Open Service Support for ATM

Jacobus Erasmus van der Merwe

St. John's College
University of Cambridge

A dissertation submitted for the degree of
Doctor of Philosophy

September 1997

Summary

Asynchronous Transfer Mode (ATM) technology provides superior data transfer capabilities in an environment in which multiple services are provided and carried by a single network. Fully exploiting this potential is hampered by the assumption by standards bodies that a single control architecture, which was derived from a mono-service network, will fulfil the needs of all applications in such a multi-service environment.

While this weakness has been widely recognised, previous efforts to address it have met with limited success. This can be largely attributed to the fact that such attempts have often proposed to replace one monolithic system with another. Avoiding this "one-size-fits-all" approach, this dissertation presents an Open Service Support Architecture (OSSA), in which multiple control architectures can be operational simultaneously in the same physical network. In this manner different control architectures, which provide diverse functionality and are designed to different models, can be accommodated.

A key concept of the OSSA is the partitioning of switch resources by a software entity called a Divider. The subset of switch resources is called a switchlet, and the Divider allows each switchlet to be controlled by a separate control architecture. The Divider polices the invocations of a control architecture to contain it in its allocated switchlet. Switchlets are combined into virtual networks, and a software entity called the Network Builder automates this process. The Network Builder allows virtual networks of arbitrary topology to be dynamically created and modified, and each virtual network is therefore controlled by a separate instance of a control architecture. The dissertation presents a proof of concept implementation of the OSSA, and reports on the efficiency of various implementations of crucial components.

The dynamic creation of virtual networks in the OSSA means that the usage of resources in an ATM network now needs to be considered on three time scales: short time scales for cell switching, longer time scales for connection creation, and even longer time scales for virtual network creation. The use of measurement based estimates of effective bandwidth to effect resource management at the two longer time scales of interest is investigated and the results presented.

Finally, the flexibility offered by the OSSA enables the use of service specific control architectures (SSCAs). An SSCA is a control architecture which utilises service specific knowledge in its manipulation of network resources, thereby providing a more efficient service than would be possible with a general purpose control architecture. The design and implementation of an SSCA for continuous media conferencing is presented.

To Monique

Preface

Except where otherwise stated in the text, this dissertation is the result of my own work and is not the outcome of work done in collaboration.

This dissertation is not substantially the same as any I have submitted for a degree or diploma or any other qualification at any other university.

No part of my dissertation has already been, or is being currently submitted for any such degree, diploma or other qualification.

This dissertation does not exceed sixty thousand words, including tables, footnotes and bibliography.

Publications

Some of the work described in this dissertation has been published, and is described in references [van der Merwe95, van der Merwe97b] and [van der Merwe97a].

A patent application has been filed to protect some of the techniques described in this dissertation [van der Merwe96].

Acknowledgements

I would like to thank my supervisor, Ian Leslie, for his support, encouragement and advice during my time at the Computer Laboratory. I would also like to thank Roger Needham and Simon Crosby for advice and support, especially at the time when my stay in Cambridge was in doubt.

I am grateful for help, advice and insights offered by the members of the Systems Research Group, who ensured a stimulating work environment.

For reading and commenting on earlier drafts of this dissertation, I am indebted to Steve Pope, Ian Leslie, Chuck Kalmanek, Rebecca Isaacs, Simon Crosby, Richard Bradbury and Richard Black.

I would like to thank Niel Viljoen for first suggesting that I should consider further study at the Computer Laboratory and for his support ever since.

I am indebted to the Cambridge Commonwealth Trust, the Overseas Research Student awards scheme and the CSIR (South Africa), for their financial support.

I am eternally grateful for the love and support of my family, in particular that of my wife and daughter who had to forgo much during our stay in Cambridge.

Finally, honour and glory be to my heavenly Father, who has richly blessed me in so many ways.

Contents

List of Figures

List of Tables

Glossary

AAL	ATM Adaptation Layer
ABR	Available Bit Rate
ARPC	ANSA Remote Procedure Call
ATM	Asynchronous Transfer Mode
AVA	ATM Video Adapter
B-ISDN	Broadband Integrated Services Digital Network
CAC	Connection Admission Control
CBR	Constant Bit Rate
CD	Compact Disc
CDV	Cell Delay Variation
CLR	Cell Loss Ratio
CMIP	Common Management Information Protocol
CORBA	Common Object Request Broker Architecture
CRC	Cyclic Redundancy Check
CTD	Cell Transfer Delay
DAT	Digital Audio Tape
DCAN	Devolved Control of ATM Networks
DIMMA	Distributed Interactive Multimedia Architecture
DPE	Distributed Processing Environment
FIFO	First-In-First-Out
GIOP	General Inter-ORB Protocol
GSMP	General Switch Management Protocol
IDL	Interface Definition Language
IETF	Internet Engineering Task Force
IIOP	Internet Inter-ORB Protocol
IN	Intelligent Networks
IP	Internet Protocol
IPv4	Internet Protocol version 4
ISDN	Integrated Services Digital Network
ITU	International Telecommunications Union
ITU-T	Telecommunications Standardisation Sector of ITU
JPEG	Joint Photographic Experts Group
LAN	Local Area Network
LEC	LAN Emulation Client

LES	LAN Emulation Server
LGN	Logical Group Node
MAC	Media Access Control
MBS	Maximum Burst Size
MCR	Minimum Cell Rate
MCU	Multipoint Control Unit
MPLS	Multiprotocol Label Switching
NNI	Network-Network Interface
nrt-VBR	Non-Real-Time Variable Bit Rate
ODP	Open Distributed Processing
OMA	Object Management Architecture
OMG	Object Management Group
ORB	Object Request Broker
OSSA	Open Service Support Architecture
PABX	Private Automatic Branch Exchange
PCM	Pulse Code Modulation
PCR	Peak Cell Rate
PGL	Peer Group Leader
PNNI	Private Network-Network Interface
PTI	Payload Type Indicator
PVC	Permanent Virtual Connection
QoS	Quality of Service
RM	Resource Management
RPC	Remote Procedure Call
rt-VBR	Real-Time Variable Bit Rate
sCGF	Scaled Cumulant Generating Function
SCR	Sustainable Cell Rate
SCSI	Small Computer System Interface
SNMP	Simple Network Management Protocol
SRG	Systems Research Group
SVC	Switched Virtual Connection
TCLR	Target Cell Loss Ratio
TCP	Transmission Control Protocol
TINA	Telecommunications Information Networking Architecture
TINA-C	TINA Consortium
TMN	Telecommunications Management Network
UBR	Unspecified Bit Rate
UDP	User Datagram Protocol
UNI	User-Network Interface

UPC	Usage Parameter Control
VC	Virtual Connection
VCC	Virtual Channel Connection
VCI	Virtual Channel Identifier
VN	Virtual Network
VNAC	Virtual Network Admission Control
VP	Virtual Path
VPC	Virtual Path Connection
VPI	Virtual Path Identifier
VPN	Virtual Private Network
WAN	Wide Area Network

Chapter 1

Introduction

This dissertation presents the design and implementation of an Open Service Support Architecture (OSSA) for Asynchronous Transfer Mode (ATM) networks. The architecture deals with the out-of-band control of ATM networks, and aims to provide an open environment which imposes few restrictions on the introduction of new services. Indeed, the OSSA provides a framework in which user manipulation of network resources can be accommodated along with the conventional services provided by network operators.

Providing multiple services in the same physical network has long been the interest of the networking research community. Such a multi-service network is expected to offer multiple data transfer capabilities which provide different Quality of Service (QoS) guarantees to services and users. It is widely accepted that the data transfer capabilities of ATM currently afford the best solution to this problem.

Supplying QoS guaranteed data transfer, however, is only part of the solution to the problem of building a multi-service network. This dissertation deals with the remaining problem, namely the required control framework in which these services can be created and deployed with ease, thereby making ATM truly multi-service.

ATM technology provides a fairly clean separation between the control and data transfer functionalities of the network[1]. Control and data transfer functions

[1] This division is not complete because *in-band* control functions are used in the data path, for example to regulate cell loss priority. In the rest of this dissertation, unless specifically indicated otherwise, control will be taken to refer to *out-of-band* control.

are respectively called the *control plane* and *user plane* in ATM terminology. While significant work and innovation went into realising and improving the user plane of ATM, the same is unfortunately not true for the control plane. By and large the existing control plane solutions very much resemble those of earlier mono-service networks. This is true despite the fact that ATM is used in a variety of communications environments, each of which has different requirements in terms of the management or control of communication.

The intrinsic separation of control and data transfer has guided the solution to the ATM control problem addressed in this dissertation. In particular the separation of control and data transfer has been taken to its logical conclusion: The user plane of ATM is considered to provide an efficient data transfer capability or framework, to which any *control architecture* can be applied. The term control architecture is used to refer to the out-of-band control and management mechanisms and services operational in a network or virtual network (or part thereof)[2]. In this manner different control architectures can be applied to the same type of (switching) hardware depending on the environment in which the equipment is used.

As a logical extension of the above, it may be useful (and in some cases essential) to be able to have more than one control architecture simultaneously operational in the same physical network. A primary motivating factor in this case is the existence of different control architectures which provide different functionality. The obvious way to achieve this is to *divide* the resources responsible for data transfer in the network, so that different control architectures are effectively multiplexed on the same hardware. In keeping with a design philosophy that has been applied to protocol stacks [McAuley90], and operating systems [Leslie96] with good effect, this multiplexing is performed at the lowest possible level. Multiplexing at such a low level means that very few assumptions are made about the control architectures operational in an ATM network, or the services that they can provide.

[2]The difference between management and control is not always clear. For the purposes of this dissertation, management will be considered those functions concerned with the general *well-being* of the network, while control will be those functions which try to do something *useful* with the network.

1.1 Motivation

ATM was adopted by the International Telecommunications Union (ITU) as the bearer service for B-ISDN [Händel]. ATM's attractive capabilities led to its use in environments other than public networks and in particular an industrial group called the ATM Forum has taken the lead in producing various ATM related specifications. A great deal of effort within the ATM Forum and other standardisation bodies goes into modifying and extending existing standards and protocols for use with ATM. Many of the standards extended in this way were developed for early single service voice and data networks. This approach of extending existing standards is understandable in terms of the need to support existing services and more importantly, to expedite the introduction of ATM. It is questionable however, whether protocols and standards, or indeed the whole paradigm, developed to a completely different set of requirements, can be made to fit current and future needs in an acceptable way.

In particular, in the current paradigm the network is viewed by the user as a monolithic entity, or black box. The user is provided with a user-network interface (UNI), through which all services are to be requested by means of a signalling protocol [ATM Forum93]. This suggests that provision must be made within this one interface (and signalling protocol) for *all* services provided by the network. As a consequence the UNI needs to be modified each time a new service is introduced. Implementing all this functionality in a single protocol also results in complex software which is difficult to maintain and verify. Service requests signalled across the UNI are relayed within the network by a network-network interface (NNI) signalling protocol [ATM Forum96]. The NNI signalling protocol can then invoke the requested service within the network by appropriate manipulation of network control interfaces.

The two most important problems with the above model are the tight integration of network services with the signalling primitives across the UNI, and the completely closed nature of the network control interfaces allowing only one monolithic control architecture to be operational in the network. This model therefore does not cater for:

- the flexible introduction of new services,
- the existence of more than one service provider using the same infrastructure,

- the existence of third party services within the network,
- the use of third party software within the network.

These problems have been widely recognised and several proposals to address them have been proposed over an extended period [Minzer91, Bubenik91, Miller92, Olsen92, Doeringer93, Lazar95, Bale95, Cidon95, Iwata95, Crosby95a, Veeraraghavan95, Davie96, Hjalmtysson97, Newman97b]. There are a number of key observations one can make from these approaches:

- All agreed on the use of ATM as the data transfer mechanism.
- All were met with limited acceptance by the networking community.
- Almost all used different networking models as their starting point, and provided different functionality to their users.

The first observation confirms the earlier claim about the general acceptance of ATM's superior data transfer capability. Probably the most important reason for the lack of acceptance of these approaches was that almost all proposals involved completely replacing the existing control architecture with a new one. This naturally leads to resistance to the new approach. More importantly, the third observation leads to a questioning of current wisdom which assumes that there is one control architecture which will satisfy the needs of all users and all applications. It is the thesis of this dissertation that this "one size fits all" approach is fatally flawed, and that, far from promoting the acceptance of ATM, it impedes the use of ATM to the extent that ATM, despite its superior characteristics, might lose out against competing technologies.

The main aim then of the Open Service Support Architecture (OSSA) presented here is to create an environment in which the inadequacies of the legacy ATM control approach can be addressed, while at the same time not preventing the use of conventional or indeed any other solutions. Furthermore, the OSSA should be able to simultaneously accommodate these different control architectures and the services built around them in the same physical network. While the OSSA successfully addresses these primary concerns, the openness of the architecture also leads to a new way of thinking about networks and the way they are operated. These features will make an OSSA a crucial part of modern multi-service networks.

1.2 Contribution

The first prerequisite for the OSSA is an open, generic, low level, switch control interface. Such an interface is not functionally different from the way switches are controlled anyway, except for the fact that in this case the control interface is well defined and public. Different control architectures can therefore be implemented to utilise the control interface, which in turn means that third party software can be used within switches and the network. In an attempt to make the control interface generic, some abstraction of the underlying hardware will inevitably be necessary. However, this abstraction should be made at the lowest possible level so as to allow maximum flexibility to different control architectures.

The above approach was followed in defining the Ariel switch control interface introduced in this dissertation[3]. Rather than specifying the exact implementation of a switch control interface, Ariel attempts to define the *functionality* which should be provided by such an interface.

An open control interface is also fundamental to IP switching [Newman97b] for which the General Switch Management Protocol (GSMP) was defined [Newman96a]. The GSMP protocol is unfortunately not very general and will be an awkward match for any control architecture that uses the QoS guarantees of ATM. Early work by the Comet group at Columbia University also defined a switch control interface [Lazar95]. However this interface was defined at such a high level of abstraction that it would not have been useful to other control architectures. More recently a low level control interface along the lines of Ariel also appears to form part of the Xbind approach [Lazar96b].

While addressing many of the legacy problems listed in Section 1.1, an open switch control interface on its own still suffers from the "one-size-fits-all" problem. This implies that the inherent multi-service nature of ATM can not be fully realised, because only a single control architecture can be operational at any moment in time.

This problem is addressed in the OSSA by a mechanism which partitions switch resources into subsets which can be operated on by different control architectures. A subset of switch resources is called a *switchlet*, and the partitioning is performed by an entity called a *Divider*. The Divider is a thin layer of software which communicates with the switch via the Ariel control interface on the physi-

[3] Credit is due to Sean Rooney of the Computer Lab for suggesting Shakespeare's The Tempest as the basis of our name space.

cal switch. The Divider exports a switchlet Ariel control interface for each of the partitions it creates. Multiple control architecture instances each control their subset of resources (i.e. their switchlet) by means of this exported switchlet Ariel interface. This means that several control architectures can be simultaneously operational on one Divider (and physical switch), in much the same way that many processes run concurrently in a multitasking operating system. The partitioning of the switch is not static and switchlets can be dynamically created and modified. However, at any moment a control architecture knows what resources are allocated to its switchlet.

The switchlet Ariel interfaces provide the exact same functionality as the Ariel interface on the physical switch. A control architecture is therefore oblivious to the fact that it is now limited to a subset of switch resources. Since the Divider entity maintains the original one-to-one relationship with the Ariel interface on the physical switch, the switch is also unaware that several control architectures are now operational on it.

Since different control architectures, potentially from different vendors, can not be expected to cooperate, the Divider is required to perform a certain amount of policing over the usage of resources, and the invocations made by different control architectures, i.e. policing the control plane. The switchlet concept relies on existing in-band policing mechanisms to protect switchlets in the data path or user plane.

The switchlet approach is radically different from other "open" control architectures such as Xbind [Lazar96a] and Hollowman [Rooney97a] where an effort is made to allow openness *within* a specific control architecture, so that, for example, different routing algorithms can be used. In the switchlet approach, because the multiplexing is done at a very low level, different control architectures can be operational within the same physical network. This is crucial in being able to cater for both existing and new solutions.

Switchlets combine, according to the topology of the physical network, into *virtual networks*. Another component of the OSSA is an entity called the *Network Builder* which has the task of coordinating the creation of switchlets and the resulting virtual networks. The Network Builder starts up the correct control architecture in a newly created virtual network, and performs garbage collection when virtual networks are liberated. The control architecture operational in a virtual network therefore determines the type of the virtual network. It also ensures an appropriate overlap of resources so that, for example, the virtual connection (VC) address space of two connected switchlet ports overlaps.

Since switchlets can be created dynamically, this means that virtual networks can likewise be created dynamically. Within the constraints of the physical network the OSSA therefore allows:

- networks of arbitrary topology to be created,
- the running of arbitrary control architectures within virtual networks,
- the movement of resources between virtual networks.

The OSSA virtual networking environment is vastly more flexible than virtual path (VP) based virtual networks, which is the way virtual networks are currently created in ATM networks. VP based virtual networks are normally used to simplify switching and connection acceptance control (CAC), or to aggregate traffic of different types in separate virtual networks [Friesen96]. In [Chan96], VP based networks were extended by means of a concept called virtual path groups. Bandwidth can be moved between a number of VPs which are grouped together as a single entity. These approaches could be enhanced by the existence of an OSSA.

An obvious application of the OSSA is a *managed intranet service*. In such an environment, multiple virtual networks of the same type, e.g. IP switching [Newman97b], could be provided to customers, and all traffic for a particular customer will be limited to its virtual network[4]. Of course, the virtual networks need not be limited to be of the same type, and indeed different types of virtual networks could be provided to the same customer.

A slightly less obvious application of the OSSA is in *change management*. Say, for example, a new version of a control architecture needs to be introduced into a network. Currently this would require loading new software onto switches, restarting the switches and hoping for the best. In an OSSA environment, a new virtual network can simply be created, on a subset of the physical network if need be, and the new control architecture introduced in it. Once the new control architecture has been sufficiently tested, the resources of that virtual network can be increased and the virtual network with the old version released.

[4] In the case of IP switching, different controllers can be used for each virtual network, so that the processing resources will also be partitioned (albeit hard partitioned in this case).

The OSSA environment also allows the deployment of *Service Specific Control Architectures* (SSCAs). An SSCA is a control architecture that provides very specific functionality tailored to the needs of a specific service. This is in direct contrast to a *general purpose* control architecture such as the ATM Forum's UNI/PNNI control architecture [ATM Forum94b, ATM Forum96], which, by its very nature, tries to provide the functionality of all current and future services. An SSCA can exploit service specific knowledge to provide a more effective and efficient service. In addition, because it is service specific, an SSCA can potentially be much simpler than a general purpose control architecture. The crucial point is that the OSSA allows the use of such SSCAs together with general purpose control architectures, therefore exploiting the benefits of both.

One example of an SSCA is group communication in a video conferencing environment. The audio and especially the video streams produced in a mesh connected multi-party conference quickly deplete network bandwidth when the conference grows to even moderate size. At the same time it is well known that in an orderly conference a limited number of speakers is active at any time. Similarly the number of video streams could easily be limited to, say, feeds from the current and previous speakers and the conference chair.

This service specific knowledge can be exploited by an SSCA in the following manner. Novel connection structures can be created in the network with the emphasis on group communication and the sharing of network resources. Thereafter the conference floor control function can directly manipulate and incrementally change these connections, or parts thereof, as dictated by speaker movement. For example, the bulk of a multicast tree can stay intact, while video from the current speaker can be grafted in by a simple localised modification. Incrementally changing existing connections in this manner is not possible using a standard general purpose control architecture, where the only alternative would be to recreate a multicast tree from the new speaker[5]. The long time involved with such a connection setup would make it unusable in all but the smallest conferences.

The combination of an OSSA and SSCAs affords the possibility for vendors of ATM endpoint equipment to provide a complete and integrated solution to customers. For example, vendors of video conferencing equipment can bundle an integrated SSCA with their equipment which would not be dependent on the functionality provided by a standard general purpose control architecture.

[5] Current control architectures [ATM Forum94b] allow branches to be added to a multicast tree, but do not allow the root of the tree to change.

Finally, the OSSA holds the promise of global provisioning for public operators. Rather than being limited to the services that the local operator can provide, an operator doing business in a foreign country, can lease an "empty" virtual network on which it can provide its own control architecture. In this manner an operator would be able to provide the exact same service in all places that it operates, regardless of whether it owns the equipment or not[6]. Indeed, in such an environment it is possible for an operator to exist without owning any of its own equipment.

1.3 Outline

The organisation of the remainder of this dissertation is as follows.

Chapter 2 briefly considers some essential background material and describes the experimental environment in which implementation work was carried out. Major research projects currently undertaken in the Computer Laboratory are described in a research context section.

The OSSA is introduced in Chapter 3. The Ariel switch control interface is described, and it is shown how this is generalised to facilitate the switchlet concept. The Network Builder and the virtual network services are then described. Some consideration is given to the practicalities of bootstrapping the OSSA, as well as the security aspects involved.

Chapter 4 presents a Divider implementation as well as several implementations of the Ariel switch control interface. These are compared and the results presented. The chapter ends with a description of the Network Builder implementation.

Various aspects of bandwidth management are considered in Chapter 5. The need for bandwidth management at both the virtual connection and the virtual network levels is explained. The chapter presents an implementation of bandwidth management in the OSSA based on measured effective bandwidth.

Chapter 6 explains the need for service specific control architectures, and considers a number of environments which could benefit from it. The design and implementation of a video conferencing SSCA which utilises the OSSA is described.

[6]This example is due to interaction with members of staff at Telia Research, Sweden.

Related work is considered in Chapter 7 and compared to the work presented in this dissertation. The dissertation ends with a conclusion and direction for future work.

Chapter 2

Background

This chapter provides essential background information to the development of the OSSA concept. Section 2.1 starts with an overview of Asynchronous Transfer Mode (ATM), and introduces relevant terminology and concepts used by this technology. Since a Distributed Processing Environment (DPE) is used in some of the work presented here, the section then presents a similar overview of DPE terminology and technology. The section ends with a definition of the term "services" in the context of an OSSA. Section 2.2 describes the experimental environment in which implementation work was undertaken. Research projects in the Computer Laboratory which are related to the OSSA are discussed in Section 2.3.

2.1 General Background

2.1.1 Asynchronous Transfer Mode

Asynchronous Transfer Mode (ATM) (or Asynchronous Time Division Multiplexing as it was originally called [Fraser93]), is a compromise technology which aims to combine the capabilities of packet switching and synchronous Time Division Multiplexing. ATM can simultaneously support voice, video and data communication over a variety of transmission speeds. ATM traffic is carried in fixed size packets called *cells*. The size of an ATM cell has been standardised by the ITU as 53 bytes, of which 5 bytes form the cell header and the remainder constitute the payload.

ATM provides a data transfer capability whereby cells are transported through a network by means of a *virtual connection* (VC) which is implemented by maintaining state in switching elements along the way. In particular, an ATM cell is switched from an input port to one or more output ports within a network element (or switch) based on the value of a small identifier which forms part of each cell header. This identifier has local significance only, and is replaced (fully or in part) when the cell passes through a switch. The identifier is made up of two parts, namely a *virtual path identifier* (VPI) and a *virtual channel identifier* (VCI). The concatenation of VPI and/or VCI pairs used on consecutive links effectively forms the VC, and such a VC has to be set up, i.e. state in switching elements has to be established, before data transmission can commence. Switching of cells at network nodes can be based on the VPI value only, on the VCI value only or on both VPI and VCI values. If switching is done based only on the VPI value of the header, the resulting VC is a *virtual path connection* (VPC) and the VCI values of cells are carried through without modification. Concatenated VCI values in a VC constitute a *virtual channel connection* (VCC). A VPC therefore transparently carries a number of VCCs.

It is possible to dedicate a proportion of the switch resources to a specific connection. These resources include buffer space and link bandwidth, and can be used by the switch to provide a certain *quality of service* (QoS) to the cells transmitted across the VC. Connections in an ATM network can either be created semi-permanently or on-demand. These are respectively called permanent virtual connections (PVCs) and switched virtual connections (SVCs). The process of dynamically setting up the VPI/VCI tables and associating resources with a VC is normally called *signalling*, or *out-of-band control*, and networks that have separate connection establishment and data transfer phases are called *connection oriented*. Signalling is considered "out-of-band" because control messages are sent on a VC separate from data connections. An important function of out-of-band control is to determine whether a newly offered connection can be accepted without violating the QoS guarantees of existing connections. This process is called *connection acceptance control* or *connection admission control* (CAC)[1]. In order to be able to make QoS guarantees to several connections, the switch has to ensure that a particular connection does not use more resources than were allocated to it. This process is normally called *policing* or *usage parameter control* (UPC).

The fixed size of ATM cells significantly simplifies the implementation of

[1]Also sometimes called *call* admission control.

switch hardware. However, the 48 byte payload offered by ATM cells is not a natural fit for any of the traffic that ATM was designed to transport (i.e. voice, video and data)[2]. At the entry point to the network this requires the breaking up of offered data into 48 byte units. Similarly, on arrival at the final destination the payloads of the ATM cells need to be converted again to the "natural" format of the data. These processes are called *segmentation* and *reassembly* respectively. The natural format of offered data cannot normally be segmented directly into ATM cells and an ATM adaptation layer (AAL) is required in endpoints above the basic data transfer service of ATM. Based on different traffic classes several AAL types have been defined. Of these AAL5 is predominantly used to transport computer (and other) data. AAL5 encapsulates variable length packets in a data unit which can be segmented into ATM cells[3].

Conventional out-of-band control in ATM, as defined by the ATM Forum, is divided into User-Network-Interface (UNI) signalling, and Network-Network-Interface (NNI) signalling. By convention, the signalling protocol uses a predefined VC and consists of a large number of well defined messages. Signalling across the UNI allows the network to assign a unique network address to an endpoint, and allows the endpoint to request creation of connections of various types and with different QoS guarantees [ATM Forum94b]. Software signalling entities on both sides of the UNI maintain state during the signalling process to allow, amongst other things, recovery from failure. NNI signalling takes place in a similar manner and has additional functionality to exchange and maintain routing information [ATM Forum96].

The data transfer capabilities of ATM have been standardised by the ATM Forum in the form of the five *service categories* [Sathaye95]. It is assumed that the characteristics of sources can be captured by a set of *traffic parameters*. Traffic parameters include the Peak Cell Rate (PCR), Sustainable Cell Rate (SCR), Maximum Burst Size (MBS) and Minimum Cell Rate (MCR) of the traffic source. The traffic parameters are then used to negotiate, by means of signalling, a *traffic contract* with the network. The network on its part is expected to provide certain Quality of Service (QoS) guarantees to connections which comply with their traffic contract. QoS guarantees include the Cell Delay Variation (CDV), the maximum

[2] The 48 byte payload was chosen as a compromise between 32 and 64 bytes. European delegates at the ITU favoured a 32 byte payload, while the US and Japan wanted 64 bytes [de Prycker91].

[3] AAL3/4 was originally proposed as the means to transport data of this type. Although some implementations still exist, AAL3/4 has largely been superseded by the more efficient AAL5.

and mean Cell Transfer Delay (CTD), and the Cell Loss Ratio (CLR) experienced by connections. The five service categories are:

- Constant Bit Rate (CBR): CBR services are intended for applications which require tight constraints on the delay and delay variation experienced by cells travelling through the network. A fixed amount of bandwidth is expected to be consistently available to such connections.
- Real-Time Variable Bit Rate (rt-VBR): rt-VBR is meant to provide a constrained delay and delay variation service to *bursty* sources, or sources which transmit at variable rates.
- Non-Real-Time Variable Bit Rate (nrt-VBR): nrt-VBR provides a service for bursty applications which only expect a bound on the *mean* CTD.
- Unspecified Bit Rate (UBR): UBR provides transfer services for applications which require no tight delay requirements, and which can be bursty. No per-connection bandwidth reservation or CLR guarantees are provided for UBR connections.
- Available Bit Rate (ABR): ABR service attempts to provide a fair share of available bandwidth to connections. The network provides a source with feedback about the current bandwidth available to that source, and the source is expected to adjust its transmission rate accordingly. The standardised feedback mechanism for ABR services is *rate* based. Special Resource Management (RM) cells are sent to a connection source to regulate its current rate of transmission.

The contribution of the Computer Laboratory to the area of ATM includes early fixed size cell networks including the Cambridge Ring [Hopper78], the Cambridge Fast Ring [Hopper88] and the Cambridge Backbone Network [Greaves90], and point-to-point switches in the Cambridge Fast Packet Switch [Newman89] and Fairisle [Leslie91]. An ATM based Multi Service Network architecture was presented in [McAuley90], and [Crosby95a] was the first to make use of distributed processing methodologies in the ATM control plane, as well as doing extensive traffic analyses on real ATM data. It was also at the Computer Laboratory that the data transfer capabilities of ATM were first used to replace a workstation bus in the Desk Area Network [Hayter93].

2.1.2 Distributed Processing Environments

Early Distributed Processing Environments (DPEs) include the Cambridge Distributed Computing System [Needham82], which later matured into the ANSA architecture [Herbert94a]. More recently the advent of object-oriented technology progressed these approaches into systems like the Common Object Request Broker Architecture (CORBA) [Vinoski97]. This development, and especially the CORBA architecture is considered below.

The Cambridge Distributed Computing System system provided shared access to devices such as printers and fileservers. The environment also included a set of computers called the *processor bank*. A user could request access to one of these machines when the task at hand was too demanding for his or her personal machine. If an appropriate machine from the processor bank was free, it would be allocated for the requesting user's exclusive use.

The ANSA architecture[4] [Herbert94a] and its ANSAware implementations [APM92] can probably claim significant influence on the shape of modern DPEs. ANSA is an architecture which enables telecommunications services and computer applications to work together despite diversity of programming languages, operating systems, computer hardware, networks and communications protocols. The ANSA architecture specifies the following [Herbert94a]:

- A set of components, which form the building blocks and tools of the architecture.
- A set of rules, which constrain how such components can be combined in implementations.
- A set of recipes for how to combine components into subsystems with certain properties.
- A set of guidelines to help designers make the right design decisions.

Using this architecture, i.e. the components, rules, recipes and guidelines, improves and eases portability and interoperability between applications despite the differences between systems. Indeed, probably the most important aim of

[4] Originally ANSA was an acronym for Advanced Networked Systems Architecture, but more recently the word appears to have taken on a meaning of its own, normally in the context of a DPE.

a DPE is to hide this *heterogeneity* which is prevalent in modern network and computer systems [Vinoski97]. This is achieved by raising the level of abstraction. While increased abstraction leads to a simplified system, it invariably leads to a loss of fine-grained control.

The CORBA architecture [OMG95b] from the Object Management Group (OMG) appears to be the most promising DPE at the time of writing[5]. CORBA, in fact, is only the communications part of a more encompassing architecture called the Object Management Architecture (OMA) [OMG97].

Fundamental to the OMA is the notion of an *object*, which is "an encapsulated entity with distinct immutable identity whose services can be accessed only through well-defined *interfaces*" [Vinoski97]. These interfaces are used by clients to request objects to perform services on their behalf. Interfaces are defined in a neutral *Interface Definition Language* (IDL), which, in the case of CORBA, is similar to C++.

The exact location and implementation of an object is hidden from the requesting client, and communication between client and object is handled by an Object Request Broker (ORB). (An ORB is an implementation of the Common Object Request Broker Architecture.) When an object is created, an *object reference* is also created. This object reference refers only to the newly created object, and is used by a client to request services from a specific object. A client obtains the object reference of a specific service by first using the services of a well known object such as a Naming or Trading service (see below).

To facilitate interworking between ORBs from different vendors, the OMG has defined a General Inter-ORB Protocol (GIOP). GIOP specifies the message formats for interaction over any connection-oriented protocol. The Internet Inter-ORB Protocol (IIOP) is a specialisation of GIOP for use over TCP/IP (Transmission Control Protocol/Internet Protocol) networks. Because of the ubiquitous nature of the TCP/IP protocol suite, support for IIOP is mandatory for a CORBA 2.0 ORB [OMG95b]. While IIOP is an inter-ORB protocol, many ORB implementations also use it internally as an intra-ORB protocol.

In addition to CORBA the OMA defines:

- Object Services, which are domain-independent services used by many applications. Examples include:

[5] Because they combine object oriented design and programming with distributed systems, modern DPEs are often grouped under the name Distributed Object Computing (DOC).

  - A Naming Service, which allows clients to find objects based on the name of the object.
  - A Trading Service, which allows clients to find objects based on the services they provide, and the properties of those services.

- Common Facilities, which are also services used by many applications, but aimed at end-user applications. An example would be database facilities.

- Domain Interfaces, which are services oriented to specific application domains such as telecommunications, manufacturing and finance.

- Application Interfaces, which are interfaces developed for a specific application. As such these interfaces are not standardised.

[Herbert94a] observes that many software systems are inherently distributed regardless of whether they are designed or managed as such. This obviously holds true for the management and control of networks including ATM networks. In confirmation of this observation, a large body of work which utilises the services of a DPE to manage and control networks has recently appeared [Nilson95, Lazar95, Bosco96, Rooney97a, Davison97]. Indeed some of the work presented in this dissertation benefited from the services of a DPE. An overview of the referenced DPE based control and management work is postponed until the related work discussion in Chapter 7.

2.1.3 A Definition of Communication Services

In the introduction, and indeed in the literature in general, the term *"service"* is often used in an ambiguous manner. Since services are a focal point of the OSSA, this section aims to clarify the meaning of the word within the context of this dissertation.

The ITU defines two *basic* services, namely bearer services and tele-services [ITU-T93a]: Bearer services provide a low level information transfer capability between network access points. Tele-services use one or more bearer services to provide the "full capacity for communication". Telephony is an example of a tele-service which can use a variety of bearer services.

Broadband services are similarly divided into broadband interactive and broadband distribution services, and these may be offered as either bearer services or tele-services depending on their function [ITU-T93b]. The ITU classification is made from the point of view of the network, and not from the point of view of the user.

In the Telecommunications Information Networking Architecture (TINA) a service is (informally) defined as [Minetti96]: " ... a meaningful set of capabilities provided by an existing or intended network to all who utilise it, like customers, end users, network providers and service providers. Each one sees a different perspective of the service". This definition has the advantage over the ITU's definition of a service, in that it recognises the existence of different perspectives on the same service.

The OSSA is more narrowly concerned with services involved with ATM networks. Furthermore, the OSSA enables an environment in which the conventional clear separation between providers and users of services will no longer hold. As such, the OSSA definition of services is closer to that of the ITU: The OSSA is mainly concerned with *network services* which are in turn divided into *transport, inherent* and *derived services.* These can be informally described as follows:

- Network transport services: these include QoS guaranteed and multicast transport services. (Indeed the "service" in QoS is understood to refer to transport services.)
- Network inherent services: these include the signalling, management, routing and naming services.
- Network derived services: these include video conferencing, file transfer and video broadcasting services, and are based upon the services provided by the transport and inherent services[6].

Based on the above breakdown of services, an Open Service Support Architecture can be defined as an environment in which *any* of these services could be easily provided, modified or replaced, by *any* authorised network user. At the same time, such actions should not adversely affect other users of the network.

[6] In the private network environment, derived services might simply be applications. Indeed, the distinction between services and applications is not always clear. For the purposes of this dissertation a service will be considered the set of "generic" mechanisms provided, while an application is something that is used by a user. Thus video conferencing *services* will include things like group and membership control and synchronisation, while a tele-teaching *application* uses these services in a structured way.

Dynamically changing network transport services, or the transport capabilities of nodes, will undoubtedly be very attractive for owners of network equipment. For example a new buffering and scheduling mechanism can be instantiated in a switch without the switch being removed from the physical network[7].

However, changing the transport services of a network element will clearly have an influence on *all* users of that element, and these services can therefore not be opened up in the same way as inherent and derived services. As such this dissertation will be largely concerned with inherent and derived services. As indicated in Chapter 1, these services are collectively called a control architecture. Whilst in the first instance this would appear to be applicable only to inherent services, it will be shown in Chapter 6 that within the context of the OSSA, derived services (or parts thereof) are also considered to be part of a control architecture.

2.2 Research Environment

The ATM environment at the Computer Laboratory consists of a Digital GigaSwitch, a number of Fore Systems switches, a number of ATML Virata switches and several locally developed Fairisle switches [Leslie91]. These switches interconnect an assortment of ATM capable workstations, a router and other ATM end devices such as ATM video adapters (the Fore Systems AVA-200). In addition, these switches provide connectivity to several other ATM networks/subnetworks including the SuperJanet ATM network and a European Public Network Operator (PNO) ATM trial.

With the exception of the Fairisle switches, this ATM environment forms part of the Computer Laboratory's service network. Despite this fact, it was decided to use the three Fore Systems switches (one ASX-100 and two ASX-200s) for experimentation mainly for the following reasons:

- The majority of ATM end devices in the Computer Laboratory are connected to the Fore Systems switches, which allowed for a useful experimental environment.

[7]Proposals have been made to perform on-the-fly modification of the data handling in networks that do not provide any QoS guarantees. Some of these proposals will be considered in Chapter 7.

- Low level programming information for the Fore Systems ASX-100 switch is publicly available.

- Fore Systems switches are commonly used by groups doing ATM research, which means that it will be easier to make the work presented here available to a wider audience.

In some of the work a distributed processing environment was used. An early implementation of the Distributed Interactive Multimedia Architecture (DIMMA) [Li95] was used for this. DIMMA is a framework ORB, which provides a common base for the construction of domain specific ORBs.

The following sections consider the Fore Switches, the workstations, the Fore AVAs and DIMMA in more detail.

2.2.1 Fore Systems ASX-100 Switch

The ASX-100 is an output buffered local area ATM switch which provides 2.5 Gbps aggregate bandwidth, and can support up to 16 ports with a variety of physical interfaces [Biagioni93]. The switch fabric consists of a time multiplexed bus which interconnects four network modules. Depending on the link speed each network module can have between one and four ports.

The bus-based switch fabric means that multicast can be supported very efficiently. However, because VPI/VCI translation is done before the cell enters the switch fabric, all output branches of a multicast connection must have the same header (i.e. the same VPI/VCI value)[8].

The switch can support two priorities of output buffers, however this facility was not used in this work, and cells were simply stored in first-in-first-out (FIFO) fashion in one output buffer. The ASX-100 can perform both VP and VC switching. In order to facilitate the latter, two sets of translation tables (for VPI and VCI translation respectively) are used. The VPI of the incoming cell is used as offset into the VPI table which means that the VPI table should have enough entries for all possible VPI values. A limited range of VCIs is supported, which limits the required size of the VCI table.

[8]This restriction has been remedied in later models of Fore Systems switches.

The ASX-100 contains a RISC-based control processor running the UNIX operating system. The control processor has an interface to the VPI/VCI translation tables to effect the creation of connections through the switch. In addition, a special control port on the switch fabric allows the control processor to send and receive cells. With appropriate software running on the control processor, this means that the control processor can be considered an ATM capable workstation on the ATM network. The throughput to the control processor is limited however, because no hardware support (e.g. cyclic redundancy check (CRC) calculation) is provided for the transmission of AAL5 frames. Some hardware support for AAL3/4 is provided.

Programming details of the interface between the control processor and the translation tables are available in the public domain through the VINCE source code [VINCE]. This, together with the fact that the control processor runs a standard operating system, allowed a considerable amount of experimentation to be performed on the ASX-100.

2.2.2 ATM Capable Workstations

Several workstations with ATM adapters were connected to the ATM network described above. A subset of these provided access to a native ATM stack and were therefore used in experimentation. This subset of workstations included HP 700 series, and Sun Ultra workstations, all of which were equipped with Fore Systems 200 series ATM adapters [FORE94].

The 200 series ATM adapter has an Intel 80960CA ("i960") processor onboard in addition to hardware CRC support for AAL3/4 and AAL5. The adapter is capable of performing DMA (direct memory access) transfers between the host memory and the adapter, and provides a packet interface to the host. Segmentation of a packet provided by the host is performed on the adapter while the packet is being transfered by DMA from host memory to the adapter. Similarly, reassembly of incoming cells is done by the adapter in buffers on the host system. The host is only interrupted when a complete packet has arrived.

Software on the host system provides a complete set of ATM data transfer facilities. These include an IP-over-ATM implementation as well as a native ATM stack. The latter provides an application programmer's interface (API) which allows sending and receiving user data using either AAL3/4 or AAL5. Signalling for the native ATM stack is provided by means of a proprietary signalling protocol called SPANS [FORE95b], as well as an implementation of the ATM Forum's UNI signalling specification [ATM Forum93, ATM Forum94a].

Of greater interest for the work presented here, however, is the fact that the native ATM API can also be used without having to use SPANS/UNI signalling. The ATM protocol stack could therefore be used together with the various control entities described in this dissertation.

2.2.3 ATM Video Adapter

The ATM Video Adapter (AVA) started life as a node on the desk area network [Pratt94, Barham95], and has since been commercialized by Fore Systems. The AVA-200, as it is now called, captures and digitises real-time audio and video, and segments these into ATM cells which can be sent across an ATM network. The AVA-200 has three video and three audio inputs and at any time data from one of each can be captured for transmission across the network.

The AVA-200 is not capable of signalling; instead, proxy signalling is performed on its behalf by a manager process running on a host connected to the same ATM network. A PVC between the manager host and the AVA-200 allows the manager to send proprietary control messages to the AVA-200. A microcontroller on the AVA-200 receives and processes these messages, and performs appropriate control functions on the video and audio coder hardware.

The video capture hardware on the AVA-200 can produce digitised video in a variety of formats including 8-bit monochrome, 16- and 24-bit colour and JPEG compressed[9]. Frame rates are fully configurable from zero to thirty frames per second. Similarly the HiFi audio codec (coder-decoder) on the AVA-200 can support a variety of encodings including 8-bit Pulse Code Modulation (PCM) in both mono and stereo, and mono and stereo 16-bit PCM encoding. Sampling rates are configurable from telephony quality (8 KHz) up to Compact Disc (CD) (44.1 KHz) and Digital Audio Tape (DAT) (48 KHz) quality.

2.2.4 DIMMA

The Distributed Interactive Multimedia Architecture (DIMMA) is a distributed processing framework which facilitates service management, service binding and

[9]JPEG is a still image compression standard from the Joint Photographic Experts Group.

service QoS management [Li95]. In recognition of the diverse requirements of applications using ORB technology, the approach taken in the DIMMA project was to structure an ORB as a modular set of components that plug into a minimal ORB framework or microORB. Rather than trying to strip down a monolithic ORB for a specific application or market, a "custom" ORB can therefore be assembled from standard replaceable components. Standard ORB interfaces such as CORBA [OMG95b] and TINA [Kelly95] are catered for in the DIMMA framework by means of "personality modules" on the DIMMA nucleus.

The DIMMA framework aims to provide generic abstractions for:

- communications policies and protocols,
- threading,
- buffer management and
- event processing.

In addition implementations of the architecture can provide abstractions for streams [Otway95b] and explicit binding [Otway95a].

The DIMMA implementation used for experimental work allowed several protocol stacks to be operational simultaneously. In addition to the CORBA IIOP protocol, the implementation provided an experimental ANSA remote procedure call (ARPC) implementation. The ARPC protocol can utilise both User Datagram Protocol (UDP) and Transmission Control Protocol (TCP) as the underlying protocol.

Despite deploying a CORBA "personality", the terminology used within DIMMA is still "generic DPE" rather than CORBA specific. For example *server* and *interface reference*, are used in favour of the more correct *object* and *object reference* respectively. To avoid confusion this DIMMA terminology will be used in the rest of this dissertation.

2.3 Research Context

The research presented in this dissertation was conducted within the Systems Research Group (SRG) at the Computer Laboratory. Three research projects undertaken within the SRG are briefly considered in this section. These projects either influenced, or were influenced by, the work presented here.

2.3.1 Pegasus

Pegasus, as is its successor Pegasus II, is a multi-party project tasked with: "Operating system support for distributed interactive multimedia" [Peg97]. Work conducted at the Computer Laboratory is centred around the development of the Nemesis operating system [Leslie96].

The Nemesis operating system is an entirely new operating system which aims to provide QoS guarantees to applications in the same way that ATM networks can provide QoS guarantees to connections. Nemesis is a single-address-space operating system and runs on several platforms including Intel, Alpha and StrongARM. Nemesis has an extremely small kernel and performs many traditional operating system functions in user domains outside the kernel.

Providing QoS guarantees to applications effectively means that an application receives a subset of the workstation's physical resources, and can use these as it sees fit. It will be shown in Chapter 3 how this is analogous to the switchlet concept in the OSSA.

Other aspects of the Nemesis operating system will be considered in the related work chapter, Chapter 7.

2.3.2 Measure

The Measure project examines the application of the technique of online measurement to the problem of resource management in networks and operating systems [Mea97].

Other approaches to resource management often assume that the behaviour of users is known, or can be modelled. These approaches are flawed because of the poor understanding of, and the rapidly changing nature of applications, especially in a multi-service environment. The use of online measurement techniques promises to simplify resource management while at the same time making the process more accurate. The "Measure-approach" is based on Large Deviations Theory.

In Chapter 5 online measurement techniques, based on the work done in the Measure project, are used to perform bandwidth management in the OSSA. The online measure approach is explained in more detail in that chapter.

2.3.3 DCAN

DCAN is an acronym for Devolved Control of ATM Networks, and the project has as its aim the design and implementation of a scalable control and management system for ATM networks [DCA97]. As indicated earlier, such a system is called a control architecture within the context of the OSSA. Indeed the original DCAN proposal [Herbert94b] served as motivation for the need for an OSSA, because this was yet *another* control architecture for ATM networks.

From its inception the DCAN approach was concerned with the use of a DPE to built the envisaged control architecture. As such the original emphasis of the project was on the requirements of a DPE that could be used in this environment. Some exploratory work was done involving an implementation of ANSAware [APM92] over the Nemesis operating system.

Another explicit aim of the DCAN project was the control of simple networked devices, such as the AVA video adapter described in Section 2.2, as well as more sophisticated devices such as workstations. This meant that the control architecture would be scalable in terms of both resource requirements and the size of the network involved.

Insofar as the work presented in this dissertation is related to the DCAN project, this research has benefited from it in the form of access to equipment and other resources. Similarly the DCAN project has benefited from the work presented here, and that to be presented in [Rooney97c], inasmuch as its current direction has been largely dictated by these approaches.

2.4 Summary

In this chapter background information essential to the OSSA was introduced. This provides the necessary platform for the remainder of the dissertation in which the OSSA and its uses are explored.

Chapter 3

An Open Service Support Architecture

3.1 Introduction

ATM is one of the networking technologies which promise the realisation of a truly multi-service networking environment. The likelihood of ATM achieving this goal is critically dependent on its ability to:

- Provide data transfer capabilities consistent with the expectations of a wide range of applications and services.
- Provide the necessary control framework in which these services can be created and deployed with ease.

There is wide acceptance of the superior data transfer capability of ATM, however, the inadequacy of existing ATM control and management strategies [ATM Forum93, ATM Forum96] to meet the demands of, for example, multimedia applications, has been widely acknowledged. Several alternatives have been proposed over an extended period of time [Minzer91, Bubenik91, Miller92, Olsen92, Doeringer93, Lazar95, Bale95, Cidon95, Iwata95, Crosby95a, Veeraraghavan95, Davie96, Veeraraghavan97, Hjalmtysson97, Newman97b][1]. As

[1]Indeed such concerns recently led to the establishment of OPENSIG as a forum to "Explore issues in network programmability and next generation open signalling technology".

explained in Chapter 1, the mechanisms that constitute a particular control and management approach are collectively called a *control architecture*[2].

Despite their obvious advantages, the uptake by standards bodies of ideas from these new control architectures has been slow, or nonexistent. One of the main reasons for this situation is that a new control architecture is normally proposed as a *replacement* of an existing one. Naturally, this leads to a reluctance to move to the new untested control architecture, even if the existing one is known to be less than ideal.

Some new control architectures are simply extensions of the "conventional" ATM out-of-band control as presented in Section 2.1.1. For example [Cidon95] propose modifications to the standard Private Network-Network Interface (PNNI) specification [ATM Forum96], mainly involved with improving the efficiency of connection setup. Similarly in [Iwata95], ATM signalling and addressing is extended to avoid the router bottleneck in the classic IP-over-ATM approach [Laubach94].

More radical are approaches in which *only* the data transfer capability of "standard" ATM is used, and a completely different control architecture is employed in the network. One example in this category is IP switching in which a normal ATM switch is effectively used as the "optimised forwarding engine" in an IP router [Newman97b]. IP packets are either routed normally through a software router, or forwarded directly through the ATM switch. The decision of whether to route or to switch is taken by the control architecture and is based on traffic patterns and the installed policy in the device [Newman96b].

These different approaches seriously question the basic assumption that there is a single control architecture that will provide for the needs of all applications and services. Furthermore, since different control architectures are built on different models and provide different functionality, it is reasonable to expect to be able have the functionality of more than one control architecture in the same physical network.

The Open Service Support Architecture (OSSA) presented in this chapter addresses this expectation by providing an environment in which several control architectures can coexist on the same ATM switch and network.

[2]For example, a UNI/NNI signalling implementation is (an example of) a control architecture, as is the IP switching architecture from Ipsilon [Ipsilon96].

An essential requirement for the OSSA is an open switch control interface. This is the interface used by a control architecture to manipulate the switch hardware in order to perform its control and management functions. An example of an operation performed through this interface is the manipulation of bits in a forwarding table in order to set up a forwarding path through the switch. The proliferation of control architectures means that a definite requirement for this interface is that it be *open*, so that different control architectures can be developed to make use of it.

The OSSA approach to open switch control is presented in Section 3.2 and involves a simple low level interface which can be used by any control architecture to exercise control of the physical switch. A switch exporting this low level interface still has the limitation that only one control architecture can be operational at any particular moment in time.

Section 3.3 addresses this problem, and the key requirement of the OSSA, by showing how a subset of the ATM switch resources can be presented to a particular control architecture as a *switchlet*. The term switchlet is used in favour of, say, "virtual" switch, to emphasise the fact that real resources are allocated to the switchlet. A switchlet presents the same open switch control interface to its control architecture, which means that the latter is oblivious of the fact that it is not in control of the whole physical switch. The partitioning of a switch into switchlets means that several control architectures can be simultaneously and seamlessly operational on the same physical switch.

Switchlets can be combined into virtual ATM networks, each of which can potentially use a different control architecture. In this way, the switchlet concept allows several control architectures to operate in the same physical network. These virtual networks can have any arbitrary topology within the constraints of the topology of the physical network.

In the first instance, the action of creating switchlets and combining them into virtual networks may be performed by human operators. The process can, however, be automated so that virtual networks can be dynamically created. The OSSA components involved with this process are considered in Section 3.4. The control architecture instantiated on the newly created virtual network can be one of a predefined set of well-known control architectures. Alternatively, an "empty" virtual network can be created in which the control architecture is provided by the "user", or the entity that requested the creation of the virtual network. In the latter case the mechanisms implementing the control architecture should be provided by the entity requesting network creation.

The elements of the OSSA mentioned above, and discussed in detail in the remainder of this chapter, therefore facilitate an environment in which "any user" can construct a network and effectively become a "network service provider" [van der Merwe95].

3.2 OSSA Approach to Open Switch Control

Figure 3.1 illustrates the control of an ATM switch by software forming part of a particular control architecture, through an Ariel *open switch control interface.* In the OSSA approach switch control is opened up by providing a simple, generic, low level switch control interface on the switch. Switch control software running on a general purpose computing environment invokes operations on the Ariel interface in order to control and manage the switch.

The switch control software and the Ariel interface have a client-server relationship. The relationship is highly asymmetric because the server is always very simple and lightweight, whereas the client (the switch control software or control architecture) is potentially very complex. For this reason, the control software is assumed to run in a general purpose environment, while the Ariel server is simple enough that it can be implemented on very simple switches. The relationship is also one-to-one or one-to-many, but not many-to-one: A single switch controller can communicate with one or more Ariel interfaces, but an Ariel interface can only have a single control entity associated with it. Note that whether the control processor is part of the switch, or physically removed from the switch is not really important and can be considered an implementation choice. What is important, however, is that some form of open control interface is exported by the switch.

The switch control software is responsible for performing all the functions required by a specific control architecture, such as setting up virtual channel identifier/virtual path identifier (VCI/VPI) mappings, admission control, resource allocation, routing etc. A control architecture is not limited to the implementation of a single signalling protocol. For example, an implementation capable of both UNI 3.0 [ATM Forum93], and SPANS [FORE95b] signalling could still be a single control architecture.

Even though communication between the control software and the Ariel interface is based on client-server principles, this does not require the facilities of a general purpose distributed processing environment (DPE). Rather, communication between the control client and Ariel server is implementation specific, for example based on message passing on a default VCI/VPI pair. The Ariel interface specifies the *functionality* required by an open switch control interface, and implementations based on different mechanisms can be carried out. Chapter 4 presents several Ariel implementations.

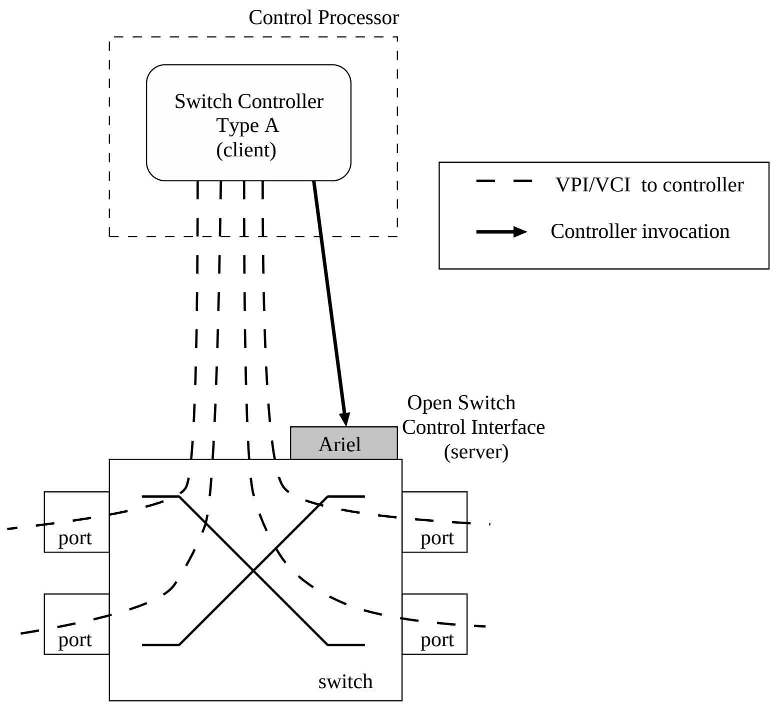

Figure 3.1: Switch control through the Ariel interface

3.2.1 The Ariel Control Interface

Ariel aims to provide an open, generic switch control interface. In particular the Ariel interface should be useful even if detailed knowledge of the switch being controlled via the interface is not available. Ariel should provide sensible control of all switches, even switches of unknown type.

Ariel consists of the following interfaces[3]:

[3]As shown here, Ariel is really made up of a set of *several* interfaces. Most of the time this set of interfaces will simply be called the Ariel control *interface*, as if it were a singular entity. At other times, each member of the set will be considered separately. A different font is used

- `Configuration` - The `Configuration` interface is primarily used to *find out* what the configuration of a switch is. It contains methods which allow the controller to obtain the capabilities of the switch in a per port list. The capabilities include the VPI and VCI ranges and the ATM Forum service categories supported. The switch vendor name and type can also be obtained, which allow the control architecture to perform switch specific optimisations.

- `Port` - A `Port` interface is provided for each port on the switch and deals with a port as a complete entity. For example, methods are provided which allow the controller to take a port out of service, or to put it in loopback mode.

- `Context` - The `Context` interface bundles QoS abstractions into a single interface. QoS aware connections are set up through Ariel by first creating a context, and then associating that context with a VPI/VCI mapping during the actual connection setup. Alternatively, a connection which does not require QoS guarantees, or simply wants a best effort guarantee, need not use the `Context` interface at all. This approach also means that all QoS issues are taken out of the `Connections` interface, which can be kept very simple.

  Dealing with QoS issues in a generic fashion at a low level is very difficult, and may not be possible. The reason for this is that the QoS *capabilities* of a switch are determined by the buffer allocation and cell scheduling policies employed. These, in turn, are what differentiate one switch from another. It is therefore unlikely that all switch vendors would be willing to make such details about their switches available to be included in an *open* interface, such as Ariel.

  One way of avoiding this problem is to hide the switch queueing and scheduling policies behind a generic interface. The ATM service categories presented in Section 2.1.1 provide the means for such an abstraction. This approach is reasonable because it can be expected that switch manufacturers will build switches with queueing and scheduling mechanisms which will support a subset of these services. The Ariel `Context` interface follows this approach[4].

for the names of the set of interfaces to highlight this difference.

[4]The ATM Forum service categories are naturally very ATM centric. It is believed that it will be possible to map simpler service categories, such as those proposed by the Internet community [Braden94], onto the ATM Forum categories. Within the Internet community

The ATM Forum service categories are parametrised by (in total) four QoS parameters and six traffic parameters, a subset of which must be specified for each service category. The `Context` interface provides methods which use these parameters to reserve and release resources on the switch at both VP and VC level. A method also provides a list of all the currently specified contexts on the switch.

- `Connections` - The `Connections` interface is responsible for basic VPI/VCI mapping, and deals with QoS issues through a context index obtained (by the controller) through the `Context` interface. Methods are provided to create and delete connections at VP and VC level, as well as to obtain lists of all current connections.

- `Statistics` - The `Statistics` interface allows the controller to obtain switch statistics and accounting information. This includes per port, per VP and per VC statistics about the number of received, transmitted, dropped and erroneous cells.

- `Alarms` - The `Alarms` interface allows the controller to be informed when certain events take place on the switch. For example the controller might be informed about port state changes such as loss of carrier, a port being removed or added etc.

Separating connection setup into two interfaces, for VP/VC and QoS setup respectively, was also used in [Crosby95a] for control of the Fairisle switch [Leslie91]. More recently a similar separation was proposed in the UNITE signalling architecture [Hjalmtysson97]. In UNITE the motivation for separation is to expedite the processing required for best effort connection setup, which does not require any QoS guarantees. The same motivation led to the separation of functionality in the Ariel interface.

Another reason for separating the `Connections` and `Context` interfaces comes from the realization that the `Context` interface provides a way to "allocate resources" on the switch, whereas the `Connections` interface provides a way to "use the resources", and that the two need not be tightly coupled. For example, a control architecture can allocate a certain amount of resources on the switch by means of the `Context` interface, and then "over commit" these resources by

some work has been done in this regard, although it assumes a conventional standard ATM environment [Garret97].

allowing more connections to be created than the resources would suggest, because it has some additional knowledge about the behaviour of the connections in question[5]. An example would be a video conference session, where the dynamics of human interaction allow some resources to be shared. This example is explored in detail in Chapter 6.

Hiding the switch details behind the `Context` QoS abstraction obviously leads to a loss of information. However, enough information is still available to perform functions such as connection acceptance control (CAC) outside the switch in the external controller.

CAC deals with the question of whether a new incoming connection request can be accepted, i.e. if its QoS requirements can be met, without adversely affecting the QoS guarantees of existing connections. This means that the CAC functions should know:

- What physical resources are available on the switch.
- What proportion of those resources is used by the current connections. (Or conversely how much of the resources is still available.)
- How much resource the new connection will require.

In order to perform CAC outside the switch, a *resource mapping function* that is used by the switch to map QoS and traffic parameters to switch resources is required, as well as knowledge about the resources available on the switch. Such a resource mapping function constitutes significantly less sensitive information than the mechanisms used to implement it. It can therefore be expected that switch manufacturers will be more willing to provide such information.

One generic resource mapping function is *effective bandwidth* [Hui88, Kelly96], a concept which aims to provide a measure of the resource usage of different sources with differing traffic characteristics and QoS requirements. Effective bandwidth expresses the bandwidth requirements of a traffic stream which lie somewhere between its peak and average rates. In its original form, effective bandwidth CAC relied on the declared traffic descriptors of connections, and on models of the behaviour of different types of sources when multiplexed together. Recent evidence [Crosby95b] suggests that effective bandwidth requirements can

[5]Most current commercial ATM switches will not allow the separation between the allocation and usage of resources, or at least not to this extent.

be more accurately estimated using *online measurements*, and the measurement approach has the additional advantage that it is not dependent on predetermined traffic types.

Because of its inherent abstraction of switch resources, the effective bandwidth approach is particularly attractive for an open control interface such as Ariel. Furthermore, there seems to be growing interest in the use of measurement based, rather than model based techniques for CAC [Lewis97, Grossglauser97b]. For these reasons, the resource management problem is revisited in Chapter 5, in which measurement based effective bandwidth estimation is applied to bandwidth management within the OSSA.

3.3 Switchlets

The approach described in Section 3.2 enables a switch to be controlled in an open fashion and allows different control architectures to be designed to utilise the Ariel interface. This brings much more openness and flexibility than is currently the case on ATM switches, but it still means that at any given time, a single control architecture is operational on a switch and within a network.

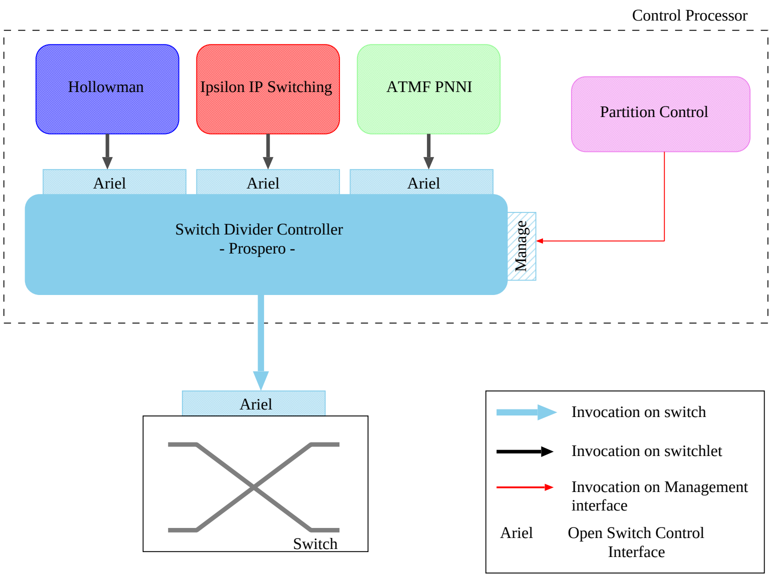

Figure 3.2: Creating switchlets

Figure 3.2 shows how the Ariel interface on a physical switch can be used by a **Switch Divider Controller** to create several *switchlets*[6]. The Switch Divider Controller is called **Prospero**. Prospero allocates a subset of the physical switch resources to a switchlet, and makes this available to the switch control software through yet another Ariel interface. Switchlet resources include a subset of the switch ports, VPI/VCI space, bandwidth and buffer space. More than one switchlet can have the same switch port, but all switchlets need not have the same set of ports. Switch control software, of a particular type, will control the switchlet by invocations on the switchlet Ariel interface, in exactly the same way as it would control a physical switch with just a single Ariel interface. As an example, Figure 3.2 shows three possible control architectures, namely the Hollowman experimental control architecture [Rooney97a], and the IP Switching [Ipsilon96] and ATM Forum's PNNI [ATM Forum96] control architectures. These control architectures are discussed in detail in Chapter 7. Figure 3.2 also shows that partitioning of the switch into switchlets is controlled through a separate Management interface on the Divider Controller which is described later.

Switchlets can be combined to form virtual networks of a certain *type*, where the type of a virtual network is defined by the operational control architecture on that particular set of switchlets. This is depicted in Figure 3.3, which shows a network of five physical switches, on which three virtual networks of different types are deployed.

The switch is completely oblivious to the fact that several control architectures are operational on it, and the one-to-one relationship with the Ariel server on the switch is maintained by a single Divider Controller. Prospero provides Ariel interfaces to the switchlet controllers. Except for the fact that fewer resources are available to it, a switchlet controller is therefore also unaware of the presence of the Divider Controller. The Switch Divider Controller polices invocations on the switchlet Ariel interfaces to ensure that switchlet controllers do not utilise resources not allocated to them, or in any other way interfere with the functioning of other switchlets[7].

[6] A patent application has been filed to protect the technique of partitioning a switch into switchlets [van der Merwe96].

[7] The Switch Divider Controller is analogous to the small kernel in the Nemesis operating system [Leslie96], which is responsible (amongst other things) for allocating system resources to scheduling domains, and for policing the actual usage of allocated resources. Indeed the switchlet concept has some similarities with the Nemesis operating system in that real resources are made available to an "application", the control architecture in the network environment, which is then allowed to use these resources according to some internal policy. This similarity is explored further in Chapter 7, where the Nemesis operating system is discussed.

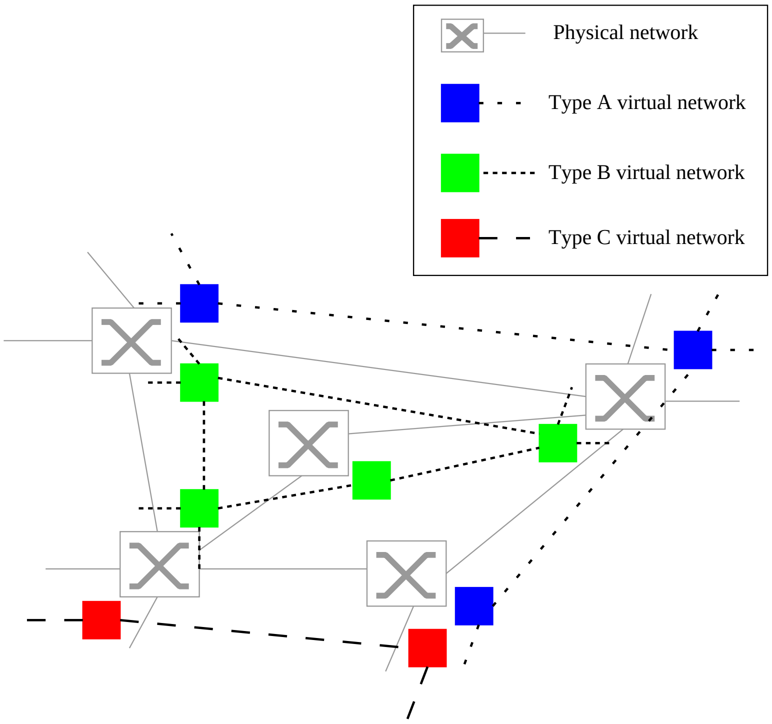

Figure 3.3: Virtual ATM networks with different control architectures

3.3.1 The Prospero Switch Divider Controller

The partitioning of switches into switchlets requires the *specification* of a switchlet in terms of a subset of the physical resources available on the switch. Resources on a switch that need to be considered include:

1. ports
2. VPI/VCI space
3. bandwidth
4. buffer space

The first two items above constitute the connection resources of a switch and can be partitioned at various levels of granularity namely the:

- port level - whereby certain ports within a switch are allocated to a switchlet,

- VP level - whereby certain VPI ranges on certain ports are allocated to a switchlet, and

- VC level - whereby certain VCI ranges on certain VPIs on certain ports are allocated to a switchlet.

Partitioning at the VC level, being the most general of the three possibilities and therefore the least restricting, is the approach which has been adopted for the Prospero Switch Divider Controller[8].

Items 3 and 4 in the list of switch resources, together with the switch queueing and scheduling policies, may be collectively viewed as the *switching capacity* of the switch. As explained in Section 3.2.1, the approach taken with the Ariel interface is to hide QoS details behind the five standardised ATM service categories. The same approach is followed in specifying switchlets: a certain portion of the resources for a particular service category will be marked as "belonging" to a certain switchlet. The control architecture operating on the switchlet can then employ the same resource mapping function mentioned in Section 3.2.1 on its subset of resources to perform admission control and allocation for its connections. It will be shown in Chapter 5 that the use of effective bandwidth to get a handle on the capacity of a switch is particularly attractive in the switchlet environment. This stems from the inherent additive property of effective bandwidth, which means that the effective bandwidth for both switchlets and the switch as a whole can be calculated.

A switchlet specification can therefore consist of the number of ports required, and then for each port the following information:

- the range of VPIs required,

- the range of VCIs per VPI required,

- the service categories required, and

- the capacity required per service category.

[8]When partitioning is done at the VC level, the switch will have to perform per-VC in band policing to protect switchlets from misbehaving users. If a switch is not capable of per-VC protection in the data path, partitioning will have to be done at the VP level. As explained in Chapter 5, dividing at the VP level, although less general, has the advantage that the policing of CAC decisions by Prospero is simplified.

The Prospero Divider Controller has to *know* the amount of resources on the physical switch, or be capable of finding out, and must only allow switchlets to be created when there is sufficient capacity to do so. Allocating resources to a switchlet does not involve any invocations (or allocations) on the physical switch. Rather, Prospero notes the allocation in its internal representation of the switch capacity, and uses that to police future invocations on a switchlet Ariel interface[9]. As indicated in Figure 3.2, partitioning takes place through a separate Management interface on the Prospero Divider Controller.

Once connections have been established in a switchlet (and switch), Prospero has to rely on in-band policing mechanisms in the physical switch to ensure that connections from one switchlet do not adversely affect those of other switchlets. These policing mechanisms exist in current ATM switches in the form of usage parameter control, and therefore require no additional functionality on the part of the switch.

The Management interface on the Divider Controller can only be accessed by the switch "owner", and is not available to the different control architectures operating on the switch. The Management interface allows the dynamic partitioning of resources into switchlets, and the associated creation of switchlet Ariel interfaces. Similarly, switchlets, and switchlet resources, can be released through this interface. The Management interface also provides methods by which resources, e.g. bandwidth, allocated to one switchlet can be "moved" to a different switchlet. The details of the Management interface will be presented in Chapters 4 and 5, where different aspects of the Prospero implementation are presented.

The Prospero Divider Controller associated with each physical switch forms the first building block of the OSSA environment. In the first instance, the Management interface on Prospero can be used by a human operator to create switchlets on a permanent or semi-permanent basis. For each switchlet thus created, the operator can then start up the appropriate control architecture software. The operator will have to ensure that switchlets that are considered part of the same virtual network have the "correct" switchlet resources. For example, to ensure connectivity between two switchlets, the outgoing VCI space of a switchlet has to overlap the incoming VCI space of an adjacent switchlet in the same virtual network.

[9]A possible exception might benefit the partitioning of ABR resources on the switch: The default fairness criteria used by the switch to assign unused bandwidth [Sathaye95], might be modified to take switchlets into account. This will enable fairness to be achieved within a particular switchlet rather that across all switchlets. Further work is needed in this regard.

Although it is feasible, the operator-based approach will undoubtedly be cumbersome and error prone. Of greater interest is the dynamic or on-demand creation of virtual networks by other software systems. Such a *Virtual Network Service* binds the stand-alone Divider Controller entities into an integrated OSSA and is considered in the following section.

3.4 Virtual Network Service

The switchlet concept presented in the previous section is a key component of the OSSA. Sets of switchlets can be combined to form virtual networks. The topologies of such virtual networks are naturally limited to the topology of the physical network, or a subset thereof. The control architectures for these different virtual networks could be instantiations of the same control architecture; alternatively a different control architecture could be operational in each virtual network.

This section extends the OSSA still further describing the components of an *on-demand virtual network service* in which switchlets are dynamically created and merged into virtual networks. A communication mechanism between the OSSA components is required, and by the very nature of the OSSA, such communication is distributed. It therefore makes sense to implement this functionality in the OSSA by means of a Distributed Processing Environment (DPE). In addition to a standardised communication facility, a DPE provides certain services, some of which can be usefully employed by the OSSA. One such service, mentioned in Section 2.1.2, is a *trading* function. If OSSA components, such as the Prospero Divider, are encapsulated as DPE services, then the OSSA becomes an environment in which network components and indeed (virtual) networks become DPE services which can be offered, traded and manipulated like any other DPE service. This concept, and the creation of virtual networks in this manner is considered in Section 3.4.1[10].

Remote access to the Prospero Management interface is required for both types of virtual network creation: slow time scale operations performed by a network operator, and fast time scale operations performed dynamically by software on

[10] At first glance, discussing the *implementation* of the OSSA by means of a DPE in what is considered an *architectural* chapter might seem contradictory or restrictive. However, far from being restrictive this provides the vocabulary and assumed functionality so that discussion can focus on the problem at hand. Furthermore, it should be clear that any other implementation can be engineered, admittedly with greater effort, if the use of a DPE is undesirable.

an on-demand basis. Remote access implies the existence of some form of networking facility. Within the OSSA, this requirement is catered for by means of a *bootstrap virtual network*, which employs a *bootstrap control architecture*. The bootstrap virtual network enables communication at startup time, and provides other facilities during the operation of the system. One such facility could be the DPE used for dynamic virtual network creation, and this arrangement is assumed in the following discussion. Section 3.4.2 considers the bootstrap virtual network in more detail.

The use of a DPE (in the bootstrap virtual network) does not mean that all control architectures have to be implemented by means of a DPE, or even be aware of the existence of a DPE. Indeed, a major strength of the approach presented here is that a conventional control architecture (e.g. an ATM Forum UNI/NNI compliant control architecture), can be instantiated in, and confined to its own virtual network. On the other hand, new control architectures are being developed that can make use of the DPE facilities or can even be implemented in a DPE environment [Lazar95, Rooney97a, van der Merwe97a]. In such cases a DPE will be required, and can be provided by means of the bootstrap virtual network. The combination of bootstrap virtual network and DPE might also lead to some simplifications in a control architecture. For example, a particular control architecture might not implement its own bootstrapping procedures, but instead rely on the bootstrap virtual network to provide such services. The VideoMan control architecture presented in Chapter 6 implements a very simple addressing and routing scheme for the video endpoints that it manipulates, but relies entirely on the bootstrap virtual network and DPE to facilitate communication between its components.

An open architecture such as the one presented in this chapter not only creates new opportunities to exploit in a multi-service network, but is also vulnerable to new forms of abuse. Section 3.4.3 considers some of the security issues involved with an OSSA.

3.4.1 Creating a Virtual Network

This section describes the system services required for the on-demand creation of virtual networks. Figure 3.4 shows the interaction between the services that are involved in this process.

The **Divider Server** encapsulates the Prospero Switch Divider Controller

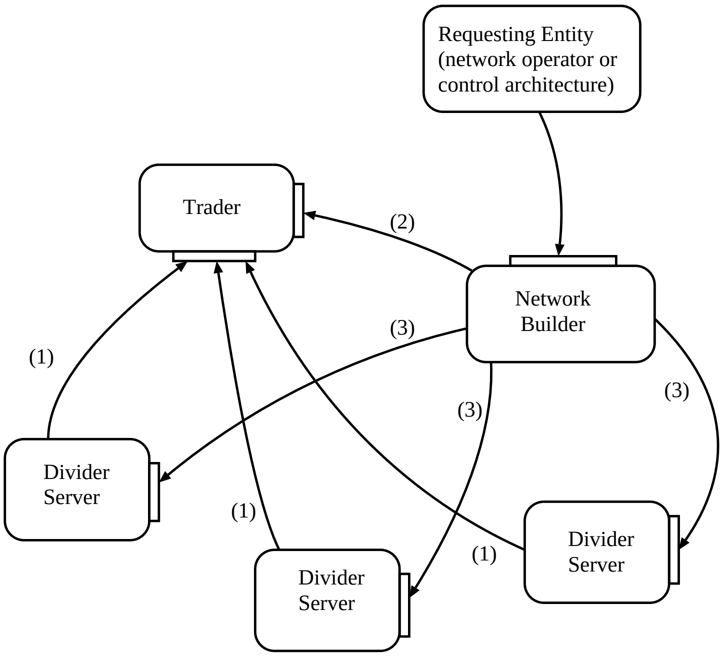

Figure 3.4: Dynamic virtual network services

presented in Section 3.3. The Divider Server informs a Trader service about its existence as well as its switch capacity. (Interaction (1) in the Figure.) As explained in Section 2.1.2, trading is the standard way of matching service providers and consumers in a DPE [APM93]. The Divider Server exports Management interfaces which allow the creation and destruction of switchlets. When a switchlet is created, the Divider Server exports an Ariel switch control interface.

In order to create a virtual network, a **Network Builder** service is provided with the "specification" of the desired network. The network is specified in terms that hitherto could only be contemplated during the network design phase, and could only with great difficulty be changed after network equipment had been commissioned. An example network specification could be:

- UNI 3.1 signalling
- CBR capacity of 15 Mbps
- Between A and B
- With redundancy
- For three hours

Or it could be as simple as a request to create "A cheap network between A and B".

The network specification could be the output of another service, or be provided by a human being.

With reference to Figure 3.4 the creation of a virtual network can be summarised with the following steps:

1. At startup the Divider Server registers with the Trader its configuration and capacity. This process is repeated whenever any of these capabilities change.
2. The Network Builder, on receipt of a virtual network specification, consults the Trader to obtain interface references to all Divider Servers which satisfy the criteria of the request.
3. The Network Builder invokes appropriate methods on the Divider Server(s) to create the switchlets and resulting virtual network.

These interactions are described in more detail in the remainder of this section.

The Network Builder has knowledge about existing virtual networks, and coordinates the creation of virtual networks so that, for example, the address space allocated to two adjacent switchlets in the same virtual network overlaps. When requested to create a virtual network, the Network Builder contacts the Trader and asks for all switches with the required capabilities and capacity according to the supplied network specification. The result of the query, invocation (2) in Figure 3.4, is a list of interface references to Divider Servers matching the criteria.

Using the supplied network specification together with topology information obtained from some topology service (which is not discussed in detail here) the Network Builder determines which switches should form part of the virtual network. The Network Builder then invokes the required operations on appropriate Divider Servers to create the switchlets, invocation(s) (3) in Figure 3.4.

Two possibilities exist in terms of the type of the control architecture which will be instantiated on a newly created virtual network:

- A predefined control architecture from a well known set can be started up when the virtual network is created. An example would be the creation of an

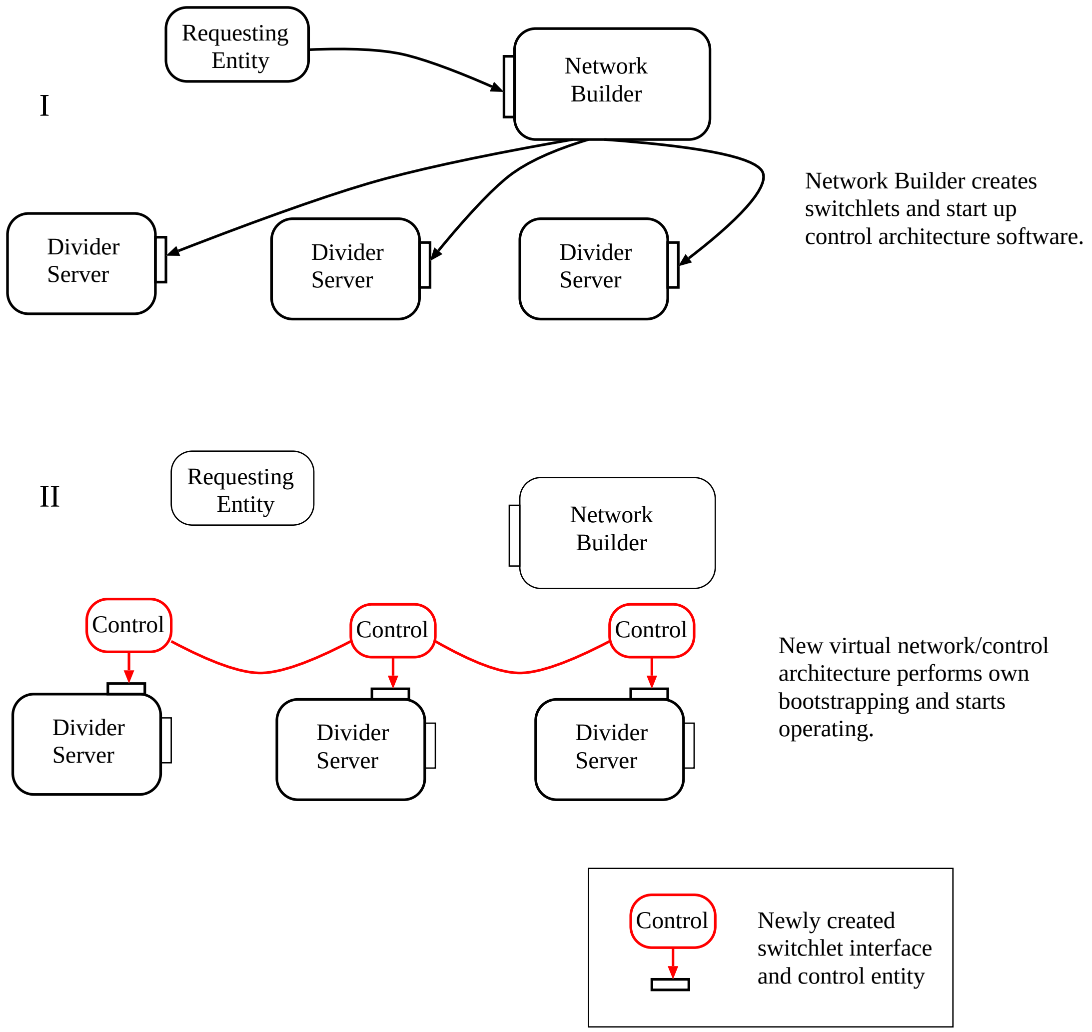

Figure 3.5: Creating a virtual network of predefined type

ATM Forum UNI/NNI compliant virtual network. In DPE terminology this would be called a *traded typed* virtual network. In this case the appropriate software entities will be started up by the Network Builder as soon as the virtual network has been created. Figure 3.5 depicts this process.

- Alternatively, is is possible to create a virtual network without a control architecture. In this case the control architecture is supplied or “filled in” by the entity that requested its creation. This would be called an *anytype* virtual network in DPE terms. In this case the Network Builder will not start up the control architecture, but rather will return an interface reference for each switchlet to the entity that requested creation of the

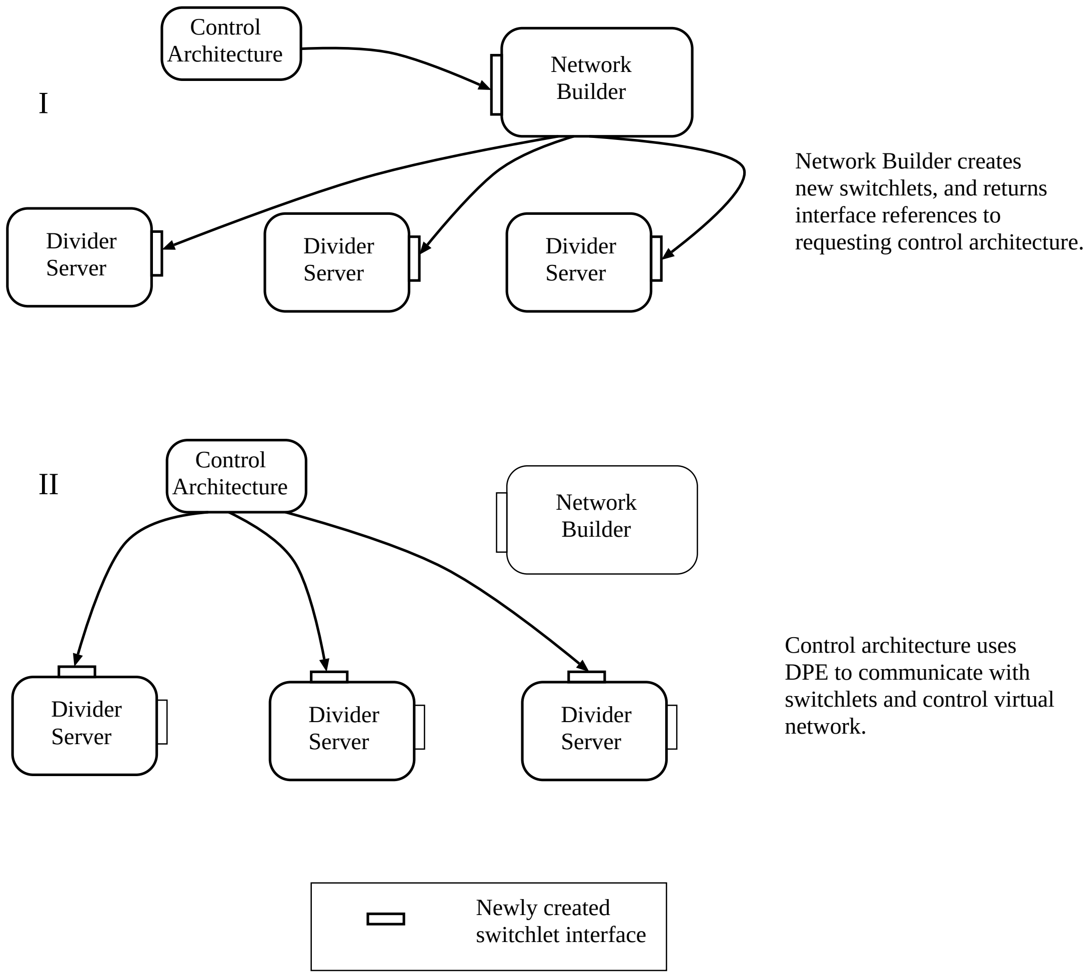

Figure 3.6: Creating an anytype virtual network

virtual network. This entity may itself be the control architecture which will exert control on the newly created virtual network. This is the case depicted in Figure 3.6. Since interaction with the switchlet interface involves the services of the DPE, these control architectures will normally be required to be DPE aware.

In this manner, control architectures of arbitrary functionality can be deployed in the network. One type of control architecture which only makes sense in an OSSA environment is a *service specific control architecture*. A service specific control architecture exploits application knowledge to provide a more efficient service. A service specific control architecture in the OSSA will typically be activated, together with a virtual network, only for the time during which the service is required. Service specific control architectures are considered in detail in Chapter 6.

Once a virtual network has been created using one of the methods described above, it performs its own initialisation, bootstrapping and finally commences normal operation. All its data transfer operations are confined to its "own" switchlets and virtual network. Note that in the case of a conventional control architecture, with its own bootstrap procedure, the operations of the control architecture can be truly confined to its virtual network. This would normally be the case for a traded typed virtual network. The situation is different for a traded anytype virtual network, where the control architecture uses facilities provided by the bootstrap virtual network, or relies on the DPE in the bootstrap virtual network for its communication. This is an important issue in terms of the resources, both network and processing, which have to be allocated to the bootstrap virtual network.

Following the creation of a switchlet, the Divider Server must update the available capacity advertised in its Trader entry, or possibly remove its entry altogether if it has insufficient capacity to permit creation of a new switchlet.

When the time period for which a virtual network has been in operation reaches the allocated or requested time negotiated with the Network Builder at creation time, its resources are released. This requires each switchlet's resources to be released and the Divider Server can then update its Trader entry. Alternatively, if an undefined time period is required, the control architecture will be responsible for periodic "keep alive" requests to the Network Builder to keep the virtual network intact.

As mentioned above, the Network Builder keeps track of all current virtual networks in the system. The Network Builder is also the only entity which is allowed to create and destroy switchlets. It does so by means of appropriate invocations on the Divider Server. This allows the Network Builder to perform garbage collection on virtual networks for which the control architecture malfunctions. Because users are allowed to supply their own control architectures, this is an important function which ensures the integrity of the network as a whole.

In the above discussion, the question of how potential users of the new virtual network get to know about its existence has not been considered. It is assumed that a virtual network is created in response to a request by potential users, or that potential users will be able to obtain this information through some external mechanism.

The following section explores the role of the bootstrap virtual network in more detail.

3.4.2 The Bootstrap Virtual Network

The interaction between OSSA components, as described in the previous section, requires some communication infrastructure. The easiest way to solve this need is by designating a default switchlet in each physical switch to be part of a *Bootstrap Virtual Network*. This bootstrap virtual network implements a *Bootstrap Control Architecture*, the combination of which provides the platform for the required communication infrastructure, which can in turn be catered for through a DPE or by some other means.

Note that the bootstrapping problem is present in all ATM (and other) networks, and is not unique to the environment described in this dissertation. By means of the bootstrap virtual network, the bootstrapping facility has, however, been generalised into something that can provide more sophisticated services. For example, in addition to providing a means of managing virtual networks, the bootstrap virtual network provides a set of generic network services which can be used by any control architecture which has no compelling reason to implement its own. For example, a topology discovery service is provided which can be used by control architectures which do not implement their own topology discovery mechanisms.

Because of the way in which network control is opened up by the OSSA environment, the network can potentially be subjected to new forms of abuse and intrusion. Specifically, because of the dependence of other control architectures on the bootstrap virtual network and control architecture, and because it provides the basis for the management of virtual networks, the bootstrap virtual network has to provide a number of security functions to ensure the integrity of virtual networks. Security in the bootstrap network will in fact be indirectly relied upon by any security mechanisms provided in any other virtual network or control architecture. Some of the security issues involved with the OSSA are considered in the next section.

In addition, the bootstrap virtual network and control architecture has to be "switchlet aware", in that it must start up using a default switchlet while at the same time allowing switchlet allocation for other virtual networks to be performed. Although it is possible to design and build a special control architecture to fulfill these requirements, a more general purpose control architecture will suffice. Indeed, in the proof of concept implementation presented in Chapter 4, an existing IP-over-ATM network was used as bootstrap virtual network. This arrangement was not architecturally clean, because the IP network was not "switchlet aware", and consequently required the pre-allocation of resources in physical switches. On the other hand, the use of a ubiquitous protocol such as IP in the bootstrap virtual network has proved very useful. However, because of its reliance on the very heavyweight ATM Forum control architectures, the Internet Engineering Task Force (IETF) IP-over-ATM mechanism [Laubach94] is not considered an appropriate solution. As far as security is concerned, an IP-based bootstrap virtual network has the useful side effect that securing the bootstrap network is equivalent to securing IP. This problem is receiving considerable attention in the IP community[11].

For these reasons, the use of IP switching [Ipsilon96] as the control architecture in the bootstrap virtual network is an attractive alternative. In the IP switching approach, no conventional ATM control mechanisms are used. Instead, the data transfer capabilities of ATM in combination with IP and a very simple hop-by-hop control protocol are used to realize an IP network. IP switching is discussed in more detail in Chapter 7.

3.4.3 OSSA Security Issues

A significant part of this dissertation motivates the need and usefulness of an open environment like the OSSA. While there appear to be compelling reasons to have such an open architecture, the openness inevitably leads to new possibilities of abuse. A thorough analysis of the security implications of the OSSA is beyond the scope of the dissertation, but this section briefly considers some security issues.

Possible security violations in the OSSA include the following scenarios:

- Resources allocated to one switchlet might accidentally be made available to another control architecture through a different switchlet.
- A control architecture in possession of a valid switchlet might make invocations on its switchlet that adversely effect other switchlets and control architectures.

[11] If the security provided by IP is not adequate, a more secure bootstrap control architecture can be used.

- A denial-of-service attack might be launched against the Network Builder or against a Divider Server.
- A malicious "control architecture" might gain access to the control interface of another control architecture and disrupt connections on that switchlet, or establish connections unknown to the legitimate control architecture.
- An attacker might gain access to the Network Builder service and cause the release of all resources thereby destroying all virtual networks.
- An impostor might request the creation of a virtual network on behalf of someone else.
- A control architecture can accidentally or intentionally modify parameters such that a different virtual network is destroyed.

The first two items on the list would be the result of a defective Divider Server implementation and, although extremely important, are not really security issues. The Divider Server has responsibility to ensure that no control architecture makes invocations that might influence resources other than those allocated to its switchlets.

A denial-of-service attack against a Divider Server might take the form of a malicious control architecture trying to swamp its Ariel interface with requests, thereby preventing other control architectures from making timely invocations. Such an attack can be mitigated if processing resources are also partitioned, for example by using the Nemesis operating system [Leslie96] (see Section 7.5). Other types of denial-of-service attacks might be more difficult to deal with, but these are found in most networking systems and are not specific to the OSSA.

All the other listed violations could be catered for by providing appropriate authentication, data integrity and access control mechanisms. Much of this functionality will be provided by a CORBA compliant DPE which implements the CORBA security specification [OMG95a].

In the OSSA, any user can create a virtual network and in so doing enter the *role* of network provider or service provider. Role based access control [Hayton96] would therefore be the appropriate form of access control to use in an OSSA. Role based access control allows a formal mechanism for defining the roles entities are allowed to play in a system as well as the interaction between roles. For example, the requirements for entering a role can be specified, as well as how entities are elected to roles and how such election can later be revoked.

Another security issue is that of securing communication between the Divider and the physical switch. The security requirements of this communication will very much depend on the exact implementation of switch and Divider combination, as well as the intended use thereof. For example, securing a number of fairly dumb switches with a single control platform which form part of a residential ATM network is very different from (physically) securing a large switch in a national backbone network.

The security aspects of the OSSA will not be considered in any more detail in this dissertation.

3.5 Summary

In this chapter the existence of several approaches to control in ATM networks has been identified. It was observed that these approaches come from different starting points and provide different functionality, and that the notion of a single unifying control architecture probably does not exist. This led to the requirement for an environment that will allow any subset of existing and new control architectures to be operational on the same switch and network.

Such an environment was proposed in the form of an Open Service Support Architecture. A basic building block of the OSSA is the concept of an ATM switchlet. Switchlets provide a mechanism whereby a subset of the resources on a physical ATM switch is presented to a control architecture to manipulate as it sees fit. A switchlet exports to the control architecture an Ariel open switch control interface. The Prospero Switch Divider Controller controls the switch by means of an Ariel interface on the physical switch, and polices invocations made by the different control architectures simultaneously operational through the switchlet interfaces.

Since the partitioning of switch resources is done at a very low level, very few restrictions are imposed on control architectures implemented in the switchlet environment. This allows both conventional control architectures based on message passing protocols, as well as control architectures implemented using DPE methodologies, to be accommodated. As a consequence of this flexibility, it was shown how the switchlet concept can be used to introduce new control architectures into an existing network in a flexible and non-disruptive manner.

The switchlet concept can be used to create *virtual ATM networks* of arbitrary topology on-demand. Each virtual network can be controlled by an arbitrary control architecture which allows a network operator to provide a virtual network service. This means that virtual networks become a service which can be offered and traded like any other service in a DPE environment. The control architecture instantiated in these virtual networks can be known *a priori* or can be supplied by the entity requesting network creation. This allows users to supply and manipulate their own control architectures for a virtual network.

Chapter 4

An Implementation of the OSSA

The feasibility of the approach presented in this thesis is critically dependent on the ability to implement the OSSA on real ATM switches and in a real ATM network. For this reason a proof of concept implementation was undertaken in the experimental environment described in Chapter 2. The implementation is presented in this chapter in two main parts: the first focusing on the Prospero Switch Divider Controller and associated Ariel Open Switch Control interface, and the second on the implementation of the Virtual Network Service.

4.1 Switch Divider Controller

The Prospero Switch Divider Controller divides the resources on an ATM switch into multiple *switchlets*, each of which encapsulates a subset of the physical switch resources. A switchlet is then presented to a control entity by means of an Ariel Open Switch Control Interface; the control architecture can control the switchlet in any way it sees fit. Prospero polices invocations made by the different control entities, and passes only compliant requests on to the physical switch[1]. This arrangement is again depicted in Figure 4.1. (Figure 4.1 is a modified version of Figure 3.2, and shows some of the implementation details.)

[1] A compliant request is one which will only affect resources allocated to the switchlet in question.

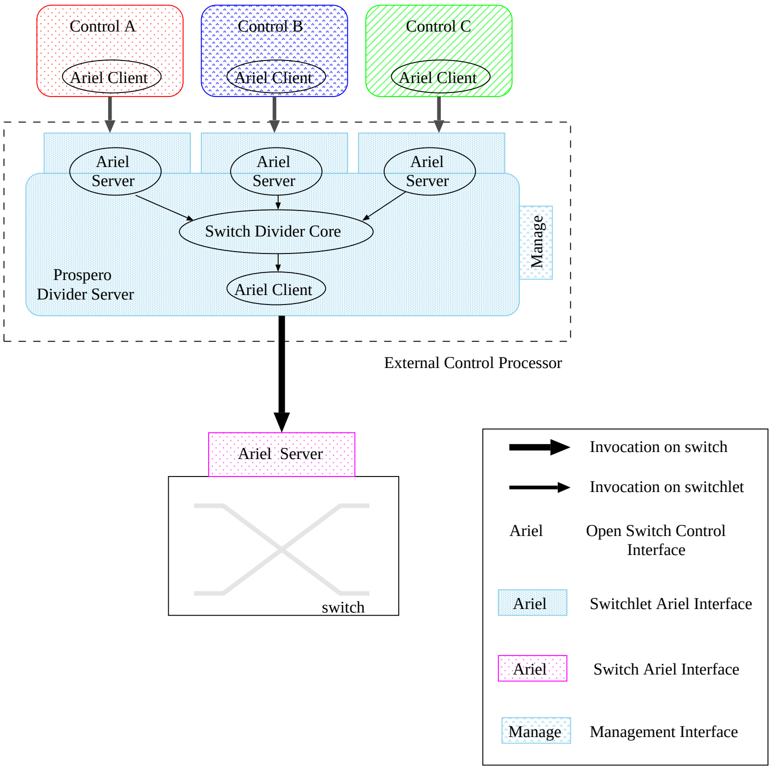

Figure 4.1: Prospero and Ariel implementation

4.1.1 Experimental Environment and Implementation

A subset of the experimental environment described in Chapter 2 was used for this implementation. In particular, the experimental environment consisted of a Fore Systems ASX-100 switch, and several HP-700 series workstations equipped with Fore Systems HPA-200 ATM adapters. As indicated in Section 2.2, the ASX-100 was used because of the availability of low level control information for the switch fabric, which enabled the implementation of several servers on the switch. The Sparc-2 control processor on the ASX-100 also provides a "general purpose" platform complete with a native ATM stack and IP-over-ATM facilities. These capabilities were fully exploited in some of the implementations. The DIMMA environment described in Section 2.2 was the DPE used in some of the implementations. In all such cases the ARPC over UDP protocol stack was used.

Figure 4.1 depicts Prospero with identical control interfaces for the switchlet control interface and the physical switch control interface. While the functionality provided by both Ariel interfaces will be the same, the technology used to implement them might be different[2]. For this reason a distinction will be made in the rest of this chapter between the *switch* Ariel interface, which connects Prospero to the physical switch, and the *switchlet* Ariel interface, which is presented by a switchlet to its control architecture. This distinction is also indicated in Figure 4.1.

Figure 4.1 also shows the Management interface used by the Network Builder to create switchlets and switchlet Ariel interfaces. This interface and the Virtual Network Service implementation are considered in detail in Section 4.2.

The core of the Prospero implementation maintains an internal view of the switch and switchlet resources, and uses a single abstract internal interface to the physical switch. This abstract interface can be mapped onto different switch Ariel interface implementations, any *one* of which can be operational at any time. This means that Prospero can communicate with the switch by means of the Ariel interface that the switch is capable of supporting. Having the switch dictate which Ariel implementation to use means that Prospero can control switches of different scale and complexity.

Similarly, on the switchlet side of Prospero, different implementations of the switchlet Ariel interface exist, and can be mapped onto the internal switchlet implementation. Several of the switchlet Ariel interfaces, of different implementations, can be operational at the same time. This means that Prospero can instantiate a switchlet Ariel implementation to match the requirements or capabilities of a particular control architecture.

In the current implementation, the types and number of switchlet interfaces, as well as the required switch implementation to use, are specified from the command line when the Divider Server is instantiated. Unless stated otherwise, all implementations are in C++. In the implementation presented here, the only resources that are divided and policed by Prospero are ports and VPI/VCI space. Each switchlet invocation is checked to make sure that the specified port and VPI/VCI (if relevant) referenced in the request belongs to the switchlet, i.e. that the request is compliant. In Chapter 5, an extension of this implementation is presented which also polices bandwidth usage.

A subset of the `Configuration` and `Connections` interface methods described in Section 3.2.1 was implemented. In particular all implementations presented

[2]Note again that Ariel specifies the required functionality, rather than a protocol or implementation.

below support the following functions:

- *getPortInfoList* - Provides a list of the switch/switchlet capabilities in a per port list. Information included in the reply is the VPI and VCI ranges available.
- *getCurrentVciList* - A list of all the current VCIs on a switch, or within a switchlet's allocated address space, is provided by this method.
- *createVci* - Allows a VCI to be created in the switchlet and physical switch.
- *deleteVci* - Allows the liberation of a previously created VCI.

These methods provide a client with the means to find out what the capabilities and characteristics of the switch are, and then to perform basic connection creation and deletion functions. More functionality than this was not required for the evaluation presented here.

Two switchlet Ariel implementations were developed. The first provides a DPE type Ariel interface, while the second would enable an IP switching controller to control a switchlet:

- Dariel: This is a DIMMA based implementation of the Ariel interface. It provides the methods listed above to enable switch control, and relies on the DPE for message passing facilities.
- GSMP: This is a server side implementation of the General Switch Management Protocol from Ipsilon [Newman96a]. GSMP is a message passing protocol with well-defined message formats which operates over AAL5 with LLC/SNAP encapsulation[3].

To facilitate comparison between different implementation mechanisms, several switch Ariel implementations were made. One of these implementations, based on the Simple Network Management Protocol (SNMP), is able to utilise the SNMP service which many ATM switches provide. All the Fore Systems switches available in the experimental environment could be controlled by means of this implementation.

[3]Since only a subset of the Ariel interfaces and methods were implemented for this comparison, the standard GSMP interface provided adequate functionality. In general however, GSMP has to be extended to overcome its very basic QoS capabilities.

The other switch Ariel implementations all required an Ariel service to be run on the switch, and were therefore limited to the Fore Systems ASX-100 switch. In all these cases, running the server on the switch did not require disabling the existing signalling capabilities on the switch (i.e. SPANS and UNI). This was effected by allocating a block of permanent virtual circuits (PVCs) as placeholders on the switch, to prevent them being used by SPANS/UNI, and then having the Ariel server "clean up" this space in the hardware VPI and VCI tables on the switch. Because of the way in which VPI/VCI space is allocated on the switch, this is a completely safe (if not architecturally clean) way of allowing several control/signalling entities to be operational simultaneously.

The following switch Ariel implementations were developed:

- SNMP: This implementation uses an SNMP client in Prospero to communicate with an SNMP daemon running on the switch to perform control operations. This approach has the advantage that no special server needs to be run on the switch. Even switches that do not provide any form of open control interface can be integrated into the OSSA environment in this way. Since SNMP was designed to operate on management rather than control time scales, however, there is a heavy performance penalty to be paid for this ease of integration. Because of this, the performance of the SNMP implementation is not really of interest. The implementation was simply built around a Scotty[4] SNMP agent. This implementation will therefore not be considered in the comparison and discussion below.

- Light: This is a locally designed message passing protocol similar in nature and functionality to GSMP. (The Light implementation was developed in the C language.) A client, which is part of Prospero, exchanges messages with an extremely simple server on the switch over a native ATM stack. AAL3/4 or AAL5 can be used for these exchanges, and since the ASX-100 provides hardware support for AAL3/4, this was used in the evaluation. This implementation can use a PVC on a well known VPI/VCI pair, but since SPANS signalling capability was available in the bootstrap virtual network, this was used to establish the ATM connection between the Ariel client and server.

- GSMP: This implementation consists of a GSMP client in Prospero and a GSMP server on the switch. Communication is via a PVC using a na-

[4]Scotty is a Tcl/TK implementation with amongst other things, extensions for SNMP [Schoenwaelder].

tive ATM stack and AAL5 in compliance with the GSMP specification [Newman96a].

- Dariel: In this case, a DIMMA Ariel server was implemented on the switch, with a DIMMA Ariel client as part of Prospero. Since the DIMMA implementation had no ATM capable protocol stack at the time, the IP-over-ATM capability of the switch control processor was exploited for these experiments.

- Prospero: Recognising that the control processor on the target Fore ASX-100 switch is essentially a "general purpose" workstation, an implementation of Prospero was made on the switch. This implementation allows a comparison of the cost of control on the switch with that of moving control out of the switch. Again, communication is by means of IP-over-ATM, this time between a Prospero server on the switch and an Ariel client on a workstation.

Figure 4.2 shows the position of these different implementations within Prospero.

Note that with the set of switch and switchlet Ariel interfaces provided, it is possible to stack more than one Prospero implementation on top of another. For example, a Prospero server can run on the switch with a Prospero workstation implementation, using its Dariel switch control interface to communicate with the switchlet Ariel interface exported from the switch. This means that switchlet resources can in turn be partitioned by another Prospero Divider Server.

The switchlet Ariel servers and the switch Ariel clients, are integrated with the core Prospero mechanisms into a single executable. Standalone versions of all the clients were also implemented for the evaluation reported in the next section.

4.1.2 Evaluation and Results

The efficiency with which control architectures can perform functions such as setting up and tearing down connections is crucial to the success of ATM. This is equally true when these control architectures are simultaneously operating in an OSSA environment. The efficiency of control architectures in the OSSA directly depends on the efficiency of the Ariel open switch control interface, and especially

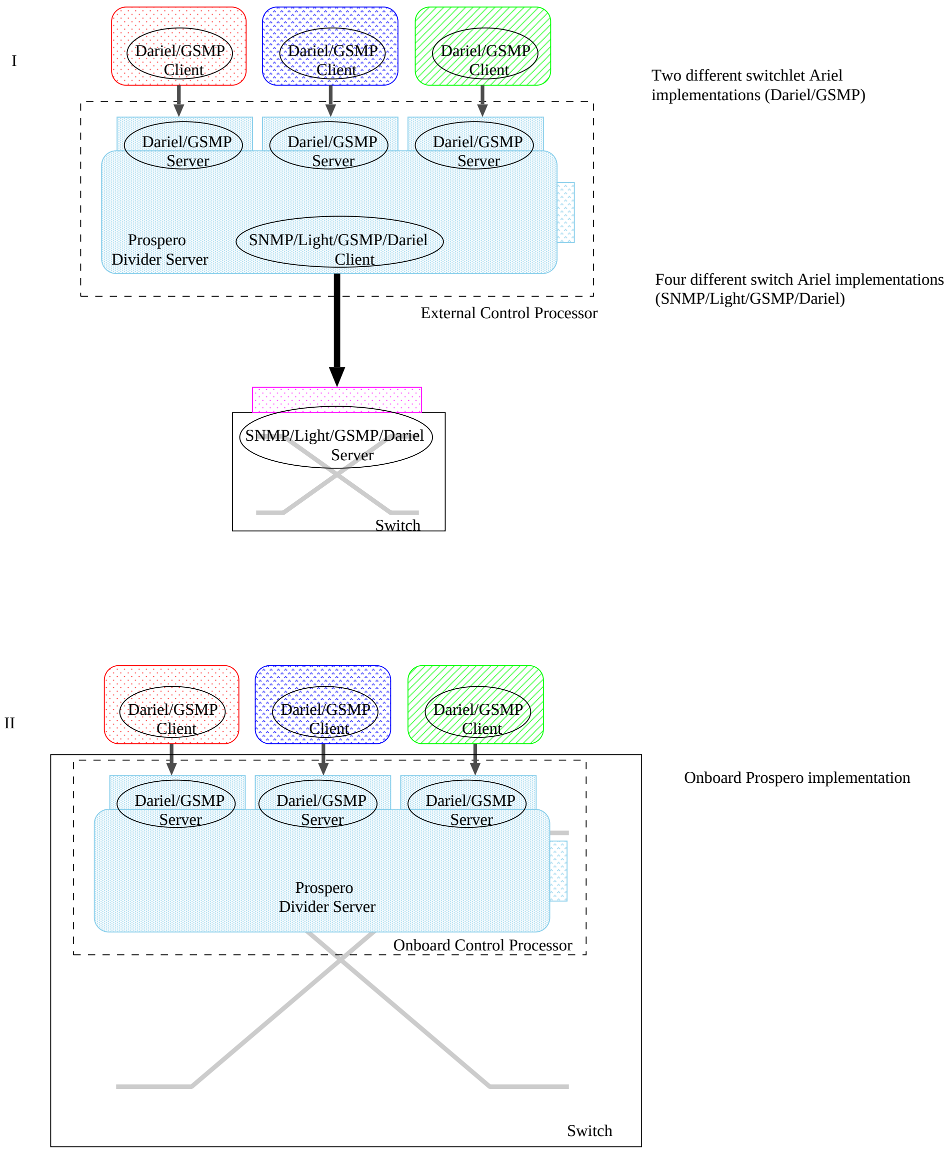

Figure 4.2: Different Ariel and Prospero implementations: (I) Divider Server on external processor, (II) Divider Server on onboard processor

on the additional cost that control operations have to incur by passing through the Prospero Switch Divider Controller.

This section addresses these issues, by evaluating the various Ariel and Prospero implementations. The tests presented below were performed on a lightly loaded network and workstations.

4.1.2.1 Efficiency of Ariel

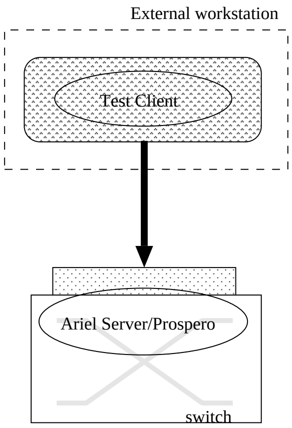

Figure 4.3: Setup to evaluate Ariel implementations

A simple test was performed to evaluate the cost of opening up switch control by means of the Ariel switch control interface. The efficiency of the different Ariel implementations described in the previous section was compared with performing similar operations directly on the switch. In all cases the relevant server was running on the ASX-100 switch, while the stand alone version of the client was running on an HP 9000/725 50 workstation running HPUX version A.09.05, which was directly connected to the switch. This setup is depicted in Figure 4.3.

The test involved the client requesting the switch configuration from the server, followed by 1000 timed invocations. Each timed invocation involved creating and deleting a VC[5]. Requests from the client side were issued from within a tight loop, with no delay between requests. Since all servers on the switch shared the same library interfacing with the low level switch hardware[6], this test was essentially an evaluation of the efficiency of the communication channel used, as well as the efficiency of different server implementations.

The time it takes a process *on* the switch to perform the create/delete invocation pair represents a lower bound on the expected invocation times. In this

[5]Both "create" and "delete" were performed as an invocation pair, as that meant that the same VCI value could be used for the complete test run. In this manner the limited VCI space available did not hamper the test.

[6]The code for this library was derived from the ASX-100 driver in the public domain Vendor Independent Network Control Entity (VINCE) version 1.1 [VINCE]. The library provides the means for a process running on the ASX-100 to map the switch fabric control interface into memory, and to then perform various control functions on the switch fabric.

Operation	Time (ms)
Local Setup/Teardown	0.14
Two Null RPCs	7.6

Table 4.1: Local operation and Null RPC: Average time for 1000 operations

case, the invocation amounts to little more than a number of local function calls. The average time (over 1000 invocations) to perform the create/delete invocation pair for such a process was measured as 0.14 ms. The time taken for a null RPC between the workstation and the switch was measured to be 3.8 ms. These times are shown in Table 4.1 and the average invocation times for connection setup/teardown by means of the various Ariel implementations are presented in Table 4.2. Note that the time for *two* null RPCs is given to allow easier comparison with the two invocations on the Ariel interface.

Ariel Setup/Teardown pair	Time (ms)
Light	4.5
GSMP[7]	8.9
Dariel	8.1
Onboard Prospero	8.3

Table 4.2: Ariel evaluation: Average time for 1000 operations

There is clearly a significant performance penalty to be paid for opening up switch control, and for performing control from an external workstation. Invoking control operations from an external workstation is between 30 and 60 times slower than the time required to do the same operations directly on the switch.

The significant difference between the Light and GSMP implementations is somewhat surprising since both are essentially message passing implementations. This difference is mainly due to the fact that the Light implementation uses AAL3/4, for which the switch has hardware support, whereas GSMP uses AAL5 which is implemented in software on the switch. As a test, the GSMP protocol was run over AAL3/4 which improved the average time for an invocation pair to 6.8 ms. The fact that the GSMP implementation is slightly more complicated

[7]GSMP specifies an adjacency protocol which detects when either the client or the server is restarted. This process involves the period exchange of messages, and the adjacency protocol was therefore disabled for the duration of the timed invocations.

due to the more robust GSMP specification, could account for the remaining discrepancy.

The average time for the two DIMMA implementations (Dariel and Prospero) is almost identical, which indicates that the policing mechanisms employed by the Prospero Divider impose a negligible overhead. Given the trivial nature of this policing in the implementation presented here, this is to be expected.

Comparing the Light implementation with the two DIMMA implementations, there is clearly a price to be paid for the ease of use of a general purpose DPE. The DIMMA implementations are nearly twice as expensive as the Light implementation.

Clearly, the major contribution to the cost of setting up and tearing down a connection is the communication cost. Twice the cost of a null RPC is only slightly less than the cost of a DIMMA invocation pair. Communication is therefore the part that needs to be optimised if a more efficient implementation is required. At the same time, the setting up and tearing down operations that were performed are the simplest that can be imagined in an ATM environment. If more complex control functions such as routing and CAC are to be performed, then the advantage of an external processor, which might be significantly more powerful than the onboard processor, might cancel the communication overhead.

4.1.2.2 The Cost of Using Prospero

The cost of using Prospero to partition a switch, and to police the usage of the resulting switchlets, is crucial to the feasibility of the OSSA approach. A number of tests were therefore performed to evaluate the cost of having Prospero in the control path. As before, the average time for 1000 VC create and delete pairs was measured. The different test setups are depicted in Figure 4.4 and are described below:

- In the first test, Prospero was run on the physical switch, with a Dariel client on an HP 9000/725 50 workstation directly connected to the switch, as shown in Figure 4.4 (I). (The result of this test was also presented in the previous section but is repeated here for clarity.) The average time for an invocation pair was 8.3 ms.
- Prospero was then run on the workstation with a Light Ariel server running on the switch, and the Dariel client running on the same workstation as

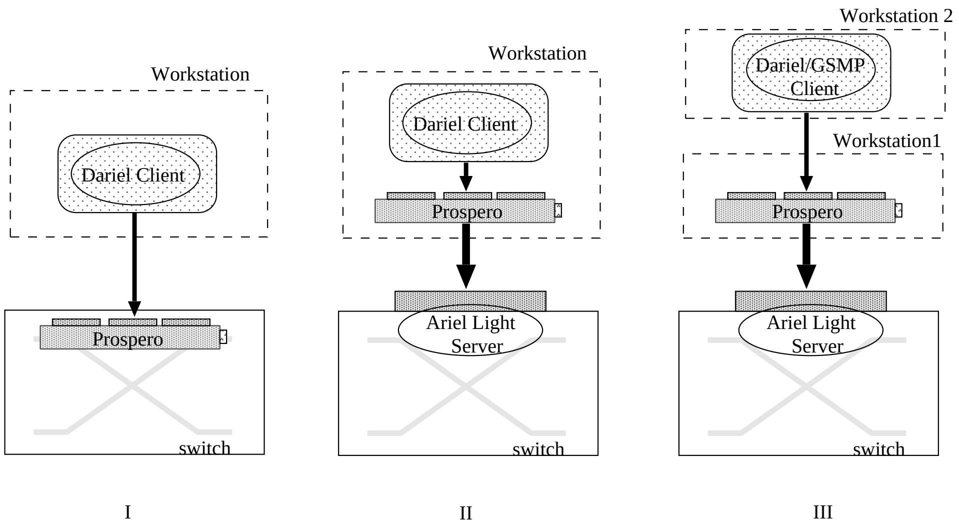

Figure 4.4: Different configurations to evaluate Prospero

Prospero. This is depicted in Figure 4.4 (II), and the average time for an invocation pair was 10.4 ms. The time for a same machine null RPC on this workstation was measured to be 2.4 ms[8].

- Third, the above test was repeated, but this time the Dariel client was run on a different HP workstation (an HP 9000/725 100) connected to the same switch, Figure 4.4 (III). (Note that the figure shows the *logical* connectivity between OSSA components, and the not the physical connectivity of the workstations to the switch.) The time for a null RPC on this faster workstation was measured as 1.2 ms. The time for a null RPC between the machine running the Dariel client, and the machine running Prospero was measured to be 2.3 ms. The average time for the create/delete invocation pair was 10.1 ms.

- Finally, the above test was repeated but with Prospero presenting a GSMP Ariel interface, and running a GSMP client on the second workstation. The average time for an invocation pair was 7.7 ms.

The results described above are also presented in short form in Tables 4.3 and 4.4.

[8]Note that DIMMA does not provide for same machine optimisation, e.g. shared memory, so invocations between the Dariel client and Prospero, were processed by the kernel IP protocol engine.

Null RPC	Time (ms)
Same machine (Dariel client workstation)	1.2
Same machine (Prospero workstation)	2.4
Inter machine	2.3

Table 4.3: Prospero evaluation: Average time for 1000 null RPCs

Configuration	Time (ms)
Prospero on switch	8.3
Prospero and Dariel client on same workstation	10.4
Prospero and Dariel client on different workstations	10.1
Prospero and GSMP client on different workstations	7.7

Table 4.4: Prospero evaluation: Average time for 1000 operations

Again it is clear that the major cost in performing these invocations lies in the communication overhead. For example, for the third test, the measured time of 10.1 ms can almost be accounted for by adding up the cost for a Light invocation pair and two (null) RPC calls. The time for a Light invocation pair, according to Table 4.2, is 4.5 ms. Add to this the time for two null RPC calls between the two workstations from Table 4.3 (2×2.3 ms = 4.6 ms) to give a total of 9.1 ms.

Running both Prospero and the Dariel client on the same machine is slightly slower than the test with the Dariel client on a separate machine. This can be attributed to the fact that an inter machine null RPC is slightly faster than a null RPC on the machine which ran both processes.

Interestingly, the average time for the GSMP client operating through Prospero is actually less than the average time for the case where the GSMP client communicated directly with a GSMP server on the switch. (Compare Tables 4.2 and 4.4.) This can be directly attributed to the hardware support for AAL5 processing on the workstation, in contrast to the switch where AAL5 processing was done in software.

The results also seem to indicate that a general purpose RPC mechanism, such as that provided by DIMMA, is too heavyweight for the relatively simple interaction between an Ariel client and server. It should be noted, however, that the null RPC times for DIMMA presented above are not necessarily the best that can be achieved with a CORBA implementation. For example, performance figures for omniORB [ORL97] report sub-millisecond null RPC times on a variety of platforms.

4.1.3 Conclusion

The results presented above were obtained from a number of implementations none of which was optimised to any great extent. Although optimisation would clearly be possible, the results appear to be in the same range as those reported elsewhere. For example in [Veeraraghavan95] a figure of between 4 and 10 ms is assumed a reasonable time to perform an operation roughly equivalent to those described above[9]. The results also compare favourably with figures quoted in [Shumate94] for a GSMP implementation to control an AN/2 switch, where a figure of 14 ms for the GSMP equivalent message for creating a VC is given. Finally, the best results for a GSMP invocation through Prospero are about three times slower than results published by the inventors of GSMP (without the Divider overhead) [Newman97b]. While this indicates significant room for improvement, the performance seems reasonable for a proof of concept implementation with limited AAL5 hardware support.

The results presented in this section seem quite promising and indicate that:

- Moving control out of the switch is not prohibitively expensive in terms of performance.
- The advantage of using a general purpose DPE to provide the control communications infrastructure might outweigh its cost when compared to message passing techniques. This will, however, depend on the availability of more efficient implementations.
- The low overhead incurred by having Prospero in the control path to the switch means that the approach presented in this dissertation is a practical way to accommodate several control architectures on the same switch.
- The presented approach is very flexible in that Prospero can either be implemented external to the switch, or on the onboard control processor. This means the approach is inherently scalable because both simple cheap

[9] The quoted figure is for "Broadband Channel Control", which performs connection acceptance control, the reservation of switch resources and the setting up of VCI/VPI entries.

switches and sophisticated expensive switches can be integrated into the OSSA. The performance/cost tradeoff, i.e. simple switches with slower external Prospero versus more expensive switches with an onboard implementation, can therefore be exploited according to the needs of a particular environment.

The implementations presented in this section show that the switchlet approach to open network control is feasible. The next section completes the picture by presenting an implementation of all the OSSA components in a Virtual Network Service implementation.

4.2 Virtual Network Service

As indicated in Section 3.4, switches are nodes within networks, and the division of switch resources into switchlets enables the creation of virtual networks. Each virtual network implements a different instance of a control architecture. If the creation of switchlets and virtual networks is automated, virtual networks can be created dynamically. This section describes an implementation of such a dynamic networking environment.

Section 3.4.2 mentioned that the use of IP in the bootstrap virtual network has some attractive properties, such as the availability of existing IP based services. One of the services that the bootstrap virtual network can supply is that of a DPE. Such a DPE (in the form of DIMMA) is used in the implementation presented below. The use of a DPE is not a requirement, and a solution based on well-defined messages and message passing can be engineered easily enough (although undoubtedly with significantly more effort).

The Ariel and Prospero implementations presented in the previous section made use of a single Fore ASX-100 switch. For the Dynamic Virtual Network implementation, the set of experimental switches was extended to include two Fore ASX-200 switches. SPANS and UNI Signalling available on the Fore Systems Switches were used as the bootstrap control architecture, with an IP-over-ATM implementation providing the communications requirements of DIMMA. The VPI/VCI space available to this bootstrap control architecture was limited, with the remainder of the VPI/VCI space being made available to a Prospero Divider Controller implementation using the previously mentioned placeholder PVCs. This arrangement is illustrated in Figure 4.5. The SNMP Ariel implementation was used to communicate with the switches because it allowed all three Fore Systems switches to be controlled. In this case, control was exercised by means of SNMP, and the PVC placeholders were used as follows: On creating a new VC, appropriate in and out VCI placeholders were removed before the new VC was created. The reverse sequence of operations was performed when such a VC was later removed. This meant that within the VCI space set aside for Prospero, valid PVCs were in place almost continuously. This, in turn, prevented the control architectures resident on the switch from using VCIs in this range.

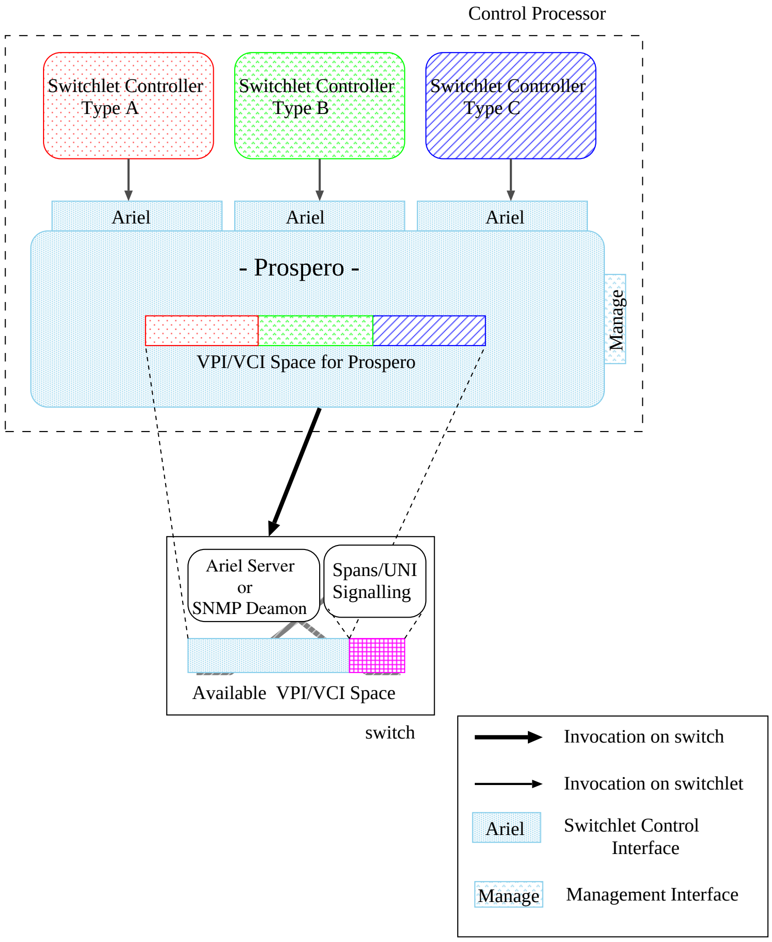

Figure 4.5: Prospero and the Bootstrap Control Architecture

4.2.1 Implementation

Figure 4.5 shows three switchlet Ariel interfaces as well as the Management interface by which switchlets and control interfaces are created. The Management interface consist of the following interfaces[10]:

- `Dynamic`: This interface has two methods which allow the dynamic creation and destruction of switchlets. The current implementation does not allow the characteristics of a switchlet to be specified; instead a limited number of "default" switchlets can be created. A side effect of creating a switchlet is that the VCI space allocated to the switchlet will be "cleaned up", i.e. all connections falling in the range allocated to the switchlet are released, so that the VCI space is presented to the control architecture in a known state. Clearing the VCI space in this way dominates the time it takes to create a switchlet in the proof of concept implementation: To create a switchlet with 7 ports and 1 VPI with 40 VCIs per port takes in the order of 40 ms.
- `Configure`: Allows a network administrator to find out what the current configuration of the Divider Server is. For example, a method is provided to list the number of Dariel and the number of GSMP switchlet interfaces. In Chapter 5, the implementation of methods that allow bandwidth to be moved between switchlets (and virtual networks) will be discussed.

The Network Builder implementation is depicted in Figure 4.6. The Network Builder exports two interfaces, namely `Dynamic` and `Topology`. The `Dynamic` interface facilitates the manipulation of virtual networks, while the `Topology` interface provides topology information to control architectures that do not have their own topology services. (As indicated in Figure 4.6, these are essentially two separate functions which could be split into two separate servers.)

A trivial access control mechanism is provided in the current implementation. A small database is kept of users who are currently allowed to create virtual networks. This database is updated by the "system administrator", who also starts up the Divider Servers and the Network Builder. The database is consulted by the Network Builder when a user requests creation of a virtual network[11].

[10] As is the case for the Ariel interface, the Management interface really consists of a set of interfaces.

[11] This trivial access control mechanism, useful though it is in the current experimental environment, would be completely inadequate in a commercial implementation. As indicated

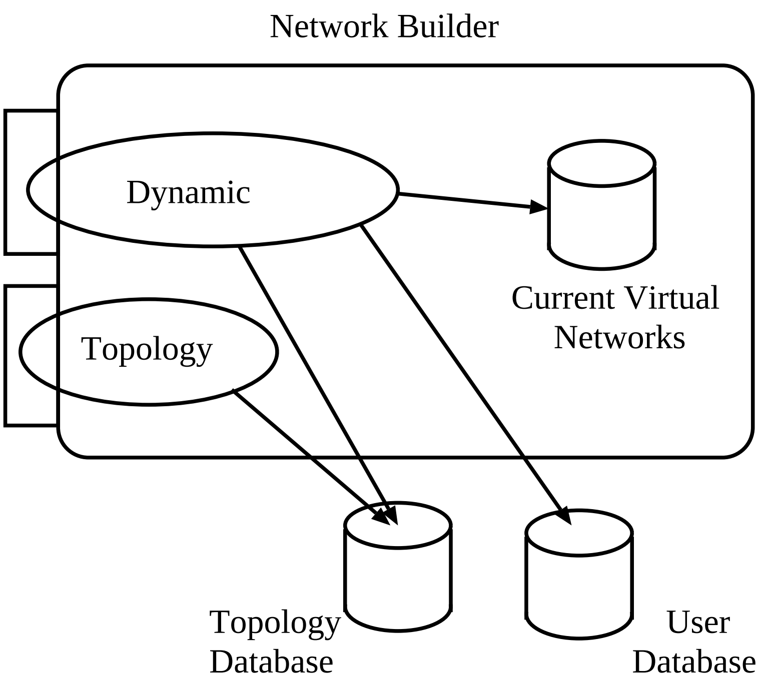

Figure 4.6: Network Builder implementation

The methods provided by the `Dynamic` and `Topology` interfaces are as follows:

- `Dynamic`: This interface allows for the creation and removal of virtual networks. In addition it allows the modification of existing (active) virtual networks:
  - *createNamedNetwork*: If successful, this method returns a network reference number to the requester. The network reference number is unique within a particular Network Builder, and is used as an argument to subsequent invocations on the Network Builder to identify the appropriate virtual network. A virtual network created with this method contains no switchlets, and switchlets have to be added by appropriate invocations on the methods below.
  - *getNamedNetwork*: A set of switchlets for a predefined virtual network topology is created, and returned to the caller together with a network number. This method is for control architectures which cannot make

in Section 3.4.3, role based access control appears to naturally fit the requirements for access control in this environment.

use of the facility to add switchlets to their virtual network one at a time.

  - *addSwitch*: Requests the Network Builder to add a switchlet for a named switch to a previously created virtual network.
  - *removeSwitch*: Requests the Network Builder to remove a previously allocated switchlet from a virtual network.
  - *removeNetwork*: Requests the destruction of an entire virtual network. The Network Builder will remove all switchlets in the virtual network which have not been removed already.

- `Topology` - In the current implementation, topology information is entered into a topology database which the Network Builder reads at startup time. (In a real implementation, this static procedure would have to be replaced with a dynamic topology discovery service.) In the database, switches and endpoints are identified with a human readable name or string. A switch is given a system wide unique identifier, while an endpoint is identified within the network by the switch identifier and switch port to which it is directly connected. This topological information is made available to control architectures through the methods described below. This information can be used by control architectures which lack their own topology discovery services. The implementation of such a control architecture is presented in Chapter 6.

  - *getSwitchTopology*: Provides complete switch topology information according to the Network Builder database. A switch topology information element consists of a switch identifier, and a list of (remote switch, local port, remote port) tuples. A sequence of these elements is returned by the Network Builder.
  - *getEndpointAddress*: Provides a ¡switch identifier, port number¿ pair, which together constitute an "endpoint address".
  - *getSwitchName*: Returns the switch name given a switch identifier.
  - *getSwitchId*: Returns the switch identifier given a switch name.

As indicated in Figure 4.6, the Network Builder maintains an internal database of the currently active virtual networks. This per virtual network information includes the switchlets in each virtual network. This means that a control architecture can simply invoke the *removeNetwork* method, upon which the Network Builder will release all resources currently held by the virtual network.

Chapter 6 will show how this dynamic virtual network environment is used by a service specific control architecture to create, modify and control a virtual network based on the requirements of the virtual network users.

4.3 Summary

In this chapter, various implementations of a subset of the Ariel open control interface for both switch and switchlet use were compared with an onboard "closed" implementation providing the same functionality. As can be expected, controlling a switch from an external control processor is significantly slower than the onboard equivalent. This performance penalty is however not prohibitively expensive, and it is expected that the communications overhead will become less significant when more complex control functions such as CAC are performed.

The cost of having the Prospero Switch Divider Controller in the control path was evaluated. In this context, the purpose of Prospero is to police invocations made by a control architecture on a switchlet Ariel interface. The cost of Prospero was not significantly more than that of a single switch Ariel interface. Furthermore, depending on the capabilities of the onboard control processor, Prospero can be implemented either on the switch, or in an external processor.

One of the implemented switchlet Ariel interfaces exports a GSMP interface. This means that an IP switching controller can be directly integrated into the OSSA environment. The mechanisms to initiate an IP switching controller, or indeed any other standard control architecture, have not yet been implemented.

A Virtual Network Service implementation which uses the Prospero and Ariel building blocks was presented. This completes the set of OSSA components, and in Chapter 6 it will be shown how a control architecture can make use of this environment. The control architecture presented in Chapter 6 is OSSA aware and makes full use of all the OSSA services.

This chapter was the first of two which describes implementation aspects of the OSSA. Bandwidth management in the OSSA, and an implementation based on measurement based estimates of effective bandwidth, is described in Chapter 5.

Chapter 5

Bandwidth Management in the OSSA

A fundamental goal of the OSSA approach is to provide a subset of switch and network resources to a control architecture to use as it sees fit. This requires the existence of effective and efficient resource management mechanisms. One of the most crucial resources in an ATM network is bandwidth, and this chapter investigates the use of the *effective bandwidth* concept as a mechanism to perform bandwidth management in the OSSA[1].

5.1 Introduction

Bandwidth management in the OSSA needs to be considered at different levels of granularity and on different time scales. Firstly, bandwidth management needs to be performed at the call or connection level. This process is normally called *connection acceptance control* or *connection admission control* (CAC). CAC deals with the question of whether, given the current state of the network, a new connection can be accepted into the network in such a way that its requested QoS parameters can be satisfied without adversely affecting existing connections[2].

[1]The definition of effective bandwidth presented in Section 5.2 will show that effective bandwidth encompass more than just the bandwidth resources of a switch. In particular, the buffering capabilities of a switch are also taken into account, and effective bandwidth is therefore actually a more general *resource* management mechanism.

[2]This definition of CAC is based on a traditional connection oriented ATM environment. The "CAC" process in IP switching will not quite fit this definition. In IP switching flows are

Equally important in the OSSA is bandwidth management at the virtual network level. The first question to be answered in this case is similar to the CAC problem, namely whether a new virtual network can be created given the current state of the network, or *virtual network acceptance control* (VNAC). Bandwidth allocated to a virtual network can be guaranteed for the duration of its existence (hard guarantee). Alternatively, a virtual network can be given a statistical or soft guarantee and ATM's potential multiplexing gain can be exploited to allow more virtual networks to be created. This is similar to peak rate allocation versus effective bandwidth allocation in CAC. A more general concept specific to virtual networks, is the ability to arbitrarily allocate and reallocate bandwidth resources in a virtual network environment. Bandwidth management at the virtual network level normally happens at a slower time scale than that required by CAC.

Bandwidth management is still an active area of research and while several solutions have been proposed no clear winner has emerged. [Perros96] gives an overview of recent CAC approaches. Recent developments in an approach based on estimation of effective bandwidth computed from online measurements have shown some very promising results [Duffield95, Crosby95b, Lewis97][3]. This approach has the further advantage that it can equally well be applied to both single connections as well as groups of connections. This feature is particularly attractive in the OSSA environment where the partitioning of switch resources into switchlets requires bandwidth management at both the call/connection level and the virtual network level. Other bandwidth management approaches have also been extended from connection level CAC [Hyman91], to virtual path and virtual network levels [Hyman93, Hyman94]. However, these approaches lack the simplicity and generality offered by the effective bandwidth approach.

A further attractive feature of the measured effective bandwidth approach is that it enables the network to exploit the multiplexing gain of several multiplexed data streams. This advantage can be used at all levels of bandwidth management. For these reasons effective bandwidth estimates based on online measurements have been investigated for use in the OSSA.

This chapter considers the requirements of bandwidth management in the

switched based on, for example, the duration of the flow or according to some user defined policy. Simple priority levels are considered sufficient in terms of the quality offered to different flows.

[3]Measured effective bandwidth is the subject of a comprehensive research project entitled "Measure" in which the Computer Laboratory is a partner. This has facilitated access to both expertise and implementations.

OSSA environment and describes an implementation based on effective bandwidth estimation. Section 5.2 gives a short overview of effective bandwidth estimation using online measurements. Section 5.3 considers the requirements of bandwidth management in the OSSA, and it is shown how effective bandwidth can be used for this purpose. Section 5.4 presents an implementation of estimating effective bandwidth in the OSSA environment by using online measurements, and Section 5.5 shows some results obtained from this implementation. The chapter ends with a discussion of the results and a summary.

5.2 Effective Bandwidth Estimation Using Online Measurements

The concept of effective bandwidth is an attempt to get a handle on the "actual" or "effective" bandwidth requirements of a connection or a group of connections. Effective bandwidth is defined informally as the rate at which a buffer should be emptied in order to keep cell losses due to buffer overflow below a certain bound. More formally: Given a certain cell arrival process, the effective bandwidth is the minimum service rate at which a single server queue should be drained to ensure that the probability of overflow is less than a given level. This section will briefly present the theory on which this approach is based [Hui88, Kelly96, Lewis96], and show how effective bandwidth can be estimated based on traffic measurements [Crosby95b, Crosby96]. It is then shown how this can be used for bandwidth management within the OSSA.

5.2.1 Theoretical Background

Effective bandwidth is based on Large Deviations Theory, also referred to as the theory of rare events. The rare events of interest for effective bandwidth are the loss of cells due to buffer overflow in an ATM switch. This section presents a brief derivation of the relevant equations.

Suppose a buffer of size b is emptied at a constant service rate s. (In an output buffered ATM switch, with a single FIFO buffer, this would correspond to the buffer size and line rate respectively.) For a given buffer size, a higher service rate would reduce the cell loss ratio (CLR). Similarly, for a certain service rate, a bigger buffer size would reduce the CLR. The CLR is thus a function of both buffer size and service rate: $CLR(b, s)$. If the arrivals process to the buffer is approximately stationary and mixing (i.e. has no long range dependence), then the loss ratio decays exponentially for large buffer sizes [Glynn94]:

$$\lim_{b \to \infty} \frac{1}{b} \log CLR(b, s) \approx -\delta(s). \tag{5.1}$$

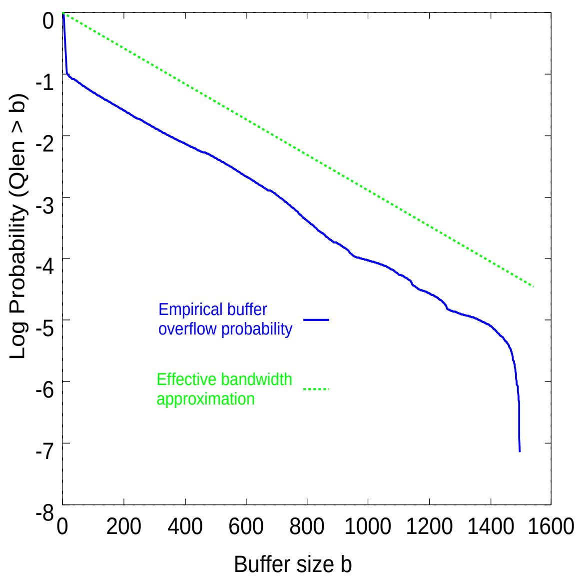

Figure 5.1: Empirical loss probability for 18 multiplexed JPEG video sources, and simple effective bandwidth approximation.

Real switches of course have finite buffer space and real traffic may not satisfy the stationary and mixing assumptions, but empirical evidence suggest that the log linear nature of Equation 5.1 holds for practical buffer sizes and real traffic sources. For example, Figure 5.1 shows the empirical buffer overflow probability for 18 multiplexed JPEG video sources. This figure is reproduced from [Lewis97], and the measurements were taken on a Fairisle switch [Leslie91] by Simon Crosby of the Computer Laboratory. The figure also shows the simple approximation of the loss probability by means of a straight line through the origin, according to Equation 5.1. This approximation can be improved by not having the straight line bound go through the origin. Indeed such refinements have been proposed [Crosby96]. The derivation presented here, and the results described in the rest of the section, use the simple approximation suggested by Equation 5.1.

The decay rate δ in Equation 5.1 is a function of the service rate s, and is related to the scaled Cumulant Generating Function (sCGF) of the arrivals process [Lewis96]:

$$\delta(s) = max\{\theta : \lambda_A(\theta) \leq s\theta\}, \tag{5.2}$$

where λ_A is the sCGF of the arrivals process and defined by:

$$\lambda_A(\theta) = \lim_{n\to\infty} \log E \exp(\theta A_n), \tag{5.3}$$

with A_n the number of cells arriving in an interval of length n.

Equation 5.1 leads to the simple effective bandwidth approximation:

$$CLR(b, s) \approx e^{-b\delta(s)}. \tag{5.4}$$

Reiterating the definition for effective bandwidth using the symbols introduced above, "Given a buffer of size b and an arrivals process A_n, what is the minimum service rate s_{eff} required to guarantee the probability of overflow is less that some target cell loss ratio $TCLR$", or more formally using Equation 5.4,

$$s_{eff}(b, TCLR) = min\{s : e^{-\delta(s)b} \leq TCLR\}, \tag{5.5}$$

leading to,

$$s_{eff}(b, TCLR) = \frac{\lambda(\theta_{eff})}{\theta_{eff}}, \tag{5.6}$$

with $\theta_{eff} = -\log(TCLR)/b$. The function $\lambda(\theta)/\theta$ is known as the effective bandwidth function.

If the arrivals A_n are weakly dependent, the sCGF can be approximated by a finite-time cumulant generating function:

$$\lambda_A(\theta) \approx \lambda_A^T(\theta) = \frac{1}{T} \log E \exp(\theta A_T), \tag{5.7}$$

for T sufficiently large. The value of the expectation can then be estimated by breaking the arrival data into blocks of length T and averaging over them, which gives the following estimation of the sCGF:

$$\hat{\lambda}_A(\theta) =: \frac{1}{T} \log \frac{1}{K} \sum_{k=1}^{k=K} \exp(\theta \tilde{X}_k), \tag{5.8}$$

where $\tilde{X}_k$ are the cell arrivals in the k^{th} block. The estimated effective bandwidth is then simply:

$$\hat{s}_{eff}(b, TCLR) = \frac{\hat{\lambda}(\theta_{eff})}{\theta_{eff}}. \tag{5.9}$$

Figure 5.2 shows an estimated sCGF which was calculated using Equation 5.8. The sCGF is for a single "bursty" source of the type used in Section 5.5.3.

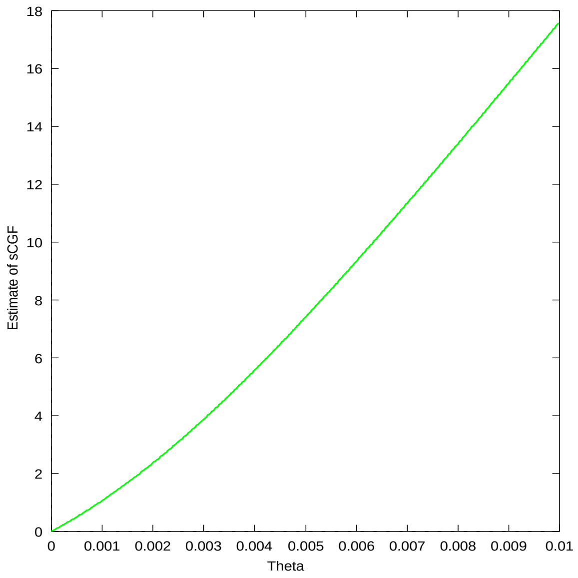

Figure 5.2: Estimated sCGF of a single "bursty" source

5.2.2 Using Measured Effective Bandwidth for CAC

By measuring the aggregate cell arrivals on a switch port and utilising Equations 5.8 and 5.9, the effective bandwidth required for the multiplex can be estimated. By doing the same for only a subset of the active connections on a switch port, the effective bandwidth requirement for this subset can be calculated. This is very attractive for the OSSA because the same technique can be used for both switchlets and physical switches.

The use of measured effective bandwidth for CAC can be divided into two parts [Lewis97]:

- Predicting the influence of the new connection.
- Estimating the effective bandwidth of the current multiplex (or in the case of the OSSA a subset of the multiplex).

In the simplest case predicting the influence of the new connection can be based on its declared peak rate, and estimating the bandwidth of the multiplex can be based on the effective bandwidth estimate in Equation 5.9. The sum of the declared peak and the measured effective bandwidth can then be compared with the line rate to effect CAC. In the switchlet environment each switchlet port is allocated a certain percentage of the line rate, and this *allocated bandwidth* is then used, together with the measured effective bandwidth, to determine if the connection can be accepted.

Intuitively any measurement based approach could lead to problems. For example, what would happen if a number of sources start transmission at a value significantly below their peak rates, and only after a while start sending at their peak rates ? If connections have been accepted during the initial period, based on the measured effective bandwidth, the switch port might well be over committed. The way to address this problem is to balance the empirical measured evidence with declared traffic parameters. For example, if the declared peak values are somehow taken into account, so that the CAC process becomes more conservative as more connections are accepted, this problem can be reduced. Similarly, if in addition to its peak rate a source declares some other traffic characteristics, these might be used to make a better prediction of its behaviour. For these reasons refinements of the basic effective bandwidth approach have been proposed in [Crosby96, Lewis97]. These enhanced algorithms are not taken into account for the work described here which is based on the basic theory presented in the previous section.

Despite these inherent problems with any measurement based approach, there appears to be consensus that these approaches [Jamin97, Grossglauser97b, Lewis97], can be more successful than those based on traffic models and predefined traffic types [Hyman91, Gelenbe97]. A detailed comparison with alternative approaches to bandwidth management is conducted in Chapter 7.

5.3 Resources Management in the OSSA

As mentioned in the introduction, bandwidth management in the OSSA takes place at different levels of granularity and at different time scales. Section 5.3.1 explains the need for CAC at various places in the OSSA. Section 5.3.2 explains how measured effective bandwidth can be used for bandwidth management at the virtual network level.

5.3.1 Connection Admission Control in the OSSA

In the discussion of the Ariel switch control interface in Section 3.2.1, the requirement to perform CAC outside the physical switch was explained: The physical switch should be kept as simple as possible, and different control architectures should be able to use different CAC algorithms if they so wish.

In Section 3.3.1 the switch resources to be considered for partitioning were listed. Because of the desire to impose as few as possible restrictions on control architectures, the partitioning is done at the lowest possible level. So for "connections resources" (i.e. ports, VPs, VCs) it is done at the VC-level.

Partitioning at the VC-level is somewhat at odds with the desire to do CAC in the control architecture. The Divider Controller is responsible for the policing of all invocations made by different control architectures on switchlet Ariel interfaces. This includes ensuring that a control architecture is not using more bandwidth than was allocated to its switchlet, and is therefore effectively a CAC process.

This does not imply, however, that the policing done in the Divider is exactly the same as the CAC carried out by the control architecture. The control architecture could for example accept or reject connections based on some additional information such as the time of day, recent history, or some control architecture specific policy. The policing done in the Divider is, on the other hand, a simple connection admittance decision based on resource usage[4].

Except for the obvious duplication of functionality, performing CAC in both the control architecture and the Divider could cause an additional problem. If for example, the CAC in the control architecture is more sophisticated than that in the Divider, the control architecture might accept a connection which is then subsequently rejected by the Divider. One way to offset this problem is to allow the Divider to take a more global view, by for example looking at the resource usage of both the switchlet and the switch as a whole. In this way, if the switch as a whole is underutilised, but a particular switchlet is close to its allocated bandwidth, a connection can still be accepted.

Another advantage of having CAC in the Divider is that some control architectures might not want to perform their own CAC and can then rely on that provided by the Divider. Alternatively, some control architectures, for example IP switching, might create connections without taking into account the QoS

[4] I am indebted to Sean Rooney of the Computer Laboratory for this line of thinking.

guarantees of existing connections. In such cases the Divider Server will be responsible for doing some form of CAC to protect other switchlets. In the case of IP switching for example, if such a switchlet is exceeding its bandwidth allocation, all further connection requests can be rejected until the usage falls below a certain threshold[5].

If partitioning is done at the VP-level the need for doing CAC in the Divider disappears. In such a scenario a switchlet would consist of a set of switch ports, and on each port a set of virtual paths. The Divider can then simply rely on the in band policing of the VP as a single entity to ensure that a control architecture stays within its allocated bounds. If the control architecture accepts more connections than it should, VP policing will result in cells from offending connections being dropped, without influencing other switchlets (and virtual networks). Partitioning at the VP-level is however considered too restrictive in the general case.

Finally it is possible that some well known control architectures, for example the predefined or typed control architectures of Section 3.4.1, will be trusted to do their own CAC. Again this will not be the general case.

The discussion above holds true regardless of which CAC algorithm or mechanism is used. The implementation described in Section 5.4 makes use of estimates of effective bandwidth based on online measurements to perform CAC, assumes that CAC is done within the Divider, and the control architecture used for testing does no CAC of its own.

5.3.2 Virtual Network Bandwidth Management

In the simplest case the allocation of switchlet (and virtual network) bandwidth resources can be simply based on some percentage of the line rate. This hard partitioning of bandwidth would be equivalent to doing CAC simply based on peak rate allocation. As in the case of peak rate allocation, this would lead to significant wastage of bandwidth resources[6]. This wastage will be aggravated in

[5]This policing on its own will not provide adequate protection, because any single connection could potentially consume more than the allocated bandwidth for a switchlet. As explained in Chapter 3, this problem is avoided by means of in-band policing mechanisms. A first level of protection for an IP switching control architecture could be to limit the bandwidth of all created connections to the bandwidth allocation of the switchlet, or a percentage thereof.

[6]Note that virtual network users might want, or be willing to pay for, such guaranteed resources.

the OSSA, because in general significant percentages of the line rate could be allocated to a switchlet, and would thus be unusable by other control architectures[7]. This could lead to two problems:

- Allowing fewer virtual networks to be operational than the physical network is capable of handling.
- The situation envisaged in the previous section, where a specific switchlet is running at capacity while the rest of the switch is under utilised.

In exactly the same way that the measured effective bandwidth of the aggregate traffic on a port is used to do CAC, the measured aggregate traffic can be used to decide whether a new virtual network can be created or not[8]. Again, as in the case of effective bandwidth CAC, the danger exists that too many virtual networks are accepted because of, for example, inaccurate estimates, or unknown traffic patterns. This problem is of course aggravated by the fact that the bandwidth usage of a virtual network will by its very nature vary as connections come and go.

In the case of VNAC however, feedback to the Divider or control architecture in the form of the allocated bandwidth for a switchlet limits the problem. For example, rather than simply using effective bandwidth measurements to do virtual network admission control, such measures can be used to move bandwidth between virtual networks as needed. Since the bandwidth allocation reduced or increased in this fashion will be taken into account by the CAC mechanism (either in the Divider or the control architecture), the over subscription of resources can be prevented or at least contained.

The above mechanism logically extends to the concept of *guaranteed* and *optimistic* capacity. A virtual network can for example be allocated a certain guaranteed bandwidth, and be assured that its bandwidth will never be decreased below that. In addition a virtual network can be optimistically allocated a percentage of unallocated bandwidth, and be allowed to exceed its allocated bandwidth up to

[7]Note that even with hard bandwidth partitions, the OSSA approach still wins in terms of economies of scale; It is cheaper to build a large switch on which several different network types can be accommodated and resources moved according to their needs, than to build several smaller switches for each of the network types.

[8]Note that the OSSA is flexible enough to simultaneously allow both hard and soft partitioning: Some virtual networks can have a fixed and protected bandwidth allocation, while for others a statistical guarantee will suffice.

this value. (A similar approach is proposed in [Floyd95] for the sharing of bandwidth in packet based networks. This link-sharing approach will be considered in detail in Chapter 7.) When bandwidth needs to be allocated as guaranteed bandwidth to a new virtual network however, the optimistic capacity of an existing network can be reduced or taken away completely. This will have the effect that all new connections for the existing virtual network will be rejected until its effective bandwidth falls below the new sum of guaranteed and optimistic bandwidths. Furthermore, this naturally leads to an environment in which control architectures can allocate and free bandwidth in much the same way that processes allocate and free memory in an operating system[9]. This concept is called *incremental bandwidth management*, indicating the fact that bandwidth can be allocated and freed as needed.

The concepts of guaranteed and optimistic bandwidth, and incremental bandwidth management are worthy of further investigation. Of more immediate interest though for the OSSA, is the ability to move bandwidth between virtual networks on a slower time scale than connection setup. Such functionality is crucial in a virtual network environment. The implementation described in Section 5.4, therefore concentrates on the mechanisms required to enable a network manager (or network management software) to perform such operations.

5.4 Implementation

The OSSA implementation presented in Chapter 4 was extended in order to:

- Show that the measured effective bandwidth approach is feasible on a real ATM switch.
- Explore the usefulness of this approach within the OSSA.

Because of the need for low level access to switch internals in order to do traffic measurements, the implementation was limited to the ASX-100 switch in the experimental environment. Figure 5.3 depicts the extended Divider Server implementation and shows the interaction between different components.

[9]Note that doing the same for connections resources (ports, VPs, VCs) is not quite as simple, because a certain amount of dependence between adjacent nodes is required to ensure that, for example, VCI spaces overlap.

The **Measure Server** in Figure 5.3 contains the algorithms for estimating effective bandwidth, and controls a database which contains all traffic measurements received from the switch. The Measure Server was derived from an early version of the "Measure Toolkit" implemented by Horst Meyerdierks of the University of Glasgow, who in turn found inspiration from an implementation by Brian McGurk of the Dublin Institute of Advanced Studies.

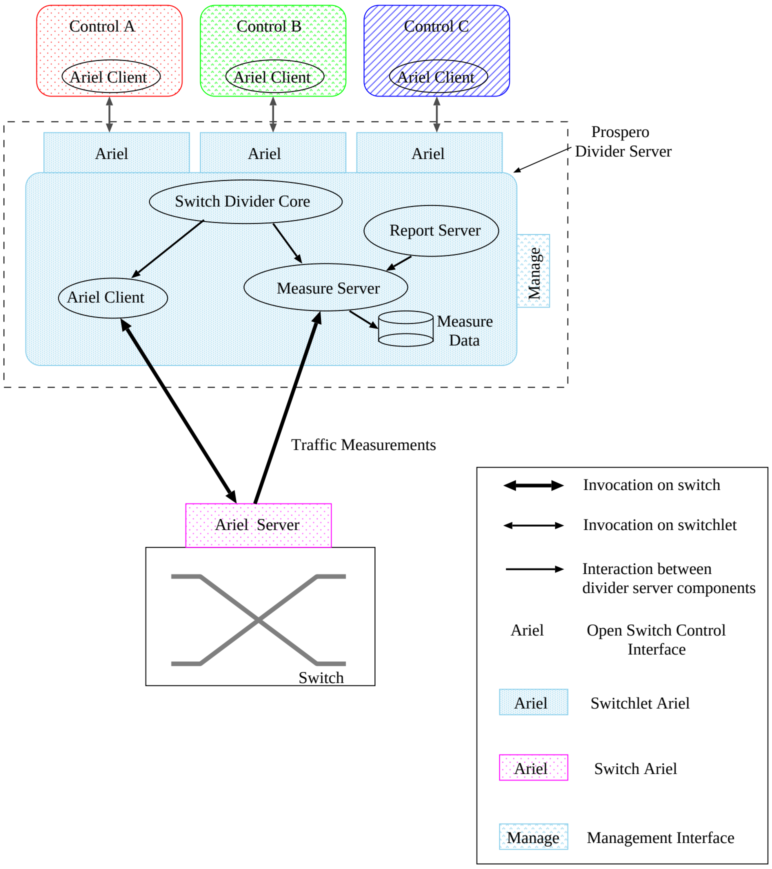

Figure 5.3: Divider Server implementation with measurement based bandwidth management

The version of the Measure Toolkit used in the OSSA implementation assumed that the aggregate cell arrivals per port were being used in the estimation process. This resulted in a number of problems:

- Looking at aggregate arrivals per port means that it is very difficult to know whether a fall in measured arrivals was caused by a connection leaving, or by a bursty connection in its "off" period. Without such knowledge all measured arrival samples have to be taken into account, and the highest arrival rate will dominate the exponential sum in Equation 5.8, thus effectively dominating the bandwidth estimate.

- Looking at aggregate arrivals per port does not allow per switchlet estimates to be performed.

One way of solving the first problem is to detect "significant" level shifts in the aggregate arrival rate. By combining this information with the time scales over which it occurs, it might be possible to determine when connections have left and thus which measurements can be ignored.

A more precise (if more tedious) way of dealing with this problem is to make use of the exact connection arrival/departure information and to keep per VCI measurements. The entity dealing with CAC can be reasonably expected to know about connection arrivals and departures. By only keeping measurements for "active" VCIs in the measurement database, the correct data can be used for the estimation process. Since keeping per VCI measurements also deals with the problem of doing per switchlet estimates, this approach was followed in the implementation described here.

Getting per VCI measurements from the physical switch of course results in significantly more data being exchanged between the physical switch and the Measure Server than in the case of aggregate measurements. This seems to favour a Measure Server implementation on the physical switch controller. On the other hand, the estimation process requires a significant amount of floating point arithmetic. The ideal implementation would therefore be a tradeoff between the increased communication involved in getting per VCI measurements off the switch in an external implementation, versus the processing capabilities of the onboard switch processor. In the implementation presented here, the Divider Server was run on an external workstation mainly because of the flexible experimental environment that such an implementation provides. As indicated in Figure 5.3, the Measure Server was integrated with the Divider Server.

The GSMP version of the Ariel interface was extended to facilitate the transfer of arrival measurements from the switch to the Divider Server. A message was added which allows the Divider Server to request the Ariel server on the switch to send it per VCI statistics at a certain rate (e.g. every 200 ms). The Ariel server will then continuously send out these statistics until the Divider Server requests it to stop. Since this departs from the original request-response nature of the

GSMP protocol [Newman96a], a sequence number was added to the message sent from the Ariel server to the client in the Divider Server, to enable the latter to detect lost messages.

To calculate and send per VCI cell arrivals for the 16 port ASX-100 switch took an average of 64 ms. Traffic measurements were taken for 256 VCIs per port and only for VPI zero. Of the 64 ms, 20 ms was contributed by the operations on the switch and the remaining 44 ms was taken up by sending measurements to the external workstation. The operations on the switch are fairly expensive because they involve walking through the complete forwarding table to sum per VCI output port statistics. Even so it is significant that the communication overhead completely dominates the off-loading of measurements.

The Measure Server exports (internal to the Divider Server) a set of interfaces through which effective bandwidth estimates of any range of VCIs on any port can be requested. In the current implementation this facility is used as follows:

- The CAC function in the Divider core, which forms part of the policing function of the Divider, invokes this interface to obtain the effective bandwidth estimate for the subset of switchlet VCIs for that port. The declared peak rate of the connection to be set up is added to this, and the result compared to the allocated bandwidth for this switchlet to decide whether the connection can be accepted or not.
- On a periodic basis the Report Server depicted in Figure 5.3 goes through the list of switchlets and obtains bandwidth estimates for each switchlet port. The current effective bandwidth estimate, as well as the maximum effective bandwidth for that switchlet port is stored with other switchlet specific information. These values are updated on a frequent basis and the current effective bandwidth estimate can therefore be used by the CAC function rather than recalculating it[10]. The actual aim of storing the current and maximum values is to aid the moving of bandwidth between virtual networks as explained below.

Storing the maximum effective bandwidth together with the current effective bandwidth is an attempt to keep some history on how the effective bandwidth changes over time. Only the maximum value is stored within the Report Server

[10]The maximum frequency at which sensible estimates can be obtained would be the frequency with which measurements are received from the switch.

to minimise storage requirements. In practice, a network manager would be interested to see how this maximum value changes over time. Rather than storing a complete history in the Report Server, the Divider Server `Configure` interface was extended with a method which allows the maximum effective bandwidth value to be obtained. Through this interface, a network manager can obtain these maximum values to construct a usage history over time scales that are of interest to it.

Knowing the resource usage of a virtual network as a function of time, enables a network manager to move bandwidth between virtual networks. For example, bandwidth from an under utilised virtual network can be moved to one that is running close to its allocated capacity. Another example would be the moving of resources between virtual networks at certain times of day, due to, for example, contractual obligations.

To enable these functions to be performed, the Divider Server `Configure` interface was extended with the following methods:

- *getSL_BW*: Returns the allocated bandwidth, in addition to the maximum and current effective bandwidth estimates for the specified switchlet port.
- *setSL_BW*: Sets the value of the allocated bandwidth for the specified switchlet and switchlet port.

The first method allows a management entity to obtain the current bandwidth allocation as well as the maximum and current effective bandwidth estimates. As explained above, this information, or information obtained over a period of time through this method, can be used to make decisions about the moving of resources between virtual networks. The actual movement or reallocation of bandwidth resources can be achieved by means of the second method. Use of the extended interface will be demonstrated in the results reported in the next section.

5.5 Results

5.5.1 Computational Efficiency of Algorithm

As indicated in Section 5.2, calculating the effective bandwidth involves a number of floating point operations. In addition, effective bandwidth calculations depend on the number of traffic measurements used in the estimation process. As can be expected, this results in a linear increase in the time it takes to calculate effective bandwidth as the number of measurements increases. The time taken to estimate the aggregate effective bandwidth for a single port, i.e. a single stream of measurements, increases linearly from 0.47 ms for 128 samples, to 13.7 ms for 4096 samples. These results were obtained on an HP 9000/725 50 workstation running HPUX version A.09.05.

It is possible to store partial sums of previous estimations if only aggregate effective bandwidth estimates are of interest. Because of the need to do per switchlet estimates, such optimisations were not explored here.

5.5.2 Simple Effective Bandwidth Experiments

The Divider Server implementation was instrumented in order to log the various results shown in this section. In all experiments, the measured arrival statistics were received from the physical switch approximately every 200 ms. In the graphs below these arrivals are indicated as "Cell Arrivals" in cells/s. Most graphs also contain a plot of the estimated effective bandwidth resulting from these arrivals. Bandwidth estimates were requested approximately every 800 ms. In all bandwidth estimates a target CLR of 10^{-2} was assumed, and the Fore ASX-100 has an output buffer size of 256 cells.

Figure 5.4 shows the arrivals and estimated effective bandwidth for a source that was peak rate limited to 2 Mbps (or 4717 cells/s). The source in this case is an ATM connected workstation continuously sending AAL5 packets. The Fore Systems ATM adapter does peak rate limiting on the packet stream at the adapter level resulting in the almost constant bit rate (CBR) arrivals in Figure 5.4[11]. Within approximately 20 s of the source starting, the effective bandwidth estimates fall exactly on the near constant cell rate.

Figure 5.5 shows the arrivals and effective bandwidth for a JPEG video source. The video traffic was produced by an AVA video adapter at five frames per second. Live video was fed into the AVA from a video camera. Figure 5.5 clearly shows the bursty nature of the JPEG source. After slight initial variations, the effective bandwidth estimates stabilise at approximately 95 percent of the maximum cell arrival rate.

[11] While fairly constant, the cell rate is somewhat below the theoretical 4717 cells/s. This appears to be the result of the way in which rate limiting is done on the Fore adapter.

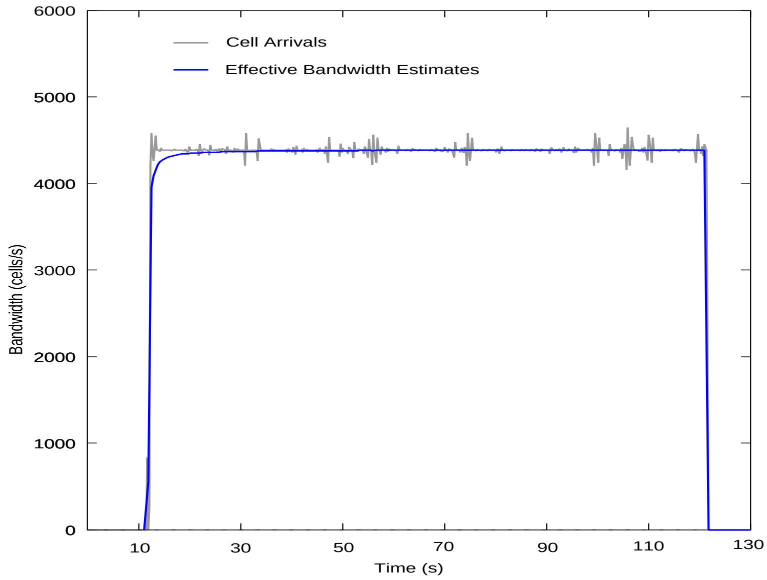

Figure 5.4: Effective bandwidth estimate of peak limited source

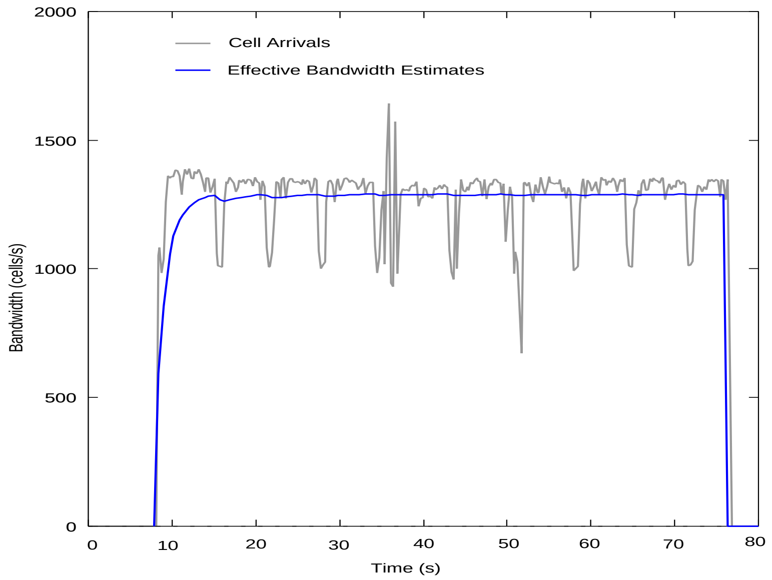

Figure 5.5: Effective bandwidth estimate of JPEG source

The traces shown in Figures 5.4 and 5.5, actually formed part of a switchlet experiment in which each of the streams were traversing separate switchlets. Figure 5.6 shows the two sets of graphs combined in time on a single graph. Figure 5.7 shows the aggregate arrivals for the port as a whole, as well as the

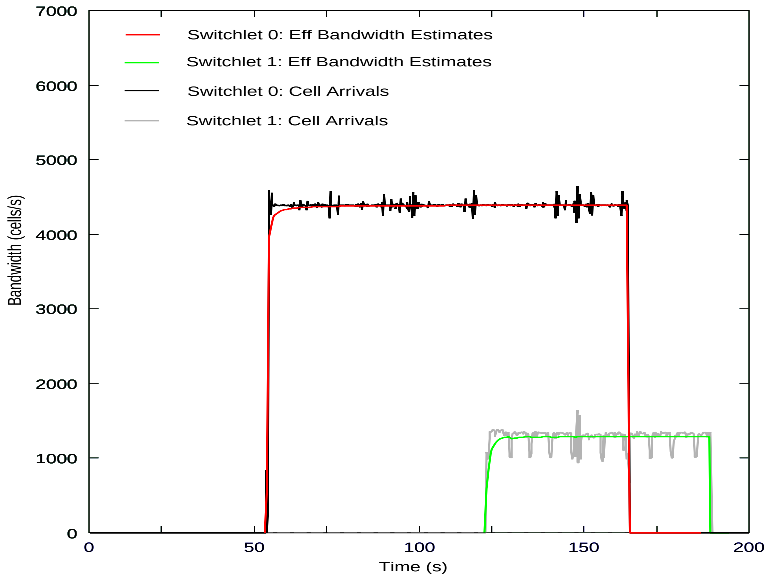

Figure 5.6: Peak limited and JPEG sources in separate switchlets

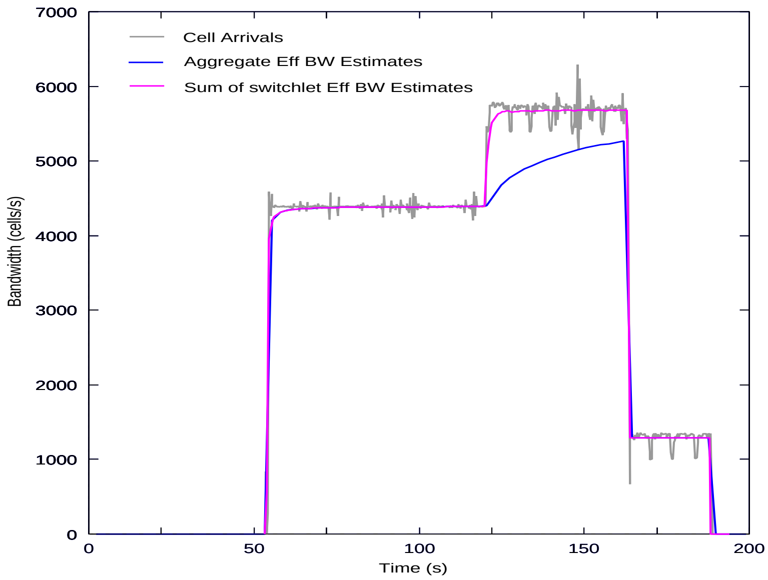

Figure 5.7: Aggregate effective bandwidth and sum of switchlet effective bandwidths

effective bandwidth for the aggregate traffic. Figure 5.7 also shows the sum of the effective bandwidths for the switchlets. The difference between the sum trace and the aggregate trace shows the multiplexing gain achieved by the multiplexing of the two sources. This also means that switchlets which use only the switchlet effective bandwidth for CAC will be acting conservatively.

5.5.3 Simple Bandwidth Management and CAC Experiments

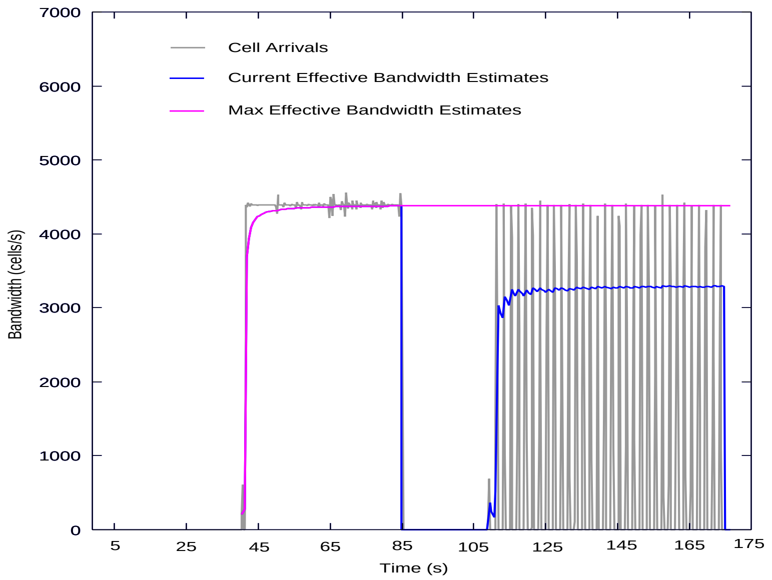

Figure 5.8: Keeping the maximum effective bandwidth for a switchlet

The first experiment only serves to show how the Report Server stores both the current and maximum effective bandwidth values. Figure 5.8 shows the arrivals for two consecutive connections: a near constant bit rate connection from approximately 40 to 85 s, followed by a very bursty connection from approximately 110 to 170 s. Figure 5.8 shows how the current effective bandwidth follows the "shape" of the arrivals; when the near CBR source leaves at 85 s, the current effective bandwidth drops to zero, and starts estimating the bursty connection from about 110 s. The maximum effective bandwidth estimates stay at the level of the first connection, which has a higher maximum than the second bursty connection. The reason why the maximum effective bandwidth curve suddenly stops after approximately 170 s, is that the switchlet was released after deletion of the second connection, and these values are only kept for active switchlets.

Again the traces in Figure 5.8 were obtained by instrumenting the Divider Server to produce the required logs. The burstiness of the second connection resulted from sending 5 packets in a tight loop from the workstation, followed by an "off" period of approximately 2 s. As for the near constant rate source, the bursty source was peak rate limited to 2 Mbps. The two sources were generated on two separate workstations.

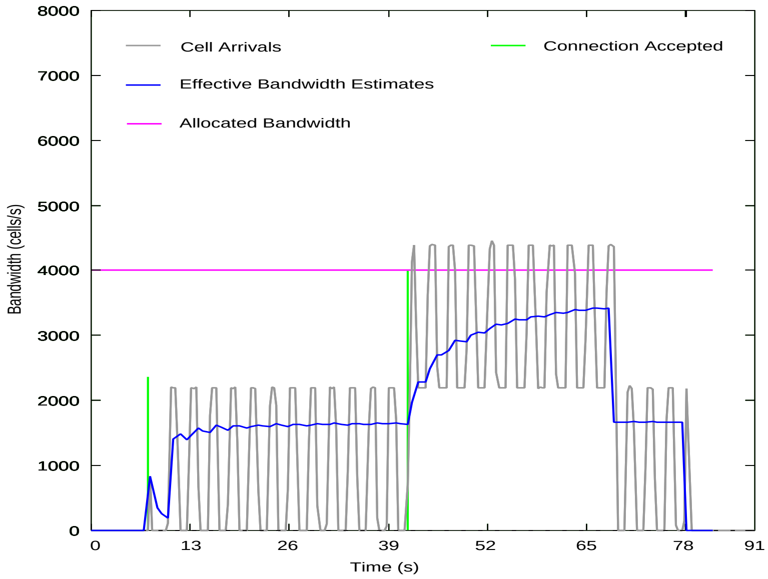

Figure 5.9: CAC with one bursty and one near CBR source

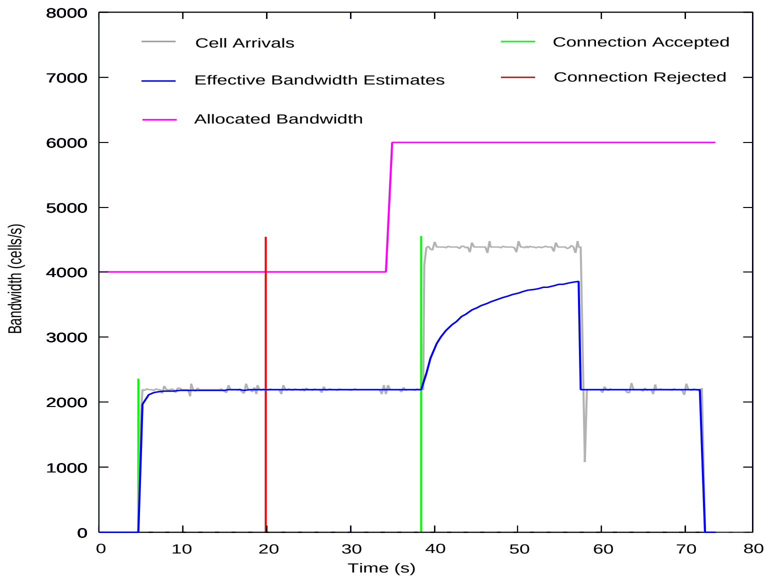

Figure 5.10: CAC and bandwidth management with two near CBR sources

The same near constant rate and bursty sources were used in the next experiment for which the results are shown in Figure 5.9. The allocated bandwidth of the switchlet under consideration was reduced to 4000 cells/s, and CAC based on the measured effective bandwidth and the declared peak rates was performed. The vertical line at approximately 6 s shows the sum of bandwidths on which the first connection is accepted. The first bursty connection is followed by the near constant rate connection which is accepted at approximately 42 s, i.e. at this point both connections are active in the switchlet. Again the height of the line indicates the sum of bandwidths on which an acceptance decision is made. This experiment shows in a very simple way the usefulness of CAC based on measured effective bandwidth in that two connections with a combined peak cell rate which exceeds the allocated bandwidth can be accepted.

Figure 5.10 shows the results when the above experiment was repeated, but in this case two near constant rate sources were used. The first connection is accepted at approximately 5 s, and because of the high effective bandwidth resulting from this source the second connection is rejected at 20 s. The vertical line at 20 s indicates the sum of measured and declared bandwidths on which the rejection decision was made. The allocated bandwidth for this switchlet port is subsequently increased from 4000 cells/s to 6000 cells/s, by means of the setSL_BW method in the Configure interface of the Divider Server. At approximately 38 s the previously rejected connection is accepted.

5.5.4 Multiple Source, Multiple Virtual Network Experiment

In this experiment two virtual networks, each containing a single switchlet on the same physical switch were created. Each switchlet was limited to have only eight VCIs, and each control architecture had eight traffic generating applications registered with it. For each virtual network these eight applications were evenly distributed across two workstations. The applications were generating the same bursty traffic used in the experiments above. In this case however, the applications would wait for a random period before requesting creation of the connection, and similarly the connection would last for a random period. The random period in both cases were between 0 and 100 s, and the **drand48()** random number generator available on HP workstations was used to produce it.

Figure 5.11 shows the cell arrivals for the two switchlets and overlaid on the

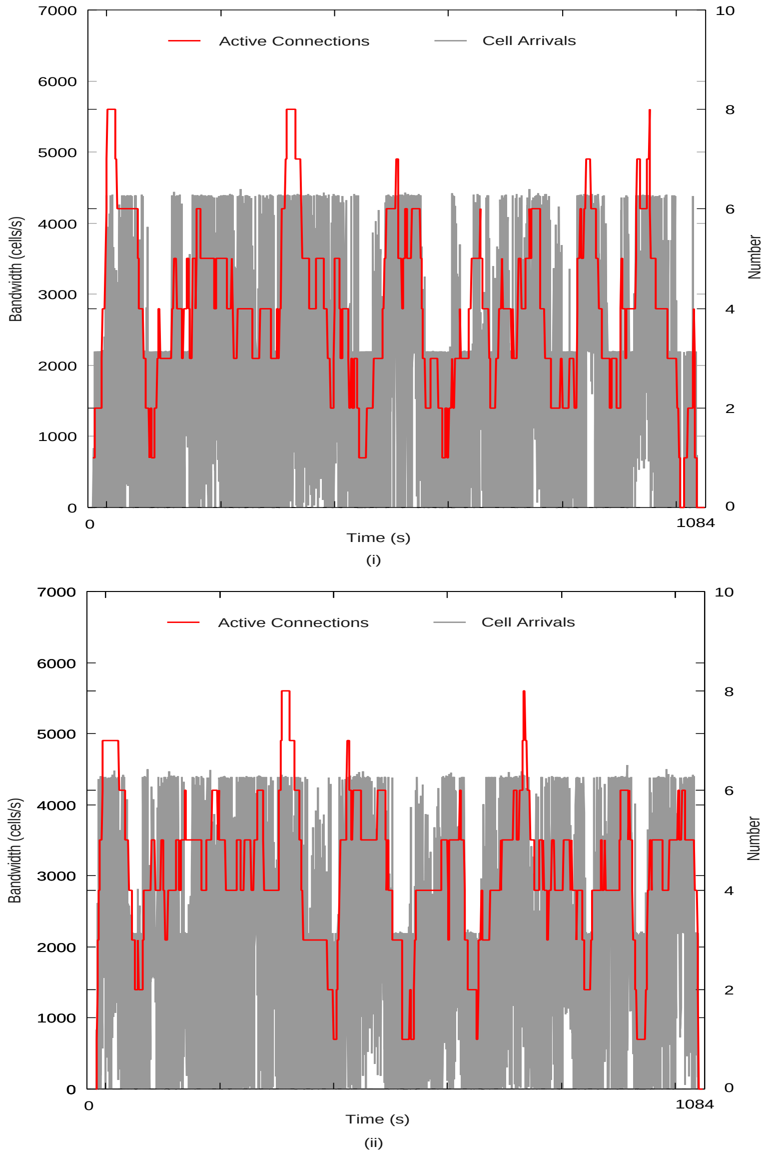

Figure 5.11: Arrivals and number of active connections for (i) Switchlet 0 and (ii) Switchlet 1

same graph (with vertical axis on the right) the number of active connections on the switchlet port. Figure 5.12 shows the cell arrivals for the two switchlets, and the resulting effective bandwidth estimates.

In Figure 5.13 the aggregate arrivals for the switch port and the resulting effective bandwidth estimates are shown. At any moment in time up to 16 connections can be active across the physical switch port. However, all connections

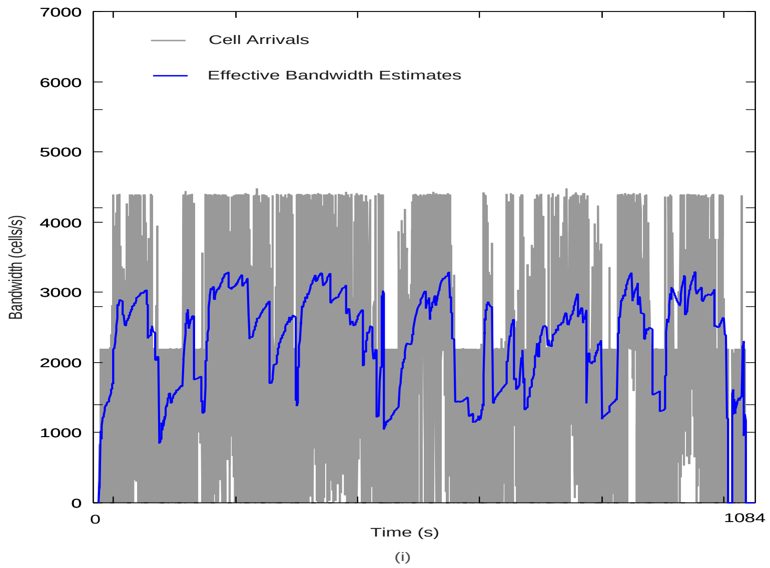

Figure 5.12: Arrivals and effective bandwidth for (i) Switchlet 0 and (ii) Switchlet 1

originate from only two workstations where access to the network is effectively serialised. The result of this is evident in Figure 5.13 where the aggregate cell arrivals are essentially limited to twice the maximum peak value of the sources.

Finally, Figure 5.14 shows the effective bandwidth of the two switchlets, the sum of these switchlet effective bandwidths, as well as the effective bandwidth of the aggregate traffic. Again the significant difference between the sum and

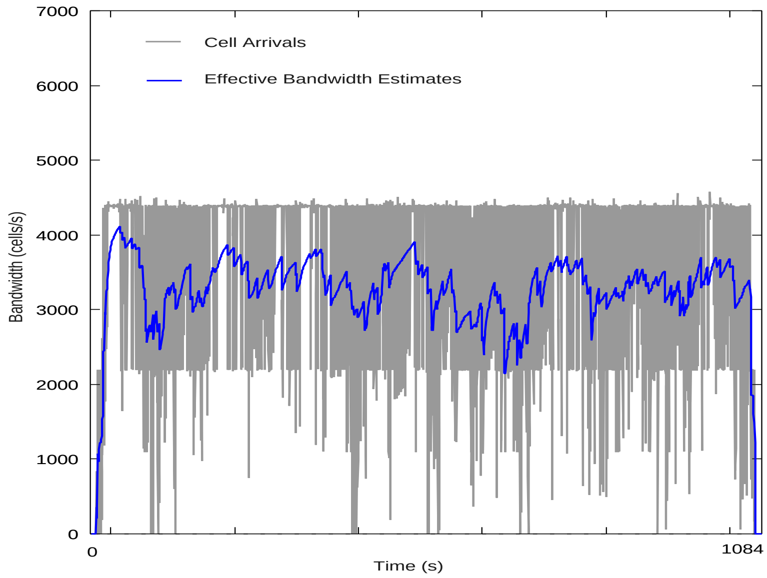

Figure 5.13: Aggregate cell arrivals and effective bandwidth estimates

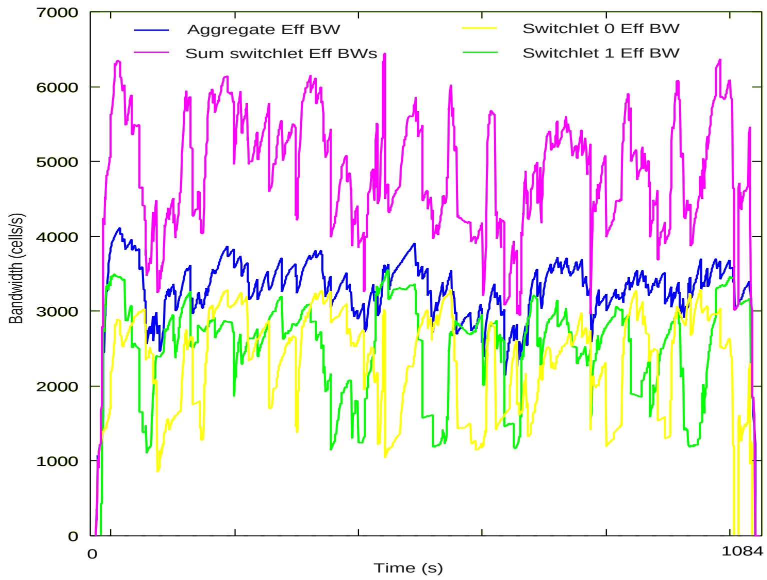

Figure 5.14: Effective bandwidth estimates for the switchlets, their sum and the switch port as a whole

the aggregate effective bandwidths indicate the multiplexing gain which can be exploited by either CAC or VNAC.

5.6 Discussion

As mentioned in the previous section, the purpose of this investigation was to investigate:

- The feasibility of implementing measured effective bandwidth mechanisms on a real switch.
- The use of measured effective bandwidth within the OSSA environment.

While this implementation proves the practical feasibility of this approach some unresolved issues remain:

- In all experiments described above the length of traffic measurements (the block length in Equation 5.8) were arbitrarily chosen to be 200 ms. Results in [Crosby96] indicate that the accuracy of bandwidth estimates depends on the "correct" length of these measurements, which is a function of the traffic characteristics[12]. If a method can be developed to pick the correct block length, it is not clear how this would influence an implementation. For example, unless the block sizes are multiples of some basic block length it would be difficult to implement sampling at arbitrary frequencies.
- The time it takes for estimates to converge to the "correct" value needs further study. In the work presented it was assumed that effective bandwidth estimates were correct at all times. However, in Figure 5.4 for example, the bandwidth estimates take several seconds to reach the correct value. While no general conclusions can be made based on the limited number of experiments conducted, it does highlight a potential problem if the technique is used to effect CAC in an environment where hundreds of connection attempts per second can be expected. The potentially serious nature of simply assuming that estimates are correct is considered in [Grossglauser97b].
- Obtaining and keeping per VCI statistics is not a scalable solution, especially if estimates are to be performed off the physical switch as is the case in the current implementation.

[12] Whilst for a specific traffic flow such an optimal value exist, recent results indicate that the presented estimation mechanism is quite robust to changes to this value [Russel97].

- As explained in Section 2.2, the ASX-100 switch on which these experiments were performed has a simple FIFO output queueing mechanism, which maps well onto the single server paradigm on which the theory is built. While the theory holds for work-conserving scheduling mechanisms, further work is needed before the effective bandwidth approach can be used on real switch architectures with sophisticated queueing and buffer management mechanisms [Rathgeb97, FORE97].

Many of the issues listed above are currently being addressed within the context of the Measure project.

Regarding the applicability of this work to the OSSA: If bandwidth estimates are used for bandwidth management in the virtual network domain, the problem of slow convergence is less severe because of the more relaxed time requirements. One could therefore conclude that in its current form, the measured effective bandwidth approach would be more useful in this domain.

5.7 Summary

This chapter considered the bandwidth management requirements of the OSSA and explored the use of effective bandwidth estimates based on online measurements to satisfy those needs.

The need for admission control at different levels in the OSSA was explained, and specifically it was shown that in some cases the Divider Server entities need to perform a CAC function as part of their policing activities.

The use of measurement based effective bandwidth estimates to perform bandwidth management at the slower virtual network time scale was explained. This enables the moving of bandwidth between virtual networks, and provides a handle on the current usage of network resources to perform virtual network admission control.

An extended implementation of the OSSA in which bandwidth management is performed by means of measurement based effective bandwidth estimates was presented. A number of simple experiments were conducted using this implementation, and the results presented.

While the implementation described in this chapter shows the potential and feasibility of the measured effective bandwidth approach, some issues remain especially when applied to the CAC problem. On the other hand this approach clearly has immediate application in the longer time scale virtual network bandwidth management environment.

Chapter 6

Service Specific Control Architectures

Recognising the existence of several different approaches to control in ATM led to one of the main goals of the work presented in this dissertation: to provide an environment which allows several such control architectures to be simultaneously operational within the same network. The preceding chapters presented an open service support architecture (OSSA) which facilitates such an environment, and presented several proof of concept implementations of key components in the architecture. Not only does the OSSA fulfil the requirements of this main goal, but it also enables a new way of thinking about networks and the ways in which they are operated. This chapter explores one such new idea, enabled by the OSSA, in the form of *Service Specific Control Architectures.*

6.1 Introduction

The conventional approach to control in ATM networks has been to adopt a single monolithic general purpose control architecture [ATM Forum93, ATM Forum96]. This general purpose control architecture then tries to provide the functionality required by all current and anticipated future users of the network.

In contrast, this chapter presents the concept of a Service Specific Control Architecture (SSCA). An SSCA utilises knowledge of the requirements of the applications it serves to control the network. In so doing, it can make better use of network resources and provide a more efficient service. This concept is demonstrated by means of a relatively simple control architecture which has knowledge about the applications it serves and can solve problems which would be very difficult to solve within a general purpose control architecture.

Section 6.2 justifies the approach of SSCAs by comparing it to conventional general purpose control architectures, and by examining its relationship with such general purpose control architectures. A number of example environments, which stand to benefit from SSCAs, are discussed.

In Section 6.3, an example SSCA is examined in detail. The specific service considered is that of multi-party conferencing with high quality continuous media (audio and video). This is a service that is supposed to be ideally suited to the ATM environment, and yet very few (if any) solutions exist that will scale well into the wide area network (WAN) environment. The design and proof of concept implementation of the VideoMan SSCA is presented.

6.2 Motivation

In the traditional approach to control in ATM networks, an application is presented with a single User Network Interface (UNI), through which all communication with a single general purpose control architecture takes place [Stiller95]. As the requirements of users and applications change, modifications are made to the control architecture which are reflected in associated changes to the UNI. While this approach is understandable, and indeed desirable, in order to provide interoperable implementations, it is built on the basic assumption that all applications require the same functionality from the network. Unfortunately, however, this "one size fits all" approach suffers from several drawbacks:

- The vocabulary of interaction across the UNI needs to be sufficiently large to deal with all ways in which users might want to communicate. Dealing with all such user requests in a single control architecture complicates its design and implementation.
- As new applications with new requirements are developed, both the UNI and the control architecture have to change, or alternatively, applications have to adapt to the inadequacies of the control architecture.

- Because of its monolithic nature, any change in either UNI or control architecture, by necessity, means network and/or application downtime, because the old architecture must be globally replaced by the new.
- Any changes to the control architecture have to take place in a synchronised fashion over the whole network.
- The control architecture has to cater for all possible applications in a generic fashion, which leads to inefficient and complex protocols.

In addition to the above shortcomings, a general purpose control architecture, by its very nature, cannot exploit characteristics of certain applications. Furthermore, in some cases the functionality provided by a general purpose control architecture simply does not meet the requirements of some ATM capable devices. In these instances the only viable solution is the use of an SSCA. Before the mechanisms for using SSCAs are considered, a number of example environments that might benefit from this approach are presented.

6.2.1 Example Environments

Video conferencing in an ATM environment can benefit from an SSCA in at least two ways. First, the limited number of active participants in a conferencing session can be exploited to make more efficient use of network resources. This is particularly important in a high quality conferencing session, where network bandwidth can very easily become a scarce resource. Second, manipulation of connections *within* the network, under direct control of the floor control function, can avoid time consuming re-establishment of connections, which would be the only alternative in a conventional ATM signalling environment. The design and implementation of an SSCA for this environment is considered in detail in Section 6.3.

Because of its attractive quality of service (QoS) guarantees, the ATM environment has seen the development of several "custom made" or "service specific" ATM capable devices. In general, these devices have varied processing capabilities, which means that the implementation of a general purpose control stack within the devices is often not feasible. More important, however, is the fact that often the functionality provided by a general purpose control architecture is not sufficient for these devices.

One example of such a device is an ATM to SCSI[1] adapter [i-cubed96]. These devices provide an adaptation facility between a SCSI bus and an ATM network at a very low level. One way of using this device is as a transparent extension of the SCSI bus across an ATM network. This means that commands and data sent across the SCSI bus are converted to ATM cells which are sent across an ATM network. A similar device at the other side of the network applies the inverse translation and converts the ATM cells into SCSI commands and data on a second SCSI bus. Clearly it would be impossible for these devices to use a general purpose control architecture because the connection setup rates required in this environment are much more demanding than in a normal ATM environment. But ATM is the ideal in-band transfer mechanism because of the QoS guarantees that it can provide.

The digital manipulation of high quality video and audio by broadcasters and the movie industry is another example environment which could benefit from the use of an SSCA. Again, the QoS guarantees of ATM make it an ideal transfer mechanism for these applications. A specific problem domain is the editing and playback of digitally stored video and audio clips without excessive copying and duplication of data. This task can best be performed using an SSCA, which can schedule the distribution and delivery of these files across the ATM network according to current loads on particular servers and network nodes.

A different approach to the same problem of audio and video editing in a networked environment would be to have a large number of moderately powerful fileservers each equipped with an ATM adapter and hard disc. Each of these fileservers would then serve as a mini fileserver for a few audio and video files. A number of editing stations connected to the fileserver clustered by means of an ATM network completes the picture. The aim in this approach is that no video and audio files are ever copied, other than their initial storage onto a fileserver. Instead, an SSCA is used which keeps track of the editing of files, and then when clips of different files are "played back" as a single entity, establishes and switches between connections to ensure correct delivery of the final sequence of clips. The SSCA can be viewed as either interacting with, or being part of, the storage system.

The usefulness of the SSCA concept is not limited to the fairly esoteric examples above. For example, when carriers provide telephony services over

[1] Small Computer System Interface (SCSI) is a peer-to-peer protocol for transferring data between up to sixteen devices connected to a common bus.

ATM[2], it might be simpler to use existing telephony signalling, accounting and other software [Modarressi90]. These can be applied to a separate ATM virtual network, rather than re-engineering it to fit standard ATM signalling [ATM Forum94a, ATM Forum96].

Finally, one can envisage the situation where a service provider provides a standards-compliant service, but customises or enhances it to satisfy the special needs of its customers, and to differentiate it from those offered by its competitors. This approach has the advantage of being standards-compliant if interaction with other service providers is required, and also of not being restricted by standards if services are only offered to the provider's own customers. For this example, the real strength of the OSSA environment comes from the possibility that a service provider can lease an empty virtual network and control it by means of its enhanced SSCA. The service provider can thereby provide the customised services to its customers even in places where it does not own the physical infrastructure.

6.2.2 Discussion

Other examples of SSCAs can be found in the areas of mobility and home networking. In almost all cases, as in the examples presented above, the potential usefulness of an SSCA stems from either or both of the following:

- In order to perform some tasks efficiently, it is necessary to manipulate connections within the network based on some service specific knowledge, or a standard control architecture does not provide the required functionality.
- ATM endpoints are not intelligent and are incapable of implementing a normal control stack.

Of course, all the functionality required by these examples could be pushed into a general purpose control architecture such as UNI/PNNI. However, this would only tend to make the already complicated general purpose control architectures even more complex, whilst the functionality is not required by all network users.

One could also argue that the second reason given above will become less important as technology develops; that all end systems will be powerful enough

[2] Such a service might use ATM right to the endpoints (telephones), rather than just using ATM in the backbone.

to contain a complete general purpose signalling system. However, this places a limit on what an end system will be (e.g. a doorbell or an active badge detector) and adds gratuitous complexity to all end systems which greatly changes the dynamics of deployment. Also, end systems tend to live longer than network equipment, and upgrading all end systems to keep up with network changes will not always be feasible.

It is possible to build dedicated ATM networks with special control architectures to achieve some of the same benefits described above. This is not sensible because, for example, an ATM workstation taking part in a video conferencing session could reasonably expect to use its ATM interface for functions other than video conferencing. Similarly, in the fileserver cluster approach to the continuous media editing problem, both fileservers and editing stations might require the use of the same ATM network as a more general purpose network. There is thus a need to be able to run both a general purpose control architecture, as well as one or more SSCAs. The switchlet mechanism proposed in this dissertation solves the problem by facilitating the simultaneous operation of more than one control architecture within the same network and indeed within the same end system. In this manner, it also obviates the need for the hard partitioning solution of a dedicated network approach.

Figure 6.1 depicts the relationship between service specific and general purpose control architectures in an example ATM network. In this network two local area ATM networks are connected by means of an ATM backbone. A general purpose control architecture is assumed to be running on all switches and on all workstations, and is shown with a black circle. An SSCA for the support of continuous media editing provides services for a number of ATM capable fileservers and workstations, and is completely contained within one of the local networks. Finally, a video conferencing control architecture provides services for ATM video equipment and workstations on all three ATM networks.

As indicated, some SSCAs are expected to be contained within private networks, while others are expected to be particularly useful in the wide area. It would be possible to statically partition and configure networks to run different control architectures. However, a more dynamic approach whereby virtual networks with appropriate SSCAs are only created when needed, holds many advantages. For example, in the case of wide area video conferencing, it would be much more sensible to release the resources associated with a conference session when the conference is over. In this manner, unused resources are not wasted, and can in fact be made available to other virtual networks. Such a dynamic

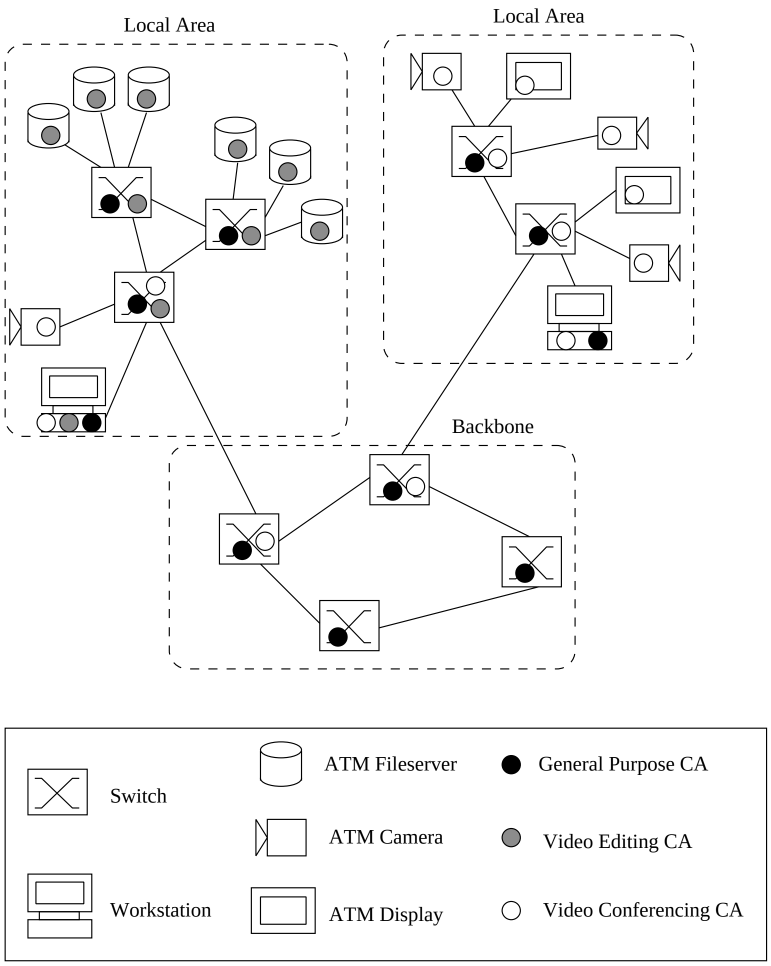

Figure 6.1: Control architectures in a physical network

environment is provided by the OSSA presented earlier in this dissertation. The design and implementation of the VideoMan control architecture detailed below makes use of these dynamic network creation facilities of the OSSA.

6.3 VideoMan - An SSCA for Continuous Media Conferencing

Having motivated the need for and usefulness of SSCAs in the preceding sections, this section explores an example SSCA in detail, and presents a proof of concept implementation. The SSCA presented is called **VideoMan** and provides a control architecture for a continuous media conferencing environment.

A multi-party conferencing environment has several characteristics which can be exploited by an SSCA to provide a scalable solution that efficiently utilises network resources:

- Most participants are data *consumers* for the complete duration of the conference.
- Some participants become audio data *producers* for fairly short periods of time.
- At any particular moment, a limited number of audio data producers should be active in an orderly conference.
- Of all the potential video data producers (all participants), only a limited number will actually be of interest. (For example, the video of the current and previous speakers.)

The VideoMan control architecture drastically reduces the network resources required for video conferencing by creating novel connection structures in the ATM network. Connection structures and endpoints are also manipulated by the control architecture as directed by the *floor control* function to ensure that QoS guarantees can be maintained for all media flows. Floor control is the process of deciding which set of participants should be producers and can be achieved in a variety of ways, such as chaired conferences or audio-triggered conferences [Clark92b, Eriksson94]. The following subsections describe the VideoMan architecture in detail: Section 6.3.1 explains the novel tree structures created in the network, and Section 6.3.2 describes how VideoMan fits into the OSSA. The section ends with a comparison of VideoMan with other approaches to video conferencing.

6.3.1 Gather and Distribution Trees

VideoMan uses the inherent multicasting and inverse multicasting[3] capabilities of ATM switches, as well as application specific knowledge to control the network.

[3]Inverse multicasting is the ability of some switches to map several incoming virtual channel identifiers (VCIs) onto the same outgoing VCI. The architecture is not critically dependent on this capability, but inverse multicasting does allow for a symmetric solution which requires fewer network resources than if it is not available.

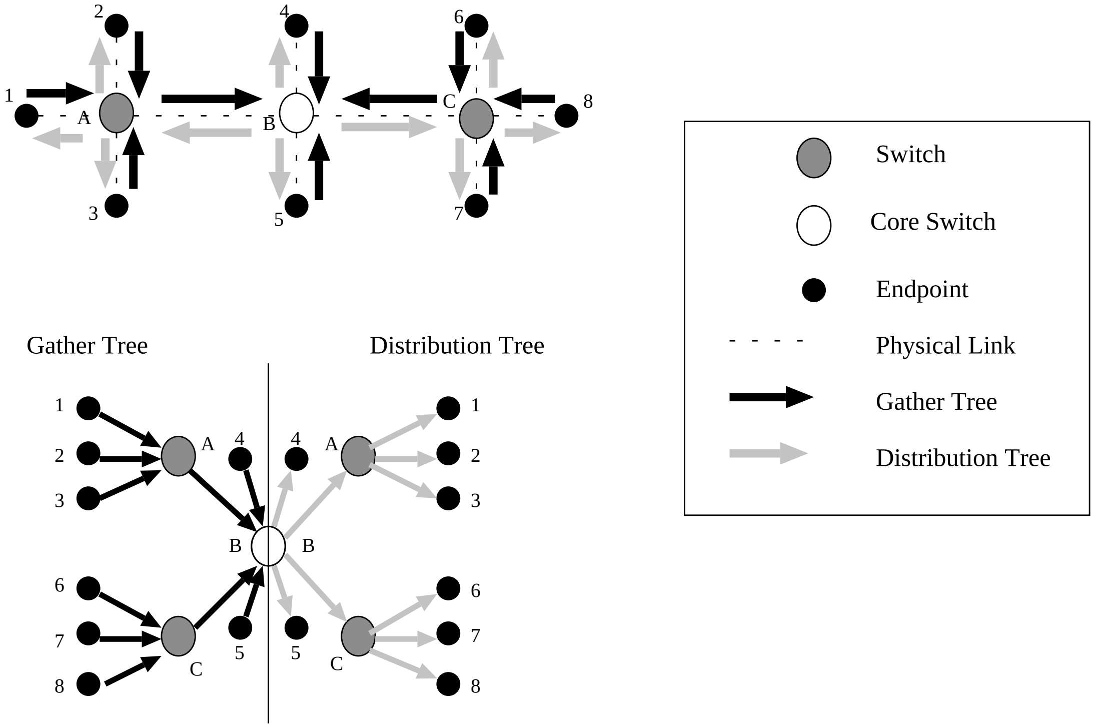

Figure 6.2: Gather and distribution trees for single site media distribution

In particular, the control architecture creates sets of *gather* and *distribution trees* as depicted in the example network of Figure 6.2. For simplicity the figure only shows a single set of tree pairs. The top part of the figure shows the example topology, while the bottom part "unfolds" the network into two mirror images for clarity. Similar tree structures have been proposed to facilitate multicasting in IP based networks [Ballardie93], to distribute utilisation updates in ATM networks [Cidon95] and, more recently, as a generic multipoint-to-multipoint mechanism in the ATM environment [Grossglauser97a]. The interaction required within the control architecture to create the tree pairs is considered in a later section[4].

In Figure 6.2 it is assumed that all endpoints are potentially producers and consumers of data. As indicated, the gather tree uses the inverse multicasting, and the distribution tree the multicasting capabilities of the switches. The final

[4]It is worthwhile to note that if an ATM switch separates the functions of "allocating resources" and "using resources" as described in Section 3.2.1, and further allows different virtual circuits to share the same resources (i.e. police them as a group rather than individually), then it would be possible to create a mesh that would not use excessive switch resources, and yet still provide QoS guarantees under some conditions. Such switches, however, do not currently exist and a different approach is required.

crucial factor in making the concept work is that the control architecture must ensure that, at any moment in time, only one producer is active in a particular set of trees to avoid the interleaving of cells from different producers on the same inverse multicasting VCI. The control architecture can achieve this either by having some control over the endpoints, or by disabling the first branch of all but one endpoint in a particular gather tree. This ability of the control architecture to directly manipulate network resources is a key feature of the VideoMan control architecture, and distinguishes it as an SSCA.

Note that this is not meant to be a general solution to the abstract problem of many-to-many multicast. It is a system which avoids solving the general problem whilst solving the specific problem of the application in hand, namely the distribution of continuous media in a video conference environment. At the same time, the VideoMan resource management provides a functionality similar to the general merging operations in RSVP [Zhang93]. The use of similar tree structures has recently been proposed to solve the general multipoint-to-multipoint problem in ATM [Gauthier96, Grossglauser97a]. Unlike the VideoMan, which requires no changes to physical switches, these approaches both require changes to the ATM data path, and will be discussed in detail in Chapter 7.

The same basic scheme as outlined above can be used to scale the architecture across various sites. A control entity at a particular site will be responsible for creating gather and distribution trees locally. The control entities at different sites cooperate to create distribution trees between them, with the incoming branch from a remote site being connected into a local distribution tree. Floor control is performed by interaction between the control entities at different sites[5]. Looping back a remotely produced stream to its originating site is prevented by obeying the following simple rule: If the active source is within the local site, then patch in the relevant remote distribution tree into the local distribution tree, otherwise disconnect all remote distribution trees.

Figure 6.3 shows a two site example, where endpoint 2 at site II is the current active producer. This means that the remote distribution tree whose source is at switch C at site II is enabled, while the reverse remote distribution tree is disabled. (The inter-site distribution trees in this simple example constitute a single branch each.) Disabling a tree involves only the invalidation of a single VCI table entry at the source of the tree while the bulk of the tree stays intact,

[5]In a chaired conference, this interaction can be either peer-to-peer or hierarchical. It is unlikely that any distributed floor control implementation would be fast enough for voice activated conferencing.

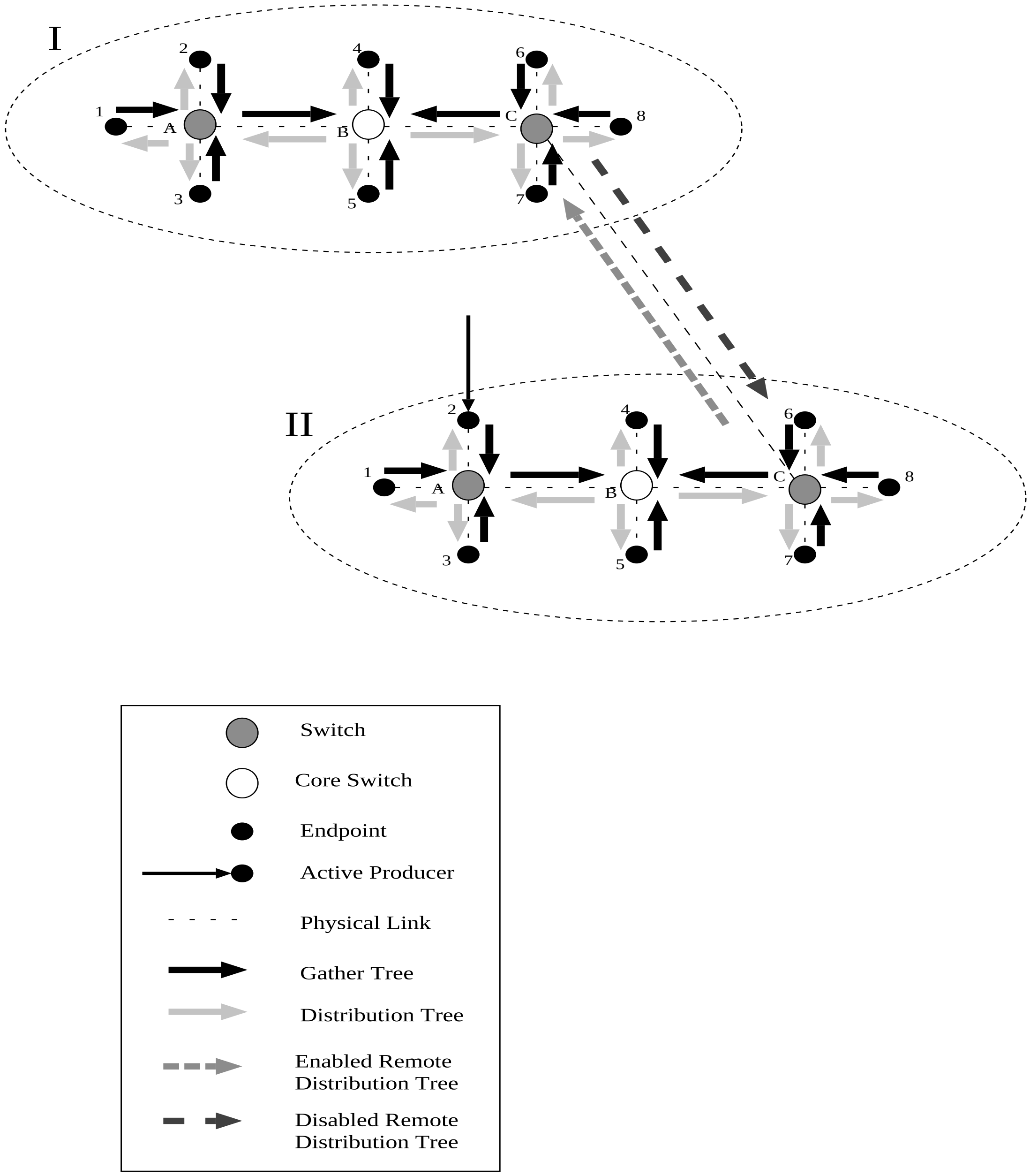

Figure 6.3: Inter-site distribution

i.e. it does not involve the tearing down of a complete end-to-end connection or multicast tree in the traditional sense.

In the discussion above, the function of a single set of trees was described. Clearly, a set of trees will be required for each media stream that needs to be consumed by an application. For example, a conferencing application might require an audio stream from the current speaker, a high quality video stream from the current speaker (main video window), and a lower quality video stream from the previous speaker (secondary video window). Each of these will require the creation of both a gather and a distribution tree.

If switches do not support the inverse multicasting capability, it is always possible to create unicast connections from each producer to the core of the tree, and have the control architecture patch the appropriate unicast connection through to the distribution tree at the appropriate time.

Note that, in the case of a chaired conference, the activation and distribution of audio from the current speaker can be handled by means of an appropriate set of gather/distribution trees. However, when audio triggered floor control is used, all audio sources must by necessity be active all the time. These streams are analysed at a single point by the floor control function to determine which video source should be activated. At the same time, the different audio streams can be mixed into a single combined audio stream which can be sent to audio sinks by means of a distribution tree. The analysing and mixing functions need not be performed by the same entity, in which case a two-branched multicast can be created from each audio source, with one branch going to the floor control function and the other going to the mixing function. If the format of the audio streams can be controlled so that audio samples are sent in self contained cells, a gather tree, rather than several unicast connections, can be used to transport audio streams to the analysing and mixing entities.

It is important to note that the control architecture utilises application specific knowledge in its manipulation of network resources. This is crucial to its correct operation, and would be impossible to achieve using a general purpose control architecture.

6.3.2 VideoMan Within a Virtual Network

The previous subsection explained the tree structures that the VideoMan control architecture maintains within the network. In this subsection, the creation and maintenance of these tree structures is explained within the context of a video conference within the OSSA environment.

Having obtained a virtual network, as described in Section 4.2, a service specific control entity, which is an instantiation of the control architecture managing the virtual network, can start to provide services to applications. Interface references to switchlet control interfaces, returned by the Network Builder, are used to manipulate the virtual network resources. This interaction was depicted in Figure 3.4 on page 41, and Figure 3.6 on page 44.

The interaction between users and the control architecture, and the interaction between the control architecture and switchlets is depicted in Figure 6.4. To simplify the diagram, the interaction with the Network Builder to instantiate and modify the virtual network is not shown. Similarly, the existence of a trader is assumed to allow clients (applications) to find out about servers (the control entity), but is not shown in the figure.

Applications register with the control entity as either producers or consumers of data or both, shown by invocation (1) in Figure 6.4. An application also provides the control architecture with the name of its host or endpoint. This name is then turned into an endpoint address and used by the control architecture to determine whether appropriate switchlets already exist within its virtual network to accommodate the requesting application. If not, appropriate invocations are made on the Network Builder to add switchlet(s) into the virtual network.

For the sake of simplicity it will be assumed that only a single set of gather and distribution trees is required although, as indicated in the previous section, several sets of trees will typically be required.

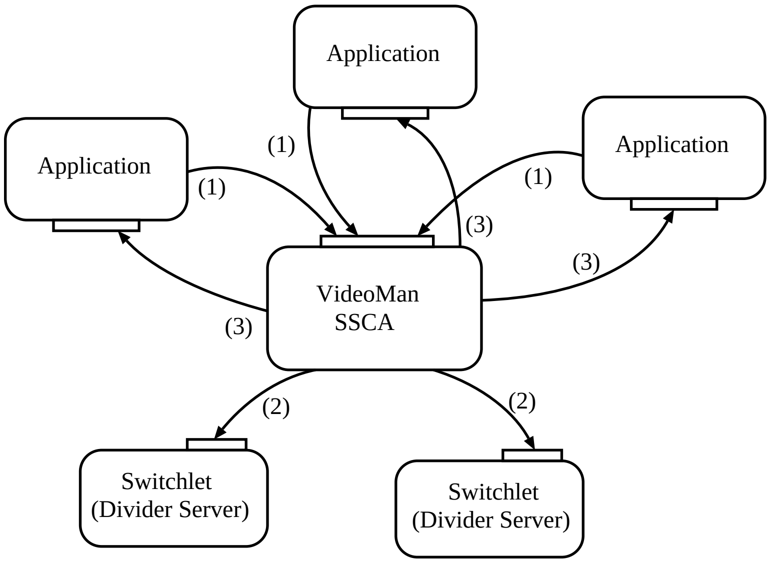

Figure 6.4: Conferencing control architecture

If an application registers as a consumer of data, the control entity determines where in the distribution tree the endpoint should be added, and carries out the appropriate invocations on the switchlet(s) to do so (2). Depending on the situation, appropriate invocations might be adding a branch to a multicast tree, or the creation of a new tree. The control architecture then returns to the application a *service access point* on which it can start receiving[6].

Similarly, when an application registers as a producer of data, the control architecture performs the required invocations on the switchlet control interface(s) to add the endpoint into the gather tree. Additionally, when an application registers as a producer, it is required to provide the control entity with a *control interface* reference. This control interface reference is used by the control entity to grant or revoke the right of a producer to transmit data into the gather tree (3). This is used to ensure that only a single producer is active at any moment in time. In order to protect the conference from misbehaving producers, or to cater for less sophisticated devices, the control entity might want to disable all gather tree endpoints except the branch which ties the current active producer into the tree. Again this is a simple local operation, and the bulk of the gather tree can be kept intact.

6.3.3 Implementation

This section describes a proof of concept implementation of the VideoMan control architecture. The implementation used the same ATM environment as described in Section 4.2.1. The control architecture used the DIMMA framework described in Section 2.2 as a DPE to provide an implementation environment. This is not a requirement however, and the control architecture could have been implemented with any other message passing, or client server implementation[7].

The current implementation creates a single distribution/gather tree pair, which is used to distribute a single video stream from any number of AVA devices (sources), to any number of workstations (sinks). (See Section 2.2.3 for a description of the AVA devices.) This can be trivially extended to create any

[6]In a DPE which supports both stream and operational interfaces, a service access point would correspond to a bound stream interface. In the proof of concept implementation in the next section, a service access point is simply the value of a VPI/VCI pair on which the application can start receiving.

[7]Interaction with the dynamic virtual network world would still require DIMMA interaction, however, this is provided in the form of a set of libraries.

number of trees in a fully functional implementation. The control architecture ensures that only one of the AVA devices is active within the gather tree by directly controlling the device (in fact, a manager of the device). The operation of this system is described below.

At startup, the control architecture obtains the current switch topology by invoking the *getSwitchTopology* method on the `Topology` interface of the Network Builder. Based on this information it selects a core switch for the video conference. The current implementation simply selects the switch with the most neighbours. The control architecture then proceeds to determine the shortest path from the core switch to all other switches. A breadth first search algorithm is used to calculate shortest paths. At this point the control architecture invokes *createNamedNetwork* on the `Dynamic` Network Builder interface, to create an "empty" virtual network, and awaits requests from applications to join the video conference.

The VideoMan control architecture exports a single `ConferenceUser` interface with the following methods:

- *registerAppl*: Invoked by application to join a conference. The control architecture passes back to the caller a conference-unique application identifier which is used in all subsequent invocations.
- *unregisterAppl*: Signals to the control architecture the intention of this application to leave the conference. The control architecture uses this to prune any connections associated with this application from the gather and distribution trees.
- *sinkJoin*: Joins a conference as a sink. The control architecture is also provided with a reference to a `MediacontrolSink` interface on the application (see below).
- *sourceJoin*: Joins a conference as a source. The control architecture is also provided with a reference to a `MediacontrolSource` interface on the application (see below).

The `MediacontrolSink` and `MediacontrolSource` interfaces provide the following methods:

- `MediacontrolSink`: The methods provided by this interface allow the control architecture to control consumers of information, for example based on who the current active speaker is:
  - *receiveOn*: Provides the application with a service access point (a VPI/VCI pair) on which it can start sending.
  - *stopReceiving*: Instructs the application to stop receiving on a particular service access point.
- `MediacontrolSource`: The methods provided by this interface are used by the floor control function of the control architecture to ensure that a single producer is active in any tree pair:
  - *sendOn*: Instructs the application to start sending on a particular service access point.
  - *stopSending*: Stops application from sending on particular service access point.

The current implementation provides a simple test "floor control function", whereby the control architecture sequentially steps through the list of producers at four second intervals, and selects one to be the current active producer. It invokes *stopSending* on the `MediacontrolSource` interface of the current active producer, and then invokes *sendOn* on the `MediacontrolSource` interface of the newly selected producer. In this proof of concept implementation the time it takes to switch to a new producer is completely dominated by the communication overhead involved with making the two remote procedure calls, which varies between 2 and 5 ms for the pair, depending on the speed of the workstations involved.

The emphasis of this implementation was to develop a working prototype which illustrates the power of an SSCA. In particular it was shown that a fairly simple SSCA can create and manipulate novel connection structures in the network, thereby greatly decreasing the network resources required for video conferencing in an ATM environment. Since application specific knowledge is used to do this, it is impossible to do the same with a general purpose control architecture. These points are illustrated in the following subsection which compares VideoMan with other possible video conferencing solutions.

6.3.4 Comparison

This section compares the VideoMan approach with more conventional video conferencing solutions which can be engineered using a general purpose control architecture. The comparison is made on the basis of resource requirements, response time and scalability. The comparison in this section is of an architectural nature, and a discussion of related work is postponed till Chapter 7.

To simplify the comparison, it will be assumed that similar (or identical) floor control functionality is available in all cases. It is also assumed that a single high quality video stream of the current speaker needs to be displayed to each participant in a multi member video conference.

- At one end of the spectrum, connections for the video stream can be created on-demand as dictated by the floor control function. A multicast connection is created from the current speaker to all other participants. When the speaker changes, a new multicast connection is created, and the previous one torn down. This approach is very efficient in terms of network resources, in that a single multicast connection is created. (If an old connection is only dismantled after the new one is created, there will often be more than one connection present in the network.) However, the time taken to create and dismantle the multicast connections will mean that changing between speakers will be unacceptably slow[8]. Since this problem will be exaggerated by larger networks and more participants, this solution does not scale well[9].

- At the opposite end of the scale, a full mesh can be created between all participants, and the floor control function can simply inform participants on which connections to send or receive. In practice, a multicast connection will therefore be required from each member to all other members. Because a multicast connection is required for each participant, this solution uses a large amount of network resources, (N multicast connections are required for N participants), and therefore does not scale well. Since all connections

[8]Commercially available control architectures report connection setup times of several milliseconds per connection per switch [FORE95a, Bellcore97, Niehaus97]. Since current specifications [ATM Forum94a, ATM Forum94b] construct point-to-multipoint connections one endpoint at a time, the time taken to set up a multicast tree may be significant.

[9]An additional problem with this approach is that a multicast connection setup might fail halfway through a speaker change because of insufficient resources. This will greatly increase the complexity of failure modes in the conference software, or leave the conference in an undefined state.

are left in place once they are created, the response time of this solution should be very good.

- One way of avoiding the problems associated with the above approaches is to make use of a special hardware unit which is often called a multipoint control unit (MCU) [Clark92b]. In this solution, a unicast connection is created from each participant to the MCU and a single multicast connection is created from the MCU to all participants. Under direction of the floor control function, the MCU switches the appropriate unicast connection from the current speaker through to the multicast connection[10]. The number of connections are reduced to one multicast and N unicast connections in the worst case, which means that fewer network resources are required. (Worst case resource usage will occur in the switch directly connected to the MCU.) Switching between unicast connections should be very efficient, so good response times can be expected. Correct placement of the MCU can, however, be problematic and might lead to network resources being tied down in a remote part of the network, i.e. physically removed from where the conference is taking place.

- The VideoMan approach is logically similar to the MCU solution presented above. However, it requires fewer network resources and is more flexible: The core switch performs a similar function to that provided by the MCU, but no additional hardware is needed. Since the core switch is a logical entity (in the sense that any switch can be the core switch), the problem of correct placement of the MCU is alleviated. If inverse multicasting can be used, only two multicast connections (a gather and a distribution tree) are required, which significantly reduces the network resources required and therefore improves scalability. (If inverse multicasting cannot be used, the resource requirements are the same as that of the MCU approach.) To switch between speakers involves disabling one source and activating another and is therefore very efficient.

Although the above comparison rightly shows the similarity in functionality provided by the MCU approach and VideoMan, the two are in fact radically different. The MCU solution is probably the best that can be achieved using a general purpose monolithic control architecture. In contrast, VideoMan presents an idea of what is possible with an SSCA which utilises service specific knowledge "inside" the network.

[10] This is the solution adopted by the manufacturers of the ATM video adapters that were used in the implementation discussed in the previous section [Nicolaou97].

6.4 Conclusion

This chapter has motivated the use of service specific control architectures in an ATM environment. Although the needs of many applications can be met by a general purpose control architecture, several example environments have been presented in which this is not the case. The requirements of these environments can only be met when the control architecture can utilise service specific knowledge to manipulate connections within the network. By its very nature, this is something that a general purpose control architecture cannot do.

An example implementation of an SSCA was presented. The VideoMan SSCA allows the creation of scalable video conferences by utilising the dynamics of user interaction in a conference to manipulate network resources and to manage communication.

It is not difficult to imagine a fully fledged video conferencing service based on the VideoMan concepts. At the same time, the VideoMan implementation is really more of an *application* specific control architecture, rather than a *service* specific one. This leads to the observation that within a dynamic virtual network environment, like the OSSA, creation of a virtual network happens over the same time scales at which large scale distributed applications are deployed.

A possible criticism of the concept of SSCAs in general is that one ends up developing a great deal of mechanism which is used only in particular services when it might be of more general use. For example, the VideoMan resource management could be used in many multipoint communication environments. Such functionality might find its way into a general purpose control architecture, but it is not clear how such functionality would be developed in the first place without an OSSA environment.

SSCAs should also not be seen as a limited set of controls for a pre-defined set of services, but rather as an enabling technology which will facilitate the development of new applications and services in a natural way. Indeed, it is the thesis of this dissertation that the OSSA and service specific control architectures are the enabling technologies which will allow an openness in the ATM environment similar to that of the Internet, without compromising the attractive features of ATM.

Chapter 7

Related Work

This chapter considers related work that is of interest either because it is comparable with, or because it is applicable to, the work presented in this dissertation. Each of the following sections discusses a particular contribution or group of contributions, and is more or less self-contained.

7.1 Control Architectures

7.1.1 Standards Based ATM Control Architectures

ATM control architectures are being standardised by bodies such as the ITU (International Telecommunications Union) and the ATM Forum. The work of the ATM Forum is to some extent based on existing ITU standards with extensions and modifications agreed upon by member companies in a process which has a significantly shorter turnaround time compared to the ITU. This section will therefore primarily focus on the work of the ATM Forum.

The relevant work of the ATM Forum is embodied in the various User Network Interface (UNI) and Network Network Interface (NNI) specifications [ATM Forum93, ATM Forum94a, ATM Forum94b, ATM Forum96]. To a large extent these specifications are derived from those that were used in narrowband ISDN and earlier systems [Modarressi90]. The specifications have been and are still being extended as new services require new functionality, mainly by adding new *information elements* to signalling messages. Indeed it is this "one-size-fits-all" approach which is the main point of criticism against the standards based proposals. A selective overview of the ATM Forum UNI/NNI control architecture is described below. The main points to note are that this control architecture is essentially:

- closed and monolithic,
- not dynamically extensible,
- complex and becoming more complex.

7.1.1.1 User-Network Interface

The basic signalling functionality provided by the UNI-3.1 ATM Forum specification is:

- point-to-point and point-to-multipoint switched connections, with symmetric or asymmetric bandwidth requirements,
- different classes of transport services,
- protocol messages, information elements and procedures to support the above.

Point-to-multipoint connections can only be established from the root (i.e. the source) and leaves (sinks) can only be added one at a time. Although QoS parameters can be specified when a connection is established, no negotiation of QoS parameters can take place. The specification also provides functionality for address registration via the UNI.

The above functionality is being extended in UNI-4.0 which adds support for:

- leaf initiated join to multicast connections,
- group addressing and anycast connections: endpoints would register as being members of a group, and when a connection setup is requested to the group address, a point-to-point connection will be established to one of the members of the group,
- negotiation of QoS parameters during connection setup.

7.1.1.2 Network-Network Interface

The latest Private Network-Network Interface (PNNI) specification [ATM Forum96], specifies both an NNI signalling protocol and a routing protocol.

Signalling performed across the UNI interface is mapped onto NNI signalling while the request is forwarded through the network. A re-mapping to UNI signalling happens when the request reaches the switch connected to the final destination.

Routing information is required for both the signalling messages and the connections set up by signalling messages. In addition, because ATM can provide QoS guarantees to connections, the current state of switches and links should be taken into account when a route to the destination is selected. The final acceptance (or not) of a connection will take place based on the local decision of a connection acceptance control function at each switch. However, the aim of QoS-aware routing is to select a path which is likely to succeed based on available QoS information.

To do this, QoS states are flooded through the network together with topology and other information. Because CAC is a local switch (and switch dependent) function, a generic CAC function is used during the routing process.

A particular aim of the PNNI work was that it be scalable to very large networks. As such, a hierarchical network model is used whereby switches are arranged into peer groups, each of which selects one switch to be the peer group leader (PGL). Topology and QoS state information is flooded through the peer group and aggregated by the PGL. At the next level up in the hierarchy, the PGL is a logical group node (LGN) which together with other LGNs forms another peer group. This peer group selects one node as the PGL and the whole process repeats. In this fashion each node in the network has complete (aggregated) knowledge of the network.

7.1.2 IETF Control Architectures

7.1.2.1 IP-over-ATM

In the IP-over-ATM working group of the IETF, the basic approach has been to treat ATM as another link layer technology on which to support IP. As such, the ATM offering of the ATM Forum (including its control architecture) was taken as a given, and significant effort was spent in making the two rather disparate approaches work together [Laubach94, Heinanen93, Luciani96, Heinanen94]. While this approach is understandable in terms of the philosophy of the IETF, it is at best a non ideal solution. For example, because ATM is a fully fledged networking technology in own right, a large amount of unnecessary duplication takes place.

The key observation here is that the IETF work has not contributed a new control architecture; rather an additional layer above the ATM Forum control architecture was introduced. This work will therefore not be considered further.

7.1.2.2 IP Switching

IP switching, invented by Ipsilon [Ipsilon96], strays from the conventional IP style in that an IP based control architecture for ATM is used, in an approach which is philosophically close to the open switch control approach described in this dissertation. All conventional ATM control is abandoned in favour of a simple hop-by-hop control protocol and ATM is simply used for its attractive data transfer capabilities.

With IP switching, an ATM switch is effectively turned into an IP router by connecting a general purpose workstation (with routing software), to one of the ports of the ATM switch. Well known default VCIs on all other switch ports are uniquely mapped onto the workstation port, which allows the workstation to route (in software) IP packets between different switch ports.

To actually make use of the data transfer capabilities of the switch, the above default behaviour is extended as follows: In addition to routing IP packets that it receives, the workstation monitors all IP "flows" going through the router. A flow is defined as a sequence of packets going from a particular source to a particular destination. If a "stable" flow is identified, the router can request the upstream node to send this flow on a separate VCI. At this moment, packets are still being routed in software by the workstation. If the downstream node from the router also identifies the flow, and requests the router to send it on a separate VCI, the router has two unique VCIs associated with the flow, which is still being routed in software. Control software on the workstation can now push these two VCIs down into the switch, to form a switched connection, so that the flow is no longer routed in software [Newman96b, Newman96c, Newman96a][1].

7.1.2.3 RSVP

The increasing use of applications such as audio and video has led to the realisation by the IETF community that the best effort model provided by IP needs changing [Braden94]. As in the ATM environment, resource reservation coupled with admission control is also seen as the way in which this can be realised in the network and RSVP is proposed as the resource setup protocol (or the control architecture in the terminology used in this document) [Zhang93].

Because ATM is one of the networking technologies which can actually provide the kind of QoS guarantees required, a mapping of RSVP onto ATM is being proposed [Berson96]. Unfortunately, this proposal follows earlier IP-over-ATM work in that the standard ATM control architecture is assumed. An approach in which RSVP is used as *the* control architecture on an ATM network would appear to be a better solution.

7.1.3 Other Control Architectures

7.1.3.1 OPENET

The development of the OPENET architecture [Cidon95], was motivated by the lack of performance optimisation, limited functionality and lack of openness for future extensions, identified within the ATM Forum's PNNI architecture. As

[1]It should be noted that a competing set of proposals on a related concept, called "Tag-switching" has been proposed to the IETF community by Cisco [Rekhter96, Davie96]. The main difference between the proposals would appear to be when, and how, IP packets are selected for switching. In the Tag-switching approach labelling is done based on topology information, while in IP switching only stable flows are switched. More recently, interest in these approaches led to the creation of an IETF working group on *Multiprotocol Label Switching* (MPLS) [Callon97]. This working group aims to standardise approaches such as IP and Tag-switching into a framework which is neither "network layer" nor "link layer" specific, i.e. it is meant to apply to protocols other than IP and switching technologies other than ATM.

such, the OPENET architecture is fairly close to a PNNI implementation with some efficiency optimisations. One such optimisation is the use of an ATM level spanning tree with unicell messages for utilisation updates. Again in the interest of efficiency, the current implementation is a kernel based implementation in the Solaris multi-threaded kernel. (Such a non-portable approach seems to contradict the emphasis on openness and extensibility).

7.1.3.2 Bell Labs DPCA

A Distributed Call Processing Architecture (DPCA) is proposed in [Veeraraghavan97]. Similar to OPENET, this architecture is an extension to the standard PNNI proposal, while the possibility of a CORBA based implementation is mentioned. DPCA is a layered architecture with separate entities responsible for switch resources, connections and calls. An interesting feature of the architecture is the existence of *application dependent route servers*. These entities can be used to route connections via special devices in the network, such as an application layer bridge which performs format conversion on a video stream. Such a route server could also be used to transparently connect endpoints to a hardware MCU in a video conference environment, as was described in Chapter 6. The DPCA also envisages exploiting parallel processing during connection setup, by having a single connection server interacting with several switch resource servers. A similar functionality is provided by the control architectures proposed in [Bloem95, Lazar95, Rooney97a].

7.1.3.3 TINA

The Telecommunications Information Networking Architecture Consortium (TINA-C) is a group of network operators and telecommunications and computer equipment suppliers, who aim to define an “open” architecture for telecommunication services [Nilson95]. The architecture is based on distributed computing and object oriented concepts and specifically considers broadband and multimedia networks. The consortium further aims to use applicable standards in telecommunications and computing, such as Open Distributed Processing (ODP), Intelligent Networks (IN), Telecommunication Management Networks (TMN), and Common Object Request Broker Architecture (CORBA).

A specific aim of the architecture is to apply partitioning and layering principles so as to separate services from the network and computing infrastructure and resources [Minetti96]. The connection management architecture (CMA), for example, has at least two technology independent layers, followed by at least three technology dependent layers [Bloem95].

In terms of engineering it is not clear whether TINA is actually meant to replace some network services (e.g. replace signalling in an ATM network with their connection management architecture), or whether existing services are meant to be encapsulated in DPE-like services. The exact nature of the DPE's kernel transport network, and whether it might be part of the managed network is not evident either.

7.1.3.4 ReTINA

A TINA related project called ReTINA aims to produce an industrial strength TINA compliant DPE [Bosco96]. Specific aims of the project include fine grained resource control to meet QoS guarantees, as well as the *encapsulation* of several communication protocols (including signalling). It would therefore appear that (at least as far as ReTINA is concerned), the network is to some extent still considered a closed entity and DPE concepts are not applied "within" the network.

7.1.3.5 Xbind

In addressing the lack of multimedia support in the standard ATM control plane, Lazar et al [Lazar96a] have also (like TINA) developed a layered model for networking. Their extended reference model consists of three levels, namely broadband network, multimedia network and service and application network. The broadband network is considered to include the physical networking equipment, and provides a QoS abstraction to the multimedia network. This allows the multimedia network to perform control and resource management and it therefore is (or contains) a control architecture. (Note that they use the term *binding architecture* probably because of a DPE based implementation [Lazar95]). The multimedia network layer in turn provides abstractions of its internal services to the service and applications network. The latter can then dynamically create new services from these abstractions.

The QoS abstractions of the broadband network are embodied in what is called the *binding information base* (BIB). They propose that this (fairly high level) abstraction be standardised, and postulate that the *binding algorithms* that operate on these states will be sufficient to provide and differentiate all envisaged services. Binding algorithms are not subject to standardisation and several proprietary algorithms can operate simultaneously. At first glance this might appear to provide similar functionality to the OSSA, however the provided functionality is in fact radically different:

- Xbind allows different *algorithms*, all conforming to the *same* model, to operate simultaneously within the same control architecture. In contrast the OSSA allows different *control architectures*, developed according to *different* models, to operate simultaneously in the same physical network. (Indeed, one of the control architectures that the OSSA could accommodate is Xbind.) It is clearly not possible to reduce a complex control architecture (such as the ATM Forum's PNNI) to a mere set of algorithms[2].

The above fundamental difference is also evident when the Xbind video conferencing service [Lazar96a] is compared with the VideoMan SSCA. The Xbind video conferencing service (services and applications layer) utilises the services of the control architecture (in the multimedia network layer) and is therefore limited to the functionality provided by this control architecture. Except for the fact that both service and control architecture are now built according to the same model (and using the same object oriented distributed technology), this is not much different from the way in which a video conferencing service would be built using standard control architectures. In contrast, the VideoMan is a control architecture in own right and since its operation is limited to its own virtual network, which is isolated from other virtual networks, it is allowed to directly manipulate the network resources that were allocated to it. This means that it can make use of service specific knowledge to, for example, optimise its usage of network resources.

Finally, it is not clear how in the Xbind model the network can be protected against misbehaving algorithms. This will have serious implications for the use of third party (untrusted) software in a Xbind network. In contrast, the partitioning enforced by the Divider in the OSSA enables untrusted control architectures to operate in a network, without jeopardising the integrity of the network as a whole.

[2] While PNNI contains various algorithms, it also consists of transport mechanisms, message formats, timers etc. In short, PNNI was developed to fit a different model from that provided by Xbind.

7.1.3.6 Hollowman

The Hollowman is an OSSA-aware control architecture based on DPE principles [Rooney97a]. The Hollowman exports source and sink service offers to a trader, and a connection manager can then "reify" these offers into network resources during connection establishment[3]. The Hollowman was the first control architecture to make use of the OSSA. Although it is a fully functional control architecture in its own right, the Hollowman can also be used as a set of building blocks with which to create DPE based control architectures, e.g. service specific control architectures [Bos97][4].

The Hollowman also provides the concept of a *call closure* [Rooney97b]. A call closure is a mechanism which allows a certain behaviour to be associated with a call or group of connections. This behaviour can be specified by a user which means that application specific knowledge might be taken into account.

7.1.3.7 The Warren

The ATM Warren [Greaves97] is another example of the use of the data transfer capabilities of ATM, but where the use of a standard control architecture is not appropriate. In the Warren approach, ATM is used as a connection mechanism between all kinds of household appliances e.g. speakers, TV, HiFi, doorbell etc. These appliances are unlikely to ever be able to economically implement a standard signalling stack. The Warren approach to this problem is to specify a special (proprietary) control protocol between a single Warren controller and all Warren devices including switches. The Warren controller could then engage in standard signalling on behalf of Warren devices when required. If integrated with an OSSA environment, the Warren control architecture might be considered a Service Specific Control Architecture.

7.1.3.8 UNITE

UNITE is a lightweight signalling protocol for ATM [Hjalmtysson97]. As is the case with many of the non-standard control architectures, this work was moti-

[3]The term "reify" is used in favour of "bind" by [Rooney97a], due to the overloaded use of the latter.

[4]The VideoMan service specific control architecture presented in Chapter 6 did not use these building blocks, but was independently developed by the author.

vated by the realisation that it is the complexity of standard control architectures which leads to their poor performance. In UNITE this problem is addressed by separating QoS management from the basic connectivity functions required. Connection setup in UNITE consists of a micro-setup phase which establishes basic connectivity, followed by an optional QoS request phase. The micro-setup phase is extremely simple and lightweight, so that it could potentially be performed in hardware. The reasoning is that many connections are of a best effort type anyway, and would not need the second QoS request phase. Another unique feature of the UNITE approach is that the QoS request phase, if present, is performed in-band, or at least the request is carried in-band. Cells belonging to a (multi-cell) QoS request are marked as such in the Payload Type Indicator (PTI) of the cell header, so that the cells can be deflected to the switch controller. Carrying QoS setup messages in-band has several potential advantages: Resources such as bandwidth, used during the QoS setup phase, are the default values allocated during the micro-setup phase. This means that QoS setup does not use any of the shared resources for the micro-setup phase and therefore does not influence the common case. The multicasting capability of switches can also be exploited to perform parts of the QoS phase in parallel.

The UNITE approach can potentially reduce the setup time for best effort traffic to the point where the setup phase is as cheap as routing in a connectionless network. It still suffers from two problems however:

- It still considers ATM an end-to-end technology in which an end-to-end connection is required before data can be transfered (end-to-end)[5].
- It presupposes that the connectivity offered by the micro-setup phase will satisfy all needs.

The first problem will essentially disappear if the micro-setup phase can be made cheap enough, i.e. no more expensive than routing a packet through an IP network. The second is more serious however, and can only be addressed by approaches such as the OSSA environment introduced in this dissertation.

[5]In UNITE, it is envisaged that data can be sent by the source before an end-to-end connection exists, if the switches involved support per-VC queueing. However, an end-to-end connection setup is still required before such data will be able to reach the destination. This is in contrast to the IP Switching approach, where data can reach the destination without an end-to-end connection being established.

7.1.4 Discussion

The most noteworthy observation relating to the above discussion of control architectures is the simple fact that there are so many of them. Indeed it was this fact which served as a major motivation for the development of the switchlet concept [van der Merwe97b]. Some of the control architectures mentioned above are experimental, and it is indeed unlikely that all will survive. However, the control architectures offer different functionality for different environments which cannot be accommodated in a single control architecture.

The main emphasis of many of the non-standard control architectures discussed in Section 7.1.3, has been to make *one* general purpose control architecture more open and extensible, by various means. Although valuable in itself, this is completely different from the work presented in this dissertation, which aims to open up control at a more fundamental level, by allowing *multiple* complete control architectures to operate within an ATM network.

Being limited to a single control architecture also means that other approaches by necessity have to be *general purpose* in nature. Again this is fundamentally different from the work discussed in Chapter 6, where the service specific control architecture concept [van der Merwe97a], allows control architectures to utilise service (and application) specific knowledge to optimise the usage of network resources.

A problem with some of the proposed alternative control architectures is the level of abstraction which is introduced. The approach taken in the OSSA was to make abstractions at the lowest possible level. This is in contrast to, for example, the Xbind approach which has several layers of abstraction. Such high level abstractions inevitably lead to some restrictions. For example, the Xbind environment can only accommodate different *algorithms* to offer different services. This is much more limiting than the OSSA environment which can utilise different *control architectures* to offer alternative services.

7.2 Open Switch Control

Of necessity, opening up control in an ATM network requires an open switch control interface which allows third parties to develop software to control a switch. This section considers two approaches to the problem and compares it with the

Ariel interface presented in Chapter 3.

7.2.1 General Switch Management Protocol

The general switch management protocol (GSMP) defined by Ipsilon as part of their IP switching offering, is an open switch control interface [Newman96a]. IP switching is an IPv4 implementation[6]. GSMP closely matches a subset of the functionality provided by Ariel, but has a much simpler notion of QoS[7].

The fact that GSMP was purpose built for one control architecture (IPv4 IP switching) is evident in the functions it supports (and does not support). For example, GSMP has a message to find out if a connection has recently carried any traffic. This information is clearly required by a control architecture in which connections are not explicitly established, like IP switching. The "general" in GSMP therefore means an interface (for a *specific* control architecture) that can be implemented on any general *switch*. In contrast, the OSSA requires an interface which can be used by any general *control architecture* to control any general *switch* or *switchlet*. The Ariel interface is a first attempt at achieving this.

7.2.2 Comet Switch Control

In early work on the Xbind project [Lazar95], the only switch control interface provided to third parties was through the virtual switch and virtual link abstractions. Furthermore, the DPE providing this abstraction was considered to be tightly integrated with the switch and in fact executing on the switch control processor in most cases.

This heritage is unfortunately evident in the open switch control API that was made public at some later date [Lazar96b]. Other than this criticism, the

[6]IPv4 (IP version 4) is the currently deployed version of IP. A new version, IPv6 (IP version 6), is currently being developed which has, amongst other things, support for network flows (or connections).

[7]In fact, since no QoS issues were taken into account for the initial OSSA implementation, GSMP is one of the Ariel impersonations which the current Prospero implementation can use. (See Section 4.1.) More recently Ipsilon has proposed their own QoS enhancements to GSMP [Newman97a]. While details about this approach is not available, it assumes a class-based queueing switch model and further assumes that CAC will be done by the switch, rather than the control architecture.

Comet API seems to offer similar functionality to the Ariel interface but contains no methods for obtaining switch statistics or for receiving alarms.

More recently the Comet group proposed QoS extensions to the GSMP protocol [Adam97]. These extensions provide the means for selecting scheduling and buffer management mechanisms, for setting various QoS constraints, and for obtaining QoS related measurements. The proposed extensions are unfortunately tightly coupled to their model of switch capacity, which involves the notion of a *schedulable region*. This concept and the various problems associated with it are discussed in Section 7.7.

7.2.3 Discussion

The need for an open switch control interface was partially satisfied by the GSMP specification. This has been proved by the use of GSMP in some of the work presented here and by the work of Comet group and others. The fact that GSMP was built to satisfy a single control architecture is however a serious limitation and it is clear that a unifying framework for the handling of QoS issues is still proving evasive. Furthermore, none of the proposed solutions has yet addressed in a satisfactory manner the influence of switch functions such as per-VC queueing, partial or early packet discard etc. Further work is therefore needed in this area.

7.3 Virtual Networks/Services

A number of different "virtual network" offerings are briefly considered below. None of these provide a comparable service to the OSSA's Virtual Network Service, but a discussion is useful to establish a broader context.

7.3.1 VPN in Legacy Networks

In the telecommunications world virtual private networks (VPNs) have been a service offering since the mid eighties [Hall94]. A loose definition of a VPN in this context would be a network with a logically closed user group implemented over a public (switched) network. Initial offerings were limited to telephony services such as virtual PABXs (private automatic branch exchanges) with features such as closed user groups, private numbering and call screening. These voice services were soon followed by data VPNs, mostly based on X.25 technology.

By and large these virtual networks are realised by allowing clever address mappings over which users can exercise some control. For example, users can modify private numbers in a virtual PABX service. Similarly, X.25 based virtual networks can provide features such as closed user groups with access control and call barring under control of the customer.

Although VPNs created in this fashion give every indication of being a dedicated virtual network as far as the customer is concerned, network resources are not necessarily dedicated to the VPN [Garrahan93].

7.3.2 The Internet

VPNs are also found in the Internet environment. Again, the aim is to provide a closed (private) network using the facilities of a single shared network, in this case the Internet. As such, the emphasis in this environment is very much on providing secure data transfer. The work of the IP security group provides the framework in which VPNs are offered [Atkinson96]. Current Internet VPN approaches concentrate on providing secure links between private IP networks through the public Internet and includes address translation capabilities if required [Doraswamy97]. Secure links are implemented by encapsulating protected data in the payload of a regular IP packet thus creating a secure tunnel through the regular IP network [Atkinson95].

Another Internet based virtual network is the Mbone [Macedonia94]. The Mbone is an overlay network in the Internet to support experimentation with audio and video. It is based on IP multicast [Deering89] and also uses IP tunnels to traverse parts of the Internet that cannot support IP multicast. The Mbone is thus a semi-permanent virtual network built on top of the Internet.

The Mbone may be thought of as a virtual network with a service specific control architecture as was presented in Chapter 6, in that it provides a virtual network for audio and video. This is where the similarity ends, however, because the Mbone provides none of the other useful features of the switchlet environment. For example, the Mbone provides no partitioning of resources, which means that Mbone traffic adversely affects other Internet traffic. More recent proposals such as *link-sharing* [Floyd95], could potentially address these problems. Link-sharing is discussed in Section 7.7.

A more sophisticated "Internet" approach is presented in [Delgrossi97]. The authors propose the use of virtual networks, called *supranets*, to fulfil the envisaged need to form groups in a networked society. Following on from earlier work by one of the authors [Gupta95], the existence of both real-time and non-real-time transport services is assumed. Although the need to specify the capacity of virtual networks is recognised, no mechanisms to realise such resource management are proposed.

7.3.3 LAN Emulation

In an attempt to expedite the acceptance of ATM, the ATM Forum has defined a LAN emulation service [ATM Forum95]. This provides a bridging service (media access control (MAC) layer service), overlaying the standard ATM service (i.e. using a standard ATM control architecture). By mapping MAC layer addresses into ATM addresses a LAN emulation client (LEC) can establish an ATM connection to the destination. A single LAN emulation server (LES) per emulated LAN performs the necessary control functions in the network. Because an emulated LAN is a logical network on which there is much less interference than on a real LAN, it is possible to create several emulated or virtual LANs on the same physical network. In this manner it would, for example, be possible to have different departments on different virtual networks.

Note that LAN emulation is essentially another version of a closed user group virtual network and like earlier virtual networks of this kind employs address mapping facilities.

7.3.4 ATM Based Virtual Networks

In the ATM environment, virtual networks are normally created by means of virtual paths (VPs). As explained in Chapter 2, a virtual path can transparently carry several virtual channels (VCs). In [Friesen96] the following advantages of VP based virtual networks are noted[8]:

[8][Friesen96] refers to VP based networks as *overlay networks*, however the term *virtual networks* is more commonly used.

- Simplified CAC which only needs to be done at VP terminations.
- Simplified switching in transit nodes (cells only need to be switched based on VPIs).
- Dynamic topology and VP capacity changes allow adaptability to varying traffic patters and network failures.
- The possibility of segregating different traffic types into separate virtual networks.

Subdivision of traffic into more homogeneous groups is also the motivation for using VP based virtual networks in [Fotedar95]. The authors make the distinction between end-to-end virtual private networks and broadband virtual private networks. The first is a virtual network built up in its entirety by end-to-end virtual paths (also called through or non-terminating virtual paths). In the second case, virtual paths can be terminated allowing both VPI and VCI switching in the virtual network. This in turn means that a virtual channel is not confined to a particular VP. The authors envisage that the second approach will lead to an increase in statistical multiplexing gain in the network.

In [Chan97] VP based virtual networks are extended by means of a concept called virtual path groups (VPGs). A virtual path group is a logical connection of end-to-end VPs which interconnect customer access points. The VPG as a whole is allocated a certain amount of resources, and within these limits the customer is allowed to move capacity between VPs. This reallocation of resources can happen without interaction with the provider, and therefore allows the customer a certain amount of freedom to customise his or her virtual network.

The VPN management architecture presented in [Schneider93] also appears to target an ATM network. The architecture is based on Telecommunications Management Network (TMN) principles and envisages a fairly sophisticated customer interface. Interfaces are suggested that would allow customers to create and modify VPNs, and allow for modification of resources within the VPN. On the down side this appears to be a paper only architecture which was not followed up with any implementations.

In [Park97] the concept of a Customer Network Management (CNM) service is discussed. CNM services allow customers to manage a portion of the provider's network which forms part of a virtual private network. Two classes of CNM service are defined; Class I provides only monitoring capabilities, while Class II allows some control capabilities. A layered architecture is proposed, whereby SNMP requests from the customer are translated by a CNM agent into CMIP (Common Management Information Protocol) requests, which are then passed on to a "normal" public network management system. The paper contains some implementation details about the SNMP/CMIP interaction, but does not say exactly what customers will be able to control, nor how interference between customers will be prevented.

7.3.5 Discussion

Based on the above discussion the following generic groups of virtual networks (VNs), of increasing sophistication, can be identified:

- Simple VNs with dedicated switching capacity: This group is essentially limited to ATM VNs based on (permanent) virtual paths. Although network resources are guaranteed, these VNs are probably the least sophisticated in terms of user control.
- Closed User Group VNs: These VNs normally don't have any dedicated network resources allocated to them and employ various address mapping facilities to implement their functionality. Early telephony and data VPNs as well as current Internet offerings would fall in this category. Although LAN Emulation is also a closed user group approach it falls somewhere between this group and the next: by virtue of the fact that it is implemented on ATM it has some network resources dedicated to it.
- Single control architecture VNs with dedicated switching capacity: These VNs are normally implemented with a single network model and control architecture in mind and allow more sophisticated user control. Examples include LAN emulation and the VPG and the supranet approaches. An OSSA environment in which multiple control architectures of the same type, e.g. IP switching, are deployed would also fall in this grouping[9].
- Multiple control architecture VNs with dedicated switching capacity: Currently the only VN offering falling in this category is the OSSA environment

[9]Note that if physically separate IP switching controllers are used in such an OSSA environment, then virtual networks will receive dedicated switching capacity as well as dedicated control capacity.

presented in this dissertation. Different services, potentially based on different networking paradigms, can be provided in the form of different control architectures. A further distinction of the virtual networking environment described in Chapter 3 is that virtual networks can be dynamically specified and modified, and that the customers can have direct control over network resources.

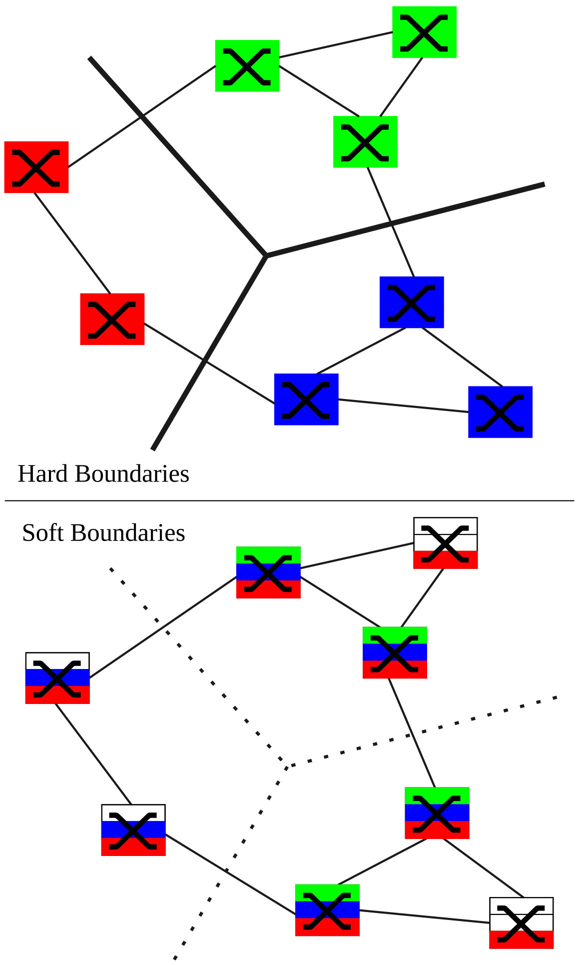

Figure 7.1: Conventional hard administrative boundaries versus OSSA soft administrative boundaries

A more subtle differentiating factor of OSSA virtual networks relates to the crossing of administrative boundaries: In conventional VPNs the creation of a virtual network is handled by means of bilateral agreements and standardised vertical interfaces between domains. Within a particular domain the service is, however, provided and controlled by the network operator in that domain and is limited in terms of functionality to that provided by the local operator. In contrast, because it provides protection between virtual networks, an OSSA VN can cross administrative boundaries, with a single service employed throughout the VN and potentially controlled by a single operator. In this manner the vertical interfaces between domains are replaced by horizontal (switchlet) interfaces which could reside in various domains and be controlled by a single control architecture. Figure 7.1 depicts the difference between these hard and soft administrative boundaries.

7.4 Active Networking

Several proposals which can be loosely grouped under the umbrella of "active networks" have recently been made. Most originate from within the Internet community, or at least assume that the unit of transfer within the network is packet based [Tennenhouse96b].

In [Yemini96] a dataflow language, called NetScript, is used to create a NetScript Virtual Network (NVN). With this approach a network is viewed as a collection of Virtual Network Engines (VNEs) connected by virtual links. A VNE can then be programmed by NetScript agents. The proponents envisage two models to cater for existing networks and protocols. In an overlay model, NetScript would form a glue layer over existing technology. A second possibility is to re-implement existing protocols using NetScript.

In a somewhat more drastic approach [Tennenhouse96a] proposes that IP packets be replaced with *capsules*. In the extreme case such a capsule will contain code which encodes how the data in the capsule should be processed at network nodes. Tennenhouse et al were the first to coin the phrase "active networks" because the networks envisaged in their proposal would be active in two ways:

- The data in a capsule could be processed as it traverses a network node.
- New programs could be injected into the network on-the-fly.

Another active networking proposal called SwitchWare [Smith96] envisages a "switch" with a programmable element or computer. The aim is to make the processing of packets fast enough that all packets can be processed by software. Sequences of active packets (which are unfortunately also called switchlets) are transported between SwitchWare switches. Like other approaches they aim to *raise* the level of abstraction in order to realise their key goals: to make networks on-the-fly programmable and thus user customisable. A proof of concept implementation of these ideas in the form of an active bridge is reported in [Alexander97]. This bridge can be upgraded on the fly from a dumb transparent bridge, to a self-learning bridge and finally a bridge supporting spanning tree algorithms to prevent looping. While this functionality appears a little contrived, they also demonstrate the ability of the bridge to gracefully change between two different media level protocols. The implementation of the active bridge depends on the ability to do bridging in software, and particularly to dynamically load modules into an execution environment.

Some of the motivations for the above work parallel that of the work presented in this dissertation. Indeed, to some extent the approaches are complementary. At the same time there are some key differences:

- A basic (stated) aim of the active network approach is to "raise" the level of abstraction in order to achieve their goals. As mentioned before, this normally leads to restrictions in that some assumptions are made about the "one" environment which will satisfy all needs.
- The active network approaches are essentially based on packet networks. Such networks do not have the distinction between data transfer and control inherent to ATM. As such, the impact of the active networking approach on any QoS guarantees that the network might have provided is not clear.
- A related issue is that their current work seems to depend on the ability to handle data transfer in software. While this is acceptable for experimental work, it is not clear how these proposals could be extended to networking elements where data transfer is primarily performed in hardware to improve performance.

7.5 The Nemesis Operating System

Although it has no direct relevance to ATM control, the Nemesis operating system [Leslie96] is considered in this section for the following two reasons:

- Its novel structure, which allows proper sharing of workstation resources, has had a direct influence on the architecture presented in this dissertation.

- Nemesis is the ideal platform on which to implement both control architectures and the OSSA[10].

Nemesis is what has become known as a *vertically integrated* operating system. System resources are scheduled at the lowest level. A very small kernel schedules a set of domains and accounts resource usage to domains. The kernel consists mainly of interrupt and trap handlers and server functionality which is normally found in an operating system kernel (e.g. device drivers), is provided in the form of "unprivileged" domains. Application crosstalk is reduced by minimising the use of shared servers: minimal functionality for a server is placed in the shared server and as much processing as possible is performed in application domains.

This provides an operating system in which applications are decoupled from each other, i.e. don't adversely effect one another. An application is also given feedback on the resources available to it and its current usage thereof, which allows applications to use resources more intelligently. (For example, application specific optimisations or adjustments can be made.)

This protection between domains makes Nemesis the ideal platform on which to run the OSSA Divider Server and, in particular, different control architectures. Such a facility would be crucial to prevent processing intensive control architectures, such as IP switching, from depriving other control architectures of essential control processor resources[11].

7.6 Multipoint Communication

7.6.1 Video Conferencing

Within the area of video conferencing numerous solutions have been proposed and implemented. Many of these have been on networks that do not offer QoS guarantees [Casner92, Eriksson94]. Despite this fundamental drawback, and the associated problems [Keshav94], the inherent openness of this environment allows a significant amount of experimentation and innovation.

[10]Nemesis was not chosen as the platform for the work presented here primarily because of its immature state at the time.

[11]A first level of defence against such *control architecture crosstalk* would be to run control architectures on physically distinct platforms. Such a hard partitioning approach will, however, not always make sense.

A multipoint conferencing system built around an Integrated Services Digital Network (ISDN), is presented in [Clark92b]. Single user workstations with a range of peripheral equipment provide the audio visual terminals (AVTs) and a hardware MCU connected to a node in the ISDN network allows multiple parties to partake in a single conference. Conferences are chaired and a distributed floor control function is used. Control data is carried in-band and is distributed to participants by the MCU.

In the ATM environment few solutions exist, mainly because of the lack of adequate support in standard control architectures. In [Wray94], a sophisticated multimedia environment for the local area is presented. Because it was developed for the local area environment, network connections (and bandwidth) were not considered a scarce resource, and were not treated in any special way. While true for the local area, this is certainly not currently true for the wide area. The use of a hardware MCU by the manufacturers of the ATM video adapters was already mentioned in Section 6.3.4.

The systems described above provide more feature rich facilities than the VideoMan proof of concept implementation described in Chapter 6. A full implementation based on the VideoMan concepts would, however, offer a more scalable service which fully utilises the potential of an ATM environment. The VideoMan also proved that the OSSA provides an open ATM environment, in which experimentation and innovation is as easy as in the Internet environment.

7.6.2 Multipoint Communication in ATM

The gather and distribution trees introduced in Section 6.3 illustrated the power of the service specific control architecture approach. These tree structures also present a very useful way to approach multipoint communication in ATM and it is worthwhile comparing this with other multipoint proposals involving ATM. In UNI-3.1 [ATM Forum94a], multipoint communication was limited to root initiated multicast trees. In [Armitage95], this UNI-3.1 capability was used to map IP multicasting onto ATM multicasting. The result was that either a multicast server was used, or a mesh of multicast trees had to be created.

The OPENET architecture [Cidon95] proposes *source* and *destination* tree structures similar to those presented in Section 6.3.1. The architecture assumes that cell interleaving will be handled by endpoints which would require support for this in the AAL used. A single cell version of the tree structures is also proposed for utilisation updates.

In [Gauthier96], many-to-many ATM connections are proposed in a demand sharing environment. A many-to-many multicast tree is assumed to be created by appropriate extensions of the standard control architectures. An in-band cell level access protocol, called SMART, then grants access to the trees, merges trees and generally ensures that there is only one source active in any tree. The in-band access control protocol relies on the special treatment of resource management (RM) cells to exchange messages.

More recently, the use of core based trees [Ballardie93], was proposed as a general multipoint-to-multipoint multicast architecture for ATM [Grossglauser97a]. This approach, called SEAM, relies on the ability of some switches to do per-VC queueing and per-packet forwarding. The latter requires a switch to detect the first cell on a VC as well as the end-of-packet cell and is used to prevent cell interleaving at merging points. Per-VC timers are used to detect the loss of end-of-packet cells. A store-and-forward mechanism of switching is proposed to prevent slow links from effectively locking outgoing ports for a particular VC during packet forwarding. The paper successfully illustrates the power of a single shared tree approach, and proposes a short-cutting mechanism which prevents unnecessary transmissions to the core switch.

The SMART and SEAM proposals achieve similar functionality to that of the VideoMan approach. Their proposals, being part of a general purpose control architecture, try to provide a general multipoint-to-multipoint architecture. While the VideoMan architecture can benefit from a more generic approach, it is not obvious that this can be achieved to the satisfaction of all applications. For example, in the case of the SMART approach, will a single access control mechanism (hardcoded in a general purpose control architecture) be sufficient for all requirements ? As mentioned before, these approaches also require modifications to the data transfer capability and/or in-band control mechanisms of ATM switches.

7.7 Bandwidth Management

Bandwidth management and CAC are extremely active research areas, and attempting to give an overview of this field is beyond the scope of this dissertation. (A recent review of CAC mechanisms can be found in [Perros96].) Rather, a few selected proposals will be described and compared with the measured effective bandwidth approach of Chapter 5.

In [Hyman91], it is assumed that different traffic classes exist. These different traffic classes group together connections with similar QoS requirements, and are allocated separate buffers in the switch. These assumptions appear to be in line with the ATM Forum defined service categories [Sathaye95], and with the way in which ATM switches are built [Rathgeb97]. The concept of a *schedulable region* is then defined [Hyman91]. A schedulable region is an N-dimensional space, the boundary of which defines the number of connections of the N different traffic types which can be operational in the switch at any moment in time, while the QoS requirements of all connections are maintained. The schedulable region also has the advantage over effective bandwidth approaches that it takes the scheduling algorithm into account. The schedulable region concept has been extended into the *admissible load region*, whereby call level statistics are used in conjunction with the schedulable region cell level abstraction to effect CAC. Finally, the approach has also been used to describe the subset of switch capacity allocated to a VP by means of what is called the *contract region*, where a contract region is effectively a partition of a schedulable region [Hyman94]. A similar approach is followed in [Bolla97].

The proposed solutions described above suffer from one major drawback, namely, determining the shape of the schedulable region. The shape of this region depends on the traffic characteristics of the sources. Two methods have been proposed to determine the schedulable region namely empirically, or by simulation if sufficient traffic models can be obtained. Neither of these approaches is very attractive or indeed practical. In the first case, the schedulable region will have to be re-evaluated whenever a new traffic source, such as a new codec, is designed. Secondly, finding traffic models that can accurately describe the behaviour of real sources appears to be elusive.

The problem of finding analytically tractable models that accurately describe the behaviour of real sources is also the Achilles heel of many other bandwidth management approaches [Perros96]. For this reason, the use of online measurement to characterise traffic behaviour and effect resource management is increasingly popular.

In [Jamin97] a measurement based CAC mechanism is proposed for use in Integrated Services Packet Networks [Clark92a]. Use of the measurement based solution is limited to *predictive* services, which offer a “fairly, but not absolutely, reliable bound on packet delivery times". Indeed the authors of [Jamin97], conclude that their approach can never be used for completely reliable services, because of the non-static nature of source behaviour. Another major difference between their approach and the work presented in Chapter 5 (and the references there), is that they do not attempt complete link utilisation and typically aim for 90 percent utilisation.

Resource management of ATM networks at the virtual path and virtual network level has so far received considerably less attention than the CAC problem. In an extension of their work on parallel connection control (see Section 7.1.3.2), a DIstributed and dynamic VP management Algorithm (DIVA) is proposed in [Srinivasan97]. An algorithm is proposed whereby the resources of virtual paths are dynamically adjusted based on either call level statistics, or the bandwidth usage of a VP. It is assumed that each switch will be able to periodically provide such information to a Connection Server which, for a group of switches, will run the algorithm and update resource allocations on the switches. Although the use of effective bandwidth is assumed, no details are provided on how statistics will be obtained. Rather than performing these resource management actions by means of management protocols such as SNMP, DIVA is proposed as an extension of the PNNI control architecture.

A Network Resource Management (NRM) system is proposed in [Woodruff97]. The system is based on TINA concepts and defines a layered network approach. According to this proposal, transmission path (TP), virtual network (VN), virtual path (VP) and virtual connection (VC) layer networks are respectively defined. Lower layers are supposed to provide a service to higher layers, and can allocate some of their resources to client layers, e.g. the VP layer to the VC layer. The system defines a set of generic functions for each layer which can then be specialised at specific layers. An example function, which lends itself particularly well to the notion of specialisation, is admission control and functionality similar to that described in Section 5.3 is proposed. The paper deals mainly with architectural issues, and also proposes the use of effective bandwidth as a promising approach to bandwidth management.

The concept of *link-sharing* in packet based networks is proposed in [Floyd95]. Link-sharing allows multiple organisations, multiple protocols, or multiple traffic types to share the bandwidth available on a link in a controlled fashion[12]. The same mechanisms are assumed to be applied to all kinds of sharing, and the

[12]Indeed this generic "definition" of link-sharing means that the approach is quite close to that of the OSSA, albeit applied to a different kind of network.

proposal allows *hierarchical* sharing of bandwidth whereby shares of shares can be allocated to different groups (or classes). The goals of link sharing are as follows:

- Over a period of time each class should receive its allocated percentage of the link bandwidth.
- Any "excess" bandwidth not currently used by any of the classes should be available for use by other classes, and allocation of such bandwidth should follow a set of reasonable guidelines.

An important consequence of the second goal is that classes are only limited to their share of bandwidth in times of congestion. At other times classes can use more than their share, but in a controlled fashion. These goals are similar to the guaranteed and optimistic capacity concepts proposed in Section 5.3.2.

7.8 Summary

This chapter has described various contributions which are related to the work presented in this dissertation. The structure of the chapter has roughly parallelled the order in which work was presented in the earlier part of the dissertation.

Various control architectures were described, again motivating the need to have several control architectures operational simultaneously in the same network. Most of these control architectures were shown to be general purpose, in contrast to the concept of service specific control architectures.

Other approaches to open switch control were considered and compared with the Ariel open switch control interface. A section on various approaches to virtual networking in ATM and other types of networks followed. It was shown that a virtual network service offering different control architectures per virtual network is unique to the OSSA solution.

Sections on active networking, the Nemesis operating system and multipoint communication followed, before the chapter ended with a section on bandwidth management. Problems with traffic model based approaches to bandwidth management were indicated. A number of resource management architectures were described and similarities with the approach presented in this dissertation indicated.

Chapter 8

Conclusion

This dissertation has presented an Open Service Support Architecture which provides the control framework to allow ATM to realise its true potential as a technology for multi-service networks. This chapter summarises the work and its conclusions, and suggests areas for further study.

8.1 Summary

Chapter 1 motivated the need for a consolidated control framework for ATM by listing a number of inadequacies of standard control architectures and highlighting the fact that these shortcomings have been widely recognised and that in spite of the various proposals that have been made to rectify them, their success has been limited. This is mainly because, by and large, they proposed to replace one monolithic system with another. Avoiding the same trap, the main goal of the architecture presented in this dissertation is as follows: *To create an environment in which the inadequacies of the legacy ATM control approach can be addressed, while at the same time not preventing the use of conventional or indeed any other solutions. Furthermore, the architecture should be able to simultaneously accommodate these different control architectures and the services built around them in the same physical network.*

In order to provide a context in which the approach to this aim could be explored, Chapter 2 provided essential background information. Terminology and concepts for both ATM networks and DPEs were described. The concept of a network "service" in the OSSA was defined to encompass network inherent and network derived services, any of which should be easily provided, modified and replaced by any authorised network user. The experimental environment and related research projects in the Computer Laboratory were described.

In Chapter 3, the Open Service Support Architecture was described and shown to achieve the main goal stated in Chapter 1. A prerequisite for this approach is an open switch control interface and a solution in the form of the Ariel switch control interface was described. On its own, an open switch control interface does not afford the openness required by the OSSA, and a key element in the OSSA solution, namely the switchlet concept, was introduced. A software entity called a Divider controls the physical switch and partitions switch resources such as bandwidth, buffer space, ports and VPI/VCI addresses. Each partition is called a switchlet and can be controlled by a separate instance of a control architecture. This means that several control architectures can be operational at the same time, on the same switch and within the same network. In addition to partitioning the physical switch into switchlets, the Divider entity has to police invocations made by different control architectures to ensure that they don't interfere. Switchlets combine to form virtual networks, each of which is then controlled by a different instance of a control architecture.

The switchlet concept provides the local mechanism required at each switch; the balance of Chapter 3 introduced the remaining OSSA components and showed how they can be integrated to yield a virtual network service. An entity called the Network Builder allows the dynamic creation of virtual networks by interacting with Divider entities associated with each switch. The control architecture controlling a newly created virtual network can either be one of a known set, such as a standards based control architecture, or it could be a completely new and unknown control architecture. Bootstrapping the OSSA by means of a bootstrap virtual network was described, and it was explained how such a bootstrap network generalises the bootstrap requirement inherent to all networks.

The proposed architecture meets the primary stated goal by allowing both conventional and new control architectures and services to simultaneously operate in the same network and by providing appropriate protection mechanisms between them.

Chapter 4 presented proof of concept implementations for various OSSA components and the virtual network service. The variety of implementations proved the flexibility of the architecture to adapt to switches of different sizes and capabilities. Experiments with these implementations showed that the opening up of switch control in this fashion is not prohibitively expensive. In particular, the cost of having a Divider entity in the control path did not significantly increase the control overhead. The implementation of the virtual network service was described in detail, showing how virtual networks can be created and modified on demand depending on user requirements.

The various implementations presented proved that the proposed architecture can be implemented and implemented efficiently. Furthermore, the varied implementations clarified the efficiency tradeoffs of different implementation options.

The bandwidth management requirements of the OSSA at various levels and time scales were considered in Chapter 5. All ATM networks require connection admission control (CAC) and in an OSSA environment this extends to the concept of virtual network admission control (VNAC). Apparent consensus among the resource management community about the potential of measurement based bandwidth management motivated a study of its applicability in the OSSA environment. In particular, measurement based effective bandwidth estimates were shown to be very attractive for this environment, because it can be applied to both single connections and groups of connections, such as switchlets. The chapter presented an implementation of effective bandwidth based bandwidth management in the OSSA. The results proved the feasibility of using such methods on real ATM switches, and indeed their suitability to the OSSA environment. Several open issues in terms of the use of measured effective bandwidth for CAC were identified, and it was concluded that the approach has immediate applicability to the longer time scales of virtual network bandwidth management.

The work presented in Chapters 3 to 5 successfully addressed the main aim of the dissertation identified in Chapter 1, namely the development of the OSSA. However, the OSSA permits more than simply the satisfaction of the goal of open control, allowing new approaches and new ideas to be applied to an ATM networking environment. One such new idea, explored in Chapter 6, is the concept of service specific control architectures (SSCAs).

An SSCA utilises knowledge about the service it provides to manipulate network resources and to supply a more efficient service. Chapter 6 motivated the requirement for SSCAs by considering several example environments which stand to benefit from the use of this approach. A specific example, namely an SSCA for continuous media conferencing, was then discussed in detail, and a proof of concept implementation described. The implemented SSCA exploited the dynamics of a conference session to construct novel connection trees in the network.

The SSCA was integrated with the floor control function of the conference, which can manipulate these tree structures within the network. It was shown how this approach makes more efficient use of network resources and in particular how the OSSA provided an open environment in which experimentation and innovation can take place.

Finally, related work was considered in Chapter 7. Several control architectures were described, and it was shown how these proposals differ from the approach described in this dissertation. Many proposed control architectures try to provide an open environment within a specific control architecture, whereas in the OSSA the openness is at a more fundamental level in that different control architectures can be accommodated. Many of the alternative control architectures are also general purpose in nature, as opposed to the service specific control architecture concept enabled by the OSSA. Other approaches to open switch control and virtual networking were considered. The OSSA approach is believed to be unique in being able to provide virtual networks of different type. Other proposed solutions to resource management were considered and compared to the measurement based effective bandwidth implementation.

The work presented here has successfully addressed the goals set out in Chapter 1 and it is the thesis of this dissertation that a control framework such as the OSSA is crucial to enable the full exploitation of the capability of ATM to be the technology of choice for multi-service networks. Furthermore, the OSSA stands to break the mainframe model of telecommunications by allowing control architectures to be developed independent of switch manufacturers and vendors. This in turn will create a more dynamic environment in which new services can be developed and deployed in a much more rapid fashion than is the norm today.

8.2 Further Work

The OSSA implementation introduced in this dissertation is of necessity a prototype system. While this was sufficient to prove the power and potential of this system, some open issues remain before the OSSA approach can be engineered for deployment into a production ATM network. A number of further research topics have also been raised as a result of this work. These are considered in this section.

One of the major strengths of the OSSA approach is that standard control architectures, such as the ATM Forum's UNI/PNNI specification, can be allowed to operate concurrently with other control architectures. The Divider implementation discussed in Chapter 4 could also export a GSMP type switchlet control interface, which means that an IP switching control architecture could already be made to operate in this environment. Despite this possibility, no standard control architectures have yet made use of the OSSA environment. While no problems are expected, other than normal engineering issues, this must be one of the most pressing outstanding concerns to fully prove the OSSA concept. Work in this regard is expected to be reported in [Rooney97c].

The current lack of a unifying open switch control approach was identified in Chapter 7. In particular, lack of consensus on how QoS issues should be handled will require further study. It is also not clear yet how newer switch functions such as per-VC queueing and partial or early packet discard will influence this interface. In fact, it might not be possible to come up with a unifying approach in which case the emphasis has to be on an extensible interface. Some in-band control functions, such as ABR control and sophisticated buffer management schemes will also have an influence on the switch control interface and the switchlet mechanism in particular, and require further investigation.

Although not dictated by the architecture, the current implementation of switch resource partitioning is done in a fairly coarse-grained fashion. Such partitioning is expected to fit more conventional control architectures which are unaware of the switchlet environment. To prevent wastage of resources that are allocated but not used, a more dynamic and finer grained partitioning of resources will be required. The incremental bandwidth management concept introduced in Chapter 5 could be used and extended to other switch resources. Control architectures will likewise have to be modified to make full use of such partitioning. Indeed, an environment can be envisaged in which resources in the OSSA are allocated and freed in similar fashion to memory in an operating system. The related concepts of guaranteed and optimistic bandwidth are also worthy of further investigation, to ensure that available bandwidth is optimally used.

Security issues of the OSSA were given cursory attention in Chapter 3. While the necessary technological building blocks to address these issues are believed to be available, further work is needed to formalise this and to integrate it into the OSSA environment.

Although not a standard ATM control architecture, the IETF's RSVP protocol could, by means of the OSSA, be applied to a pure ATM environment.

Again, only the data transfer capability of ATM will be used, and the normal IP addressing, routing etc. required by RSVP will be operational. If layered encoding is used to effect the merging facility of RSVP, then tree structures such as the distribution and gather trees introduced in Chapter 6 can be employed. Indeed this might be one approach to generalise the VideoMan solution into a more generic multipoint-to-multipoint facility.

Using RSVP as the control architecture in an ATM environment would also contribute to the apparent convergent paths of the ATM and Internet communities: Approaches such as IP switching are already blurring traditionally held boundaries, and the emergence of several in-band control actions [Gauthier96, Hjalmtysson97] have abolished the "purist" ATM approach of data that is transparently switched. Assuming state keeping routers with switching capabilities [Callon97] and frame forwarding ATM switches [Grossglauser97a] further blurs the distinction between switching and routing, and a major area of future research has to be aimed at finding out if and how proposals like this, and indeed the work presented in this dissertation, will fit together.

Finally, the switchlet concept could potentially be applied to other types of networks, especially connection oriented networks. An example might be a mobile (phone) network: In this environment the hand-off characteristics might be different depending on whether the device is in fact a mobile phone, or a different mobile device, such as a mobile computer. The switchlet concept can provide a solution by allowing different control architectures to offer the diverse functionality.

Bibliography

[Adam97] C. M. Adam, A. A. Lazar, and M. Nandikesan. *QOS Extensions to GSMP.* Technical Report CU/CTR/TR 471-97-05, Center for Telecommunications Research, Columbia University, New York, April 1997. (p 128)

[Alexander97] D. Scott Alexander, Marianne Shaw, Scott M. Nettles, and Jonathan M. Smith. *Active Bridging.* Computer Communications Review (SIGCOMM 97), 27(4):101–111, October 1997. (p 135)

[APM92] APM. *ANSAware 4.1 System Programming in ANSAware.* APM Limited, Castle Park, Cambridge, UK, Release 4.1 edition, 1992. (pp 15, 25)

[APM93] APM. *The ANSA Model for Trading and Federation.* Technical Report AR.005.00, APM Limited, Castle Park, Cambridge, UK, 1993. (p 41)

[Armitage95] G.J. Armitage. *Multicast and Multiprotocol support for ATM based Internets.* Computer Communications Review, 25(2):34–46, April 1995. (p 137)

[Atkinson95] R. Atkinson. *IP Encapsulating Security Payload (ESP).* Internet RFC 1827, August 1995. (p 129)

[Atkinson96] Randall Atkinson. *Security Architecture for the Internet Protocol.* Internet Draft: draft-ietf-ipsec-arch-sec-01.txt, November 1996. (p 129)

[ATM Forum93] ATM Forum. *ATM User-Network Interface Specification - Version 3.0.* Prentice Hall, 1993. (pp 3, 22, 26, 29, 97, 116)

[ATM Forum94a] ATM Forum. *ATM User-Network Interface Specification - Version 3.1.* The ATM Forum, 1994. (pp 22, 101, 113, 116, 137)

[ATM Forum94b] ATM Forum. *ATM User-Network Interface Specification - Version 4.0.* The ATM Forum, 1994. ATM Forum document: af-sig-0061.000. (pp 8, 13, 113, 116)

[ATM Forum95] ATM Forum. *LAN Emulation Over ATM Specification - Version 1.* The ATM Forum, 1995. (p 130)

[ATM Forum96] ATM Forum. *Private Network-Network Interface Specification Version 1.0 (PNNI 1.0).* ATM Forum document: af-pnni-0055.000 , March 1996. (pp 3, 8, 13, 26, 27, 35, 97, 101, 116, 118)

[Bale95] M.C. Bale. *Signalling in the intelligent network.* BT Technology Journal, 13(2):30–42, April 1995. (pp 4, 26)

[Ballardie93] Tony Ballardie, Paul Francis, and Jon Crowcroft. *Core Based Trees (CBT), An Architecture for Scalable Inter-Domain Multicast Routing.* Computer Communications Review (SIGCOMM 93), 23(4):85–95, 1993. (pp 105, 138)

[Barham95] P. Barham, M. Hayter, D. McAuley, and I. Pratt. *Devices on the Desk Area Network.* IEEE Journal on Selected Areas in Communication, 13(4):722–732, May 1995. (p 22)

[Bellcore97] Bellcore. *Q.port Portable ATM Signalling Software Overview.* Available as: `http://www.bellcore.com/QPORT/qport-ov.html`, 1997. (p 113)

[Berson96] S. Berson. *IP Integrated Services Support in ATM.* Internet Draft: draft-ietf-issll-atm-support-00.ps, June 1996. (p 120)

[Biagioni93] Edoardo Biagioni, Eric Cooper, and Robert Samsom. *Designing a Practical ATM LAN.* IEEE Network, 7(2):32–39, March 1993. (p 20)

[Bloem95] Jack Bloem, Juan Pavon, Hiroshi Oshigiri, and Mike Schenk. *TINA-C Connection Management Components.* Proceedings TINA Conference, Melbourne, Australia, February 1995. (pp 121, 122)

[Bolla97] Raffaele Bolla, Franco Davoli, and Mario Marchese. *Bandwidth Allocation and Admission Control in ATM Networks with Service Separartion.* IEEE Communictions Magazine, 35(5):130–137, May 1997. (p 139)

[Bos97] H. Bos. *Active Distributed File Servers.* Presented at: ERSADS '97, March 1997. (p 124)

[Bosco96] P.G. Bosco, Eirik Dahle, Michel Gien, Andrew Grace, Nicolas Mercouroff, Nathalie Perdigues, and Jean-Bernard Stefani. *The ReTINA Project.* Technical Report TR-96-15, 1996. Available from `http://www.chorus.com/Chorus/Research/retina.html`. (pp 17, 122)

[Braden94] R. Braden, D. Clark, and S. Shenker. *Integrated Services in the Internet Architecture: an Overview.* Internet RFC 1633, June 1994. (pp 31, 120)

[Bubenik91] Rick Bubenik, John DeHart, and Mike Gaddis. *Multipoint Connection Management in High Speed Networks.* In INFOCOM, pages 59–68. IEEE, April 1991. (pp 4, 26)

[Callon97] R. Callon, P. Doolan, N. Feldman, A. Fredette, G.Swallow, and A. Viswanathan. *A Framework for Multiprotocol Label Switching.* Internet Draft: draft-ietf-mpls-framework-01.txt, July 1997. (pp 120, 147)

[Casner92] Stephen Casner and Stephen Deering. *First IETF Internet Audiocast.* Computer Communications Review, 22(3):1–6, July 1992. (p 136)

[Chan96] M.C. Chan, H. Hadama, and R. Stadler. *An architecture for broadband virtual networks under customer control.* In Proceedings of NOMS '96 - IEEE Network Operations and Management Symposium, 1996. (p 7)

[Chan97] M.C. Chan, A.A. Lazar, and R. Stadler. *Customer Management and Control of Broadband VPN Services.* In A. Lazar, R. Saracco, and R. Stadler, editors, *Integrated Network Management V*, pages 301–314. IFIP & IEEE, Chapman & Hall, May 1997. (p 131)

[Cidon95] Israel Cidon, Tony Hsiao, Asad Khamisy, Abhay Parekh, Raphael Rom, and Moshe Sidi. *The OPENET Architecture.* Technical Report SMLI TR-95-37, Sun Microsystems Laboratories, 2550 Garcia Avenue, Mountain View, CA 94043, December 1995. (pp 4, 26, 27, 105, 120, 137)

[Clark92a] David D. Clark, Scott Shenker, and Lixia Zhang. *Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanism.* Computer Communications Review (SIGCOMM 92), 22(4):14–26, 1992. (p 139)

[Clark92b] William J. Clark. *Multipoint Multimedia Conferencing.* IEEE Communications Magazine, 30(5):44–50, May 1992. (pp 104, 114, 137)

[Crosby95a] Simon Crosby. *Performance Management in ATM Networks.* Technical Report 393, University of Cambridge, Computer Laboratory, UK, 1995. (pp 4, 14, 26, 32)

[Crosby95b] Simon Crosby, Ian Leslie, John Lewis, Neil O'Connell, Raymond Russell, and Fergal Toomey. *Bypassing Modelling: an Investigation of Entropy as a Traffic Descriptor in the Fairisle ATM Network.* In Proceedings Twelfth UK Teletraffic Symposium, Windsor, England, March 1995. IEE, Springer Verlag. (pp 33, 71, 72)

[Crosby96] Simon Crosby, Ian Leslie, Merial Huggard, John Lewis, Brian McGurk, and Raymond Russell. *Predicting Effective Bandwidths of ATM and Ethernet Traffic.* In Proceedings Thirteenth UK Teletraffic Symposium, Strathclyde University, Glasgow, March 1996. IEE. (pp 72, 73, 76, 94)

[Davie96] Bruce Davie, Paul Doolan, Jeremy Lawrence, Keith McCloghrie, Yakov Rekhter, Eric Rosen, and George Swallow. *Use of Tag Switching With ATM.* Internet Draft: draft-davie-tag-switching-atm-00.txt, October 1996. (pp 4, 26, 120)

[Davison97] R. Davison and M. Azmoodeh. *Performance Management of Public ATM networks - A Scaleable and Flexible Approach.* In A. Lazar, R. Saracco, and R. Stadler, editors, *Integrated Network Management V*, pages 275–286. IFIP & IEEE, Chapman & Hall, May 1997. (p 17)

[DCA97] *Devolved Control of ATM Networks*. Project overview available from: `http://www.cl.cam.ac.uk/Research/SRG/dcan/`, 1997. (p 25)

[de Prycker91] Martin de Prycker. *ASYNCHRONOUS TRANSFER MODE Solution for Broadband ISDN*. Ellis Horwood, 1991. (p 13)

[Deering89] Steve Deering. *Host Extensions for IP Multicasting*. Internet RFC 1112, August 1989. (p 129)

[Delgrossi97] Luca Delgrossi and Domenico Ferrari. *A Virtual Network Service for Integrated-Services Internetworks*. In 7th International Workshop on Networks and Operating System Support for Digital Audio and Video (NOSSDAV), pages 307–311. IEEE and ACM, May 1997. (p 130)

[Doeringer93] W.A. Doeringer, H.D. Dykeman, A. Engbersen, R. Guerin, A. Herkersdorf, and L. Heusler. *Fast Connection Establishment in Large-Scale Networks*. In INFOCOM, pages 489–496. IEEE, April 1993. (pp 4, 26)

[Doraswamy97] Naganand Doraswamy. *Implementation of Virtual Private Networks (VPNs) with IP Security*. Internet Draft: draft-ietf-ipsec-vpn-00.txt, March 1997. (p 129)

[Duffield95] N.G. Duffield, J.T. Lewis, Neil O'Connell, Raymond Russell, and Fergal Toomey. *Entropy of ATM Traffic Streams: A Tool for Estimating QoS Parameters*. IEEE Journal on Selected Areas in Communication, 13(6):981–990, August 1995. (p 71)

[Eriksson94] Hans Eriksson. *MBONE: The Multicast Backbone*. Communications of the ACM, 37(8):54–60, August 1994. (pp 104, 136)

[Floyd95] Sally Floyd and Van Jacobson. *Link-Sharing and Resource Management Models for Packet Networks*. IEEE/ACM Transactions on Networking, 3(4):365–386, August 1995. (pp 80, 129, 140)

[FORE94] FORE. *200-Series ATM Adapter: Design and Architecture.* Anonymous ftp from fore.com in Jan 1994. FORE Systems, Inc., 1000 FORE Drive, Warrendale, PA 15086-7502, USA., 1994. (p 21)

[FORE95a] FORE. *ForeRunner ASX-200 ATM Switch Users's Manual.* FORE Systems, Inc., 1000 FORE Drive, Warrendale, PA 15086-7502, USA., 1995. (p 113)

[FORE95b] FORE. *SPANS UNI: Simple Protocol for ATM Signalling.* FORE Systems, Inc., 1000 FORE Drive, Warrendale, PA 15086-7502, USA., 1995. Release 3.0. (pp 22, 29)

[FORE97] FORE. *ATM Network Switches.* FORE Systems, Inc., 1000 FORE Drive, Warrendale, PA 15086-7502, USA.`http://www.fore.com/products/index.html`, 1997. (p 95)

[Fotedar95] Shivi Fotedar, Mario Gerla, Paola Crocetti, and Luigi Fratta. *ATM Virtual Private Networks.* Communications of the ACM, 38(2):101–109, Feb 1995. (p 131)

[Fraser93] A.G. Fraser. *Early Experiments with Asynchronous Time Division Networks.* IEEE Network, 7(1):12–26, Jan 1993. (p 11)

[Friesen96] V.J. Friesen, J.J. Harms, and J.W. Wong. *Resource Management with Virtual Paths in ATM Networks.* IEEE Network, 10(5):10–20, September/October 1996. (pp 7, 130)

[Garrahan93] James J. Garrahan, Peter A. Russo, Kenichi Kitami, and Roberto Kung. *Intelligent Network Overview.* IEEE Communications Magazine, 31(3):30–36, March 1993. (p 129)

[Garret97] Mark W. Garret and Marty Borden. *Interoperation of Controlled-Load Service and Guaranteed Service with ATM.* Internet Draft: draft-ietf-issll-atm-mapping-03.txt, July 1997. (p 32)

[Gauthier96] Eric Gauthier, Jean-Yves Le Boudec, and Philippe Oechslin. *Shared Access to Many-to-Many ATM Connections.* In IEEE Global Telecommunications Conference, pages 2123–2127. IEEE Globecom, November 1996. (pp 106, 138, 147)

[Gelenbe97] Erol Gelenbe, Xiaowen Mand, and Raif Önvural. *Bandwidth Alocation and Call Admission Control in High-Speed Networks.* IEEE Communications Magazine, 35(5):122–129, May 1997. (p 76)

[Glynn94] Peter W. Glynn and Ward Whitt. *Logarithmic asymtotics for steady-state tail probabilities in a single-server queue.* Journal of Applied Probability (Special Edition), 31A:131–156, 1994. (p 73)

[Greaves90] David Greaves, Dimitris Lioupis, and Andy Hopper. *The Cambridge Backbone Ring.* Technical Report 90.2, Olivetti Research Ltd., 24a Trumpington Street, Cambridge CB2 1QA, England, 1990. (p 14)

[Greaves97] David J. Greaves and Richard J. Bradbury. *Low Cost ATM and The ATM Warren.* To appear, 1997. (p 124)

[Grossglauser97a] M. Grossglauser and K.K. Ramakrishnan. *SEAM: A Scheme for Scalable and Efficient ATM Multipoint-to-Multipoint Communication.* In INFOCOM. IEEE, April 1997. (pp 105, 106, 138, 147)

[Grossglauser97b] Matthias Grossglauser and David Tse. *A Framework for Robust Measurement-Based Admission Control.* Computer Communications Review (SIGCOMM 97), 27(4):237–248, 1997. (pp 34, 76, 94)

[Gupta95] A. Gupta and D. Ferrari. *Resource Partitioning for Real-Time Communication.* IEEE/ACM Transactions on Networking, 3(5):501–508, Oct 1995. (p 130)

[Hall94] Jane Hall. *D6.4A, Virtual Private Networks Vol. I, VPN State of the Art.* Available from: `http://www.fokus.gmd.de/minos/prepare/entry.html`, August 1994. (p 128)

[Händel] Rainer Händel, Manfred N. Huber, and Stefan Schröder. *ATM Networks Concepts, Protocols, Applications.* Addison-Wesley, 2nd edition, 94. (p 3)

[Hayter93] M.D. Hayter. *A workstation architecture to support multimedia.* Technical Report 319, University of Cambridge Computer Laboratory, November 1993. (p 14)

[Hayton96] Richard Hayton. *OASIS An Open Architecture for Security Interworking Services.* PhD thesis, University of Cambridge, Computer Laboratory, UK, 1996. Available as Technical Report No. 399. (p 48)

[Heinanen93] Juha Heinanen. *Multiprotocol Encapsulation over ATM Adaption Layer 5.* Internet RFC 1483, July 1993. (p 119)

[Heinanen94] Juha Heinanen and Ramesh Govindan. *NBMA Address Resolution Protocol (NARP).* Internet RFC 1735, December 1994. (p 119)

[Herbert94a] Andrew Herbert. *An ANSA Overview.* IEEE Network, 8(4):18–23, January/February 1994. (pp 15, 17)

[Herbert94b] Andrew Herbert, Ian Leslie, Derek McAuley, and Cosmos Nicolaou. *Scalable Distributed Control of ATM Networks.* Project proposal, University of Cambridge, Computer Laboratory, UK. Project overview available from: `http://www.ansa.co.uk/DCAN/index.html`, 1994. (p 25)

[Hjalmtysson97] Gisli Hjalmtysson and K.K. Ramakrishnan. *UNITE - An Architecture for Lightweight Signalling in ATM Networks.* To be published (Also presented at OpenSig97, Cambridge, U.K.), April 1997. (pp 4, 26, 32, 124, 147)

[Hopper78] Andrew Hopper. *Local Area Computer Communication Networka.* PhD thesis, University of Cambridge, Computer Laboratory, UK, 1978. Available as Technical Report No. 7. (p 14)

[Hopper88] Andrew Hopper and Roger M. Needham. *The Cambridge Fast Ring Networking System.* IEEE Transactions on Computers, 37(10):1214–1223, October 1988. (p 14)

[Hui88] Joseph Y Hui. *Resource Allocation for Broadband Networks.* IEEE Journal on Selected Areas in Communication, 6(9):1598–1608, December 1988. (pp 33, 72)

[Hyman91] Jay M. Hyman, Aurel A. Lazar, and Giovanni Pacifici. *Real-Time Scheduling with Quality of Service Constraints.* IEEE Journal on Selected Areas in Communication, 9(7):1052–1063, September 1991. (pp 71, 76, 139)

[Hyman93] Jay M. Hyman, Aurel A. Lazar, and Giovanni Pacifici. *A Seperation Principle between Scheduling and Admission Control for Broadband Switching.* IEEE Journal on Selected Areas in Communication, 11(4):605–616, May 1993. (p 71)

[Hyman94] J.M. Hyman, A.A. Lazar, and G. Pacifici. *VC, VP and VN Resource Assignment Strategies for Broadband Networks.* In Proceedings of the 4th International Workshop on Network and Operating System Support for Digital Audio and Video, pages 99–110, 1994. (pp 71, 139)

[i-cubed96] i-cubed. *SASCiA (the SCSI ATM switch): product overview.* i-cubed Ltd., Rustat House, 62 Clifton Road, Cambridge, UK, 1996. (p 100)

[Ipsilon96] Ipsilon. *IP Switching: The Intelligence of Routing, the Performance of Switching.* Available from: `http://www.ipsilon.com/productinfo/techwp1.html`, 1996. (pp 27, 35, 47, 119)

[ITU-T93a] ITU-T. *Integrated Services Digital Network (ISDN) Service Capabilities.* ITU-T Recommendation I.210, 1993. (p 17)

[ITU-T93b] ITU-T. *Integrated Services Digital Network (ISDN) Service Capabilities - B-ISDN Service Aspects.* ITU-T Recommendation I.211, 1993. (p 18)

[Iwata95] A. Iwata, N. Mori, C. Ikeda, H. Suzuki, and M. Ott. *ATM Connection and Traffic Management Schemes for Multimedia Interworking.* Communications of the ACM, 38(2):72–89, Feb 1995. (pp 4, 26, 27)

[Jamin97] Sugih Jamin, Peter B. Danzig, Scott J. Shenker, and Lixia Zhang. *A Measurement-based Admission Control Algorithm for Integrated Services Packet Networks (Extended Version).* IEEE/ACM Transactions on Networks, 5(1):56–70, February 1997. (pp 76, 139, 140)

[Kelly95] Edward Kelly, Nicolas Mercouroff, and Peter Graubmann. *TINA-C DPE Architecture and Tools.* In Proceedings TINA Conference, Melbourne, Australia, pages 39–54, February 1995. (p 23)

[Kelly96] Frank Kelly. *Notes on Effective Bandwidths.* In S. Zachary F.P. Kelly and I.B. Ziedins, editors, *Stochastic Networks: Theory and Applications*, pages 141–168. Oxford University Press, 1996. (pp 33, 72)

[Keshav94] S. Keshav. *Experience with Large Videoconferences on Xunet II.* In INET'94/JENC5, 1994. (p 136)

[Laubach94] Mark Laubach. *Classical IP and ARP over ATM.* Internet RFC 1577, January 1994. (pp 27, 47, 119)

[Lazar95] Aurel A. Lazar, Shailendra K. Bhonsle, and Koon Seng Lim. *A Binding Architecture for Multimedia Networks.* Journal of Parallel and Distributed Computing, 30(2):204–216, November 1995. (pp 4, 5, 17, 26, 40, 121, 122, 127)

[Lazar96a] A. A. Lazar, K. S. Lim, and F. Marconcini. *Realizing a Foundation for Programmability of ATM Networks with the Binding Architecture.* IEEE Journal on Selected Areas in Communication, 14(7):1214–1227, Sept 1996. (pp 6, 122, 123)

[Lazar96b] A. A. Lazar and Franco Marconcini. *Towards an Open API for ATM Switch Control.* Available from: `http://www.ctr.columbia.edu/comet/xbind/xbind.html`, 1996. (pp 5, 127)

[Leslie91] I.M. Leslie and D.R. McAuley. *Fairisle: An ATM Network for the Local Area.* Computer Communications Review (SIGCOMM 91), 21(4):327–336, September 1991. (pp 14, 19, 32, 73)

[Leslie96] I.M. Leslie, D. McAuley, R. Black, T. Roscoe, P. Barham, D. Evers, R. Fairbairns, and E. Hyden. *The Design and Implementation of an Operating System to Support Distributed Multimedia Applications.* IEEE Journal on Selected Areas in Communication, 14(7):1280–1297, Sep 1996. (pp 2, 24, 35, 48, 135)

[Lewis96] John T. Lewis and Raymond Russell. *An Introduction to Large Deviations.* Available from: `http://www.cl.cam.ac.uk/Research/SRG/measure/`

`tutorial/Lausanne96-tutorial.ps.gz`, August 1996. (pp 72, 74)

[Lewis97] John T. Lewis, Raymond Russell, Fergal Toomey, Brian McGurk, Simon Crosby, and Ian Leslie. *Practical Connection Admission Control for ATM Networks Using On-line Measurements.* Invited paper for Computer Communications. To appear, 1997. (pp 34, 71, 73, 75, 76)

[Li95] Guangxing Li. *DIMMA Nucleus Design.* Technical Report APM.1551.00.05, APM Limited, Castle Park, Cambridge, UK, 1995. (pp 20, 23)

[Luciani96] James V. Luciani, Dave Katz, David Piscitello, and Bruce Cole. *NBMA Next Hop Resolution Protocol (NHRP).* Internet Draft: draft-ietf-rolc-nhrp-10.txt, 1996. (p 119)

[Macedonia94] M.R. Macedonia and D.P. Brutzman. *Mbone provides Audio and Video across the Internet.* IEEE Computer, 27(4):30–36, April 1994. (p 129)

[McAuley90] D.R. McAuley. *Protocol Design for High Speed Networks.* PhD thesis, University of Cambridge, Computer Laboratory, New Museums Site, Cambridge CB2 3QG, UK, 1990. Available as Technical Report No. 186. (pp 2, 14)

[Mea97] *Measure - On-line Measurement of Resource Management.* Project overview available from: `http://www.cl.cam.ac.uk/Research/SRG/measure/`, 1997. (p 24)

[Miller92] Patrick A. Miller and Petre N. Turcu. *Generic Signalling Protocol: Switching, Networking, and Interworking.* IEEE Transactions on Communications, 40(5):967–979, May 1992. (pp 4, 26)

[Minetti96] R. Minetti. *Service Architecture - Definition of Service Architecture - Version 4.0.* TINA-C deliverable available from: `http://www.tinac.com/`, October 1996. (pp 18, 122)

[Minzer91] Steven Minzer. *A Signalling Protocol for Complex Multimedia Services.* IEEE Journal on Selected Areas in Communication, 9(9):1383–1394, December 1991. (pp 4, 26)

[Modarressi90] Abdi R. Modarressi and Ronald A. Skoog. *Signalling System No. 7: A Tutorial.* IEEE Communications Magazine, 28(7):19–35, July 1990. (pp 101, 116)

[Needham82] R.M. Needham and A.J. Herbert. *The Cambridge Distributed Computing System.* Addison-Wesley Publishing Company, 1982. (p 15)

[Newman89] Peter Newman. *Fast packet switching for integrated services.* PhD thesis, University of Cambridge, Computer Laboratory, 1989. Available as Techinal Report No 165. (p 14)

[Newman96a] P. Newman, W. Edwards, R. Hinden, F. Ching Liaw E. Hoffman, T. Lyon, and G.Minshall. *Ipsilon's General Switch Management Protocol Specification Version 1.1.* Internet RFC 1987, 1996. (pp 5, 54, 56, 83, 120, 127)

[Newman96b] P. Newman, W. L. Edwards, R. Hinden, E. Hoffman, F. Ching Liaw, T. Lyon, and G. Minshall. *Ipsilon Flow Management Protocol Specification for IPv4 - Version 1.0.* Internet RFC 1953, May 1996. (pp 27, 120)

[Newman96c] P. Newman, W.L. Edwards, R. Hinden, E. Hoffman, F. Ching Liaw, T. Lyon, and G. Minshall. *Transmission of Flow Labelled IPv4 on ATM Data Links - Ipsilon Version 1.0.* Internet RFC 1954, 1996. (p 120)

[Newman97a] Peter Newman and Greg Minshall. *Quality of Service Enhancements to the General Switch Management Protocol (GSMP).* Presentation at: Sprint Applied Networking Research Symposium, March 1997. Slides available from: `http://www.ipsilon.com/~pn/`. (p 127)

[Newman97b] Peter Newman, Greg Minshall, Tom Lyon, and Larry Huston. *IP Switching and Gigabit Routers.* IEEE Communications Magazine, 35(1):64–69, Jan 1997. (pp 4, 5, 7, 26, 27, 63)

[Nicolaou97] Cosmos Nicolaou. *Private communication.* FORE Systems Inc., Audio/Video Systems, Cambridge, UK, 1997. (p 114)

[Niehaus97] Douglas Niehaus, Abdella Battau, Andrew McFarland, Basil Decina, Henry Dardy, Vinay Sirkay, and Bill Edwards. *Performance Benchmarking of Signaling in ATM Networks.* IEEE Communications Magazine, 35(8):134–143, August 1997. (p 113)

[Nilson95] Gunnar Nilson, Fabrice Dupuy, and Martin Chapman. *An Overview of the Telecommunications Information Networking Architecture.* Proceedings TINA Conference, Melbourne, Australia, pages 1–12, February 1995. (pp 17, 121)

[Olsen92] Robert T. Olsen and Lawrence H. Landweber. *NoW (No Waiting) Virtual Channel Establishment in ATM-like Networks.* Technical Report 1082, University of Wisconsin-Madison, 1992. (pp 4, 26)

[OMG95a] OMG. *CORBA Security*, 1995. OMG Document Number 95-12-1. (p 48)

[OMG95b] OMG. *The Common Object Request Broker: Architecture and Specification*, 1995. Version 2.0. (pp 16, 23)

[OMG97] OMG. *A Discusssion of the Object Management Architecture.* Available from: `http://www.omg.org/`, January 1997. (p 16)

[ORL97] ORL. *OmniORB version 2.* Available from: `http://www.orl.co.uk/omniORB/omniORB.html`, 1997. (p 63)

[Otway95a] Dave Otway. *The ANSA Binding Model.* Technical Report APM.1392.01, APM Limited, Castle Park, Cambridge, UK, 1995. (p 23)

[Otway95b] Dave Otway. *Streams and Signals.* Technical Report APM.1393.02, APM Limited, Castle Park, Cambridge, UK, 1995. (p 23)

[Park97] J.T. Park, J.H. Lee, J.W. Hong, Y.M. Kim, and S.B. Kim. *A VPN Management Architecture for Supporting CNM Services in ATM Networks.* In A. Lazar, R. Saracco, and R. Stadler, editors, *Integrated Network Management V*, pages 44–57. IFIP & IEEE, Chapman & Hall, May 1997. (p 131)

[Peg97] *Operating system support for distributed interactive multimedia.* Project overview available from: `http://www.cl.cam.ac.uk/Research/SRG/pegasus/`, 1997. (p 24)

[Perros96] Harry G. Perros and Khaled M. Elsayed. *Call Admission Control Schemes: A Review.* IEEE Communications Magazine, 34(11):82–91, November 1996. (pp 71, 138, 139)

[Pratt94] Ian Pratt and Paul Barham. *The ATM Camera V2 (AVA200).* University of Cambridge Computer Laboratory - ATM Document Collection 3, March 1994. (p 22)

[Rathgeb97] Erwin P. Rathgeb, Wolfgang Fischer, Christian Hinterberger, Eugen Wallmeier, and Regina Wille-Fier. *The MainStreetXpress Core Services Node - A Versatile ATM Switch Architecture for the Full Service Network.* IEEE Journal on Selected Areas in Communication, 15(5):795–806, June 1997. (pp 95, 139)

[Rekhter96] Yakov Rekhter, Bruce Davie, Dave Katz, Eric Rosen, and George Swallow. *Tag Switching Architecture Overview.* Internet Draft: draft-rfced-info-rekhter-00.txt, September 1996. (p 120)

[Rooney97a] S. Rooney. *An Innovative Control Architecture for ATM Networks.* In A. Lazar, R. Saracco, and R. Stadler, editors, *Integrated Network Management V*, pages 369–380. IFIP & IEEE, Chapman & Hall, May 1997. (pp 6, 17, 35, 40, 121, 124)

[Rooney97b] S. Rooney. *Connection Closures: Adding Application-Defined Behavior to Network Connections.* Computer Communications Review, 27(2):74–88, April 1997. (p 124)

[Rooney97c] Sean Rooney. *The Structure of Switch Independent ATM Control.* PhD thesis, Cambridge University, Computer Laboratory, UK, 1997. In preparation. (pp 25, 146)

[Russel97] Raymond Russel and Fergal Toomey. *Private communication.* Dublin Institute of Advanced Studies, 1997. (p 94)

[Sathaye95] Shirish S. Sathaye. *ATM Forum Traffic Management Specification Version 4.0.* ATM Forum Technical Committee - Contribution 95-0013, 1995. (pp 13, 38, 139)

[Schneider93] J.M. Schneider, T. Preub, and P.S. Nielsen. *Management of Virtual Private Networks for Integrated Broadband Communication.* Computer Communications Review (SigComm '93), 24(4):224–237, October 1993. (p 131)

[Schoenwaelder] Juergen Schoenwaelder. *Scotty - Tcl Extensions for Network Management Applications.* Available from: `http://wwwsnmp.cs.utwente.nl/~schoenw/scotty/`. (p 55)

[Shumate94] Scott W. Shumate. *A Performance Analysis of an Off-Board Signalling Architecture for ATM Networks.* Master's thesis, University of Kansas, 1994. (p 63)

[Smith96] J.M. Smith, D.J. Farber, C.A. Gunter, S.M. Nettles, D.C. Feldmeier, and W.D. Sincoskie. *SwitchWare: Accelerating Network Evolution (White Paper).* Available from: `http://www.cis.upenn.edu/~jms/SoftSwitch.html`, June 1996. (p 134)

[Srinivasan97] S. Srinivasan and M. Veeraraghavan. *DIVA: A DIstributed & dynamic VP management Algorithm.* In A. Lazar, R. Saracco, and R. Stadler, editors, *Integrated Network Management V*, pages 287–298. IFIP & IEEE, Chapman & Hall, May 1997. (p 140)

[Stiller95] Burkhard Stiller. *A survey of UNI Signalling Systems and Protocols for ATM Networks.* ACM Computer Communications Review, 25(2):21–33, April 1995. (p 98)

[Tennenhouse96a] David L. Tennenhouse and David J. Wetherall. *Towards an Active Network Architecture.* Computer Communications Review, 26(2):5–18, April 1996. (p 134)

[Tennenhouse96b] D.L. Tennenhouse, S.J. Garland, L. Shrira, and M.F. Kaashoek. *From Internet to ActiveNet.* Request for Comments, January 1996. Available from: `http://www.sds.lcs.mit.edu/activeware/`. (p 134) [van der Merwe95] Kobus van der Merwe and Shaw-Cheng Chuang. *Support for Open Multi Service Networks.* pages 60–68. Regional International Teletraffic Seminar, Pretoria, South Africa. Available from: `http://www.cl.cam.ac.uk/users/jev1001/`, September 1995. (pp i, 29)

[van der Merwe96] J.E. van der Merwe and I.M. Leslie. *Switching System.* UK Patent (pending) 9621248.5, 1996. (pp i, 35)

[van der Merwe97a] J.E. van der Merwe and I.M. Leslie. *Service Specific Control Architectures for ATM.* Accepted for publication in IEEE JSAC, 1997. (pp i, 40, 126)

[van der Merwe97b] J.E. van der Merwe and I.M. Leslie. *Switchlets and Dynamic Virtual ATM Networks.* In A. Lazar, R. Saracco, and R. Stadler, editors, *Integrated Network Management V*, pages 355–368. IFIP & IEEE, Chapman & Hall, May 1997. (p 126)

[Veeraraghavan95] Malathi Veeraraghavan, Thomas F. La Porta, and Wai Sum Lai. *An Alternative Approach to Call/Connection Control in Broadband Switching Systems.* IEEE Communications Magazine, 33(11):90–96, November 1995. (pp 4, 26, 63)

[Veeraraghavan97] Malathi Veeraraghavan. *Connection Control in ATM Networks.* Bell Labs Technical Journal, 2(1), 1997. Winter edition. (pp 26, 121)

[VINCE] VINCE. *VINCE:Vendor Independent Network Control Entity.* Available from: ftp://hsdndev.harvard.edu:pub/mankin. (pp 21, 58)

[Vinoski97] Steve Vinoski. *CORBA: Integrating Diverse Applications Within Distributed Heterogeneous Environments.* IEEE Communications Magazine, 35(2):46–55, February 1997. (pp 15, 16)

[Woodruff97] G. Woodruff, N. Perinpanathan, F. Chang, P. Appanna, and A. Leon-Garcia. *ATM Network Resources Management using Layer and Virtual Network Concepts.* In A. Lazar, R. Saracco, and R. Stadler, editors, *Integrated Network Management V*, pages 315–326. IFIP & IEEE, Chapman & Hall, May 1997. (p 140)

[Wray94] Stuart Wray, Tim Gluart, and Andy Hopper. *Networked Multimedia: The Medusa Environment.* IEEE Multimedia, pages 54–63, 1994. Winter edition. (p 137)

[Yemini96] Yechiam Yemini and Sushil da Silva. *Towards Programmable Networks.* IFIP/IEEE International Workshop on Distributed Systems: Operations and Management, October 1996. Available from: `http://www.cs.columbia.edu/~dasilva/content/netscript/`. (p 134)

[Zhang93] Lixia Zhang, Stephen Deering, Deborah Estrin, Scott Shenker, and Daniel Zappala. *RSVP: A New Resource ReSerVation Protocol.* IEEE Network, 7(5):8–18, September 1993. (pp 106, 120)