Rewriting (part of) the ATM specification in LOTOS

Patrick Bihan-Faou
&
Eleye Mahamad

Supervised by Prof Luigi Logrippo

July 23, 1995

Contents

List of Figures

List of Tables

Abstract

Today's networks are moving from a situation where there were many of disjoint specialized network design for very specific task to an integration of the maximum number of services into one general communication medium. In the future, as the frontier between each type of communication are fading away, we will need one efficient way to transfer any type of information. The ATM protocol (Asynchronous Transfer Mode) was design with this purpose. But the complexity of such a task is huge, and eventhough people have been working on that for many years now, many things are not yet designed and standardized.

Formal Description Techniques (FDTs) offer great benefits in the design of computer and communication systems. The purpose of FDTs is to avoid errors and ambiguities that can arise when using natural language descriptions for complex system, such as telecommunication systems. LOTOS (Language Of Temporal Ordering Specification) is one of the three FDTs developed within ISO in the eighties for the formal specification of open distributed systems.

The purpose of this project was first to understand how ATM works. The second and most important goal was then to try to specify formally one major aspect of it (the signalization) in order to test it. This specification will be written using the language LOTOS (Language Of Temporal Oredering Specification) and the tools developed at the University of Ottawa.

Chapter 1

Introduction

This report describes a LOTOS specification of The B-ISDN signaling protocol. The first part is an overview of the ATM protocol. The second part gives an overview of the B-ISDN signaling protocol. After words, Patrick describes his work on the SSCOP protocol and Eleye his work on the Q2931 protocol.

Why is it useful to have a LOTOS specification of a system such as ATM ?

A specification such as the one we have written has several useful features:

- it is abstracted and implementation-independent
- it constitudes on executable model of a system, similar to a flight simulation for a real aircraft
- it allows to perform validation checkers on the specification, before implementation starts. Are there any unintented deadlocks, were all cases considered and properly dealt with ? It is specially important to perform such validation checkers when new functionalities are added to existing system.
- test cases can be derived from the specification .

Several of these aspects will be dealt with in this paper, especially in section 4,5 and 6.

Chapter 2

A Quick Presentation Of ATM

In the run for integrated multimedia networks, several standards have been created. The *Asynchronous Transfer Mode* is one of them. But unlike other types of networks, its development has been the work of many different entities in different countries. After describing what are the main characteristics of ATM, we will present how ATM networks are structured and the organization of the ATM protocol also known as the B-ISDN Protocol Reference model. Finally, we will describe what are the different parties involved in the standardization process and the available documentation.

2.1 Basic principles of ATM

ATM has been created to enhance the functionalities of ISDN (*Integrated Service Data Network*) to higher bandwidths. This evolution is often designated by the acronym B-ISDN (*Broadband ISDN*). The major characteristics of ATM are:

- It is a packet switching system using *small, fixed sized* packets also known as cells.
- ATM is connection oriented: a connection must be established before any data transfer. If the necessary resources for a call are available they are reserved for it. If not the call is rejected.
- The cells contain enough information to be transported through the network. This information consist of virtual channels / paths identifiers

located in the cell header.

- No error control or flow is provided on link-by-link basis. Error control is omitted for the information in cells (but not for their headers) because of the low rates of modern transmission systems. End-to-end error control for data connections can be provided by terminal equipment

- A policing function is provided. This enables the network to ensure that the user is not sending or receiving cells at a greather rate than he has requested.

- The cells are small allowing easier and better multiplexing and a more efficient bandwidth use. It minimises packetizing delay at the sending terminal and queuing delays at the switching nodes.

All these new features and characteristics make ATM a very different protocol, but at the same time, some constraints apply. The main one is that ATM must remain compatible with ISDN, or at leat offer ISDN-like capabilities. The second one is that it must conform to the needs of large scale public networks (it was originally designed to replace the telephone network), but also private (and usually smaller) networks. These constraints associated to the wide range of applications that can run on ATM make the field of related topics really large, and the problems are multiple and different. In this project, we have decided to focus on different aspects of the signaling problem.

2.2 The Global Organization of ATM Networks

Because of the applicability of ATM to both LAN's and WAN's, different kinds of interface exist to identify the type of connection between two nodes, and the protocols they should follow. First we can identify two main kind of nodes in an ATM network:

- Customer End Points : These are the access points to the network.They can be equipments such as fax machines, computers, TV sets, phones, etc.

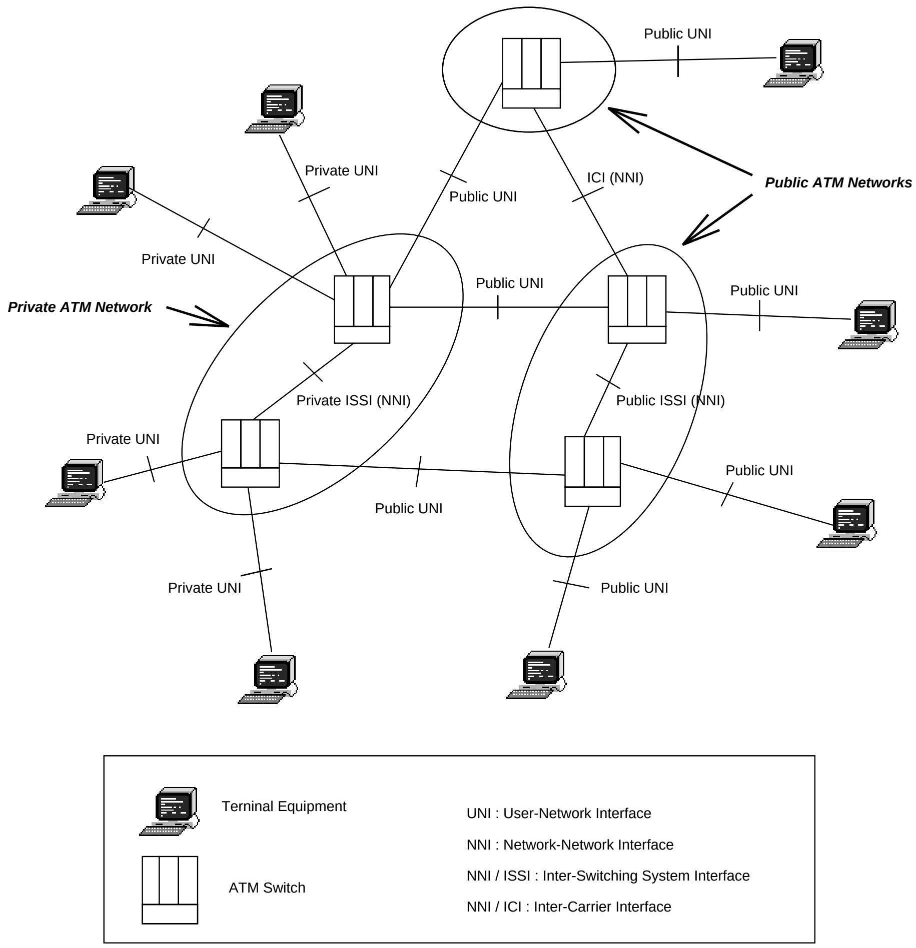


Figure 2.1: Global organization of an ATM network

- Intermediate Nodes / Switches : Their purpose is to relay the information from node to node through the network to make information exchange possible between two or more users (customer end points). Two kind of switches can be identified depending on their ownership: some of them are owned by carriers or service providers, some others may be private switches inside organizations.

All the nodes in the networks are linked with one-to-one links. There is no bus-like or other shared medium architecture in an ATM network. There is no predefined topology to be used in an ATM network.This organization is represented in figure 2.1.

The fact that there are two different types of nodes implies that there are also two main kind of interfaces:

- The User-Network Interface (UNI) which describes the interactions between the customer end points and the switches.

- The Network-Network Interface (NNI) which describes the interactions between switches.

2.2.1 The User-Network Interface (UNI)

The User-Network Interface was originally used to describe the interactions between a Customer Premise Equipment and an ATM switch. But with the work of ATM-Forum and the generalization of ATM networks, the term UNI refers now to any interface between an end-device and an ATM switch (public or private). There is a distinction between the *private UNI* and the *public UNI*. The private UNI refers to an interface between a stand-alone device and a private ATM switch. The public UNI is used to describe a CPE interface to a public ATM switch, where the CPE could be either a stand-alone device (a workstation) or a private ATM switch.

2.2.2 The Network Node Interface (NNI)

The NNI is used to describe a switch-to-switch interface (excluding private-to-public switch interface which are included in the public UNI). The NNI is also known as *Inter Switching System Interfaces* (ISSI) which can be further categorized into the public and private ISSI depending on the ownership of the switches.

The public ISSI itself is further divided into the *intra-LATA ISSI* when both switches belong to the same carrier and inter-LATA ISSI or *Inter-Carrier Interfaces* (ICI) when the two switches belong to different carrier (for example Bell and AT&T).

2.3 The ATM protocol : the B-ISDN Protocol Reference Model

The ATM Protocol Reference Model is based on the ISDN Protocol Reference Model (ISDN PRM) which like the OSI Reference Model organizes the communication functions into layers and describe the relation of these layers with respect to each other. The differences between the ISDN PRM and the OSI RM lies in that the ISDN PRM was especially designed for telecommunication purposes whereas the OSI RM is not dedicated to a particular type of network.

The ATM Protocol Reference Model is composed of three layers: the physical layer (PHY), the ATM layer (ATM), and the ATM Adaptation layer (AAL). The relationship between the lower layers of the OSI model and the AAL, ATM, and PHY layers of the B-ISDN PRM are not yet defined.

The ATM Protocol Reference Model is not only composed of layers, but also of planes. A plane is a layered stack of protocol entities with a single functional grouping. Three different planes are identified in the ATM PRM.

- The User Plane (U-Plane) whose purpose is to transfer user information transparently.
- The Control Plane (C-Plane) whose task is the transfer of information for the control of user plane connections (establishing/clearing connections, monitor the characteristics of a connection, etc.)
- The Management Plane (M-Plane)

The ATM Protocol Reference Model is shown in figure 2.2. Further informations on this subject can be found in [12] for the ISDN Protocol Reference Model and in [13] for the B-ISDN PRM. In the following sections, we will

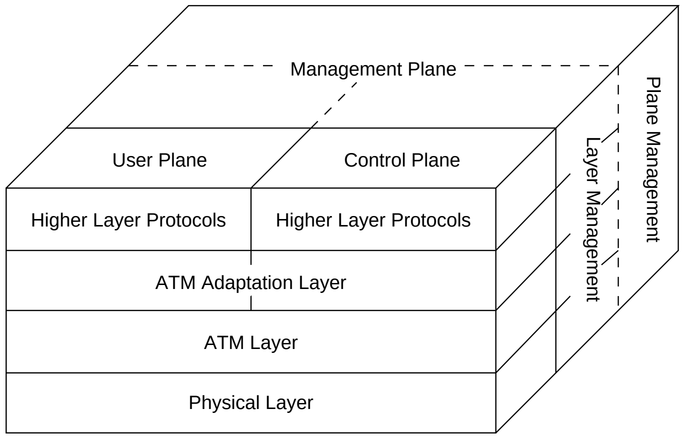


Figure 2.2: The B-ISDN ATM Protocol Reference Model

first detail the functions of the layers of the B-ISDN PRM, then the functions of the planes.

2.4 Vertical layering

The ATM protocol is divided in 3 distinct layers:

- The physical layer which defines everything that is related to the medium used and bit transfering.
- The ATM layer which defines the cell format and how to transfer the cells from one point to another.
- The ATM adaptation layer which maps the services offered by the ATM layer to the needs of the user.

In the following sections, we will discuss quickly each of these layers and their functions. Figure 2.3 summarizes the organization of the layers.

2.4.1 The Physical Layer

The physical layer of ATM is further composed of two sublayers:

Functions	Sublayer	Layer
Convergence	CS	AAL
Segmentation and reassembly	SAR	
Generic flow control Cell VPI/VCI translation Cell multiplex and demultiplex		ATM
Cell rate decoupling HEC header sequence generation / verification Cell delineation Transmission frame adaptation Transmission frame generation / recovery	TC	PHY
Bit timing Physical medium	PM	

Figure 2.3: The protocol reference model sublayers and functions

- the Physical Medium (PM) sublayer which supports pure medium dependent bit functions,
- and the Transmission Convergence (TC) sublayer that converts the ATM cell stream into bits to be transported over the physical medium.

The Physical Medium sublayer has to ensure a proper bit timing reconstruction at the receiver. Therefore the transmitting peer entity is responsible for the insertion of required bit timing information and line coding. This sublayer depends on the type of medium used. Different PM sublayers have been specified by the ITU-T and the ATM Forum.

The Transmission Convergence sublayer has to perform five basic functions:

- The first function is the transmission frame generation and recovery.
- The second function is the transmission frame adaptation. This function performs the actions needed to format the cell flow according to the payload structure of the transmission frame, and to extract this cell flow from the received transmission frame.

- Identify the cells within the stream of bits. This function is known as cell delineation. It uses the HEC field of the cell. The HEC field itself is generated by the TC sublayer.

- Correction and detection of errors in the cell header. Isolated one bit errors in the cell header are corrected, but as soon as multiple consecutive cells show header error, the TC sublayer tries to resynchronize the bit stream with the cell stream.

- The TC sublayer must also perform cell rate decoupling. This is done by inserting or deleting idle cells in the valid cells stream. The purpose of this is to adapt the cell rate to the payload capacity of the transmission system.

The physical layer also has some functions related to Operation and Maintenance (OAM) functions.

2.4.2 The ATM Layer

This is the heart of the ATM suite of protocols. This layer defines the cell format, and the method used to transfer a cell across the network. Some other functions are related to the insurance of the quality of service and traffic flow control and traffic shaping.

The cell format

An ATM cell is 53 bytes long, with 48 bytes for user data and 5 bytes for the header. The information contained in the 48 bytes is transparent for the ATM layer. The header is created an managed by the ATM layer. The information it contains is:

- The Generic Flow Control (GFC) which is only used at the UNI. This field is not present at the NNI.

- The Virtual Path and Virtual Circuit Identifiers (VPI/VCI) which are the routing information. The pair VCI/VPI is unique in a ATM node and only has local meaning. It is used to identify unambiguously a connection through the node. The VPI field size is increased at the NNI by using the bits assigned to the GFC field at the UNI.

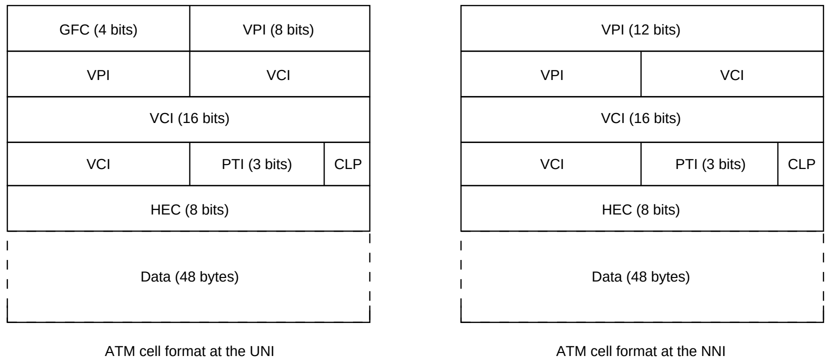


Figure 2.4: The ATM cell format at the UNI and at the NNI

- A Payload Type Indicator (PTI) This field is there to make the distinction between different types of cells and also to report the conditions encountered in the network (for example if congestion has been encountered or not).
- The Cell Loss Priority bit (CLP) which identifies if a given cell may be discarded or not by the network in case of congestion, or if the agreed flow parameters have been violated or not.
- An Header Error Control field (HEC) which is used to make sure that the header has not been corrupted.

The organization of a cell is described in figure 2.4.

The functions

The ATM layer is fully independent of the physical medium used to transport the ATM cells. The following main functions are provided by the ATM layer.

- Multiplexing and demultiplexing of cells of different connections into a single cell stream on a physical layer. This is done by using different VCI and/or VPI values for each connection.
- Translation of the cell identifier (the VCI and VPI fields) which is required when switching a cell from one physical link to another in an ATM switch.

- Providing the user with one or more Quality of Service (QoS) class for each connection.
- Management functions : the header of user information cells provides congestion indication and user to user indications. Some cells identified either by pre-defined VCI values or PTI codes may carry any kind of management information.
- Generation and extraction of the cell header.
- Implementation of a flow control mechanism on the UNI. This is supported by the GFC bits in the header.

2.4.3 The ATM Adaptation Layer

The ATM Adaptation Layer (AAL) enhances the service provided by the ATM layer by performing functions for the user, control and management planes. The functions performed by the AAL depend on the higher layer requirements.

Many different AAL's might be defined depending on the requirements of the higher layers. As a starting point, the ITU-T and the ATM Forum have defined 4 different AAL's, but this is an open field.

The functions

The AAL layer is subdivided in 2 sublayers : the segmentation and reassembly sublayer (SAR) and the convergence sublayer (CS).

The main function of the SAR sublayer is to decompose the higher layer information into blocks that can fit in a ATM cell at one end, and to perform the reverse operation at the other end.

The convergence sublayer performs functions such as message identification, time/clock recovery, etc. For some AAL types, the convergence sublayer has been further subdivided in a Common Part Convergence Sublayer (CPCS) and a Service Specific Convergence Sublayer (SSCS).

Some AAL service users may find the ATM service sufficient for their needs. In that case the AAL protocol may be empty. The only requirements is that the same AAL be present at both end of the communication link so that the data are exchanged in a consistant way.

The different AAL's

As a beginning, four types of AAL have been defined by the ITU-T for different traffic requirements. These different AAL's are defined in table 2.1.

These four AAL's were defined to correspond to the different types of traffic identified by the ITU-T. These traffic classes are:

Class A constant bit rate, highly synchronous. This correspond to the actual voice telephone service with no compression: the data are generated at a constant rate and have very strong timing relations between them.

Class B Variable bit rate, highly synchronous. Such characteristics would fit the transfer of compressed information such as voice or video that don't generate data at a constant bit rate, but where the timing relation between consecutive data is very strong.

Class C Variable bit rate, mainly asynchronous, connection oriented. This corresponds to the type of traffic observed on wide area computer networks such as X25. The data generated are mainly bursty and they are not sensitive to uncertain delays.

Class D Variable bit rate, no timing requirements, connectionless. This is typically the kind of traffic generated by computer local area networks. The connectionless aspect of this type of traffic is the main problem on ATM which is by definition connection oriented.

Originally the ALL 1 through 4 were specified, but after a while AAL 3 and AAL 4 were merged together for providing both class C and D traffic, and a new AAL (AAL 5) was created to offer a simpler way to have class C traffic.

2.5 Horizontal layering

As defined in the B-ISDN Protocol Reference Model, the ATM protocol has several planes. Each of them is responsible of a different set of functions. In the following sections we describe what are the different planes and their functions.

Traffic Type	Class A	Class B	Class C	Class D
Timing Relation	Required		Not Required	
Bit Rate	Constant	Variable		
Connection Mode	Connection Oriented			Connectionless
AAL Type	1	2	3/4, 5	3/4

Table 2.1: The different AAL types

2.5.1 The User Plane

The user plane is the one that will be used during the exchange of data between the end-nodes. Its main function is the transfer of *user information*. All associated mechanisms, like flow control and recovery from errors, are included. A layered approach is used within the user plane: it is the one that we described earlier.

2.5.2 The Control Plane

The control plane is responsible for the call control and connection control functions. These are all the necessary functions used to set up, supervise and release a call or connection. A layered approach is also used in this plane, and some of the layers are shared with the user plane. The common layers are the physical layer and the ATM layer. The AAL used in the control plane is based on the AAL 5 common part (i.e. the SAR sublayer and the CPCS sublayer), but two specific sublayers have been introduced leading to a new AAL which is called the Signalling AAL (S-AAL).

2.5.3 The Management Plane

The management plane includes two types of functions called layer management and plane management functions.

The plane management functions are responsible for providing coordination between all the planes. No layered structure is used within this plane.

The layer management has a layered structure. It performs the management functions related to resources and parameters residing in its protocol entities. The layer management handles the specific Operation and Maintenance (OAM) information flows for each layer (mainly the physical and the ATM layers). Another function of the layer management is to support a meta-signalling protocol. This protocol is used when the same ATM link is shared by several equipments to set dynamically the signalling channels that can be used by a particular equipment.

2.6 Available Documentation On ATM

2.6.1 Who works on ATM, who specifies it ?

Mainly two organization work on ATM now to set the standards: the ITU-T[1] (formerly CCITT[2]) and the ATM Forum. The ITU-T concentrates on proper standardization of ATM in the long run. But this process may slow down a bit the emergence of a usable ATM standard. Thus in 1991, the ATM Forum composed of private equipment manufacturers, telecom operators and users was created. The work of the ATM Forum concentrates on specifying Customer Premises Equipment and private switching whereas the work of the ITU-T is mainly concerned with the goals of network operators and national administrations.

Documentation can be found at those two sources for anything related to ATM. The ITU-T publishes a set of recommendation on every field of ATM. The ATM Forum released specifications for the Private and Public User-Network Interface and have intention to proceed with specifications in other areas.

2.6.2 A list of some major documents on ATM

Besides the official recommendations and standards from ITU-T or the ATM-Forum, several books can be found that describe ATM and give a global understanding of it. The most well known of them are

- "Asynchronous Transfer Mode, solution for the broadband ISDN" by Martin De Prycker (see [20]),

[1]ITU-T is a sector of the International Telecommunications Union (ITU.)

[2]CCITT stands for Comité Consultatif International des Télégraphes et Téléphones.

- "ATM Networks concepts, protocols and applications" by Rainer Händel, Manfref N. Huber and Stefan Schröder (see [9]),
- and "Réseaux ATM" by Marc Boisseau, Michel Demange and Jean-Marie Munier (see [3]).

The technical documentations include the ITU-T recommendations (mainly the I series) and the *ATM User-Network Interface specification v3.0* [6] by the ATM-Forum. The table 2.2 summarize the content of some of main ITU-T recommendations.

Number & title	**Contents**
I.121: Broadband Aspects of ISDN	Principle of B-ISDN
I.150: B-ISDN ATM Functional Characteristics	Description of the ATM layer functions and of the ATM cell header.
I.320: ISDN Protocol Reference Model	Definition of the global organization of the ISDN PRM : the layers, the planes and the interaction between them.
I.321: B-ISDN Protocol Reference Model and its Application	Extension of the recommendation I.320, and overview of the functions in the layers and sublayers.
I.361 / I.362 / I.363	Description and specification of the ATM layer and the AAL layer.
I.610: B-ISDN Operation and Maintenance Principles and Functions	Definition of the OAM principles and the OAM functions at the UNI.
Q.2100: B-ISDN Signaling ATM Adaptation Layer (SAAL) Overview Description	General description of the SAAL: the common base and the service specific sublayers (SSCOP and SSCF).
Q.2931: B-ISDN User-Network Interface Layer 3 Protocol	Specification of the signaling protocol at the UNI.

Table 2.2: Some ITU-T recommendations for ATM

Chapter 3

B-ISDN Signalling

In this chapter we describe the signalling protocol in a quick but complete way, then we identify the scope of our project .

3.1 B-ISDN Signalling Requirements

Signalling is needed to setup,maintain and release bearers between users. For B-ISDN , not only the characteristics of bearers will change -e.g. flexible bandwidth-, but also the number of bearers allowed between two users and the maximun number of users involved in one call. Two ways of bandwidth allocation will be offered.

Peak allocation : The bandwidth needed for a given bearer is specified. The specified value indicates the maximun bandwidth.

Variable allocation: Not only the maximum bandwidth is specified but also the average . This way of allocation is attractive for applications that need a variable bitrate.

The following new services will be supported by B-ISDN Signalling protocol:

multi-party: The number of users involved in one can be greater than two.

Multi-connection: The number of connections between two users can be greater than one.

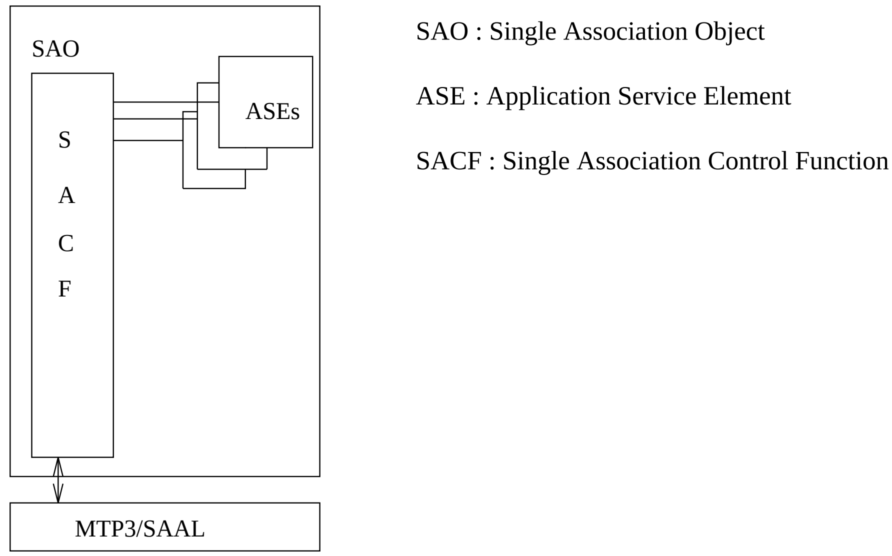


Figure 3.1: Protocol structure as used for Release 2 and 3 BAP protocols

service negotiation: In B-ISDN, the establishment of the association between the users involved and the establishment of bearers between these users, can be separated. This means that the users involved may exchange information before bearers are established.

Distribution: Allowing the distribution of information to many subscribers. This services could be used for broadcasting radio and television programs.

In order to have ATM-based networks available as early as possible, the ITU-T has decided to build the signalling protocol in three steps. Table 3.1 summarize the features that will be introduced at each release. In this table, different aspects are considered for each one of the release.

The signalling requirement for Release 1 is related to the bandwidth of bearers which is not restricted but ranges from 1 Kbit/s to 150Mbit/s. For the transport of B-ISDN information, the ATM is selected.Above the ATM, the Signalling ATM Adaptation Layer (SALL) is used .

Most of the new signalling requirements imposed by B-ISDN, are supported by Release 2 and 3. The Release 2 and 3 signalling requirements do not have an impact on the layers involved in the transporting and routing signalling information. The impact of the Release 2 and 3 signalling requirements on the signalling protocol however, is very large. First a separation of call control and connection is required, second a more flexible protocol structure has to be adopted. A Structure has been choosen for the Release 2 and 3 protocols -e.g Broadband Application Part (BAP)- which offers the flexibility of adding new protocols and new relations between them. This structure is based on the OSI Application Layer Structure (OSI ALS). A simplified drawing of this structure is given in Figure 3.1. The OSI ALS has the folowing advantage: Before a dialogue between two entities is established, the context of the dialogue is negotiated. The context of the dialgue, called the application context, identifies the set of protocols needed for that dialogue and the coordination functions needed between these protocols. The coordination functions are represented by the single Association Control Function (SACF) block in Figure 3.1. The protocols needed are represented by the Application Service Element (ASE) blocks.

For this project we worked with the release 1 of the signalling protocol as described in the ITU-T recommendations. We will describe how signalling works and then describe our work.

3.2 How does the signalling protocol work ?

The signalling protocol is part of the *Control Plane*. It is handled in a different way depending on the type of node concerned. In this section we will describe the two possibilities.

3.2.1 On the User Side

Depending on the type of configuration (point-to-point or point-to-multipoint), the signalling will involve different mechanisms.

In point-to-point configurations, there is only one signalling entity on the user side. Thus only a single permanent link (combination of VCI and VPI) is needed between the two points. This link is used for call offering, establishment and release.

In point-to-multipoint configurations, there may be several terminals at the user side. In this case, meta signalling is needed to establish the con-

Release	**Functionalities**
1	- Constant bit rate, Connection-oriented service - Point-to-point connections, Indication of peak bandwidth - Peak rate allocation - Interworking with 64 kbit/s ISDN, Meta-signalling
2	- Variable bit rate, Connection-oriented service, Quality of service indication by user - Point-to-multipoint connections, Negotiation and renegotiation of bandwidth - Bandwidth allocation based on traffic characteristics - Supplementary services
3	- Multimedia and distributive service, Quality of service negotiation - Broadcast connections

Table 3.1: Characteritics of the different implementation phase of the ATM signalling protocol

nection over which the signalling protocol will occur between two points. This meta signalling protocol is part of the OAM functions handled by the *Management Plane*.

The signalling protocol at the user side is derived from the one used in ISDN networks. The definition of the DSS2 (Digital Subscriber Signalling System No. 2) can be found in the recommendation Q.2931 from ITU-T [30].

3.2.2 On the Network Side

On the Network Side, the protocol defined by ITU-T for the Phase 1 is derived from SS7 (Signalling System 7). This protocol is described in the recommendations Q.2761 to Q.2764 [26, 27, 28, 29]. This choice was made in order to introduce ATM networks quickly. But it is obvious that more specific ATM-based protocols will be needed in order to take care of all the capabilities of ATM.

Only point-to-point configurations are defined for the NNI, so meta-signalling is not required on those nodes.

3.2.3 Where to use what ?

The type of the nodes that are connected together defines the protocol that is to be used between them. Table 3.2 describes what are the possible configurations.

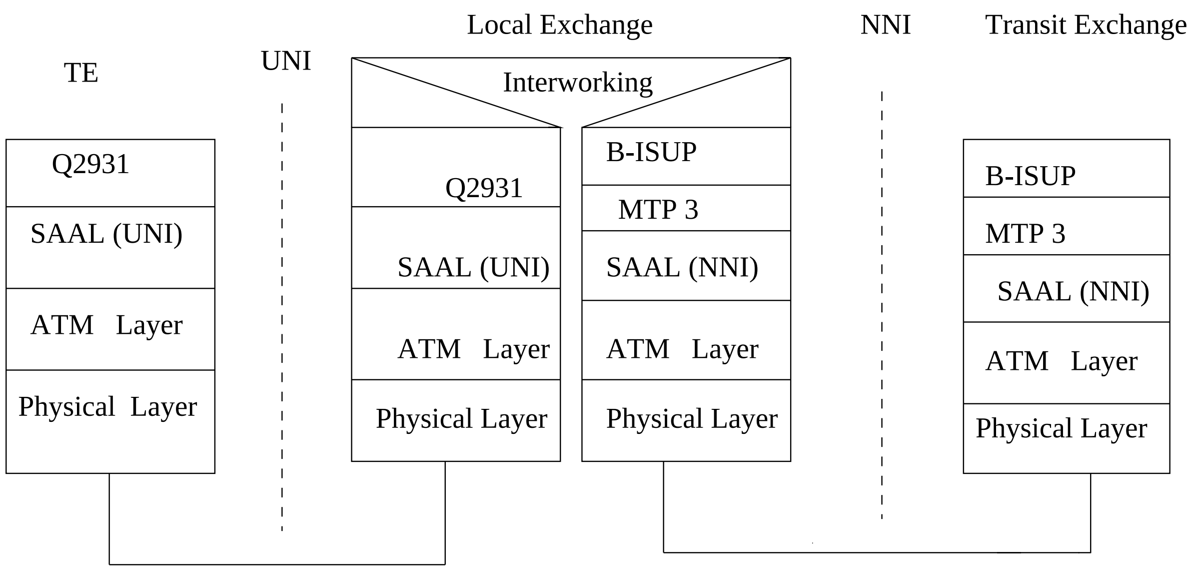
The different types of interfaces are:

- The Private / Public UNI : They use the DSS2 protocol.
- The Private NNI : which uses the SS7 Protocol.
- The B-ICI (Broadband Inter-Carrier Interface) which uses the SS7 protocol.

3.3 More details on the protocols

In this section, we will try to describe the protocols that are present for a connection between two terminal equipments . These protocols, knows as Q2931 and B-ISUP are based on existing protocols. They both use ATM as

Architecture of B-ISDN Signalling



The B-ISDN Release I Signalling Stack at the UNI and NNI

Q2931 : Signalling protocol at the UNI.
B-ISUP : Signalling protocol at the NNI.

The other elements of the Stack provide the transport of the signalling information.

Figure 3.2: The B-ISDN Release I Signalling Stack at the UNI and the NNI

Between	End-Point Equipment	Private ATM network	Public ATM Network
End-Point Equipment	Private UNI	Private UNI	Private UNI
Private ATM Network	Private UNI	Note	Public UNI
Public ATM Network	Public UNI	Public UNI	B-ICI

Note Private ATM networks may use the *private UNI* signalling, but after Phase 1, specific features will be added to the *private NNI* signalling protocol.

Table 3.2: Reference Configuration for the signalling protocols

the communication medium. This implies that there is an *ATM Adaptation Layer* especially designed for this purpose. It appears that because the needs of both protocols are similar, the AAL is almost the same in both cases.

3.3.1 Common part for the two protocols : the S-AAL

The *Signalling ATM Adaptation Layer* has to map the functions and capabilities of the ATM Layer to the needs of the signalling protocols. The capabilities involved are of class C (connection oriented without strong timing relations) type, but some modifications have to be provided:

- The ordering of the submitted information has to be preserved.
- It must be possible to send "connectionless" information.

None of the standard ATM Adaptation Layers provided these capabilities, so a new AAL had to be designed for the signalling protocols. Because the basic capabilities required by the S-AAL are already provided by both the AAL-4 and the AAL-5, the Signalling AAL is based on them, and we can distinguish a *Common Part* (CP) which can be either the AAL-4 or the AAL-5, and a *Service Specific Part* (SSP). The AAL-5 has been finally chosen for the CP because of the small overhead it introduces.

As any AAL, the functionalities of the S-AAL are divided in two categories:

- The *Segmentation And Reassembly* part (SAR) whose purpose is to split messages into ATM cells payload and to reconstruct those messages from the received cells.
- The *Convergence Sublayer* (CS) whose task is to build the PDUs corresponding to the primitives present at the S-AAL SAP. This sublayer is further divided into a common part (CPCS) (which is the AAL-5's CS) and a service specific part (SSCS).

Thus we can see that the S-AAL Common Part is composed of the SAR and the CPCS described in the AAL-5. The Service Specific Part is then just a sublayer of the Convergence Sublayer.

The SSCS is again divided in two parts:

- The *Service Specific Connection Oriented Protocol* (SSCOP). This protocol provides mechanisms to establish and release connections, exchange signalling informations, and also exchange OAM information without dedicated connections.
- The *Service Specific Coordination Function* (SSCF) which depends on the interface (UNI or NNI). Its task is to map the capabilities of the SSCOP to the SAP of the S-AAL for the different interfaces.

The resulting structure of the S-AAL is shown in figure 3.3. More details about the Service Specific Part are available in the ITU-T' recommendations [21, 23, 24, 25].

3.3.2 The Q.2931 Protocol: Digital subscriber SIgnalling System 2 (DSS2)

The Q.2931 protocol is used at the *User-Network Interface*. It is based on the Q.931 ISDN Signalling protocol and the Q.933 Frame Relay Signalling protocol. Its purpose is to accomodates the requirements for the release 1 of the signalling protocol. One of the main constraints is to obtain a working product as fast as possible. Its definition by the ITU-T is supposed to be finished in 1996. This, along with the requirement of being interoperable with the ISDN is why the ITU-T used the work done for both ISDN and frame relay to build this protocol.

3.3.3 The Broadband ISDN User Part Protocol (B-ISUP)

In a similar way as for the DSS2 (Signalling at the UNI), the B-ISUP is an adaptation of the ISDN User Part (ISUP) which is the signalling protocol for the Network Node Interface (NNI).

The B-ISUP protocol operates over the *Message Transfer Protocol 3* (MTP3), identical to the one used in *Signalin System 7* (SS7) for out-of-band signalling. This will allow B-ISDN network signalling the flexibility to operate over existing signalling networks or directly over new ATM networks.

More information about this protocol can be found in the ITU-T recommendations Q.2761 through Q.2764 [26, 27, 28, 29]. The recommendation

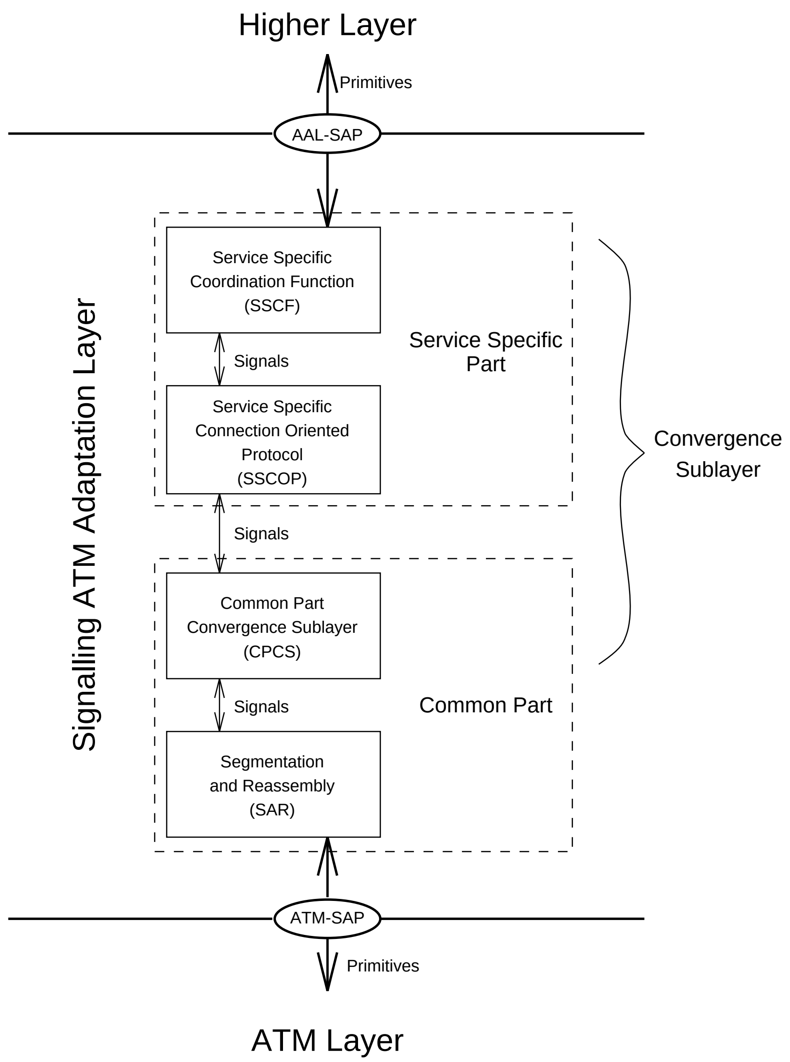


Figure 3.3: Organization of the S-AAL layer

Q.2140 [25] specifies the SSCF function which is on top of the SSCOP at the NNI.

3.3.4 Addressing and Routing

Finally there is a very important issue in the signalization problem:

- How do we identify a particular node ?
- and, How can we go to a defined node ?

These two problems are solved by using a proper addressing through the network, and appropriate routing algorithms. These two problems have already been encountered in the existing switched networks, and many solutions are available.

Currently two types of ATM addressing plans are being considered in the standard bodies to identify an ATM UNI address: the Network Service Access Point (NSAP) format defined in ISO 8348 [**?**], and the ITU-T E.164 standard [4].

This topic is out of the scope of this project, and we will not develop it any further.

3.4 The Project

In this section, we try to identify the work that can be done on the signalling protocol.

3.4.1 Validation of the SSCOP

Because the first release of the signalling protocol is mainly an adaptation of existing signalling protocols to ATM and some of its characteristics, the most important result of the work done by the standardization bodies is the definition of the *Signalling ATM Adaptation Layer* (S-AAL). Its purpose is to provide a common reliable way to transport the messages of the different signalling protocols.

In this work for a common stable S-AAL, the most important part is the design of the SSCOP protocol. Thus it is interesting to study this protocol and to verify that all the requirements that are associated with it are met. This work is detailed in 4.

3.4.2 Validation of the *Q2931*

Because the specification of the signalling protocol at the UNI prsented by the ITU-T *Q.2931* and the *ATM Forum UNI version 3.0* are both in a narration text, it is interesting to build a formal description to study this protocol and verify that all the requirements that are associated with it are met. This work is detailed in 5.

3.4.3 Future work

As we are only at the first release of the signalling protocol, and because the study period for it is not even finished, we can infer that a lot of work can still be done on this subject.

One of the big issues will be to test the compatibility of the different releases together: the protocol for the release 2 should be able to accomodate the messages of release 1 and provide at least the same level of services.

On the other hand, validation will always be needed to verify that the capabilities required by the new protocols are met.

As we can see, even if the signalling problem is not that complicated, the fact that we cannot predict the type of applications that will be used over ATM in the future, and the need to have protocols that suit our present needs as fast as possible, make it an open problem.

Chapter 4

The SSCOP Protocol

The *Service Specific Connection Oriented Protocol* (SSCOP) is a new protocol designed for the *Signalling ATM Adaptation Layer* (S-AAL) in order to provide the assured and unassured data delivery services needed by the signalling entity.

In this chapter I will first go through a description of the protocol and its particularities, then detail how I have represented it in LOTOS and finally present the results of my work on it.

4.1 Description of the SSCOP

The SSCOP is a peer to peer protocol which resides in the Service Specific Part (SSP) of the S-AAL. It is used to transfer variable length Service Data Units (SDUs) between users of SSCOP. This transfer of information can be done in assured or unassured mode. The difference between these two modes resides in the fact that a connection must be established for the first one and not for the second one. The assured mode also preserve the sequence integrity of the submitted SDUs.

The protocol also provides other services such as error reporting or keep alive functions. The full set of functions included in the SSCOP is:

- transfer of user data with sequence integrity,
- error correction by selective retransmission,
- flow control,

- connection control,
- error reporting to layer management,
- connection maintenance in the prolonged absence of data transfer,
- local data retrieval by the user,
- error detection of protocol control information, and
- status reporting.

The full description of this protocol can be found in recommendation Q 2110 [22] from ITU-T. In this section, I will first detail each of the functions provided by the protocol, then describe its interactions with the rest of the S-AAL, and finally explain how it works.

4.1.1 Detailed Description of the Functions

Sequence Integrity This function preserves the order of SSCOP SDus that were submitted for transfer in assured (connected) mode.

Error Correction by Selective Retransmission Through the sequencing mechanism, the receiving SSCOP entity can detect missing SSCOP SDUs. It then asks for retransmission to the sending SSCOP entity. This function corrects sequence errors through retransmission.

Flow Control This function allows an SSCOP receiver to control the rate at which the peer SSCOP transmitter entity may send information.

Error Reporting to Layer Management Whenever an error occurs in the protocol, it is detected and an appropriate message is sent to the Layer Management.

Keep Alive This function verifies that the two peer SSCOP entities participating in a connection are remaining in a connected state even if non data transfer occurs for a prolonged time.

Local Data Retrieval This function allows the local SSCOP user to retrieve in-sequence SDUs which have not yet been released by the SSCOP entity.

Connection Control This function performs the establishment, release, and resynchronization of an SSCOP connection. It also allows the retransmission of variable length user-to-user information without a guarantee of delivery.

Transfer of User-Data This function is used for the conveyance of user data between users of the SSCOP. SSCOP supports both assured and unassured data transfer.

Protocol Error Detection and Recovery This function detects and recovers from errors in the operation of the protocol.

Status Reporting This function allows the transmitter and receiver peer entities to exchange status information.

4.1.2 Interaction with the other components of the S-AAL

As mentioned earlier, the SSCOP is contained in the Service Specific Part of the S-AAL. More specifically, it is a component of the Service Specific Convergence Sublayer (SSCS). Its purpose is to provide services to the signalling entity, using the basic AAL-5 functions. The functions provided by the SSCOP are then mapped to the primitives defined for the S-AAL. Thus the SSCOP has to interact with two entities.

On the lower boundary, we find the CPCS (Common Part Convergence Sublayer) defined for the AAL-5, from which it uses the unassured data transmission functions. The description of the CPCS and the AAL-5 can be found in the recommendation I 363 from ITU-T [17].

On the higher boundary, the SSCF (Service Specific Coordination Function) maps the primitives of the SSCOP to the primitives of the S-AAL. Two different SSCFs are defined depending on the interface (either UNI or NNI). Their definition can be found in the ITU-T recommendations Q 2130 [25] for the UNI and Q 2140 [25] for the NNI.

Because each of these entities (the SSCOP, the SSCF, and the CPCS) are not layers by themselves, but rather sublayers of the S-AAL, the term "signal" instead of "primitive" is used to describe the interactions between them.

Parameter	CPCS-UnitData.invoke	CPCS-UnitData.signal
LP	0	ignored
CI	0	ignored
UU	0	ignored
ID	SSCOP PDU	SSCOP PDU

Table 4.1: Parameter of the signals present between the CPCS and the SSCOP

Finally, the SSCOP communicates with the Layer Management to report errors in the protocol and transfer Layer Management data between SSCOP users.

Interaction with the CPCS

The actual specification of the SSCOP assumes the CPCS message mode of operations. Other functioning modes are for further study.

The signals present between the SSCOP and the CPCS are CPCS-UnitData.invoke and CPCS-UnitData.signal. The PDU to be sent or received are situated in the ID (Interface Data) parameter of the signals. The other parameters for those two signals are set to zero at the transmitter side and ignored at the receiver side. Table 4.1 shows a summary of the use of the parameters.

Interaction with the SSCF

Many different signals are defined between the SSCF and the SSCOP to access the functions of the SSCOP. Here is a quick description of them:

AA-ESTABLISH These signals (request, indication, response and confirm) are used to establish a point-to-point connection for assured data transfer between the peer user entities.

AA-RELEASE (request, indication and confirm) signals are used to terminate a point-to-point connection.

AA-DATA (request and indication) signals are used for the assured point-to-point transfer of SDUs between the peer user entities.

Generic Signal Name	Type			
	Request	Indication	Response	Confirm
AA-ESTABLISH	User Information Buffer Release	User Information	User Information Buffer Release	User Information
AA-RELEASE	User Information	User Information Source		No Parameter
AA-DATA	User Information	User Information Serial Number		
AA-RESYNC	User Information	User Information	No Parameter	No Parameter
AA-RECOVER		No Parameter	No Parameter	
AA-UNITDATA	User Information	User Information		
AA-RETRIEVE	Retrieval Number	User Information		
AA-RETRIEVE COMPLETE		No Parameter		

Table 4.2: Parameters of the signals present between the SSCOP and the SSCF

AA-RESYNC (request, indication, response and confirm) signals are used to resynchronize an established SSCOP connection.

AA-RECOVER (indication and response) signals are used during recovery of protocol errors.

AA-UNITDATA (request and indication) signals are used for the non-assured, broadcast and point-to-point transfer of user information.

AA-RETRIEVE (request and indication) signals are used to retrieve SDUs submitted to the SSCOP for transmission , but not yet released.

AA-RETRIEVE COMPLETE (indication) signal is used to indicate that there are no additional SDUs to be returned to the SSCOP user.

The different parameters used for those signals are summarized in table 4.2.

Interaction with the SSCS Layer Management

The interface between the SSCOP and the SSCS Layer Management entity is simple. Only two different signals have to be defined: one for reporting

Generic Signal Name	Type	
	Request	Indication
MAA-ERROR		Code Count
MAA-UNITDATA	User Information	User Information

Table 4.3: Parameters fo the signals present between the SSCOP and the Layer Management entity

errors and one for data transmission.

MAA-ERROR (indication) signal is used to report SSCOP protocol errors and certain events to the Layer Management entity.

MAA-UNITDATA (request and indication) signals are used for the non-assured, broadcast and point-to-point transfer of SDUs between SSCOP and peer Layer Management entities.

The paramters used with these signals are described in table 4.3.

4.1.3 SSCOP Internals

After having described the interaction of the SSCOP with its environment, I will describe some of its internal characteristics: its states, its variables, its timers and finally I will present a diagram showing its functioning.

The States

The SSCOP has 10 different states depending on the interaction that have taken place between the SSCOP and its environment. These states are:

Idle (1) In this state, no connection is established. The only possible actions are establishment of a connection or transfer of unassured data.

Outgoing Connection Pending (2) This state is initiated when the user of the SSCOP tries to establish a connection with a peer entity. This states is left when it receives an answer from the peer entity (positive or negative) or after a certain amount of time.

Incoming Connection Pending (3) The protocol is in this state when a peer entity tries to establish a connection with the SSCOP user. A this state, the protocol waits for an answer from the user before moving to another state.

Outgoing Disconnection Pending (4) This state is reached when the user requested a disconnection. The protocol then waits for an acknowledgment form the peer entity.

Outgoing Resynchronization Pending (5) This state is reached when the user requests for a resynchronization of the connection.

Incoming Resynchronization Pending (6) The protocol goes in this state when the peer entity requested a resynchronization of the connection.

Outgoing Recovery Pending (7) This state is initiated by the SSCOP itself, when the recovery of lost data between the two peer entities is necessary.

Recovery Response Pending (8) In this state, the SSCOP has notified its user that a recovery is in progress and waits for a response either from the user or from the peer entity.

Incoming Recovery Pending (9) The protocol goes in this state when the peer entity requested recovery of data.

Data Transfer Ready (10) In this state, a connection is established and the transmission of assured data can take place.

In each of these states, the unassured transfer of data can take place. A disconnection is also always possible either from the user or from the peer entity.

The PDUs

In order to communicate with the peer SSCOP entity, a set of PDUs has been defined. This set covers the functions and the primitives of the protocol. There are 15 different type of PDUs that can be split in 7 groups depending on their functionality. Table 4.4 describes the PDUs.

Functionality	PDU Name	Description
Establishment	BGN	Request initialization of the peer SSCOP entity.
	BGAK	Acceptance of connection by peer entity
	BGREJ	Connection reject by peer entity
Release	END	Disconnect command
	ENDAK	Disconnect acknowledgment
Resynchronization	RS	Resynchronization command: used to resynchronized the buffers and variables.
	RSAK	Resynchronization acknowledgement
Recovery	ER	Recovery command: used to recover from protocol errors
	ERAK	Recovery acknowledgment
Assured Data transfer	SD	Transfer sequentially numbered PDUs in assured mode.
	POLL	Transmitter state information with request for receiver state information.
	STAT	Solicited receiver state information (in answer to a POLL PDU).
	USTAT	Unsolicited receiver state information.
Unacknowledged Data Transfer	UD	Unnumbered user data.
Management Data Transfer	MD	Unnumbered management data

Table 4.4: Description of the SSCOP PDUs

Each of these PDUs can have a variable length. But the size must be less or equal to 65528 at the most. The length must also be a multiple of 4 octets, thus a padding may be used when necessary.

The Variables, Buffers and Timers

Three different sets of variables are defined.

The receiver set contains information about the number of the next SD PDU to receive, the size of the window etc.

The sender set contains the same kind of information as the receiver set.

The last set is composed of parameters for the protocol, such as "how long should we keep an idle connection alive ?" etc.

Associated with the variables, there are some buffers that are used to maintain the ordering of the received PDUs and make sure that all the information has been transfered.

Finally, some timers are defined to detect suspicious inactivity on the line.

4.1.4 Automata for the SSCOP

The figure 4.1 shows the interaction between the SSCF and the SSCOP. The interactions between the CPCS and the SSCOP are not shown on this diagram in order to keep it clear and understandable. As a consequence the state "Recovery Pending (8)" covers in fact two states of the protocol ("outgoing Recovery Pending (8)" and "Incoming Recovery Pending (9)") which cannot be distinguished at the boundary between the SSCF and the SSCOP.

Also as the transfer of information in the unacknowledged mode is available at any time in the protocol, the associate primitives are not represented in the diagram. Anyway these exchanges of information do not affect the state of the protocol.

4.2 Specification of the SSCOP in LOTOS

The recommendation Q 2110 [22] contains a SDL description of the protocol. This is what I used to produce the LOTOS specification.

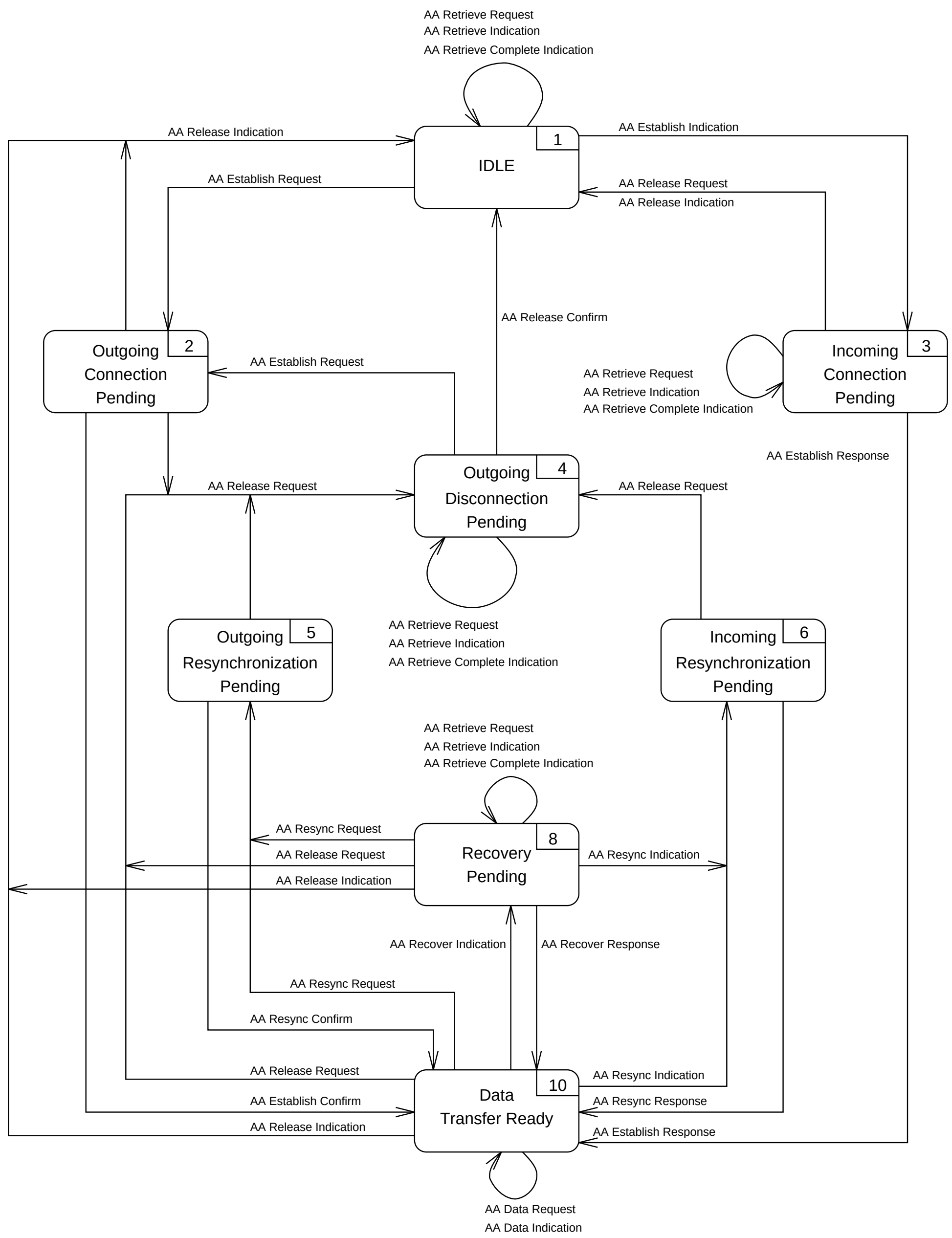


Figure 4.1: State Transition Diagram for Interactions between the SSCF and the SSCOP

The first goal of this specification was to be as complete as possible with respect to the recommendation. This means that as few details as possible were kept out of the specification. The second objective was to make it easy to reuse and to integrate with other specification. This is important in order to build more complete applications later.

4.2.1 Organization of the Specification

Because it has been derived from a SDL description, the specification is written in state oriented style. I chose the explicit state oriented style because it would be easier to read.

The Gates

I chose to use a very small set of gates for the specification. In fact I reduced it to the number of interfaces that are present in the SSCOP. The gates are:

SSCF for the interactions between the SSCOP and the SSCF

CPCS for the interactions between the SSCOP and the CPCS

LM for the interactions between the SSCOP and the Layer Management

SSCOP_I for the actions that are internal to the SSCOP

With all the gates, the first data experiment tells what is the primitive that was used on the interface. Then all the parameters for the primitives are given.

Between the CPCS and the SSCOP sublayers

All the signals exchanged between the CPCS and the SSCOP sublayers are passed through the gate "CPCS". The signals used are:

- CPCS_UnitData_invoke with a PDU as argument
- CPCS_UnitData_signal with a PDU as argument

Between the SSCOP and the SSCF sublayers

All the signals exchanged between the SSCOP and the SSCF sublayers are passed through the gate "SSCF". The signal used are:

- AA-Establish (request / indication / response / confirm)
- AA-Release (request / indication / confirm)
- AA-Data (request / indication)
- AA-Resync (request / indication / response / confirm)
- AA-Recover (indication / response)
- AA-UnitData (request / indication)
- AA-Retrieve (request / indication)
- AA-Retrievecomplete (indication)

The coding of the PDUs

The PDUs exchanged between two entities of the SSCOP are coded in the abstract data type "SSCOP_PDU". This data type consists of a set of contructors, each of them corresponding to one type of PDU.

4.2.2 The resulting LOTOS specification outline

```
Specification SSCOP_Specification [SSCF, CPCS] : noexit

Behaviour
(
   SSCF_sublayer [SSCF]
     |[SSCF]|
   SSCOP_sublayer [SSCF, CPCS]
     |[CPCS]|
   CPCS_sublayer [CPCS]
)
Where
```

```
   Process SSCOP [SSCF, CPCS] : noexit :=
   hide SSCOP_I in
   (
      SSCOP_internal [SSCOP_I]
         |[SSCOP_I]|
      SSCOP_states [SSCOP_I]
   )

   ...

   EndProc
...

EndSpec
```

As much as possible, the abstract data types and the process definitions that are "private" are hidden from the rest of the specification.

4.2.3 Justification of the Style of the Specification

It has to be noted that the explicit State Oriented Style is not the usual style for a LOTOS specification. We have chosen it because of the format of the available documentation and because everything is defined with respect to states in the protocol.

In LOTOS, however, normaly one uses no direct representation of the states, rather traces and uses cases are identified , and they are integrated in the specification (resource-oriented style).

4.3 Utilization of the Specification

Once the specification is written, it has to be used. The two main uses I found for the SSCOP specification are:

- Simulation of the protocol. This can be interesting because it is a new protocol, and no real study has been made on it.

- Derivation of Test Cases. Along with the description of the protocol, there is *Protocol Implementation Conformance Statement* (PICS) Pro Forma in the recommendation Q 2110. This PICS states some capabilities that should be provided by an implementation of the SSCOP in order to be considered valid. But this list of capabilities doesn't come with a lot of indications. That is why I tried to generate some test sequences that would exercise those capabilities.

For the first part, a tool like Random Walk seems appropriate, because it can go through all the parts of the specification with no predefined order. This way the "execution" of the simulation is free of the "operator effect". What I call the operator effect is what makes a person executing a sofware see only what works, by executing sequences of actions that are so meaningfull that they will never fail.

The second part of the utilization can be better done with more complex tools like LMC or GOE. In this case we try to express the capabilities with a formula, and then using the tools, we try to verify them.

4.3.1 Simulation

To execute the simulation of the protocol with Random Walk, I had to simulate a network with two entities of the SSCOP working together.

This is done by using the construct shown in figure 4.2. The network is simulated by an abstraction of the CPCS. The actions possible at the second entity of the SSCOP are limited by a stub of the SSCF that will not synchronize on any actions that should be generated by the user of the SSCOP.

Unfortunately, the results obtained with Random Walk are not very good. This is a consequence of the fact that at any time a connection can be released. Then because of the very fine granularity of the specification, the number of actions necessary to establish a connection is very high. The consequence is that even with a high number of actions, it is almost impossible to get a connection set up and data exchange reached. Unfortunatly, all the trace generated by Random Walk in this configuration have the following look:

```
SSCF ! AA Establish request
```

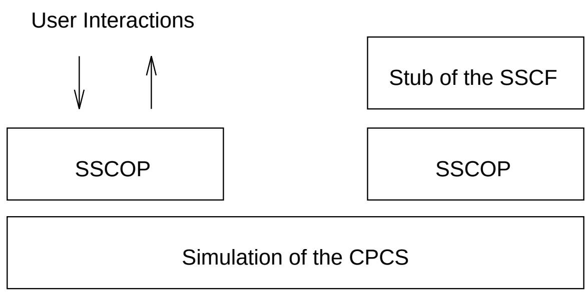


Figure 4.2: Testing construct for using Random Walk with the SSCOP specification

```
CPCS ! UnitData invoke ! BGN
...
SSCF ! AA Release request
...
```

These are not good results for a simulation.

4.3.2 Verification of Properties

Verification of properties is useful to check that the protocol meets its requirements. Such a verification is usefull first when a design of the protocol is built with enough details, then when an implementation of the protocol is realized.

The verification of properties at design stage can help us establish some test-sequences that will be used with the implementation to check that it corresponds to the design.

What are the properties to be checked

The conformence requirements are divided into three categories in the recommendation Q 2110 [22]. First the *Protocol Capabilities* (PC) section points out the basic requirments for the protocol and its behaviour. Then the *SSCOP PDU* (PD) section is there to verify the conformence of the PDUs used by the implementation and their description in the recommendation.

Finally, the *SSCOP System Parameters* (SP) section is a way to state the chosen value for some of the protocol parameters.

In this work, the only relevant section in the conformance document is the *Protocol Capabilities* section. This is where we have to define test sequences to determine whether or not a capability is implemented. These test-sequences, if expressed in an appropriate abstract way, will be reusable with thereal implementation of the protocol. The second section (PDUs) has no interest for us as we just have a description of the behaviour of the protocol and not a real implementation. The third section (System Parameters) can be of informative interest in order to set up the implementation under test in the same way the description in LOTOS was set up.

For the PC section, 16 points have been distinguished.

1. Does IUT support *Keep Alive* function ?
2. Does IUT support the *Local Data Retrieve* function ?
3. Does IUT support *SSCOP initiated error recovery* due to protocol error ?
4. Does the IUT recognize all of the messages regardless of state ?
5. (a) In the absence of protocol error, does the IUT support *assured data transfer with sequence integrity* ?
   (b) Does IUT support the sending of the *Unassured Data PDU* ?
   (c) Does IUT support the sending of the *Management Data PDU* ?
6. Does IUT support *user invoked re-synchronization* procedures ?
7. Does IUT support *the establishment procedures* for an SSCOP connection ?
8. Does IUT support *the release procedures* for an SSCOP connection ?
9. Does IUT support *polling after retransmission* ?
10. Does IUT support the *segmenting of STAT PDUs* ?
11. Can the IUT *initiate* SSCOP connection ?

12. Can the IUT *reject (BGREJ)* the establishment of a SSCOP connection form its peer ?

13. Does IUT support *error reporting* to layer management ?

14. Does IUT support the *Protocol Error Detection* function ?

15. When no SSCOP connection exists, is a connection established only upon receipt of a BGN PDU or a request from the SSCOP user ?

16. Does the SSCOP permit the conveyance of SSCOP User-to-User information between user of the SSCOP ?

How can we check them ?

Most of the properties presented here are trivial in the sense that one does not need special test sequences in order to check them. Many different test constructs may also be needed to verify the properties. This is due to the fact that the different properties have different scope.

Verifying with two peer entities

One of the possible constructs to exercise some properties is to have two peer SSCOP entities linked by a simulated CPCS sublayer. Each SSCOP entity has a driver running on top of it that will simulate a user (see figure 5.1). Finally the test has succeeded if the end of the sequence coded in the drivers (the SSCOP users) is reached. In order to check that, we can add a special gate telling that the sequence is finished.

For example, properties number 7 and 8, can be checked with the following test construct and drivers:

```
behaviour

hide I_SSCOP_1, I_SSCOP_2 in
(
   Test_Entity_1 [ SSCF_1 ]
      |[SSCF_1]|
   SSCOP_sublayer [ SSCF_1, CPCS_1, I_SSCOP_1 ]
      |[CPCS_1]|
   CPCS_Sublayer [ CPCS_1, CPCS_2 ]
```

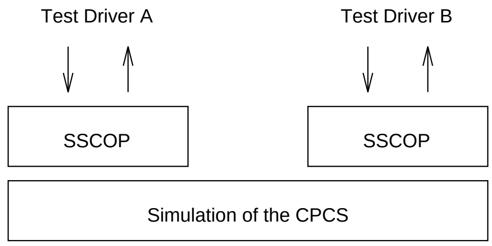


Figure 4.3: Testing with two peer entities

```
      |[CPCS_2]|
   SSCOP_sublayer [ SSCF_2, CPCS_2, I_SSCOP_2 ]
      |[SSCF_2]|
   Test_Entity_2 [SSCF_2]
)
where

Process Test_Entity_1 [SSCF] : noexit :=
hide Test_Finished_1 in
(
   (* Establish a connection *)
   SSCF ! AA_Establish_req ! uu00 ! true;
   SSCF ! AA_Establish_conf ? uu:SSCOP_UU;

   (* Do nothing for a  while *)

   (* Disconnect *)

   SSCF ! AA_Release_req ! uu02;
   SSCF ! AA_Release_conf;
   Test_Finished_1;
   stop

)
EndProc

Process Test_Entity_2 [SSCF] : noexit :=
hide Test_Finished_2 in
(
   SSCF ! AA_Establish_ind ? uu:SSCOP_UU;
```

```
   SSCF ! AA_Establish_resp ! uu01 ! true;

   (* Do nothing for a while *)

   (* Accept the deconnection *)

   SSCF ! AA_Release_ind ? uu:SSCOP_UU ? s:SSCOP_Source;
   Test_Finished_2;
   stop
)
EndProc

...

EndSpec
```

To know if such a test is successful or not, we have to check that for every possible path generated by the specification, the two actions named `Test_Finished_1` and `Test_Finished_2` have been executed before the occurance of the deadlock. LMC (LOTOS Model Checker, see C) is very well suited for such a task.

The first step of the verification is to build the behaviour tree of the specification synchronized with the test drivers. Then we write the property that must be verified, and we use LMC to check that the property holds.

In our case the property would be:

`s̃ome((ẽf('Test_Finished_1') or ẽf('Test_Finished_2')) until deadlock).`

or its negation:

`all((af('Test_Finished_1') and af('Test_Finshed_2')) before deadlock).`

Both properties hold in this case (see figure 4.4), so we can say that the protocols offers the capabilities 7 and 8.

Verifying with one entity and two drivers

Some other properties like number 10 (Does IUT support the segmenting of STAT PDUs ?) need some special settings of the lotos specification, and a proper test sequence. The best way to do it is to use a stub for the CPCS sublayer that will be used directly instead of a peer SSCOP entity.

The rest of the procedure is similar to the previous case. Again we use LMC to check if the test was succesful or not. All the results are detailed in

```
1 - Compute the lmc_tree.
2 - Save the current lmc_tree.
3 - Load a lmc_tree in memory.
4 - Check a formula on the current tree.
5 - View the trace of the checking of the last formula.
6 - Set maximum width and depth for the next tree.
7 - Leave.
Enter your choice ==> 4

Formula you want to check (ending with a `.'):
    all((af('Test_1') & af('Test_2')) before deadlock).
The formula holds.
```

Figure 4.4: Result produced by LMC to check if the properties 7 and 8 are provided by The SSCOP

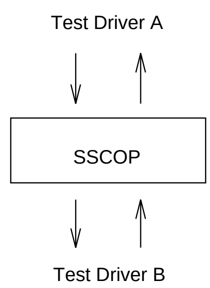


Figure 4.5: Testing with drivers for both the higher and lower layers

appendix A.

Verifying using ISLA

Finally, some properties may be difficult to verify. For example the property number 4 (Does the IUT recognize all of the messages regardless of state ?) requires that 15 checks (one for each type of PDU) be run for each of the 10 states of the protocol. To perform such a task, one needs to have a sequence to go in each of the states of the protocol, and then using a tool like ISLA see if none of the PDUs will provoke a deadlock. If such a deadlock occurs, then the property is not verified.

Because of the time required to perform such a test completely, I have executed it on the first state of the protocol (Idle) only. The resulting trace (see appendix A) shows that for any type of PDU received in that state, the protocol has some actions to perform. Thus it does not deadlock on any received PDU. This verifies the property for the first state .

Systematic methods to test protocols specified as finite state machines exist, but we did not have the time to use them.

Chapter 5

The Q2931 Protocol

The current version of the *ATM Forum UNI version 3.0* [6] signaling specification and the ITU-T Q.2931 [30] standard are closely aligned. First the functions of Q.2931 are summarized, followed by a description of how the *ATM Forum UNI 3.0* both subsets *Q.2931*, and defines additional functions.

The *Recommendation Q.2931* specifies the procedures for dynamically establishing, maintening and clearing of network connection at the B-ISDN user-network interface.The procedures are defined in terms of messages exchanged. The procedures are applied at the interface between a B-ISDN terminal equipment and a B-ISDN public network as well as at the interface betwen a B-ISDN customer network and a B-ISDN public network.

In this chapter I will first go through a description of the protocol and its particularities, then detail how I have represented it in LOTOS and finally present the results of my work on it.

5.1 Description of Q2931

The version of the protocol described by the ITU-T only provides point-to-point connections control (establishment, release, etc.). The set of messages defined for this protocol is actually a subset of the signalling protocol for ISDN. A lot of effort has been put in order that this protocol be very tolerant with unkonwn messages that could be received by an entity. This is done in order to provide some upward compatibility with the further releases, and also with Narrowband ISDN.

The signalling protocol is in charge of negotiating the bandwidth allocated to a connection. In the *Q.2931* this is done by indicating the peak level of information flow of the future connection to the network. This information is passed to the network through the initial request for connection. If any of the intermediate nodes can not guarantee the request bandwidth, the connection may be rejected. Once a connection is established, the bandwidth is allocated statically.

The major functions defined in ITU-T *Recommendation Q.2931* are:

- Point-to-point connection setup and release
- VPI/VCI selection and assignement
- Quality of Service (QoS) class request
- Identification of calling party
- Basic error handling
- Communication of specific information in setup request
- Subaddress support
- Specification of Peak Cell Rate (PCR) traffic parameters
- Transit network selection

The *ATM Forum UNI version 3.0* specification defines in addition to *Q.2931* the support for a call originator setup of point-to-multipoint call.

Technical Details

Here are the messages defined for the *Q.2931* signalling protocol.

SETUP This message is sent by the calling user to the network and by the network to the called user to initiate B-ISDN call and connection establishement.

CALL PROCEEDING This message is sent by the called user to the network or by the network to the calling user to indicate that the call establishement request has been received and no more call establishement information will be accepted until the present one is concluded.

ALERTING This message is sent by the called user to the network and by the network to the calling user to indicate that the called user alerting has been initiated.

CONNECT This message is sent by the called user to the network and by the network to the calling user to indicate call acceptance by the called user.

CONNECT ACKNOWLEDGE This message is sent by the network to the called user to indicate that the user has been awarded the call. It is also sent by the calling user to the network to allow symmetrical call control procedures.

RELEASE The message is sent by the user to request the release of an end-to-end connection to the network. It may also be sent by the network to a user to indicate the a connection has been cleared.

RELEASE COMPLETE This message is sent to acknowledge the release of a connection along with the associated resources.

STATUS ENQUIRY This message may be sent by any party involved in the connection to request the peer entity for a STATUS message.

STATUS This message is sent in response to a STATUS ENQUIRY message or to perform some error reporting to the peer entity.

NOTIFY This message is used to indicate information pertaining to a call / connection.

RESTART / RESTART ACKNOWLEDGE These messages are used to restart indicated virtual channels and paths associated to a connection.

The Version of the ATM Forun

The version described by the ATM Forum (ATMF) in the *ATM UNI Specification v3.0* [6] is slightly different from the ITU-T one. The reason is that some parties involved in the ATMF wanted to achieve something more complete than just the release 1 proposed by the ITU-T. The main differences rely in the capability of establishing and controlling point-to-multipoint connections. This signalling protocol is based on the same work as the ITU-T one. Thus they should be fairly compatible. Moreover, the fact that a lot of work as been done in order to obtain protocols that are resistant to unknown messages should help the interworking of the two descriptions.

The additional messages for the point-to-multipoint connection control are:

ADD PARTY : This is used to create a new connection with a node that is already connected to the source node (using the standard SETUP message).

ADD PARTY ACKNOWLEDGE : To accept the establishement of a new connection.

ADD PARTY REJECT : To refuse the new connection.

DROP PARTY : To release one of the connections on a node that has more than one established connection with the source. The last connection is released with the usual RELEASE message.

DROP PARTY ACKNOWLEDGE

The parameters transmitted with these messages are similar to the ones used with the SETUP / CONNECT ACKNOWLEDGE / RELEASE ... messages already described.

Each signaling message has a number of Information Elements, some of which are mandatory and others of which are optional. The key mandatory information elements used in the protocol are:

- ATM user cell rate requested
- Called party number
- Connection identifier (assigned VPI/VCI value)
- QOS Class requested

The messages related to a particular call attempt each contain a common mandatory information element, the *call reference*, that is unique on a signaling interface. All messages must also contain an information elements for their type , length, and protocol discriminator. There is an even larger number of optional parameters, such as:

- Broadband bearer capability requested
- Broadband lower and higher-layer information
- AAL parameters
- Called party subaddress
- Calling party number and subaddress
- Transit network selection
- Cause Code
- Endpoint reference identifier and endpoint state number

5.1.1 Interaction with the S-AAL

Many different primitives are defined between the Q2931 and the SAAL to access the services of the SAAL. Here is a quick description of them:

AAL-ESTABLISH These primitives (request, indication and confirm) are used to establish a point-to-point connection for assured data transfer between the peer user entities.

AAL-RELEASE (request, indication and confirm) primitives are used to terminate a point-to-point connection.

AAL-DATA (request and indication) primitives are used for the assured point-to-point transfer of SDUs between the peer user entities.

AAL-UNITDATA (request and indication) primitives are used for the non-assured, broadcast and point-to-point transfer of user information.

The *Q2931* uses the data link services of the SAAL. Before the procedures of the *Q2931* are invoked, an assured mode Signalling AAL (SAAL) must be established between the user and the network.

Establishment of Signalling AAL connections (SAAL) is initiated by transferring an AAL-ESTABLISH.Request primitive to the SAAL.

On receipt of an AAL-ESTABLISH.confirm or AAL-ESTABLISH.indication primitive from the SAAL, access signalling may begin. The AAL-ESTABLISH.indication primitive will be received in the case of the SAAL establishment request by the peer entity, and AAL-ESTABLISH.confirm in response to a local request to establish an SAAL connection.

All the layer 3 messages shall be sent to the signalling AAL using a AAL-DATA.Request primitive.

5.1.2 Q2931 Internals

After having described the interaction of the *Q2931* with its environment, I will describe some of its internal characteristics: its states and its timers.

The States

The *Q2931* has different states depending on the interactions that have taken place between the*Q2931* and its environment. These states are:

Null(1) In this state, no call exits.

Call Initiated (1) This state exists for an outgoing call when the user requests call establishment from the network.

Outgoing Call Proceeding (3) This state exists for an outgoing call when the user has received acknowledgement that the nework has received all call information to affect call establishment.

Call Delivered (4) This state exists for an outgoing call when the user has received an indication that the remote user alerting has been initiated.

Call Present (6) This state exists for an incoming call when the user has received a call establishment request but has not yet responded.

Call Received (7) This state exists for an incoming call when the user has indicated alerting but has not yet answered.

Connect Request(8) This state exists for an incoming call when the user has answered the call and is waiting to be awarded the call.

Incoming Call Proceeding (9) This state exists for an incoming call when the user has sent acknowledgement that the user has received all call information necessary to affect call establishment.

Active(10) This state exists for an incoming call when the user has received an acknowledgement from the network that the user has been awarded the call. This state exists for an outgoing call when the user has received an indication that the remote user has answered the call.

Release Request(11) This state exists when the user has requested the network to clear the end-to-end connection (if any) and is waiting for a response.

Release Indication(12) This state exists when the user has received an invitation to disconnect because the network has disconnected the end-to-end connection (if any).

5.2 Specification of the Q2931 in LOTOS

The *ATM Forum UNI version 3.0* specification [6] uses a narative text to specify the Signaling Protocol. A significant portion of the specification is involved with handling error cases, invalid messages, inconsistent parameters, and a number of other unlikely situations. These are all important functions since the signaling protocol must be highly reliable in order for users to accept it.

This is what I used to produce the LOTOS specification.

The first goal of this specification was to be as complete as possible with respect to the recommendation. This means that as few details as possible should be kept out of the specification. The second objective was to make it easy to reuse and to integrate with other specification. This is important in order to build more complete applications later.

5.2.1 Organization of the Specification

The specification is written in state oriented style. I chose the explicit state oriented style because it would be easier to read (see also Section 4.2.3) .

The specification is a description of the *Q2931* finite state machine. This specification supports the call reference errors, the Status and Status Enquiry procedures, the Restart procedure, the SAAL reset and failures, and the timers.

The action structure employed in our specification is as follows:

g !sp !p

where sp is a sort of service primitive and p a sort of Layer3_PDU.

The Gates

Service acces points (SAPs) are abstract means of interaction between service users and a service provider. A simple way of representing SAPs in LOTOS is to employ one gate to represent a whole service boundary. The *Q2931* uses the SAAL services :

g is the gate used for the interactions between the Q2931 and the SAAL

All the messages exchanged between the *Q2931* and the SAAL layers are passed through the gate "g". The primitives used are:

- AAL-Establish (request / indication / response / confirm)
- AAL-Release (request / indication / confirm)
- AAL-Data (request / indication)

The internal gates of the *Q2931* are:

L3_I for the selection of the call reference.

L3_Timer for the synchronisation with the timeouts

5.2.2 The resulting LOTOS specification outline

```
specification Q2931_specification[g] : noexit

(* Description of data type*)
...

 behaviour

Q2931[g]

where

  Process Q2931[g] :noexit:=
     hide L3_I,L3_Timer in
      (
        Layer3[g,L3_I,L3_Timer](NULL, TimerSet(...),Indefini)
       )
EndProc (*Q2931*)


 Process Layer3[g,L3_I,L3_Timer](S:Layer3_States, ts:SetofTimer, call_ref:Nat): noexit:=

    (
       Layer3_Action[g,L3_I,L3_Timer](S,ts,call_ref)
 >> accept S:Layer3_States, ts:SetofTimer, call_ref:Nat
         in  Layer3[g,L3_I,L3_Timer](S, ts, call_ref)

     )
  EndProc(*Layer3*)


   Process Layer3_Action[g,L3_I,L3_Timer](S:Layer3_States,
ts :SetofTimer,call_ref :Nat): exit(Layer3_States,SetofTimer,Nat):=
   (
     ....
   )
   EndProc (*Layer3_Action*)
```

```
Endspec (*Q2931_specification*)
```

5.3 Utilization of the Specification

One of the purpose of using LOTOS is to be able to demonstrate the correctness of the specification. LOTOS is based on mathematical background that allows to formally analyse the system. It provides means to write precise and unambiguous specifications as well as to reason about them formally using mathematical rules. Validation is a common term to design the activity to check a specification in order to increase the confidence in its correctness. Three techniques are used to achieve the validation phase: Verification, simulation, and testing.

Simulation of the protocol: Executing the specification may help to discover, at an early stage, design errors.If the specification is not complex, simulation gives a good coverage of errors. But, when the model becomes complex, it is impossible to cover all the cases due to time and space explosion. Simulation of LOTOS specifications is supported by several tools: ISLA, SELA .See Appendix C.

Verification : Verification means the activity of proving formally that a system satisfies a given property, using mathematical rules for the proof. The tools that supporte verification are : LMC (LOTOS Model Checker) and GOE (Goal Oriented Execution). The following properties can be verified:

- There is no deadlock in the specification
- After a certain event a, an event b must happen.

5.3.1 Simulation using ISLA

ISLA offers a step-by-step execution mode which gives the ability to monitor and trace specified execution sequences. Pratically, the generation/verification of test-sequences using step-by-step analysis is easy. I have used it to verify many trivial properties.

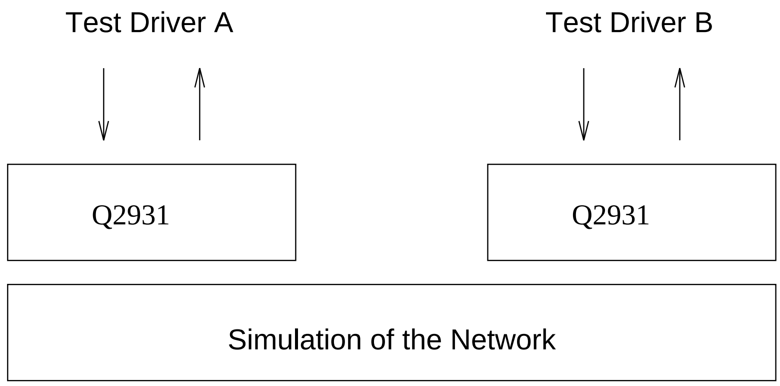


Figure 5.1: Validation of some test-sequences

5.3.2 Verification of Properties

Verification of properties is useful to check that the protocol meets its requirements. Such a verification is usefull first when a design of the protocol is built with enough details, then when an implementation of the protocol is realized.

The verification of properties at design stage can help us establish some test-sequences that will be used with the implementation to check that it corresponds to the design.

One of the possible ways of to exercising some properties is to have two peer *Q2931* entities linked by a simulated network. Each *Q2931* entity has a driver running on top of it that will simulate a user (see figure 5.1). Finally the test has succeeded if the end of the sequence coded in the drivers (the *Q2931* users) is reached. The idea is to synchronize the specification of the *Q2931* with different processes representing test sequences. Some test-sequences are in Appendix B

To know if such a test is successful or not, we have to check that all actions have been executed before reaching the deadlock.

Chapter 6

Conclusions

After this project has been completed, I came to several conclusions on different points. First considering the ATM signalling protocol, but also on LOTOS and its possible use in a software development methodology.

6.1 LOTOS

This project gave me a good opportunity to test LOTOS for large specifications. I came to several conclusion about its usefulness, strengths and weaknesses. These observation can be categorized in three categories:

- The language as itself and what I missed in it.
- LOTOS as a specification technique, and what its uses should be.
- And finally, a review of what I found useful with the tools provided by the University of Ottawa Toolkit.

Each of these point will be developed further.

6.1.1 The Language

The only way to implement Loops in LOTOS is to use recursivity, which may be a problem for users not familiar with it.

6.1.2 LOTOS as a Specification Technique

To evaluate LOTOS as a specification technique, I have to compare it with other languages that exists. In my case, I will compare LOTOS with SDL. To derive the specification of the SSCOP, I had to work with the SDL diagrams published in the specification Q.2120 [23] from ITU-T and I came to several conclusions.

A Specification as a Communication Tool

The first thing is that obviously, LOTOS is a language offering many of possibilities, and it can be used to specify anything that one can expressed with another specification language (in this case SDL). Its strict syntax forces one to resolve all the ambiguities that may be present. But on the other hand, the resulting specification is not easy to read and understand at a first glance. Especially when the specification gets bigger and one starts to use complicated data types and powerful operators such as parallel composition or interleaving, one can get very confused with it.

If we compare that to SDL, we can see that the problem is the reverse : as it is mainly a graphical language, an SDL representation can be understood very quickly. But when it comes to express details , SDL reaches its limits, and the graphics are spread across many pages (typically the SSCOP specification is composed of 51 pages of SDL diagrams).

My first point is that LOTOS's strong formality is a good thing for unambiguous representation, but makes it harder to understand what is specified. A language with a less rigid formalism ,such as SDL, proves to be easier to use.

A Specification as a Verification Tool

A language like LOTOS that offers very strong formalisms to express a specification offers the advantage to make possible the use of tools to transform it and manipulate the specification itself. This makes verification of properties and early debugging of products feasible. This has the advantage to the cost of detecting and correcting mistakes, because it can be performed in the early stages of software development.

With a less formal approach such as in SDL, one does not get that power of verification. Thus an SDL specification may still contain ambiguities or mistakes that will be discovered later at greater costs.

So my second point is that the strong formalism of LOTOS makes it useful in the early stage of specification and design to validate some choices that may be hard to verify in other ways.

How to Use LOTOS ?

We can divise a software development methodology that uses the advantages provided by both approaches. The first step should be to build a informal description of the product to built. Then using a semi-formal technique, such as finite automata for example, derive a specification with a low level of formalism. Often this transformation will help finding the major problems in the original description. Then when one has obtained a fairly stable semi-formal description of the product, a more formal description technique should be used to resolve the remaining ambiguities and to verify in a formal way that the requirements are met.

Finally, when in comes to the implementation both the semi-formal and the formal representation should be used, allowing a rapid understanding of the design and an unambiguous reference.

Table 6.1 summarize the software development method based on the combination of LOTOS and other semi-formal technique that I describe below. Figure 6.1 also offers additional information.

The requirements analysis phase

This phase should be conducted with informal methods. The major concern at this stage is to identify what are the important points and not the formality of their expressions.

The Specification Phase

The first part of the specification phase should be conducted with semi-formal methods such as finite automata. The purpose here is to understand how the system will behave, and what are the functions that have to be implemented and how they should be executed. Then the use of formal methods such as LOTOS allows us to add more precision in the specification.

At the same time, the main requirements should also be expressed in a formal way, in order to check them against the formal description at a latter stage.

Using both semi-formal and formal method has the advantage to obtain an easily understandable specification which should be free of ambiguities.

The Design Phase

The design phase should be conducted with formal methods. At this stage of the software development, we want to be able to verify that the design meets the requirements before we go to the implementation phase. So mainly this should be a process of refinement of the formal specification and its testing against the requirements expressed in a formal way. From comparison, we should derive precise test cases that will be used to verify the correctness of the implementation.

6.1.3 The Tools

Using the LOTOS Toolkit of the University of Ottawa, I can also present some remarks about the tools.

Random Walk

Random Walk is one of the newest tools developed. The principle is to randomly chose the next action to be executed at each step. Eventually, we will get a trace that may show some problems. The main type of problem that can be found then is unexpected execution paths or unexpected deadlocks.

The Problem

Even if the theory seems promising, the reality of the tool is a lot less useful. In fact Random Walk helped me find some bugs in the simplified reprensentation of the CPCS sublayer, but nothing in the SSCOP protocol itself. After I fixed the CPCS sublayer, the trace that were generated by RW did not contain any deadlock, but they did not do any useful work either.

The problem is that with protocol (like the SSCOP or telephony systems) where there is a disconnection action (hangup for the phones) that can be executed at any time, RW doesn't go deep in the execution traces. By

Software Development Steps	Requirements Analysis	Specification	Design
Description Techniques	Natural Languages	Semi-Formal Language (SDL)	Formal Language (LOTOS)
Uses & Benefits	Rapid capturing of the requirements	- Easy understanding - Reveals the major problems - Definition of general test-cases	- Correct ambiguities - Validation and Verification - Design of specific Test-Sequences
Drawbacks	Can be very ambiguous and imprecise	- Often not precise enough - Can become complicated if the system is too big	- Difficult to understand - Sensitive to syntax and semantics errors

Table 6.1: Tools for the different software development phases

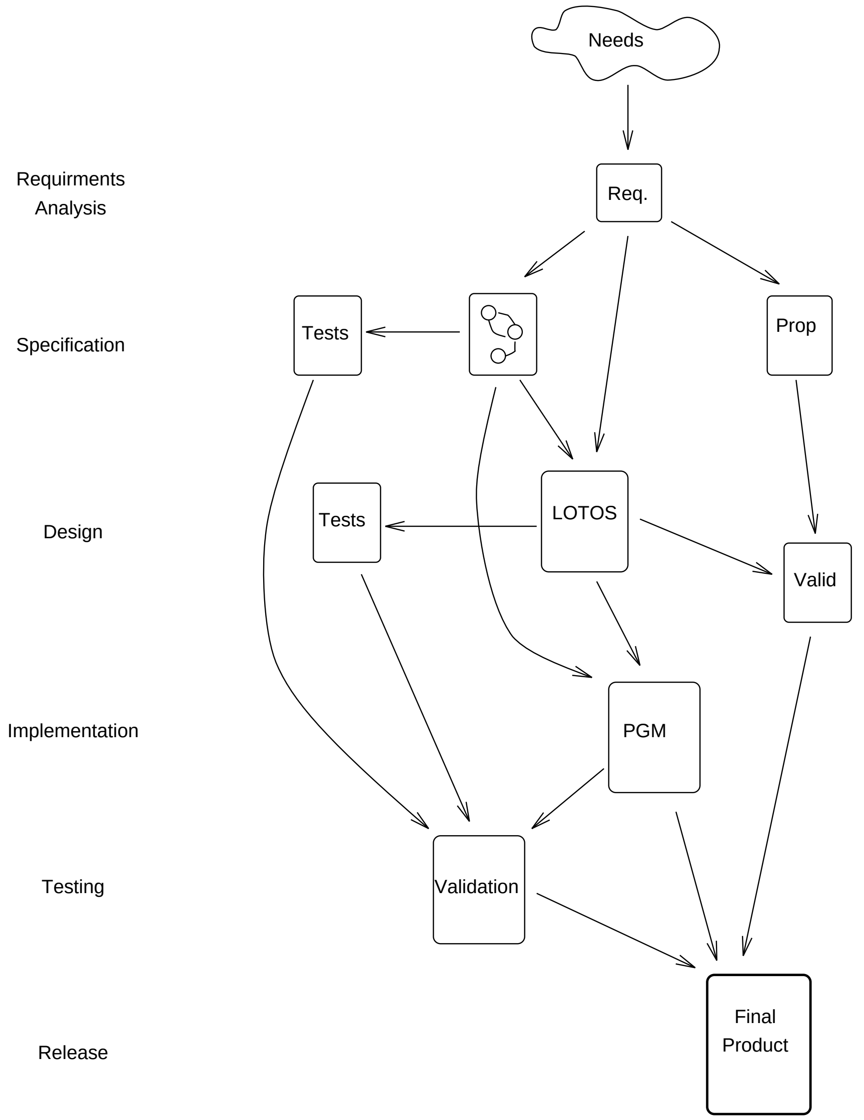


Figure 6.1: A simplified view of a software development method

deep I mean that even with a high number of actions (400 in my case) no connection has been established between the two entities. So RW exercises only a very small part of the specification. Also, this part is always the same: the begining of the connection establishment procedure followed by a disconnection.

A Possible Solution

In fact I feel that Random Walk could be an interesting approach if we have a way to control it. I think that combining RW with Goal Oriented Execution could produce some good results. The idea would be to define a set of states for the specification. These states should be meaningful, but not necessarily too tightly linked to the specification structure. Then for each of these states we could define a set of possible next actions.

At this point Random Walk could be used to pick one of the proposed actions, and then Goal Oriented would try to reach one of the predefined states from that randomly picked action.

With this method, by excluding actions like "disconnect" from most of the places where we know it will not produce anything interesting, we could obtain some meaningful and "deep" traces.

This solution may not be completely satisfaction in the sense that it alters dramatically the fairness that should be a characteristic of Random Walk.

6.2 Future Work

After this project, there are several possibilities for further work:

- First, it could be interesting to specify the remaining parts of the S-AAL, and integrate everything together.

- Then, another interesting point would be to test the interworking of the Q.2931 signalling protocol and the equivalent specification from the ATM-Forum. Or the Q 2931 and the B-ISUP (the signalling protocol at the NNI).

- Finally, some work could be done on a software development method which would bring the full advantages of LOTOS with easy to use (more descriptive) specification tools. This could be very helpful especially in the early stages of specification.

Appendix A

Results of the Tests Performed on the SSCOP

This appendix displays all the results obtained during the testing of the SSCOP protocol as well as the test constructs that were used.

A.1 Tests Using LMC

For this serie of tests, the technique was to synchronize the specification of the SSCOP with different processes representing test sequences. The test sequences are terminated by a specific action and then a explicit deadlock.

LMC (LOTOS Model Checker) is used to determine if the test as been succesful or not: the terminating test actions must be present in any trace before the deadlock occurs.

A.1.1 Properties 7 and 8: Does the IUT Support Establishement and Release Procedures ?

For this test I used to peer entity of the SSCOP protocol along with a simulated CPCS sublayer. The tests are specified by two processes simulating the users of the two SSCOP entities. The resulting LOTOS specification is as follow:

```
behaviour
```

```
hide I_SSCOP_1, I_SSCOP_2 in
(
   Test_Entity_1 [ SSCF_1 ]
      |[SSCF_1]|
   SSCOP_sublayer [ SSCF_1, CPCS_1, I_SSCOP_1 ]
      |[CPCS_1]|
   CPCS_Sublayer [ CPCS_1, CPCS_2 ]
      |[CPCS_2]|
   SSCOP_sublayer [ SSCF_2, CPCS_2, I_SSCOP_2 ]
      |[SSCF_2]|
   Test_Entity_2 [SSCF_2]
)
where

Process Test_Entity_1 [SSCF] : noexit :=
hide Test_Finished_1 in
(
   (* Establish a connection *)
   SSCF ! AA_Establish_req ! uu00 ! true;
   SSCF ! AA_Establish_conf ? uu:SSCOP_UU;

   (* Do nothing for a  while *)

   (* Disconnect *)

   SSCF ! AA_Release_req ! uu02;
   SSCF ! AA_Release_conf;
   Test_Finished_1;
   stop

)
EndProc

Process Test_Entity_2 [SSCF] : noexit :=
hide Test_Finished_2 in
(
   SSCF ! AA_Establish_ind ? uu:SSCOP_UU;
   SSCF ! AA_Establish_resp ! uu01 ! true;

   (* Do nothing for a while *)

   (* Accept the deconnection *)
```

```
   SSCF ! AA_Release_ind ? uu:SSCOP_UU ? s:SSCOP_Source;
   Test_Finished_2;
   stop
)
EndProc

...

EndSpec
```

The property to be verified with LMC is:

s̃ome((ẽf('Test_Finished_1') or ẽf('Test_Finished_2')) until deadlock).

And the result of the execution is:

```
1 - Compute the lmc_tree.
2 - Save the current lmc_tree.
3 - Load a lmc_tree in memory.
4 - Check a formula on the current tree.
5 - View the trace of the checking of the last formula.
6 - Set maximum width and depth for the next tree.
7 - Leave.
Enter your choice ==> 4

Formula you want to check (ending with a `.'): ~some( (~ef('Test_1') or ~ef('Test_2')) until deadlock).
The formula holds.
```

A.1.2 Property # 10: Does IUT support the segmenting of STAT PDUs ?

In this case, I used only one entity of the SSCOP specification synchronized with a test driver representing the user and another one representing the CPCS sublayer.

The parameters controlling the protocol have to be set correctly so that the messages exchanged are meaningful. The setting I used was:

VR(MR) = 6. This means that the value of the anticipation window of the protocol is of 6 messages.

MaxSTAT = 3. This is the maximum number of elements in the list of PDU sequence numbers contained in a STAT PDU. This value implies that no more than one interval of missing SD PDUs can be specified in each STAT PDU. Thus we will have segmenting as soon as we have two gaps in the sequence of SD PDUs.

Timers . Their initial value is set so that they will not expire during the simulation of the protocol.

The values affected to the other variables do not influence the results of this test.

The test construct in LOTOS is as follow:

To verify that the test as been succesfull, we can use the same LMC formula as for the previous test.

A.1.3 Property 5.a: In absence of protocol error, does the IUT support assured data transfer with sequence integrity ?

The same test as the one used for property # 10 can be used to check this property. The difference is in the formula that we will use with LMC.

To check if all the Data Units are delivered in sequence, we can use the following formulas:

`all('SSCF ! AA_UnitData_ind ! uu00 ! 0' before 'SSCF ! AA_UnitData_ind ! uu01 ! succ(0)')` `all('SSCF ! AA_UnitData_ind ! uu01 ! 1' before 'SSCF ! AA_UnitData_ind ! uu02 ! succ(succ(0))')` `all('SSCF ! AA_UnitData_ind ! uu02 ! 2' before 'SSCF ! AA_UnitData_ind ! uu01 ! succ(0)')` `all('SSCF ! AA_UnitData_ind ! uu03 ! 3' before 'SSCF ! AA_UnitData_ind ! uu01 ! succ(0)')`

Appendix B

Test-sequences of Q2931

This appendix displays some test-sequences corresponding to the user actions to verify their conformance to my specification. These test-sequences are message flow should be generated using the state machine. This is used to verify that the specification meets functionnal requirements.

```
(*****************************)
(* body of the specification*)
(****************************)


behaviour
      hide g1,g2,L3_I1,L3_I2,L3_Timer in
     (
   ( UserA3[g1,L3_I1](succ(0))
|[g1,L3_I1]|
     Layer3[g1,L3_I1,L3_Timer](NULL,TimerSet(...),Indefini)
   )
|[g1]|
Network[g1,g2]
|[g2]|
   ( UserB3[g2](succ(0))
|[g2]|
    Layer3[g2,L3_I2,L3_Timer](NULL, TimerSet(...),Indefini)
            )

      )
```

```
 where


(*Simulation of the network*)

 Process Network[g1,g2] :noexit:=
(
  one_direction[g1,g2]
 |||
  one_direction[g2,g1]
)
  EndProc (*Network*)

     Process one_direction[g1,g2] :noexit:=
(
   g1 !AAL_Data_Req ?p:Layer3_PDU;
   g2 !AAL_Data_Ind !p;
   one_direction[g1,g2]
)
     EndProc (*one_direction*)

....


}





(****************)
(*Test-sequences*)
(****************)


(******************************************************************************)
(*Connection setup UserA to UserB, with Call_Proceeding then clearing by UserA*)
(******************************************************************************)

 Process userA[h,L3_I](call_ref:Nat) :noexit:=
   L3_I !call_ref;
   h !AAL_data_Req !Setup(call_ref);
   h !AAL_Data_Ind !call_proceeding(call_ref);
   h !AAL_Data_Ind !Connect(call_ref);
```

```
   h !AAL_Data_Req !ConnectAck(call_ref);
   h !AAL_Data_Req !Release(call_ref);
   h !AAL_Data_Ind !Release_complete(call_ref);
   Stop
EndProc (*UserA*)

 Process userB[h](call_ref:Nat) :noexit:=
     h !AAL_data_Ind !Setup(call_ref);
    h !AAL_Data_Req !call_proceeding(call_ref);
     h !AAL_Data_Req !Connect(call_ref);
     h !AAL_Data_Ind !ConnectAck(call_ref);
    h !AAL_Data_Ind !Release(call_ref);
    h !AAL_Data_Req !Release_complete(call_ref);
         Stop
EndProc (*UserB*)


(****************************************************************)
(*Connection setup UserA to UserB, with call rejection by UserB*)
(****************************************************************)


 Process userA[h,L3_I](call_ref:Nat) :noexit:=
   L3_I !call_ref;
   h !AAL_data_Req !Setup(call_ref);
   h !AAL_Data_Ind !Release_complete(call_ref,cause);
   Stop
EndProc (*UserA*)


 Process userB[h](call_ref:Nat) :noexit:=
   h !AAL_data_Ind !Setup(call_ref);
   h !AAL_Data_Req !Release_complete(call_ref,cause_reject);
   Stop
EndProc (*UserB*)


(***************************************************************************)
(* Connection setup A to B with immmediate clearing by A. Thus, B's Connect
 reaches A in Release Rquest state where it is answered by a STATUS*)
(***************************************************************************)

Process userA[h,L3_I](call_ref:Nat) :noexit:=
   L3_I !call_ref;
```

```
   h !AAL_data_Req !Setup(call_ref,param);
   h !AAL_Data_Req !Release(call_ref);
   h !AAL_Data_Ind !Connect(call_ref);
   h !AAL_Data_Req !Status(call_ref,Release_Request_11,cause_97);
   h !AAL_Data_Ind !Release_complete(call_ref);
   Stop
EndProc (*UserA*)

 Process userB[h](call_ref:Nat) :noexit:=
     h !AAL_data_Ind !Setup(call_ref);
       h !AAL_Data_Req !Connect(call_ref);
     h !AAL_Data_Ind !Release(call_ref);
     h !AAL_Data_Req !Release_complete(call_ref);
 h !AAL_Data_Ind !Status(call_ref,Release_Request_11,cause_97);
         Stop
EndProc (*UserB*)


(******************************************************************)
(*Setup on a slightly lossy link, where the first RELEASE is lost.*)
(******************************************************************)


Process userA[h,L3_I](call_ref:Nat) :noexit:=
   L3_I !call_ref;
   h !AAL_data_Req !Setup(call_ref);
   h !AAL_Data_Ind !Connect(call_ref);
   h !AAL_Data_Req !ConnectAck(call_ref);
   h !AAL_Data_Req !Release(call_ref);
   h !AAL_Data_Req !Release(call_ref);
   h !AAL_Data_Ind !Release_complete(call_ref);
   Stop
EndProc (*UserA*)

 Process userB[h](call_ref:Nat) :noexit:=
     h !AAL_data_Ind !Setup(call_ref);
       h !AAL_Data_Req !Connect(call_ref);
 h !AAL_Data_Ind !ConnectAck(call_ref);
     h !AAL_Data_Ind !Release(call_ref);
     h !AAL_Data_Req !Release_complete(call_ref);
         Stop
EndProc (*UserB*)
```

Appendix C

LOTOS

In this chapter, I recall the origin of LOTOS as a formal specification technique, and then present quickly its main operators, its uses, and the tools that were developed at the University of Ottawa.

C.1 Formal Methods

Formal methods, in particular process algebras, proved their usefulness in capturing descriptions of complex, concurrent, and communicating systems. Their main advantages are:

- They make it possible to formalize unambiguous specifications.
- They ease the derivation of design.
- They support formal analysis and validation methods.

LOTOS (Language Of Temporal Ordering Specifications) is an algebraic specification language and a Formal Description Technique (FDT). It was especially developed for the formal description of the OSI architecture (interfaces, services and protocols), although it is applicable to distributed and concurrent systems in general. Today, people try to extend its field of action to hardware, telephony, operating systems, embedded systems and real-time systems. LOTOS has been an ISO standard (8807) since 1989 [18].

The basic idea of LOTOS is to describe a system by defining the temporal relations along the interactions that constitutes the system's externally observable behaviour. The process part of LOTOS (known as *Basic LOTOS*) is based on ideas found in CCS [19] and CSP [11]. The data part of LOTOS (included in *Full LOTOS*) is based on the theory of abstract data types and comes from the language ACT ONE [5].

C.2 Operators

In LOTOS, systems are described in terms of *processes*. A process is viewed as a black box interacting with its *environment* via its observable *gates*. Its internal actions are unobservable by the environment. The *behaviour expression* is built by combining LOTOS actions by means of operators and possibly instantiations of other processes.

The basic element of a behaviour expression is the *action* which represents a synchronization between processes, between a process and its environment, or both. An action consists of a gate name, a list (possibly empty) of value experiment offers (value offers or interaction parameters), and possibly a predicate that imposes conditions on the event to be accepted. Actions are *atomic* in a sense that they occur instantaneously, without consuming time.

C.3 Uses of a LOTOS Specification

C.3.1 Interactive Simulation

Execution of LOTOS specifications is made possible by using suitable interpreters (simulators). The existing interpreters for LOTOS today are based on step-by-step execution. This means that the specification is executed action by action, and the set of the next possible actions is determined after the execution of an action. The user plays the role of the environment and resolves non-determinism, by deciding which action should be taken and by providing the required value expressions. This execution method is time consuming, especially if there are several alternatives for each action. This type of execution can still be useful to understand what happens in very particular cases.

<table>
<tr><th>Name</th><th>Behaviour Expression</th><th>Comment</th></tr>
<tr><td colspan="3">Basic Behaviour Expressions</td></tr>
<tr><td>Inaction</td><td><code>stop</code></td><td>Cannot engage in any interaction (deadlock).</td></tr>
<tr><td>Successful Termination</td><td><code>exit</code></td><td>Indicates that a process has successfully performed all its actions.</td></tr>
<tr><td>Process Instantiation</td><td><code>ProcName [g1, ..., gn]</code></td><td>Creates an instance of a process.</td></tr>
<tr><td>Internal Action</td><td><code>i</code></td><td>This is a special action that a process can execute independently.</td></tr>
<tr><td colspan="3">Basic Operators</td></tr>
<tr><td>Action Prefix</td><td><code>a ; B</code></td><td>Use to prefix a behaviour expression <code>B</code> with an action <code>a</code>.</td></tr>
<tr><td>Choice</td><td><code>B1 [] B2</code></td><td>Allows the user to define different alternatives for a given process.</td></tr>
<tr><td colspan="3">Enabling and Disabling</td></tr>
<tr><td>Enabling</td><td><code>B1 >> B2</code></td><td>Used to sequence two behaviour expressions. <code>B1</code> has to <code>exit</code> for <code>B2</code> to be executed.</td></tr>
<tr><td>Disabling</td><td><code>B1 [> B2</code></td><td>Used to express situations where <code>B1</code> can be interrupted by <code>B2</code> during normal functioning.</td></tr>
<tr><td colspan="3">Composition</td></tr>
<tr><td>Parallel Composition</td><td><code>B1 |[g1, ..., gn]| B2</code></td><td>Composition in which <code>B1</code> and <code>B2</code> behave independently, except for the gates <code>g1</code>, ..., <code>gn</code> where <code>B1</code> and <code>B2</code> must synchronize.</td></tr>
<tr><td>Interleaving</td><td><code>B1 ||| B2</code></td><td>Composition in which <code>B1</code> and <code>B2</code> behave independently (the synchronization set is empty).</td></tr>
<tr><td>Full Synchronization</td><td><code>B1 || B2</code></td><td>Composition in which <code>B1</code> and <code>B2</code> are synchronized on all their gates.</td></tr>
<tr><td colspan="3">Other Operators</td></tr>
<tr><td>Hiding</td><td><code>hide g1, ..., gn in B</code></td><td>Used to hide actions which are internal to a system. These actions cannot synchronize with the environment.</td></tr>
<tr><td>Guarded Behaviour</td><td><code>[P] -> B</code></td><td><code>B</code> can be executed if <code>P</code> is true.</td></tr>
<tr><td>Local Definition</td><td><code>let x:s = E in B</code></td><td>Substitutes a value expression () by a value identifier (<code>x</code>) of sort <code>s</code> in the behaviour expression <code>B</code>.</td></tr>
</table>

Table C.1: Main LOTOS constructors

C.3.2 Symbolic Execution

This mode of execution tries to derive a symbolic behaviour tree from the specification, without the intervention of the user. The values that should be provided by the environment are replaced by symbolic values. The system attempts to go as far as possible, but this means that reduction methods, such as loop detection, have to be applied in order to avoid the state explosion problem as much as possible [7].

C.4 The University of Ottawa LOTOS Toolkit

A set of tools have been developed at the University of Ottawa. They provide the user with the methods described earlier to validate and verify a given LOTOS specification. The tools available in this toolkit, called *XELUDO*, are described in this section.

C.4.1 SELA: Symbolic Expander for LOTOS Applications

SELA [2] is a tool that generates the behaviour tree of a specification symbolically, that is, without theuse of actual values. It takes a specification and generates a symbolic behaviour tree. Also, SELA provides the means to translate the generated tree into a monolithic style specification and other formats useful for in-depth analysis of the original specification. This is often used as the first step for other tools like LMC or GOE (see below).

C.4.2 ISLA: Interactive System for LOTOS Application

ISLA [8] is an interpreter that helps simulating the specifications and discovering design errors by giving the user the ability to monitor and trace some execution sequence. ISLA provides a wide range of services. First the syntax and the static semantics of a LOTOS specification are checked and an internal representation is generated. This internal form is used to execute the specification in a step-by-step mode, or to generate a symbolic execution tree. It is possible to go back to previous points in the execution and explore other alternatives. This tool provides an *interactive simulation* method to use a LOTOS specification.

C.4.3 LMC: LOTOS Model Checker

Lotos Model Checker provides the possibility to check the conformance of the specification to a certain property. This Tool is used to prove that a given property is verified or not.

C.4.4 GOE: Goal Oriented Execution

Goal oriented execution allows to find test-sequences with certain characteristics, for example paths that start with one particular action and end with another one.

C.4.5 RW: Random Walk

This tool generates randomly a tree of possible test-sequences from the specification.

Appendix D

Acronyms

AAL	ATM Adaptation Layer
ATM	Asynchronous Transfer Mode
ATMF	ATM Forum
B-ISDN	Broadband ISDN
B-ISUP	SS7 Broadband ISDN User Part
CCITT	Commite Consultatif International des Telegraphes et Telephones
CLP	Cell Loss Priority (ATM cell header field)
CP	Common Part
CPCS	Common Part Convergence Sublayer (CS sublayer)
CS	Convergence Sublayer (AAL sublayer)
DSS2	Digital Subscriber Signalling System No. 2
GFC	Generic Flow Control (ATM cell header field)
GOE	Goal Oriented Execution (XELUDO Tool)
HEC	Header Error Control (ATM cell header field)
ICI	Inter Carrier Interface
ISDN	Integrated Service Digital Network
ISLA	Interactive System for LOTOS Applications (XELUDO Tool)
ISO	International Standard Organization
ISSI	Inter Switching System Interface
ITU	International Telecom Union (formally CCITT)
LM	Layer Management
LMC	LOTOS Model Checker (XELUDO Tool)
LOTOS	Language Of Temporal Ordering Specification
MTP3	Message Transfer Protocol 3

N-ISDN	Narrow-Band ISDN
NNI	Network Node Interface
OAM	Operation And Maintenance
OSI	Open System Interconnection
PDU	Protocol Data Unit
PHY	ATM Physical Layer
PICS	Protocol Implementation Conformance Statement
PM	Physical Medium (PHY sublayer)
PRM	Protocol Reference Model
PTI	Payload Type Indicator (ATM cell header field)
QoS	Quality of Service
RW	Random Walk (XELUDO Tool)
SAAL	Signalling AAL
SAP	Service Access Point
SAR	Segmentation And Reassembly sublayer (AAL sublayer)
SDL	Simple Description Language
SDU	Service Data Unit
SELA	Symbolic Expander of LOTOS Applications (XELUDO Tool)
SSCF	Service Specific Coordination Function
SSCOP	Service Specific Connection Oriented Protocol
SSCS	Service Specific Convergence Sublayer (CS sublayer)
SSP	Service Specific Part
SS7	Signalling System No. 7
TC	Transmission Convergence (PHY sublayer)
UNI	User Network Interface
VCI	Virtual Circuit Identifier (ATM cell header field)
VPI	Virtual Path Identifier (ATM cell header field)

Appendix E

The SSCOP Specification In LOTOS

Because the SSCOP specification is long, I did not include it in this document. However you can find it in the following file:
`~u684327/pfe/spec/sscop.rwk.l`

Appendix F

The ATM Forum UNI version 3.0 Specification In LOTOS

Bibliography

[1] Anthony Alles. ATM in Private Networking. Tutorial, Hughes LAN Systems, 1993.

[2] P. Ashkar. Symbolic Execution of LOTOS Specifications. Master's thesis, University of Ottawa, 1992.

[3] Marc Boisseau, Michel Demange, and Jean-Marie Munier. *Réseaux ATM*. Eyrolles, 1994. TK 5103.7 .B65 1994.

[4] Recommendation E.164. 'Numbering Plan for the ISDN Era'. Technical report, ITU-T, 1991.

[5] B. Ehrig and B. Mahr. *Fundamentals of Algebraic Specifications*. Springer Verlag, 1985.

[6] The ATM Forum, editor. *ATM User-Network Interface specification v3.0*. Prentice Hall, 1993.

[7] R. Guillemot and Luigi Logrippo. Derivation of Useful Execution traces from LOTOS Specifications by using an Interpreter. In K.J. Turner, editor, *Formal Description Techniques*, pages 311 – 325. North Holland, 1989.

[8] M. Haj-Hussein. ISLA: An Interactive System for LOTOS Applications. Master's thesis, University of Ottawa, 1989.

[9] Rainer Hände, Manfred N. Huber, and Stefan Schröder. *ATM Networks Concepts, Protocols, Applications*. Addison-Wesley, second edition, 1994. TK 5103.7 .H255 1994.

[10] McDysan David E, Spohn Darren L. *ATM Theory And Application.* McGraw-Hill Series on Computer Communications, 1995. TK 5105.5 .M45 1995

[11] C. A. R. Hoare. *Communicating Sequential Processes.* Prentice Hall, 1985.

[12] Recommendation I.320. 'ISDN Protocol Reference Model'. Technical report, ITU-T, November 1993.

[13] Recommendation I.321. 'B-ISDN Protocol Reference Model And Its Application'. Technical report, ITU-T, 1991.

[14] Recommendation I.327. 'B-ISDN Functional Architecture'. Technical report, ITU-T, March 1993.

[15] Recommendation I.361. 'B-ISDN ATM Layer Specification'. Technical report, ITU-T, March 1993.

[16] Recommendation I.362. 'B-ISDN ATM Adaptation Layer (AAL) Functional Description'. Technical report, ITU-T, March 1993.

[17] Recommendation I.363. 'B-ISDN ATM Adaptation Layer (AAL) Specification'. Technical report, ITU-T, March 1993.

[18] ISO 8807. 'Information Processing Systems - Open Systems Interconnection, LOTOS: A Formal Description Technique based on the Temporal Ordering of Observational behaviour'. standard, ISO, May 1989.

[19] R. Milner. A Calculus of Communicating Systems. In *Lectures Note in Comupter Science.* Springer Verlag, 1980. No. 92.

[20] Martin De Prycker. *Asynchronous Transfer Mode, solution for the broadband ISDN.* Ellis Horwood, second edition, 1994. TK 5103.75 .P79 1993.

[21] Recommendation Q.2100. 'B-ISDN Signalling Adaptation Layer (SAAL) Overview Description'. Technical report, ITU-T, 1993.

[22] Recommendation Q.2110. 'B-ISDN Signalling Adaptation Layer (SAAL) – Service Specific Oriented Protocol (SSCOP)'. Technical report, ITU-T, 1993.

[23] Recommendation Q.2120. 'B-ISDN Meta-Signalling Protocol'. Technical report, ITU-T, 1993.

[24] Recommendation Q.2130. 'B-ISDN Signalling Adaptation Layer (SAAL) – Service Specific Coordination Function for support Signalling at the User-to-Network Interface (SSCF at the UNI)'. Technical report, ITU-T, 1993.

[25] Recommendation Q.2140. 'B-ISDN Signalling Adaptation Layer (SAAL) – Service Specific Coordination Function for support Signalling at the Network-Node Interface (SSCF at the NNI)'. Technical report, ITU-T, 1993.

[26] Recommendation Q.2761. 'Functional Description of the B-ISDN User Part of Signalling System No. 7'. Technical report, ITU-T, 1993.

[27] Recommendation Q.2762. 'General Functions of Messages and Signals'. Technical report, ITU-T, 1993.

[28] Recommendation Q.2763. 'Formats and Codes'. Technical report, ITU-T, 1993.

[29] Recommendation Q.2764. 'Signalling Procedures'. Technical report, ITU-T, 1993.

[30] Recommendation Q.2931. 'B-ISDN User-Network Interface Layer 3 Protocol'. Technical report, ITU-T, 1993.