1

National Library
of Canada

Bibliothèque nationale
du Canada

Canadian Theses Service

Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, tests publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30.

Distribution of Protocol Functions in an Interworking-based
ISDN Switching Node Implementation

by

James Doble, B.Sc.

A thesis submitted to the
~~Faculty of Graduate~~ Studies and Research
in partial fulfilment of the requirements
for the degree of
Master of Engineering

Ottawa-Carleton Institute for Electrical Engineering
Faculty of Engineering
Department of Systems and Computer Engineering
Carleton University

March 2, 1988

© copyright

1988, James Doble

Permission has been granted to the National Library of Canada to microfilm this thesis and to lend or sell copies of the film.

The author (copyright owner) has reserved other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without his/her written permission.

L'autorisation a été accordée à la Bibliothèque nationale du Canada de microfilmer cette thèse et de prêter ou de vendre des exemplaires du film.

L'auteur (titulaire du droit d'auteur) se réserve les autres droits de publication; ni la thèse ni de longs extraits de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation écrite.

ISBN 0-315-40649-6

The undersigned hereby recommends to the Faculty of Graduate Studies and Research acceptance of the thesis,

"Distribution of Protocol Functions in an Interworking-based ISDN Switching Node Implementation"

submitted by James Doble, B.Sc., in partial fulfillment of the requirements for the degree of Master of Engineering

Thesis Supervisor

Chairman,
Department of Systems and
Computer Engineering

Department of Systems and Computer Engineering

Faculty of Engineering

Carleton University

Ottawa, Canada

April 15, 1988

ABSTRACT

Although the fundamental premise behind the Integrated Services Digital Network (ISDN) is the provision of a variety of communication services using a single integrated digital network, it is clear that many initial ISDN switching node implementations will be based on the interworking of existing dedicated switching components. Within such implementations, protocol and service functions required to provide ISDN access can be distributed amongst the various components which comprise the implementation. This research document provides a systematic study of the various function distribution alternatives which may be used to provide an interworking-based ISDN switching node.

ACKNOWLEDGEMENTS

I would like to express my gratitude to God, who has been a constant source of help and strength, especially during recent years as I completed this thesis.

I would like to thank my thesis supervisor, Professor S. Mahmoud for his guidance and advice during the course of my thesis work. I would also like to thank Bell-Northern Research for supporting this research. In particular, I would like to thank Charles Wiebe, Dr. R. Jackson, and Mike Thomas for their assistance.

In addition, I would like to thank CCITT study group VII (ISDN/VII) for making $\tilde{I}.462$ (X.31) sufficiently interesting to be worth studying, and the following members of the original DMS/DPN interworking team, Mehmet Unsoy, Ed Juskevicius, Bal Shastri, Lapkin Ip, Kwan Wong, and Maurice Juneau, for clearly demonstrating that there were a great many ways in which interworking could be achieved.

TABLE OF CONTENTS

1.0	Introduction	1
1.1	Introduction to ISDN	1
1.2	Integration vs Interworking	3
1.3	Scope	6
1.4	Approach	11
1.4.1	Modeling of Protocol Functions	16
1.4.2	Modeling of Architectural Components	18
1.4.3	Modeling of Function Allocations	21
2.0	Introduction to the Access Requirements & Protocols	23
2.1	Layer 1 (Physical Layer)	25
2.2	Layer 2 (Data Link Layer)	29
2.3	Layer 3 (Call Control Layer)	33
3.0	Model Description	37
3.1	The Protocol Function Model	38
3.1.1	Layer 1 Protocol Functions	41
3.1.2	Layer 2 Protocol Functions	43
3.1.3	Layer 3 Protocol Functions	48
3.1.4	Protocol Function Model Summary	54
3.1.5	Service Functions	57
3.2	The Architectural Component Model	65
3.3	Allocation Constraints	70

3.4	Allocation Metrics	74
3.4.1	Minimum AT Complexity Metric (M1)	76
3.4.2	Maximum Service Provision Metric (M2)	77
4.0	Allocation Algorithms	79
4.1	The Allocation Algorithm	79
4.2	Metric Algorithms	86
5.0	Model Analysis	88
6.0	Extended Model Description and Analysis	98
6.1	Extended Model Description	98
6.2	Extended Model Analysis	109
6.2.1	Allocation IV	118
6.2.1.1	Service Provision Capability	118
6.2.1.2	AT Protocol Functions	119
6.2.1.3	AT Service Interfaces	120
6.2.2	Allocation III	122
6.2.2.1	Service Provision Capability	122
6.2.2.2	AT Protocol Functions	123
6.2.2.3	AT Service Interfaces	124
6.2.3	Allocation II	127
6.2.3.1	Service Provision Capability	127
6.2.3.2	AT Protocol Functions	128
6.2.3.3	AT Service Interfaces	129
6.2.4	Allocation I	132
6.2.4.1	Service Provision Capability	132

6.2.4.2	AT Protocol Functions	132
6.2.4.3	AT Service Interfaces	133
7.0	Conclusions	136
7.1	Suggestions For Future Work	139
8.0	References	150
Appendix A.	Call Examples	159
A.1	Circuit Call Setup and Clearing	159
A.2	Packet Call Setup and Clearing	170
A.2.1	Outgoing Packet Call: (D-channel)	173
A.2.2	Outgoing Packet Call: (existing B-channel)	175
A.2.3	Outgoing Packet Call: (new B-channel)	176
A.2.4	Incoming Packet Call: (network chooses the D-channel)	179
A.2.5	Incoming Packet Call: (network chooses an existing B-channel)	182
A.2.6	Incoming Packet Call: (user chooses the channel)	185
A.2.6.1	SAPI=s Call Offering	185
A.2.6.2	SAPI=p Call Offering	192
Appendix B.	SAPI=s Call Router	199
B.1	Generation of Unique Call References	201
B.2	Routing of Messages Received From the Terminals	204

B.3 Routing of Messages Received From Call Control

Functions 206

B.4 Provision of a Shared Link Establishment/Release

Interface 207

LIST OF ILLUSTRATIONS

Figure 1. ISDN Architecture	8
Figure 2. ISDN Service Provision Architecture	12
Figure 3. Function Model Example	17
Figure 4. Component Model Example	20
Figure 5. Allocation Examples	22
Figure 6. Layer 1 Implementation Model	44
Figure 7. Layer 2 Implementation Model	49
Figure 8. Layer 3 Implementation Model	55
Figure 9. Protocol Function Model	58
Figure 10. Architectural Component Model	69
Figure 11. Model Analysis Summary	89
Figure 12. Allocation I	90
Figure 13. Allocation II	91
Figure 14. Allocation III	92
Figure 15. Allocation IV	93
Figure 16. Allocation V	94
Figure 17. Subfunction Application Example	104
Figure 18. Subfunction Allocation Examples	105
Figure 19. Extended Model Analysis Summary	111
Figure 20. Allocation I	112
Figure 21. Allocation II	113
Figure 22. Allocation III	114
Figure 23. Allocation IV	115

Figure 24. Allocation V	116
Figure 25. Allocation VI	117
Figure 26. Call Examples Legend	160
Figure 27. Circuit Call Setup and Clearing	161
Figure 28. Outgoing Packet Call: (D-channel)	174
Figure 29. Outgoing Packet Call: (existing B-channel)	177
Figure 30. Outgoing Packet Call: (new B-channel)	180
Figure 31. Incoming Packet Call: (network chooses the D-channel)	181
Figure 32. Incoming Packet Call: (network chooses an ex- isting B-channel)	184
Figure 33. Incoming Packet Call: (user chooses a new B- channel)	188
Figure 34. Incoming Packet Call: (user chooses an exist- ing B-channel)	190
Figure 35. Incoming Packet Call: (user chooses the D- channel)	191
Figure 36. Incoming Packet Call: (user chooses a new B- channel)	194
Figure 37. Incoming Packet Call: (user chooses an exist- ing B-channel)	196
Figure 38. Incoming Packet Call: (user chooses the D- channel)	198

1.0 INTRODUCTION

1.1 INTRODUCTION TO ISDN

In recent years, the International Telegraph and Telephone Consultative Committee (CCITT) has been actively working towards the development of the I-series of recommendations which specify the characteristics of an Integrated Services Digital Network, or ISDN. The CCITT effort has largely been based on the following set of conceptual principles:

- "The main feature of an ISDN is the support of a wide range of service capabilities, including voice and non-voice applications, in the same network by offering end-to-end digital connectivity.
- "The key element of service integration for an ISDN is the provision of a limited set of standard multipurpose user-network interfaces. These interfaces represent a focal point both for the development of ISDN network components and configurations, and for the development of ISDN terminal equipment and applications.

- "An ISDN is recognized by the service characteristics available through user-network interfaces, rather than its internal architecture, configuration or technology. This concept plays a key role in permitting user and network technologies and configurations to evolve separately" (CCITT recommendation I.410 [1]).

In keeping with these principles, the majority of CCITT's ISDN standardization effort has been focused on the specification of a small number of user-network access interface structures as well as a set of integrated service access protocols which can be used to access the services provided at an interface by the network.

For the most part, CCITT's standard interface structures have been designed to allow support of a wide variety of bearer services, as described in recommendation I.211 [2]. Access to a specific bearer services is provided using a corresponding set of access protocol procedures. At present, however, only the procedures for circuit-mode services (including both circuit-mode voice and circuit-mode data), and packet-mode services (data only) have been completely defined in the I-series of recommendations. As new services are identified, access protocol procedures for these services may be added to the recommendations.

A more detailed introduction to the ISDN concepts and terminology relevant to this research document can be found in CCITT recommendations I.410 [1], I.211 [2], I.320 [3], I.411 [4], and I.412 [5]. In addition to the CCITT recommendations, a number of useful conference papers and technical articles are also available. Irmer [6] has written a paper which describes the birth of the ISDN concept, and the subsequent development of the I-series of recommendations from the CCITT perspective. Chen [7] and Kostas [8] have written articles which deal primarily with the business motivation behind ISDN, in terms of enhanced service provision, and economies of scale. Decina and Scace [9] have written a paper which gives a brief overview of the structure and content of the I-series recommendations. Excellent tutorials on ISDN concepts, terminology, and standards activity have been provided in papers by Pfyffer [10] and Davies [11]. Similar tutorial material can be found in references [12,13,14].

1.2 INTEGRATION VS INTERWORKING

Although evolution toward a fully-integrated, multi-service network is an important ISDN goal, the primary constraint on ISDN switching node implementation is the provision of integrated services access, according to the I-series protocol

recommendations. Given that only the access characteristics are constrained, a wide variety of internal switching node architectures can be used to provide initial ISDN implementations. Although the number of possible architectures is large, the majority can be grouped into the following categories:

- Integrated Architectures:

Switching node architectures where circuit and packet service functions are integrated within common switch components.

- Interworking Architectures:

Switching node architectures where circuit and packet service functions are implemented within distinct, interconnected switch components.

A survey of current industry literature would seem to indicate that the major switching equipment manufacturers are largely split between the integrated and interworking alternatives. Switching nodes based on integrated architectures include AT&T's 5ESS [15,16,17], GTE's EAX [18], Italtel's UT [19], ITT's System 12 [20,21], and NEC's NEAX61 [22,23], while those based on interworking include Northern Telecom's DMS-100 [24,25,26,27], Ericsson's AXE [28], Siemens' EWSD

[29,30], and the URN switch developed by the French PTT [31].

In general, the interworking approach appears to be favored by manufacturers which had a strong packet-switching product line, prior to the advent of ISDN. Using the interworking approach, it is possible for these manufacturers to provide integrated access to circuit and packet-mode services, by implementing a switching node based on the interworking of existing circuit and packet-switching products. The main advantages of this approach are listed below:

- Switching equipment manufacturers can reuse existing hardware and software, thus reducing the implementation effort required.
- Operating companies can provide ISDN services using existing installed networks, thus avoiding the cost of providing a new network, and allowing ISDN services to become generally available much sooner than would otherwise be possible.
- Operating companies can support both existing and new ISDN services simultaneously, using common switching equipment.

Within an interworking-based switching node architecture, the various access protocol and service functions required to provide ISDN access may be implemented within either the circuit switch component, the packet switch component, or any other components which complete the switching node implementation. For the most part, it is expected that functions which are specific to the provision of circuit-mode service should be implemented within the circuit-switching component, while functions which are specific to the provision of packet-mode service should be implemented within the packet-switching component. While keeping this goal in mind, there are a number of ways in which access protocol functions can be distributed amongst the various components of the switching node implementation. The purpose of this research document is to provide a systematic study of the various function distribution alternatives, in an attempt to identify those alternatives which are the most desirable for reasons of economy and/or practicality.

1.3 SCOPE

As has been stated previously, the only bearer services for which protocol procedures are currently well defined in the CCITT ISDN recommendations are circuit and packet-mode ser-

vices. Although proposals for new services, such as "frame switching" have been made [32], the protocol procedures for such services have yet to be adopted into the recommendations. Given the current state of the recommendations, coupled with the availability of existing circuit and packet switches and networks, this research document deals only with the provision of circuit and packet-mode services within an ISDN access framework. The impact of new services on the results of the study can be determined once protocol standards for the new services are adopted.

The problem of interworking circuit and packet-switches to support ISDN can be divided into two components:

- Interworking of a single circuit switch and a single packet switch to form an integrated services switching node, providing circuit and packet-mode services between locally connected users.
- Networking of switching nodes to provide widespread ISDN circuit and packet switching capability.

The relationship between these two components is illustrated in Figure 1 on page 8.

Although CCITT has specified the protocols which are to be used to support circuit and packet service networking (Sig-

Figure 1. ISDN Architecture

naling System No. 7 - ISUP and X.75 respectively), inter-working architectures which combine circuit and packet switches to provide an integrated services switching node remain an important open area of study. As such, this research document is focused on switching node architecture, rather than on switching node networking.

Within an integrated services switching node, it is assumed that the circuit-mode bearer service is entirely provided by the circuit switch, and that the packet-mode bearer service is entirely provided by the packet switch. The provision of each individual service can be divided into the following functional groups:

- Call processing functions.
- Operations, administration and maintenance functions (OA&M).

Call processing functions include those "real-time" functions which are performed in the handling and management of individual user service requests, or "calls". Some examples of call processing functions are listed below:

- Call control signaling.
- Tracking of call states.

- Validation of service requests by individual users.
- Arbitration between service requests by multiple users.
- Data transfer phase functions for packet calls.

OA&M functions, on the other hand, include those functions involved with the operation, administration and maintenance of the actual switching equipment. Some examples of OA&M functions are listed below:

- Storage, maintenance and distribution of user service profile and privileges information.
- Maintenance of switch hardware components, including fault detection and recovery.
- Collecting and reporting of user billing and accounting information.

Although both call processing and OA&M functions are equally essential, the existing ISDN standards are more strongly related to call processing than to OA&M. As such, this research document is focused primarily on the provision of call processing functions rather than OA&M functions.

In general, call processing functions can be divided into two categories:

- --Functions relating to the provision of basic calls.
- Functions relating to the provision of call-related features and facilities (supplementary services).

At present, however, the protocol procedures for basic calls are relatively well defined, while procedures for supplementary services are under debate, and have yet to be adopted within the ISDN standards. As such, this research document deals only with those call processing functions which are related to the provision of basic calls.

The relationship between the various service and functional groupings described in this section is illustrated in Figure 2 on page 12.

1.4 APPROACH

The goal of this research document is to provide a systematic study of the various ways in which access protocol functions can be distributed amongst the various components

Figure 2. ISDN Service Provision Architecture

of an interworking-based ISDN switching node, in an attempt to identify those implementation alternatives which are the most desirable for reasons of economy and/or practicality. The alternatives selected should adhere to the following requirements:

- The switching node implementation must provide standard ISDN access using the protocols as specified within the CCITT ISDN recommendations.
- The switching node implementation must provide integrated access to both circuit and packet-mode bearer services.
- The overall architecture of the switching node implementation should be based on the interworking of a circuit-switching component, a packet-switching component, and any other components required to complete the implementation.
- Functions should be allocated to the various components in a manner which has minimum impact on the architecture of existing circuit and packet-switching components.
- Functions which are specific to the provision of circuit-mode service should be provided within the circuit-switching component, while functions which are

specific to the provision of packet-mode service should be provided within the packet-switching component.

In order to achieve this goal, it is useful to introduce a formal model for the problem, which will allow the study to be carried out in a systematic and rigorous manner. In addition, the use of a model allows the study to generate results which have general applicability and significance, despite wide variations in the proprietary switch architectures and interface protocols supported by various switching equipment manufacturers.

Given the attributes of this problem, a formal model can be generated in the following steps:

1. Given the CCITT access protocol recommendations, derive a model describing a switching node implementation of the protocols, consisting of a set of communicating protocol functions.
2. Given the specified architecture of the interworking-based switching node implementation, derive a model which describes the properties of the architecture, consisting of a set of communicating architectural components.

3. Identify all of the practical constraints governing the implementations, and translate these constraints into the context of the model. Each constraint should in some way govern allocation of the model protocol functions to the various model architectural components.
4. Determine the attributes of an implementation which are desirable for reasons of economy and/or practicality, and translate these attributes into metrics which can be used to compare model implementations.

Once the requirements, constraints and metrics have been determined, the actual study of the problem can be carried out in the following steps:

1. Using a systematic algorithm, generate all possible allocations of protocol functions to architectural components, subject to the practical constraints. Each such allocation describes a function distribution implementation alternative.
2. From the set of allocations generated, use the metrics to select the "most desirable or practical" allocations.
3. For each selected allocation, characterize the functions performed within each architectural component, as well

as the information flows across the interfaces between the components.

Examples, illustrating the application of this modeling technique are described in the following sections.

1.4.1 Modeling of Protocol Functions

Given the CCITT recommendations describing the operation of the ISDN access protocols, it is possible to model a switching node implementation of these protocols in the form of a set of "elementary functions", "F". Given that the implementation of these functions should be based on a layered protocol hierarchy, where higher layer functions "use" service interfaces provided by lower layer functions, interactions between the functions can be modeled using a relation, "U", on the set F, where $f_2 U f_1$ implies that function f_2 directly uses a service interface provided by function f_1 . An example, illustrating this type of model is shown in Figure 3 on page 17.

Relation U can be used to govern function allocation based on the following rule:

$F = \{f1, f2, f3, f4, f5, f6, f7\}$

$U = \{(f2, f1), (f3, f1), (f4, f2), (f5, f3), (f6, f3),$
 $(f7, f5), (f7, f6)\}$

Figure 3. Function Model Example

Rule 1:

An allocation is "realizable" only if for all functions f_1 which are present in the allocation, all functions f , such that $f_1 \cup f$, are also present in the allocation.

Note that relation U is only used to represent situations where a particular function needs direct access to the service interface provided by another function. Using rule 1, it can be observed that if a given function, f_1 , uses another function, f_2 , realizable allocations cannot include f_1 , unless f_2 is also allocated. Also, f_2 cannot be allocated unless the functions which it uses are also allocated. If any of the functions which f_2 uses are not allocated, f_2 cannot be allocated, and, hence, f_1 cannot be allocated. Note that there is an indirect dependency of f_1 on the functions used by f_2 , which is effectively captured within rule 1. As such, there is no need to represent this dependency directly in relation U . Thus, relation U need not be transitive.

1.4.2 Modeling of Architectural Components

The switching node implementation architecture can be modeled as a set of architectural components, " C ", along with

an "interface relation", "I", on the set C , describing allowed interactions between component pairs. If an interface exists between two components, c_1 and c_2 , such that functions allocated to component c_1 can use functions allocated to component c_2 , this situation is stated mathematically as $c_1 \text{ I } c_2$. Note that the relation I should be reflexive, but need not be symmetric. An example, illustrating this type of model is shown in Figure 4 on page 20.

Using relation I, it is possible to restrict the allocation of functions to components according to the following rule:

Rule 2:

An allocation is "realizable" only if for all functions f_1 which are present in the allocation, where function f_1 is allocated to component c_1 , all functions f_i , such that $f_1 \cup f_i$, have been allocated to components c_i , such that for all c_i , $c_1 \text{ I } c_i$.

Note that rule 2 presents a more strict set of requirements than rule 1, in that functions, f_i , used by a function, f_1 , need not only to be allocated, but must also be allocated in components which can be directly accessed from the component in which f_1 is allocated. Thus, rule 2 supercedes rule 1.

$C = \{c1, c2, c3, c4\}$

$I = \{(c1, c1), (c2, c1), (c2, c2), (c3, c1),$
 $(c3, c3), (c4, c3), (c4, c4)\}$

Figure 4. Component Model Example

1.4.3 Modeling of Function Allocations

Once the required protocol functions and the architectural components have been modeled, each possible implementation can be represented as an allocation of a set of functions from set F, to a set of components from set C. An allocation is "realizable" only if the constraints specified in Rule 2 are met.

For each allocation, a relation "A" exists where $f1 \ A \ c1$ implies that function f1 has been allocated to component c1. Thus, an allocation can simply be described by listing the (function,component) pairs of the corresponding relation A. A number of allocation examples, based on the previous function and component model examples are shown in Figure 5 on page 22.

Example 1

$A = \{(f_1, c_1), (f_2, c_2), (f_3, c_3), (f_4, c_2), (f_5, c_3), (f_6, c_4), (f_7, c_4)\}$

Example 2

$A = \{(f_1, c_1), (f_2, c_1), (f_3, c_1), (f_4, c_2), (f_5, c_3)\}$

Figure 5. Allocation Examples

2.0 INTRODUCTION TO THE ACCESS REQUIREMENTS & PROTOCOLS

Before the protocol function model can be described in detail, it is first necessary to study the implementation requirements implied by the standard ISDN access protocols. In this section, a brief summary of the access protocols is provided, concentrating on those details which are relevant to the problem at hand. Gifford [33] has written a paper which gives a more detailed introduction to the structure and characteristics of ISDN user-network interfaces. Similar tutorial information is provided in references [34,35].

At present, the CCITT ISDN recommendations (I.420 [36] and I.421 [37]) have defined two standard access interface structures, as described below:

- Basic Rate Access (BRA):

A multipurpose access interface providing a small number of time-multiplexed user access channels, intended for normal subscriber applications.

- Primary Rate Access (PRA):

A multipurpose access interface providing a larger number of time-multiplexed user access channels, intended primarily for Private Branch Exchange (PBX) and main-frame computer access applications.

The protocols for both BRA and PRA interfaces are organized as a set of three protocol layers, as listed below:

- Layer 1 (Physical Layer).
- Layer 2 (Data Link Layer).
- Layer 3 (Call Control Layer).

Although the physical layer protocols are necessarily different for BRA and PRA, the data link and call control layer protocols used within the D-channel are largely the same for both access interface structures. As BRA is capable of supporting more complex topologies than PRA, the BRA circuit/packet integration requirements tend to be more complex as well. As a result, this document is focused primarily on a study of BRA integration requirements. Since PRA tends to support a subset of the BRA protocols, results obtained from a study of BRA should be applicable to PRA as well.

In the following sections the three access protocol layers are summarized, concentrating on those details which are relevant to circuit/packet interworking.

2.1 LAYER 1 (PHYSICAL LAYER)

BRA and PRA interface structures both support a set of time-multiplexed bi-directional digital information transfer channels. Each channel has a designated channel type, according to the application for which it is intended. Although CCITT has defined a number of different channel types, only two are currently targeted for common use. These two channel types are described below:

- B-channel:

Used to carry data or voice in circuit-switched applications. May also be used to carry packet data in circuit-switched connections to a packet handler.

- D-channel:

Used to carry signaling information (protocol layers 2 and 3) between users and the network. May also carry low-speed packet data.

The BRA interface structure provides two 64 kilobits per second (kbps) B-channels and one 16kbps D-channel. The PRA interface structure (in North America) provides 23 64kbps B-channels and one 64kbps D-channel, although it is possible to add more B-channels in groups of 24. In both BRA and PRA, the D-channel is used as a signaling channel, carrying layer 2 and 3 protocol information between the users and the network. B-channels, on the other hand are generally intended to provide transparent connections (through the network) between users, although it is possible to provide connections between users and a packet handler.

Although the PRA interface structure is intended for point-to-point topologies (one piece of equipment on either side of the interface), the BRA interface structure has been designed to support point-to-multipoint topologies, allowing a number of user terminals to be connected to a single BRA interface. All user terminals on a BRA interface contend for the same D-channel, using a frame-based contention resolution algorithm. The details of this algorithm are largely transparent to the network.

In addition to sharing the D-channels, users also share B-channels on a call-by-call basis. Although it is not currently possible for two terminals to use the same B-channel at the same time, a particular B-channel may be used by different terminals on subsequent calls. Arbitration for B-channel access is provided by the network as part of the layer 3 protocol.

The service interface provided at the layer 1/2 boundary by the layer 1 protocol has been specified by CCITT in the form of abstract service primitives. These primitives, serve to characterize the information flow across the layer boundary, without constraining the actual implementation of protocol entities or interfaces. The primitives which have been identified are listed below:

- PH-Data-Request:

Used by layer 2 to request transmission of a message frame to its peer entity. The actual content of the message frame is assumed to be conveyed across the boundary as part of the primitive. Note that the actual framing mechanism used on the D-channel is defined to exist within the layer 1 protocol rather than the layer 2 protocol.

- PH-Data-Indication:

Used by layer 1 to indicate to layer 2 that a message frame has been received from a peer entity. The actual content of the message frame is assumed to be conveyed across the boundary as part of the primitive.

- PH-Activate-Request:

Used by layer 2 to request activation of the physical interface.

- PH-Activate-Indication:

Used by layer 1 to indicate to layer 2 that the physical interface has been activated.

- PH-Deactivate-Request:

Used by layer 2 to request deactivation of the physical interface.

- PH-Deactivate-Indication:

Used by layer 1 to indicate to layer 2 that the physical interface has been deactivated.

More detailed information describing the layer 1 protocols for BRA and PRA can be found in recommendations I.430 [38]

and I.431 [39], as well as a tutorial paper by de Julio and Pellegrini [40].

2.2 LAYER 2 (DATA LINK LAYER)

The function of the layer 2 or data link layer protocols used within D or B-channels is to provide a mechanism to support error-free and reliable transmission of layer 3 message frames between users and the network. Within these protocols, frame checking sequences and sequence numbers are used in order to ensure that no frames are lost, and that frames are delivered in the same order in which they were sent.

The layer 2 protocol used in the B-channel (used only for B-channel access to a packet handling function) is referred to as LAP-B, while the layer 2 protocol used in the D-channel is referred to as LAP-D. While LAP-B is an existing protocol, originally defined to support X.25 access, LAP-D is a new protocol which was developed by CCITT specifically to support ISDN.

What sets the LAP-D protocol apart from many other link layer protocols is the use of frame addressing as a means to

support a multiplicity of "logical" data links on the same D-channel. Each logical link is identified by a Service Access Point Identifier (SAPI) and a Terminal Endpoint Identifier (TEI), allowing each terminal to have its own logical link to each service provided by the network, each link having its own control states and sequence numbers.

Although SAPIs are statically assigned to specific services, TEIs can be dynamically assigned to any terminal equipment connected to a BRA interface. In order to support dynamic TEI assignment, TEI management procedures have been included in the LAP-D protocol. Note that TEI management procedures are not required in the case of PRA as the access topology is always point-to-point.

In addition to the use of sequenced logical links between users and the network, the LAP-D protocol has also made provision for the broadcast of call control message frames from the network to all user terminals on a particular access interface. Frame broadcast is only required in the case of point-to-multipoint topologies, and therefore, it is not required for PRA access interfaces.

The service interface provided at the layer 2/3 boundary by the LAP-D protocol has been specified by CCITT in the form of abstract service primitives, similar to those defined for

the layer 1/2 boundary. The primitives which have been identified are listed below:

- DL-Establish-Request:

Used by a layer 3 protocol entity to request establishment of a particular LAP-D logical link.

- DL-Establish-Indication:

Used by the layer 2 protocol to indicate to a layer 3 entity that a particular LAP-D logical link has been established.

- DL-Release-Request:

Used by a layer 3 protocol entity to request release of a particular LAP-D logical link.

- DL-Release-Indication:

Used by the layer 2 protocol to indicate to a layer 3 entity that a particular LAP-D logical link has been released.

- DL-Data-Request:

Used by the layer 3 protocol to request that a layer 3 message be transferred to a peer entity across a particular LAP-D logical link. The actual content of the message is assumed to be conveyed across the boundary as part of the primitive.

- DL-Data-Indication:

Used by the layer 2 protocol to indicate to a layer 3 entity that a layer 3 message has been received from a peer entity on a particular LAP-D logical link. The actual content of the message is assumed to be conveyed across the boundary as part of the primitive.

- DL-Unit-Data-Request:

Similar to DL-Data-Request, except that the message is to be transmitted using unacknowledged operation rather than acknowledged operation.

- DL-Unit-Data-Indication:

Similar to DL-Data-Indication, except that the message was received using unacknowledged operation rather than acknowledged operation.

More detailed information describing the layer 2 protocol for BRA and PRA can be found in recommendations I.440 (Q.920) [41] and I.441 (Q.921) [42], as well as a tutorial paper by Kano [43].

2.3 LAYER 3 (CALL CONTROL LAYER)

The layer 3 or call control layer within the D-channel (often referred to by its CCITT recommendation numbers, Q.931 or I.451 [44]) describes the format of a set of call control messages, as well as protocol procedures specifying the messages which need to be exchanged between users and the network, in order to control the setup and takedown of circuit or packet calls.

In the current Q.931 standard, B-channel circuit-switched calls are entirely controlled by messages in the D-channel, while B-channel packet-switched calls can be established in a two-stage approach, using D-channel messages to establish a B-channel connection between a user and a packet handler, and standard X.25 procedures within the B-channel, once the connection has been made. Although a number of proposals have been made which would allow B-channel packet call setup to be handled entirely out-of-band (i.e. within the D-

channel), these procedures have yet to be incorporated into Q.931. As such, this document is concentrated primarily on the current two-stage procedures for packet call setup. A number of call examples, illustrating the use of two-stage procedures for packet calls are provided in Appendix A.

The various layer 3 protocol procedures specified and/or required by the Q.931 recommendation can be divided into the following categories:

- SAPI=s¹ Q.931 procedures:

Q.931 message exchange using a SAPI=s LAP-D logical link within the D-channel. Used to support the following call types:

The current Q.931 recommendations describe the use of two SAPI values, "0" or "s", which may be used for circuit call setup as well as incoming packet call offering, and "16" or "p", which may be used for D-channel access to the packet handler as well as incoming packet call offering. The choice of whether to use SAPI=s or SAPI=p procedures for packet call offering is a network implementation option.

- Incoming and outgoing circuit-switched calls including calls to a packet handler.
- Optionally, may be used to support incoming packet call offering and channel negotiation.
- SAPI=p¹ Q.931 procedures:

Q.931 message exchange using a SAPI=p LAP-D logical link within the D-channel. Optionally used to support incoming packet call offering and channel negotiation.

- D-channel X.25 procedures:

X.25 message exchange using a SAPI=p LAP-D logical link within the D-channel. Used to support D-channel packet calls.

- B-channel X.25 procedures:

X.25 message exchange using a LAP-B logical link within a B-channel. Used to support B-channel packet calls.

More detailed information describing the layer 3 protocol for BRA and PRA can be found in recommendations I.450 (Q.930) [45] and I.451 (Q.931) [44], as well as a tutorial paper by Kano [43]. Specific details describing the proce-

dures for accessing packet-mode services are provided in recommendations 1.462 (X.31) [46] and X.25 [47], as well as tutorial papers by Unsoy [48], Parodi [49] and Rumsey [50].

3.0 MODEL DESCRIPTION

The modeling process used for the purposes of this study can be summarized in the following steps:

1. Derive a protocol function model, describing an switching node implementation of the required protocols, as a set of communicating protocol functions.
2. Derive a architectural component model, describing the architecture of the interworking-based switching node, as a set of communicating architectural components.
3. Express practical implementation constraints within the context of the model.
4. Express desirable implementation properties in the form of metrics within the context of the model.

The details of these modeling steps are described in the following sections.

3.1 THE PROTOCOL FUNCTION MODEL

The definition of the protocol function model, based on the CCITT ISDN access protocol recommendations, should proceed according to the following principles:

- The switching node implementation should be modeled as a set of "elementary functions" which communicate and cooperate in order to provide the required protocol functionality.
- In order to allow maximum flexibility in the allocation of functions to different architectural components, the model should be divided into as many different elementary functions as possible, especially in cases where functionality required to support circuit-mode service can be separated from that required to support packet-mode service.
- Although it is desirable to divide the model into many elementary functions, in cases where a set of related protocol functions need to be implemented within the same architectural component (for reasons of practicality, economy or system performance), the related functions should be grouped into a single elementary function.

Clearly, even when the above principles are followed carefully, the modeling process is somewhat arbitrary, and may result in a variety of different models, depending on the desired granularity (i.e. "fineness" or "coarseness") of the division of elementary functions. In order to produce results which have general applicability, it is useful to base the modeling process on the CCITT recommendations, using the same functional divisions which the CCITT has used to describe the protocol functions as a basis for the divisions between the elementary functions.

Using this approach, the modeling process should proceed according to the following principles, in addition to the principles described above.

- The elementary functions which comprise the implementation model should be organized into layers based on the layered protocol model which has been defined by the CCITT.
- Where possible, the elementary functions which comprise the implementation model should correspond with state machines which have been identified by the CCITT. Additional functions may be introduced, however, whenever necessary to complete the implementation, or in order to increase allocation flexibility.

- Where possible, information flow between elementary functions should be based on the service primitive interfaces which have been identified by the CCITT.

Another advantage of using CCITT-defined structures as a basis for the model is that the functionality of the state machines and the primitive interfaces between the machines has already been fully specified within the recommendations, allowing this study to concentrate on the allocation of functions to architectural components, rather than on the detailed implementation of the functions themselves.

In addition to the modeling principles which have been identified, the definition of the protocol function model is based on the following assumptions:

- The given protocol function model is intended to represent an interface which is capable of supporting a multiplicity of users and access interfaces. This implies that an elementary function which represents a particular protocol state machine may, in fact, be implemented as an array of identical state machines, one for each concurrent user of that particular protocol function.
- Where primitive interfaces are defined between communicating elementary functions, it is assumed that sufficient addressing information is conveyed with each

primitive to allow identification of a particular interface, link, call or user, as required. The details of the addressing mechanisms which can be used are considered to be beyond the scope of this study.

The protocol function model used for the purposes of this study is described in detail in the following sections.

3.1.1 Layer 1 Protocol Functions

Given the functions performed within the layer 1 protocol, and the service primitive interfaces which have been identified, it is possible to divide the functionality of a layer 1 implementation into three elementary functions, as described below:

- **Physical Termination:**

The physical termination function handles the mechanical and electrical requirements of the physical interface, including the multiplexing of B and D channels and activation/deactivation of the physical interface. The physical termination function provides demultiplexed information channels corresponding to the B and D chan-

nels, and also provides the activation/deactivation primitive interface to layer 2.

- Channel Router:

The channel router function provides the means to route the information channels corresponding to the B and D channels between the the physical termination function and other elementary functions. The channel router is assumed to be controlled by a higher-layer management function, via the following primitive interface:

- CR-Connect <source channel>, <destination channel>:

Used by a higher-layer management function to request establishment of a connection between the source (B or D) channel, and a specified destination channel.

- CR-Disconnect <source channel>, <destination channel>:

Used by a higher-layer management function to request disconnection of an existing connection between the source (B or D) channel, and a specified destination channel.

- **D-Channel Frame Termination:**

The D-channel frame termination function provides the framing mechanism used to delimit the D-channel information channel into message frames. The D-channel frame termination function receives the D-channel information channel from the channel router function, and provides the data request/indication primitive interface to layer 2. Note that frame termination would normally be considered to be a layer 2 function. However, in the case of ISDN BRA, CCITT has specified the framing mechanism as part of layer 1, due to the relationship between the framing mechanism, and the layer 1 D-channel contention algorithm.

The layer 1 portion of the implementation model is illustrated in Figure 6 on page 44.

3.1.2 Layer 2 Protocol Functions

Given the functions performed within the layer 2 protocol, and the service primitive interfaces which have been identified, it is possible to divide the functionality of a layer

Figure 6. Layer 1 Implementation Model

2 implementation into six elementary functions, as described below:

- B-channel Frame Termination:

The B-channel frame termination function provides the framing mechanism used to delimit B-channel information into message frames. The B-channel frame termination function receives the B-channel information from the channel router function, and provides a data request/indication service interface to the LAP-B link control function (see below), similar to that provided by the D-channel frame termination function. Note that the B-channel frame termination function is considered to be a layer 2 function, unlike the D-channel frame termination function, which is considered to be a layer 1 function.

- LAP-B Link Control:

The LAP-B link control function provides protocol procedures for LAP-B links as specified by the existing X.25 recommendations. Functions performed include the handling of link states, sequence numbers, protocol timers and message retransmission. The LAP-B link control function uses the service interface provided by the B-channel frame termination function and provides a

service interface to layer 3, similar to the CCITT-specified layer 2/3 primitive interface for LAP-D.

- D-channel SAPI Router:

The D-channel SAPI router function receives LAP-D frames from the D-channel frame termination function, and, based on the SAPI in these frames, routes the frames to either the TEI management function (see below), the SAPI=s LAP-D link control function (see below), or the SAPI=p LAP-D link control function (see below). In the transmit direction the D-channel SAPI router receives frames from these three functions, and routes the frames to the D-channel frame termination function. The D-channel SAPI router uses the data request/indication primitive interface provided by the D-channel frame termination function, and provides a similar interface to the TEI management and LAP-D link control functions.

- TEI Management:

The TEI management function provides TEI management as specified by the CCITT recommendations. Functions performed include handling of TEI assignment procedures, TEI removal procedures and TEI checking procedures. The TEI manager uses the data request/indication primitive interface provided by the D-channel SAPI router func-

tion, and provides the following primitive interface to the LAP-D link control functions:

- TEI-Assign-Indication:

Used by the TEI manager to indicate to the LAP-D link control functions that a new TEI has been assigned.

- TEI-Remove-Indication:

Used by the TEI manager to indicate to the LAP-D link control functions that an existing TEI has been removed.

• SAPI=s LAP-D Link Control:

The SAPI=s LAP-D link control function provides protocol procedures for SAPI=s logical links as specified by the CCITT recommendations. Functions performed include the handling of link states, sequence numbers, protocol timers and message retransmission. The SAPI=s LAP-D link control function uses the data request/indication primitive interface provided by the D-channel SAPI router function, and provides the CCITT-specified layer 2/3 primitive interface to layer 3.

- **SAPI=p LAP-D Link Control:**

The SAPI=p LAP-D link control function is identical to the SAPI=s function, except that it is used for SAPI=p links, instead of SAPI=s links. LAP-D link control is modeled as two elementary functions rather than a single combined function in order to allow the flexibility to allocate the two functions to different components.

The layer 2 portion of the implementation model is illustrated in Figure 7 on page 49.

3.1.3 Layer 3 Protocol Functions

Given the functions performed within the layer 3 protocol, and the service primitive interfaces which have been identified, it is possible to divide the functionality of a layer 3 implementation into six elementary functions, as described below:

- **B-channel Manager:**

The B-channel manager function controls the allocation, deallocation, connection and disconnection of B-channels

Figure 7. Layer 2 Implementation Model

on behalf of the various call control functions (see below). The B-channel manager uses the primitive interface provided by the channel router function² and provides the following primitive interface to the call control functions:

- BCM-Allocate-Request <B-channel>:

Used by a call control function to request allocation of a B-channel to a particular call.

- BCM-Allocate-Confirm <B-channel>:

² Note that the B-channel manager is considered to be a layer 3 protocol function, and directly uses the channel router, which has been considered to be a layer 1 protocol function. Although this would seem to contradict the principles of Open Systems Interconnection (OSI) modeling, it should be noted that the channel router function can be viewed as an internal, management function within the switching node, and is not, strictly speaking, a layer 1 function from the OSI perspective. Thus, the placement of the channel router function within the layer 1 category is somewhat arbitrary.

Used by the B-channel manager function to indicate to a call control function that a requested B-channel allocation has either succeeded or failed.

- BCM-Deallocate-Request <B-channel>:

Used by a call control function to request deallocation of a B-channel from a particular call.

- BCM-Connect-Request <B-channel, destination channel>:

Used by a call control function to request connection of a B-channel to a specified destination channel.

- BCM-Disconnect-Request <B-channel>:

Used by a call control function to request disconnection of an existing B-channel connection.

• SAPI=p Call Router:

The SAPI=p call router routes information frames between the SAPI=p LAP-D link control function, and various call control functions (see below), based on the first octet of the information field of each frame. Frames having a

first octet value equal to the standard Q.931 protocol discriminator value are routed to the SAPI=p packet call offering call control function (see below), while all other frames are routed to the D-channel X.25 call control function' (see below). The SAPI=p call router uses the layer 2/3 primitive interface provided by the SAPI=p LAP-D link control function, and provides a similar interface to the call control functions.

- SAPI=s Call Control:

The SAPI=s call control function supports all Q.931 signaling procedures required for SAPI=s logical links, including the following:

Note that the first octet of all Q.931 messages is set to the standard Q.931 protocol discriminator value (hex 08), while in all X.25 packets, the most-significant-nibble of the first octet contains a general format indicator. Since the most-significant-nibble of the Q.931 protocol discriminator (hex 0) corresponds to a reserved general format indicator value, the first octet can be used to differentiate between Q.931 messages and X.25 packets.

- Q.931 signaling for incoming and outgoing circuit-switched calls, including outgoing calls to a packet handler.

- Incoming call offering and channel negotiation for incoming packet calls.

Functions performed include the generation and decoding of Q.931 messages, tracking of call states, handling of protocol timers and message retransmission. The SAPI=s call control function uses the layer 2/3 primitive interface provided by the SAPI=s link control function, as well as the primitive interface provided by the B-channel manager function.

- SAPI=p Call Control:

The SAPI=p call control function supports Q.931 signaling procedures used for SAPI=p packet call offering. The SAPI=p call control function uses the layer 2/3 primitive interface provided by the SAPI=p call router function, as well as the primitive interface provided by the B-channel manager function.

- D-channel X.25 Call Control:

The D-channel X.25 call control function provides protocol procedures for X.25 packet calls as specified by the existing X.25 recommendations. The D-channel X.25 call control function uses the primitive interface provided by the SAPI=p call router function.

- B-channel X.25 Call Control:

The B-channel X.25 call control function provides protocol procedures for X.25 packet calls as specified by the existing X.25 recommendations. The B-channel X.25 call control function uses the primitive interface provided by the LAP-B link control function.

The layer 3 portion of the implementation model is illustrated in Figure 8 on page 55

3.1.4 Protocol Function Model Summary

Given the protocol function models for the three protocol layers, it is possible to define the complete protocol function model, as summarized below. The set F, consists of the following functions:

Figure 8. Layer 3 Implementation Model

- f1: Physical Termination.
- f2: Channel Router.
- f3: D-Channel Frame Termination.
- f4: B-channel Frame Termination.
- f5: LAP-B Link Control.
- f6: D-channel SAPI Router.
- f7: TEI Management..
- f8: SAPI=s LAP-D Link Control.
- f9: SAPI=p LAP-D Link Control.
- f10: B-channel Manager.
- f11: SAPI=p Call Router.
- f12: SAPI=s Call Control.
- f13: SAPI=p Call Control.
- f14: D-channel X.25 Call Control.

f15: B-channel X.25 Call Control.

The "uses relation", U, consists of the following function pairs:

$$U = \{ (f2, f1), (f3, f2), (f4, f2), (f5, f2), (f5, f4), \\ (f6, f3), (f7, f6), (f8, f1), (f8, f6), (f8, f7), \\ (f9, f1), (f9, f6), (f9, f7), (f10, f2), (f11, f9), \\ (f12, f8), (f12, f10), (f13, f10), (f13, f11), \\ (f14, f11), (f15, f5) \}$$

The complete protocol functional model is illustrated in Figure 9 on page 58.

3.1.5 Service Functions

While the given protocol function model represents the protocols which are required in order to implement an ISDN interface, it does not provide a representation of the actual services which may be provided using the protocol implementation. In order to allow the implementability of a particular service to be expressed within the context of the model, it is useful to extend the model to include additional "service functions", which "use" the primitive inter-

Figure 9. Protocol Function Model

faces supported by the call control functions (f12-f15). Similar to the protocol functions, service functions are divided up as finely as possible, in order to maximize allocation flexibility. The service functions, and the call control functions which they use are listed below:

f16: Handling of outgoing circuit-mode calls, using new (i.e. not currently allocated) B-channels, excluding calls to the packet handler.

(uses function f12)

f17: Handling of outgoing packet-mode calls in the D-channel.

(uses function f14)

f18: Handling of outgoing packet-mode calls in an existing B-channel connection to the packet handler.

(uses function f15)

f19: Handling of outgoing packet-mode calls in a new B-channel connection to the packet handler.

(uses functions f12 and f15)

f20: Handling of incoming circuit-mode calls in a new B-channel (excluding calls from the packet handler).

(uses function f12)

f21: Handling of incoming packet-mode calls, where the network chooses to offer the call in the D-channel, without using Q.931 call offering procedures.

(uses function f14)

f22: Handling of incoming packet-mode calls where the network chooses to offer the call in an existing B-channel connection to the packet handler, without using Q.931 call offering procedures.

(uses function f15)

f23: Handling of incoming packet-mode calls, where the network uses SAPI=s call offering procedures to offer the call, specifying that the user may receive the call only on a new B-channel connection to the packet handler.

(uses functions f12 and f15)

) -
f24: Handling of incoming packet-mode calls, where the network uses SAPI=s call offering procedures to offer the call, specifying that the user may receive the call only on an existing B-channel connection to the packet handler.

(uses functions f12 and f15)

f25: Handling of incoming packet-mode calls, where the network uses SAPI=s call offering procedures to offer the call,, specifying that the user may receive the call only on the D-channel.

(uses functions f12 and f14)

f26: Handling of incoming packet-mode calls, where the network uses SAPI=s call offering procedures to offer the call, specifying that the user may receive the call on a new B-channel connection to the packet handler, or on an existing B-channel connection to the packet handler.

(uses functions f12 and f15)

f27: Handling of incoming packet-mode calls, where the network uses SAPI=s call offering procedures to offer the call, specifying that the user may receive the call on

a new B-channel connection to the packet handler, or on the D-channel.

(uses functions f12, f14 and f15)

f28: Handling of incoming packet-mode calls, where the network uses SAPI=s call offering procedures to offer the call, specifying that the user may receive the call on an existing B-channel connection to the packet handler, or on the D-channel.

(uses functions f12, f14 and f15)

f29: Handling of incoming packet-mode calls, where the network uses SAPI=s call offering procedures to offer the call, specifying that the user may receive the call on a new B-channel connection to the packet handler, an existing B-channel connection to the packet handler, or on the D-channel.

(uses functions f12, f14 and f15)

f30: Handling of incoming packet-mode calls, where the network uses SAPI=p call offering procedures to offer the call, specifying that the user may receive the call only on a new B-channel connection to the packet handler.

(uses functions f12, f13 and f15)

f31: Handling of incoming packet-mode calls, where the network uses SAPI=p call offering procedures to offer the call, specifying that the user may receive the call only on an existing B-channel connection to the packet handler.

(uses functions f13 and f15)

f32: Handling of incoming packet-mode calls, where the network uses SAPI=p call offering procedures to offer the call, specifying that the user may receive the call only on the D-channel.

(uses functions f13 and f14)

f33: Handling of incoming packet-mode calls, where the network uses SAPI=p call offering procedures to offer the call, specifying that the user may receive the call on a new B-channel connection to the packet handler, or on an existing B-channel connection to the packet handler.

(uses functions f12, f13 and f15)

f34: Handling of incoming packet-mode calls, where the network uses SAPI=p call offering procedures to offer the

call, specifying that the user may receive the call on a new B-channel connection to the packet handler, or on the D-channel.

(uses functions f12, f13, f14 and f15)

f35: Handling of incoming packet-mode calls, where the network uses SAPI=p call offering procedures to offer the call, specifying that the user may receive the call on an existing B-channel connection to the packet handler, or on the D-channel.

(uses functions f13, f14 and f15)

f36: Handling of incoming packet-mode calls, where the network uses SAPI=p call offering procedures to offer the call, specifying that the user may receive the call on a new B-channel connection to the packet handler, an existing B-channel connection to the packet handler, or on the D-channel.

(uses functions f12, f13, f14 and f15)

3.2 THE ARCHITECTURAL COMPONENT MODEL

The primary goal of this research document is to study the various ways in which integrated circuit and packet service access can be provided by distributing protocol functions within a switching node consisting of a circuit-switch, a packet switch, and any other components required to complete the implementation. The study attempts to identify, for a given level of service provision capability, those protocol functions which can be implemented within the circuit-switching component, those protocol functions which can be implemented within the packet-switching component, and those protocol functions which must be implemented within additional, shared components.

Although an actual interworking-based switching node would be implemented using existing circuit and packet switches, the architectural component model for the switching node should make use of generic architectural components, rather than being based on particular switching products, currently provided by various manufacturers. Thus, the equipment which provides circuit-switching functions can be represented as a generic circuit-switching (CS) component, while the packet-switching equipment can be modeled as a generic packet-switching (PS) component. The use of generic components

allows the study to produce results which have general applicability.

In order to minimize the impact of ISDN access on the existing switches which comprise a switching node implementation, it is useful to assume that the circuit switch simply views ISDN access as a new circuit-service access interface, to be provided in addition to other access interfaces and protocols which the switch presumably already supports. Similarly, the packet switch should simply view ISDN access as a new packet-service access interface. The impact of this assumption, stated in the context of the model is that while higher-layer protocol functions within the CS or PS components may use service interfaces provided by functions implemented in other components, functions implemented within the CS and PS components should not provide service interfaces to be used by functions implemented in other components. This restriction is based on the observation that provision of service interfaces to other equipment is generally not characteristic of the single-service access interfaces provided by existing switching equipment.

Given the above assumption, whenever a protocol function provides a service interface which must be used by both the CS and PS components, that function cannot be implemented within either the CS or PS components, and, therefore, must be allocated to some additional, shared component. Although

one of the goals of the study is to identify those functions which must be provided outside of the CS and PS components, the actual number and architecture of the additional, shared components required is not of primary concern. Thus it suffices to simply introduce one additional component into the model, the "Access Termination" (AT) component, which acts as a "catch-all", providing all functions which cannot be implemented in either the CS or PS components.

In summary, then, the components which comprise the architectural component model are listed below:

- Circuit Switch (CS):

Provides basic circuit-switching functions including circuit call processing, making and breaking of circuit-switched connections, circuit service access functions, circuit service networking functions and circuit service OA&M.

- Packet Switch (PS):

Provides basic packet-switching functions including packet call processing, support for the establishment and management of packet virtual circuits, packet service access functions, packet service networking functions and packet service OA&M.

- Access Termination (AT):

Provides those protocol functions which are "shared" between circuit and packet-switching services, and, therefore, cannot be implemented within either the CS or PS components.

Having determined the set of components, $C = \{CS, PS, AT\}$, which comprise the architectural component model, all that remains is to identify the "I" relation, which describes the allowed interactions between functions implemented within different components. Given the restriction that functions implemented in the CS or PS components cannot provide service interfaces to functions in other components, relation I can simply be described as listed below:

$$I = \{(AT,AT), (CS,AT), (CS,CS), (PS,AT), (PS,PS)\}$$

Note that relation I is reflexive, implying the obvious fact that functions implemented within a particular component should certainly be able to use other functions which are implemented within the same component.

A diagram, illustrating the architectural component model is given in Figure 10 on page 69.

$C = \{AT, CS, PS\}$

$I = \{(AT,AT), (CS,AT), (CS,CS), (PS,AT), (PS,PS)\}$

Figure 10. Architectural Component Model

3.3 ALLOCATION CONSTRAINTS

In order to reduce the number of allocations which need to be studied, it is useful to introduce allocation constraints into the model, which reflect various practical aspects of an actual switching node implementation. Although practical constraints tend to depend on the characteristics of the equipment which comprises the switching node, it is possible to introduce a number of generic constraints, which should apply to all switching node implementations.

In order to describe the generic constraints, it is useful to introduce the following terminology:

- An allocation is said to be "realizable" if for all functions, f_1 , which are present in the allocation, where function f_1 is allocated to component c_1 , all functions, f_i , such that $f_1 \cup f_i$ have been allocated to components, c_i , such that for all c_i , $c_1 \cap c_i$ (Rule 2).
- An allocation is said to be "service function maximal" if and only if there is no service function, f_1 , which is not allocated to any component, although there exists a component, c_1 , to which f_1 could be allocated, while still satisfying the realizability constraint. Given that service functions were introduced into the model

primarily in order to determine which services can be implemented for a particular protocol function allocation, there is no point in generating allocations where service functions can be allocated, but have been omitted.

- An allocation is said to be "protocol function minimal" if and only if there is no protocol function allocated which is not used by any allocated protocol or service functions. Given that the purpose of protocol functions is to provide the capability to support service functions, there is no point in generating allocations which contain superfluous protocol functions.

In addition to the generic constraints, practical implementation constraints, related to a particular switching node implementation, can be expressed in the form of allocation restrictions. Each allocation restriction limits the allocation of particular functions to particular components. Examples of allocation restrictions are described below:

- In order to make maximum use of existing switching equipment, service functions related to circuit switching should be restricted to the CS component, while those related to packet switching should be restricted to the PS component. Thus, for the given model, functions f16 and f20 should be restricted to the CS compo-

nent, while all other service functions should be restricted to the PS component.

- In order to minimize the complexity of the AT component, the higher-layer "call control" functions (f12 - f15) should be restricted to the CS and PS components. Given that both service functions and call control functions should be restricted to the CS and PS components, this restriction also ensures that the complex interactions between service functions and call control functions do not take place across inter-component interfaces.

Within the context of the model, allocation restrictions can be stated mathematically by introducing a new relation, "V", where $f1 \ V \ c1$ implies that function f1 may be allocated to component c1. Where a (function, component) pair is not described within relation V, the allocation of that function to that component is not allowed. Using the allocation restrictions described above, relation V for the given model can be described as listed below. Note that unless specified above, protocol functions should be allocatable to any of the AT, CS, or PS components.

$$V = \{(f1, AT), (f1, CS), (f1, PS), (f2, AT), (f2, CS), (f2, PS), \\ (f3, AT), (f3, CS), (f3, PS), (f4, AT), (f4, CS), (f4, PS), \\ (f5, AT), (f5, CS), (f5, PS), (f6, AT), (f6, CS), (f6, PS), \\ (f7, AT), (f7, CS), (f7, PS), (f8, AT), (f8, CS), (f8, PS), \\ (f9, AT), (f9, CS), (f9, PS), (f10, AT), (f10, CS), \\ (f10, PS), (f11, AT), (f11, CS), (f11, PS), (f12, CS), \\ (f12, PS), (f13, CS), (f13, PS), (f14, CS), (f14, PS), \\ (f15, CS), (f15, PS), (f16, CS), (f17, PS), (f18, PS), \\ (f19, PS), (f20, CS), (f21, PS), (f22, PS), (f23, PS), \\ (f24, PS), (f25, PS), (f26, PS), (f27, PS), (f28, PS), \\ (f29, PS), (f30, PS), (f31, PS), (f32, PS), (f33, PS), \\ (f34, PS), (f35, PS), (f36, PS)\}$$

Note that all relations, A , which specify allocations of protocol functions to architectural components (as described in section "Modeling of Function Allocations" on page 21), subject to the restrictions expressed in relation V , can be viewed as subsets of relation V , where within each relation, A , each protocol function is allocated to at most one component.

3.4 ALLOCATION METRICS

Given a set of allocations which satisfy the generic constraints as well as the specified allocation restrictions, it may still be desirable to select certain allocations over others as being more desirable for reasons of practicality and/or economy. In order to do this, it is useful to introduce metrics, which can be used to evaluate and compare allocations.

Before introducing metrics which can be used in the context of the model, it is useful to list some desirable implementation properties which can be measured using the metrics.

- Since the AT component represents new development within an actual switching implementation, it is desirable to keep the number of functions which are allocated to the AT to a minimum.
- For a given allocation of functions to the AT, it is desirable to maximize the higher layer service functions which are allocated to the existing CS and PS components.

In order to describe metrics which can be used to measure these properties, it is useful to introduce the following terminology.

AT Function Set:

For any particular allocation, the "AT function set" is defined as the set of functions which are allocated to the AT component.

AT Allocation Class:

Allocations which have identical AT function sets are said to belong to the same "AT allocation class".

Service Function Set:

For any particular allocation, the "service function set" is defined as the set of service functions which are allocated, regardless of the actual components to which individual functions are allocated.

Service Allocation Class: Allocations which have identical service function sets are said to belong to the same "service allocation class".

Using this terminology, various metrics can be introduced, as described in the following sections:

3.4.1 Minimum AT Complexity Metric (M1)

In order to measure the extent to which AT complexity has been minimized in a particular allocation, it is possible simply to count the number of functions in the AT function set for that allocation. This count can be referred to as the "M1 value" for the specified allocation.

Rather than calculating the M1 value for all allocations, and simply selecting the allocation with the lowest value, it is more useful to divide the set of all allocations into service allocation classes, selecting the allocation with the lowest M1 value within each class. The use of this approach makes it possible to study the relationship between minimum AT function sets and related maximum service function sets.

Note that this approach can be generalized by generating an "implementation cost value" for the allocation of each function to each component. The resulting allocation cost value would be obtained by adding up the implementation cost for

all the functions which are present in the allocation, and could be used to select allocations which provide a minimum total implementation cost within a particular service allocation class.

3.4.2 Maximum Service Provision Metric (M2)

In order to measure the extent to which service function allocation has been maximized in a particular allocation, it is possible, simply to count the number of service functions which have been allocated in that allocation. This count can be referred to as the "M2 value" for the specified allocation.

Rather than calculating the M2 value for all allocations, and simply selecting the allocation with the highest value, it is more useful to divide the set of all allocations into AT allocation classes, selecting the allocation with the highest M2 value within each class.

Given, however, that service functions may not be of equivalent value, it is more useful to subdivide each AT allocation class into subclasses, where allocations which have disjoint service function sets are separated into different

subclasses. Using this approach, the allocation having the highest M2 value can be selected from each subclass. Note that a particular allocation may appear in one or more subclasses, provided that its service function set is not disjoint from those of the other members of each class to which it belongs.

4.0 ALLOCATION ALGORITHMS

As has been described previously, the analysis of the given models should proceed according to the following steps:

1. Systematically generate all allocations, subject to, the practical constraints.
2. Use the metrics to select the "most desirable or practical" allocations.

Algorithms which can be used to generate allocations as well as to apply constraints and metrics are described in the following sections.

4.1 THE ALLOCATION ALGORITHM

In order to generate all possible allocations in a systematic manner, it is necessary to develop an allocation algorithm, which generates allocations in an efficient manner, while rejecting any allocations which do not meet the specified constraints.

The algorithm used for the purposes of this study is a simple "backtracking" algorithm, using trial and error to exhaustively generate all allocations. Although it may be possible to generate more efficient algorithms, the development of such algorithms is considered to be beyond the scope of this study.

Exhaustive generation of allocations can be simplified, provided that the following assumption can be proved true:

Assumption 1:

It is assumed that the set of functions, F , can be ordered into a list, such that for each function F in the list, all functions, f_1 , such that $f \cup f_1$, precede f in the list.

By inspection, it can be observed that the functions of the given protocol function model, listed in ascending numerical order (from f_1 to f_{36}) form such an ordering.

Given the ordered list of functions, where " $f(i)$ " indicates the " i "th member of the ordered function list, an exhaustive allocation algorithm can be stated using the following recursive algorithm.

```

procedure try(i);
begin
  for c = {CS,PS,AT} do begin
    if <"f(i)" can be allocated to "c"> then begin
      <allocate "f(i)" to "c">;
      if (i= max_function_index) then
        <check allocation>
      else
        try(i+1);
      <deallocate "f(i)">;
    end;
  end;
  {also try allocations with "f(i)" deallocated}
  if (i= max_function_index) then
    <check allocation>
  else
    try(i+1);
end;

```

Given this procedure, the generation of all allocations is initiated as follows:

```

program allocate;
begin
  <mark all functions as deallocated>;
  try(1);
end;

```

The step to check whether a particular function can be allocated to a particular component can be performed using the following algorithm:

```
function valid_allocation(i,c): boolean;  
begin  
    valid_allocation := false;  
    if <"f(i) V c"> then  
        if <uses requirements for "(f(i),c)" are met> then  
            valid_allocation := true;  
    end;
```

The step to check whether the uses requirements of a particular (function, component) pair can be met (i.e. the resulting allocation is realizable), can be performed using the following algorithm:

```

function uses_requirements_met(i,c): boolean;
begin
    for f1 := 1 TO f(i-1) do
        if <"f(i) U f1"> then
            if <"f1 is allocated"> then begin
                c1 := <the component to which function
                    "f" is currently allocated>;
                if not <"c I c1"> then
                    return(false);
            end
            else
                return(false);
        end
    return(true);
end;

```

The step to check whether generated allocations are protocol function minimal can be performed using the following algorithm:


```

function check_allocation: boolean;
begin
    for i := 1 to max_function_index do
        if <"f(i)" is allocated> then
            if <"f(i)" is a protocol function> then begin
                used := false;
                for j := i+1 to max_function_index do
                    if <"f(j)" is allocated> then
                        if <"f(j) U f(i)"> then
                            used := true;
                        if not used then
                            return(false);
                    end;
                end;
            end;
        end;
    end;
    return(true);
end;

```

The only constraint which has not been handled within the given algorithms is the check to ensure that an allocation is "service function maximal". Rather than checking allocations for this property, it turns out, to be more efficient to modify the original allocation algorithm, so that only allocations which satisfy this property are generated. The required modification can be obtained by observing that if, it is possible to allocate a service function to any component, solutions which do not have that function allocated need not be considered. The modified allocation algorithm is described below:

```

procedure try(i);
begin
    allocated := false;
    for c = {CS,PS,AT} do begin
        if <"f(i)" can be allocated to "c"> then begin
            <allocate "f(i)" to "c">;
            allocated := true
            if (i = max_function_index) then
                <check allocation>
            else
                try(i+1);
            <deallocate "f(i)">;
        end;
    end;
    if <"f(i)" is a protocol function>
    or (allocated = false) then
        {try allocations with "f(i)" deallocated}
        if (i = max_function_index) then
            <check allocation>
        else
            try(i+1);
    end;
end;

```

2

MICROD

4.2 METRIC ALGORITHMS

Each of the metrics which were described in section "Allocation Metrics" on page 74 can be implemented in the form of an allocation "filter", taking as input a set of allocations, and generating as output a subset of the original input allocation set. The operation of each metric filter proceeds according to the following steps:

1. Read in the input allocations.
2. Divide the input allocations into the classes, or subclasses as appropriate for the given metric.
3. Within each class or subclass calculate the metric value for each allocation and select only those allocations which have the lowest or highest metric value. Note that where two or more allocations are "tied" for the lowest or highest value, all such allocations should be selected.
4. Write out the selected allocations.

Using this approach, it is possible to select the "most desirable or practical" allocations by applying the metric filter algorithms in a cascaded manner, with each filter

taking as input, the output of the previous filter. The first filter algorithm takes as input the output of the allocator algorithm.

In general, the order in which metric filter algorithms are applied should depend on the relative priorities of the properties which the given metrics measure. In this case, however, both metrics can be considered to have equal priority. The effects of varying the filter application order can be studied by applying the given metric filters in the following orders, for the same allocator output:

- (M1, M2).
- (M2, M1).

5.0 MODEL ANALYSIS

Given the protocol function model and the component architecture model, the allocation algorithm, described in section "The Allocation Algorithm" on page 79 generates a total of 14076 allocations, of which, 496 allocations are protocol function minimal. Rather than describing each of these allocations, it is more useful to apply the various metric filter algorithms, described in section "Metric Algorithms" on page 86, and to describe the results generated by cascading these filter algorithms in various orders.

After applying the metric filter algorithms in both the (M1,M2) and (M2,M1) orders, it can be observed that both select the same five allocations, as tabulated in Figure 11 on page 89, and illustrated in Figure 12 on page 90 through Figure 16 on page 94.

The selected allocations represent the set of realizable switching node implementation options, subject to the given constraints and metrics, demonstrating the relationship between AT complexity and service provision capability. In each selected allocation, given the AT function set, service provision is maximized. Where access to required service interfaces cannot be provided for particular service func-

Function	I Comp.	II Comp.	III Comp.	IV Comp.	V-- Comp.
f1	AT	AT	AT	CS	PS
f2	AT	AT	AT	CS	PS
f3	AT	AT	CS	CS	PS
f4	PS	PS	PS	-	PS
f5	PS	PS	PS	-	PS
f6	AT	AT	CS	CS	PS
f7	AT	AT	CS	CS	PS
f8	CS	CS	CS	CS	PS
f9	PS	PS	-	-	PS
f10	AT	CS	CS	CS	PS
f11	PS	PS	-	-	PS
f12	CS	CS	CS	CS	PS
f13	PS	-	-	-	PS
f14	PS	PS	-	-	PS
f15	PS	PS	PS	-	PS
f16	CS	CS	CS	CS	-
f17	PS	PS	-	-	PS
f18	PS	PS	PS	-	PS
f19	-	-	-	-	PS
f20	CS	CS	CS	CS	-
f21	PS	PS	-	-	PS
f22	PS	PS	PS	-	PS
f23	-	-	-	-	PS
f24	-	-	-	-	PS
f25	-	-	-	-	PS
f26	-	-	-	-	PS
f27	-	-	-	-	PS
f28	-	-	-	-	PS
f29	-	-	-	-	PS
f30	-	-	-	-	PS
f31	PS	-	-	-	PS
f32	PS	-	-	-	PS
f33	-	-	-	-	PS
f34	-	-	-	-	PS
f35	PS	-	-	-	PS
f36	-	-	-	-	PS

Figure 11. Model Analysis Summary

Figure 12. Allocation I

Figure 13. Allocation II

Figure 14. Allocation III

Figure 15. Allocation IV

Figure 16. Allocation V

tions, these service functions necessarily remain unallocated. Similarly, in cases where protocol functions cannot be allocated, or cannot contribute to the allocation of further service functions, these protocol functions also remain unallocated.

Although the results of the allocation algorithm and metric filter algorithms are somewhat interesting, from an academic standpoint, they are not very interesting from a practical standpoint, due to the following deficiencies:

- None of the allocations generated is capable of supporting the complete set of service functions which have been identified.
- The given model is too restrictive in situations where a particular function does not require the complete service provided by a used function.

The effect of the second deficiency can be illustrated by comparing allocations I and II. The only difference between the two allocations is that in allocation I, function f10 (B-channel Manager) is allocated to the AT component, rather than in the CS component, allowing allocation of the following additional functions to the PS:

f13: SAPI=p Call Control

f31: Handling of incoming packet-mode calls, where the network uses SAPI=p call offering procedures to offer the call, specifying that the user may receive the call only on an existing B-channel connection to the packet handler.

f32: Handling of incoming packet-mode calls, where the network uses SAPI=p call offering procedures to offer the call, specifying that the user may receive the call only the D-channel.

f35: Handling of incoming packet-mode calls, where the network uses SAPI=p call offering procedures to offer the call, specifying that the user may receive the call on an existing B-channel connection to the packet handler, or on the D-channel.

Note that function f13 (SAPI=p Call Control) uses function f10 (B-channel Manager) in order to support service functions which require B-channel negotiation. However, none of service functions f31, f32 or f35 require B-channel negotiation. As such, these services could have been provided in allocation II, by implementing a subset of function f13, which does not provide B-channel negotiation, and, therefore, does not require access to function f10. In this case, allocations I and II would provide identical service, re-

sulting in the selection of allocation II, based on minimum AT complexity.

It can be concluded that in situations where a particular function does not require the complete service provided by a used function, the current requirement that the used function have access to all its used functions may be overly restrictive, and does not take into account the possibility of implementing useful subsets of protocol functions.

In order to overcome the two observed deficiencies, an extended model is described and analyzed in the following section.

6.0 EXTENDED MODEL DESCRIPTION AND ANALYSIS

In this section, an extended model is proposed, in an attempt to overcome the observed deficiencies in the given model. A description of the proposed extensions, and the analysis of the resulting model are provided in the following sections.

6.1 EXTENDED MODEL DESCRIPTION

In a previous section, the analysis of the given model revealed the following deficiencies:

- None of the allocations generated is capable of supporting the complete set of service functions which have been identified.
- The given model is too restrictive in situations where a particular function does not require the complete service provided by a used function.

By inspecting the model, it can be observed that the first deficiency results from the fact that function f12 (SAPI=s call control) may only be allocated within either the CS or PS components, but must be used by a variety of service functions. Some of the service functions must be allocated within the CS component, while others must be implemented within the PS component. Thus, it is not possible to allocate all of the service functions which need to use this protocol function.

One solution to this problem, would simply be to remove the restriction which prevents function f12 from being allocated to the AT component. Given that function f12 is a large, complex function, however, this would be counter to the goal of minimizing AT complexity.

Another solution is to attempt to split function f12 into smaller functions, allowing the functions which are required for circuit switching to be allocated to the CS component, while packet switching functions can be allocated to the PS component. In order to perform this split, the protocol function model should be modified as follows:

- Function f12 can be removed from the model, while adding the following new functions:

- SAPI=s Call Router (f12a):

The SAPI=s call router function routes information frames between the SAPI=s LAP-D link controller, and various call control functions (see below), based on a call routing algorithm. The details of this algorithm are too complex to be presented here, but are described in Appendix B. The SAPI=s call router uses the layer 2/3 primitive interface provided by the SAPI=s LAP-D link control function, while providing a similar interface to the call control functions.

- SAPI=s CS Call Control (f12b):

The SAPI=s CS call control function supports all Q.931 signaling procedures required to support circuit switched calls, excluding calls to and from the packet handler. Functions performed include the generation and decoding of Q.931 messages, tracking of call states, handling of protocol timers and message retransmission. The SAPI=s CS call control function uses the layer 2/3 primitive interface provided by the SAPI=s call router function, as well as the service interface provided by the B-channel manager function.

- SAPI=s PS Call Control (f12c):

The SAPI=s PS call control function supports all Q.931 signaling procedures required to support packet switched calls, including circuit switched calls to the packet handler, and incoming packet call offering. Functions performed include the generation and decoding of Q.931 messages, tracking of call states, handling of protocol timers and message retransmission. The SAPI=s PS call control function uses the layer 2/3 primitive interface provided by the SAPI=s call router function, as well as the service interface provided by the B-channel manager function.

- The "uses relation", U, should be modified, removing the following pairs,

(f12,f8), (f12,f10), (f16,f12), (f19,f12), (f20,f12),
(f23,f12), (f24,f12), (f25,f12), (f26,f12), (f27,f12),
(f28,f12), (f29,f12), (f30,f12), (f33,f12), (f34,f12),
(f36,f12)

and adding the following pairs:

(f12a,f8), (f12b,f10), (f12b,f12a), (f12c,f10),
(f12c,f12a), (f16,f12b), (f19,f12c), (f20,f12b),
(f23,f12c), (f24,f12c), (f25,f12c), (f26,f12c),
(f27,f12c), (f28,f12c), (f29,f12c), (f30,f12c),
(f33,f12c), (f34,f12c), (f36,f12c)

- The V relation (allocation restrictions) should be modified, removing the following pairs,

(f12,CS), (f12,PS)

and adding the following pairs:

(f12a,AT), (f12a,CS), (f12a,PS),
(f12b,CS), (f12b,PS),
(f12c,CS), (f12c,PS)

The second deficiency results from the fact that some protocol and service functions may not require the full service capability provided by certain of the protocol functions which they use. In these cases, it may not be necessary for the used functions to have access to all of the protocol functions which they in turn use. This effect may lead to superfluous allocation of protocol functions, as was observed in the analysis of the given model.

One approach which can be used to overcome this deficiency is to introduce the concept of "subfunctions". Subfunctions can be defined as functions which implement a subset of the functionality provided by an associated "root function". In cases where subdivision is useful, a particular protocol function can be replaced within the context of the model by a collection of subfunctions. The use of subfunctions is illustrated in Figure 17 on page 104 and Figure 18 on page 105.

The introduction of subfunctions is not intended to allow the implementation of root functions to be distributed accross multiple components, but rather, to determine the extent to which the associated root functions need to be implemented within single components. As such, the allocation of subfunctions should be constrained to ensure that all of the subfunctions associated with a particular root function are allocated to the same component.

Using the subfunction concept, the protocol function model should be modified as follows:

- Function f12c should be subdivided into the following subfunctions:

f12c-1: SAPI=s PS Call Control (B-channel Management).

Before:

- f4 and f5 depend on allocation of f1 and f2.

After:

- f4 does not depend on allocation of f2.

Figure 17. Subfunction Application Example

Figure 18. Subfunction Allocation Examples

(uses function f10).

f12c-2: SAPI=s PS Call Control (SAPI=s Signaling).

(uses function f12a).

- Function f13 should be subdivided into the following subfunctions:

f13-1: SAPI=p PS Call Control (B-channel Management).

(uses function f10).

f13-2: SAPI=p PS Call Control (SAPI=p Signaling).

(uses function f11).

- The "uses relation", U, should be modified, removing the following pairs,

(f12c,f10), (f12c,f12a), (f13,f10), (f13,f11),
(f19,f12c), (f23,f12c), (f24,f12c), (f25,f12c),
(f26,f12c), (f27,f12c), (f28,f12c), (f29,f12c),
(f30,f12c), (f30,f13), (f31,f13), (f32,f13),
(f33,f12c), (f33,f13), (f34,f12c), (f34,f13),
(f35,f13), (f36,f12c), (f36,f13)

and adding the following pairs:

(f12c-1,f10), (f12c-2,f12a), (f13-1,f10),
(f13-2,f11), (f19,f12c-1), (f19,f12c-2),
(f23,f12c-1), (f23,f12c-2), (f24,f12c-2),
(f25,f12c-2), (f26,f12c-1), (f26,f12c-2),
(f27,f12c-1), (f27,f12c-2), (f28,f12c-2),
(f29,f12c-1), (f29,f12c-2), (f30,f12c-2),
(f30,f13-1), (f30,f13-2), (f31,f13-2),
(f32,f13-2), (f33,f12c-2), (f33,f13-1),
(f33,f13-2), (f34,f12c-2), (f34,f13-1),
(f34,f13-2), (f35,f13-2), (f36,f12c-2),
(f36,f13-1), (f36,f13-2)

- The V relation (allocation restrictions) should be modified, removing the following pairs,

(f12c,CS), (f12c,PS),
(f13,CS), (f13,PS)

and adding the following pairs:

(f12c-1,CS), (f12c-1,PS), (f12c-2,CS), (f12c-2,PS),
(f13-1,CS), (f13-1,PS), (f13-2,CS), (f13-2,PS)

In order to deal with the constraint that associated subfunctions must be allocated to the same component, the por-

tion of the allocation algorithm which checks the validity of allocations should be modified as follows:

```
function valid_allocation(i,c): boolean;
begin
    valid_allocation := false;
    if <"f(i) V c"> then
        if <uses requirements for "(f(i),c)"
            are met> then begin
            valid_allocation := true;
            if <"f(i)" is a subfunction> then
                if <any associated subfunctions
                    are allocated> then
                    if <associated subfunctions
                        are not allocated to "c"> then
                        valid_allocation := false;
            end;
        end;
    end;
```

While subfunctions provide additional flexibility during the allocation stage, once the allocations have been generated, it is useful to transform the model, replacing the subfunctions with their associated root functions, before proceeding to the metric filter analysis stage. For purposes of metric filter analysis and model interpretation, the allocation results should be interpreted according to the following rules:

-
- If any of the subfunctions associated with a particular root function have been allocated to a particular component it is implied that the root function is implemented within that component.
 - If none of the subfunctions associated with a particular root function have been allocated, it is implied the root function has not been allocated.
 - The implementation of the root function need only provide the total functionality of the associated subfunctions which have been allocated.
 - The implementation of the root function need only use those service interfaces which are required by the associated subfunctions which have been allocated.

6.2 EXTENDED MODEL ANALYSIS

Using the extended model, the allocation algorithm generates a total of 197482 allocations, of which, 1186 allocations are protocol function minimal. After applying the metric filter algorithms in both the (M1,M2) and (M2,M1) orders, it can be observed that both select the same six allocations,

as tabulated in Figure 19 on page 111, and illustrated in Figure 20 on page 112 through Figure 25 on page 117.

By studying the allocations which were generated using the extended model, it can be seen that there is one allocation (I) which can provide all of the specified service functions, and five allocations (II-VI) which provide a subset of the service functions. Of these remaining allocations, two allocations (V and VI) are degenerate cases, having all allocated protocol functions implemented within the same component. These allocations are not of interest given that neither allocation can simultaneously provide both circuit and packet-mode service functions.

Allocations I through IV are described, in order of increasing AT complexity, in the following sections. Each allocation is described in terms of the service provision capability of the component, the protocol functions allocated to the AT component, and the service interfaces provided by the AT component.

Function	I Comp.	II Comp.	III Comp.	IV Comp.	V Comp.	VI Comp.
f1	AT	AT	AT	AT	CS	PS
f2	AT	AT	AT	AT	CS	PS
f3	AT	AT	AT	CS	CS	PS
f4	PS	PS	PS	PS	-	PS
f5	PS	PS	PS	PS	-	PS
f6	AT	AT	AT	CS	CS	PS
f7	AT	AT	AT	CS	CS	PS
f8	AT	AT	CS	CS	CS	PS
f9	PS	PS	PS	-	-	PS
f10	AT	CS	CS	CS	CS	PS
f11	PS	PS	PS	-	-	PS
f12a	AT	AT	CS	CS	CS	PS
f12b	CS	CS	CS	CS	CS	-
f12c-1	PS	-	-	-	-	PS
f12c-2	PS	PS	-	-	-	PS
f13-1	PS	-	-	-	-	PS
f13-2	PS	PS	PS	-	-	PS
f14	PS	PS	PS	-	-	PS
f15	PS	PS	PS	PS	-	PS
f16	CS	CS	CS	CS	CS	-
f17	PS	PS	PS	-	-	PS
f18	PS	PS	PS	PS	-	PS
f19	PS	-	-	-	-	PS
f20	CS	CS	CS	CS	CS	-
f21	PS	PS	PS	-	-	PS
f22	PS	PS	PS	PS	-	PS
f23	PS	-	-	-	-	PS
f24	PS	PS	-	-	-	PS
f25	PS	PS	-	-	-	PS
f26	PS	-	-	-	-	PS
f27	PS	-	-	-	-	PS
f28	PS	PS	-	-	-	PS
f29	PS	-	-	-	-	PS
f30	PS	-	-	-	-	PS
f31	PS	PS	PS	-	-	PS
f32	PS	PS	PS	-	-	PS
f33	PS	-	-	-	-	PS
f34	PS	-	-	-	-	PS
f35	PS	PS	PS	-	-	PS
f36	PS	-	-	-	-	PS

Figure 19. Extended Model Analysis Summary

Figure 20. Allocation I

Figure 21. Allocation II

Figure 22. Allocation III

Figure 23. Allocation IV

Figure 24. Allocation V

Figure 25. Allocation VI

6.2.1 Allocation IV

6.2.1.1 Service Provision Capability

Allocation IV provides call control functions f12b and f15, allowing provision of the following service functions:

- f16: Handling of outgoing circuit-mode calls using new (i.e. not currently allocated) B-channels, excluding calls to the packet handler.
- f18: Handling of outgoing packet-mode calls in an existing B-channel connection to the packet handler.
- f20: Handling of incoming circuit-mode calls in a new B-channel (excluding calls from the packet handler).
- f22: Handling of incoming packet-mode calls, where the network chooses to offer the call in an existing B-channel connection to the packet handler, without using Q.931 call offering procedures.

In summary, allocation IV is capable of providing only circuit service and provisioned (reserved) B-channel packet

service, and is incapable of providing switched B-channel packet service, D-channel packet service, or any form of incoming packet call offering.

6.2.1.2 AT Protocol Functions

In allocation IV, the following protocol functions are allocated to the AT component:

f1: Physical Termination.

f2: Channel Router.

In summary, in allocation IV, the AT acts as a simple channel terminator and router, routing the D-channel to the CS component and routing the B-channels either to the CS or PS components on a provisioned basis rather than a demand (per-call) basis.

6.2.1.3 AT Service Interfaces

In allocation IV, the AT provides the following service interfaces across the AT/CS link:

- The physical layer activation/deactivation service interface (i.e PH-Activate/PH-Deactivate).
- The channel router service interface (i.e. CR-Connect/CR-Disconnect).
- Demultiplexed B and D-channel information streams.

The following service interfaces are provided across the AT/PS link:

- The physical layer activation/deactivation service interface (i.e PH-Activate/PH-Deactivate).
- The channel router service interface (i.e. CR-Connect/CR-Disconnect).
- Demultiplexed B-channel information streams.

Since both the CS and PS components have access to the channel router service interface, it is necessary that the OA&M

functions for the CS and PS components are coordinated, so that both components are aware of which B-channels for a particular interface need to be routed to which components.

In order to provide the physical layer activation/deactivation service interface to both the CS and PS components, the AT component needs to arbitrate between conflicting requests which may be received from the two components. One approach to solve this problem is described in the following set of rules.

- If the interface is activated by a terminal, indications should be sent to both components.
- If the interface is activated by either the CS or PS components, the other component should be notified as if the interface had been activated by the terminal.
- In order to prevent one component from requesting deactivation while the other component is providing a service, only deactivations generated by a terminal should be actioned.

6.2.2 Allocation III

6.2.2.1 Service Provision Capability

In addition to the call control functions provided by allocation IV, allocation III also provides call control functions f13 (subfunction f13-2 only) and f14, allowing provision of the following additional service functions:

f17: Handling of outgoing packet-mode calls in the D-channel.

f21: Handling of incoming packet calls, where the network chooses to offer the call in the D-channel, without using Q.931 call offering procedures.

f31: Handling of incoming packet-mode calls, where the network uses SAPI=p call offering procedures to offer the call, specifying that the user may receive the call only on an existing B-channel connection to the packet handler.

f32: Handling of incoming packet-mode calls, where the network uses SAPI=p call offering procedures to offer the

call, specifying that the user may receive the call only on the D-channel.

f35: Handling of incoming packet-mode calls, where the network uses SAPI=p call offering procedures to offer the call, specifying that the user may receive the call on an existing B-channel connection to the packet handler, or on the D-channel.

In summary, allocation III is capable of providing circuit service, provisioned (reserved) B-channel packet service, D-channel packet service, and limited SAPI=p call offering, but is incapable of providing switched B-channel packet service, or SAPI=s packet call offering.

6.2.2.2 AT Protocol Functions

In addition to the protocol functions provided by the AT in allocation IV, allocation III requires the following functions to be implemented within the AT component:

f3: D-channel Frame Termination.

f6: D-channel SAPI Router.

f7: TEI Management.

In summary, in allocation III, the AT is required to terminate the D-channel framing, routing SAPI=s information frames to the CS component and SAPI=p frames to the PS component. In addition, the AT provides a TEI management function which is shared by both the CS and PS components.

6.2.2.3 AT Service Interfaces

In allocation III, the AT provides the following service interfaces across the AT/CS link:

- The physical layer activation/deactivation service interface (i.e PH-Activate/PH-Deactivate).
- The channel router service interface (i.e. CR-Connect/CR-Disconnect).
- Demultiplexed B-channel information streams.
- The primitive interface provided by the D-channel SAPI router (i.e. DL-Data-Request/Indication).

- The primitive interface provided by the TEI management function (i.e. TEI-Assign/Remove).

The following service interfaces are provided across the AT/PS link:

- The physical layer activation/deactivation service interface (i.e PH-Activate/PH-Deactivate).
- The channel router service interface (i.e. CR-Connect/CR-Disconnect).
- Demultiplexed B-channel information streams.
- The primitive interface provided by the D-channel SAPI router (i.e. DL-Data-Request/Indication).
- The primitive interface provided by the TEI management function (i.e. TEI-Assign/Remove).

In addition to the rules governing use of the physical layer activation and channel router service interfaces which were introduced for allocation IV, allocation III requires use of the following rules:

- Frames identified with SAPI=s are routed to the CS component, while frames identified with SAPI=p are routed to the PS component.
- Whenever a new TEI is requested by a terminal and assigned by the TEI manager, both the CS and PS components should be notified.
- Whenever an existing TEI is removed by the AT component (due to failure of the TEI checking procedures), both the CS and PS components should be notified.
- The CS and PS components are not allowed to remove TEIs.

6.2.3 Allocation II

6.2.3.1 Service Provision Capability

In addition to the call control functions provided by allocation III, allocation II also provides call control function f12c (subfunction f12c-2 only), allowing provision of the following additional service functions:

f24: Handling of incoming packet-mode calls, where the network uses SAPI=s call offering procedures to offer the call, specifying that the user may receive the call only on an existing B-channel connection to the packet handler.

f25: Handling of incoming packet-mode calls, where the network uses SAPI=s call offering procedures to offer the call, specifying that the user may receive the call only on the D-channel.

f28: Handling of incoming packet-mode calls, where the network uses SAPI=s call offering procedures to offer the call, specifying that the user may receive the call on

an existing B-channel connection to the packet handler, or on the D-channel.

In summary, allocation II is capable of providing circuit service, provisioned (reserved) B-channel packet service, D-channel packet service, and limited SAPI=p or SAPI=s packet call offering, but is incapable of providing switched B-channel packet service.

6.2.3.2 AT Protocol Functions

In addition to the protocol functions provided by the AT in allocation III, allocation II requires the following functions to be implemented within the AT component:

f8: SAPI=s LAP-D Link Control.

f12a: SAPI=s Call Router.

In summary, in allocation II, the AT is required to terminate the D-channel framing, handling SAPI=s link control as well as SAPI=s call routing, while relaying SAPI=p frames to the PS component. In addition, the AT provides a TEI management function which is shared by the PS component.

6.2.3.3 AT Service Interfaces

In allocation II, the AT provides the following service interfaces across the AT/CS link:

- The channel router service interface (i.e. CR-Connect/CR-Disconnect).
- Demultiplexed B-channel information streams.
- The primitive interface provided by the SAPI=s call router (i.e. DL-Data-Request/Indication).

The following service interfaces are provided across the AT/PS link:

- The physical layer activation/deactivation service interface (i.e PH-Activate/PH-Deactivate).
- The channel router service interface (i.e. CR-Connect/CR-Disconnect).
- Demultiplexed B-channel information streams.
- The primitive interface provided by the D-channel SAPI router (i.e. DL-Data-Request/Indication).

- The primitive interface provided by the TEI management function (i.e. TEI-Assign/Remove).
- The primitive interface provided by the SAPI=s call router (i.e. DL-Data-Request/Indication).

Given that in allocation II, the AT component provides the SAPI=s LAP-D Link Control function, (f8), the following rules must be followed, in order to allow shared access to the physical layer activation/deactivation service interface and the TEI management service interface:

- If the interface is activated by a terminal, an indication should be sent to the PS component.
- If the interface is activated by the AT component, the PS component should be notified as if the interface had been activated by the terminal.
- If the PS component requests deactivation, while the AT component is providing service, the request should be ignored. The AT component cannot request deactivation on its own, given that it has no knowledge of whether or not the PS component is providing service. Terminal deactivation requests can be actioned at any time.

- Frames identified with SAPI=s are retained within the AT component, while frames identified with SAPI=p are routed to the PS component.
- Whenever a new TEI is requested by a terminal and assigned by the TEI manager, the PS component should be notified.
- Whenever an existing TEI is removed by the AT component (due to failure of the TEI checking procedures), the PS component should be notified.
- The PS component is not allowed to remove TEIs.

The SAPI=s Call Router function is governed by the rules described in Appendix B.

6.2.4 Allocation I

6.2.4.1 Service Provision Capability

In addition to the call control functions provided by allocation II, allocation I also provides the full functionality of functions f12c and f13, allowing provision of all remaining service functions, including switched B-channel packet service, and full SAPI=s and SAPI=p call offering.

6.2.4.2 AT Protocol Functions

In addition to the protocol functions provided by the AT in allocation II, allocation II requires the following functions to be implemented within the AT component:

f10: B-channel Manager.

In summary, in allocation I, the AT is required to terminate the D-channel framing, handling SAPI=s link control as well as SAPI=s call routing, while relaying SAPI=p frames to the

PS component. In addition, the AT provides a TEI management function which is shared by the PS component, and a B-channel management function, which is shared by both the CS and PS components.

6.2.4.3 AT Service Interfaces

In allocation I, the AT provides the following service interfaces across the AT/CS link:

- The B-channel manager service interface (i.e. BCM-Allocate/Deallocate, BCM-Connect/Disconnect).
- Demultiplexed B-channel information streams.
- The primitive interface provided by the SAPI=s call router (i.e. DL-Data-Request/Indication).

The following service interfaces are provided across the AT/PS link:

- The physical layer activation/deactivation service interface (i.e. PH-Activate/PH-Deactivate).

- The channel router service interface (i.e. CR-Connect/CR-Disconnect).
- Demultiplexed B-channel information streams.
- The primitive interface provided by the D-channel SAPI router (i.e. DL-Data-Request/Indication).
- The primitive interface provided by the TEI management function (i.e. TEI-Assign/Remove).
- The B-channel manager service interface (i.e. BCM-Allocate/Deallocate, BCM-Connect/Disconnect).
- The primitive interface provided by the SAPI-s call router (i.e. DL-Data-Request/Indication).

The rules governing shared access to the physical layer activation/deactivation service interface and the TEI management service interface are identical to those for Allocation II.

Given that the PS has access to the B-channel manager service interface, which is capable of making and breaking B-channel connections, the channel router service interface is unlikely to be used.

The SAPI=s Call Router function is governed by the rules described in Appendix B.

7.0 CONCLUSIONS

Systematic analysis of the original and extended models has demonstrated that it is possible to provide integrated access to circuit and packet services in an integrated services switching node based on interworking. Within the switching node, it is possible to implement circuit service call control functions within a circuit-switching component, and packet service call control function within a packet-switching component. The extent to which service provision can be achieved depends on the implementation of certain shared functions in such a way that these functions can be accessed by both the circuit and packet switching components.

Analysis of the extended model reveals the following relationships between service provision capability and the sharing of protocol functions:

- In order to provide any combination of circuit and packet service functions within the same switch, the physical termination (f1) and channel router (f2) functions must be shared.

- In order to provide D-channel packet service, and limited SAPI=p call offering, the D-channel frame termination (f3), D-channel SAPI router (f6) and TEI management (f7) functions must be shared.
- In order to provide limited SAPI=s call offering, the SAPI=s LAP-D link control (f8) and SAPI=s call router (f12a) functions must be shared.
- In order to provide switched B-channel access to the packet handler, as well as complete SAPI=p and SAPI=s call offering, the B-channel manager (f10) function must be shared.

By inspecting these relationships, it can be observed that while some packet service functionality can be provided by splitting the circuit and packet access apart at lower protocol layers. It is not possible, however, to provide complete packet service functionality without providing relatively complicated shared functionality as high as layer 3. This result can be attributed to the following observations:

- Where SAPI=s incoming packet call offering is provided, this functionality must coexist with signaling related to circuit service calls on the same SAPI.

- Where B-channel negotiation is performed as part of incoming packet call offering, this negotiation must coexist with B-channel negotiation related to circuit service calls.

Since it is difficult to distribute call control functions by service, it might be argued the benefits of such distribution are outweighed by the costs. Certainly one of the main thrusts behind the introduction of new packet mode services for ISDN, is the the assertion that call control for circuit and packet services is most efficiently provided within a single processor [51].

While the results of the modeling process should be considered to be more important than the process itself, the usefulness and accuracy of the results obtained using the modeling and analysis technique described in this research document demonstrate that the technique provides an effective, systematic means to explore function allocation within distributed systems. It should be noted, however, that the results of the analysis are only as accurate as the models which are used to produce them. In order to produce good results, considerable thought must be given to the following aspects of the modeling process:

- Partitioning of protocol and service functionality.

- Translation of practical requirements into appropriate model constraints and metrics.

Once the problem under study has been modeled accurately, carefully constructed allocation and analysis algorithms should produce useful results.

7.1 SUGGESTIONS FOR FUTURE WORK

Based on the work described in this research document, the following suggestions for future study can be proposed:

- One of the major assumptions used in this research document is that while both the CS and PS components have access to the AT component, neither component can have access to each other (i.e. there is no CS/PS interface). The effect of adding a CS/PS interface can be studied by modifying the "I-relation", and analysing the resulting model. The following cases should be considered:
 - The CS component is allowed to access the PS component, but not vice versa.

- The PS component is allowed to access the CS component, but not vice versa.
- Both the CS and PS components can access each other.
- Within this research document, it has been assumed that the various components represented within the model are packaged together as a single switch, possibly provided by a single manufacturer. As such, no consideration was given to the provision of "open" intercomponent interfaces, based on protocol standards. However, given that there has been some discussion in various standards bodies concerning specification of a standard interface for packet handlers, it would be useful to study the impact of this approach on function allocation and service provision. As an example, an interface based on the CCITT primary rate access interface standard could be modeled as follows:
 - Provide two components, ET (corresponding to AT and CS) and PH (corresponding to PS).
 - Restrict allocation of the physical termination function to the ET component.

- Add appropriate protocol functions to the model, representing protocol functionality required to facilitate communication across the ET/PH interface.
 - Allow ET functions to use PH functions and PH functions to use ET functions, subject to the restrictions implied by the ET/PH protocol. In order to accurately model these restrictions, it may be necessary to add additional constraint relations to the model and analysis algorithms.
- The modeling and analysis carried out in this research document has been based on the current protocol recommendations I.451 (Q.931) and I.462 (X.31). However, it is clear from the literature [48,51,52,53,54] that a number of proposed extensions to these recommendations are likely to be adopted, in order to allow evolution towards new packet mode services. The proposed changes involve the following:
- Addition of the capability to provide LAP-D within B-channels, in addition to LAP-B. This capability allows establishment of multiple logical links within a B-channel, and is intended to allow more efficient utilization of the 64 kbps B-channels, as well as possible provision of new frame relay and switching services [48].

- Introduction of new call setup procedures for packet mode services, allowing packet calls to be established using out-of-band Q.931 signaling in the D-channel. This capability is intended to allow more efficient implementation of circuit and packet-mode services, especially within integrated architectures, where call control for both services is provided within a single processor, and also, to allow the integration of supplementary services for both circuit and packet-mode [48].

- With appropriate modifications to the protocol function model, it is possible to use the given modeling and analysis technique to explore the impact of these changes on function allocation and service provision in an interworking architecture.
- Although this research document describes the flow of information across the various intercomponent interfaces, specification of addressing techniques required to support a multiplicity of users and access interfaces was considered to be beyond the scope of the study. It would be useful to study the various ways in which such addressing might be provided. In addition, it was assumed that an interworking architecture consisted only of a single CS component and a single PS component. It would also be useful to study the use of addressing and

4
routing required to support a multiplicity of CS and PS components within the same switch.

- Another consideration which should be addressed is the handling of user service subscription parameters within an interworking architecture. Such parameters are used to provide service discrimination, allowing different subscribers to be provided with different levels of service. As an example, Bell Communications Research (Bellcore) has issued a technical recommendation [55], which provides a detailed description of subscription parameters which can be used to govern access to both circuit and packet-mode services. It would be useful to study various ways of implementing these subscription parameters in the context of the various switch architectures generated within this research document. The following aspects of implementation should be studied:

- Categorization of the proposed parameters according to the services which are affected (i.e., CS only, PS only, both).
- Distribution and maintenance of subscription parameter information across the various components of the architecture.

- Impact of various intercomponent addressing schemes on the distribution and maintenance of subscription parameter information.
- Impact of separated vs integrated subscription parameter OA&M for the CS and PS components. Study the possibility of providing an integrated OA&M "shell" which provides an integrated craftsperson interface, while distributing subscription parameter information to the appropriate components.
- Within this research document, no concern was given to delay and performance issues. It would be useful to study the impact of function distribution on overall system performance. Given that queueing and transmission delays for intercomponent interfaces are likely to be more significant than delays related to intracomponent interfaces, a simple measure of performance can be obtained by introducing a new metric which counts the number of times service interfaces are used across intercomponent interfaces. Using this metric, it is possible to select allocations which have superior performance characteristics. A more accurate measure could be obtained by identifying the number of messages which need to pass across intercomponent interfaces for a variety of call examples, and weighting the metric accordingly.

- One of the techniques used in this research document was the division of protocol functionality by service, in order to achieve maximum allocation flexibility. The impact of this approach is that nearly identical protocol functions often need to be allocated in different components, resulting in duplicated implementation effort. By combining identical protocol functions (such as SAPI=s LAP-D link control and SAPI=p LAP-D link control) into a single function, then analysing the resulting model, it is possible to study the impact of function division on service provision.
- Although the modeling and analysis technique used in this research document is general in nature, it is not capable of handling the following more complicated scenarios:
 - Situations where a pair of functions each need to use the other, or more generally, "uses loops".
 - Situations where the uses requirements of a particular function can be met using any of a number of different service interface combinations (i.e. "uses options").

Although it is possible to represent uses loops within the model, the proposed generation algorithm cannot be

used, given that the existence of uses loops violates one of the algorithm's assumptions (Assumption 1). A more time-consuming algorithm could involve simply generating all possible allocations, then testing each resulting allocation for realizability. The problem with this approach is that the number of allocations which need to be checked in this case is several orders of magnitude higher than that for the original algorithm. It would be useful to study the generation problem for uses loops, in an attempt to identify more efficient generation algorithms.

Although it is possible to handle uses options with the existing generation algorithm, it is not possible to represent uses options using the existing "U-relation". One way of dealing with this problem would be to extend the definition of the U-relation to allow a function to be related to a set of used function sets, rather than a single used function set. On the other hand, it might be possible to model uses options using model transformations. As an example, in cases where a particular function has an optional uses list, that function can be divided into a number of instances, one for each uses option. Within the resulting allocations, the function is considered allocated if any of the optional instances of the function are allocated. The problem with this approach, however, is that the introduction of multiple

instances creates optional uses for other functions which use the original function, possibly causing an explosion of protocol functions. It would be useful to study various ways of handling uses option situations, including extensions to existing relations, creation of new relations and model transformations. Where model transformations are introduced, it would be useful to consider the automation of the transformation process.

- The various constraint relations (U, I, V, etc.), introduced in the models used in this research document, were each introduced to model a specific type of practical constraint. In turn, the generation and analysis algorithms were specifically tailored to handle the set of specified relations. A more general approach to the problem of model analysis would be to provide a single means (possibly a single relation) to capture all allocation constraints, along with generalized generation and analysis algorithms. It would be useful to study the various ways in which the given practical constraints might be commonly represented.

Using this approach, the current set of relations can be viewed as a "parameterization" of the more general relation. It would be useful to study various ways of "parameterizing" the general relation, in an attempt to achieve greater modeling generality. Ideally the general

relation could be parameterized using a small set of relations, each of which deals with an "orthogonal" practical constraint.

The problem with this approach, however, is that generalized algorithms cannot take into account various properties of the practical constraint relations, which were used to increase the efficiency of the original allocation algorithm. It would be useful to study the relationship between various constraint properties and accompanying algorithm efficiencies.

- The allocation generation and analysis algorithms used in this research document are largely based on a trial and error "backtracking" approach, where all possible allocations which meet the required constraints are generated. Once allocations are generated, metrics are used to select the "most desirable" allocations. It would be useful to study the following possible improvements to model analysis:

- Integration of metric selection and allocation generation into a single step, in order to improve analysis efficiency.
- "Backwards analysis", where a desired set of service functions are identified, and the various ways of

providing the uses requirements for these functions are systematically generated.

- Transformation of the problem into a more common form (possibly compatibility relations/classes or a graph theoretical approach), allowing analysis to be carried out using existing, known algorithms.

8.0 REFERENCES

1. CCITT Recommendation I.410, "General Aspects and Principles Relating to Recommendations on ISDN User-Network Interfaces".
2. CCITT Recommendation I.211, "Bearer Services Supported by an ISDN".
3. CCITT Recommendation I.320, "ISDN Protocol Reference Model".
4. CCITT Recommendation I.411, "ISDN User-Network Interfaces - Reference Configurations".
5. CCITT Recommendation I.412, "ISDN User-Network Interfaces - Channel Structures and Access Capabilities".
6. T. Irmer, "An Idea Turns Into Reality - CCITT Activities on the Way to ISDN", IEEE Journal on Selected Areas in Communications, Vol. SAC-4, No. 3, May, 1986.
7. R. K. Chen, "ISDN - The Network of the Future", Telecommunications, Vol. 17, No. 12, December, 1983.

8. D. J. Kostas, "Transition to ISDN - An Overview", IEEE Communications Magazine, Vol. 22, No. 1, January, 1984.
9. M. Decina, E. L. Scafe, "CCITT Recommendations on the ISDN", IEEE Journal on Selected Areas in Communications, Vol. SAC-4, No. 3, May, 1986.
10. H.K. Pfyffer, "Overview of ISDN", IEEE International Conference on Communications '86, Vol. 1, June, 1986.
11. D. R. Davies, "Introduction to ISDN", IEE 4th Vacation School on Data Communications and Networks, September, 1985.
12. J. Claus, "ISDN-Services - Status and Preview on Service Standardization", IEEE International Conference on Communications '86, Vol. 1, June, 1986.
13. G. Robin, "New Signals From Technology", Telephone Engineer and Management, Vol. 87, No. 18, September, 1983.
14. A. M. Rutkowski, "Integrated Services Digital Networks", Artech House, Inc., 610 Washington St., Dedham, MA 02026, 1985.
15. J. P. Delatore, H. Oehring, L. C. Stecher, "Implementation of ISDN on the 5ESS Switch", IEEE Journal on Se-

lected Areas in Communications, Vol. SAC-4, No. 8, November, 1986.

16. J. W. Johnson, N. J. Burzinski, J. A. Davis, J. Sharpless, "Integrated Digital Services on the 5ESS System", IEEE International Switching Symposium '84, May, 1984.
17. R. A. Fishel, "ISDN Implementation in the 5ESS Switch", IEEE Global Telecommunications Conference '86, Vol. 1, 1986.
18. K. T. Ma, J. Coalwell, "Integrated Circuit and Packet Switching Capabilities in the ISDN Environment", IEEE International Switching Symposium '87, Vol. 1, 1987.
19. F. Melindo, G. Valbonesi, "Network and System Architectures: the Integrated Approach to ISDN in UT Line", IEEE Journal on Selected Areas in Communications, Vol. SAC-4, No. 8, November, 1986.
20. A. Chalet, R. Drignath, "Data Module Architecture Including Packet Operation", Electrical Communication, Vol. 59, No. 1/2, 1985.

21. J. E. Cox, R. E. Pickett, "Role of the Digital Adjunct in Network Enhancement", Electrical Communication, Vol. 59, No. 1/2, 1985.
22. A. Kitamura, T. Kondo, A. Minato, Y. Yoshida, T. Takabatake, H. Manabe, "ISDN Adjunct Switch for Existing Central Offices and NEAX61 ISDN Digital Switching System", IEEE International Conference on Communications '86, Vol. 2, June, 1986.
23. D. Gorzocoski, "Adjunct Systems for ISDN Services", IEEE Midcon '86, September, 1986.
24. R. de Hoog, R. G. Wood, "Digital Switch Architecture for ISDN", IEEE International Switching Symposium '87, Vol. 1, 1987.
25. C. L. Wong, R. Wood, "DMS-100 Evolution - An ISDN Perspective", IEEE Global Telecommunications Conference '86, Vol. 1, 1986.
26. R. de Hoog, C. Wiebe, "A Switching Architecture for ISDN", IEEE International Conference on Communications '86, Vol. 2, June, 1986.

27. E. Juskevicius, "Provisioning of Packet Switching in ISDN", IEEE International Conference on Communications '86, Vol. 3, June, 1986.
28. M. Noren, C. G. Perntz, "AXE for Packet Switching Integration", IEEE International Conference on Communications '86, Vol. 2, June, 1986.
29. T. LeMinh, S. Cannon, "ISDN - CENTREX: The EWSD Approach", IEEE Global Telecommunications Conference '86, Vol. 2, 1986.
30. A. V. Kienlin, J. Klunker, "Packet Switching and ISDN - A Powerful Alliance", IEEE International Switching Symposium '87, Vol. 2, 1987.
31. G. Rathier, B. Mobasser, H. Savary, "Integrated Services Digital Subscriber Connection Unit and ISDN Bearer Service", IEEE International Switching Symposium '84, Vol. 2, May, 1984.
32. A. Le Roux, J. L. Dauphin, J. Laurens, "ISDN Packet Switching: From Experimentation to System Integration", IEEE International Switching Symposium '87, Vol. 1, 1987.

33. W. S. Gifford, "ISDN User-Network Interfaces", IEEE Journal on Selected Areas in Communications, Vol. SAC-4, No. 3, May, 1986.
34. S. Kano, "ISDN User-Network Interface Layers 2 and 3 Protocols: New Developments After 1984", IEEE International Conference on Communications '86, Vol. 1, June, 1986.
35. K. Gleen, "ISDN: Access and Interworking", IEE 4th Vacation School on Data Communications and Networks, September, 1985.
36. CCITT Recommendation I.420, "Basic User-Network Interface".
37. CCITT Recommendation I.421, "Primary Rate User-Network Interface".
38. CCITT Recommendation I.430, "Basic User-Network Interface - Layer 1 Specification".
39. CCITT Recommendation I.431, "Primary Rate User-Network Interface - Layer 1 Specification".

40. U. de Julio, G. Pellegrini, "Layer 1 ISDN Recommendation", IEEE Journal on Selected Areas in Communications, Vol. SAC-4, No. 3, May, 1986.
41. CCITT Recommendation I.440 (Q.920), "ISDN User-Network Interface Data Link Layer General Aspects".
42. CCITT Recommendation I.441 (Q.921), "ISDN User-Network Interface Data Link Layer Specification".
43. S. Kano, "Layers 2 and 3 ISDN Recommendations", IEEE Journal on Selected Areas in Communications, Vol. SAC-4, No. 3, May, 1986.
44. CCITT Recommendation I.451 (Q.931), "ISDN User-Network Interface Layer 3 Specification".
45. CCITT Recommendation I.450 (Q.930), "ISDN User-Network Interface Layer 3 General Aspects".
46. CCITT Recommendation I.452 (X.31), "Support of Packet Mode Terminal Equipment by an ISDN".
47. CCITT Recommendation X.25, "Interface Between Data Terminal Equipment (DTE) and Data Circuit-Terminating Equipment (DCE) for Terminals Operating in the Packet Mode on Public Data Networks".

48. M. Unsoy, "ISDN Packet Services Evolution", IEEE International Switching Symposium '87, Vol. 1, 1987.
49. R. Parodi, "ISDN Data Interfaces", IEEE International Conference on Communications '86, Vol. 1, June, 1986.
50. D. C. Rumsey, "Support of Existing Data Interfaces by the ISDN", IEEE Journal on Selected Areas in Communications, Vol. SAC-4, No. 3, May, 1986.
51. G. Barberis, U. Padulosi, "Packet Data Within ISDN", 1986 International Zurich Seminar on Digital Communications, New Directions in Switching Networks, March, 1986.
52. G. Barberis, M. Guarneri, P. Macrina, "Handling Packet Services Within ISDN", Computer Communications, Vol. 10, No. 3, June, 1987.
53. N. Baba, N. Morita, M. Tashiro, J. Matsumoto, K. Yamazaki, "An ISDN Switching System with Advances Service Capabilities", IEEE International Switching Symposium '87, Vol. 2, 1987.
54. G. Robin, "Evolution of ISDN Towards New Packet Modes", ISDN, Proceedings of the International Conference (Europe), Vol. 1, June, 1986.

55. Bellcore Technical recommendation TR-TSY-000268, "ISDN Basic Access Call Control Switching and Signaling Requirements".

APPENDIX A. CALL EXAMPLES

The current Q.931 procedures for setup and clearing of circuit and packet calls are described in the following sections. The information in these sections has been derived from I-series recommendations I.451 (Q.931) and I.462 (X.31), as well as Bellcore Technical Recommendation TR-TSY-000268 [55].

A number of figures are provided in the following sections, illustrating the various call examples. A legend, explaining the use of symbols and abbreviations in these figures is provided in Figure 26 on page 160.

A.1 CIRCUIT CALL SETUP AND CLEARING

The steps involved in the setup and clearing of a basic circuit-switched call are listed below. The Q.931 messaging required for the call is illustrated in Figure 27 on page 161.

Symbols

- ← = Q.931 messages, sent over a SAPI=s link in the D-channel.
- ← = Q.931 messages, sent over a SAPI=p link in the D-channel.
-|||..... = X.25 messages, sent over a LAP-B link in a B-channel.
-|||..... = X.25 messages, sent over a SAPI=p link in the D-channel.
- [...] = Layer 3 messages

Abbreviations

Layer 2

- UI = Unnumbered Information frame
- I = Information frame
- (a,b) = Layer 2 Address (TEI=a,SAPI=b)
- (B) = Layer 2 Address (LAP-B)
- x,y = LAP-D TEIs

Layer 3

- BC = Bearer Capability
- CID = Channel Identification
- CDN = Called Party Number
- CGN = Calling Party Number
- CSE = Cause

Notes

- Exchange of control frames for link establishment and release is not shown.
- Link layer acknowledgement frames are not shown.
- Establishment and clearing of multiple calls in B-and D-channel links is not shown.
- Carrying of X.25 Incoming Call message fields in the Q.931 SETUP message is not shown.

Figure 26. Call Examples Legend

Figure 27. Circuit Call Setup and Clearing

1. An ISDN terminal initiates a call by sending a SETUP message to the network. The SETUP message may contain the following information:

- Protocol discriminator (used to identify the Q.931 protocol).
- Call reference (used to identify the call to which a given message is associated).
- Message type (in this case, the message type indicates a SETUP message).
- Bearer capability information (used to indicate the type of call (i.e. voice/data) and any parameters associated with the call type such as voice encoding algorithm, data rate, and data protocol).
- Channel identification information (used to indicate the B-channel, if any, which is preferred for the call).
- Called party number information (used to identify the destination of the call; may be specified using either the Keypad information element, or the Called Party Number information element).

- Calling party number information (used to identify the originator of the call; this information is useful when there are a number of terminals on the same access interface which have different addresses).
 - Additional information (may include calling or called party subaddresses, low and high layer compatibility information, and user-user information; all such information is not processed by the network, but is simply passed transparently to the call destination).
2. When the network receives the SETUP message, the following checks are performed:
- a. The message is checked for syntax errors, including the following:
- Message less than three octets long.
 - Invalid protocol discriminator.
 - Invalid call reference.
 - Call reference already in use on another call.
 - Missing mandatory information elements.

- b. The bearer capability information is checked against the user's subscription parameter information (stored within the network) to see if the use of that particular bearer capability is allowed.
- c. The channel identification information is checked to see if the requested or preferred B-channel is available and/or allowed according to the user's subscription parameter information.
- d. The called party number information is checked to ensure that a valid destination address has been specified.
- e. The calling party number information is screened to ensure that it specifies a valid address for the given access interface.
- f. When additional information is present, the user's subscription parameter information is checked to determine whether transfer of the given information is allowed for this user.
- g. If any of the checks listed above indicate that the call cannot proceed, the network sends a RELease COMplete message to the terminal indicating the

cause of the call failure. Otherwise the call continues according to the steps described below.

3. Once the network has determined that the SETUP request received from the terminal is valid and complete, the following steps are taken:

- a. Called party number information is used to route the call, identifying, in this case, the access interface of the called ISDN user.
- b. The called user's subscription parameter information is checked to determine the following:
 - 1) Whether or not a call with the given bearer capability is allowed for the called user.
 - 2) Whether or not the user has subscribed to the delivery of calling party information.
 - 3) Whether or not the user has subscribed to the delivery of additional call information.
- c. If no channel is available on the called user's interface to offer the call, the calling user is given "busy treatment". Otherwise, the call proceeds according to the steps described below.

d. Given that the network has determined that the call can be offered to the called user, the network sends a CALL PROCEEDING message to the calling user. The CALL PROCEEDING message contains channel identification information, identifying the B-channel which has been selected for the call.

e. The call is offered to the called user, according to the following steps:

1) A free B-channel is reserved for the call.

2) A new call reference is allocated for the call.

3) A SETUP message is sent to the called user, containing the following information:

- Protocol discriminator.
- Call reference.
- Message type.
- Bearer capability information (indicating the bearer capability which was specified by the calling user).

- Channel identification information (indicating the channel which has been reserved for the call).
- Calling party number information (this information will be present only if allowed by both the calling and called user's subscription parameter information).
- Called party number information.
- Additional call information (this information will be present only if allowed by both the calling and called user's subscription parameter information).

4. If the call is unacceptable to the called user, the called user will reject the call by sending a RELease COMplete message to the network, which will cause the calling user to receive "call rejected treatment". Otherwise, the user may respond with an ALERTing message, indicating that called party alerting has been initiated, and the call will proceed according to the steps described below. The ALERTing message contains channel identification information which may indicate either that the B-channel reserved by the network was acceptable, or that an alternate channel is preferred.

5. On receiving the ALERTing message from the called user the network checks the B-channel indicated by the called user, following channel negotiation procedures if necessary and where allowed. If the channel indicated by the called user is acceptable, the network sends an ALERTing message to the calling user. If any user-user information has been included in the called user's ALERTing message, this information will be sent to the calling user if allowed by both the calling and called user's subscription parameter information.
6. When the called user is ready to answer the call, a CONNECT message is sent to the network.
7. On receiving the CONNECT message from the called user, the network sends an CONNECT message to the calling user, and a CONNECT ACKnowledge message to the called user. If any user-user information has been included in the called user's CONNECT message, this information will be sent to the calling user if allowed by both the calling and called user's subscription parameter information. At this point the network completes the two-way channel connection between the calling and called users, and the call is considered to be active.

8. On receiving the CONNECT message, the calling user may send a CONNECT ACKnowledge message to the network. This message is optional, and is ignored by the network.
9. When either the calling or called users is finished with the call, that user disconnects from the B-channel and sends a DISConnect message to the network. The DISConnect message includes cause information, identifying the reason for call clearing.
10. On receiving the DISConnect message from one of the users, the network performs the following steps:
 - a. A RELease message is sent to the user which sent the DISConnect message.
 - b. A DISConnect message is sent to the other user, with the cause value as specified by the first user.
11. On receiving the RELease message the user which initiated the disconnect sends a RELease COMPLETE message to the network. On receiving the RELease COMPLETE message, the network releases the B-channel and call reference which had been allocated to the call.

12. On receiving the DISConnect message, the other user disconnects from the B-channel and sends a RELease message to the network.
13. On receiving the RELease message from the other user, the network releases the B-channel and call reference which had been allocated to the call, and sends a RELease COMPLETE message to that user.

A.2 PACKET CALL SETUP AND CLEARING

Procedures for the setting up and clearing of packet calls are more complicated than those for circuit calls, for the following reasons:

- Packet call setup and clearing may be "two-stage", first requiring Q.931 messaging to establish a B-channel connection to the packet handler, followed by normal X.25 call setup in the B-channel. When clearing the call, the X.25 call is cleared normally, then the B-channel connection may be cleared via Q.931 messaging.
- When originating a call, the calling user has the following options:

- The call may be originated in the D-channel, using standard X.25 procedures.
- The call may be originated in an existing B-channel connection to the packet handler (if present), using standard X.25 procedures.
- The call may be originated in a new B-channel connection to the packet handler, requiring the B-channel connection to be set up prior to using standard X.25 procedures in the B-channel.

When offering an incoming call to the called user, the network may choose one of the following options, depending on the parameters associated with the call and the called user's subscription parameter information:

- The call may be offered in the D-channel, using standard X.25 procedures.
- The call may be offered in an existing B-channel connection between the called user and the packet handler (if present), using standard X.25 procedures.
- The call may be offered to the called party using Q.931 procedures (using SAPI=s or SAPI=p'), using

channel negotiation procedures to allow the called user to choose the channel which the network should use to offer the call. The called user may be given some or all of the following options:

- The call may be offered in the D-channel, using standard X.25 procedures:
- The call may be offered in an existing B-channel connection between the called user and the packet handler (if present), using standard X.25 procedures.
- The call may be offered in a new B-channel connection between the user and the packet handler, requiring the new connection to be set up before using standard X.25 procedures in the B-channel.

The various procedures for call setup and takedown for the various options are described in the following sections.

CCITT has defined two different sets of Q.931 procedures for packet call offering and channel negotiation. One set uses SAPI=s messaging, while the other uses SAPI=p messaging. The choice of which set of procedures to use has been left as a network implementation option.

A.2.1 Outgoing Packet Call: (D-channel)

An outgoing packet call in the D-channel proceeds according to the following steps. The messaging required for the call is illustrated in Figure 28 on page 174.

1. In order to originate a packet call in the D-channel, the calling user sends an X.25 Call Request packet to the network within a LAP-D logical link. The logical link must be identified by SAPI=p (indicates packet service) and the TEI which has been allocated to that terminal. If necessary (i.e. the link has not already been established), the link must be established via an exchange of LAP-D control frames before the Call Request packet can be sent to the network.
2. Within the network, the Call Request packet is routed to a packet handler, which processes the packet and routes the call according to standard X.25 procedures. When the call is answered by the called user, the network returns a Call Connected packet to the calling user, and the call enters the data transfer phase.
3. If the calling user wishes to clear the call, a Clear Request packet is sent to the network. When the call has

Figure 28. Outgoing Packet Call: (D-channel)

been cleared, the network responds with a Clear Confirmation packet.

4. If the called user initiates call clearing, the network will send a Clear Indication packet to the calling user. The calling user will respond with a Clear Confirmation packet.
5. Multiple X.25 virtual calls may be established within a SAPI=p LAP-D link to the packet handler. After clearing the last virtual call in a particular LAP-D link, the link can be released via an exchange of control frames.

A.2.2 Outgoing Packet Call: (existing B-channel)

An outgoing call in an existing B-channel connection to the packet handler proceeds according to standard X.25 procedures as described in "Outgoing Packet Call: (D-channel)" on page 173 above. In this case, however, a single LAP-B link exists within the B-channel, and the X.25 packets are sent in this link, rather than in a SAPI=p link. If necessary (i.e. the link has not already been established), the link must be established via an exchange of LAP-B control frames before the Call Request packet can be sent to the network.

Multiple X.25 virtual calls may be established within an existing B-channel connection to the packet handler. When a packet connection is no longer required, it can be cleared using X.25 procedures as described in "Outgoing Packet Call: (D-channel)" on page 173. After clearing the last virtual call in a particular B-channel, the LAP-B link can be released via an exchange of LAP-B control frames.

The messaging required for the call is illustrated in Figure 29 on page 177.

A.2.3 Outgoing Packet Call: (new B-channel)

Setup of an outgoing packet call in a new B-channel connection to the packet handler consists of the following steps:

1. The circuit-switched B-channel connection to the packet handler is set up using standard Q.931 circuit-switched call setup procedures as described in "Circuit Call Setup and Clearing" on page 159, with the following exceptions:

Figure 29. Outgoing Packet Call: (existing B-channel)

- The bearer capability in the SETUP message indicates that the call is a packet-mode call.
 - The called party number information indicates that the call is destined to a packet handler.
 - When the network determines that the setup request is valid, a CALL PROCEEDING message is sent to the calling user, indicating the B-channel which has been selected for the call. Once the connection is set up, the network sends a CONNECT message to the calling user. The calling user may, optionally, respond with a CONNECT ACKNOWLEDGE message.
2. Once the B-channel connection has been established, the packet call is setup using X.25 procedures in the B-channel as described in "Outgoing Packet Call: (existing B-channel)" on page 175.

Once a B-channel connection to the packet handler has been established, multiple X.25 virtual calls can be established within that channel. When a packet connection is no longer required, it can be cleared using X.25 procedures as described in "Outgoing Packet Call: (D-channel)" on page 173. Once all virtual calls in a particular B-channel have been cleared, the LAP-B link can be released via an exchange of LAP-B control frames, and the B-channel connection can be

cleared using Q.931 procedures as described in "Circuit Call Setup and Clearing" on page 159.

The messaging required for setup and clearing of B-channel connections and packet calls are illustrated in Figure 30 on page 180.

A.2.4 Incoming Packet Call: (network chooses the D-channel)

When the network chooses to offer an incoming packet call in the D-channel, the call proceeds according to the following steps. The messaging required for the call is illustrated in Figure 31 on page 181.

1. In order to offer an incoming packet call in the D-channel, the network sends an X.25 Incoming Call packet to the called user within a SAPI=p logical link, using the TEI which has been allocated to the called user. If necessary (i.e. the link has not already been established), the link must be established via an exchange of LAP-D control frames before the Call Request packet can be sent to the called user.

Figure 30. Outgoing Packet Call: (new B-channel)

Figure 31. Incoming Packet Call: (network chooses the D-channel)

2. When the called user is ready to accept the call, a Call Accepted packet is returned to the network, and the call enters the data transfer phase.
3. If the called user wishes to release the call, a Clear Request packet is sent to the network. When the call has been cleared, the network responds with a Clear Confirmation packet.
4. If the calling user initiates call release, the network will send a Clear Indication packet to the called user. The called user will respond with a Clear Confirmation packet.
5. Multiple X.25 virtual calls may be established within a SAPI=p link to the packet handler. After clearing the last virtual call in a particular link, the link can be released via an exchange of LAP-D control frames.

A.2.5 Incoming Packet Call: (network chooses an existing B-channel)

When an incoming packet call is offered to the called user in an existing B-channel connection to the packet handler,

3 of /de 3

the call proceeds according to standard X.25 procedures as described in "Incoming Packet Call: (network chooses the D-channel)" on page 179 above. In this case, however, a single LAP-B link exists within the B-channel, and the X.25 packets are sent in this link, rather than in a SAPI=p link. If necessary (i.e. the link has not already been established), the link must be established via an exchange of LAP-B control frames before the Call Request packet can be sent to the network.

Multiple X.25 virtual calls may be established within an existing B-channel connection to the packet handler. When a packet connection is no longer required, it can be cleared using X.25 procedures as described in "Incoming Packet Call: (network chooses the D-channel)" on page 179. After clearing the last virtual call in a particular B-channel, the LAP-B link can be released via an exchange of LAP-B control frames.

The messaging required for the call is illustrated in Figure 32 on page 184.

Figure 32. Incoming Packet Call: (network chooses an existing B-channel)

A.2.6 Incoming Packet Call: (user chooses the channel)

In some cases, the network may decide (by looking at call parameters and called user subscription parameter information) that an incoming packet call should be offered to the user using Q.931 messaging, in order to allow the called user to specify the channel (new B, existing B or D) which should be used for the call. CCITT has identified two different sets of procedures which can be used for this purpose, as described in the following sections:

A.2.6.1 SAPI=s Call Offering

In the case of SAPI=s call offering procedures, packet call offering is integrated into the circuit-switched call control procedures. When offering an incoming call, the network broadcasts a Q.931 SETUP message (using the SAPI=s broadcast link) to offer the incoming call, as described in "Circuit Call Setup and Clearing" on page 159, with the following exceptions:

- The bearer capability information will indicate that the call is a packet-mode call.

- The channel identification information will indicate which channels (B and/or D) are acceptable for the call.
- The SETUP message may also include X.25-specific information, such as information rate information and X.25 facilities information.

If the called user can accept the call, a CONNECT message is sent to the network indicating the channel which has been chosen for the call. The procedures followed by the network depend on the channel which has been selected:

- If the user has requested a new B-channel connection, the network makes the connection, returns a CONNECT ACKnowledge message to the called user, then presents the incoming X.25 call in the new B-channel as described in section "Incoming Packet Call: (network chooses an existing B-channel)" on page 182.

Once a B-channel connection to the packet handler has been established, multiple X.25 virtual calls can be established within that channel. When a packet connection is no longer required, it can be cleared using X.25 procedures as described in "Incoming Packet Call: (network chooses the D-channel)" on page 179. Once all virtual calls in a particular B-channel have been cleared, the LAP-B link can be released via an exchange of LAP-B con-

trol frames, and the B-channel connection can be cleared using Q.931 procedures as described in "Circuit Call Setup and Clearing" on page 159.

The messages required in this case are illustrated in Figure 33 on page 188.

- If the user has requested an existing B-channel connection, the network will send a CONNECT ACKnowledge message followed by a RELease message to the called user. On receiving the RELease message, the called user will send a RELease COMPLETE message to the network. The network, then, presents the incoming X.25 call in the selected B-channel as described in "Incoming Packet Call: (network chooses an existing B-channel)" on page 182.

Multiple X.25 virtual calls can be established within an existing B-channel connection to the packet handler. When a packet connection is no longer required, it can be cleared using X.25 procedures as described in "Incoming Packet Call: (network chooses the D-channel)" on page 179. Once all virtual calls in a particular B-channel have been cleared, the LAP-B link can be released via an exchange of control frames.

Figure 33. Incoming Packet Call: (user chooses a new B-channel)

The messages required in this case are illustrated in Figure 34 on page 190.

- If the user has requested that the call be presented in the D-channel, the network will send a CONNECT ACKnowledge message followed by a RELease message to the called user. On receiving the RELease message, the called user will send a RELease COMPLETE message to the network. The network, then, presents the incoming X.25 call in the D-channel as described in "Incoming Packet Call: (network chooses the D-channel)" on page 179.

Multiple X.25 virtual calls can be established within a SAPI=p LAP-D link to the packet handler. When a packet connection is no longer required, it can be cleared using X.25 procedures as described in "Incoming Packet Call: (network chooses the D-channel)" on page 179. Once all virtual calls in a particular link have been cleared, the LAP-D link can be released via an exchange of control frames.

- The messages required in this case are illustrated in Figure 35 on page 191.

Figure 35. Incoming Packet Call: (user chooses the D-channel)

A.2.6.2 SAPI=p Call Offering

In the case of SAPI=p call offering procedures, packet call offering is handled separately from circuit-switched call offering. When offering an incoming call, the network broadcasts a Q.931 SETUP message (using the SAPI=p broadcast link) to offer the incoming call. Except that a different SAPI is used, the SETUP message is identical to that used for SAPI=s call offering, as described in "SAPI=s Call Offering" on page 185.

If the called user can accept the call, a CONNECT message is sent to the network (using a SAPI=p logical link) indicating the channel which has been chosen for the call. The procedures followed by the network depend on the channel which has been selected:

- If the user has requested a new B-channel connection, the network may (as a network option) terminate the SAPI=p procedures by sending a RELEASE message to the called user. The called user then replies with a RELEASE COMPLETE message.

In order to establish the specified B-channel connection, the network sends a new SETUP message to the called user (using SAPI=s and the TEI which was used in

the CONNECT message). On receiving the new SETUP message, the called user sends a CONNECT message to the network. The network then makes the connection, sends a CONNECT ACKnowledge message to the called user, and presents the incoming X.25 call in the new B-channel as described in section "Incoming Packet Call: (network chooses an existing B-channel)" on page 182.

Once a B-channel connection to the packet handler has been established, multiple X.25 virtual calls can be established within that channel. When a packet connection is no longer required, it can be cleared using X.25 procedures as described in "Incoming Packet Call: (network chooses the D-channel)" on page 179. Once all virtual calls in a particular B-channel have been cleared, the LAP-B link can be released via an exchange of LAP-B control frames, and the B-channel connection can be cleared using Q.931 procedures as described in "Circuit Call Setup and Clearing" on page 159..

The messages required in this case are illustrated in Figure 36 on page 194.

- If the user has requested an existing B-channel connection, the network may (as a network option) terminate the SAPI=p procedures by sending a RELease message to the called user. The called user then replies with a

Figure 36. Incoming Packet Call: (user chooses a new B-channel)

RELease COMplete message. The network, then, presents the incoming X.25 call in the selected B-channel as described in "Incoming Packet Call: (network chooses an existing B-channel)" on page 182.

Multiple X.25 virtual calls can be established within an existing B-channel connection to the packet handler. When a packet connection is no longer required, it can be cleared using X.25 procedures as described in "Incoming Packet Call: (network chooses the D-channel)" on page 179. Once all virtual calls in a particular B-channel have been cleared, the LAP-B link can be released via an exchange of LAP-B control frames.

The messages required in this case are illustrated in Figure 37 on page 196.

- If the user has requested that the call be presented in the D-channel, the network may (as a network option) terminate the SAPI=p procedures by sending a RELease message to the called user. The called user then replies with a RELease COMplete message. The network, then, presents the incoming X.25 call in the D-channel as described in "Incoming Packet Call: (network chooses the D-channel)" on page 179.

Figure 37. Incoming Packet Call: (user chooses an existing B-channel)

Multiple X.25 virtual calls can be established within a SAPI=p LAP-D link to the packet.handler. When a packet call is no longer required, it can be cleared using X.25 procedures as described in "Incoming Packet Call: (network chooses the D-channel)" on page 179. Once all virtual calls in a particular link have been cleared, the LAP-D link can be released via an exchange of control frames.

The messages required in this case are illustrated in Figure 38 on page 198.

Figure 38. Incoming Packet Call: (user chooses the D-channel)

APPENDIX B. SAPI=S CALL ROUTER

The function of the SAPI=s call router function (f12a), in the extended protocol function model, is to route messages, relating to various SAPI=s calls, between the SAPI=s LAP-D link control function (f8) and the following call control functions:

f12b: SAPI=s CS Call Control

f12c-2: SAPI=s PS Call Control (signaling subfunction).

Since the call routing function should be largely transparent to the call control functions, the service interface provided by the SAPI=s call router function should be identical to that provided by the SAPI=s LAP-D link control function (i.e. DL-Establish/Release/Data/Unit-Data).

While the SAPI=p call routing function can route frames based on the first octet in the information field of each frame, the SAPI=s call routing function is more complicated, given that calls related to circuit switching and packet switching use the same frame types and message formats, and can be carried within the same link. As such it is necessary

to provide an algorithm describing the operation of the SAPI=s call router.

Various calls within a particular logical link are identified using Q.931 call references. In the case of calls originated by the terminal, the call reference is generated by the terminal; otherwise, it is generated by the network. The functions provided by the SAPI=s call router, then, can be described as follows:

- Ensure that calls originated by the network, for a particular interface, have unique call references.
- Route messages received from the terminal to the appropriate call control function, based on the call reference.
- Route all messages received from the call control functions to the SAPI=s LAP-D link control function.
- Provide a shared link establishment/release interface to both call control functions.

Details describing how these functions can be implemented are provided in the following sections.

B.1 GENERATION OF UNIQUE CALL REFERENCES

Given that the call routing function is intended to be largely transparent to the call control functions, it is assumed that call references for Q.931 calls are generated by the call control functions. Given that the two call control functions which use this function are likely to be implemented in different components, some strategy must be used, in order to ensure that there are no call reference conflicts.

A simple strategy would be to assign different call reference spaces to the two call control functions (f12b and f12c-2). For example, if a user on a particular interface has subscribed to the use of one-octet call references, function f12b could be allowed to generate call reference values ranging from 1 to 63, while function f12c-2 could be allowed to generate values ranging from 64 to 127. In this case, the call router function does not need to provide any extra capability, except possibly the screening of messages, to ensure that the various components stay within their appropriate ranges. The only drawback of this strategy is that the assignment of ranges is quite rigid and may need to be maintained in both the CS and PS components.

Another strategy would be to let the call control functions generate whatever call references they like, but have the call router function modify the call reference in the message before sending it on to the link control function. Similarly, messages received from the link related to these calls would need to have the call reference modified before being passed on to the appropriate call control function. In order to provide this functionality, the call router function would need to maintain a table, storing tuples associating call references provided by the call control functions, with those actually used on the link. The table would be used as follows:

- If a message is received from a call control function with a call reference which is currently stored in the table, the call reference is modified to the associated call reference.
- If a message is received with a call reference which is not currently stored in the table (i.e. a new call reference), the call router would have to generate a new unique call reference, and store the appropriate tuple in the table.

A further complication of this approach is that the call router is not aware of when call references are released, so that tuples can be removed from the table. In order to han-

dle this problem it is necessary to add an additional service primitive to the interface provided by the call router, allowing the call control functions to inform the call router that a specified call reference has been released.

Another approach to the problem of generating unique call references is to have the call router provide a call reference manager function, allowing the call control functions to request unique call references and, subsequently, to release them. The following primitive interface could be used:

- CR-Assign-Request:

Used by a call control function to request assignment of a unique call reference.

- CR-Assign-Confirm:

Used by the call routing function to indicate that a unique call reference has been assigned. The assigned call reference would be carried along with the primitive.

- CR-Release-Request:

Used by a call control function to request release of a specified call reference.

B.2 ROUTING OF MESSAGES RECEIVED FROM THE TERMINALS

In order to route messages received from terminals (via the SAPI=s LAP-D link control function) to the appropriate call control function, it is necessary to maintain a table for each access interface, storing tuples associating each call reference with the related call control function. For each received message, the call router function extracts the call reference from the message, and routes the message to the call control function stored in the table against that call reference.

Given that a table must be used, it is necessary to determine the strategy for adding and removing tuples. An example of such a strategy can be described as follows:

- In cases where the call reference was generated by one of the call control functions, the tuple associating the call reference and the call control function can be stored in the table on receipt of the first message using that call reference.
- In cases where the call reference was generated by a terminal, it is necessary for the call router function to determine the appropriate call control function on receipt of the first message received with a particular

call reference. Once the appropriate call control function has been identified, an tuple can be stored in the table for subsequent messages.

Given that the first message to use a particular call reference must be the SETUP message, it is possible for the call router to route messages based on information contained in this message. In particular, for calls to the packet handler, the "transfer mode" field in the mandatory Bearer Capability (BC) information element must indicate "packet mode", while the field is set to "circuit mode" for circuit-switched calls. The call router function can use this information by parsing the SETUP message, obtaining the value of the transfer mode field, and routing the message accordingly. Packet mode calls would always be routed to function f12c-2, while circuit mode calls would always be routed to function f12b. If the BC information element was missing from the message, or formatted incorrectly, the call router function could send a RELease COMplete message to clear the call.

Although this assumption is true for basic calls, it may not be true in the case of call related features. An approach for message routing in a feature environment is considered to be beyond the scope of this study.

- In either case, it is necessary to add an additional service primitive to the interface provided by the call router, allowing the call control functions to inform the call router that a specified call reference has been released, so that the associated tuple may be removed from the table.

B.3 ROUTING OF MESSAGES RECEIVED FROM CALL CONTROL FUNCTIONS

Since all messages received from call control functions are routed to the SAPI=s LAP-D link control function, this routing function is trivial. In order to maintain the table associating call references with appropriate call control functions, however, the call router may check each message against the table, according to the following algorithm:

7

```
IF <received call reference is in the table> THEN
    IF <the message was received from a different
        call control function than that stored in the
        table> THEN
        <handle the error>
    ELSE
        <relay the message>
ELSE
    BEGIN
        <store an association between the received
            call reference and the appropriate call control
            function in the map>
        <relay the message>
    END
```

B.4 PROVISION OF A SHARED LINK ESTABLISHMENT/RELEASE INTERFACE

In order to ensure proper operation of a shared link establishment/release interface, the call router function should operate according to the following rules:

- If a link establishment indication is received from the SAPI=s LAP-D link control function, the indication should be passed to both functions f12b and f12c-2.
- If a link establishment request is generated by either call control function, on establishment of the link, a data link establishment indication should be passed to the other function.
- In order to prevent one call control function from requesting link release while the other function is providing calls, only the terminal will be allowed to initiate link release.

END

09-08-88

FIN