regards to the experiments with 2 Mbyte and 10 Mbyte files, the mathematical model results show that the file size does not affect the optimal frame size. The exception to this is when the files are about 100 Kbytes or smaller, which is discussed below. The experiments with a 2 Mbyte file produced slightly lower throughput rates than what was achieved with a 5 Mbyte file. However, this was expected since the satellite propagation delay for the first frame and the frames is the tail of the transmission to reach the receivers naturally has a greater affect on the total transmission time with smaller files. Hence, the experiments with a 10 Mbyte file produced slightly better throughput rates than the experiments with a 5 Mbyte file.

The simulations with 2 Mbyte, 5 Mbyte and 10 Mbyte file transmissions produced the same throughput variation as the mathematical model computations. Regarding the optimal frame size results with the 2 Mbyte and 10 Mbyte transmissions, the simulations were consistent with the mathematical model results for the 3 receivers and 0.01 *Base Error Rate* combination. That is, the optimal frame size (768 bits—896 bit frames were not simulated) was the same for the 2 Mbyte, 5 Mbyte and 10 Mbyte transmissions. However, the simulations with 6 receivers and a 0.001 *Base Error Rate*, resulted with three different optimal

frame sizes for the three file sizes. The initial 5 Mbyte file transmission resulted with an optimal frame size of 1,280 bits (model S3-001-160); it was explained above that the model S3-001-160 had a very favourable run with errors. The corresponding 2 Mbyte and 10 Mbyte file transmissions resulted with optimal frame sizes of 1,536 and 2,048 respectively. These two frame sizes were the nearest frame sizes simulated either side of the optimal frame size (1,792 bits) derived by the mathematical model for the 6 receivers and a 0.001 *Base Error Rate* combination. Therefore, this result is not surprising, and simply, one model had a better run with errors with a 2 Mbyte file transmission, and the other model had a better run with the 10 Mbyte file transmission.

With files smaller than about 100 Kbytes, the propagation delay for the frames in the tail of the transmission has a much greater effect on the transmission time. Therefore, with small files, in most cases it works out more economical to use smaller frames. This is because the lower FER reduces the number of frames sent in the tail of the transmission, and thus, reduces the effect of the satellite propagation delay. The tail of the transmission refers to the concluding part of the transmission where the sender has idle periods between each of the frames transmitted. The differences in optimal frame sizes for 100 Kbyte and 5 Mbyte files for the extreme *Network Size* and *Base Error Rate* combinations are shown in table 7.3.

Table 7.3. *Optimal frame sizes for 100 Kbyte files.*

Network Size	Base Error Rate	Optimal Frame Size (bits)	
		100 Kbyte File	5 Mbyte File
3	0.01	512	896
3	0.0001	7,808	7,936
12	0.01	640	640
12	0.0001	3,712	3,840

The transmission efficiency is maximised with the use of an optimal frame size, and obviously, the greater the deviation from the optimal frame size the lower the efficiency gets. If the frame size is too small then the frame overheads (frame headers) are more influential. If the frame size is too large then the frame error rate increases, which in turn increases the retransmission requirements. Figure 7.5 shows the effects of deviating from the optimal frame sizes for four *Network Size* and *Base Error Rate* combinations:

- 3 receivers and a 0.01 *Base Error Rate*;
- 12 receivers and a 0.01 *Base Error Rate*;
- 3 receivers and a 0.0001 *Base Error Rate*; and
- 12 receivers and a 0.0001 *Base Error Rate*.

In analysing figure 7.5, it can be seen that the two lines that are based on a 0.01 *Base Error Rate* (lines c and d) curve downward more sharply than the two lines that are based on a 0.0001 *Base Error Rate* (lines a and b). The sharper line curvature of lines c and d indicates that the frame size has a greater influence on the effective throughput when the error rate is high. This is confirmed by figure 7.6, which shows how deviation from the optimal frame size effects the throughput. The achievable throughput represents the throughput as a percentage of the throughput achieved with an optimal frame size. Figure 7.6 also shows that the *Network Size* does not greatly influence the effect of a non-optimal frame size on the performance. The exception is with the 12 receivers and 0.01 *Base Error Rate* combination (line d) for frame sizes that are more than six times larger than the optimal. From figure 7.5 and figure 7.6, it can be concluded that a firm strategy for the selection of the frame size is required only if the error rate is high. If the error rate is low, then the performance is not greatly reduced unless a very small frame size is used. With low error rates, the optimal frame sizes are large to begin with, and it is not likely that anyone would use frames sizes large enough to reduce the transmission efficiency substantially. The effect that non-optimal frame sizes have on the RPC (Remote Procedure Call) and TCP protocol performance is given in [25].

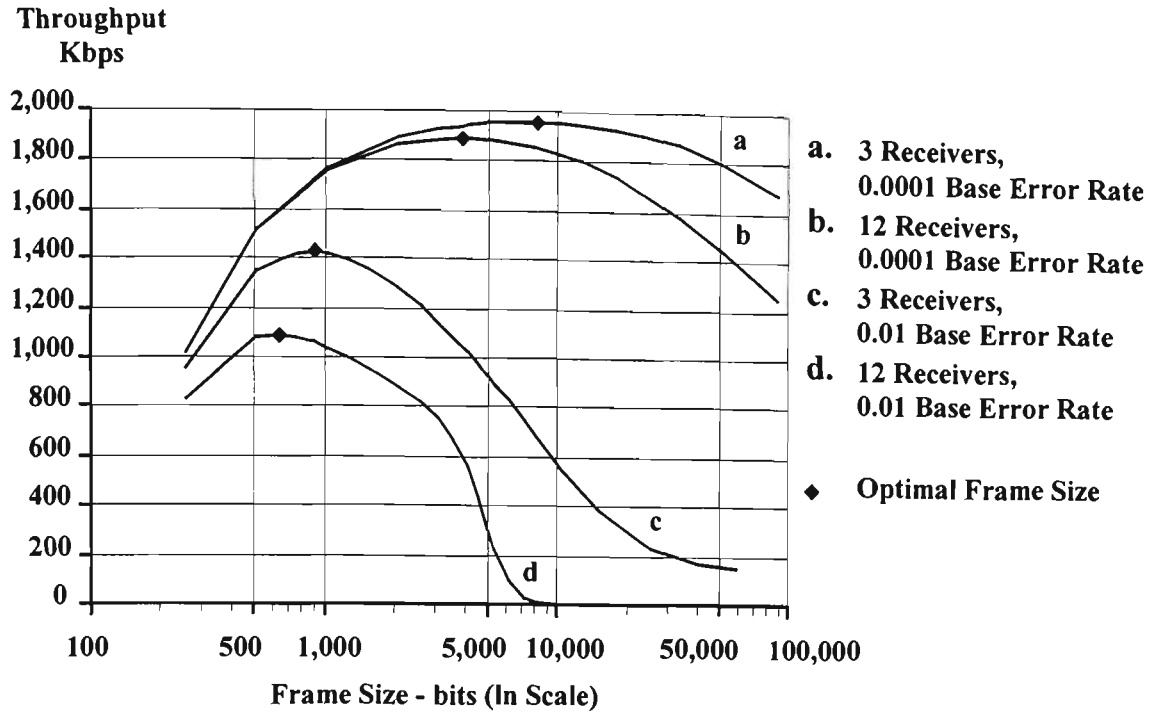


Figure 7.5. *Throughput for non-optimal frame sizes.*

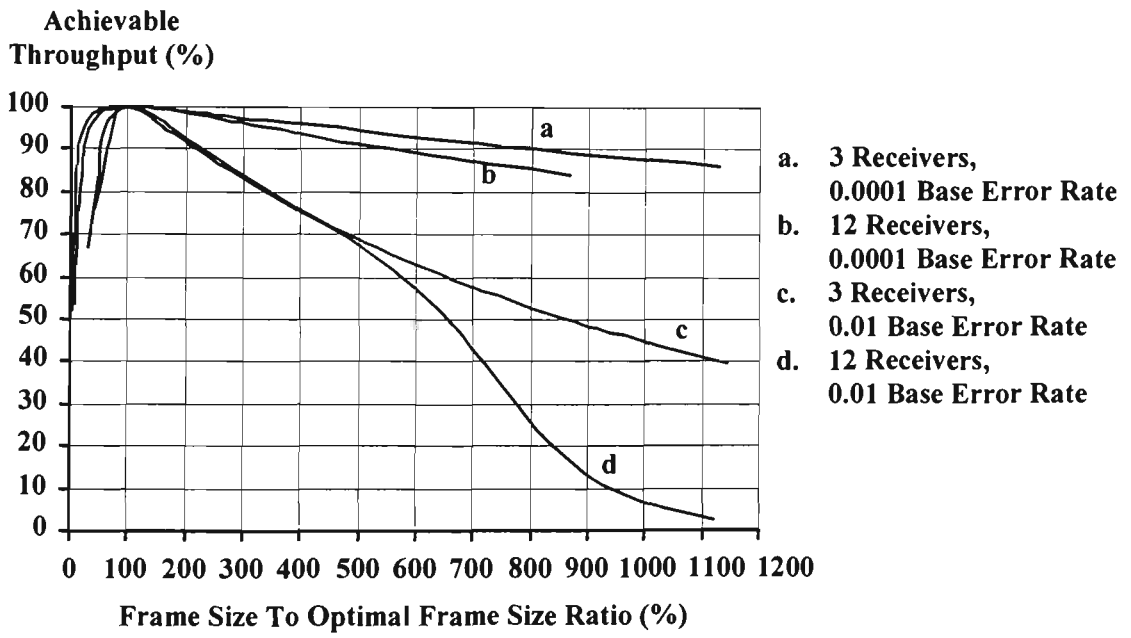


Figure 7.6. *Non optimal frame size effect on throughput.*

Considering that the lowest *Base Error Rate* used by the experiments in this thesis was 0.0001, which is equivalent to a BER of 2.3×10^{-6} and still quite high, the optimal frame sizes should generally be large. RSBT has an advantage over other protocols in that it is not restricted in its selection of frame sizes. For

example, TCP is restricted to a maximum packet size of 1024 bytes [52]. The RPC protocol is basically restricted to a maximum packet size of 960 bytes because the processing overheads for larger packets reduce the throughput [25]. While Novell's Burst Mode Protocol packet size may be up to 1500 bytes, but it uses the Internet Packet Exchange protocol, which has a maximum packet size of 512 bytes [20]. However, in many cases, there may not be much advantage in maximising the frame size just because the error rate is low. For example, model M12-0001-480 had a throughput of 1,889,692 bps (see table 7.2) and the frame header represented only 3.3% of frame length. If the error rate were to decrease, the throughput would increase simply because of the reduced retransmissions. The throughput could be further improved by increasing the frame size to the optimal size for the new error rate. The larger frame size would reduce the already small frame overheads, but this would be counteracted to a large degree by the increase in the FER. Therefore, a frame size between say 4096 and 8192 bits (512 and 1024 bytes) would be a good choice because it would still perform well with low error rates, and if the error rate suddenly increased then the effect on the throughput would not be as great.

7.3.2 Throughput

For the system configuration that the RSBT protocol was developed for (see figure 3.1), the most important factor is the effective throughput; reliability is taken for granted—a conventional Reverse Error Control method is used. The RSBT experiments produced very good throughput results. In the worst case, as shown in table 7.1, model S12-01-064 produced a throughput of 1,081,380 bps, which is better than all the results found in the literature, involving various protocols in point-to-point satellite transmission.

The throughput results for experiments with 5 Mbyte files, shown in tables 7.1 and 7.2, ranged from 1,081,380 bps with model S12-01-064 to 1,957,678 bps with model M3-0001-992. This throughput range translates to a transmission efficiency range from 52.8% to 95.6%. All throughput results for the 5 Mbyte file transmissions are shown in figure 7.7, which highlights two facts. First, the

throughput difference between the three network sizes is greater when the error rate is high. This is expected purely because of the greater retransmission demands with a larger network. Note also that the larger network sizes resulted with smaller optimal frame sizes. The smaller optimal frame sizes reduce the frame error rate and thus the number of retransmissions that are required, but they increase the influence of the frame overheads. And second, the throughputs for all three network sizes are close to peak performance at a *Base Error Rate* of 0.0001, which is equivalent to a 2.3 x 10^{-6} BER. Thus, RSBT does not require the error rate to be very low to get good performance.

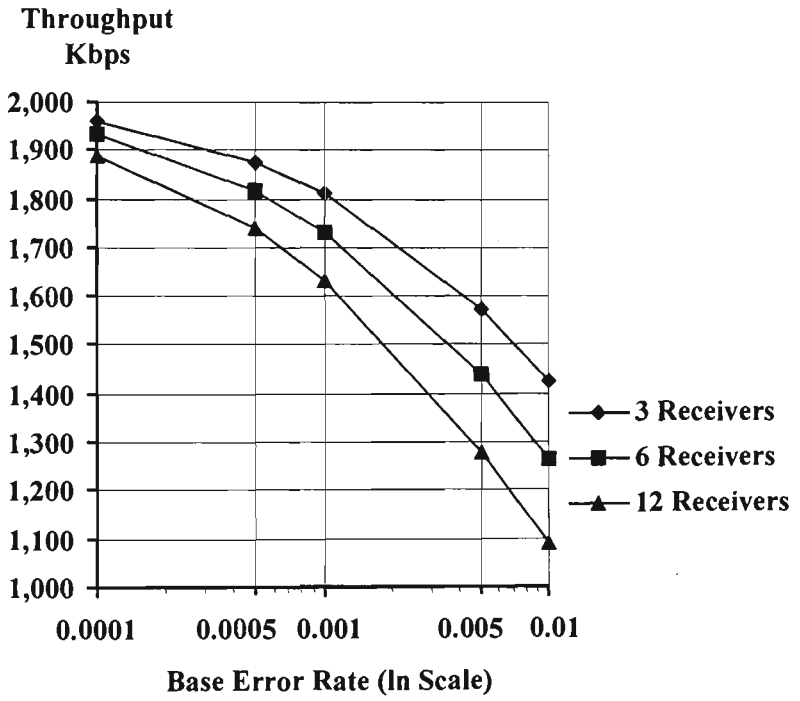


Figure 7.7. *Throughput results.*

The RSBT throughput results for the 2 Mbyte files are not much lower than that for 5 Mbyte files. Table 7.2 shows that model M3-01-112 had a throughput of 1,370,667 bps with a 2 Mbyte file, which is 3.8% less than the 1,424,650 bps achieved with a 5 Mbyte file. While Model M6-001-224 had a throughput of 1,732,625 bps with a 5 Mbyte file and 1,705,390 bps with a 2 Mbyte file, which is 1.6% lower. Table 7.2 also shows that the 10 Mbyte transmissions produced higher throughput rates than the 5 Mbyte transmissions: 1.2% higher for model M3-01-112 and 0.5% higher for model M6-001-224.

Transmission of smaller files will always result in a lower throughput. This is because the satellite propagation delay for the first frame, and for frames in the tail of the transmission, represents a larger proportion of the transmission time (see figure 7.8 and figure 7.9 in section 7.3.3). Table 7.4 compares throughput results of 100 Kbyte and 5 Mbyte transmissions for the same four *Network Size* and *Base Error Rate* combinations shown in table 7.3.

Table 7.4. *Throughput results for 100 Kbyte transmissions.*

Network Size	Base Error Rate	Throughput (bps)	
		100 Kbyte File	5 Mbyte File
3	0.01	700,689	1,424,650
3	0.0001	1,148,885	1,957,678
12	0.01	586,379	1,088,652
12	0.0001	1,141,683	1,889,692

RSBT has the option of using pre-emptive retransmission to improve performance with small files. That is, when the sender has transmitted all of the frames in the queue and becomes idle, it may start retransmitting frames while it is awaiting acknowledgments; the pre-emptive retransmission would use the round trip time to determine which frames may still be negatively acknowledged. With very small files, pre-emptive retransmission could result in the file being transmitted multiple times before the sender has received any acknowledgments.

The performance of TCP and the Burst Mode Protocol is also affected by the satellite propagation delay in the same way that RSBT's performance is affected. The problem is compounded for TCP because it uses a slow-start algorithm [24],[25], which slowly increase the number of acknowledgments that can be outstanding. Thus, the TCP throughput is greatly restricted in the early stage of the transmission, and therefore, TCP performs very poorly with files that are smaller than 100 Kbytes. Results in [24] show that with 100 Kbyte files, TCP has a throughput of approximately 170 Kbps using a window size of 64 Kbytes;

TCP approaches its peak performance with files larger than 500 Kbytes [24]. The Burst Mode Protocol also uses an algorithm similar to the slow-start algorithm used by TCP, and thus, performs poorly with small file transmissions [6],[20]. In fact, results in [6] show that TCP and the Burst Mode Protocol have identical performance for files up to 100 Kbytes; the Burst Mode protocol also approaches peak performance with transmissions larger than 500 Kbytes [6].

Experiments in [24],[25], which used TCP/IP over a satellite link, produced throughputs peaking at approximately 300 Kbps, regardless of the channel speed. The main causes for the limit in the TCP throughput were said to be the windowing flow control mechanism and the slow start algorithm [24],[25]. Hence, TCP/IP in a modified form, a larger window, increased the throughput to the proximity of 500 Kbps [24],[25]. Better results were achieved by two other protocols. The first, a Quick File Transfer Protocol (QFTP), produced a throughput of 910 Kbps over a 2 Mbps satellite channel [25]. QFTP used the User Datagram Protocol (UDP) with a large window size and a Go-back-N recovery protocol, over a predominantly error free channel [25]. The second, Novell's Burst Mode Protocol, had a throughput of 2.4 Mbps over a 10.4 Mbps satellite channel [6]. All of the results quoted from [6],[24],[25] were based on very low error rates and point-to-point transmissions. In contrast, the experiments with the RSBT protocol were based on high error rates and broadcast transmissions.

TCP experiments over a satellite link in [28] produced far better results than the experiments discussed above. One experiment in [28] used of a link layer protocol (with Selective Repeat retransmission) to shield TCP from the satellite link, resulting in a throughput of 1.22 Mbps. A throughput of 1.1 Mbps was achieved by another experiment in [28], which split the TCP transmission into three parts: local transmission on the sending network, satellite transmission, and local transmission on the receiving network. Recall from section 3.3.1 that the RSBT protocol is also used in a three-step transmission. The experiments in [28] were based on an 8 Mbyte file transmission, a 2 Mbps satellite channel and a BER of 1.9×10^{-6}.

The experiments that used a 0.0001 *Base Error Rate*, which is equivalent to a BER of 2.3 x 10^{-6}, can be compared to the performance results in [28]. However, note that the throughput results given in this thesis are for the satellite transmission only. Therefore, to calculate the end-to-end throughput a portion of the TCP transmission time on the sending LAN would have to be added to the RSBT transmission time. Only a portion of the TCP transmission time would be included in the calculations because RSBT can start transmitting frames over the satellite link as soon as enough data for one frame has been received over the LAN. Therefore, the end-to-end throughput would not be much lower than the satellite link throughput. The end-to-end throughput calculations should not include the receivers' TCP transmission time over the LAN because it does not involve the sender or affect the senders throughput capacity; while the receivers are transmitting data over the LAN, the sender can start a new file transfer over the satellite. If the complete end-to-end transmission time (and throughput) is desired, then the transmission time is augmented by the time required to transmit half the file over a receiving LAN. The TCP transmission is halved because statistically, on average, the last frame to be retransmitted is the frame in the middle of the sequence (see equations (5.12), (5.13) and (5.14). Therefore, because receivers can commence the transmission of TCP packets (in sequence) as frames arrive over the satellite link, half of the file would have been sent over the LAN by the time the final frame is received over the satellite.

One reason for the RSBT protocol's good performance is the use of generally large frames, which minimises the frame overheads. In the worst case, model S12-01-64 had a frame header that made up 25% of the frame. Conversely, model M3-001-992 had a frame header that made up only 1.6% of the frame. However, the main reason for RSBT's good performance is the continuous transmission; RSBT does not use flow control (see section 3.5), and thus, it does not stop to wait for acknowledgments. Therefore, the satellite propagation delay does not affect the RSBT performance in the same way that it affects the performance of TCP and other protocols.

7.3.3 Satellite Link Utilisation

There is a strong correlation between the satellite link utilisation and the protocol performance; this has been illustrated in [25]. Basically, a good throughput is not possible without a high satellite link utilisation. The RSBT protocol does not use any flow control, and thus, has a very high link utilisation. In fact, the simulation plots have shown the link utilisation to be at 100% for most of the transmissions. The high satellite link utilisation indicates that the satellite propagation delay has little effect on the throughput for large files.

Figure 7.8 shows the satellite link utilisation for model S6-005-128 with a 5 Mbyte file; other models differed slightly only in the way that the link utilisation tapered off in the tail of the transmission. As shown in tables 7.1 and 7.2, most of the experiments had a satellite link utilisation between 97% and 99%, except models S3-01-96 and M3-01-112 which drooped to about 94% for 2 Mbyte files. The link utilisation results derived by the mathematical model for the 100 Kbyte transmissions (shown in tables 7.3 and 7.4), dropped to between 51% and 60%. The much lower link utilisation for smaller files was confirmed by re-simulating model S6-005-128 with a 100 Kbyte file; the satellite link utilisation is shown in figure 7.9. By comparing figure 7.8 and figure 7.9, it is clear that the propagation delay has a much greater effect on the transmission of small files; this was explained in section 7.3.2. Model S6-005-128 took 28.399 seconds to transmit the 5 Mbyte file, and the link utilisation started to drop off sharply at 27.832 seconds which was 0.567 seconds before completion; the exact times were provide by the Network II.5 analysis tools. While with the 100 Kbyte file, model S6-005-128 took 1.295 seconds to complete the transmission, and the link utilisation started to drop off sharply at 0.507 seconds which was 0.688 seconds before completion. These results show that the throughput drops off drastically in the tail of the transmission for all file sizes, but the tail represents a much bigger proportion of the transmission with small files. The experiments with the 100 Kbyte files were not part of the main experimentation objectives, and thus, were not included in the results given in Appendix B and Appendix C.

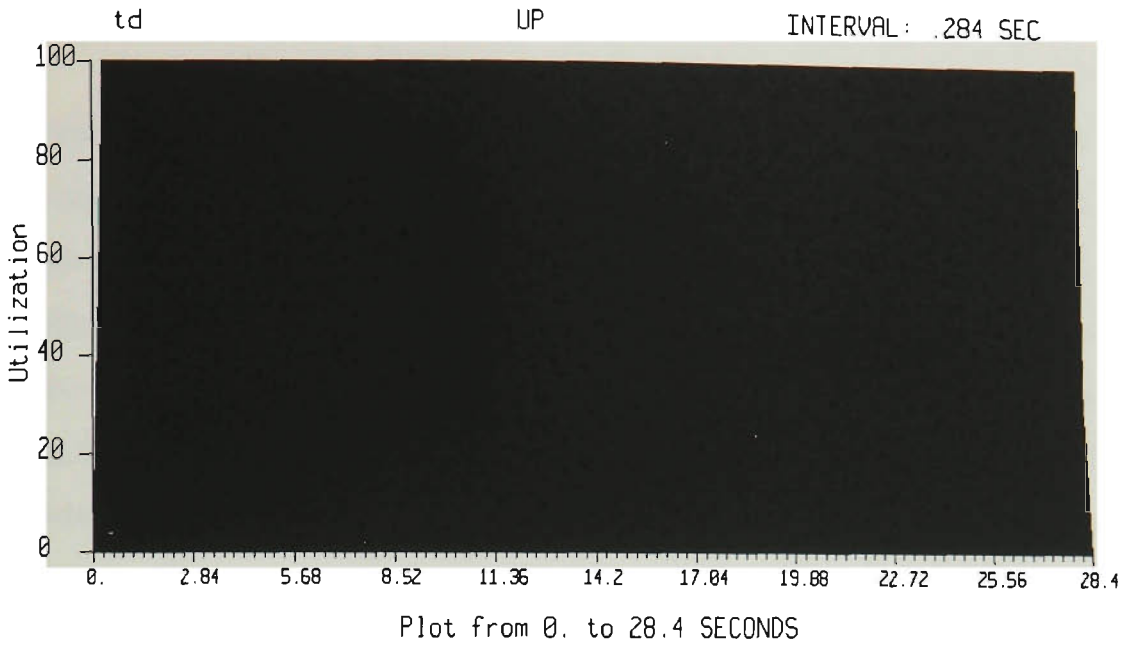


Figure 7.8. *Satellite link utilisation for model S6-005-128 with a 5 Mbyte file.*

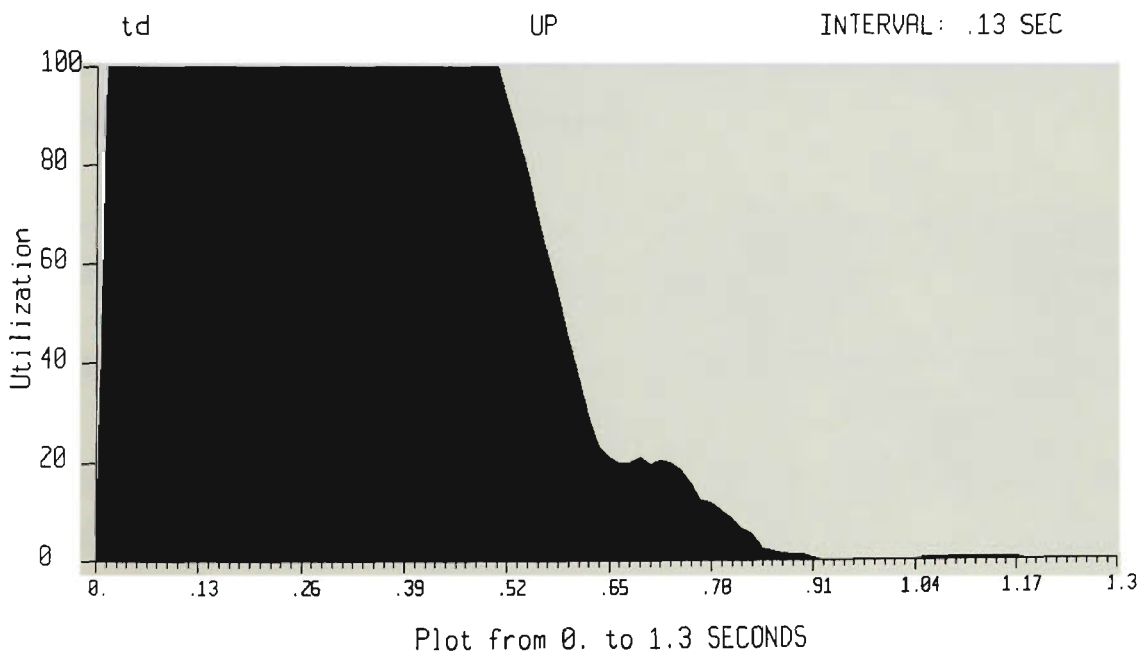
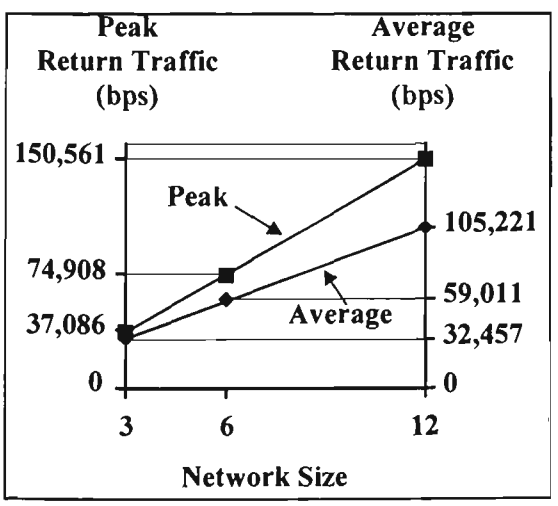


Figure 7.9. *Satellite link utilisation for model S6-005-128 with a 100 Kbyte file.*

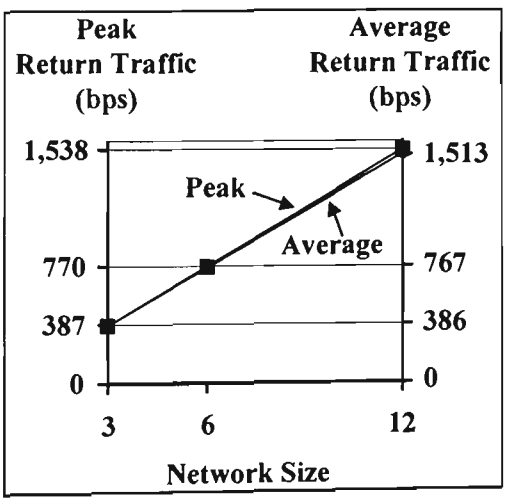
7.3.4 Return Traffic

In providing reliability to a broadcast transmission, there must be some concern about the sender's incoming traffic, i.e. acknowledgments, which were simulated as 80 bit frames. Figure 7.10 shows the RSBT return traffic that was derived by the mathematical model for the 0.01 and 0.0001 *Base Error Rates* with optimal frame sizes. The return traffic results for all *Network Size* and *Base Error Rate* combinations (with optimal frame sizes) obtained by the mathematical model and the simulations, are given in tables 7.1 and 7.2 respectively; the full results are given in Appendix B and Appendix C. Figure 7.10 shows that in the worst case—the 12 receivers and 0.01 *Base Error Rate* combination—the peak return traffic rate was 151 Kbps. Recall from section 7.2, that the lowest *Base Error Rate* expected is 4.3×10^{-3} (10^{-4} BER), so this volume of return traffic for 12 receivers would be very rare. The use of non-optimal frame sizes does not greatly affect the return traffic volume, particularly the peak return traffic; this can be verified by analysing the results in Appendix B and Appendix C.

The results show that the return traffic should not produce any congestion at the sending site, and that much larger networks can be served by the RSBT protocol before a bottleneck is encountered.



a) With a 0.01 *Base Error Rate*



b) With a 0.0001 *Base Error Rate*

Figure 7.10. *Return traffic.*

7.4 Conclusion

This chapter presented the experimentation plan and an analysis of the results. The experiment plan, which was outlined in section 7.2, minimised the number of experiments that had to be carried out, yet satisfied the experimentation objectives, which were given in section 6.3. The mathematical model results are more accurate than the simulation results because the mathematical model computations were run with frame sizes of all multiples of 128; the simulations were run with a selected subset of frame sizes. Furthermore, the simulations used some degree of randomness in the selection of certain events, while the mathematical model used precise expected values, which provided consistency between experiments. However, in most cases there is only a minor differences in the results produced by the mathematical model and those produced by the simulations, i.e. throughput, satellite link utilisation and return traffic.

The optimal frame sizes that were derived by the experiments are generally quite large, resulting in low frame overheads. The experiments with the lowest *Base Error Rate*—0.0001—resulted in the transmission efficiency approaching its peak. This indicates that there would not be much advantage in using frame sizes larger than 1,024 bytes even if the error rates is lower; trying to keep the frame size as close as possible to the optimal all of the time would require continuous monitoring of the channel error rates. The file size does not affect the optimal frame size directly. This however, is not true when very small files are transmitted, because the effect of the satellite propagation delay does force the optimal frame sizes down.

The throughput results are very high. This is because the RSBT protocol was tailored for satellite transmission, and thus avoids procedures that hinder performance over long delay links. The good performance can be attributed to three main factors: low frame overheads; the use of Selective Repeat retransmission; and most importantly, the removal of flow control. It should be noted that the RSBT throughput results are not end-to-end results; they represent the throughput for the satellite hop only. However, RSBT can start transmitting

frames over the satellite link as soon as enough data for one frame has been received over the LAN. Therefore, the end-to-end throughput would not be much lower than what is shown in tables 7.1 and 7.2. The throughput calculations should not include the receivers' LAN transmission time because, as explained in section 7.3.2, it does not affect the sender in any way.

High throughput over a satellite link is not possible without high link utilisation. The RSBT link utilization is very high because flow control was not incorporated into the protocol.

The return traffic rates produced by the RSBT protocol are reasonably low; except for the *Acknowledge All Frames* acknowledgment protocol. Low return traffic rates was a design objective for the RSBT acknowledgment protocols, which will allow the RSBT protocol to service much larger networks than the 12 receivers used in these experiments.

8. Discussion

8.1 Thesis Summary

This thesis has presented the research carried out to formulate the RSBT protocol and analyse its performance. Figure 8.1 shows each stage of the RSBT protocol development and the information flow that linked and ordered each of these stages.

The need for the RSBT protocol came from Australia's Bureau of Meteorology, who disseminate data to multiple receivers nation-wide on a regular basis. Thus, the aim of the thesis was based on the Bureau's problem definition and requirements. Once the thesis aim was defined, a literature review was carried out to see what research has already been done in the area of reliable broadcasting/multicasting over a satellite link and on terrestrial networks. The literature review was also carried out to uncover shortcomings and good features of existing protocols, such as TCP and NETBLT, when used for point-to-point satellite communication. Using the information gathered from the literature, the RSBT protocol procedures were formulated. The RSBT protocol development was completed by formally specifying and verifying the protocol with SDL.

To evaluate the RSBT protocol, a mathematical model and a simulation model were developed and experimented with. The mathematical model results and the simulation results were analysed, compared against each other, and compared against results of protocols used for point-to-point satellite transmission. Finally, conclusions were drawn; these are given in Chapter 9 along with a discussion on future research.

The following seven sections discuss the main points of each stage of the RSBT protocol development and evaluation.

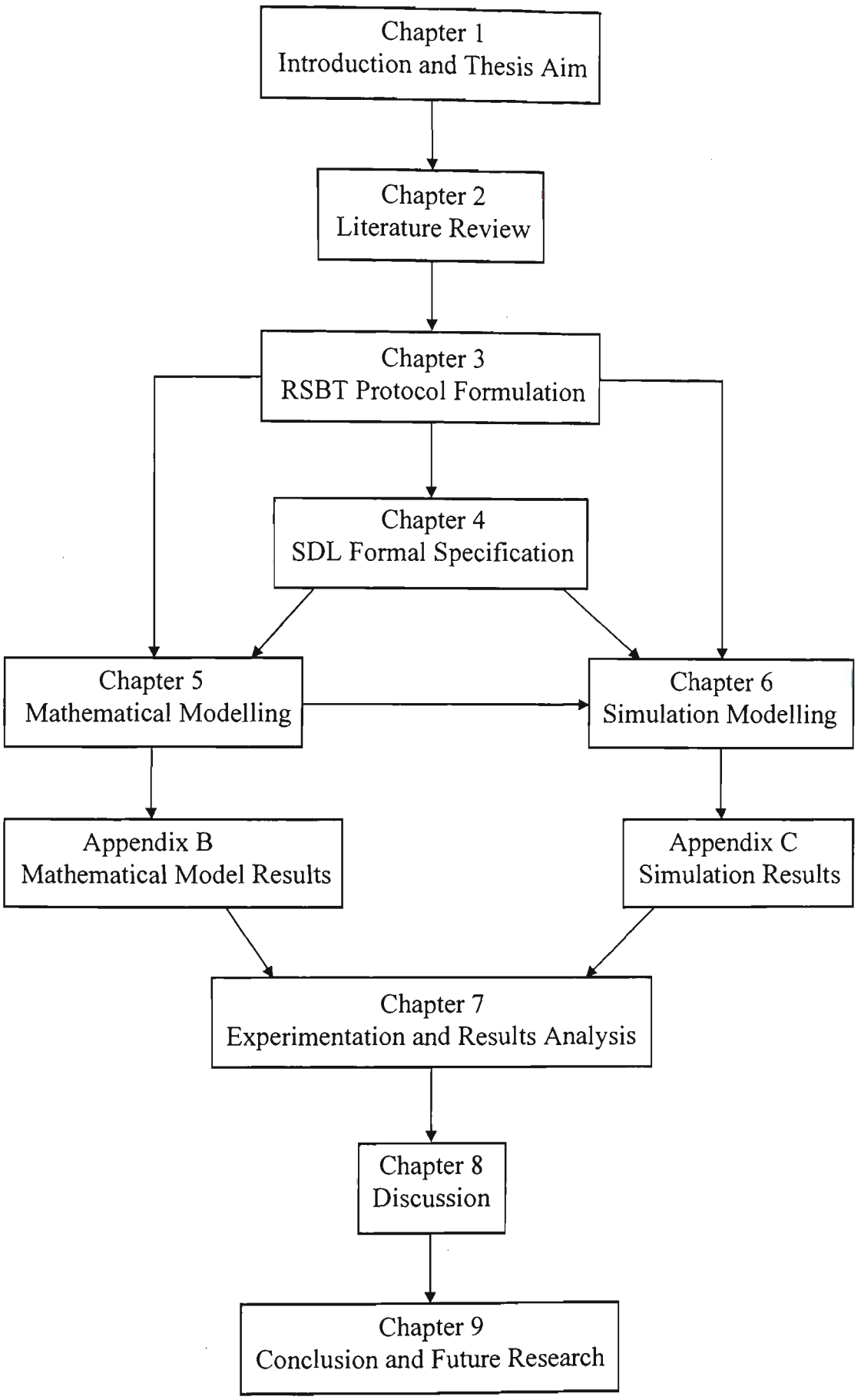


Figure 8.1. *Thesis structure and information flow.*

8.2 Thesis Aim

The aim of this thesis was to develop a protocol that can deliver data reliably to multiple receivers over a satellite link (see section 1.1). Surprisingly, very little work has been done in this area. Therefore, the development of the RSBT protocol was an ideal thesis topic. The research objectives, which expanded on the aim (see section 1.1), were designed to ensure that the RSBT protocol was efficient and semantically correct, and that suitable experimentation results could be obtained for comparison against results (in the literature) obtained by other protocols.

8.3 State of Play

As mentioned in section 1.1, the Australian Bureau of Meteorology requires a suitable protocol for reliable satellite transmission to multiple receivers (broadcasting). A reliable broadcasting protocol in its self is not sufficient. The protocol must also be efficient – more efficient than the current method. A literature survey did not uncover any existing reliable broadcasting/multicasting protocol (until recently). Moreover, protocols that are commonly used for reliable point-to-point communication, such as TCP, are very inefficient when used over a satellite link.

Some protocols such as HDLC cater for broadcast/multicast addressing, but when a group address is used, the transmission reliability procedures are disregarded. Similarly, the Multicasting Backbone (RFC 1112) [9], [53], does not provide reliability to a multicast transmission either. Research presented in [33] described three methods for multicasting across an ATM networks. Two of these methods were reliable, albeit not ideal. The three methods of multicasting presented in [33] are as follows:

1. *Separate Addressing.* This method is nothing more than multiple unicast transmissions. The efficiency of this method rapidly diminishes as the number of receivers increases.

2. *Multicast Tree.* This method reduces much of the sender's workload. The sender transmits data only to a few receivers. Each of those receivers then transmits the data to a few other receivers, and so on until all receivers have been covered. This method, like separate addressing, provides reliability, but is much more efficient when a large number of receivers are involved.
3. *Hardware Delivery.* This method uses a switch to deliver data to all receivers simultaneously. This method is of course the most efficient, but provides no reliability.

The Canadian Meteorological Centre has implemented a system very similar to the proposed system configuration shown in figure 3.1. However, reliability procedures are not built into the protocol used there. Instead, retransmission requests are made by automatic email, triggered by a user who spots the error [11].

Some interesting research on methods for improving the TCP performance over a satellite link has been presented in [28]. Three basic methods were described. These are, in ascending order of performance improvement, as follows:

1. *End-to-End Proposals.* With this method, TCP uses selective acknowledgment to recover from multiple packet losses in a window, without resorting to a course time-out. In addition, an Explicit Loss Notification mechanism allowed the sender to distinguish between congestion losses and other forms of losses.
2. *Split-Connection Proposals.* This method completely hides the satellite link from the sender by terminating the connection at the satellite base station. This allows another (more efficient) protocol to be used over the link.
3. *Link-Layer Proposals.* This approach hides most of the link related losses from the sender by using local retransmission and perhaps Forward Error Correction.

It was stated above that a protocol for reliable broadcasting/multicasting was not found in the literature. However, this is no longer true. Several protocols for reliable multicasting and related research have appeared in the literature recently [54],[55],[56],[57],[58],[59]. These protocols have concentrated on transmission across terrestrial network, but are still very interesting and will be reviewed when further research is done with the RSBT protocol.

8.4 System Configuration

The ideal system configuration, as depicted in figure 3.1, includes a satellite interface on each LAN. The transmission procedure is similar to the *split-connection* approach outlined in section 8.3. That is, data is sent to the sending satellite interface using FTP/TCP. The sending satellite converts the FTP packets to RSBT frames (see section 3.3.3) and sends them across the satellite. Finally, each receiving satellite interface converts the frames back to FTP packets and sends the data to the appropriate node on the network using TCP.

The Australian Bureau of Meteorology uses their supercomputer to generate the files that are to be broadcast. Therefore, although the *link-layer* approach mentioned in section 8.3 produced the best results, it would seem logical for the satellite broadcasting to be performed by another node (a satellite interface) on the LAN. The supercomputer could not disregard other computing requirements each time a file needs to be disseminated. Thus, the broadcast transmission performance would suffer. Moreover, it is assumed that the *split-connection* approach, outlined in section 8.3, does not begin the satellite transmission until the TCP transmission across the LAN has been completed. Otherwise, the *split-connection* approach would be more efficient. As stated in section 3.2, RSBT may begin transmission across the satellite link as soon as enough data for one frame has been received by the sending satellite interface. A satellite interface is also used on each receiving LAN, mainly to cater for the continuous incoming stream of data. The receiving satellite interfaces also streamline the acknowledgment process, and remove the need for additional RSBT implementations—because FTP/TCP is used across the LANs.

8.5 RSBT Features

RSBT was designed not only for reliable broadcast transmission, but also for efficient transmission over a satellite link with minimal return traffic. The following five subsections discuss the RSBT features that enabled these goals to be realised. The complete RSBT procedures are given in chapter 3.

8.5.1 Session Checking

As stated in section 3.5.1, the communication channel session check is optional. In most cases, the session check would not be necessary, and in some cases, it could even cause unnecessary long delays. RSBT deals with multiple receivers and it is not likely that all receivers will lose their reception at the same time. Therefore, unlike point-to-point communication, the transmission is never likely to be a complete waste of time. Moreover, if a receiver regains its session part way through the broadcast, it simply sends the appropriate NAKs. A session check may be suitable for small networks—say two or three receivers—or for networks where all receivers are in close proximity, which would be unusual for networks involving a satellite link.

8.5.2 Flow Control

RSBT does not use any flow control; all frames are transmitted contiguously. Without flow control, the affect of the satellite propagation delay is all but eliminated. The propagation delay still has a significant affect on small file transfers, but this problem can be alleviated to a large degree with pre-emptive retransmission (see section 8.5.5). The RSBT protocol does not require any flow control because it has been designed for transmission over a dedicated satellite link, with no intermediate nodes. Moreover, the 2 Mbps satellite link that RSBT has been tailored for, is not fast enough to cause overflow losses. It has been shown that flow control greatly inhibits throughput over a satellite link [16],[18]. The NETBLT protocol addresses this problem by putting a number of frames into a buffer, and transmitting all of the frames in the buffer contiguously. While the sender is waiting for acknowledgments for one buffer, it begins transmission of

another buffer. Thus, the main difference between RSBT and NETBLT is that NETBLT splits the file into several pieces (buffers).

8.5.3 Error Detection

The RSBT protocol uses one FCS for the frame header and another FCS for the frame data; protocols such as NETBLT [7] and ATM [33] also use two FCSs. The use of two FCSs allows for a more efficient acknowledgment process, particularly in high error rate scenarios.

In most cases, the separate CRC for the frame header allows RSBT to send a NAKs for a corrupted frame earlier. This is because the frame can be negatively acknowledged only if the frame header is intact; if only one CRC is used, then the header integrity can not be guaranteed. The advantage of using two CRCs can be appreciated by considering the fact that the frame header will, in most case, be much smaller than the frame data, and thus, will have a correspondingly lower loss rate. The FER for the experiments carried out with optimal frame sizes, ranged from approximately 0.003 to 0.07 (see Appendix B and Appendix C). While the header error rate is equivalent to the Base Error Rate (0.01 to 0.0001). Thus, without a frame header FCS, the delays in the transmission of NAKs would have a compounding effect for frames lost in the tail of the transmission; this could be very significant with high error rates and small files.

8.5.4 Return Traffic

Return traffic from multiple receivers could easily overwhelm the sender if an appropriate method of acknowledgment is not employed. Thus, the RSBT protocol was designed with four alternatives for the acknowledgment and retransmission process (see section 3.5.2). Succinctly, the four alternatives are:

1. *Acknowledge All Frames*
2. *Acknowledge Last Frame Only*
3. *Acknowledge First and Last Frames*
4. *Acknowledge Lapse in Transmission*

Acknowledgment method 1 (*Acknowledge All Frames*) is based on positive acknowledgement, and makes no attempt to minimise the volume of return traffic—receivers send an ACK for each frame. However, this method is the most efficient, and therefore, is suitable for small networks. This method is the most efficient because the sender retransmits frames that have not been positively acknowledged. Negative acknowledgement is not as efficient for two reasons. First, when a frame header is lost, a receiver can not transmit a NAK until another frame header is received intact. Therefore, a delay in the transmission is incurred if the sender has no other frames to retransmit. And second, the sender must wait the round trip time (at least) for a NAK to be received back, before the frame can be retransmitted. With method 1, the sender keeps retransmitting all frames still awaiting acknowledgement, if there are no other frames to retransmit.

Acknowledgement methods 2, 3 and 4 were motivated by the need to minimise the return traffic. These three methods are not significantly less efficient than method 1, yet they reduce the volume of return traffic drastically. This is because these methods are based on negative acknowledgment and receivers send only one ACK when all frames have been received; with method 3, receivers also send an ACK for frame one if it is received intact. The return traffic paucity is evident in the results; this can be seen in the summary of the results in tables 7.1 and 7.2, and in the full results given in Appendix B and Appendix C. Although the experiments were based on the *Acknowledge Last Frame Only* protocol, the return traffic will not differ significantly with methods 3 and 4.

8.5.5 Retransmission

The RSBT protocol is based on Selective Repeat retransmission; it also employs pre-emptive retransmission to some degree, depending on the acknowledgment method used. Selective Repeat is the most efficient retransmission technique across a satellite link, and is an obvious choice for broadcast transmission. The way in which Selective Repeat retransmission is used by RSBT was motivated by a desire to reduce the effect of the satellite propagation delay.

Numerous research results in the literature, [18],[37] for example, have compared Selective Repeat against Go-back-N retransmission, and concluded that Selective Repeat is far superior across satellite links. The reason for this is simple: the amount of data in the stream is very large, and thus, Go-back-N retransmits large volumes of data for no reason. This has a significant affect on the TCP performance when error rates are high, and renders the protocol basically unusable when the BER is greater than 10^{-5} [25]. Thus, if the use of Go-back-N retransmission can create problems for point-to-point satellite transmission, then obviously it can not be used for broadcast/multicast transmission. Figure 3.15 shows an exaggerated example of a RSBT transmission, highlighting the inefficiency of Go-back-N retransmission.

Although many protocols use Selective Repeat retransmission, RSBT differs to other protocols in that retransmissions do not have precedence; frames requiring retransmission are placed at the back of the queue. RSBT re-orders frames in the retransmission queue as required. NAKs may be received out of order due to the varying delays of individual return links, and because of the delay in the transmission of NAKs if the frame header was not received intact. Retransmissions are not given precedence because it would seem inappropriate to allow receivers with bad reception to delay receivers with good reception; the senders transmission time would not be affected either way.

The acknowledgement methods 1, 3 and 4 mentioned in section 8.5.4, allow some pre-emptive retransmission which further improves the RSBT performance. Moreover, RSBT has the option of full pre-emptive retransmission, regardless of the acknowledgement method, for small file transfers. Full pre-emptive retransmission means that the retransmission of frames is carried out as soon as the sender has sent the frames once; very small files could be sent several times before and acknowledgements have had the time to arrive. Full pre-emptive retransmission would greatly improve the RSBT performance with small file transfers, which are generally very inefficient across a satellite link [24].

8.6 RSBT Specification and Verification

Due to the complex nature of todays communication protocols, Formal Description Techniques (FDTs) are used to ensure that protocol specifications are unambiguous, clear, concise, complete, consistent, tractable, and conformed to. There are three main FDTs: SDL, Estelle and Lotos. The RSBT protocol was formally specified and verified with SDL, which was an appropriate choice for three reasons. First, SDL was developed specifically for communications systems and is supported by complete formal semantics [43]. Second, SDL has a graphical representation form, which is far easier to learn, use and analyse. And third, a CASE tool based on SDL was locally available.

8.7 Modelling

The RSBT protocol was evaluated by two methods: a mathematical model and a simulation model—the two models were used to validate each other. To create models that are accurate representations of the real system, the two models were developed in accordance to a well-defined 10-step procedure [49], which is outlined in section 5.1. The assumptions that were made to simplify the modelling process were the same for both the mathematical model and the simulation model (see sections 4.2 and 5.3). More importantly though, the assumptions should not have any significant impact on the accuracy of the performance results, even under extreme conditions such as very high error rate.

8.7.1 The Mathematical Model

The mathematical model was developed mainly to produced the expected values for the total number of frames transmitted and the total transmission time. However, the model consists of many steps, and thus, a detailed breakdown of the transmission was available. The model was implemented in a spreadsheet, which produced instant results. Thus, the frame sizes used by the mathematical model were fine-tuned to obtain accurate optimal solutions.

8.7.2 The Simulation Model

The RSBT protocol was modelled and simulated with the Network II.5 package. Although the simulation results were not exactly the same as the mathematical model results, they were very similar.

The Network II.5 package was suitable for modelling the RSBT protocol because it was developed specifically for evaluating network designs and network applications. The package facilitates detailed specification of processing elements, transfer devices and storage devices; software modules are specified at an abstract level, which was sufficient. The only shortcoming of the Network II.5 package was its inability to number individual frames (*messages*) for acknowledgement purposes. This problem was overcome by using statistical expected values obtained from the mathematical model.

The simulation results differed slightly from the mathematical model results. This is because Network II.5 uses a normal distribution for statistical or probability related issues such as the frame error rate. Therefore, two simulations with the same parameter settings would produce a different number of retransmission and a different transmission time; the mathematical model always produces the same expected value. However, when the two models were formally validated (see section 5.8), the simulation was repeated six times and the average results were very close to the mathematical model results.

8.8 Experimentation Results

The RSBT experiments were carried out as planned in section 7.2—the results (broadcasting) eclipsed all point-to-point satellite transmission results found in the literature. The experiments were based on 15 *Network Size* and *Base Error Rate* combinations—all combinations of three *Network Sizes* and five *Base Error Rates*.

The three *Network Sizes* used by the experiments were 3, 6 and 12 receivers. Larger *Network Sizes* were not used because even with 12 receivers, the

simulations were over demanding on the computers RAM, and some of the Network II.5 imposed limits created difficulties. An example of a Network II.5 limit is the maximum number of concurrent modules (32,000), which limited the number of pending NAKs; the model had to be adjusted to overcome this problem. The mathematical model did not have any problem with the computers RAM, but larger networks would have increased the model's complexity substantially. Furthermore, the experiments were carried out by two methods for validation purposes. Therefore, it was desirable that the mathematical model experiments mirror the simulations.

The five *base error rates* used in the experiments ranged from 0.01 to 0.0001, which is equivalent to a BER range of 2.3×10^{-4} to 2.3×10^{-6}. The expected BER, is in the 10^{-4} to 10^{-8} range [25] (see section 7.2). Thus, the RSBT experiments were based on high to worse-than-expected error rates. The selection of the *base error rate* range was motivated by the view that it is more important to know how the protocol will perform under bad conditions, rather than good conditions. Furthermore, all experiments with the lowest *base error rate*—0.0001—resulted in transmission efficiencies greater than 92%. Therefore, reducing the error rate further would not improve the performance results significantly.

The experimentation results were analysed in four categories: optimal frame size, throughput, satellite link utilisation and return traffic. The following four sections discuss the results in terms of these categories.

8.8.1 Optimal Frame Size

The frame size can have a significant impact on a protocol's performance. If the frame size is too small, then the overheads are high, and if the frames are too large, then there are too many retransmissions. Most protocols suffer from the former. RSBT does not restrict the frame size in any way, and thus, the optimal frame size may to used to get the best possible performance.

The experimentation results, which are summarised in tables 7.1 and 7.2, show that RSBT frames may be quite large: 7,936 bits for the 3 receivers and 0.0001 base error rate combination. The optimal frame size was reduced to 512 bits for the 12 receivers and 0.01 base error rate combination, which is an extreme case. Recall from section 7.2 that the expected base error rate range is 4.3×10^{-3} to 4.3×10^{-7}. Therefore, for small networks, the optimal frame size will generally much larger than the 7,936 bits mentioned above.

The RSBT frame size must be selected before transmission begins—this selection would be based on historical error rate data. If the error rate for a given satellite link is generally high the frame size should be selected meticulously to ensure that it is close to the optimal size. Conversely, if the error rate is generally low—and the optimal frame size is very large—the frame size may be selected by less accurate methods (quicker methods). Moreover, it would not be a bad idea to use a smaller than optimal frame size for links with low error rates. The addition overheads (header to data ratio) would of course reduce the efficiency. However, if the selected frame size is 4 Kbits or higher, the overheads are very small, and thus, the performance loss is not great. Furthermore, the effects of the greater overheads are negated somewhat by the lower frame error rate of the smaller frames. Further still, if the error rate suddenly rises, the smaller frame size reduces the impact on throughput.

8.8.2 Throughput

The RSBT protocol was designed specifically for satellite communication. The benefits of tailoring RSBT for a specific environment are evident in the performance results. By avoiding the shortcomings of other protocols that are used for satellite transmission, the RSBT broadcasting performance surpasses the point-to-point transmission performances of all protocols found in the literature.

The RSBT throughput for the worst case—12 receivers and a 0.01 base error rate—was 1.089 Mbps. While for the best case—3 receivers and a 0.0001 base error rate—the throughput was 1.958 Mbps (see table 7.1). Although all of

the results found in the literature were based on point-to-point transmission and error rates lower than the lowest used in the RSBT experiments, they were generally far inferior to the RSBT results. Experiments in [28] produced a throughput of 1.21 Mbps, which was closer to RSBT's performance than any other results found in the literature. These experiments split the TCP transmission over a satellite link into three parts. TCP was used on both the sending LAN and receiving LAN. While the transmission over the satellite link was carried out by a link layer protocol, which shielded TCP from the satellite link. TCP, when used over a satellite link, has a peak throughput of about 300 Kbps [24],[25].

The RSBT performance is far superior to that of other protocols because RSBT was tailored for a specific environment, and thus avoided features that inhibit the performance of generic protocols over a satellite link. Briefly, the three main reasons for RSBT's high throughput rates are:

- RSBT does not use any flow control. This is appropriate because intermediate nodes were not considered, and because receivers should not have any problems with data arriving at 2 Mbps. The poor performance of TCP over a satellite link is attributed mainly to the flow control window. Hence, much of the research on improving the TCP performance over a satellite link revolves around expanding the window size [24],[25],[32],[60]. Flow control must be used in most networks, and so most protocols incorporate it. However, some communication configurations are suitable for continuous transmission, and thus, some protocols (e.g. SNR [15]) optionally transmit without any flow control.
- The retransmission is based on Selective Repeat, which is far superior to Go-back-N over a satellite link [25],[28],[37].
- RSBT frame sizes are not restricted. Therefore, the frame overheads can be minimised by using optimal frame sizes. The frame overheads obviously approach insignificance once the frames get larger than about 4 Kbits.

8.8.3 Satellite Link Utilisation

Transmission efficiency is the measure of throughput against the link capacity—high efficiency is dependent on high link utilisation. The RSBT protocol maximise the link utilisation, with most of the results falling between 97% and 99% (see tables 7.1 and 7.2). RSBT can achieve such high link utilisation because it does not use any flow control. The poor performance of TCP over a satellite link is a legacy of the low link utilisation caused by the flow control procedure; of course, the Go-back-N retransmission also inhibits the performance significantly.

8.8.4 Return Traffic

The use of Reverse Error Control for a broadcast transmission, can easily result in the sender being overwhelmed with acknowledgments—the larger the network, the higher the probability of this happening. Thus, the return traffic was a major consideration in the design of RSBT's acknowledgment protocols.

As shown in table 7.1, the mathematical model experiment with three receivers and a 0.0001 *base error rate* (best case), produced a peak return traffic of only 387 bps. While the experiment with 12 receivers and a 0.01 base error rate (worst case), produced a peak return traffic of 150 Kbps. However, recall from section 7.2 that the highest expected base error rate is 4.3×10^{-3} (10^{-4} BER). The closest *base error rate* used by the experiments was 0.005, which with 12 receivers resulted in a peak return traffic of 76 Kbps. Moreover, if a satellite service such as Omnicast Digital were used, the FEC would reduce the error rate, and thus, the return traffic. Also, note that the return traffic results were based on NAKs representing single frames, but NAKs can in fact carry a list of frame sequence numbers (this has not be discussed before). Therefore, RSBT can serve much larger networks before the return traffic becomes a problem.

For networks that are too large to avoid return traffic bottlenecks, other acknowledgement strategies could be implemented. These strategies have been left for future research.

9. Conclusion and Future Research

9.1 Conclusion

The aim of this research project was to develop a protocol for reliable satellite broadcast transmission—RSBT. The protocol was based on the data dissemination requirements of Australia's Bureau of Meteorology and Optus Communications' Omnicast Digital satellite service. The demand for reliable broadcasting/multicasting has grown substantially over the past few years. Hence, several protocols have been developed and much research has been published recently—particularly since 1997. However, the new protocols have all been designed for transmission across terrestrial links. Therefore, the development of the RSBT protocol was a good research topic from two perspectives: the demand for such protocols; and the lack of research regarding reliable multicasting over satellite links—while satellites are ideal for point-to-multipoint transmission.

Most protocols are very generic and may be used across most media. However, this portability has a cost in efficiency. Thus, although no literature on reliable multicasting/broadcasting was found (in the initial stages of this research project), information on several protocols for point-to-point communication was invaluable. That is, protocol features that inhibit good performance over a satellite link, and features that enhance performance were identified.

The RSBT design had two main goals: to transmit data across a satellite link efficiently, and to deliver data reliably to multiple receivers. Thus, the ideas gained from the literature review on efficient transmission over a satellite link were augmented by features required for reliable point-to-multipoint transmission. The acknowledgment process received special consideration because of the return traffic significance in point-to-multipoint communication. Four acknowledgment methods were developed to cater for various communication demands.

The RSBT protocol procedures were formally specified and verified with SDL. SDL is an FDT developed specifically for communications systems and is supported by complete formal semantics. SDL is commonly used in the telecommunications industry and is supported by many CASE tools. Thus, the use of SDL was most appropriate.

The evaluation of the RSBT protocol was carried out by two methods: a mathematical model and a simulation model. The mathematical model provided expected values for the transmission volume—including retransmission—and the total transmission time. The RSBT protocol operation is reasonably simple and so the mathematical model did not require any significant assumptions. Therefore, the mathematical model is reasonably straightforward and does not suffer under extreme parameter values.

The simulation model was developed so that the RSBT performance could be evaluated by a second method. Although the simulation model used a few statistical values that were derived by the mathematical model, the two models are intrinsically different. Thus, the two models creditably validated each other's results. The Network II.5 simulation tool employs an element of randomness and so the results are different for each simulation. Therefore, the mathematical model provides more accurate (consistent) results. However, the simulations provided more confidence in the results because the simulation tool facilitates very detailed hardware specification; the software components of the simulation models were small, and thus easily traced (checked).

The experimentation results showed that RSBT performs substantially better than any other protocol (found in the literature) over a satellite link. RSBT's performance is even more noteworthy because it was compared against point-to-point performances of other protocols—no reliable satellite broadcasting experiments were found in the literature. RSBT's good performance is attributed to three main features: the continuous transmission (no flow control); the Selective Repeat retransmission; and the use of optimal frame sizes.

In conclusion, not only did this research project successfully develop the RSBT protocol, it developed a protocol that is extremely efficient—highlighting the benefits of tailoring RSBT specifically for satellite transmission.

9.2 Future Research

The RSBT protocol has been designed, verified and evaluated—with very good results. However, because RSBT was designed for a specific environment, and because of the time limitation, some issues have been left for future research. These include:

- **Protocol conversion**. The FTP to RSBT conversion—and back to FTP—was described in section 3.3.3. However, further research should be carried out to explore possible protocol conversion alternatives and conversion from other protocols used on LANs
- **Transmission of multiple files**. The satellite propagation delay renders small file transmissions inefficient, particularly if the files are smaller than 100 Kbytes (see section 7.3.2). This inefficiency could cause major delays in scenarios where many small files are queued for transmission. Therefore, future research should include investigation into possible methods of sending several files (bundled) as a single transmission.
- **Multicasting**. The RSBT protocol was developed for reliable broadcasting, and although the *File ID* (in the frame headers) facilitates multicasting, it is limited. Therefore, if RSBT is to cater for complex multicasting scenarios, particularly if it involves receivers from other organisations, then several issues must be addressed. For example, is the *File ID* sufficient (or appropriate); how will error recovery be handled if there are many receivers; how will a transmission session begin; etc. However, it may be better not to expand the RSBT capabilities, because it may reduce the protocol's efficiency!

- **Operation on large networks**. The experiments that were carried out in this research project involved a maximum of 12 receivers. Although the performance results were very good, they are incomplete for large organisations such as shopping chains or car dealers. Therefore, further research is required on broadcasting to a large number of receivers. Three specific topics for further research in this area are:
  - **Network size limitations**. This would involve RSBT experiments with large networks to determine the number of receivers that the RSBT protocol can serve under various error rates. The throughput and the sender's processing requirements may be two metrics, but certainly, the most important measure would be the volume of return traffic.
  - **Group acknowledgments**. The main problem with large networks is that the sender may be overwhelmed with return traffic—not to mention congestion problems on the return links. The return traffic problem could be largely alleviated if receivers were to send acknowledgments as groups. That is, all receivers in a group would send acknowledgments to a designated member, which would send the acknowledgments to the sender without any duplication.
  - **Proxy-senders**. This would be an extension to the group acknowledgment proposal (above) in that the designated member of each group would be a *proxy-sender*. That is, these receivers would not only be responsible for the group's acknowledgments, but would also perform the required retransmissions within the group. The *proxy-senders* would send NAKs as normal for frames that they did not receive, and these frames would be retransmitted over the satellite link. Thus, receivers could get their retransmissions either across the satellite link or from the *proxy-sender*.

REFERENCES

[1] Australian Bureau Of Meteorology, *Annual Report*, 1992-93.

[2] Australian Bureau of Meteorology, *Report on Initial Point to Multi-Point Transmission Requirements For Operational Data To Regional Forecast Offices,* 1993.

[3] Optus Communications (Australia), *Omnicast Seminar Information Book,* 1993.

[4] P. J. Kuhn, Director, Institute of Communications Switching and Data Technics, University of Stuttgart, Germany, *Personal Communication*, December 1994.

[5] N. Aylott, *Multipoint Services—When, Where, and How,* Australian Telecommunication Networks & Applications Conference, pp. 585-590, Melbourne, December 1994.

[6] P. Komisarczuk, M. H. Hadjitheodosiou, F. Coakley, T. Jeans and C. Smythe, *End-to-end protocol performance in ATM/satellite interconnected LANs*, Second International Conference on Broadband Services, Systems and Networks, pp. 87-91, Brighton, 1993.

[7] D. D. Clark, L. M. Lambert and L. Zhang, *NETBLT: A Bulk Data Transfer Protocol*, Internet RFC 998, 1987.

[8] S. Irani and P. Takats, *A Network Management Architecture For Satellite Based Personal Communications*, 1st International Conference on Universal Personal Communication—ICUPC'92, pp. 395-401, Dallas, September 1992.

[9] S. Deering, *Host Extensions for IP Multicasting,* Internet RFC 1112, 1989.

[10] J. Sadubin, *Use of Packet Switching in Canada,* Presented to the World Meteorological Organisation: CBS/WG-TEL/Study Group On Communication Techniques and Protocols, Fifth Session, Geneva, November 1993.

[11] J. Fortin (Head Graphics, Communication) and A. Gagne (Analyst, National Network Management), Canadian Meteorological Centre, *Personal Communication*, 1994-95.

[12] D. D. Clark, L. M. Lambert and L. Zhang, *NETBLT: A Bulk Data Transfer Protocol*, Internet RFC 969, 1985.

[13] A. Netravali, W. Roome and K. Sabnani, *Design and Implementation of a High Speed Transport Protocol,* IEEE Transactions on Communication, Vol. 38, November 1990.

[14] G. M. Lundy and R. C. McArthur, *Formal Modal of a High Speed Transport Protocol,* Protocol Specification, Testing and Verification XII, North-Holland, The Netherlands, 1992.

[15] G. M. Lundy and H. A. Tipici, *Specification and Analysis of the SNR High Speed Transport Protocol,* IEEE/ACM Transactions on Networking, Vol. 2, No. 5, pp. 483-496, October 1994,.

[16] D. R. Cheriton, *VMTP: A Transport Protocol for the Next Generation of Communication Systems,* SIGCOMM '86 Symp., Vol. 16, No. 3, 1986.

[17] G. Chesson, *Xpress Transfer Protocol (XTP)—A Tutorial,* Computer Communications Review, Vol. 20, No. 5, October 1990.

[18] W. T. Strayer, B. J. Dempsey and A. C. Weaver, *XTP: The Xpress Transfer Protocol,* Addison Wesley, New York, 1992.

[19] L. Morgan, *Novell's Burst Mode Protocol Software Provides a High-performance LAN/WAN Solution,* Application Note, Telecommunications, Vol. 27, No. 4, pp. 55, April 1993.

[20] R. Thomas, *Burst Mode Boosts Netware's Wide-area Acumen*, Data Communications International, pp. 65-69, September 1992.

[21] G. Fairhurst, *Hand shaking by satellite*, 16th Simula Users Conference, pp. 51-59, Innsburg, 1988.

[22] J. Postel, *Transmission Control Protocol (TCP),* Internet RFC 793, 1981.

[23] J. D. Spragins, *Telecommunications Protocols and Design*, Addison Wesley, 1994.

[24] H. Kruse, *Performance of common data communications protocols over long delay links; an experimental examination*, International Conference on Communication Systems, Nashville, 1995.

[25] G. Fairhurst, *An analysis of TCP/IP in the CODE network*, Third European Conference on Satellite Communication (ECSC-3), pp. 16-20, Manchester, 1993.

[26] D. Clark, V. Jacobson, J. Romkey and H. Salwen, *An Analysis of TCP Processing Overhead*, IEEE Communications Magazine, June 1989.

[27] A. DeSimone, M. C. Chuah and O. C. Yue, *Throughput Performance of Transport-Layer Protocols Over Wireless LANs,* Globecom'93, December 1993.

[28] H. Balakrishnan, V. N. Padmanabhan, S. Seshan and R. H. Katz, *A Comparison of Mechanisms for Improving TCP Performance Over Wireless Links,* Computer Communications Review, ACM SIGCOMM, Vol. 26, No. 4, pp. 256-269, October 1996.

[29] A. Bakre and B. R. Badrinath, *I-TCP: Indirect TCP for Mobile Hosts*, 15th International Conference on Distributed Computing Systems (ICDCS), May 1995.

[30] H. Balakrishnan, S. Seshan and R. H. Katz, *Improving Reliable Transport and Handoff Performance in Cellular Wireless Networks,* ACM Wireless Networks, Vol. 1, No. 4, December 1995.

[31] R. Yavatkar and N. Bhagwat, *Improving End-to-End Performance of TCP Over Mobile Internetworks,* Mobile'94: Workshop on Mobile Computing Systems and Applications, December 1994.

[32] V. Jacobson, R. Braden and D. Borman, *TCP Extensions for High Performance*, Internet RFC 1323, 1992.

[33] C. Huang and P. K. McKinley, *Communication issues in parallel computing across ATM networks*, IEEE Transactions on Software Engineering, pp. 73-86, Winter 1994.

[34] T. Suzuki, *ATM Adaption Layer Protocol*, IEEE Communications Magazine, Vol. 32, pp. 80-83, April 1994.

[35] W. R. Stevens, *TCP Illustrated, Volume 1*, Addison Wesley, Reading, 1994.

[36] J. H. Holzmann, *Design and Validation of Computer Protocols,* Prentice Hall, 1991.

[37] T. R. Henderson, *Design Principles and Performance Analysis of SSCOP: A New ATM Adaption Layer Protocol*, Computer Communications Review, ACM SIGCOMM, Vol. 25, No. 2, pp. 47-59, April 1995.

[38] J. Postel and J. Reynolds, *File Transfer Protocol (FTP),* Internet RFC 959, 1985.

[39] F. Halsall, *Data Communications, Computer Networks and Open Systems, 3rd Edition*, Addison Wesley, 1993.

[40] D. Russell, *The Principles of Computer Networks*, Cambridge Computer Science Texts 25, Cambridge University Press, 1991.

[41] ISO 4335, *High Level Data Link Control (HDLC) Procedures.*

[42] ITU-T Recommendation Q.2110, *B-ISDN-ATM Adaption Layer—Service Specific Connection Oriented Protocol (SSCOP)*, COM-11-R30, Geneva, 1994.

[43] K. J. Turner, *Using Formal Description Techniques; An Introduction to Estelle, Lotos and SDL,* Wiley, West Sussex, 1993.

[44] ITU-T Recommendation Z.100 Appendices I and II, *Programming Languages: SDL Methodology Guidelines, SDL Bibliography*, 1993.

[45] ITU-T Recommendation Z.120, *Criteria For The Use and Applicability of Formal Description Techniques; Message Sequence Chart (MSC)*, 1993.

[46] ISO/IEC 9074, *Estelle: A Formal Description Technique Based on an Extended State Transition Model*, 1997.

[47] ISO/IEC 8807, *LOTOS—A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour*, 1989.

[48] H. C. Chow and S. S. Lam, *PROSPEC: An Interactive Programming Environment for Designing and Verifying Communication Protocols,* IEEE Transaction on Software Engineering, Vol. 14, No. 3, March 1988.

[49] A. Alan and B. Pritsker, *Introduction to Simulation and SLAM II—Third Edition*, Halsted Press (division of John Wiley and Sons, New York) 1986.

[50] CACI Products Co. *Network II.5 User's Manual—Release 11.0*, 1996.

[51] M. D. Balkovic, H. W. Klanger, S. W. Klare and W. G. McGruther, *High-speed Voiceband Data Transmission Performance on the Switched Telecommunications Network*, The Bell System Technical Journal, Vol. 50, No. 4, pp. 1349-1384, April 1971.

[52] K. Washburn and J. Evans, *TCP/IP Running a Successful Network—2nd Edition*, Addison Wesley, Longman, 1996.

[53] G. J. Armitage, *Multicast and Multi Protocol Support for ATM based Internets*, ACM SIGCOMM, Vol. 25, No. 2, pp. 34-46, April 1995.

[54] J. C. Lin and P. Sanjoy, *RMTP: A Reliable Multicast Transport Protocol*, IEEE INFOCOM'96, San Francisco, March 1996.

[55] S. Floyd, V. Jacobson, S. McCanne, C. Liu and L. Zhang, *A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing*, IEEE/ACM Transactions on Networking, Vol. 5, No. 6, pp. 784-803, December 1997.

[56] M. Hofmann, *Adding Scalability to Transport Level Multicast*, 3rd COST 237 Workshop—Multimedia Telecommunications and Applications, Barcelona, November 1996.

[57] J. Nonnenmacher, E. Biersach and D. Towsley, *Parity-based Loss Recovery for Reliable Multicast Transmission*, ACM SIGCOMM'97, Cannes, September 1997.

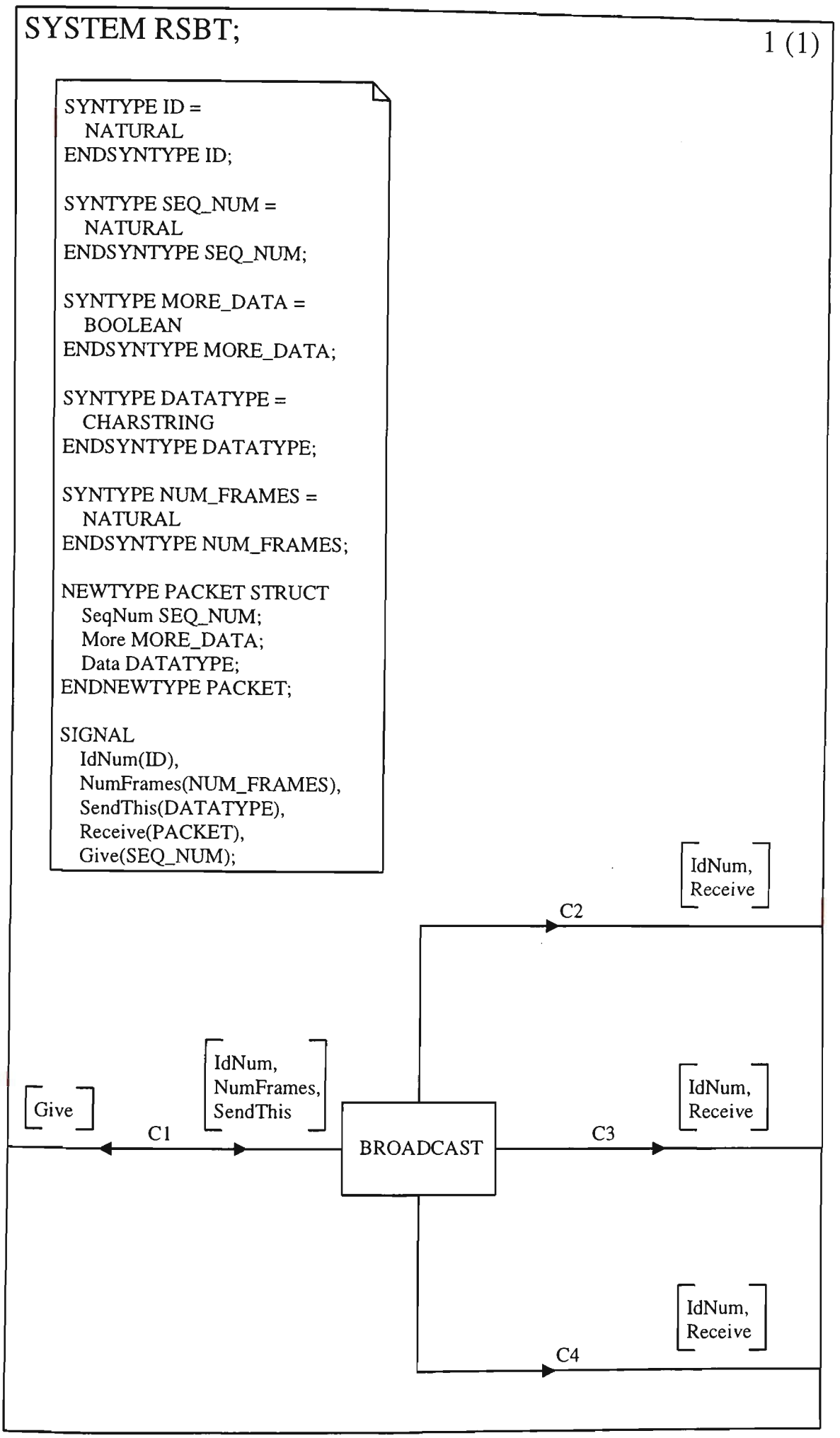
[58] M. T. Lucas, B. J. Dempsey and A. C. Weaver, *MESH: Distributed Error Recovery for Multimedia Streams in Wide-Area Multicast Networks*, IEEE ICC'97, Montreal, June 1997.

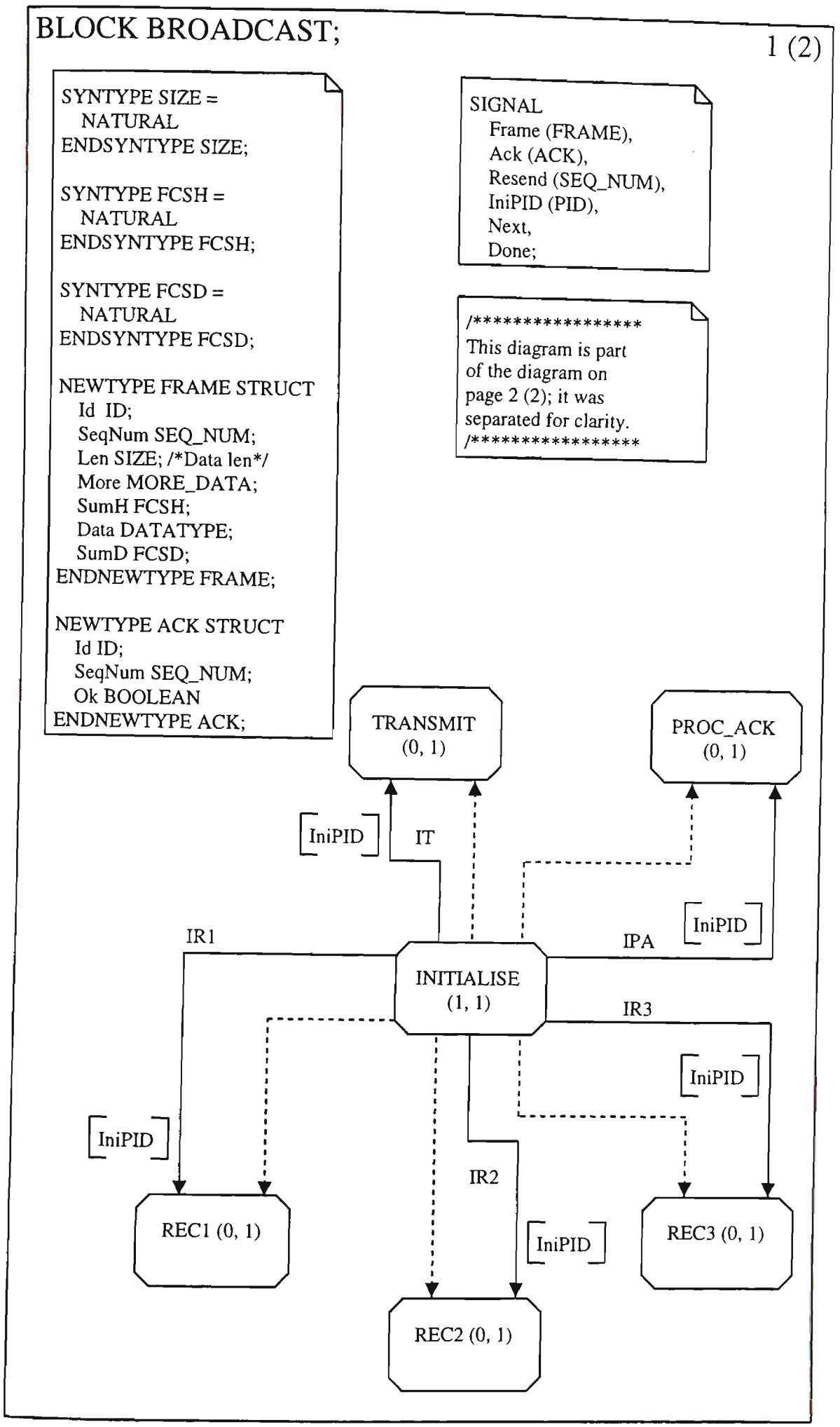
[59] A. Koifman and S. Zabele, *RAMP: A Reliable Adaptive Multicast Protocol*, IEEE INFOCOM'96, San Francisco, March 1996.

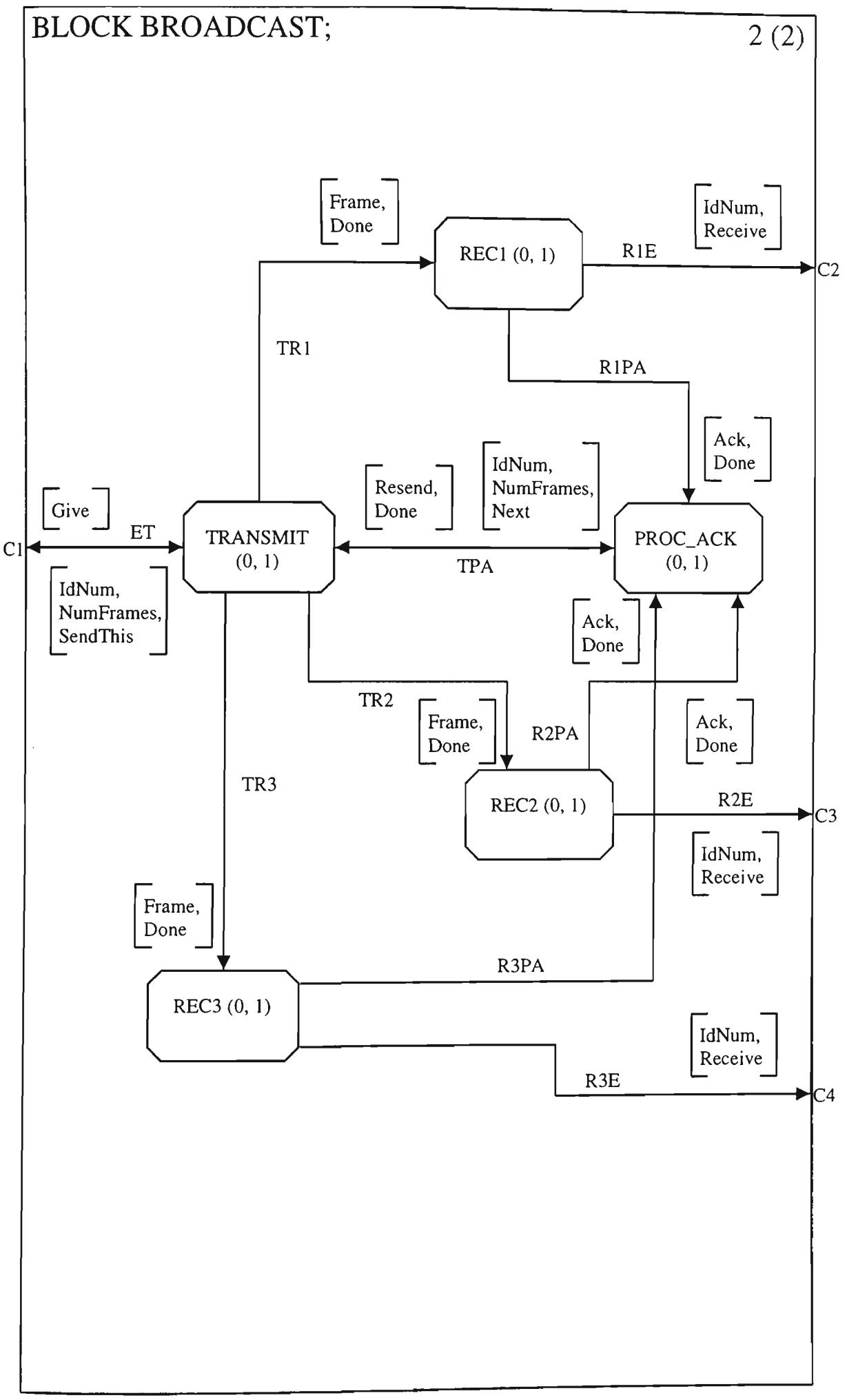
[60] A. L. Olah and M. S. Heemstra de Groot, *Analysis of a Reliable Data-transfer Protocol for Broadband Networks*, Australian Telecommunications Networks & Applications Conference ATNAC'95, pp. 585–590, Sydney, 1995.

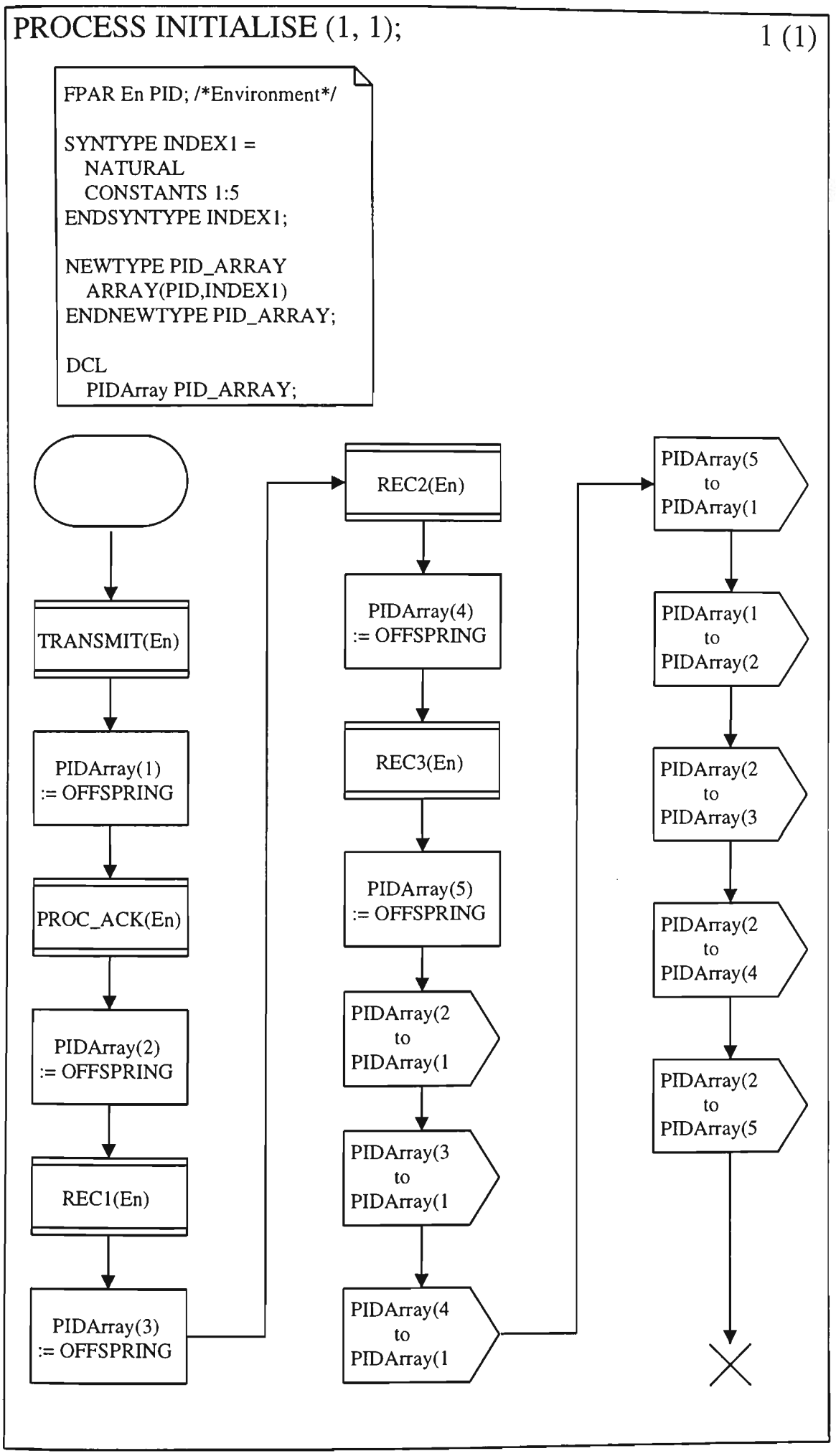
APPENDIX A. SDL SPECIFICATIONS

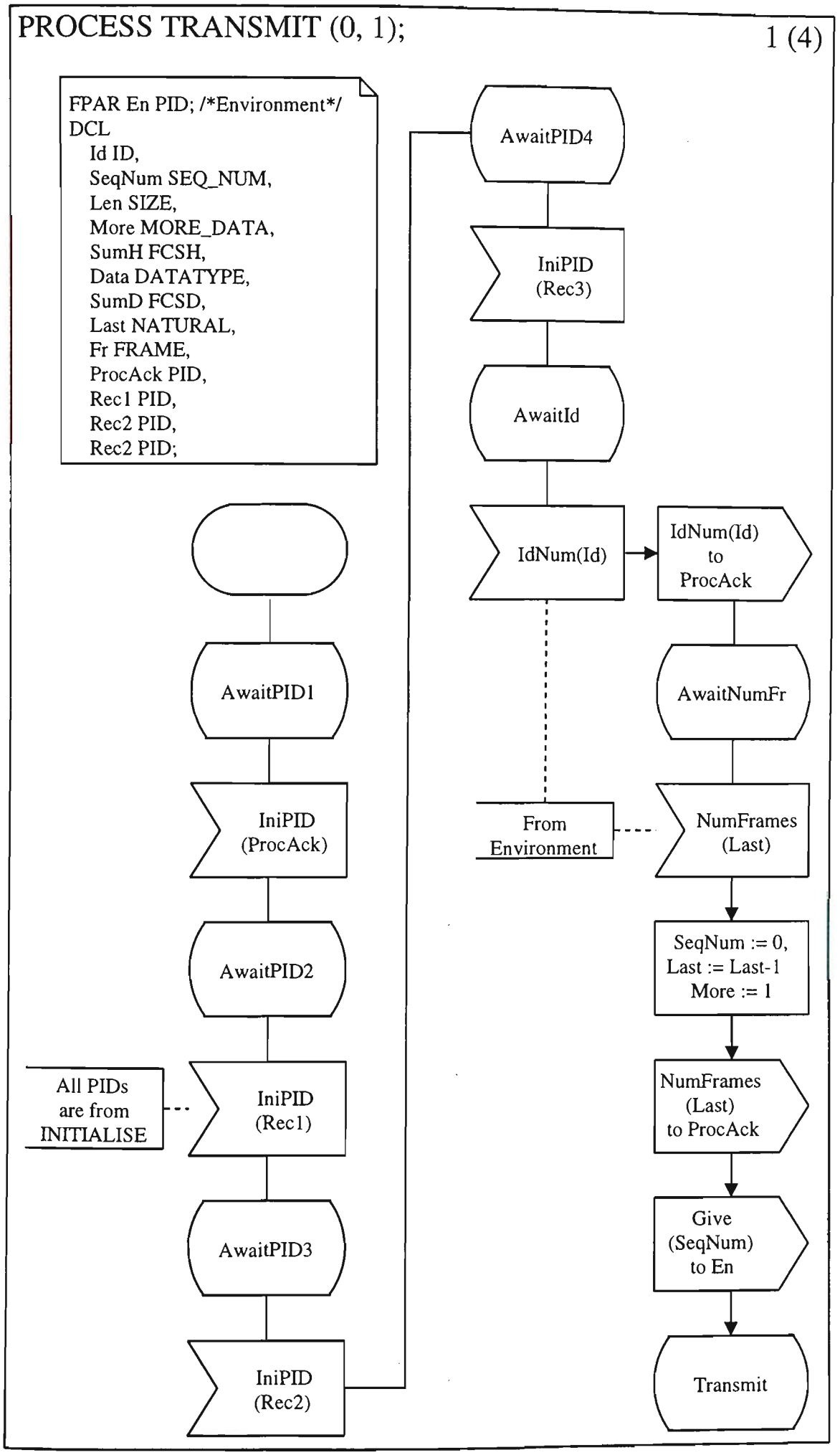
This Appendix presents the SDL specifications for the RSBT protocol. An explanation of the specifications was given in Chapter 4.

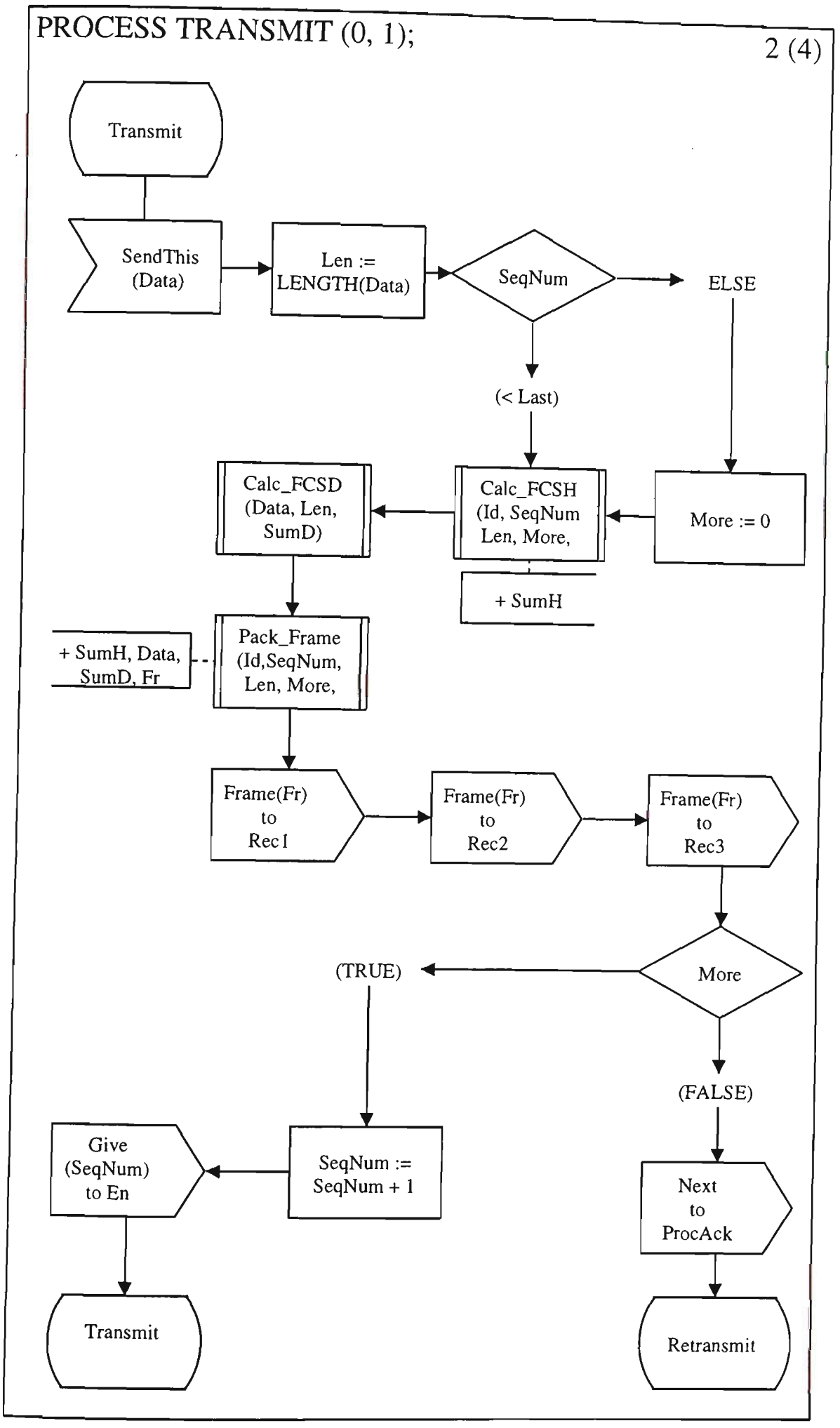

SYSTEM RSBT;
1 (1)
SYNTYPE ID =
NATURAL
ENDSYNTYPE ID;
SYNTYPE SEQ_NUM =
NATURAL
ENDSYNTYPE SEQ_NUM;
SYNTYPE MORE_DATA =
BOOLEAN
ENDSYNTYPE MORE_DATA;
SYNTYPE DATATYPE =
CHARSTRING
ENDSYNTYPE DATATYPE;
SYNTYPE NUM_FRAMES =
NATURAL
ENDSYNTYPE NUM_FRAMES;
NEWTYPE PACKET STRUCT
SeqNum SEQ_NUM;
More MORE_DATA;
Data DATATYPE;
ENDNEWTYPE PACKET;
SIGNAL
IdNum(ID),
NumFrames(NUM_FRAMES),
SendThis(DATATYPE),
Receive(PACKET),
Give(SEQ_NUM);
IdNum,
Receive
C2
IdNum,
NumFrames,
SendThis
Give
C1
BROADCAST
C3
IdNum,
Receive
IdNum,
Receive
C4

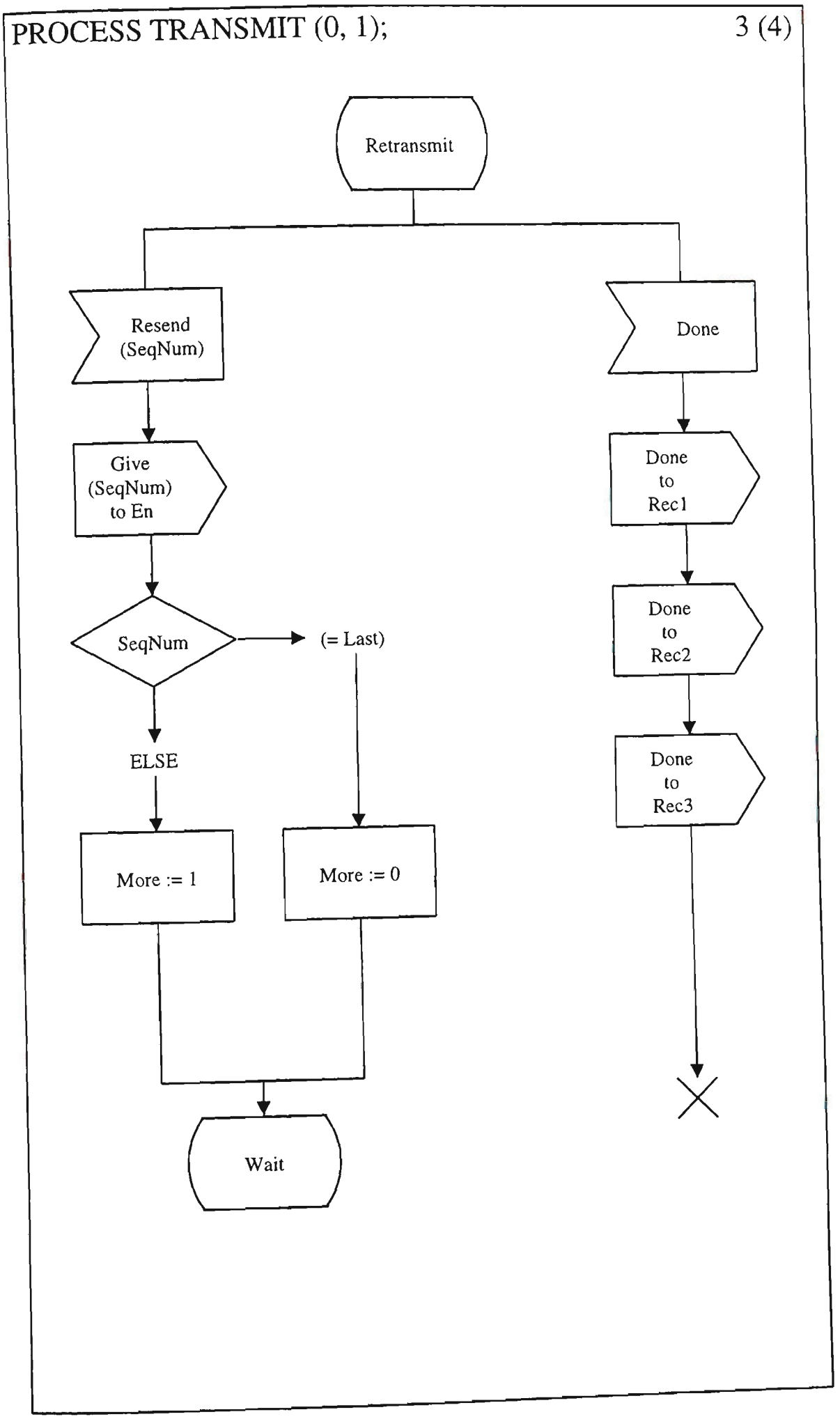

BLOCK BROADCAST;
1 (2)
SYNTYPE SIZE =
NATURAL
ENDSYNTYPE SIZE;
SYNTYPE FCSH =
NATURAL
ENDSYNTYPE FCSH;
SYNTYPE FCSD =
NATURAL
ENDSYNTYPE FCSD;
NEWTYPE FRAME STRUCT
Id ID;
SeqNum SEQ_NUM;
Len SIZE; /*Data len*/
More MORE_DATA;
SumH FCSH;
Data DATATYPE;
SumD FCSD;
ENDNEWTYPE FRAME;
NEWTYPE ACK STRUCT
Id ID;
SeqNum SEQ_NUM;
Ok BOOLEAN
ENDNEWTYPE ACK;
SIGNAL
Frame (FRAME),
Ack (ACK),
Resend (SEQ_NUM),
IniPID (PID),
Next,
Done;
/*****************
This diagram is part
of the diagram on
page 2 (2); it was
separated for clarity.
/*****************
TRANSMIT
(0, 1)
PROC_ACK
(0, 1)
IniPID
IT
IniPID
IR1
IPA
INITIALISE
(1, 1)
IR3
IniPID
IniPID
IR2
REC1 (0, 1)
IniPID
REC3 (0, 1)
REC2 (0, 1)

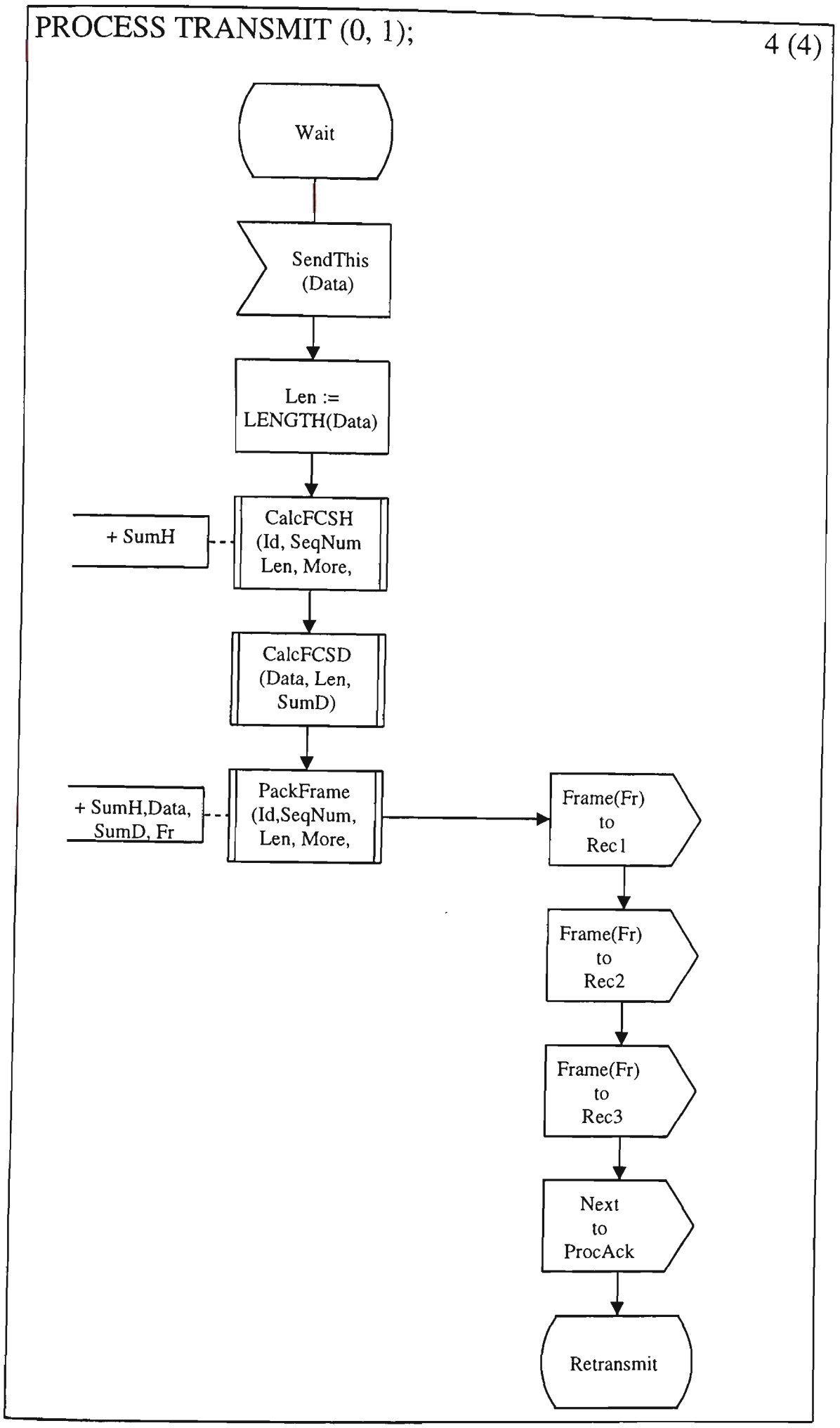

BLOCK BROADCAST;
2 (2)
Frame, Done
REC1 (0, 1)
IdNum, Receive
R1E
C2
TR1
R1PA
Ack, Done
IdNum, NumFrames, Next
Resend, Done
Give
ET
C1
TRANSMIT (0, 1)
TPA
PROC_ACK (0, 1)
IdNum, NumFrames, SendThis
Ack, Done
TR2
Frame, Done
R2PA
Ack, Done
TR3
REC2 (0, 1)
R2E
C3
IdNum, Receive
Frame, Done
R3PA
REC3 (0, 1)
IdNum, Receive
R3E
C4

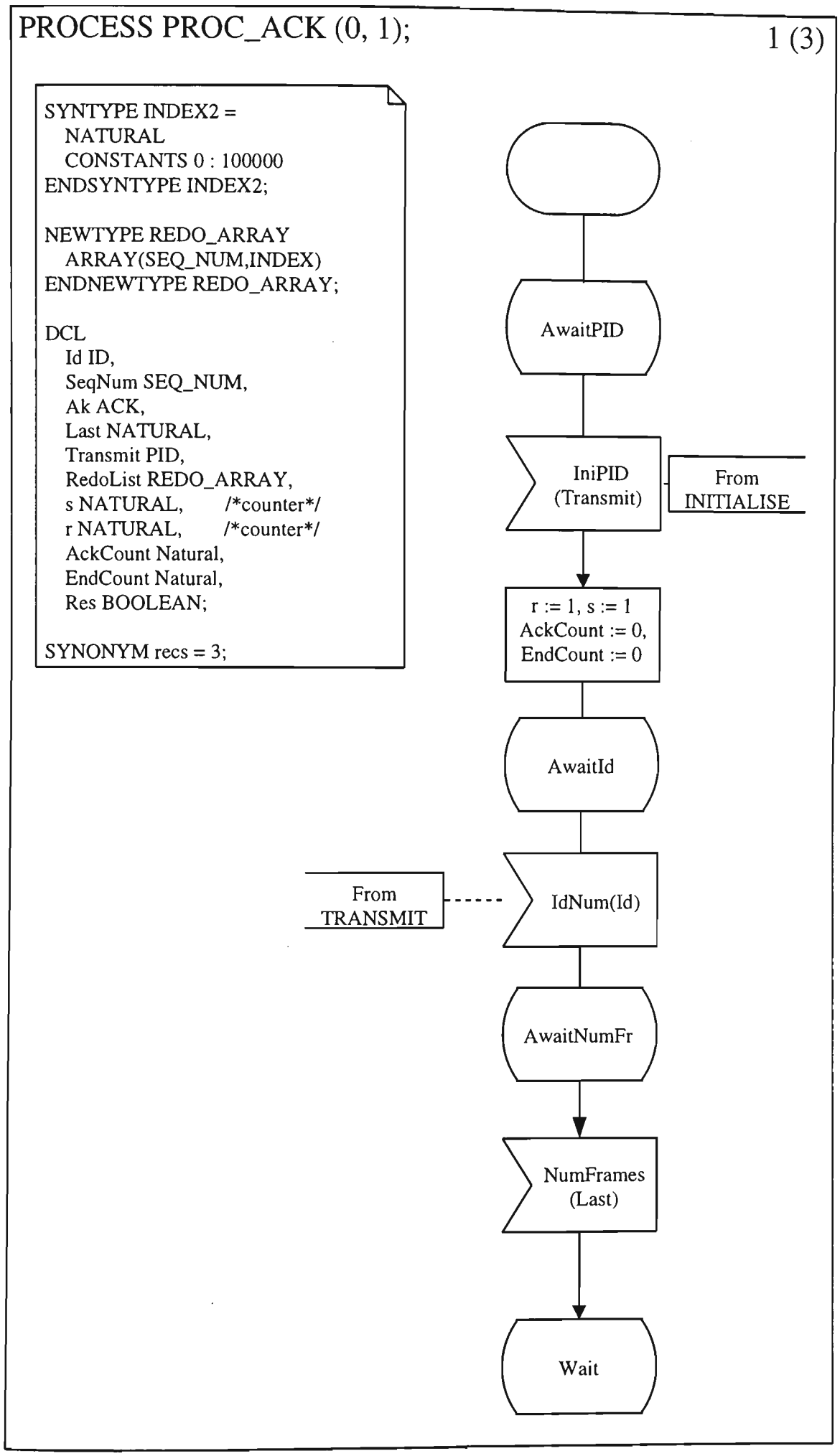

PROCESS INITIALISE (1, 1);
1 (1)
FPAR En PID; /*Environment*/
SYNTYPE INDEX1 =
NATURAL
CONSTANTS 1:5
ENDSYNTYPE INDEX1;
NEWTYPE PID_ARRAY
ARRAY(PID,INDEX1)
ENDNEWTYPE PID_ARRAY;
DCL
PIDArray PID_ARRAY;
TRANSMIT(En)
PIDArray(1) := OFFSPRING
PROC_ACK(En)
PIDArray(2) := OFFSPRING
REC1(En)
PIDArray(3) := OFFSPRING
REC2(En)
PIDArray(4) := OFFSPRING
REC3(En)
PIDArray(5) := OFFSPRING
PIDArray(2 to PIDArray(1
PIDArray(3 to PIDArray(1
PIDArray(4 to PIDArray(1
PIDArray(5 to PIDArray(1
PIDArray(1 to PIDArray(2
PIDArray(2 to PIDArray(3
PIDArray(2 to PIDArray(4
PIDArray(2 to PIDArray(5

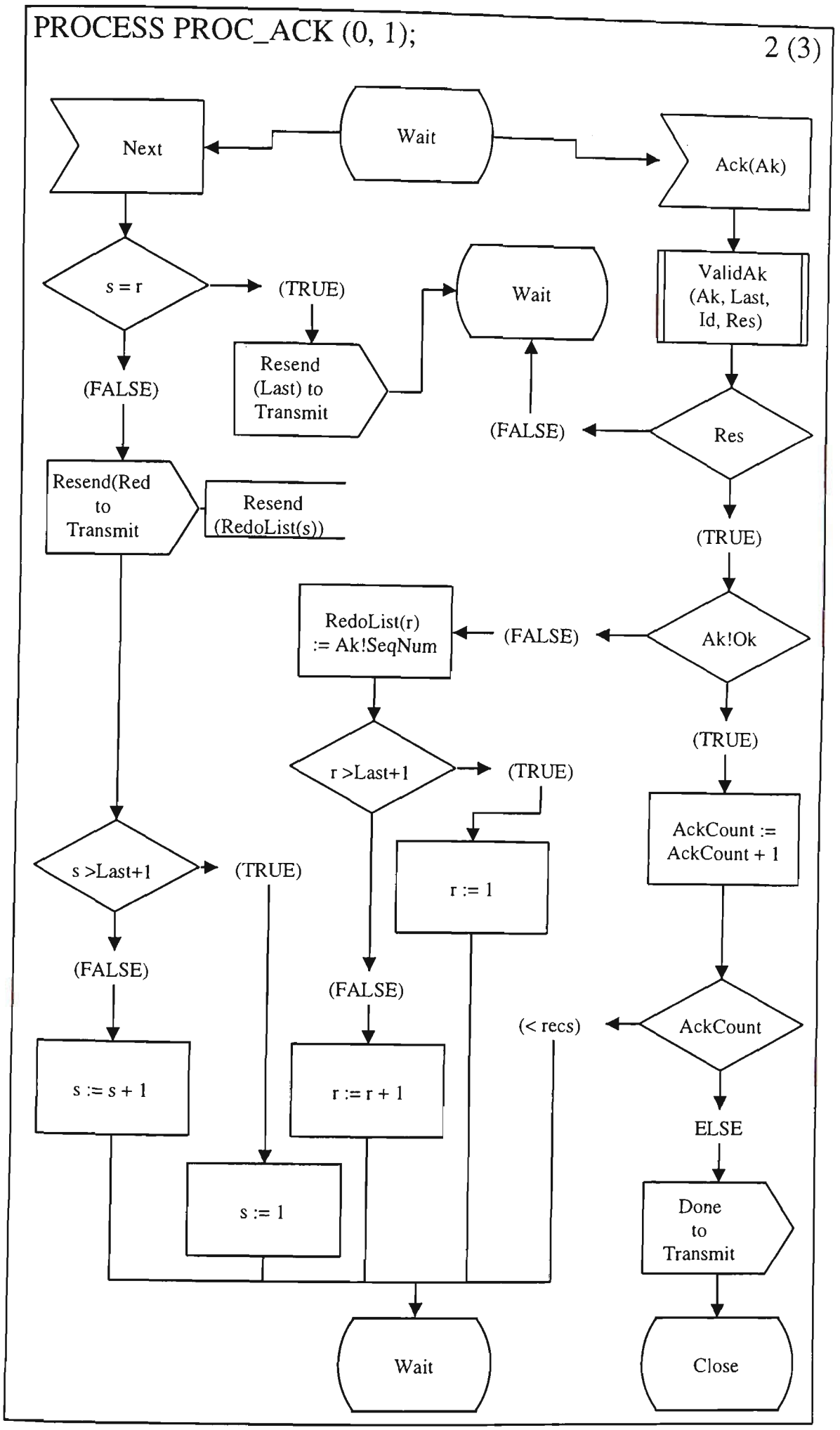

PROCESS TRANSMIT (0, 1);
1 (4)
FPAR En PID; /*Environment*/
DCL
Id ID,
SeqNum SEQ_NUM,
Len SIZE,
More MORE_DATA,
SumH FCSH,
Data DATATYPE,
SumD FCSD,
Last NATURAL,
Fr FRAME,
ProcAck PID,
Rec1 PID,
Rec2 PID,
Rec2 PID;
AwaitPID1
IniPID (ProcAck)
AwaitPID2
All PIDs are from INITIALISE
IniPID (Rec1)
AwaitPID3
IniPID (Rec2)
AwaitPID4
IniPID (Rec3)
AwaitId
IdNum(Id)
IdNum(Id) to ProcAck
AwaitNumFr
From Environment
NumFrames (Last)
SeqNum := 0, Last := Last-1 More := 1
NumFrames (Last) to ProcAck
Give (SeqNum) to En
Transmit

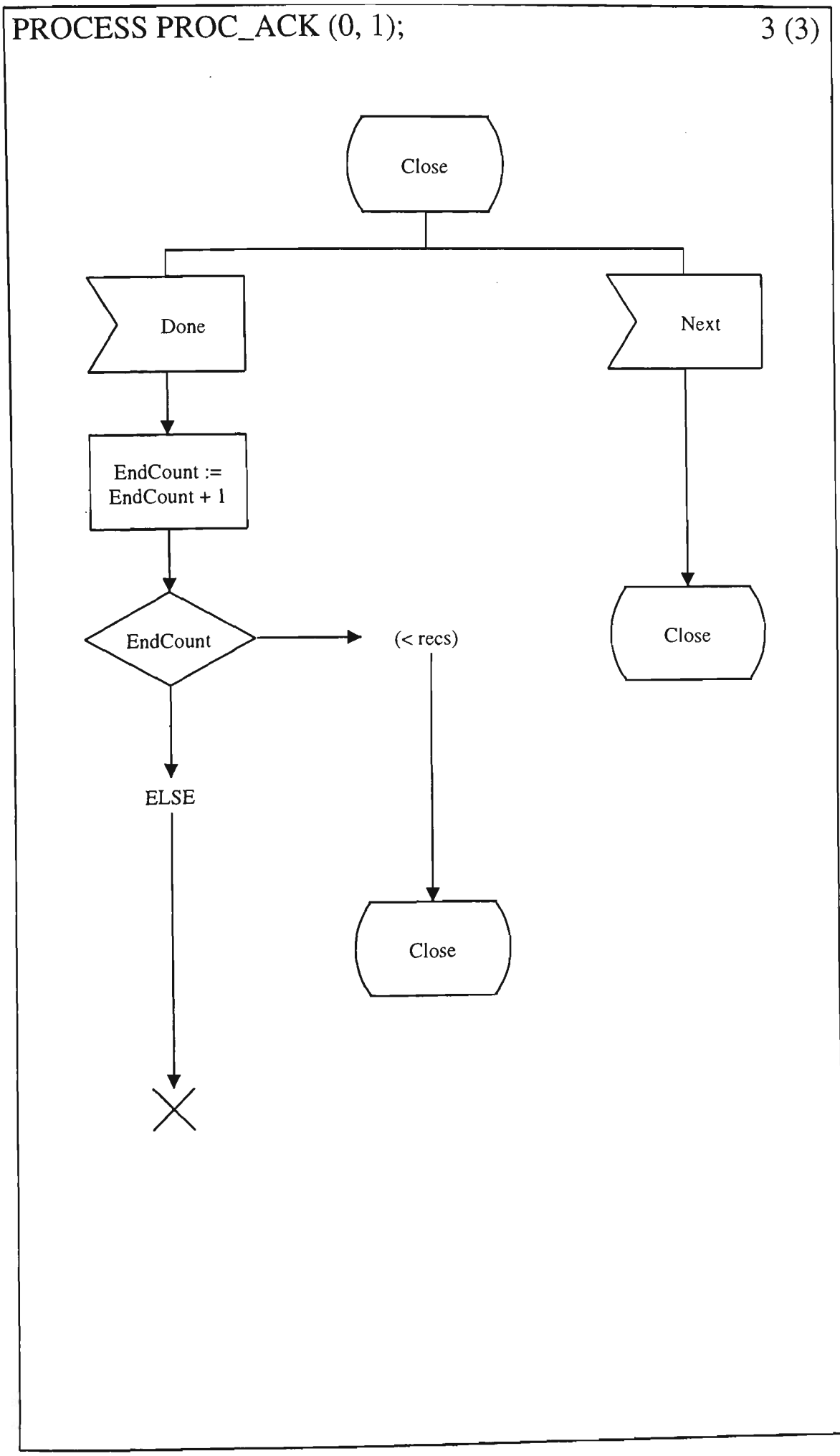

PROCESS TRANSMIT (0, 1);
2 (4)
Transmit
SendThis (Data)
Len := LENGTH(Data)
SeqNum
ELSE
(< Last)
Calc_FCSD (Data, Len, SumD)
Calc_FCSH (Id, SeqNum Len, More,
More := 0
+ SumH
+ SumH, Data, SumD, Fr
Pack_Frame (Id,SeqNum, Len, More,
Frame(Fr) to Rec1
Frame(Fr) to Rec2
Frame(Fr) to Rec3
More
(TRUE)
(FALSE)
Give (SeqNum) to En
SeqNum := SeqNum + 1
Next to ProcAck
Transmit
Retransmit


PROCESS TRANSMIT (0, 1);
3 (4)
Retransmit
Resend (SeqNum)
Done
Give (SeqNum) to En
Done to Rec1
SeqNum
(= Last)
Done to Rec2
ELSE
Done to Rec3
More := 1
More := 0
Wait


PROCESS TRANSMIT (0, 1);
4 (4)
Wait
SendThis (Data)
Len := LENGTH(Data)
+ SumH
CalcFCSH (Id, SeqNum Len, More,
CalcFCSD (Data, Len, SumD)
+ SumH,Data, SumD, Fr
PackFrame (Id,SeqNum, Len, More,
Frame(Fr) to Rec1
Frame(Fr) to Rec2
Frame(Fr) to Rec3
Next to ProcAck
Retransmit

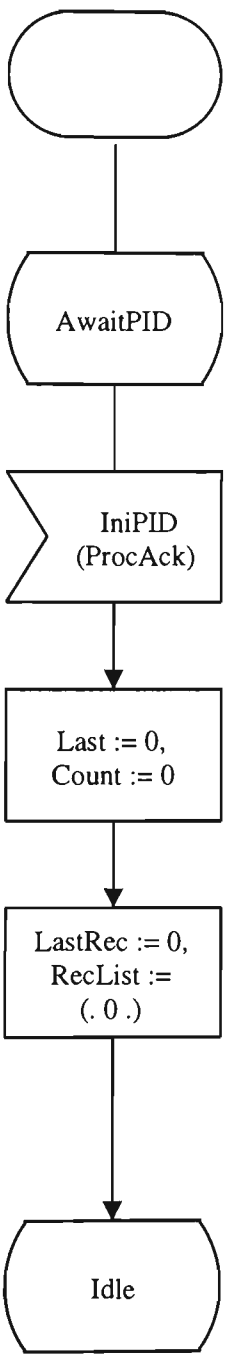

PROCESS PROC_ACK (0, 1);
1 (3)
SYNTYPE INDEX2 =
NATURAL
CONSTANTS 0 : 100000
ENDSYNTYPE INDEX2;
NEWTYPE REDO_ARRAY
ARRAY(SEQ_NUM,INDEX)
ENDNEWTYPE REDO_ARRAY;
DCL
Id ID,
SeqNum SEQ_NUM,
Ak ACK,
Last NATURAL,
Transmit PID,
RedoList REDO_ARRAY,
s NATURAL, /*counter*/
r NATURAL, /*counter*/
AckCount Natural,
EndCount Natural,
Res BOOLEAN;
SYNONYM recs = 3;
AwaitPID
IniPID
(Transmit)
From
INITIALISE
r := 1, s := 1
AckCount := 0,
EndCount := 0
AwaitId
From
TRANSMIT
IdNum(Id)
AwaitNumFr
NumFrames
(Last)
Wait

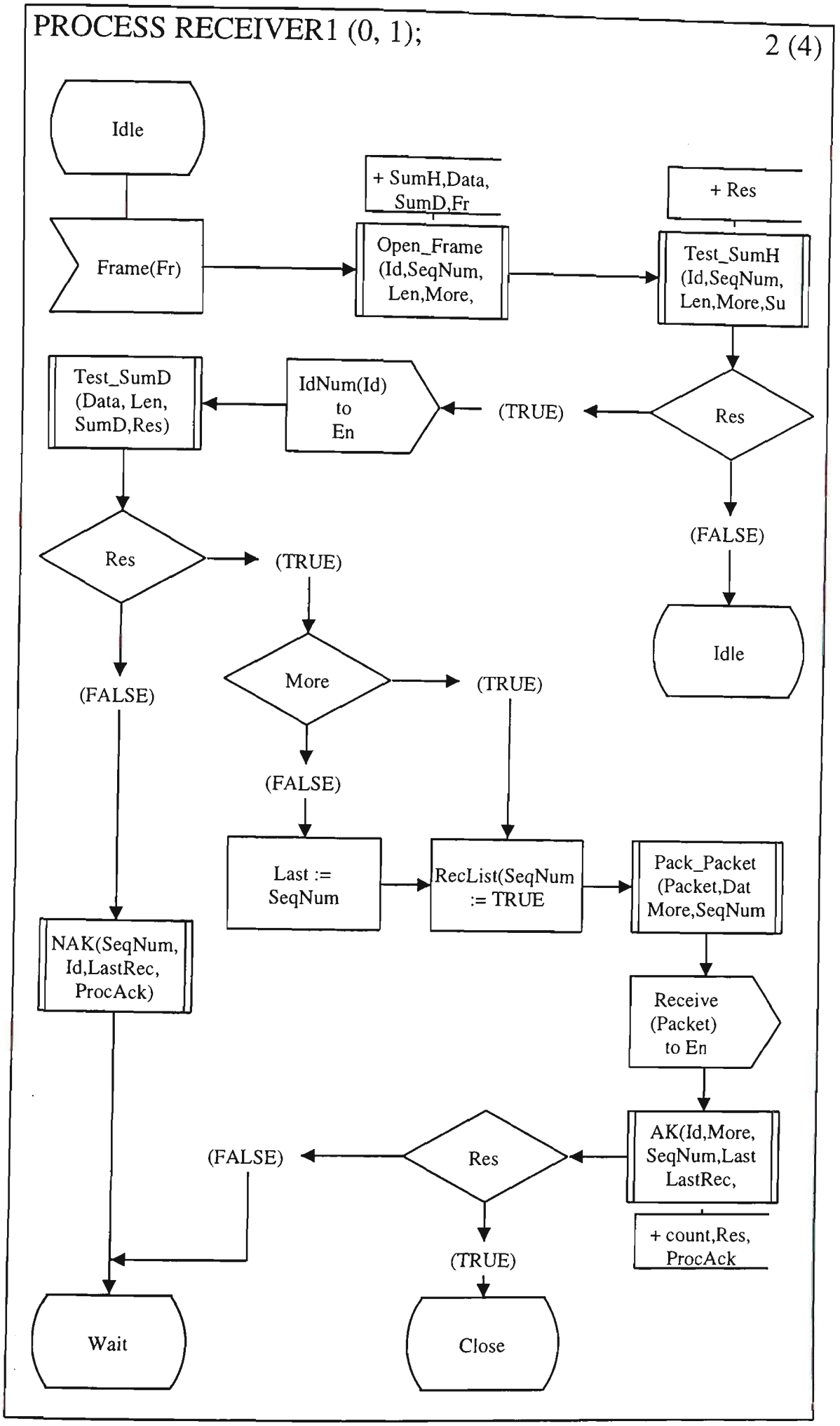

PROCESS PROC_ACK (0, 1);
2 (3)
Next
Wait
Ack(Ak)
s = r
(TRUE)
Wait
ValidAk
(Ak, Last,
Id, Res)
(FALSE)
Resend
(Last) to
Transmit
(FALSE)
Res
Resend(Red
to
Transmit
Resend
(RedoList(s))
(TRUE)
RedoList(r)
:= Ak!SeqNum
(FALSE)
Ak!Ok
r >Last+1
(TRUE)
(TRUE)
AckCount :=
AckCount + 1
s >Last+1
(TRUE)
r := 1
(FALSE)
(FALSE)
(< recs)
AckCount
s := s + 1
r := r + 1
ELSE
s := 1
Done
to
Transmit
Wait
Close

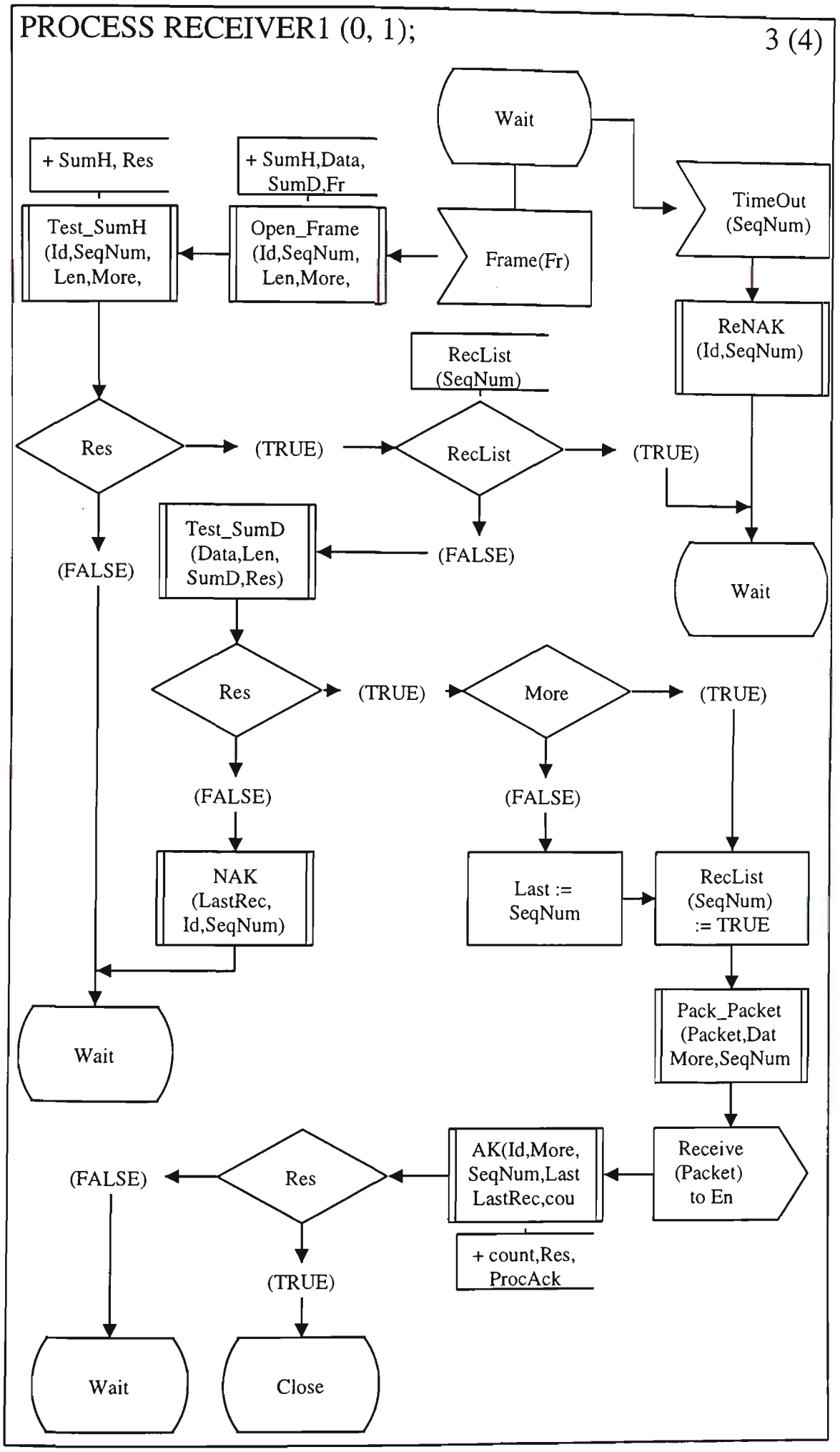

PROCESS PROC_ACK (0, 1);
3 (3)
Close
Done
Next
EndCount :=
EndCount + 1
EndCount
(< recs)
Close
ELSE
Close

PROCESS RECEIVER1 (0, 1); 1 (4)

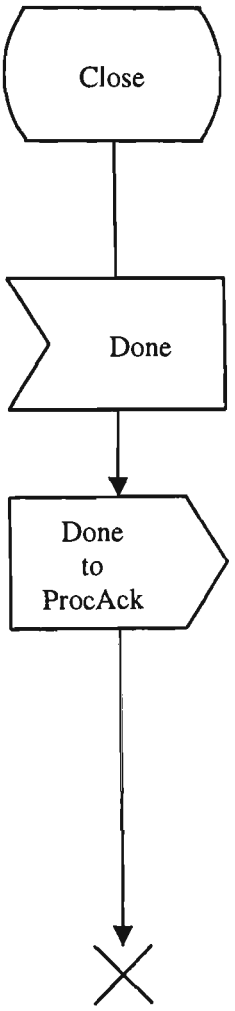
```
FPAR En PID; /*Environment*/

SYNTYPE INDEX3 =
   NATURAL
   CONSTANTS 0 : 100000
ENDSYNTYPE INDEX3;

NEWTYPE REC_ARRAY1
   ARRAY(BOOLEAN,INDEX3)
ENDNEWTYPE REC_ARRAY1;

DCL
   Id ID,
   SeqNum SEQ_NUM,
   Len SIZE,
   More MORE_DATA,
   SumH FCSH,
   Data DATATYPE,
   SumD FCSD,
   Fr FRAME,
   Packet PACKET,
   Last NATURAL,
   LastRec SEQ_NUM,
   Count NATURAL,
   Res BOOLEAN,
   RecList REC_ARRAY1;

TIMER TimeOut(SEQ_NUM);
```

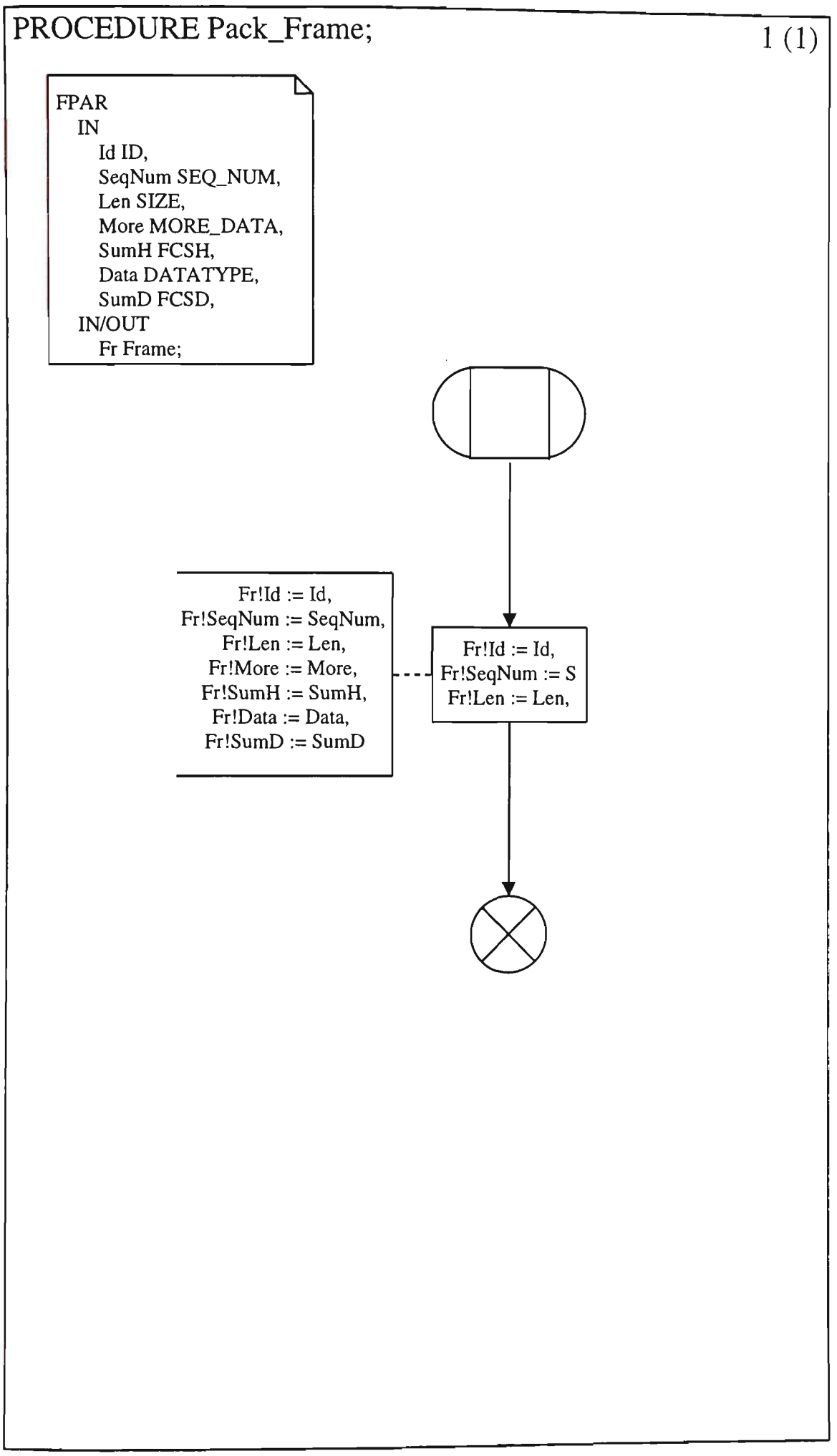


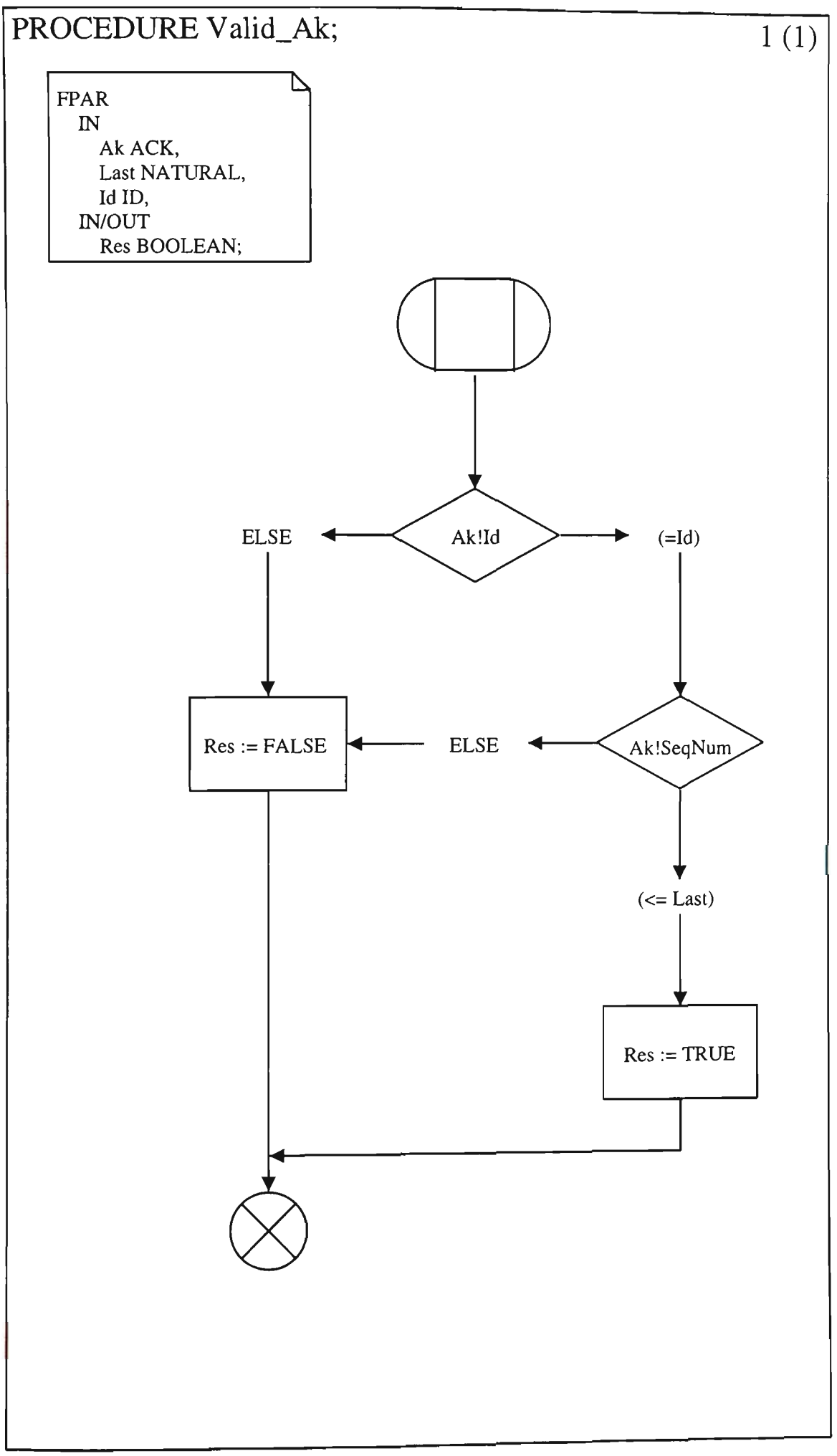

PROCESS RECEIVER1 (0, 1);
2 (4)
Idle
Frame(Fr)
+ SumH,Data, SumD,Fr
Open_Frame (Id,SeqNum, Len,More,
+ Res
Test_SumH (Id,SeqNum, Len,More,Su
Res
(TRUE)
IdNum(Id) to En
Test_SumD (Data, Len, SumD,Res)
Res
(FALSE)
Idle
(TRUE)
(FALSE)
More
(TRUE)
(FALSE)
Last := SeqNum
RecList(SeqNum := TRUE
Pack_Packet (Packet,Dat More,SeqNum
NAK(SeqNum, Id,LastRec, ProcAck)
Receive (Packet) to En
AK(Id,More, SeqNum,Last LastRec,
+ count,Res, ProcAck
Res
(FALSE)
(TRUE)
Wait
Close

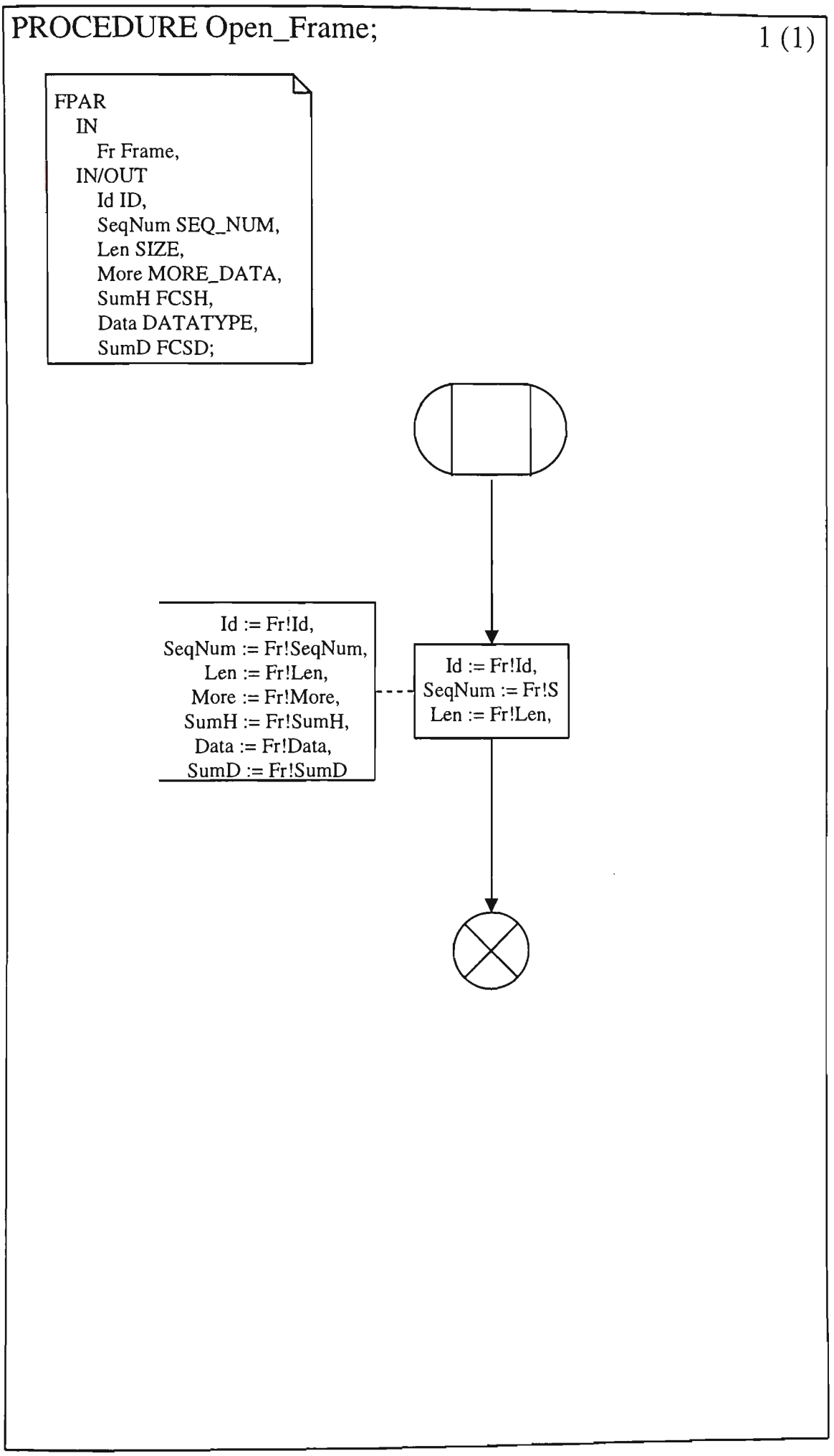

PROCESS RECEIVER1 (0, 1);
3 (4)
Wait
+ SumH, Res
+ SumH,Data, SumD,Fr
TimeOut (SeqNum)
Test_SumH (Id,SeqNum, Len,More,
Open_Frame (Id,SeqNum, Len,More,
Frame(Fr)
ReNAK (Id,SeqNum)
RecList (SeqNum)
Res
(TRUE)
RecList
(TRUE)
(FALSE)
Test_SumD (Data,Len, SumD,Res)
(FALSE)
Wait
Res
(TRUE)
More
(TRUE)
(FALSE)
(FALSE)
NAK (LastRec, Id,SeqNum)
Last := SeqNum
RecList (SeqNum) := TRUE
Pack_Packet (Packet,Dat More,SeqNum
Wait
Receive (Packet) to En
AK(Id,More, SeqNum,Last LastRec,cou
Res
(FALSE)
+ count,Res, ProcAck
(TRUE)
Wait
Close

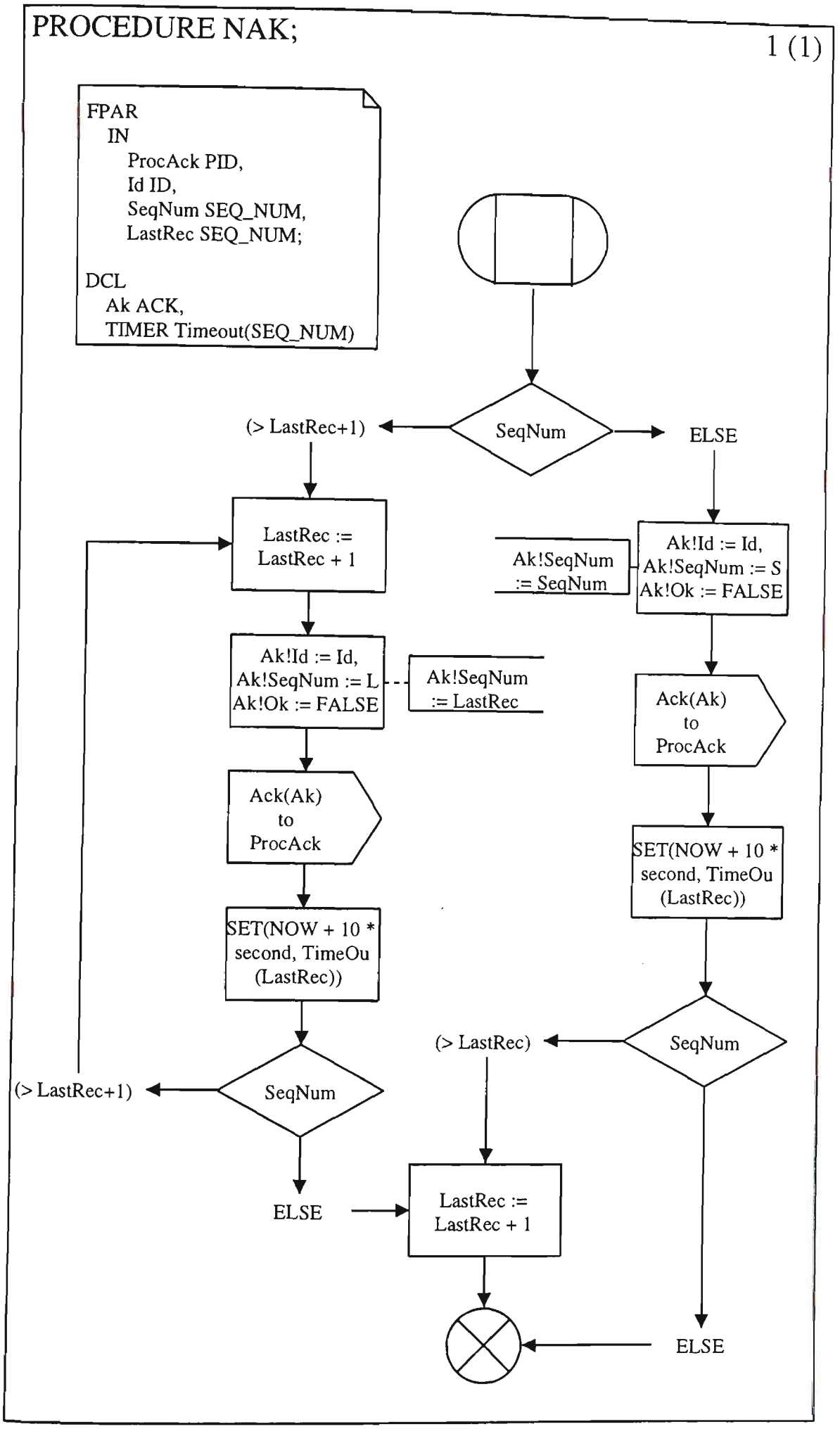
PROCESS RECEIVER1 (0, 1); 4 (4)



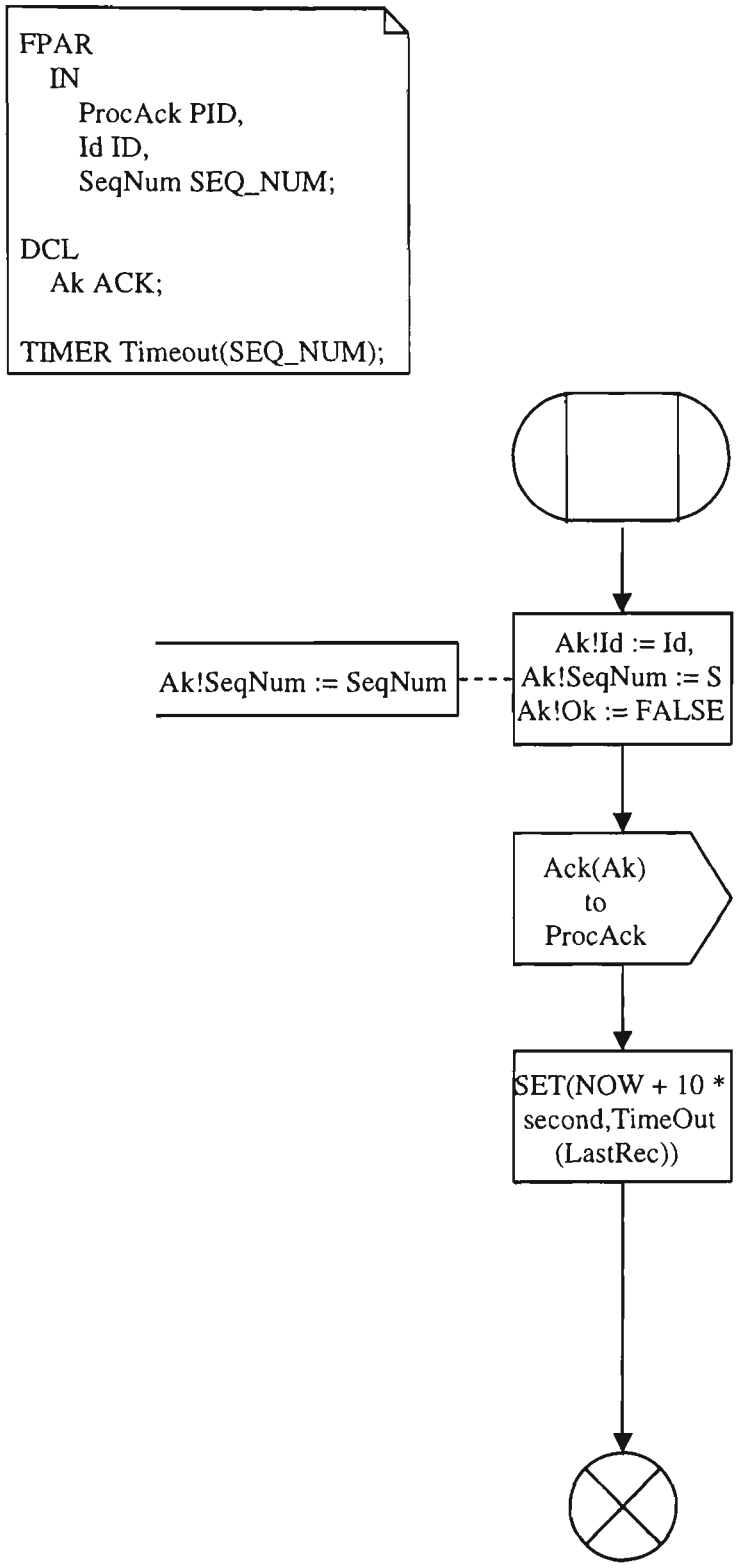

PROCEDURE Pack_Frame;
1 (1)
FPAR
IN
Id ID,
SeqNum SEQ_NUM,
Len SIZE,
More MORE_DATA,
SumH FCSH,
Data DATATYPE,
SumD FCSD,
IN/OUT
Fr Frame;
Fr!Id := Id,
Fr!SeqNum := SeqNum,
Fr!Len := Len,
Fr!More := More,
Fr!SumH := SumH,
Fr!Data := Data,
Fr!SumD := SumD
Fr!Id := Id,
Fr!SeqNum := S
Fr!Len := Len,

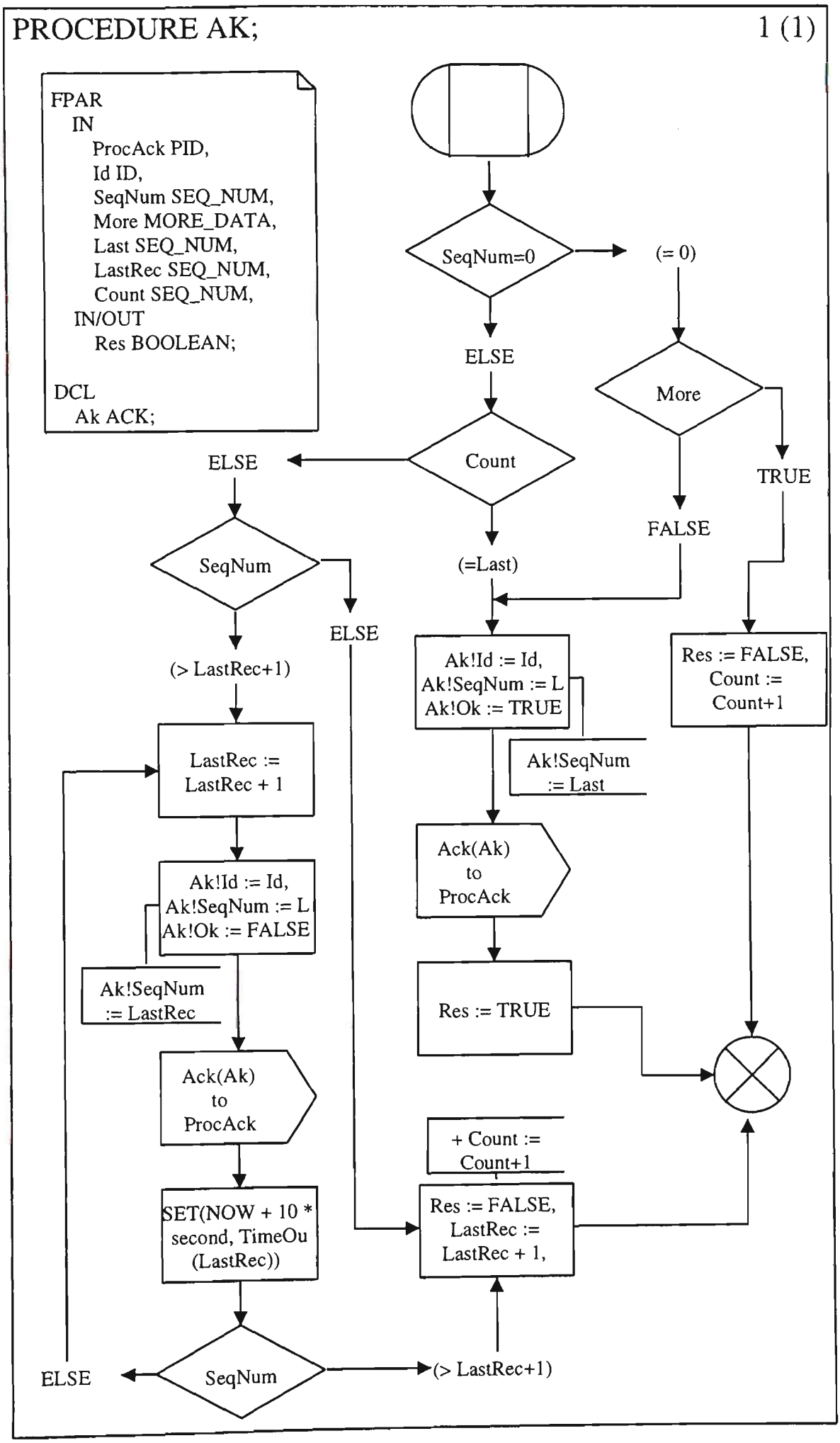

PROCEDURE Valid_Ak;
1 (1)
FPAR
IN
Ak ACK,
Last NATURAL,
Id ID,
IN/OUT
Res BOOLEAN;
Ak!Id
ELSE
(=Id)
Res := FALSE
ELSE
Ak!SeqNum
(<= Last)
Res := TRUE


PROCEDURE Open_Frame;
1 (1)
FPAR
IN
Fr Frame,
IN/OUT
Id ID,
SeqNum SEQ_NUM,
Len SIZE,
More MORE_DATA,
SumH FCSH,
Data DATATYPE,
SumD FCSD;
Id := Fr!Id,
SeqNum := Fr!SeqNum,
Len := Fr!Len,
More := Fr!More,
SumH := Fr!SumH,
Data := Fr!Data,
SumD := Fr!SumD
Id := Fr!Id,
SeqNum := Fr!S
Len := Fr!Len,

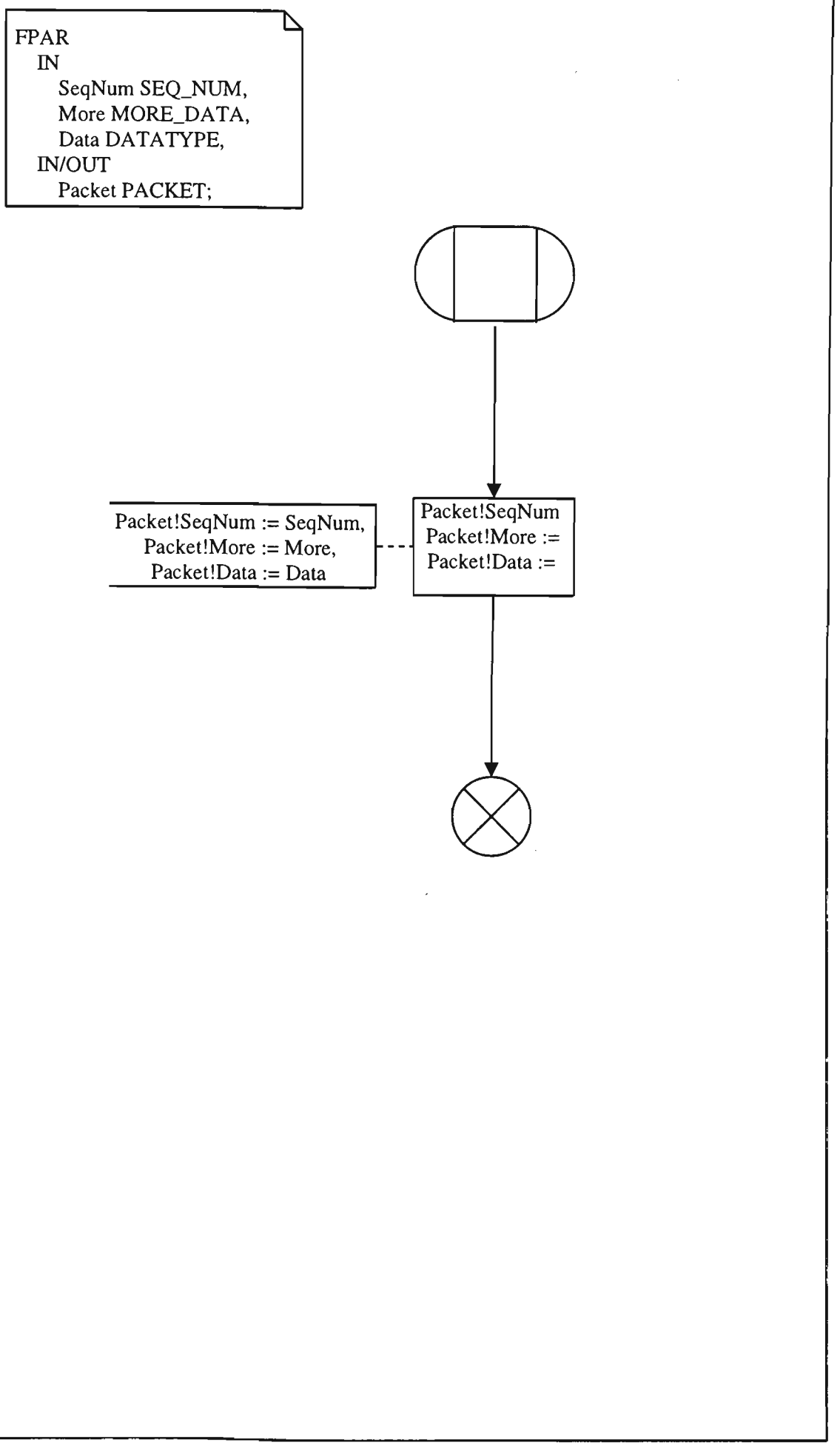

PROCEDURE NAK;
1 (1)
FPAR
IN
ProcAck PID,
Id ID,
SeqNum SEQ_NUM,
LastRec SEQ_NUM;
DCL
Ak ACK,
TIMER Timeout(SEQ_NUM)
SeqNum
(> LastRec+1)
ELSE
LastRec := LastRec + 1
Ak!Id := Id, Ak!SeqNum := L Ak!Ok := FALSE
Ak!SeqNum := LastRec
Ack(Ak) to ProcAck
SET(NOW + 10 * second, TimeOu (LastRec))
SeqNum
(> LastRec+1)
ELSE
Ak!SeqNum := SeqNum
Ak!Id := Id, Ak!SeqNum := S Ak!Ok := FALSE
Ack(Ak) to ProcAck
SET(NOW + 10 * second, TimeOu (LastRec))
SeqNum
(> LastRec)
LastRec := LastRec + 1
ELSE

PROCEDURE ReNAK; 1 (1)




PROCEDURE AK;
1 (1)
FPAR
IN
ProcAck PID,
Id ID,
SeqNum SEQ_NUM,
More MORE_DATA,
Last SEQ_NUM,
LastRec SEQ_NUM,
Count SEQ_NUM,
IN/OUT
Res BOOLEAN;
DCL
Ak ACK;
SeqNum=0
(= 0)
ELSE
More
Count
ELSE
TRUE
FALSE
SeqNum
(=Last)
ELSE
(> LastRec+1)
Ak!Id := Id,
Ak!SeqNum := L
Ak!Ok := TRUE
Res := FALSE,
Count :=
Count+1
LastRec :=
LastRec + 1
Ak!SeqNum
:= Last
Ak!Id := Id,
Ak!SeqNum := L
Ak!Ok := FALSE
Ack(Ak)
to
ProcAck
Ak!SeqNum
:= LastRec
Res := TRUE
Ack(Ak)
to
ProcAck
+ Count :=
Count+1
SET(NOW + 10 *
second, TimeOu
(LastRec))
Res := FALSE,
LastRec :=
LastRec + 1,
ELSE
SeqNum
(> LastRec+1)

PROCEDURE Pack_Packet; 1 (1)



APPENDIX B. MATHEMATICAL MODEL RESULTS

This Appendix presents the results of the mathematical model computations that were carried out as planned in section 7.2. These results are a reflection of the simulation results (given in Appendix C), except that the mathematical model was evaluated with a greater array of frame sizes. Tables B.1 through B.15 show the results for each of the 15 *Network Size* and *Base Error Rate* combinations with seven frame sizes. The optimal frame size is indicated by an asterisk (*) in the **Frame Size** field. Tables B.16 through B.19 show the results of various *Network Size* and *Base Error Rate* combinations that were re-evaluated with 2 Mbyte and 10 Mbyte files. The items in the tables that may not be clear are as follows:

- **Model Number.** The model number is made up of an "M"—for mathematical model—and three numbers representing three main parameters. For example, the model M3-01-64 has a network size of 3, a *Base Error Rate* of 0.01 and a frame size of 64 bytes (512 bits).
- **Total Bits Sent.** This represents the total transmission volume, including retransmitted frames and the frame overheads.
- **Total Data Sent.** This is the total transmission volume excluding the frame headers.
- **Satellite Link Utilisation.** This represents the link utilisation rate over the entire transmission time.
- **Average Return Traffic.** This is the return traffic—in bps—averaged over the entire transmission, with consideration to the initial propagation delay.
- **Peak Return Traffic.** This is the return traffic—in bps—during the first transmission, with consideration to the propagation initial propagation delay.

Table B.1. *Mathematical model results: network size = 3, base error rate = 0.01 and file size = 5 Mbytes.*

Model Number	M3-01-48	M3-01-64	M3-01-80	M3-01-96	M3-01-112	M3-01-128	M3-01-144
Frame Size (bits)	384	512	640	768	896 *	1,024	1,152
Frame Error Rate	0.02970	0.03940	0.04901	0.05852	0.06793	0.07726	0.08648
Transmission							
F_1 Frames	160,000	106,667	80,000	64,000	53,333	45,714	40,000
N_1 NAKs	14,256.48	12,609.32	11,762.39	11,235.81	10,869.48	10,594.95	10,377.93
F_2 Frames	13,837.24	12,118.98	11,195.33	10,591.12	10,147.78	9,797.51	9,506.29
N_2 NAKs	423.43	496.86	576.47	657.52	738.41	818.52	897.51
F_3 Frames	423.06	496.09	575.09	655.27	735.01	813.64	890.82
N_3 NAKs/F_4 Frames	12.58	19.58	28.25	38.48	50.16	63.23	77.62
N_4 NAKs/F_5 Frames	0.37	0.77	1.38	2.25	3.41	4.89	6.71
N_5 NAKs/F_6 Frames	0.0111	0.0304	0.0679	0.1318	0.2315	0.3774	0.5805
N_6 NAKs/F_7 Frames	0.00	0.00	0.00	0.01	0.02	0.03	0.05
Total Frames Sent	174,273.26	119,302.45	91,800.13	75,287.26	64,269.61	56,393.68	50,482.07
Total Bits Sent	66,920,932	61,082,856	58,752,084	57,820,614	57,585,574	57,747,125	58,155,345
Total Data Sent (bits)	44,613,955	45,812,142	47,001,667	48,183,845	49,359,064	50,528,734	51,693,640
Total NAKs Sent	14,692.87	13,126.55	12,368.57	11,934.20	11,661.71	11,481.99	11,360.41
Performance							
Satellite Link Utilisation	98.32%	97.91%	97.71%	97.74%	97.80%	97.85%	97.81%
Average Return Traffic (bps)	35,376	34,480	33,790	33,062	32,457	31,884	31,312
Peak Return Traffic (bps)	38,020	37,831	37,643	37,456	37,270	37,086	36,902
Transmission Time (sec.)	33.234	30.463	29.361	28.884	28.751	28.817	29.032
Effective Throughput (bps)	1,232,482	1,344,569	1,395,054	1,418,069	1,424,650	1,421,396	1,410,837

Table B.2. *Mathematical model results: network size = 3, base error rate = 0.005 and file size = 5 Mbytes.*

Model Number	M3-005-80	M3-005-96	M3-005-112	M3-005-128	M3-005-144	M3-005-160	M3-005-176
Frame Size (bits)	640	768	896	1,024	1,152 *	1,280	1,408
Frame Error Rate	0.02475	0.02963	0.03448	0.03931	0.04411	0.04889	0.05365
Transmission							
F_1 Frames	80,000	64,000	53,333	45,714	40,000	35,556	32,000
N_1 NAKs	5,940.30	5,688.48	5,516.66	5,390.63	5,293.25	5,214.98	5,149.96
F_2 Frames	5,794.48	5,521.61	5,328.64	5,181.52	5,063.20	4,964.18	4,878.63
N_2 NAKs	147.03	168.54	190.21	211.89	233.49	254.96	276.27
F_3 Frames	146.94	168.39	189.98	211.56	233.03	254.35	275.48
N_3 NAKs/F_4 Frames	3.64	4.99	6.56	8.33	10.30	12.46	14.82
N_4 NAKs/F_5 Frames	0.09	0.15	0.23	0.33	0.45	0.61	0.80
N_5 NAKs/F_6 Frames	0.0022	0.0044	0.0078	0.0129	0.0200	0.0298	0.0427
N_6 NAKs/F_7 Frames	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Total Frames Sent	85,945.15	69,695.14	58,858.41	51,115.75	45,307.00	40,787.64	37,169.77
Total Bits Sent	55,004,898	53,525,868	52,737,140	52,342,531	52,193,669	52,208,174	52,335,032
Total Data Sent (bits)	44,003,919	44,604,890	45,203,262	45,799,715	46,394,372	46,987,357	47,577,302
Total NAKs Sent	6,091.06	5,862.16	5,713.67	5,611.19	5,537.51	5,483.05	5,441.89
Performance							
Satellite Link Utilisation	98.16%	98.09%	98.03%	97.96%	97.88%	97.79%	97.67%
Average Return Traffic (bps)	17,819	17,611	17,410	17,215	17,024	16,835	16,649
Peak Return Traffic (bps)	19,012	18,965	18,918	18,871	18,824	18,777	18,730
Transmission Time (sec.)	27.360	26.644	26.268	26.090	26.037	26.069	26.164
Effective Throughput (bps)	1,497,078	1,537,328	1,559,293	1,569,957	1,573,168	1,571,208	1,565,513

Table B.3. *Mathematical model results: network size = 3, base error rate = 0.001 and file size = 5 Mbytes.*

Model Number	M3-001-256	M3-001-272	M3-001-288	M3-001-304	M3-001-320	M3-001-336	M3-001-384
Frame Size (bits)	2,048	2,176	2,304 *	2,432	2,560	2,688	3,072
Frame Error Rate	0.01588	0.01686	0.01785	0.01883	0.01981	0.02079	0.02373
Transmission							
F_1 Frames	21,333	20,000	18,824	17,778	16,842	16,000	13,913
N_1 NAKs	1,016.34	1,011.88	1,007.90	1,004.28	1,000.98	997.98	990.30
F_2 Frames	1,000.29	994.91	990.02	985.49	981.28	977.38	966.99
N_2 NAKs	16.14	17.07	17.99	18.91	19.83	20.75	23.50
F_3 Frames	16.14	17.06	17.98	18.90	19.82	20.74	23.48
N_3 NAKs/F_4 Frames	0.26	0.29	0.32	0.36	0.39	0.43	0.56
N_4 NAKs/F_5 Frames	0.00	0.00	0.01	0.01	0.01	0.01	0.01
N_5 NAKs/F_6 Frames	0.0001	0.0001	0.0001	0.0001	0.0002	0.0002	0.0003
N_6 NAKs/F_7 Frames	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Total Frames Sent	22,349.68	21,012.26	19,832.33	18,782.75	17,843.50	16,998.56	14,904.04
Total Bits Sent	45,772,148	45,722,687	45,693,688	45,679,654	45,679,364	45,692,126	45,785,222
Total Data Sent (bits)	42,911,389	43,033,118	43,155,150	43,275,461	43,395,395	43,516,311	43,877,505
Total NAKs Sent	1,032.74	1,029.24	1,026.22	1,023.55	1,021.21	1,019.17	1,014.37
Performance							
Satellite Link Utilisation	98.64%	98.60%	98.57%	98.53%	98.50%	98.46%	98.34%
Average Return Traffic (bps)	3,657	3,647	3,638	3,628	3,618	3,609	3,580
Peak Return Traffic (bps)	3,815	3,813	3,811	3,809	3,807	3,805	3,800
Transmission Time (sec.)	22.659	22.642	22.635	22.636	22.645	22.660	22.734
Effective Throughput (bps)	1,807,655	1,809,041	1,809,614	1,809,501	1,808,809	1,807,620	1,801,705

Table B.4. *Mathematical model results: network size = 3, base error rate = 0.0005 and file size = 5 Mbytes.*

Model Number	M3-0005-320	M3-0005-368	M3-0005-384	M3-0005-400	M3-0005-416	M3-0005-432	M3-0005-448
Frame Size (bits)	2,560	2,944	3,072	3,200	3,328 *	3,456	3,584
Frame Error Rate	0.00995	0.01144	0.01193	0.01243	0.01292	0.01341	0.01391
Transmission							
F_1 Frames	16,842	14,545	13,913	13,333	12,800	12,308	11,852
N_1 NAKs	502.87	499.05	498.00	497.00	496.09	495.25	494.44
F_2 Frames	497.88	493.37	492.08	490.85	489.71	488.63	487.59
N_2 NAKs	5.00	5.71	5.94	6.18	6.41	6.64	6.88
F_3 Frames	5.00	5.71	5.94	6.17	6.41	6.64	6.87
N_3 NAKs/F_4 Frames	0.05	0.07	0.07	0.08	0.08	0.09	0.10
N_4 NAKs/F_5 Frames	0.00	0.00	0.00	0.00	0.00	0.00	0.00
N_5 NAKs/F_6 Frames	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
N_6 NAKs/F_7 Frames	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Total Frames Sent	17,344.93	15,044.14	14,411.09	13,830.10	13,296.20	12,803.37	12,346.57
Total Bits Sent	44,403,030	44,289,946	44,270,878	44,256,324	44,249,763	44,248,434	44,250,092
Total Data Sent (bits)	42,182,879	42,364,296	42,426,258	42,486,071	42,547,849	42,609,603	42,669,732
Total NAKs Sent	507.92	504.83	504.01	503.25	502.59	501.98	501.41
Performance							
Satellite Link Utilisation	98.81%	98.79%	98.79%	98.78%	98.77%	98.77%	98.76%
Average Return Traffic (bps)	1,863	1,856	1,854	1,851	1,849	1,847	1,845
Peak Return Traffic (bps)	1,915	1,913	1,913	1,912	1,912	1,911	1,911
Transmission Time (sec.)	21.943	21.890	21.882	21.876	21.874	21.875	21.877
Effective Throughput (bps)	1,866,678	1,871,078	1,871,832	1,872,300	1,872,518	1,872,513	1,872,310

Table B.5. *Mathematical model results: network size = 3, base error rate = 0.0001 and file size = 5 Mbytes.*

Model Number	M3-0001-640	M3-0001-944	M3-0001-960	M3-0001-976	M3-0001-992	M3-0001-1008	M3-0001-1024
Frame Size (bits)	5,120	7,552	7,680	7,808	7,936 *	8,064	8,192
Frame Error Rate	0.00399	0.00588	0.00598	0.00608	0.00618	0.00628	0.00638
Transmission							
F_1 Frames	8,205	5,517	5,424	5,333	5,246	5,161	5,079
N_1 NAKs	98.27	97.37	97.34	97.30	97.28	97.24	97.21
F_2 Frames	97.88	96.80	96.76	96.71	96.68	96.63	96.59
N_2 NAKs	0.39	0.57	0.58	0.59	0.60	0.61	0.62
F_3 Frames	0.39	0.57	0.58	0.59	0.60	0.61	0.62
N_3 NAKs/F_4 Frames	0.00	0.00	0.00	0.00	0.00	0.00	0.00
N_4 NAKs/F_5 Frames	0.00	0.00	0.00	0.00	0.00	0.00	0.00
N_5 NAKs/F_6 Frames	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
N_6 NAKs/F_7 Frames	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Total Frames Sent	8,303.27	5,614.37	5,521.35	5,430.31	5,343.28	5,258.25	5,176.22
Total Bits Sent	42,512,744	42,399,743	42,403,962	42,399,833	42,404,298	42,402,498	42,403,558
Total Data Sent (bits)	41,449,926	41,681,103	41,697,229	41,704,754	41,720,358	41,729,442	41,741,002
Total NAKs Sent	98.66	97.94	97.93	97.90	97.88	97.86	97.83
Performance							
Satellite Link Utilisation	98.91%	98.95%	98.95%	98.96%	98.96%	98.96%	98.96%
Average Return Traffic (bps)	388	386	386	386	386	386	386
Peak Return Traffic (bps)	388	387	387	387	387	387	387
Transmission Time (sec.)	20.987	20.922	20.924	20.921	20.923	20.922	20.922
Effective Throughput (bps)	1,951,624	1,957,623	1,957,661	1,957,676	1,957,678	1,957,660	1,957,628

Table B.6. *Mathematical model results: network size = 6, base error rate = 0.01 and file size = 5 Mbytes.*

Model Number	M6-01-48	M6-01-64	M6-01-80	M6-01-96	M6-01-112	M6-01-128	M6-01-144
Frame Size (bits)	384	512	640	768 *	896	1,024	1,152
Frame Error Rate	0.029701	0.039404	0.04901	0.0585199	0.0679347	0.0772553	0.0864828
Transmission							
F_1 Frames	160,000	106,667	80,000	64,000	53,333	45,714	40,000
N_1 NAKs	28,512.96	25,218.63	23,524.78	22,471.62	21,738.95	21,189.89	20,755.86
F_2 Frames	26,477.80	22,861.07	20,823.97	19,429.55	18,364.73	17,495.20	16,753.34
N_2 NAKs	846.65	993.15	1,151.72	1,312.70	1,472.81	1,630.60	1,785.33
F_3 Frames	845.00	989.87	1,146.05	1,303.83	1,459.89	1,612.80	1,761.79
N_3 NAKs/F_4 Frames	25.15	39.16	56.51	76.96	100.33	126.47	155.24
N_4 NAKs/F_5 Frames	0.75	1.54	2.77	4.50	6.82	9.77	13.43
N_5 NAKs/F_6 Frames	0.02	0.06	0.14	0.26	0.46	0.75	1.16
N_6 NAKs/F_7 Frames	0.00	0.00	0.01	0.02	0.03	0.06	0.10
Total Frames Sent	187,347.95	130,557.09	102,026.53	84,810.35	73,257.98	64,948.53	58,670.48
Total Bits Sent	71,941,613	66,845,232	65,296,979	65,134,352	65,639,150	66,507,290	67,588,388
Total Data Sent (bits)	47,961,075	50,133,924	52,237,583	54,278,626	56,262,128	58,193,879	60,078,567
Total NAKs Sent	29,385.54	26,252.55	24,735.91	23,866.06	23,319.40	22,957.55	22,711.12
Performance							
Satellite Link Utilisation	98.29%	98.09%	98.17%	98.27%	98.15%	97.96%	97.82%
Average Return Traffic (bps)	65,789	63,129	60,946	59,011	57,142	55,416	53,866
Peak Return Traffic (bps)	76,037	75,658	75,282	74,908	74,537	74,168	73,801
Transmission Time (sec.)	35.740	33.276	32.477	32.363	32.656	33.151	33.739
Effective Throughput (bps)	1,146,049	1,230,920	1,261,204	1,265,654	1,254,278	1,235,567	1,214,030

Table B.7. *Mathematical model results: network size = 6, base error rate = 0.005 and file size = 5 Mbytes.*

Model Number	M3-005-80	M3-005-96	M3-005-112	M3-005-128	M3-005-144	M3-005-160	M3-005-176
Frame Size (bits)	640	768	896 *	1,024	1,152	1,280	1,408
Frame Error Rate	0.0247512	0.0296275	0.0344794	0.039307	0.0441104	0.0488899	0.0536454
Transmission							
F_1 Frames	80,000	64,000	53,333	45,714	40,000	35,556	32,000
N_1 NAKs	11,880.60	11,376.96	11,033.32	10,781.27	10,586.50	10,429.97	10,299.92
F_2 Frames	11,169.26	10,566.84	10,124.88	9,775.74	9,485.49	9,235.28	9,013.48
N_2 NAKs	294.02	336.99	380.28	423.54	466.61	509.38	551.78
F_3 Frames	293.61	336.33	379.29	422.15	464.71	506.88	548.58
N_3 NAKs/F_4 Frames	7.28	9.99	13.12	16.66	20.60	24.93	29.64
N_4 NAKs/F_5 Frames	0.18	0.30	0.45	0.65	0.91	1.22	1.59
N_5 NAKs/F_6 Frames	0.00	0.01	0.02	0.03	0.04	0.06	0.09
N_6 NAKs/F_7 Frames	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Total Frames Sent	91,470.15	74,913.16	63,850.29	55,928.54	49,970.80	45,323.10	41,591.71
Total Bits Sent	58,540,898	57,533,304	57,209,857	57,270,824	57,566,365	58,013,564	58,561,121
Total Data Sent (bits)	46,832,718	47,944,420	49,037,020	50,111,971	51,170,102	52,212,207	53,237,383
Total NAKs Sent	12,182.08	11,724.24	11,427.19	11,222.15	11,074.66	10,965.56	10,883.02
Performance							
Satellite Link Utilisation	98.28%	98.21%	98.13%	98.03%	97.91%	97.94%	98.05%
Average Return Traffic (bps)	33,523	32,806	32,131	31,490	30,877	30,346	29,870
Peak Return Traffic (bps)	38,021	37,926	37,832	37,738	37,644	37,550	37,457
Transmission Time (sec.)	29.086	28.605	28.467	28.525	28.709	28.924	29.163
Effective Throughput (bps)	1,408,252	1,431,897	1,438,858	1,435,913	1,426,732	1,416,142	1,404,497

Table B.8. *Mathematical model results: network size = 6, base error rate = 0.001 and file size = 5 Mbytes.*

Model Number	M6-001-128	M6-001-160	M6-001-176	M6-001-192	M6-001-208	M6-001-224	M6-001-240
Frame Size (bits)	1,024	1,280	1,408	1,536	1,664	1,792 *	1,920
Frame Error Rate	0.0079721	0.0099551	0.0109452	0.0119342	0.0129223	0.0139094	0.0148955
Transmission							
F_1 Frames	45,714	35,556	32,000	29,091	26,667	24,615	22,857
N_1 NAKs	2,186.61	2,123.79	2,101.47	2,083.07	2,067.59	2,054.27	2,042.79
F_2 Frames	2,143.49	2,071.63	2,044.80	2,021.90	2,001.94	1,984.15	1,968.22
N_2 NAKs	17.43	21.14	23.00	24.86	26.72	28.57	30.43
F_3 Frames	17.43	21.14	22.99	24.85	26.71	28.56	30.41
N_3 NAKs/F_4 Frames	0.14	0.21	0.25	0.30	0.35	0.40	0.45
N_4 NAKs/F_5 Frames	0.00	0.00	0.00	0.00	0.00	0.01	0.01
N_5 NAKs/F_6 Frames	0.00	0.00	0.00	0.00	0.00	0.00	0.00
N_6 NAKs/F_7 Frames	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Total Frames Sent	47,875.06	37,648.97	34,068.05	31,138.05	28,695.99	26,628.11	24,856.08
Total Bits Sent	49,024,058	48,190,686	47,967,811	47,828,043	47,750,125	47,717,570	47,723,674
Total Data Sent (bits)	42,896,051	43,371,617	43,607,101	43,842,372	44,077,038	44,309,172	44,542,096
Total NAKs Sent	2,204.18	2,145.14	2,124.73	2,108.23	2,094.66	2,083.25	2,073.68
Performance							
Satellite Link Utilisation	98.83%	98.75%	98.70%	98.66%	98.61%	98.56%	98.51%
Average Return Traffic (bps)	7,300	7,222	7,183	7,145	7,108	7,070	7,033
Peak Return Traffic (bps)	7,657	7,649	7,645	7,641	7,638	7,634	7,630
Transmission Time (sec.)	24.220	23.829	23.729	23.671	23.644	23.640	23.656
Effective Throughput (bps)	1,691,163	1,718,937	1,726,127	1,730,372	1,732,371	1,732,625	1,731,500

Table B.9. *Mathematical model results: network size = 6, base error rate = 0.0005 and file size = 5 Mbytes.*

Model Number	M6-0005-288	M6-0005-304	M6-0005-320	M6-0005-336	M6-0005-352	M6-0005-384	M6-0005-448
Frame Size (bits)	2,304	2,432 *	2,560	2,688	2,816	3,072	3,584
Frame Error Rate	0.0089619	0.0094574	0.0099526	0.0104477	0.0109424	0.0119313	0.0139059
Transmission							
F_1 Frames	18,824	17,778	16,842	16,000	15,238	13,913	11,852
N_1 NAKs	1,012.19	1,008.80	1,005.73	1,002.98	1,000.45	996.00	988.88
F_2 Frames	989.78	985.25	981.04	977.14	973.47	966.76	955.13
N_2 NAKs	9.07	9.54	10.01	10.48	10.95	11.88	13.75
F_3 Frames	9.07	9.54	10.01	10.48	10.94	11.88	13.74
N_3 NAKs/F_4 Frames	0.08	0.09	0.10	0.11	0.12	0.14	0.19
N_4 NAKs/F_5 Frames	0.00	0.00	0.00	0.00	0.00	0.00	0.00
N_5 NAKs/F_6 Frames	0.00	0.00	0.00	0.00	0.00	0.00	0.00
N_6 NAKs/F_7 Frames	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Total Frames Sent	19,822.93	18,772.87	17,833.15	16,987.73	16,222.54	14,891.78	12,821.07
Total Bits Sent	45,672,029	45,655,631	45,652,855	45,663,009	45,682,665	45,747,541	45,950,699
Total Data Sent (bits)	43,134,694	43,252,703	43,370,212	43,488,580	43,606,180	43,841,394	44,309,602
Total NAKs Sent	1,021.34	1,018.43	1,015.84	1,013.57	1,011.51	1,008.02	1,002.82
Performance							
Satellite Link Utilisation	98.82%	98.81%	98.80%	98.79%	98.78%	98.77%	98.73%
Average Return Traffic (bps)	3,642	3,632	3,623	3,614	3,605	3,587	3,551
Peak Return Traffic (bps)	3,827	3,826	3,825	3,825	3,824	3,822	3,818
Transmission Time (sec.)	22.568	22.562	22.562	22.569	22.581	22.617	22.726
Effective Throughput (bps)	1,814,995	1,815,476	1,815,403	1,814,861	1,813,916	1,811,036	1,802,388

Table B.10. *Mathematical model results: network size = 6, base error rate = 0.0001 and file size = 5 Mbytes.*

Model Number	M6-0001-512	M6-0001-624	M6-0001-640	M6-0001-656	M6-0001-672	M6-0001-688	M6-0001-800
Frame Size (bits)	4,096	4,992	5,120	5,248 *	5,376	5,504	6,400
Frame Error Rate	0.003195	0.0038926	0.0039922	0.0040918	0.0041914	0.004291	0.0049878
Transmission							
F_1 Frames	10,323	8,421	8,205	8,000	7,805	7,619	6,531
N_1 NAKs	197.89	196.68	196.54	196.41	196.28	196.16	195.45
F_2 Frames	196.32	194.77	194.59	194.41	194.24	194.07	193.03
N_2 NAKs	0.63	0.77	0.78	0.80	0.82	0.84	0.97
F_3 Frames	0.63	0.77	0.78	0.80	0.82	0.84	0.97
N_3 NAKs/F_4 Frames	0.00	0.00	0.00	0.00	0.00	0.00	0.00
N_4 NAKs/F_5 Frames	0.00	0.00	0.00	0.00	0.00	0.00	0.00
N_5 NAKs/F_6 Frames	0.00	0.00	0.00	0.00	0.00	0.00	0.00
N_6 NAKs/F_7 Frames	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Total Frames Sent	10,519.95	8,616.54	8,400.37	8,195.22	8,000.06	7,813.91	6,725.01
Total Bits Sent	43,089,736	43,013,777	43,009,910	43,008,491	43,008,345	43,007,766	43,040,060
Total Data Sent (bits)	41,743,181	41,910,860	41,934,662	41,959,504	41,984,337	42,007,585	42,179,259
Total NAKs Sent	198.53	197.45	197.32	197.21	197.11	197.00	196.43
Performance							
Satellite Link Utilisation	98.99%	99.02%	99.02%	99.03%	99.02%	99.01%	98.85%
Average Return Traffic (bps)	770	767	767	767	766	766	762
Peak Return Traffic (bps)	771	770	770	770	770	770	770
Transmission Time (sec.)	21.255	21.211	21.208	21.206	21.207	21.211	21.261
Effective Throughput (bps)	1,927,135	1,931,060	1,931,305	1,931,490	1,931,457	1,931,080	1,926,689

Table B.11. *Mathematical model results: network size = 12, base error rate = 0.01 and file size = 5 Mbytes.*

Model Number	M12-01-32	M12-01-48	M12-01-64	M12-01-80	M12-01-96	M12-01-112	M12-01-128
Frame Size (bits)	256	384	512	640 *	768	896	1,024
Frame Error Rate	0.0199	0.029701	0.03940399	0.04900995	0.05851985	0.06793465	0.07725531
Transmission							
F_1 Frames	320,000	160,000	106,667	80,000	64,000	53,333	45,714
N_1 NAKs	76,416.00	57,025.92	50,437.26	47,049.55	44,943.25	43,477.91	42,379.79
F_2 Frames	68,582.99	48,573.89	40,822.51	36,227.47	32,960.55	30,405.73	28,294.81
N_2 NAKs	1,520.68	1,693.73	1,987.43	2,305.90	2,630.07	2,953.66	3,274.08
F_3 Frames	1,517.37	1,685.53	1,970.55	2,275.68	2,581.12	2,879.91	3,168.94
N_3 NAKs/F_4 Frames	30.26	50.31	78.31	113.01	153.91	200.66	252.94
N_4 NAKs/F_5 Frames	0.60	1.49	3.09	5.54	9.01	13.63	19.54
N_5 NAKs/F_6 Frames	0.01	0.04	0.12	0.27	0.53	0.93	1.51
N_6 NAKs/F_7 Frames	0.00	0.00	0.00	0.01	0.03	0.06	0.12
Total Frames Sent	390,151.23	210,326.22	149,554.79	118,634.20	99,716.59	86,844.60	77,461.66
Total Bits Sent	99,878,715	80,765,268	76,572,053	75,925,890	76,582,345	77,812,760	79,320,742
Total Data Sent (bits)	49,939,357	53,843,512	57,429,040	60,740,712	63,818,621	66,696,651	69,405,649
Total NAKs Sent	77,967.55	58,771.49	52,506.22	49,474.28	47,736.80	46,646.84	45,927.97
Performance							
Satellite Link Utilisation	98.87%	98.51%	98.63%	98.53%	98.40%	98.20%	98.21%
Average Return Traffic (bps)	126,475	117,475	110,829	105,221	100,523	96,473	93,193
Peak Return Traffic (bps)	152,834	152,071	151,314	150,561	149,813	149,070	148,332
Transmission Time (sec.)	49.325	40.031	37.909	37.625	38.000	38.692	39.436
Effective Throughput (bps)	830,413	1,023,198	1,080,472	1,088,652	1,077,882	1,058,621	1,038,626

Table B.12. *Mathematical model results: network size = 12, base error rate = 0.005 and file size = 5 Mbytes.*

Model Number	M12-005-48	M12-005-64	M12-005-80	M12-005-96	M12-005-112	M12-005-128	M12-005-144
Frame Size (bits)	384	512	640	768 *	896	1,024	1,152
Frame Error Rate	0.0149251	0.0198505	0.0247512	0.0296275	0.0344794	0.039307	0.0441104
Transmission							
F_1 Frames	160,000	106,667	80,000	64,000	53,333	45,714	40,000
N_1 NAKs	28,656.24	25,408.72	23,761.20	22,753.91	22,066.65	21,562.54	21,173.00
F_2 Frames	26,417.09	22,810.26	20,779.12	19,389.02	18,327.62	17,460.97	16,721.62
N_2 NAKs	427.70	504.38	588.12	674.14	760.84	847.56	933.95
F_3 Frames	427.17	503.28	586.14	670.90	755.89	840.39	924.02
N_3 NAKs/F_4 Frames	6.38	10.01	14.56	19.97	26.23	33.31	41.20
N_4 NAKs/F_5 Frames	0.10	0.20	0.36	0.59	0.90	1.31	1.82
N_5 NAKs/F_6 Frames	0.00	0.00	0.01	0.02	0.03	0.05	0.08
N_6 NAKs/F_7 Frames	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Total Frames Sent	186,865.75	130,004.09	101,392.68	84,092.48	72,455.31	64,061.42	57,699.91
Total Bits Sent	71,756,447	66,562,095	64,891,317	64,583,024	64,919,961	65,598,894	66,470,294
Total Data Sent (bits)	47,837,631	49,921,571	51,913,053	53,819,186	55,645,681	57,399,032	59,084,706
Total NAKs Sent	29,090.42	25,923.31	24,364.24	23,448.64	22,854.66	22,444.77	22,150.05
Performance							
Satellite Link Utilisation	98.64%	98.53%	98.46%	98.38%	98.27%	98.36%	98.47%
Average Return Traffic (bps)	65,545	62,900	60,598	58,554	56,713	55,166	53,789
Peak Return Traffic (bps)	76,419	76,229	76,039	75,849	75,660	75,472	75,284
Transmission Time (sec.)	35.520	32.986	32.181	32.053	32.256	32.566	32.962
Effective Throughput (bps)	1,153,142	1,241,739	1,272,802	1,277,876	1,269,841	1,257,743	1,242,655

Table B.13. *Mathematical model results: network size = 12, base error rate = 0.001 and file size = 5 Mbytes.*

Model Number	M12-001-112	M12-001-128	M12-001-144	M12-001-160	M12-001-176	M12-001-192	M12-001-208
Frame Size (bits)	896	1,024	1,152	1,280 *	1,408	1,536	1,664
Frame Error Rate	0.006979	0.0079721	0.0089641	0.0099551	0.0109452	0.0119342	0.0129223
Transmission							
F_1 Frames	53,333	45,714	40,000	35,556	32,000	29,091	26,667
N_1 NAKs	4,466.55	4,373.21	4,302.76	4,247.57	4,202.94	4,166.14	4,135.18
F_2 Frames	4,299.03	4,186.47	4,096.84	4,022.55	3,958.94	3,903.27	3,853.58
N_2 NAKs	31.17	34.86	38.57	42.29	46.00	49.72	53.44
F_3 Frames	31.16	34.85	38.55	42.26	45.97	49.68	53.39
N_3 NAKs/F_4 Frames	0.22	0.28	0.35	0.42	0.50	0.59	0.69
N_4 NAKs/F_5 Frames	0.00	0.00	0.00	0.00	0.01	0.01	0.01
N_5 NAKs/F_6 Frames	0.00	0.00	0.00	0.00	0.00	0.00	0.00
N_6 NAKs/F_7 Frames	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Total Frames Sent	57,675.08	49,947.03	44,146.99	39,632.35	36,016.42	33,055.46	30,585.50
Total Bits Sent	51,676,875	51,145,760	50,857,331	50,729,407	50,711,121	50,773,194	50,894,279
Total Data Sent (bits)	44,294,464	44,752,540	45,206,516	45,656,466	46,101,019	46,542,095	46,979,334
Total NAKs Sent	4,497.95	4,408.36	4,341.68	4,290.28	4,249.45	4,216.46	4,189.32
Performance							
Satellite Link Utilisation	98.85%	98.78%	98.72%	98.65%	98.58%	98.51%	98.43%
Average Return Traffic (bps)	14,134	13,988	13,846	13,708	13,573	13,441	13,312
Peak Return Traffic (bps)	15,317	15,310	15,302	15,294	15,287	15,279	15,272
Transmission Time (sec.)	25.527	25.281	25.155	25.108	25.117	25.167	25.248
Effective Throughput (bps)	1,604,538	1,620,180	1,628,324	1,631,343	1,630,740	1,627,514	1,622,349

Table B.14. *Mathematical model results: network size = 12, base error rate = 0.0005 and file size = 5 Mbytes.*

Model Number	M12-0005-192	M12-0005-208	M12-0005-224	M12-0005-240	M12-0005-256	M12-0005-272	M12-0005-320
Frame Size (bits)	1,536	1,664	1,792 *	1,920	2,048	2,176	2,560
Frame Error Rate	0.0059835	0.0064805	0.0069773	0.0074738	0.0079701	0.0084661	0.0099526
Transmission							
F_1 Frames	29,091	26,667	24,615	22,857	21,333	20,000	16,842
N_1 NAKs	2,088.80	2,073.80	2,060.95	2,049.95	2,040.31	2,031.86	2,011.47
F_2 Frames	2,021.41	2,001.45	1,983.68	1,967.75	1,953.20	1,939.87	1,904.93
N_2 NAKs	12.50	13.44	14.38	15.32	16.26	17.20	20.02
F_3 Frames	12.50	13.44	14.38	15.32	16.26	17.20	20.01
N_3 NAKs/F_4 Frames	0.07	0.09	0.10	0.11	0.13	0.15	0.20
N_4 NAKs/F_5 Frames	0.00	0.00	0.00	0.00	0.00	0.00	0.00
N_5 NAKs/F_6 Frames	0.00	0.00	0.00	0.00	0.00	0.00	0.00
N_6 NAKs/F_7 Frames	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Total Frames Sent	31,135.89	28,692.81	26,623.92	24,850.89	23,313.26	21,967.84	18,777.67
Total Bits Sent	47,824,733	47,744,840	47,710,068	47,713,711	47,745,548	47,802,014	48,070,838
Total Data Sent (bits)	43,839,338	44,072,160	44,302,206	44,532,797	44,761,451	44,990,131	45,667,296
Total NAKs Sent	2,101.38	2,087.32	2,075.43	2,065.38	2,056.70	2,049.21	2,031.69
Performance							
Satellite Link Utilisation	98.91%	98.90%	98.89%	98.88%	98.87%	98.86%	98.83%
Average Return Traffic (bps)	7,161	7,125	7,089	7,054	7,019	6,984	6,884
Peak Return Traffic (bps)	7,662	7,661	7,659	7,657	7,655	7,653	7,647
Transmission Time (sec.)	23.610	23.572	23.557	23.561	23.579	23.609	23.749
Effective Throughput (bps)	1,734,894	1,737,652	1,738,712	1,738,441	1,737,109	1,734,925	1,724,703

Table B.14. *Mathematical model results: network size = 12, base error rate = 0.0005 and file size = 5 Mbytes.*

Model Number	M12-0005-192	M12-0005-208	M12-0005-224	M12-0005-240	M12-0005-256	M12-0005-272	M12-0005-320
Frame Size (bits)	1,536	1,664	1,792 *	1,920	2,048	2,176	2,560
Frame Error Rate	0.0059835	0.0064805	0.0069773	0.0074738	0.0079701	0.0084661	0.0099526
Transmission							
F_1 Frames	29,091	26,667	24,615	22,857	21,333	20,000	16,842
N_1 NAKs	2,088.80	2,073.80	2,060.95	2,049.95	2,040.31	2,031.86	2,011.47
F_2 Frames	2,021.41	2,001.45	1,983.68	1,967.75	1,953.20	1,939.87	1,904.93
N_2 NAKs	12.50	13.44	14.38	15.32	16.26	17.20	20.02
F_3 Frames	12.50	13.44	14.38	15.32	16.26	17.20	20.01
N_3 NAKs/F_4 Frames	0.07	0.09	0.10	0.11	0.13	0.15	0.20
N_4 NAKs/F_5 Frames	0.00	0.00	0.00	0.00	0.00	0.00	0.00
N_5 NAKs/F_6 Frames	0.00	0.00	0.00	0.00	0.00	0.00	0.00
N_6 NAKs/F_7 Frames	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Total Frames Sent	31,135.89	28,692.81	26,623.92	24,850.89	23,313.26	21,967.84	18,777.67
Total Bits Sent	47,824,733	47,744,840	47,710,068	47,713,711	47,745,548	47,802,014	48,070,838
Total Data Sent (bits)	43,839,338	44,072,160	44,302,206	44,532,797	44,761,451	44,990,131	45,667,296
Total NAKs Sent	2,101.38	2,087.32	2,075.43	2,065.38	2,056.70	2,049.21	2,031.69
Performance							
Satellite Link Utilisation	98.91%	98.90%	98.89%	98.88%	98.87%	98.86%	98.83%
Average Return Traffic (bps)	7,161	7,125	7,089	7,054	7,019	6,984	6,884
Peak Return Traffic (bps)	7,662	7,661	7,659	7,657	7,655	7,653	7,647
Transmission Time (sec.)	23.610	23.572	23.557	23.561	23.579	23.609	23.749
Effective Throughput (bps)	1,734,894	1,737,652	1,738,712	1,738,441	1,737,109	1,734,925	1,724,703

Table B.16. *Mathematical model results: network size = 3, base error rate = 0.01 and file size = 2 Mbytes.*

Model Number	M3-01-48	M3-01-64	M3-01-80	M3-01-96	M3-01-112	M3-01-128	M3-01-144
Frame Size (bits)	384	512	640	768	896 *	1,024	1,152
Frame Error Rate	0.02970	0.03940	0.04901	0.05852	0.06793	0.07726	0.08648
Transmission							
F_1 Frames	64,000	42,667	32,000	25,600	21,333	18,286	16,000
N_1 NAKs	5,702.59	5,043.75	4,704.96	4,494.32	4,347.75	4,238.07	4,151.17
F_2 Frames	5,534.90	4,847.62	4,478.13	4,236.45	4,059.08	3,919.09	3,802.52
N_2 NAKs	169.37	198.74	230.59	263.01	295.36	327.41	359.00
F_3 Frames	169.22	198.44	230.04	262.11	294.00	325.46	356.33
N_3 NAKs/F_4 Frames	5.03	7.83	11.30	15.39	20.07	25.29	31.05
N_4 NAKs/F_5 Frames	0.15	0.31	0.55	0.90	1.36	1.95	2.69
N_5 NAKs/F_6 Frames	0.0044	0.0122	0.0271	0.0527	0.0926	0.1510	0.2322
N_6 NAKs/F_7 Frames	0.00	0.00	0.00	0.00	0.01	0.01	0.02
Total Frames Sent	69,709.30	47,721.20	36,720.05	30,114.90	25,707.60	22,557.96	20,192.83
Total Bits Sent	26,768,373	24,433,257	23,500,834	23,128,245	23,034,014	23,099,355	23,262,138
Total Data Sent (bits)	17,845,582	18,324,943	18,800,667	19,273,538	19,743,440	20,211,936	20,677,456
Total NAKs Sent	5,877.15	5,250.65	4,947.43	4,773.68	4,664.64	4,592.90	4,544.16
Performance							
Satellite Link Utilisation	96.14%	95.54%	94.95%	94.25%	94.09%	94.15%	94.20%
Average Return Traffic (bps)	34,601	33,658	32,771	31,893	31,240	30,691	30,169
Peak Return Traffic (bps)	38,024	37,835	37,647	37,460	37,274	37,090	36,907
Transmission Time (sec.)	13.595	12.487	12.085	11.982	11.953	11.980	12.058
Effective Throughput (bps)	1,205,132	1,312,079	1,355,746	1,367,432	1,370,667	1,367,647	1,358,782

Table B.16. *Mathematical model results: network size = 3, base error rate = 0.01 and file size = 2 Mbytes.*

Model Number	M3-01-48	M3-01-64	M3-01-80	M3-01-96	M3-01-112	M3-01-128	M3-01-144
Frame Size (bits)	384	512	640	768	896 *	1,024	1,152
Frame Error Rate	0.02970	0.03940	0.04901	0.05852	0.06793	0.07726	0.08648
Transmission							
F_1 Frames	64,000	42,667	32,000	25,600	21,333	18,286	16,000
N_1 NAKs	5,702.59	5,043.75	4,704.96	4,494.32	4,347.75	4,238.07	4,151.17
F_2 Frames	5,534.90	4,847.62	4,478.13	4,236.45	4,059.08	3,919.09	3,802.52
N_2 NAKs	169.37	198.74	230.59	263.01	295.36	327.41	359.00
F_3 Frames	169.22	198.44	230.04	262.11	294.00	325.46	356.33
N_3 NAKs/F_4 Frames	5.03	7.83	11.30	15.39	20.07	25.29	31.05
N_4 NAKs/F_5 Frames	0.15	0.31	0.55	0.90	1.36	1.95	2.69
N_5 NAKs/F_6 Frames	0.0044	0.0122	0.0271	0.0527	0.0926	0.1510	0.2322
N_6 NAKs/F_7 Frames	0.00	0.00	0.00	0.00	0.01	0.01	0.02
Total Frames Sent	69,709.30	47,721.20	36,720.05	30,114.90	25,707.60	22,557.96	20,192.83
Total Bits Sent	26,768,373	24,433,257	23,500,834	23,128,245	23,034,014	23,099,355	23,262,138
Total Data Sent (bits)	17,845,582	18,324,943	18,800,667	19,273,538	19,743,440	20,211,936	20,677,456
Total NAKs Sent	5,877.15	5,250.65	4,947.43	4,773.68	4,664.64	4,592.90	4,544.16
Performance							
Satellite Link Utilisation	96.14%	95.54%	94.95%	94.25%	94.09%	94.15%	94.20%
Average Return Traffic (bps)	34,601	33,658	32,771	31,893	31,240	30,691	30,169
Peak Return Traffic (bps)	38,024	37,835	37,647	37,460	37,274	37,090	36,907
Transmission Time (sec.)	13.595	12.487	12.085	11.982	11.953	11.980	12.058
Effective Throughput (bps)	1,205,132	1,312,079	1,355,746	1,367,432	1,370,667	1,367,647	1,358,782

Table B.18. *Mathematical model results: network size = 6, base error rate = 0.001 and file size = 2 Mbytes.*

Model Number	M6-001-128	M6-001-160	M6-001-176	M6-001-192	M6-001-208	M6-001-224	M6-001-240
Frame Size (bits)	1,024	1,280	1,408	1,536	1,664	1,792 *	1,920
Frame Error Rate	0.0079721	0.0099551	0.0109452	0.0119342	0.0129223	0.0139094	0.0148955
Transmission							
F_1 Frames	18,286	14,222	12,800	11,636	10,667	9,846	9,143
N_1 NAKs	874.66	849.49	840.59	833.20	827.05	821.71	817.13
F_2 Frames	857.41	828.63	817.92	808.73	800.79	793.66	787.30
N_2 NAKs	6.97	8.46	9.20	9.94	10.69	11.43	12.17
F_3 Frames	6.97	8.45	9.20	9.94	10.68	11.42	12.16
N_3 NAKs/F_4 Frames	0.06	0.08	0.10	0.12	0.14	0.16	0.18
N_4 NAKs/F_5 Frames	0.00	0.00	0.00	0.00	0.00	0.00	0.00
N_5 NAKs/F_6 Frames	0.00	0.00	0.00	0.00	0.00	0.00	0.00
N_6 NAKs/F_7 Frames	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Total Frames Sent	19,150.44	15,059.17	13,627.22	12,454.79	11,478.61	10,651.24	9,942.65
Total Bits Sent	19,610,052	19,275,732	19,187,124	19,130,559	19,100,408	19,087,028	19,089,887
Total Data Sent (bits)	17,158,796	17,348,159	17,442,840	17,536,346	17,631,146	17,723,669	17,817,228
Total NAKs Sent	881.69	858.03	849.89	843.26	837.88	833.30	829.49
Performance							
Satellite Link Utilisation	96.97%	97.21%	97.16%	97.11%	97.06%	97.01%	96.96%
Average Return Traffic (bps)	7,192	7,139	7,101	7,063	7,026	6,989	6,953
Peak Return Traffic (bps)	7,662	7,654	7,650	7,647	7,643	7,639	7,635
Transmission Time (sec.)	9.875	9.682	9.643	9.619	9.609	9.607	9.613
Effective Throughput (bps)	1,659,213	1,692,139	1,699,104	1,703,212	1,705,149	1,705,390	1,704,297

Table B.19. *Mathematical model results: network size = 6, base error rate = 0.0005 and file size = 10 Mbytes.*

Model Number	M6-001-160	M6-001-192	M6-001-208	M6-001-224	M6-001-240	M6-001-256	M6-001-320
Frame Size (bits)	1,280	1,536	1,664	1,792 *	1,920	2,048	2,560
Frame Error Rate	0.0099551	0.0119342	0.0129223	0.0139094	0.0148955	0.0158806	0.0198111
Transmission							
F_1 Frames	71,111	58,182	53,333	49,231	45,714	42,667	33,684
N_1 NAKs	4,247.51	4,166.14	4,135.11	4,108.63	4,085.58	4,065.45	4,003.91
F_2 Frames	4,143.19	4,043.80	4,003.80	3,968.38	3,936.43	3,907.43	3,810.77
N_2 NAKs	42.28	49.72	53.43	57.15	60.85	64.56	79.32
F_3 Frames	42.27	49.70	53.41	57.12	60.82	64.52	79.24
N_3 NAKs/F_4 Frames	0.42	0.59	0.69	0.79	0.91	1.03	1.57
N_4 NAKs/F_5 Frames	0.00	0.01	0.01	0.01	0.01	0.02	0.03
N_5 NAKs/F_6 Frames	0.00	0.00	0.00	0.00	0.00	0.00	0.00
N_6 NAKs/F_7 Frames	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Total Frames Sent	75,296.89	62,276.10	57,390.90	53,257.30	49,712.16	46,639.97	37,575.58
Total Bits Sent	96,380,016	95,656,085	95,498,459	95,437,078	95,447,348	95,518,666	96,193,488
Total Data Sent (bits)	86,742,015	87,684,745	88,152,423	88,620,144	89,084,192	89,548,750	91,383,814
Total NAKs Sent	4,290.22	4,216.46	4,189.24	4,166.58	4,147.36	4,131.06	4,084.83
Performance							
Satellite Link Utilisation	99.27%	99.18%	99.14%	99.09%	99.03%	98.99%	99.03%
Average Return Traffic (bps)	7,250	7,173	7,135	7,098	7,061	7,025	6,900
Peak Return Traffic (bps)	7,647	7,640	7,636	7,632	7,628	7,624	7,609
Transmission Time (sec.)	47.406	47.091	47.036	47.029	47.060	47.115	47.430
Effective Throughput (bps)	1,728,058	1,739,618	1,741,640	1,741,898	1,740,761	1,738,743	1,727,152

APPENDIX C. SIMULATION RESULTS

This Appendix presents the results of the Network II.5 simulations that were carried out as planned in section 7.2. Tables C.1 through C.15 show the results for each of the 15 *Network Size* and *Base Error Rate* combinations with three frame sizes—the optimal solution is always shown in the middle column. Tables C.16 through C.19 show the results of various *Network Size* and *Base Error Rate* combinations that were re-simulated with 2 Mbyte and 10 Mbyte files. The items in the tables that may not be clear are as follows:

- **Model Number.** The model number is made up of an "S"—for simulation model—and three numbers representing three main parameters. For example, the model S3-01-64 has a network size of 3, a base error rate of 0.01 and a frame size of 64 bytes (512 bits).
- **F_4 Frames.** This represents frames that are sent in the fourth and all following transmissions, similarly, **N_3 NAKs** represent NAKs sent for frames in the third and all following transmissions.
- **Total Bits Sent.** This represents the total transmission volume, including retransmitted frames and the frame overheads.
- **Total Data Sent.** This is the total transmission volume excluding the frame headers.
- **Satellite Link Utilisation.** This represents the link utilisation rate over the entire transmission time.
- **Average Return Traffic.** This is the return traffic—in bps—averaged over the entire transmission, with consideration to the initial propagation delay.
- **Peak Return Traffic.** This is the return traffic—in bps—during the first transmission, with consideration to the propagation initial propagation delay.

Table C.1. *Simulation results: network size = 3, base error rate = 0.01 and file size = 5 Mbytes.*

	Model Number		
	S3-01-64	S3-01-96	S3-01-128
Frame Size (bits)	512	768	1,024
Frame Error Rate	0.03940	0.05852	0.07726
Sender			
F_1 Frames	106,667	64,000	45,714
F_2 Frames	12,066	10,731	9,798
F_3 Frames	491	679	851
F_4 Frames	30	42	69
Total Frames Sent	119,254	75,452	56,432
Total Bits Sent	61,058,048	57,947,136	57,786,368
Total Data Sent (bits)	45,793,536	48,289,280	50,563,072
Receivers			
N_1 NAKs	12,511	11,343	10,617
N_2 NAKs	491	680	854
N_3 NAKs	30	42	69
Total NAKs Sent	13,032	12,065	11,540
Communication Links			
Satellite Link Utilisation	97.76%	98.37%	97.51%
Average Return Traffic (bps)	34,195	33,565	31,911
Peak Return Traffic (bps)	37,536	37,813	37,163
Performance			
Transmission Time (sec.)	30.496	28.763	28.938
Effective Throughput (bps)	1,343,150	1,424,029	1,415,450

Table C.2. *Simulation results: network size = 3, base error rate = 0.005 and file size = 5 Mbytes.*

	Model Number		
	S3-005-96	S3-005-128	S3-005-160
Frame Size (bits)	768	1,024	1,280
Frame Error Rate	0.02963	0.03931	0.04889
Sender			
F_1 Frames	64,000	45,714	35,556
F_2 Frames	5,548	5,076	4,949
F_3 Frames	164	227	253
F_4 Frames	2	7	11
Total Frames Sent	69,714	51,024	40,769
Total Bits Sent	53,540,352	52,248,576	52,184,320
Total Data Sent (bits)	44,616,960	45,717,504	46,965,888
Receivers			
N_1 NAKs	5,691	5,283	5,209
N_2 NAKs	164	227	253
N_3 NAKs	2	7	11
Total NAKs Sent	5,857	5,517	5,473
Communication Links			
Satellite Link Utilisation	98.23%	98.17%	97.35%
Average Return Traffic (bps)	17,616	16,993	16,737
Peak Return Traffic (bps)	18,973	18,494	18,755
Performance			
Transmission Time (sec.)	26.613	25.987	26.174
Effective Throughput (bps)	1,539,124	1,576,171	1,564,923

Table C.3. *Simulation results: network size = 3, base error rate = 0.001 and file size = 5 Mbytes.*

	Model Number		
	S3-001-256	S3-001-320	S3-001-384
Frame Size (bits)	2,048	2,560	3,072
Frame Error Rate	0.01588	0.01981	0.02373
Sender			
F_1 Frames	21,333	16,842	13,913
F_2 Frames	1,012	939	923
F_3 Frames	17	14	20
F_4 Frames	1	1	0
Total Frames Sent	22,363	17,796	14,856
Total Bits Sent	45,799,424	45,557,760	45,637,632
Total Data Sent (bits)	42,936,960	43,279,872	43,736,064
Receivers			
N_1 NAKs	1,027	958	948
N_2 NAKs	17	14	20
N_3 NAKs	1	1	0
Total NAKs Sent	1,045	973	968
Communication Links			
Satellite Link Utilisation	97.99%	97.95%	98.05%
Average Return Traffic (bps)	3,674	3,438	3,418
Peak Return Traffic (bps)	3,855	3,644	3,638
Performance			
Transmission Time (sec.)	22.821	22.711	22.728
Effective Throughput (bps)	1,794,811	1,803,487	1,802,199

Table C.4. *Simulation results: network size = 3, base error rate = 0.0005 and file size = 5 Mbytes.*

	Model Number		
	S3-0005-320	S3-0005-384	S3-0005-448
Frame Size (bits)	2,560	3,072	3,584
Frame Error Rate	0.00995	0.01193	0.01391
Sender			
F_1 Frames	16,842	13,913	11,852
F_2 Frames	484	481	489
F_3 Frames	0	1	4
F_4 Frames	0	0	0
Total Frames Sent	17,326	14,395	12,345
Total Bits Sent	44,354,560	44,221,440	44,244,480
Total Data Sent (bits)	42,136,832	42,378,880	42,664,320
Receivers			
N_1 NAKs	487	487	495
N_2 NAKs	0	1	4
N_3 NAKs	0	0	0
Total NAKs Sent	487	488	499
Communication Links			
Satellite Link Utilisation	98.91%	98.91%	98.91%
Average Return Traffic (bps)	1,790	1,799	1,830
Peak Return Traffic (bps)	1,854	1,871	1,913
Performance			
Transmission Time (sec.)	21.896	21.831	21.842
Effective Throughput (bps)	1,870,668	1,876,240	1,875,304

Table C.5. *Simulation results: network size = 3, base error rate = 0.0001 and file size = 5 Mbytes.*

	Model Number		
	S3-0001-640	S3-0001-800	S3-0001-1024
Frame Size (bits)	5,120	6,400	8,192
Frame Error Rate	0.00399	0.00499	0.00638
Sender			
F_1 Frames	8,205	6,531	5,079
F_2 Frames	92	91	101
F_3 Frames	1	0	0
F_4 Frames	0	0	0
Total Frames Sent	8,298	6,622	5,180
Total Bits Sent	42,485,760	42,380,800	42,434,560
Total Data Sent (bits)	41,423,616	41,533,184	41,771,520
Receivers			
N_1 NAKs	94	92	101
N_2 NAKs	1	0	0
N_3 NAKs	0	0	0
Total NAKs Sent	95	92	101
Communication Links			
Satellite Link Utilisation	98.70%	98.86%	98.86%
Average Return Traffic (bps)	373	363	397
Peak Return Traffic (bps)	373	365	402
Performance			
Transmission Time (sec.)	21.018	20.933	20.959
Effective Throughput (bps)	1,948,807	1,956,874	1,954,168

Table C.6. *Simulation results: network size = 6, base error rate = 0.01 and file size = 5 Mbytes.*

	Model Number		
	S6-01-64	S6-01-96	S6-01-128
Frame Size (bits)	512	768	1,024
Frame Error Rate	0.03940	0.05852	0.07726
Sender			
F_1 Frames	106,667	64,000	45,714
F_2 Frames	22,996	19,2907	17,607
F_3 Frames	925	1,325	1,664
F_4 Frames	37	86	165
Total Frames Sent	130,625	84,701	65,150
Total Bits Sent	66,880,000	65,050,368	66,713,600
Total Data Sent (bits)	50,160,000	54,208,640	58,374,400
Receivers			
N_1 NAKs	25,360	22,381	21,301
N_2 NAKs	901	1,332	1,683
N_3 NAKs	37	86	165
Total NAKs Sent	26,298	23,799	23,149
Communication Links			
Satellite Link Utilisation	98.57%	97.79%	97.94%
Average Return Traffic (bps)	63,519	58,633	55,692
Peak Return Traffic (bps)	76,082	74,606	74,557
Performance			
Transmission Time (sec.)	33.129	32.480	33.261
Effective Throughput (bps)	1,236,374	1,261,076	1,231,456

Table C.7. *Simulation results: network size = 6, base error rate = 0.005 and file size = 5 Mbytes.*

	Model Number		
	S6-005-96	S6-005-128	S6-005-160
Frame Size (bits)	768	1,024	1,280
Frame Error Rate	0.02963	0.03931	0.04889
Sender			
F_1 Frames	64,000	45,714	35,556
F_2 Frames	10,478	9,725	9,266
F_3 Frames	365	404	504
F_4 Frames	11	16	31
Total Frames Sent	74,854	55,859	45,357
Total Bits Sent	57,487,872	57,199,616	58,056,960
Total Data Sent (bits)	47,906,560	50,049,664	52,251,264
Receivers			
N_1 NAKs	11,311	10,709	10,443
N_2 NAKs	365	405	505
N_3 NAKs	11	16	31
Total NAKs Sent	11,687	11,130	10,979
Communication Links			
Satellite Link Utilisation	98.36%	98.35%	97.94%
Average Return Traffic (bps)	32,779	31,371	30,361
Peak Return Traffic (bps)	37,706	37,485	37,597
Performance			
Transmission Time (sec.)	28.538	28.399	28.945
Effective Throughput (bps)	1,435,269	1,442,313	1,415,116

Table C.8. *Simulation results: network size = 6, base error rate = 0.001 and file size = 5 Mbytes.*

	Model Number		
	S6-001-128	S6-001-160	S6-001-192
Frame Size (bits)	1,024	1,280	1,536
Frame Error Rate	0.00797	0.00996	0.01193
Sender			
F_1 Frames	45,714	35,556	29,091
F_2 Frames	2,157	2,096	2,037
F_3 Frames	0	20	26
F_4 Frames	0	0	1
Total Frames Sent	47,871	37,672	31,155
Total Bits Sent	49,019,904	48,220,160	47,854,080
Total Data Sent (bits)	42,892,416	43,398,144	43,866,240
Receivers			
N_1 NAKs	2,193	2,148	2,107
N_2 NAKs	0	20	26
N_3 NAKs	0	0	1
Total NAKs Sent	2,193	2,168	2,134
Communication Links			
Satellite Link Utilisation	99.02%	98.96%	98.06%
Average Return Traffic (bps)	7,278	7,310	7,185
Peak Return Traffic (bps)	7,679	7,736	7,729
Performance			
Transmission Time (sec.)	24.173	23.793	23.828
Effective Throughput (bps)	1,694,476	1,721,504	1,719,024

Table C.9. *Simulation results: network size =6, base error rate = 0.0005 and file size = 5 Mbytes.*

	Model Number		
	S6-0005-320	S6-0005-384	S6-0005-448
Frame Size (bits)	2,560	3,072	3,584
Frame Error Rate	0.00995	0.01193	0.01391
Sender			
F_1 Frames	16,842	13,913	11,852
F_2 Frames	997	959	916
F_3 Frames	10	5	15
F_4 Frames	0	0	0
Total Frames Sent	17,849	14,877	12,783
Total Bits Sent	45,693,440	45,702,144	45,814,272
Total Data Sent (bits)	43,408,768	43,797,888	44,178,048
Receivers			
N_1 NAKs	1,014	986	949
N_2 NAKs	10	5	15
N_3 NAKs	0	0	0
Total NAKs Sent	1,024	991	964
Communication Links			
Satellite Link Utilisation	98.31%	98.95%	98.23%
Average Return Traffic (bps)	3,631	3,536	3,408
Peak Return Traffic (bps)	3,857	3,783	3,664
Performance			
Transmission Time (sec.)	22.694	22.553	22.773
Effective Throughput (bps)	1,804,892	1,816,127	1,798,629

Table C.10. *Simulation results: network size = 6, base error rate = 0.0001 and file size = 5 Mbytes.*

	Model Number		
	S6-0001-512	S6-0001-640	S6-0001-800
Frame Size (bits)	4,096	5,120	6,400
Frame Error Rate	0.00320	0.00399	0.00499
Sender			
F_1 Frames	10,323	8,205	6,531
F_2 Frames	194	175	179
F_3 Frames	2	0	0
F_4 Frames	0	0	0
Total Frames Sent	10,519	8,380	6,710
Total Bits Sent	43,085,824	42,905,600	42,944,000
Total Data Sent (bits)	41,739,392	41,832,960	42,085,120
Receivers			
N_1 NAKs	194	178	180
N_2 NAKs	2	0	0
N_3 NAKs	0	0	0
Total NAKs Sent	196	178	180
Communication Links			
Satellite Link Utilisation	98.10%	98.87%	98.87%
Average Return Traffic (bps)	754	695	702
Peak Return Traffic (bps)	756	698	709
Performance			
Transmission Time (sec.)	21.445	21.189	21.207
Effective Throughput (bps)	1,910,041	1,933,077	1,931,513

Table C.11. *Simulation results: network size = 12, base error rate = 0.01 and file size = 5 Mbytes.*

	Model Number		
	S12-01-32	S12-01-64	S12-01-96
Frame Size (bits)	256	512	768
Frame Error Rate	0.01990	0.03940	0.05852
Sender			
F_1 Frames	320,000	106,667	64,000
F_2 Frames	68,781	40,800	32,906
F_3 Frames	1,568	1,983	2,603
F_4 Frames	32	89	165
Total Frames Sent	390,380	149,539	99,674
Total Bits Sent	99,937,280	76,563,968	76,549,632
Total Data Sent (bits)	49,968,640	57,422,976	63,791,360
Receivers			
N_1 NAKs	76,599	50,433	44,941
N_2 NAKs	1,573	2,002	2,656
N_3 NAKs	31	89	165
Total NAKs Sent	78,203	52,524	47,762
Communication Links			
Satellite Link Utilisation	98.87%	98.70%	98.52%
Average Return Traffic (bps)	126,776	110,959	100,740
Peak Return Traffic (bps)	153,200	151,301	149,806
Performance			
Transmission Time (sec.)	49.356	37.878	37.938
Effective Throughput (bps)	829,881	1,081,380	1,079,645

Table C.12. *Simulation results: network size = 12, base error rate = 0.005 and file size = 5 Mbytes.*

	Model Number		
	S12-005-64	S12-005-96	S12-005-128
Frame Size (bits)	512	768	1,024
Frame Error Rate	0.01985	0.02963	0.03931
Sender			
F_1 Frames	106,667	64,000	45,714
F_2 Frames	23,034	19,341	17,455
F_3 Frames	528	695	792
F_4 Frames	7	18	25
Total Frames Sent	130,236	84,054	63,986
Total Bits Sent	66,680,832	64,553,472	65,521,664
Total Data Sent (bits)	50,010,624	53,794,560	57,331,456
Receivers			
N_1 NAKs	25,538	22,705	21,664
N_2 NAKs	528	699	797
N_3 NAKs	7	18	25
Total NAKs Sent	26,073	23,422	22,486
Communication Links			
Satellite Link Utilisation	98.55%	98.54%	97.95%
Average Return Traffic (bps)	63,165	58,608	55,106
Peak Return Traffic (bps)	76,616	75,686	75,827
Performance			
Transmission Time (sec.)	33.037	31.988	32.661
Effective Throughput (bps)	1,239,825	1,280,497	1,254,075

Table C.13. *Simulation results: network size = 12, base error rate = 0.001 and file size = 5 Mbytes.*

	Model Number		
	S12-001-128	S12-001-160	S12-001-192
Frame Size (bits)	1,024	1,280	1,536
Frame Error Rate	0.00797	0.00996	0.01193
Sender			
F_1 Frames	45,714	35,556	29,091
F_2 Frames	4,230	4,051	3,874
F_3 Frames	33	45	49
F_4 Frames	0	0	1
Total Frames Sent	49,977	39,652	33,015
Total Bits Sent	51,176,448	50,754,560	50,711,040
Total Data Sent (bits)	44,779,392	45,679,104	46,485,120
Receivers			
N_1 NAKs	4,409	4,286	4,128
N_2 NAKs	33	45	49
N_3 NAKs	1	0	1
Total NAKs Sent	4,443	4,331	4,178
Communication Links			
Satellite Link Utilisation	98.34%	98.66%	98.10%
Average Return Traffic (bps)	14,026	13,832	13,281
Peak Return Traffic (bps)	15,435	15,433	15,139
Performance			
Transmission Time (sec.)	25.410	25.118	25.240
Effective Throughput (bps)	1,611,970	1,630,722	1,622,853

Table C.14. *Simulation results: network size = 12, base error rate = 0.0005 and file size = 5 Mbytes.*

	Model Number		
	S12-0005-192	S12-0005-256	S12-0005-320
Frame Size (bits)	1,536	2,048	2,560
Frame Error Rate	0.00598	0.00797	0.00995
Sender			
F_1 Frames	29,091	21,333	16,842
F_2 Frames	1,990	1,958	1,939
F_3 Frames	11	18	16
F_4 Frames	0	0	0
Total Frames Sent	31,092	23,309	18,797
Total Bits Sent	47,757,312	47,736,832	48,120,320
Total Data Sent (bits)	43,777,536	44,753,280	45,714,304
Receivers			
N_1 NAKs	2,042	2,037	2,039
N_2 NAKs	11	18	16
N_3 NAKs	0	0	0
Total NAKs Sent	2,053	2,055	2,055
Communication Links			
Satellite Link Utilisation	97.95%	97.95%	98.70%
Average Return Traffic (bps)	6,939	6,949	6,946
Peak Return Traffic (bps)	7,491	7,642	7,752
Performance			
Transmission Time (sec.)	23.806	23.796	23.806
Effective Throughput (bps)	1,720,554	1,721,262	1,720,592

Table C.15. *Simulation results: network size = 12, base error rate = 0.0001 and file size = 5 Mbytes.*

	Model Number		
	S12-0001-448	S12-0001-512	S12-0001-640
Frame Size (bits)	3,584	4,096	5,120
Frame Error Rate	0.00280	0.00320	0.00399
Sender			
F_1 Frames	11,852	10,323	8,205
F_2 Frames	409	372	351
F_3 Frames	1	0	2
F_4 Frames	0	0	0
Total Frames Sent	12,262	10,695	8,558
Total Bits Sent	43,947,008	43,806,720	43,816,960
Total Data Sent (bits)	42,377,472	42,437,760	42,721,536
Receivers			
N_1 NAKs	415	380	364
N_2 NAKs	1	0	2
N_3 NAKs	0	0	0
Total NAKs Sent	416	380	366
Communication Links			
Satellite Link Utilisation	98.89%	98.90%	98.90%
Average Return Traffic (bps)	1,578	1,450	1,398
Peak Return Traffic (bps)	1,604	1,476	1,423
Performance			
Transmission Time (sec.)	21.699	21.628	21.633
Effective Throughput (bps)	1,887,642	1,893,880	1,893,336

Table C.16. *Simulation results: network size = 3, base error rate = 0.01 and file size = 2 Mbytes.*

	Model Number		
	S3-01-64	S3-01-96	S3-01-128
Frame Size (bits)	512	768	1,024
Frame Error Rate	0.03940	0.05852	0.07726
Sender			
F_1 Frames	42,667	25,600	18,286
F_2 Frames	4,733	4,229	3,866
F_3 Frames	212	251	330
F_4 Frames	12	20	30
Total Frames Sent	47,624	30,100	22,512
Total Bits Sent	24,383,488	23,116,800	23,052,288
Total Data Sent (bits)	18,287,616	19,264,000	20,170,752
Receivers			
N_1 NAKs	4,931	4,490	4,190
N_2 NAKs	212	251	332
N_3 NAKs	12	20	30
Total NAKs Sent	5,155	4,761	4,552
Communication Links			
Satellite Link Utilisation	94.59%	94.44%	94.16%
Average Return Traffic (bps)	32,784	31,889	30,483
Peak Return Traffic (bps)	36,989	37,424	36,669
Performance			
Transmission Time (sec.)	12.586	11.952	11.954
Effective Throughput (bps)	1,301,727	1,370,866	1,370,573

Table C.17. *Simulation results: network size = 3, base error rate = 0.01 and file size = 10 Mbytes.*

	Model Number		
	S3-01-64	S3-01-96	S3-01-128
Frame Size (bits)	512	768	1,024
Frame Error Rate	0.03940	0.05852	0.07726
Sender			
F_1 Frames	213,333	128,000	91,429
F_2 Frames	24,438	21,094	19,742
F_3 Frames	980	1,346	1,608
F_4 Frames	45	82	133
Total Frames Sent	238,796	150,522	112,912
Total Bits Sent	122,263,552	115,600,896	115,621,888
Total Data Sent (bits)	91,697,664	96,334,080	101,169,152
Receivers			
N_1 NAKs	25,373	22,382	21,312
N_2 NAKs	981	1,352	1,617
N_3 NAKs	45	82	133
Total NAKs Sent	26,399	23,816	23,062
Communication Links			
Satellite Link Utilisation	98.96%	99.06%	98.80%
Average Return Traffic (bps)	35,013	33,442	32,290
Peak Return Traffic (bps)	38,061	37,305	37,297
Performance			
Transmission Time (sec.)	60.325	56.980	57.144
Effective Throughput (bps)	1,357,967	1,437,698	1,433,569

Table C.18. *Simulation results: network size = 6, base error rate = 0.001 and file size = 2 Mbytes.*

	Model Number		
	S6-001-160	S6-001-192	S6-001-256
Frame Size (bits)	1,280	1,536	2,048
Frame Error Rate	0.00996	0.01193	0.01588
Sender			
F_1 Frames	14,222	11,636	8,533
F_2 Frames	837	781	752
F_3 Frames	3	12	15
F_4 Frames	0	0	0
Total Frames Sent	15,062	12,429	9,300
Total Bits Sent	19,279,360	19,090,944	19,046,400
Total Data Sent (bits)	17,351,424	17,500,032	17,856,000
Receivers			
N_1 NAKs	851	806	781
N_2 NAKs	3	12	15
N_3 NAKs	0	0	0
Total NAKs Sent	854	818	796
Communication Links			
Satellite Link Utilisation	95.16%	95.62%	95.16%
Average Return Traffic (bps)	6,955	6,762	6,565
Peak Return Traffic (bps)	7,668	7,397	7,331
Performance			
Transmission Time (sec.)	9.892	9.748	9.773
Effective Throughput (bps)	1,656,253	1,680,636	1,676,369

Table C.19. *Simulation results: network size = 6, base error rate = 0.001 and file size = 10 Mbytes.*

	Model Number		
	S6-001-192	S6-001-256	S6-001-320
Frame Size (bits)	1,536	2,048	2,560
Frame Error Rate	0.01193	0.01588	0.01981
Sender			
F_1 Frames	58,182	42,667	33,684
F_2 Frames	4,019	3,938	3,810
F_3 Frames	51	73	61
F_4 Frames	0	0	2
Total Frames Sent	62,252	46,678	37,557
Total Bits Sent	95,619,072	95,596,544	96,145,920
Total Data Sent (bits)	87,650,816	89,621,760	91,338,624
Receivers			
N_1 NAKs	4,128	4,083	4,011
N_2 NAKs	51	73	61
N_3 NAKs	0	0	2
Total NAKs Sent	4,179	4,156	4,074
Communication Links			
Satellite Link Utilisation	99.22%	99.37%	99.09%
Average Return Traffic (bps)	7,115	7,088	6,889
Peak Return Traffic (bps)	7,570	7,657	7,623
Performance			
Transmission Time (sec.)	47.054	46.973	47.377
Effective Throughput (bps)	1,740,995	1,744,003	1,729,116