gNET: An ATM LAN Signalling Protocol

Version 1.0

Technical Report

Grenville J. Armitage
(email: g.armitage@ee.mu.OZ.AU)

February 1994

Department of Electrical Engineering, University of Melbourne.
Grattan Street, Parkville 3052, Australia

ISBN 85867 077 1

ABSTRACT

An Asynchronous Transfer Mode (ATM) LAN is envisaged as being a collection of ATM end-nodes and switch nodes. The physical links between switches and end-nodes may be point-to-point, or some form of shared medium. This technical report describes a signalling protocol designed to support environments where multiple ATM nodes may share physical switch ports and access media. Signalling units are contained within a single cell payload. The addressing model supports 64 bit node addresses, and 12 bit AAL User identifiers. Connection requests may specify Virtual Channel Connections (VCCs) with any AAL type, null AALs, or even create Virtual Path Connections (VPCs). One-to-many multicast VCCs and VPCs may be supported by the specified set of signalling units. A framework is provided for future extensions to support a variety of QoS based connection admission schemes, using restricted QoS indications.

CONTENTS

1. Introduction.......... 1
    1.1 Scope.......... 1
    1.2 Format.......... 1
    1.3 Acknowledgment.......... 1
2. The gNET ATM-LAN Protocol.......... 3
    2.1 The initial goals and constraints.......... 4
    2.2 Addressing AAL Users, and describing the gNET End Node.......... 5
    2.3 The Signalling channel, gNET Signalling Units, and switch nodes.......... 7
    2.4 Managing the traffic on a virtual connection.......... 9
        2.4.1 Specifying Quality of Service.......... 10
        2.4.2 Policing AAL User traffic.......... 10
        2.4.3 A Two-tier peak rate specification.......... 11
        2.4.4 An Average rate and CDV specification.......... 12
        2.4.5 Further QoS types.......... 13
    2.5 Building a Unicast Virtual Connection.......... 14
    2.6 Building Multicast Virtual Connections.......... 15
    2.7 Managing the allocation of VPIs and VCIs.......... 16
    2.8 Network Initialisation.......... 16
    2.9 Why support for Shared-media?.......... 16
    2.10 Details of GSUs, and inter node signalling sequences.......... 16
    2.11 References.......... 17
A. Appendix A - The gNET Protocol Description.......... 19
    A.1 gNET services to AAL User processes.......... 19
        A.1.1 Management of outgoing connections.......... 19
        A.1.2 Establishing a termination point for incoming connections.......... 20
    A.2 Inter-GSE signalling traffic.......... 21
        A.2.1 Node Initialisation.......... 22
        A.2.2 VPI and VCI management.......... 23
            A.2.2.1 Requesting a VPI/VCI pair or block.......... 23
            A.2.2.2 Returning a VPI/VCI pair or block.......... 24
            A.2.2.3 Caching VPI/VCI blocks.......... 25
            A.2.2.4 Combining the PVS and ICS functions.......... 25
        A.2.3 Naming Signalling points.......... 26
        A.2.4 Establishing outgoing virtual connections.......... 27
            A.2.4.1 Unicast Virtual Channel Connections.......... 27
            A.2.4.2 Multicast connections.......... 30
            A.2.4.3 Allocating a VPI/VCI to a link.......... 35
            A.2.4.4 Unicast and Multicast Virtual Path Connections.......... 35
        A.2.5 Removing virtual connections.......... 36
            A.2.5.1 Sender initiated removal from a unicast connection.......... 36
            A.2.5.2 Receiver initiated removal from a unicast connection.......... 38
            A.2.5.3 Dropping leaf nodes from a multicast connection.......... 39
            A.2.5.4 Leaf node initiated removal.......... 40
            A.2.5.5 Releasing a VPI/VCI no longer in use.......... 40
        A.2.6 Recovering from lost GSUs.......... 40
            A.2.6.1 Loss during connection establishment.......... 41
            A.2.6.2 Loss during connection removal.......... 42
            A.2.6.3 Clearing ICS nodes after signalling failure.......... 43

A.2.7 Network control of ADLI Policing Units............................ 45
A.2.7.1 Peak cell rate control.. 46
A.2.7.2 Other traffic shaping functions.............................. 47
A.3 Routing between ICS nodes.. 47
A.4 Pre-allocated ATIA and AEPI values.. 48
A.5 Distributing the ICS, PVS, and switch fabric functions....................... 49
A.6 Support for other signalling systems.. 50
A.6.1 Interworking gNET with the wider B-ISDN........................ 50
A.6.2 Overlaying gNET with new signalling services..................... 50
A.7 Many to One multicast at the AAL User level.................................. 52
A.8 GSU types, and error codes during connection establishment................ 52
A.8.1 Numerical values for GSU Type field................................ 52
A.8.2 Error codes returned in CONNECT_NAK.......................... 53
A.9 References.. 53

1. Introduction.

1.1 Scope.

This technical report describes the gNET signalling protocol developed as part of research into building ATM LANs, and implementing ATM Adaptation Layer services in software on conventional workstations. Version 1.0 defines some basic services, and provides a framework for future evolution of the gNET protocol to provide additional services.

A subset of gNET was implemented in a SunOS 4.1.1 environment, and operated over a novel ATM network using Ethernet as the physical cell transport layer. This report only covers the network model and signalling elements of gNET. It does not describe the trial network in any detail, nor the software components. Broad descriptions may be obtained from the following documents:

G.J. Armitage, K.M. Adams, "Using the Common LAN to Introduce ATM Connectivity", accepted for publication in Proc. IEEE 18th Conference on Local Computer Networks, Minneapolis, MN, September 19-22, 1993.

G.J. Armitage, K.M. Adams, "Prototyping an ATM Adaptation Layer in a Multi-Media Terminal", Int. J. of Digital and Analog Communications Systems, Vol.6, No.1, p.3-14, Jan-Mar 1993.

The earliest work on this project can be found in:

G.J. Armitage, K.M. Adams, "Implementing an ATM Adaptation Layer in a Multimedia Terminal", Proc. Australian Broadband Switching and Services Symposium, Melbourne, July 1991, pp.316-323.

G.J. Armitage, K.M. Adams, "Architecture of a Multimedia Desktop Workstation", Proc. Australian Video Communications Workshop, Melbourne, July 1990, pp.76-85.

Section 2 introduces gNET itself, and Appendix A provides detailed descriptions of gNET signalling elements and procedures.

1.2 Format.

The pages of each section are numbered so that they may be photocopied into a doublesided format after printing. The odd page numbers correspond to right hand pages. Each section starts on a right hand page.

1.3 Acknowledgment

This work was supported by Telecom Research Laboratory, Australia, under TRL contract 7064.

2. The gNET ATM-LAN Protocol.

One of the key goals of this research project was to develop a prototype ATM LAN complete with multimedia workstations. The most fundamental part of any ATM LAN is the signalling protocol that can enable end users to dynamically manage end to end virtual connections. During the early stages of this project (around 1990) there were no ATM based signalling protocols in existence, despite indications that CCITT would be modifying its ISDN protocol, Q.931, as a stop-gap measure. Other ATM LAN protocols and architectures have emerged in the mean-time, such as Fore System's SPANS, and the Multi-Service Network Architecture (MSNA) at the University of Cambridge. However, they had not yet been developed or released when our need arose. gNET was designed to fill the gap, and support our vision of ATM LAN architecture and services.

The basic design concepts upon which gNET is based were first described in a report to Telecom Research Laboratories [2-6], and published in July 1992 [2-1]. In mid-1993 a paper outlining an early version of gNET was released to the general research community [2-2]. This technical report supersedes all previous description of gNET.

An ATM LAN is envisaged as being a collection of ATM end-nodes and switch nodes, as shown in Figure 2.1.1. Some switches have clusters of end-nodes hanging off them (e.g. Switch 1 and Switch 4); others provide switching services for connections between different regions (Switch 2), and some may provide an element of both (Switch 3). The physical links between switches and end-nodes may be point-to-point, or some form of shared medium (e.g. Node 7 and Node 8) [2-3].

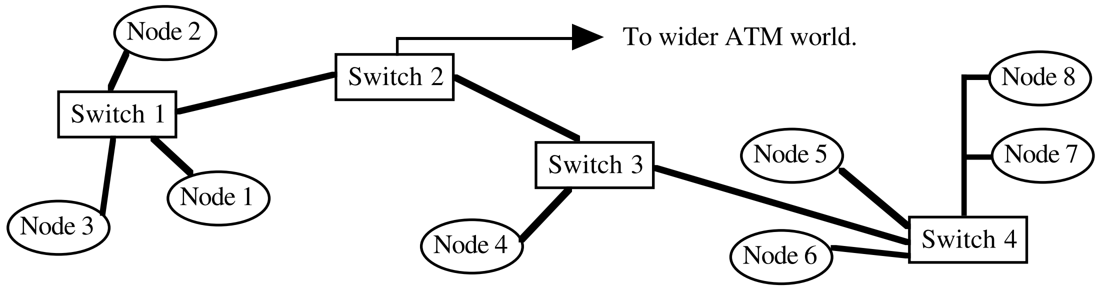


Figure 2.1.1

Defining the behaviour of a signalling protocol involves discussion of three different areas. These are:

(a) Services offered to AAL User processes.
(b) Interaction between local node's gNET signalling entity and ATM/AAL hardware.
(c) Peer to peer communication between gNET signalling entities across the network.
(d) Control of switch hardware by resident gNET signalling entities.

The ATM LAN model we wish to provide to higher layer services is described in (a). It covers things such as node addressing, QoS specification, and connection admission policies. The function of (c) is to describe how the underlying network of ATM links and switch nodes may be utilised to provide the services described in (a). Point (b) involves the model of a local node, and what functions a gNET entity must perform in response to messages under points (a) and (c). Lastly, (d) covers the manner in which gNET entities control the behaviour of basic switch fabrics to create end to end virtual connections.

2.1 The initial goals and constraints.

The broad aim of gNET was to provide only basic support for higher layer services (AAL Users). Connection establishment functions were governed by the following constraints:

- Virtual connections were to be **unidirectional**.
- Connection admission would be based on a restricted set of traffic parameters.
- Each gNET entity and its associated ATM Interface is identified by a single 64 bit **ATM Terminal Interface Address** (ATIA).
- AAL Users on a given node are identified by a 16 bit **AAL Endpoint Protocol Identifier** (AEPI).
- One-to-many unidirectional multicast connections to be supported with appropriate switch hardware.
- Both Virtual Channel Connections (VCCs) and Virtual Path Connections (VPCs) should be supported, given appropriate switch hardware.

The decision to support unidirectional virtual connections occurred at a time when CCITT had not resolved the question in favour of bidirectional connections. Unidirectional connections were subsequently retained because it allowed gNET to remain as simple as I required. AAL Users are left with the responsibility of establishing bidirectional communication as they see fit.

The desire to emulate conventional LAN services led to connection endpoints being identified by <ATIA, AEPI> pairs. The ATIA identifies a particular ATM layer (i.e. Physical interface) - analogous to the MAC layer address used in Ethernet, or the 64 bit addresses used in the I.364 Connectionless service PDU [2-5]. The AEPI identifies the AAL User process that is to be the destination of the virtual connection. It is analogous to Ethernet's 'EtherType' field, or the 'Higher Layer Protocol Identifier' (HLPI) of the I.364 PDU.

At the physical network level the following architecture had to be considered:

- gNET signalling entities on end nodes and switch nodes are peers.
- Support for 'shared medium' links, where multiple nodes may share access to a common bus or medium, and can 'see' each others cell transmissions.
- A single VPI/VCI allocated for peer to peer gNET signalling traffic on any given switch port (and hence the end node(s) attached to the port).

The decision to support 'shared medium' lines had a crucial impact of the design of gNET. The need for every gNET entity to communicate on the same VPI/VCI led to the adoption of gNET signalling units (GSUs) that fitted within a single cell. In turn the scarcity of space in a cell payload enforced the requirement for gNET to provide only rudimentary service.

Support for VPCs was included in order to ensure gNET supplied the widest possible variety of basic connection management services. Although most LAN services are not likely to require VPCs at this stage, it would unnecessarily limit gNET's usefulness to exclude the functionality.

As gNET was being built to support our own multimedia terminal, it needed to support the peculiarities of our design. The most significant issue here was our use of 2 bits in the VPI/VCI to indicate whether incoming traffic was destined for the codec or the SPARCStation. As the particular bit pattern was fixed into hardware at the receiving workstation, the following constraint was applied to gNET:

- During connection establishment, the receiving node on any link of a virtual connection must specify the VPI/VCI it wishes cells to arrive on.

The next few sections will elaborate on these decisions.

2.2 Addressing AAL Users, and describing the gNET End Node.

One of the driving forces behind the development of gNET was the desire to emulate conventional LAN services at the MAC layer (described in sections 4 and 5). This requirement suggested a similar, simple, endpoint addressing scheme should be implemented to encourage the transfer of LAN based network layers over to a gNET based ATM network.

An ATM virtual connection originates from, and terminates on, specific instances of AAL service. Typically each instance of an AAL function is associated with a particular higher layer AAL user connection. Multiple AAL users may be served through a common ATM layer, which would be associated with a given physical layer interface. Figure 2.2.1 shows how the ATIA and AEPI identify the elements in a typical end node's protocol stack.

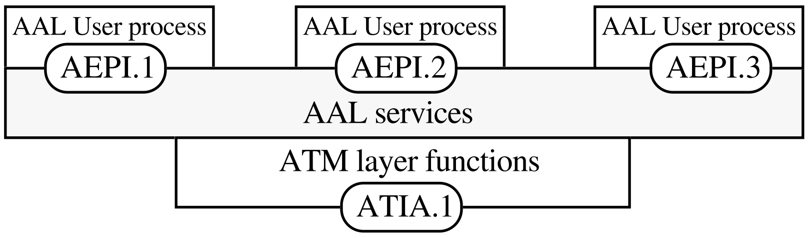


Figure 2.2.1

Figure 2.2.2 shows the two subfields of the AEPI - a 4 bit 'AAL service type', and a 12 bit 'Protocol ID'. This enables an <ATIA, AEPI> pair to explicitly identify the ATM endpoint, AAL User process, and AAL type required at the end of the connection. Each AAL Service type may support up to 4096 AAL User processes - corresponding roughly to Network layer or LLC layer entities in traditional LAN protocol stacks.

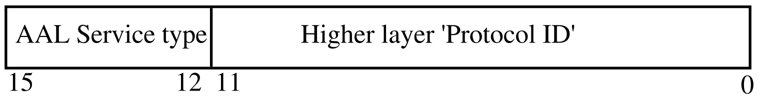


Figure 2.2.2

The following table shows the AAL Services associated with currently defined AAL Service type field values. New service types can be added as new AAL services at the SAR and CPCS level are developed and clarified. The 'Null AAL' service means that cells are passed in their entirety directly to the AAL User without any processing.

AAL Service type	Meaning
0	VCC: Null AAL
1	VCC: AAL1 - CBR circuit mode emulation.
2	VCC: AAL2 - VBR circuit mode emulation.
3	None defined.
4	VCC: AAL3/4 - Packet/data traffic.
5	VCC: AAL5 - Packet/data traffic
6..14	None defined.
15	VPC: Null AAL. ‘Protocol ID’ is further subdivided.

The same protocol ID may be used for different AAL services, as this still results in a unique AEPI. For example a videophone application may have two entities requiring connections - a video codec using AAL2, and a control entity using an AAL5 link. Both may use a protocol ID of 0x7F0 (for example), but the AEPIs would be 0x27F0 and 0x57F0 respectively. gNET treats both endpoints as quite independent - the choice of the protocol ID field in the AEPI is entirely up to the gNET users.

Termination of a VPC at a null AAL is considered an AAL Service by gNET. Type 15 specifies that a VPC should be established between the AAL Users. The destination AAL User is responsible for any further VCI based demultiplexing and processing of the incoming stream. VPCs consume significant amounts of the VPI space on inter-switch links. Bits 10 and 11 of the AEPI are interpreted as a ‘VCI subset’ field, to enable an AAL User to indicate when it only requires a subset of the VCI range. The ‘Protocol ID’ field is reduced to 10 bits, as shown in figure 2.2.3.

0xF	VCI subset	Higher layer 'Protocol ID' 0 .. 0x3FF
15 12	11 10	9 0

Figure 2.2.3

The following table gives the interpretation of each VCI subset field value. gNET refers to VPC requests that specify VCI subset values 0, 1, or 2 as ‘subset-VPCs’.

VCI subset	Meaning
0	VCIs limited to the range 0 to 0xF.
1	VCIs limited to the range 0 to 0xFF.
2	VCIs limited to the range 0 to 0xFFF.
3	VCIs may cover the full range of 0 to 0xFFFF.

The gNET end-node must contain an ATM layer, one or more AAL services, and a gNET signalling entity (GSE). The gNET entity must have means for registering higher layer entities, binding them to AEPIs, and dynamically creating or destroying internal data paths between them and their required AAL service. These functions may be contained within one physical unit (in the case of a basic workstation simply running IP over ATM), or distributed (when different presentation media require AAL services implemented in geographically distributed hardware).

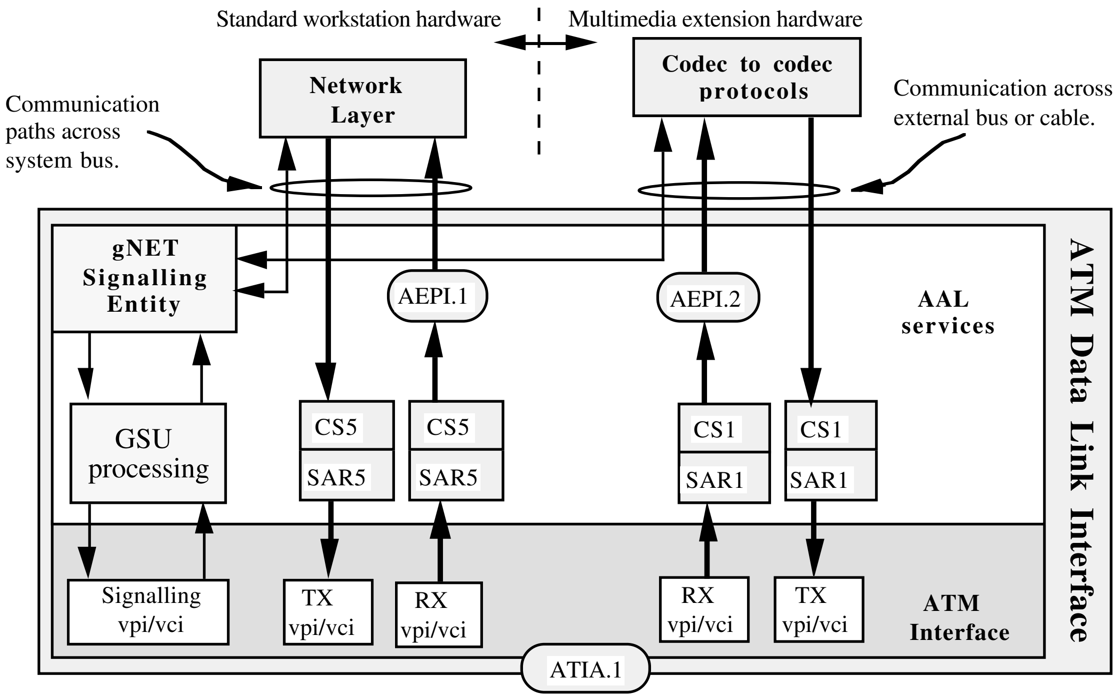


Figure 2.2.4

Figure 2.2.4 is an example of what I have termed the ATM Data Link Interface (ADLI). It shows two established AAL5 connections (one in, and one out) for a conventional network layer, and two AAL1 connections (one in, and one out) for a videophone connection that terminates on an external piece of codec hardware.

An ADLI need not support all AAL Service types. If an AAL User indicates that it needs an unsupported service, connection requests will be denied with an appropriate error indication.

gNET allows very simple evolution of an ATM LAN from network protocol support to multimedia support because of the way AEPIs define connection endpoints and AAL service. A basic ATM LAN might initially use ADLIs that provide only AAL5 services, to support a wide area IP network. As codec and ADLI technology improves, machines within this LAN may be upgraded individually to support both AAL5 and AAL1 services.

ADLI upgrades will involve no change to GSEs within the rest of the ATM LAN. Switch nodes will only require modification if they do not already support the new functions, e.g. multicasting or virtual path switching.

2.3 The Signalling channel, gNET Signalling Units, and switch nodes.

Figure 2.3.1 shows the most general shared-medium configuration that gNET was expected to support. With shared medium a form of ‘distributed switching’ occurs between each node’s ATM layer. Each node receives a copy of each cell sent from the switch port, and filters out the unwanted ones based on their VPI/VCI fields. Conversely, cells being transmitted to the switch port by any one node will also be received by the other nodes. The more conventional ‘one node per port’ configuration is treated as a subset of figure 2.3.1.

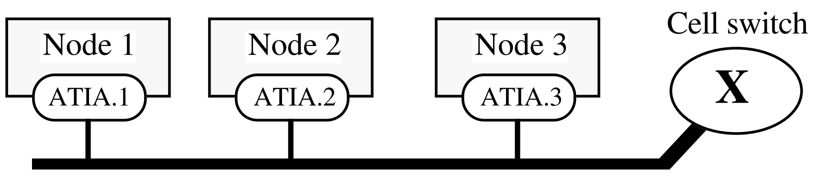


Figure 2.3.1

To address the problem of initial signalling between end-nodes and ‘the network’, ITU-T defined a ‘metasignalling’ channel - a globally unique and predefined VPI/VCI value [2-4]. gNET uses a similar scheme by defining a single, fixed signalling channel (VPI 0, VCI 16) for the use of all GSEs on a given physical link. ‘The network’ is represented by a GSE residing on the local switch node, which uses the common signalling channel to exchange requests and responses with GSEs in the end nodes.

When applied to situations such as Figure 2.3.1, the signalling channel behaves as a broadcast channel - the GSEs on all attached nodes receive each other’s signalling traffic. As there is no synchronisation between the emission of cells by each node, multi-cell signalling messages would become scrambled, making it difficult for receiving gNET entities to reconstruct the original message.

The solution provided by gNET is to restrict signalling messages to a single cell. (This is the same approach pursued by CCITT SGXI for the Metasignalling protocol, as they encountered the same problem. These developments were, however, quite independent).

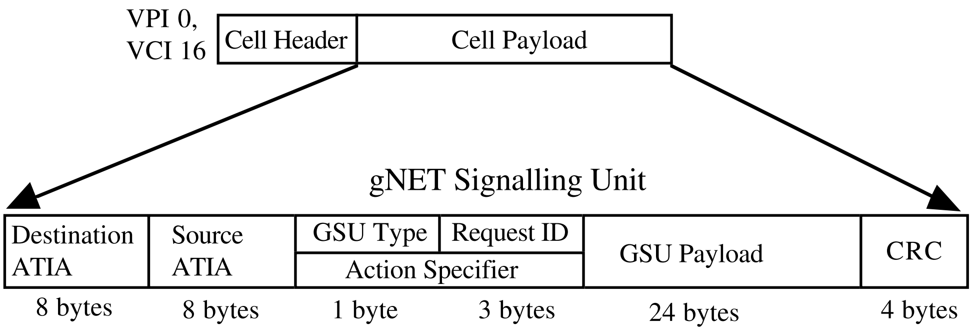


Figure 2.3.2

Figure 2.3.2 shows the gNET Signalling Unit (GSU), containing source/destination ATIAs and a 32 bit ‘Action Specifier’ - made up of an 8 bit ‘type’ field, and a 24 bit ‘Request ID’ field. The ‘type’ field specifies what gNET signalling action is being requested or replied to. The ‘Request ID’ field differentiates between new and repeat GSUs of the same ‘type’. The payload contents depend on the signalling function being requested by the ‘type’ field.

Functionally related fields in the GSU header and payload are aligned on 32 bit boundaries - the goal being to create a GSU that is ‘silicon friendly’. This should increase the ease with which a gNET entity may be implemented in hardware, possibly within the network interface containing the ATM layer and AAL services.

Given that most ATM LANs will need to support at least AAL5, it appears reasonable to believe a signalling entity embedded into an ATM LAN interface may make use of the AAL5 CPCS 32 bit CRC function. This is the reason for the 4 byte CRC is placed at the end of the GSU (for implementation ease - it is hardly likely that such a large CRC is necessary for a 44 byte payload). An earlier version of gNET encapsulated the GSU within an AAL3/4 SAR_PDU for a similar reason - I planned to use the AAL3/4 CRC check to combat errors on the signalling channel. This was changed as it became apparent that AAL3/4 was second choice to AAL5 for LAN applications.

GSUs are exchanged by filling the 'destination' field with the ATIA of the destination GSE, filling the 'source' with its own ATIA, and transmitting the cell on the signalling channel. Whenever a GSE receives a GSU it inspects the Destination ATIA field for a match to its own ATIA. If it matches, or the GSU is a broadcast message, the rest of the GSU is inspected.

The switch nodes shown in figure 2.1.1 represent two distinct entities that work together to interlink end nodes. The first entity is the actual switch function, described in section 6.1.4. The second is the resident GSE that monitors the signalling channel on each port for GSUs being sent to it by other GSEs. Together they form what I have termed the **Intelligent Cell Switch** (ICS).

When an AAL User requests an outgoing virtual connection to a particular <ATIA, AEPI>, the local GSE sends the request GSU to the ICS node it is attached to. The ICS is then responsible for establishing a route to the destination, and eventually returning an indication of success or failure to the local GSE. If it is successful, the ICS will return the VPI/VCI on which the AAL User's connection must be carried. The ICS is assumed to have set up the appropriate mappings within its switch fabric. Inter-ICS signalling is also carried out using the gNET signalling channel and single cell GSUs.

Figure 2.3.3 shows the relationship between end nodes and ICS from the GSE's perspective, making clear the localised nature of the gNET signalling channel. Only end user virtual connection traffic passes straight through the switch fabric (a variant on this theme, using a distributed ICS over a collection of switch fabrics, is discussed in Appendix A).

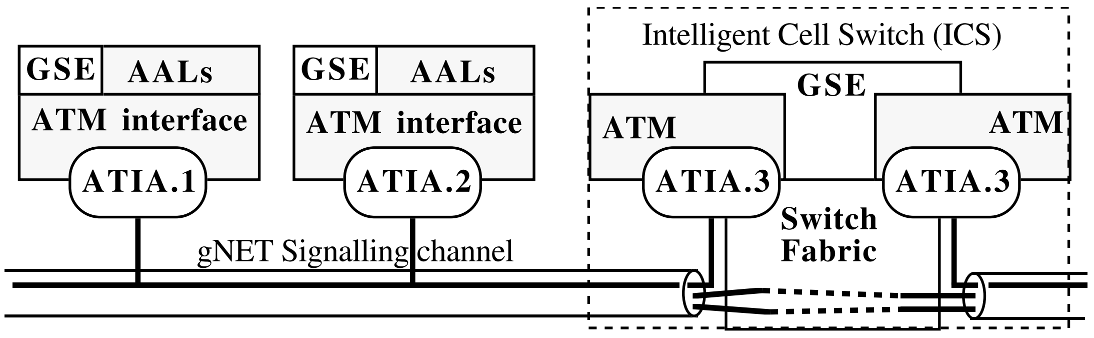


Figure 2.3.3

2.4 Managing the traffic on a virtual connection.

gNET does not specify any particular algorithms relating connection admission and quality of service considerations. It provides a limit set of facilities which may be used to support connection admission based 'preventative control'. It also assumes the existence of policing units in the end node that may be activated and altered while a virtual connection is active. This allows a broad form of reactive control to be implemented.

When a VPC is established, the traffic parameters requested are taken to apply to the aggregate cell traffic on all VCIs within the VPC.

2.4.1 Specifying Quality of Service.

Every request for a virtual connection must carry some indication of the network resources required by the AAL User. gNET supports a restricted range of ‘Quality of Service’ (QoS) descriptions, in part because only 32 bits of the 24 byte GSU payload are available for this purpose. Figure 2.4.1 shows the breakdown of the field into a 4 bit ‘QoS Type’ and 28 bit ‘QoS Parameter’.

32 bit QoS field from CONNECT_RQ

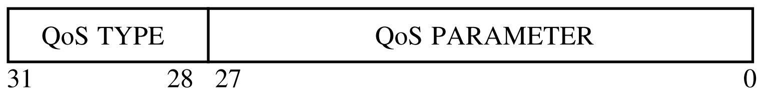


Figure 2.4.1

This provides 16 different ways of interpreting a 28 bit parameter - sufficient to describe a suitably selected range of traffic parameters. Selecting the range of traffic parameters involves qualitative and quantitative analysis - qualitative to decide the type of traffic parameter being controlled, and quantitative to decide what range of values the user needs to specify for that parameter.

Each QoS Type will have a set of fixed characteristics, with the QoS Parameter field only being used to provide the AAL User with a limited range of variations.

The QoS field does not define or directly control the end user’s actual offered traffic, it provides gNET with an *indication* of an end user’s traffic ‘shape’ or requirements. Using its knowledge of previously accepted connections running through it, each ICS node uses the QoS field to determine whether the additional traffic may be supported within the desired constraints.

The end user is quite free to send traffic at rates within the bounds indicated by the QoS they requested. For example, a codec to codec connection could be established with a QoS requesting a cell rate entirely unrelated to the rate the codecs will actually use. The next section describes how the GSE might also use the QoS field to initialise policing functions on each virtual connection.

2.4.2 Policing AAL User traffic.

When traffic shaping occurs at a point remote from the source, only fairly crude methods such as cell dropping and limited buffering may be implemented. gNET allows for the existence of end user policing units, which will be implemented between the AAL function and ATM layer within each ADLI, and will operate on a per-connection basis.

Implementation within the end user node allows for tighter coupling between the policing unit an the AAL User. Mechanisms such as Leaky Bucket and Cell Spacing may be augmented by the exertion of ‘back-pressure’ on the AALs, and hence the AAL User applications. Figure 2.4.4 shows the arrangement.

Local policing units are initialised by the GSE when a connection has been established, based on the QoS field supplied by the AAL User. If the shaping function indicated by the QoS field is not specifically supported by the policing unit an approximate setting is calculated and used.

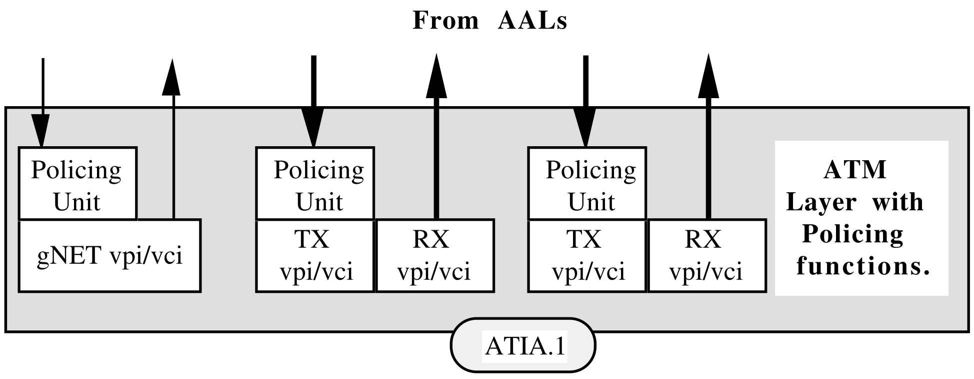


Figure 2.4.4

Appendix A provides further details of gNET's mechanism for dynamic modification of a connection's policing function, and its applications.

2.4.3 A Two-tier peak rate specification.

Some data transfer applications will find the network's basic or default cell loss/delay characteristics quite acceptable. They will only require the ability to specify or guarantee a particular peak throughput. Two issues face this type of AAL User - how much it costs to reserve channel capacity, and what minimum bit rate does the application need in order to provide useable service.

My two-tier model makes the following assumptions:

- A conservative 'deterministic multiplexing' admission scheme is being used.
- Two cell rates are specified by the user; the maximum bandwidth they wish to pay for, and the minimum they believe they can operate with.
- The local GSE can control cell transmission rate on a given virtual connection, either through direct control of the AAL User or through a policing unit.
- The cell rate allowed by 'the network' may be varied at any time if newer connections require additional bandwidth.

The network will generally allow the connection to operate up to the requested (top tier) rate. The minimum (second tier) rate may be applied when the network is attempting to cope with an unusually high number of simultaneous connections. When the number of connections drops off, the network will restore active connections back to their original rates. Figure 2.4.2(a) shows the Type 0 QoS field, with the QoS Parameter broken into two 14 bit subfields representing the Peak and Minimum rates.

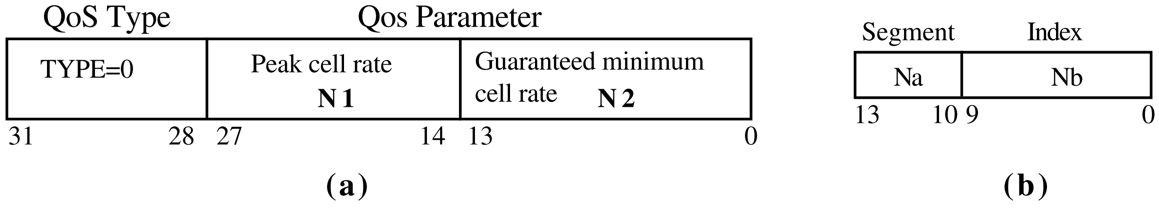


Figure 2.4.2

The rates are specified in cells/second. Matching desired bit rates to required cell rates is an issue for the AAL User.

The services supported by ATM connections are projected to span a vast range of effective bit rates, from 100's of kbit/sec to Gbit/sec speeds. This range cannot be covered by a linear function of a 14 bit field, which can only represent 16384 distinct values.

Na	Base	Cell/sec (max)	Mbit/sec (max)
0	0	1023	0.3746
1	1024	3070	1.1243
2	3072	7164	2.6235
3	7168	15352	5.6221
4	15360	31728	11.619
5	31744	64480	23.613
6	64512	129984	47.602
7	130048	260992	95.578
8	261120	523008	191.531
9	523264	1047040	383.438
10	1047552	2095104	767.250
11	2096128	4191232	1534.88
12	4193280	8383488	3070.13
13	8387584	16768000	6140.63
14	16776192	33537024	12281.6
15	33553408	67075072	24563.6

The novel solution used by gNET is to apply a discrete, non-linear mapping between N1 and the actual cell rate. Figure 2.4.2(b) shows the subdivision of the 14 bit field into a 4 bit 'segment' field (Na) and a 10 bit 'index' field (Nb).

The table above shows how Na varies between 0 and 15, and specifies a particular linear sub-segment of the N1 to cell rate mapping. Within a given sub-segment Nb varies between 0 and 1023. The equivalent bit rates are calculated as the number of payload bits per second (48 bytes per cell). The actual cell rate is calculated by:

$$\boxed{\text{Cell Rate (cells/second)} = \text{Base} + (\text{Nb} * 2^{\text{Na}})}$$

The total range covers rates from 1 cell/second to over 67 million cells/second (approximately 24 Gbit/sec). Nb allows a cell rate to be selected with a granularity of under 0.01% of the maximum for a given sub-segment. For example, rates between 1535 Mbit/sec and 3070 Mbit/sec (when Na=12) may be specified with a granularity of 1.5Mbit/sec. This seems more than adequate for bursty services using a Type 0 QoS.

Converting from cell rates to Na and Nb values simply involves looking up the table for the Na value, and calculating the Nb value giving the closest higher cell rate.

Specifying either N1 = 0 or N2 = 0 is considered an invalid QoS and will be NAKed.

The Type 0 QoS field supports a simplistic but useful connection admission and congestion control scheme.

2.4.4 An Average rate and CDV specification.

Another major source of traffic on a gNET network will be CBR video or audio links for video phone and conference applications. Earliest implementations will run across AAL1, which provides 'constant bit rate' adaptation services and is further advanced than AAL2 for variable bit rate services. For these AAL Users the major issue is cell delay variation, with the network being interested in the nominal cell rate required. This becomes QoS Type 1, shown in Figure 2.4.3.

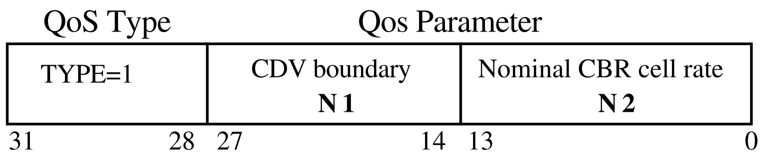


Figure 2.4.3

The nominal cell rate required for the service is specified by the bottom 14 bits, using the same format as QoS Type 0. The rate indicated by N2 must be equal to, or greater than, the rate that the AAL user will inject cells into the network. This ensures that sufficient resources are allocated along each hop the connection traverses. As mentioned at the start of this section, the AAL user is free to send traffic within the constraints of the requested QoS. For example, a 4 Mbit/sec codec might select Na=3, Nb=1023 (peak rate of around 5.6 Mbit/sec).

Specifying a Type 1 QoS with N2 = 0 is considered an invalid QoS, and the connection request will be refused.

As an ATM connection traverses switch nodes subject to transient congestion conditions, an originally smooth stream of cells may become ‘clumped’ or ‘bursty’. N1 remains to convey the bounds that the end user wishes to place on variations in the inter-cell arrival time (or CDV).

AAL User’s will implement local buffering between the network and the codec. When a connection is first established, the receiving end will partially fill its buffer before starting to pass cells to the codec. This enables the buffers to:

(a) Provide cells to the codec when the rate from the network drops below the consumption rate of the codec, and
(b) Soak up cells when the rate from the network exceeds the consumption rate of the codec.

The sender will specify N1 with the receivers buffering in mind. As the effect of CDV is cumulative, the simplest thing for N1 to represent was the ‘maximum number of cell times that the incoming cell stream may gain or lose with respect to a notional fixed emission rate’. Assuming the buffers are designed to be half full on average, N1 is set to be half the physical size of the buffer.

The lowest bound on the length of a ‘cell time’ is derived from the approximate cell rate indicated by N2. How the network uses this information is undefined in this report. Specifying N1 = 0 implies the network’s default CDV is acceptable, whatever that happens to be.

2.4.5 Further QoS types.

This report does not cover any further QoS types, although further work on AAL User services will uncover new categories that may be implemented.

Unfortunately the most important class of LAN service, the bursty data source that benefits from statistical multiplexing at the switch, is also the hardest to classify. The Type 0 QoS can quite happily support such traffic, for example by setting N1 and N2 to the same value and simply allowing the source to burst between zero and the specified cell rate. However, this will not allow the network to take advantage of statistical multiplexing gains in its connection admission procedure. To do this a new QoS type is required that characterises the 'burstiness' of the traffic, as well as some indication of its mean or peak rates.

However, bursty traffic models are not sufficiently well understood in the literature for a restricted subset to be defined for gNET. This will have to wait for further work in the area.

2.5 Building a Unicast Virtual Connection.

A VCC or VPC exists in two senses within a gNET system. In the 'ATM sense' it is represented by the concatenation of individual links represented by particular VPI/VCIs between switches. In the 'gNET sense' it is represented by a set of logical mappings within GSEs along the path of the connection.

When an AAL User issues a request for a virtual connection to a given <ATIA, AEPI>, its local GSE transmits a request GSU to the 'nearest' ICS node. This GSU identifies the required destination, the source <ATIA, AEPI>, and the QoS to be associated with the connection. The request GSU is then propagated from ICS node to ICS node, until it is finally delivered to the end node with the desired ATIA. At this point the end node may accept or reject the request, with the response backtracking along the path taken by the original request. Figure 2.5.1 attempts to illustrate this.

The nodes marked "ADLI" are end user nodes that may initiate or accept a virtual connection. In this case an AAL User on ADLI 1 has requested a connection to an AAL User on ADLI 3. The request is represented by the CONNECT_RQ. As each ICS node processes the request it tentatively allocates resources that the connection will need if it is accepted.

The acknowledgment (CONNECT_ACK) carries back with it the VPI/VCI that is to be associated with the virtual connection on each link. ADLI 3 may specify a particular VPI/VCI to ICS 3, and ICS 3 may specify a different VPI/VCI to ICS 2. The virtual connection is considered to be confirmed and active when every ICS has a VPI/VCI/port mapping.

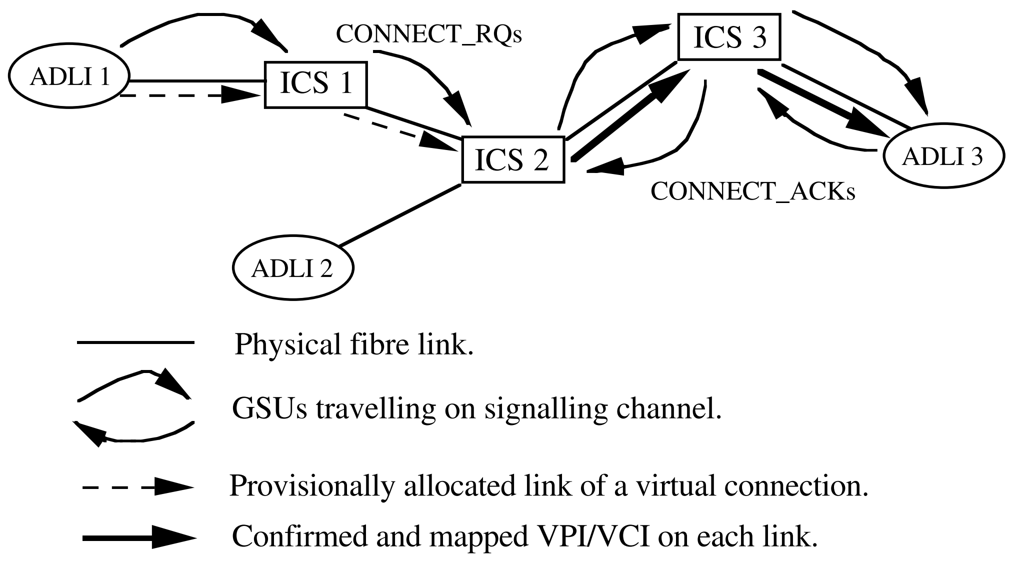

CONNECT_RQs
ADLI 1
ICS 1
ICS 2
ICS 3
ADLI 3
CONNECT_ACKs
ADLI 2
Physical fibre link.
GSUs travelling on signalling channel.
Provisionally allocated link of a virtual connection.
Confirmed and mapped VPI/VCI on each link.

Figure 2.5.1

If the desired QoS cannot be supported at any ICS node (e.g. because of previously accepted connections), the ICS may pre-emptively reject the request by generating a CONNECT_NAK GSU. This would retrace the request's path back to ADLI 1, releasing the provisionally allocated resources in each ICS it passes through. If ADLI 3 could not accept the request (e.g. the request AEPI did not exist) it would also respond with a CONNECT_NAK.

Each ICS node uses the total of confirmed and provisional resource allocations when deciding if a new connection may be supported. This can lead to temporary incorrect denial of service to other connection requests. Section A discusses the management of unicast connections, and this source of error, in further detail.

Each ICS along the path of the connection request checks for AAL Service type 15 in the AEPI, and handles VPCs slightly differently. Optimisation of inter-ICS node VPI usage, based on the VCI subset indicated in the AEPI, is decided on a hop-by-hop basis. This is described in greater detail in Appendix A.

2.6 Building Multicast Virtual Connections.

gNET can support multicast connections in an ATM LAN where the ICS nodes have multicast-capable switch fabrics. The following characteristics apply to gNET's multicast connection establishment facilities:

(a) Only one-to-many unidirectional multicast is supported at the cell level. One ADLI is the 'root node', with one or more ADLIs as 'leaf nodes'.
(b) Leaf nodes may be added or removed at any time, but only by the root node.
(c) A single QoS is specified and applied to all links in the multicast connection.
(d) Some limitations apply to leaf nodes sharing the same port of a switch and participating in the same multicast group.

Points (a) and (b) provide the basic definition of a multicast service. As gNET is intended to provide only basic support it is considered sufficient that only the root node be able to modify the multicast group. More complex functions, such as leaf node initiated connection, should be provided at the AAL User process level rather than by the gNET protocol.

Point (c) represents an obvious and necessary limitation. A multicast connection carries traffic of a particular service type - which immediately dictates the QoS required to convey the cells on each and every path established.

Point (d) is a consequence of gNET's basic design. It allows the destination ADLI to specify the VPI/VCI it wishes to accept incoming traffic on. If more than one ADLI share the same port, and wish to receive the same multicast traffic, then a copy of each cell is required for every destination ADLI sharing the port. Each copy will have the VPI/VCI requested by the ADLI they are destined for.

From the perspective of a switch fabric, a VPC is merely a VCC with a restricted bit pattern in the cell header acting as a connection identifier. A switch that can multicast on VPI/VCI should also be able to multicast on the VPI alone. The gNET signalling architecture requires no special extensions to support VPC multicast connections. The only restriction is that all leaf nodes are restricted to the VCI subset allocated to the first leaf node's VPC.

Multicast connections are a simple extension of the basic unicast connection. From the perspective of destination AAL Users, they are indistinguishable. The details of their construction, and the associate GSUs, are provided in Appendix A.

2.7 Managing the allocation of VPIs and VCIs.

An important part of gNET is its definition of a centralised mechanism for allocating VPIs and VCIs on particular links. Every ADLI and ICS node that accepts an incoming connection needs to know what VPI/VCI combinations it may choose on the incoming physical medium, without clashing with already active virtual connections.

The solution is for every GSE to access a **Permanent VPI/VCI Server** (PVS) through the signalling channel. The PVS behaves as a central clearing house, handing out unused VPI/VCIs when requested, and ensuring that no clashes occur. It has its own ATIA, and 'exists' on the gNET signalling channel in the same way as the ICS node. Only one PVS serves a given physical medium. Although they are quite independent entities, a PVS may be physically implemented within an ICS node, to serve each port into the ICS.

2.8 Network Initialisation.

Finally, gNET provides a solution to the problem of a node's initial signalling connection to 'the network'. The four most important pieces of information for a GSE are:

- Its own ATIA,
- The VPI/VCI of the gNET signalling channel,
- The ATIA of at least one local ICS node, and
- The ATIA of the local PVS node.

The local ATIA and signalling channel can be preconfigured into each interface. The identity of the ICS node(s) and PVS node serving the medium are established by transmitting a 'probe' GSU to the broadcast ATIA. The ICS and PVS nodes are expected to reply to these probes, with their own ATIAs carried in the 'source ATIA' fields of the reply GSUs.

2.9 Why support for Shared-media?

gNET was designed to support shared-medium architectures from the very beginning. This enables a single signalling protocol to support an ATM network built around a variety of different media. One example is 'wireless ATM', which is not a novel idea anymore (e.g. [2-8], page 5). The shared medium 'connected' to one port of the switch in figure 2.3.1 is a radio channel covering a localised area, potentially with mobile units acting as end nodes. gNET's architecture is able to support the wireless ATM nodes without any changes at all.

The limitations of designing a signalling protocol around a point-to-point fibre model have been avoided.

2.10 Details of GSUs, and inter node signalling sequences.

Detailed descriptions of the signalling traffic between GSEs, and the formats of the GSUs, are contained in the appendix of this report. The GSU formats and field usage have been developed specifically for gNET, and not copied from any other signalling work. Although designated an appendix, it contains important further details of how gNET provides an elegant solution to the problem presented.

2.11 References.

[2-1] G.J. Armitage, K.M. Adams, “Towards Shared ATM Fibres, A Pseudo Metasignalling Protocol, and ATM Lans.”, Proc. Australian Broadband Switching and Services Symposium, Melbourne, July 1992, p.271.

[2-2] G.J. Armitage, K.M. Adams, “gNET - A Multi-Media ATM LAN Protocol.”, April 1993. Available by anonymous FTP from ftp.ee.mu.OZ.AU, as “/papers/kma/gja.gNET.1st.sub.IEEE.Network.ps.Z” from September 1993.

[2-3] CCITT Draft Recommendation I.413, “B-ISDN User Network Interface”, Geneva, June 1990.

[2-4] CCITT SGXVIII Revised recommendation I.361, "B-ISDN ATM Layer specification", June 1992.

[2-5] CCITT SGXVIII Temporary document 58, Recommendation I.364, “Support of Broadband Connectionless Data Service on B-ISDN”, June 1992.

[2-6] K.M. Adams, G.J. Armitage, K.E. Forward, S. Liew, I.A. Morrison, F.A. Wilson, “Research and Development of an Experimental Broadband Integrated Services Terminal”, Stage 6 Progress Report, April 1992, pp.18-36.

[2-7] Fore Systems Inc., “SPANS: Simple Protocol for ATM Network Signalling”, August 1992. (Also available by anonymous FTP from cell-relay.indiana.edu, as "/pub/cell-relay/docs/current/spans.ps.Z”).

[2-8] David J. Greaves, Derek McAuley, “Private ATM Networks”, Proceedings of IFIP WG 6.4 workshop ‘Protocols for High Speed Networks’, held at the Swedish Institute of Computer Science, Summer 1992.

A. Appendix A - The gNET Protocol Description.

This appendix provides a more detailed description of the actual inter-GSE signalling traffic that supports the gNET model. In addition, the GSE to AAL User interface is broadly defined by a set of ‘C’ functions. Finally, the issues of sharing routing information between ICS nodes, supporting other signalling systems, and alternative PVS and ICS architectures will be covered.

Concepts outlined in section 2 will not be repeated in this appendix.

A.1 gNET services to AAL User processes.

How a particular AAL User will be connected to the AAL services, and how it will interact with the local gNET entity, is going to be implementation specific. Only a basic functional specification is provided here for the services the gNET must provide AAL Users. These fall into two categories:

(a) Management of outgoing virtual connections (unicast and multicast).
(b) Creation of termination point for incoming virtual connections.

As gNET provides only unidirectional connections these two categories are functionally independent. Category (a) involves the ‘on demand’ creation and removal of virtual connections to specified <ATIA, AEPI> pairs. Category (b) involves the establishment of a local AAL/AAL-User interface such that gNET may associate incoming virtual connections with particular AAL Users.

This section will describe the required functions in terms of a ‘C’ language library of functions, where the AAL and GSE have been provided as part of the underlying kernel of a workstation on the ATM LAN.

A.1.1 Management of outgoing connections.

The two most basic functions required by AAL Users are unicast (point-to-point) connection establishment and removal. An AAL User wishing to establish an outgoing virtual connection must first establish a local connection (access point) to the AAL services managed by the gNET entity. The gNET entity is then requested to associate this local access point with an ATM virtual connection to a specified <ATIA, AEPI>, with a specified QoS.

The local access point may be created in similar way to a file or socket being opened:

```
fd = gNET_connect()
```

‘fd’ now holds an access point identifier which the gNET entity uses to identify the AAL User. With this established the most complicated function is connection establishment:

```
gNET_create(fd, Remote ATIA, Remote AEPI, Local AEPI, QoS)
```

The local gNET entity may allow AAL Users to specify their local AEPI value in the connection request, however the minimum information needed is ‘fd’, the remote ATIA, the remote AEPI, and the desired QoS. This function will only return success if the GSE has been able to establish the requested virtual connection. The function should return error codes indicating failure reasons if the connection could not be established.

The AAL User is not entirely free in the choice of AEPI. The Local AEPI field is used to decided what transmit side AAL processing to provide, and the Remote AEPI field is used to set up the receive side processing. Obviously gNET must disallow requests to connect incompatible AEPI types, or requests that require AAL processing not supported by the underlying ADLI implementation. The only variation on this rule is when the Remote AEPI specifies service type 0 (null AAL) - any source AAL service may connect to such a destination.

`gNET_send(fd, data)`

The AAL User process may send data across the connection identified by ‘fd’ as soon as `gNET_create()` returns.

`gNET_disconnect(fd)`
`gNET_destroy(fd)`

The AAL User originating the connection may disconnect it at any time. Only the local identifier ‘fd’ needs to be provided. A further function may be implemented, gNET_destroy(), to disconnect both the ATM connection and the local access point simultaneously.

gNET also offers basic support for creating one-to-many multicast connections. A local AAL User may explicitly associate multiple destinations with a particular local access point ‘fd’. Subsequent data sent with gNET_send() will be sent to all destinations without further intervention.

An initial unicast link to one member of the desired multicast group is created with gNET_create(), followed by special requests to add new members to the group. The initial unicast destination has no special significance, it is only used to establish a route out of the local node that further connections may be added to.

`gNET_add(fd, Remote ATIA, Remote AEPI)`

gNET_add() adds the specified destination to the group of nodes receiving data transmitted through access point ‘fd’. New members to the group inherit the QoS and Local AEPI specified in the original gNET_create().

`gNET_drop(fd, Remote ATIA, Remote AEPI)`

Single destinations may be removed with gNET_drop(). Removing all destinations (in any order) is equivalent to a gNET_disconnect(). The local GSE must keep track of all Remote ATIA/AEPI pairs associated with a given local access point. If a gNET_disconnect() is issued by the AAL User it can then be converted (by the GSE) into a sequence of gNET_drop()’s on each member of the multicast group.

A.1.2 Establishing a termination point for incoming connections.

Before gNET can accept an incoming virtual connection an AAL User must have established a local access point to the AAL service entity, and registered a particular AEPI it wishes to be associated with. Again the function `gNET_connect()` is used to establish the local AAL/AAL-User interface.

`gNET_bind(fd, Specific AEPI)`

This creates an association between the local access point ‘fd’ and any incoming virtual connections destined for <Local ATIA, Specific AEPI>. Once this association is made the

GSE will accept incoming virtual connections without further involvement from the AAL User. This is similar to LANs such as Ethernet, where packet arrival is unpredictable once the physical cabling is connected and transceivers switched on. `gNET_bind()` will fail if the AEPI specifies an AAL service that the underlying ADLI implementation cannot support.

`gNET_recv(fd, Local_buffer)`

The AAL User must be able to retrieve completely reassembled AAL_SDUs in their entirety with a single call. How this is implemented cannot be specified here - AAL_SDUs may be buffered by the AAL entity until retrieved, or 'posted' to the AAL User process through some form of message passing mechanism. Where the AAL User has requested 'Null AAL' or VPC service, `gNET_recv()` returns complete ATM cells.

An ADLI must support the convergence of multiple virtual connections on to a single <Local ATIA, Specific AEPI>. This will occur whenever multiple remote nodes each request an incoming connection to the same <Local ATIA, Specific AEPI>. Each connection will be allocated its own VCC through the network, and instance of the AAL. However, each instance of the AAL will pass the reassembled AAL_SDUs to the common access point 'fd' representing the <Specific AEPI>.

The receiving AAL User will have to use information carried within the AAL_SDUs themselves to deduce their source.This provides service similar to LANs such as Ethernet, where packets may arrive randomly from multiple sources without prior knowledge by the LLC or Network layer. It also provides a 'fake' many-to-one multicast capability - at least from the AAL User's perspective. However, because it uses a new virtual connection through the network for each one it is not multicast at the VPI/VCI level.

Terminating consecutive connection requests on different local access points to the same AAL User may be provided as an implementation dependent extension.

gNET must also provide some mechanism for AAL Users to enquire how many incoming virtual connections are currently terminating on its access point(s), and where they are from. It may also provide a notification service to let an AAL User know when a virtual connection appears or is removed by the source node, but this is implementation dependent.

Receiver initiated disconnection of a virtual connection is supported within the gNET protocol (section A.2.5.2), so it is reasonable to offer this to the AAL User. However, as multiple virtual connections may converge on a given local AAL/AAL-User connection, the mechanism will be implementation specific.

Special support for multicast connections is not required. The leaf nodes of a one-to-many multicast connection only 'see' a single incoming unicast connection. Multicast specific processing only occurs at the originating node and intervening ICS nodes.

A.2 Inter-GSE signalling traffic.

The inter-GSE signalling traffic may be broken into a collection of subgroups:

- Node initialisation and identification.
- VPI/VCI management.
- Connection establishment.
- Connection removal.
- Traffic management.

This is the order they will be dealt with in section A.2. Multicast and unicast connections will be covered under the same subsection.

A.2.1 Node Initialisation.

The boot-time problem for all signalling protocols is how an end-node establishes its initial signalling connection to 'the network', or the local 'B-ISDN exchange'.

gNET uses a technique that is analogous to the IP suite's various Address Resolution Protocol (ARP) mechanisms over Ethernet [A-1, A-2], and AppleTalk's dynamic node ID assignment, AARP, and routing table maintenance schemes [A-3]. A 'broadcast' GSU is transmitted on the signalling channel to elicit a response from either the ICS or PVS nodes.

All destination bits set to '1'	64 bits	ATIA of probing GSE
ATIA of probing GSE	64 bits	ATIA of ICS
<WHOISICS>	8 bits	<IAMICS>
Null ID	24 bits	Null ID
Empty Payload		Empty Payload
WHOISICS		IAMICS

Figure A.2.1.1

Figure A.2.1.1 shows the request/reply GSUs used to locate an ICS node. A destination ATIA of all '1's is the broadcast address, all GSEs on the local medium will receive the WHOISICS request GSU. Any connected ICS nodes will reply with an IAMICS, and the probing node then extracts the ICS's ATIA from the source field of the reply. The same mechanism is also used to locate the PVS node serving the medium - using WHOISPVS and IAMPVS messages respectively (Figure A.2.1.2).

All destination bits set to '1'	64 bits	ATIA of probing GSE
ATIA of probing GSE	64 bits	ATIA of PVS
<WHOISPVS>	8 bits	<IAMPVS>
Null ID	24 bits	Null ID
Empty Payload		Empty Payload
WHOISPVS		IAMPVS

Figure A.2.1.2

ICS nodes may also use these two probe GSUs to identify if other ICS and PVS nodes are attached to their ports. ICS nodes also need to keep track of the ADLIs that are directly attached to them (in order to provide routing in formation to surrounding ICS nodes). The method of establishing this information must allow for timely updating of routing tables when end-nodes are booted, removed, or shifted to new ports.

Two mechanisms are used by the ICS to identify directly attached ADLIs. Initially each ICS creates a table of <source ATIA, port> entries for all WHOISICS traffic it responds to. A timer is then attached to each table entry, which is reset whenever the ICS node sees further gNET signalling traffic from that node. Until the timer expires, the ADLI is considered to be still attached and active.

If the timer expires, the ICS actively probes the end-node to ensure it is still there. This is performed by the transmission of the HEARTBEAT GSU (Figure A.2.1.3) to the ADLI whose timer has expired. The HEARTBEAT carries no information except the sender's ATIA, and serves no other purpose than to identify that the sender is still active. An end-node must always respond to a HEARTBEAT with a reply HEARTBEAT back to the ICS. Receipt of a HEARTBEAT reply constitutes gNET signalling traffic at the ICS, so the timer attached to the <ATIA, port> entry is reset. If a number of HEARTBEAT probes go unanswered, the ICS removes the <ATIA, port> entry from its routing table and propagates this information to its peers.

ATIA of destination GSE	64 bits
ATIA of source GSE	64 bits
<HEARTBEAT>	8 bits
0	24 bits
Empty Payload	

HEARTBEAT

Figure A.2.1.3

A.2.2 VPI and VCI management.

There are six GSUs associated with the management of VPIs and VCIs through the PVS. Three of them provide VPI/VCI request service, and the other three provide for the 'return' of VPI/VCIs to the PVS. There is no particular time at which a GSE should request or return VPI/VCI pairs - they may cache a few before connection requests arrive, or request them only for the duration of a connection.

A.2.2.1 Requesting a VPI/VCI pair or block.

Figure A.2.2.1 shows the two GSUs associated with a successful PVS request from a GSE. Each INEEDVCI from a given GSE has a unique Request ID, which the PVS uses to differentiate between repeat and new INEEDVCIs.

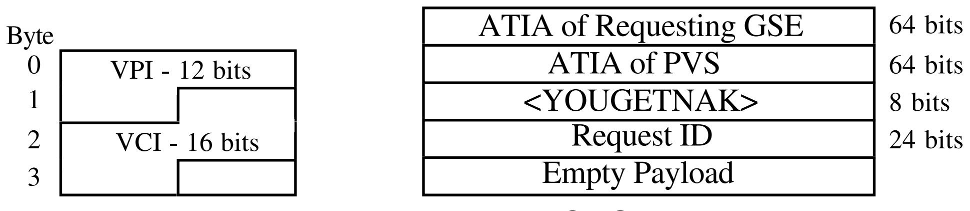
ATIA of PVS	64 bits	ATIA of Requesting GSE
ATIA of Requesting GSE	64 bits	ATIA of PVS
<INEEDVCI>	8 bits	<YOUGETVCI>
Request ID	24 bits	Request ID
Start VPI/VCI	32 bits	Start VPI/VCI
End VPI/VCI	32 bits	End VPI/VCI
INEEDVCI		YOUGETVCI

Figure A.2.2.1

Currently the PVS supports three different types of request, depending on the contents of the Start and End VPI/VCI fields in the INEEDVCI request:

(a) Start and End are zero.
(b) End VPI/VCI is zero, Start VCI is zero, Start VPI is non-zero.
(c) Start and End VPI/VCI fields are non-zero.

A type (a) request means that the PVS is free to allocate a single VPI/VCI of whatever value it likes, and returns the result in the Start field of a YOUGETVCI. A type (b) request forces the PVS to choose any VCI it likes within the VPI specified by the Start VPI of the INEEDVCI. This may be needed by a GSE that has associated a particular VPI with a type of service, but is happy to allow the PVS to choose a VCI (An example of this is the workstation developed during this project - the GSE had to be able to request a particular VPI pattern for traffic destined to the video codec). Finally a type (c) request allows a Connection AAL to request a range of VPI/VCI pairs from the Start VPI/VCI up to, but not including, the End VPI/VCI. The allocated range is returned in the YOUGETVCI. This format may be used when a whole VPI needs to be allocated for a VPC.



32 bit VPI/VCI field.

YOUGETNAK

Figure A.2.2.2

The VPI/VCI field format is shown in Figure A.2.2.2, along with the GSU that is returned if the request could not be honoured. The top 28 bits contain the VPI/VCI in ATM cell header format. A YOUGETNAK is issued if, for example, the requested range is too large or already in use.

Loss of an INEEDVCI, YOUGETVCI, or YOUGETNAK is handled by timeouts and retransmission by the Requesting GSE. A repeat INEEDVCI must have the same Request ID as the original one. The PVS keeps track of VPI/VCI usage on the medium by <Requesting GSE, Request ID>, so repeat INEEDVCIs are easy to distinguish from new ones. If a repeat INEEDVCI is detected the PVS simply recreates its previous response from stored information.

If the GSE fails to obtain a response from the PVS after a number of repeated INEEDVCIs it should begin to signal an error condition to the management entity of the node to which the ADLI is attached. This indicates a serious problem with the network, regardless of whether it is the requests or the replies that are being lost. The fall back position is for the GSE to continue issuing requests until either communication is re-established, or the network is reset by external means.

A.2.2.2 Returning a VPI/VCI pair or block.

VPI/VCI pairs will often be scarce resources, so GSEs are encouraged to ‘release’ pairs that they no longer need. Figure A.2.2.3 shows the two GSUs associated with a successful release. The RELEASEVCI carries the Request ID of the original INEEDVCI that first allocated the specified VPI/VCI(s). The PVS replies with RELEASEACK, again carrying the same Request ID. Upon receipt of a RELEASEACK the Connection AAL removes all local references to the VPI/VCI(s).

If the PVS considers the RELEASEVCI to be invalid (for example, the specified VPI/VCI(s) were not actually allocated to that GSE, or it has already been released) it issues a RELEASEACK anyway, to ensure the GSE shuts up. As for INEEDVCI, a sequence of unacknowledged RELEASEVCIs must result in a network error being signalled.

ATIA of PVS	64 bits	ATIA of Requesting GSE
ATIA of Requesting GSE	64 bits	ATIA of PVS
<RELEASEVCI>	8 bits	<RELEASEACK>
Request ID	24 bits	Request ID
Start VCI/VPI	32 bits	Empty Payload
End VCI/VPI	32 bits	RELEASEACK
RELEASEVCI		

Figure A.2.2.3

PVS nodes may pre-empt the release of VPI/VCIs by issuing a broadcast RECLAIMRQ, shown in Figure A.2.2.4. When prompted by the reception of a RECLAIMRQ, a GSE must release any VPI/VCI pairs it was allocated, has used, but has not yet returned to the PVS.

64 bits	All destination bits set to '1'
64 bits	ATIA of PVS
8 bits	<RECLAIMRQ>
24 bits	Null ID
	Empty Payload
	RECLAIMRQ

Figure A.2.2.4

A.2.2.3 Caching VPI/VCI blocks.

The PVS has two roles - it monitors VPI/VCI use, and 'stores' unused VPI/VCIs. When a GSE requests a VPI/VCI it must also 'store' the VPI/VCI(s) it has been allocated. Requests for VPI/VCIs create gNET signalling traffic, which can be minimised by requesting blocks of VPI/VCIs, and caching them locally until needed. However, the trade off is that local memory use rises within the GSE as its VPI/VCI cache grows.

When virtual connections are rapidly being established and removed there are benefits associated with local caching. It removes the overhead of PVS requests during the connection establishment and removal phases. If virtual connections are modified irregularly then caching becomes less important.

There are many similarities between VPI/VCI allocation and memory allocation or disk caching algorithms in operating systems such as Unix. However, these decisions will be implementation dependent. In this report I only identify the following points:

- Simple 'on-demand' allocation suffices to support all gNET functions.
- Caching increases the complexity of GSE implementations.
- Cache utilisation statistics should be used to control dynamic and/or adaptive request and release procedures, to maximise the number of free VPI/VCIs available to the PVS at any one time.

A.2.2.4 Combining the PVS and ICS functions.

For complete generality the ICS and PVS services have been introduced quite separately. The PVS may be a separate physical entity, or be implemented within an ADLI or ICS node. However, in practice there is a major benefit in implementing the PVS within the ICS node - it may serve ADLIs on all ICS ports. As each physical medium needs a separate PVS entity, the PVS-capable ICS node will have to implement a separate table of VPI/VCI allocations for gNET entities visible to each port. The PVS service for each port should be independently configurable - if more than one PVS-capable ICS is connected to a single medium, only one ICS node should have its PVS entity active on that medium.

This approach reduces GSU traffic, as requests for VPI/VCIs by the ICS node can be handled through internal signalling paths instead.

A.2.3 Naming Signalling points.

The descriptions of GSU traffic associated with establishing and removing connections will be based on the simple model shown in Figure A.2.3.1. The network consists of a single ICS node with two ports and attached ADLIs.

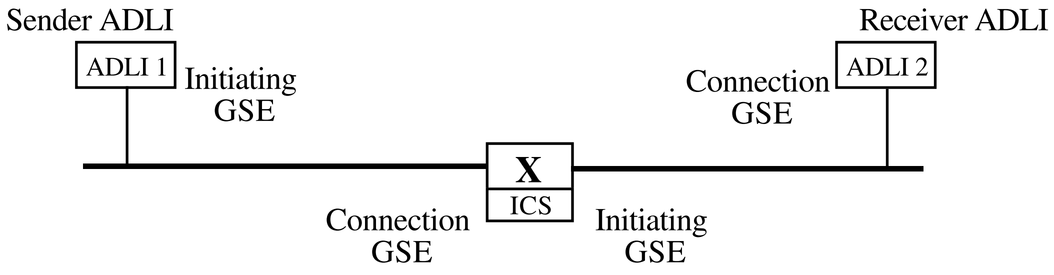


Figure A.2.3.1

For the remainder of this subsection ‘Sender ADLI’ and ‘Receiver ADLI’ represent the source and ultimate destination of traffic across the virtual connection - i.e. the combination of physical ADLI and the AAL User identified by the <ATIA, AEPI> address. The Sender ADLI initiates the establishment of a virtual connection with a connection request GSU, which propagates along each segment of fibre until it reaches the intended Receiver ADLI. GSUs travel from ICS node to ICS node, until they reach an ICS node that has a direct gNET signalling channel link to the requested ‘Receiver ADLI’.

On each leg the GSU travels from an ‘Initiating GSE’ to a ‘Connection GSE’. The Initiating GSE chooses a new destination for the GSU, the Connection GSE, which it believes is one step closer to the desired Receiver ADLI. On the first leg the Sender ADLI is also the Initiating GSE, on subsequent legs the Initiating GSE is always the output port of an ICS node. On intermediate legs the Connection GSE is the input port of an ICS node, and on the final leg the Connection GSE is the Receiver ADLI.

Each ICS node behaves as a form of ‘router’ for gNET signalling traffic. They also record state information about signalling requests they propagate. When a connection request is received, a provisional mapping is created within the gNET management entity for the virtual connection being established. Only when an acknowledgment is received travelling the other way, is the provisional mapping loaded into the switching fabric. The management entity retains all information relating to this connection, so that subsequent disconnect messages (or repeat connection requests) may be identified and handled appropriately.

A.2.4 Establishing outgoing virtual connections.

Unicast connection establishment will be covered first, followed by the extensions required to create multicast groups. The most important issue in the implementation of multicast support is the database architecture and software support, not the GSUs. Much of the following descriptions of 'how things work' assume sensible ICS management entity implementations, that perform sanity checks on whatever they do in response to the various GSUs.

The following descriptions simply assume that ICS nodes know enough about the surrounding network to make routing decisions. How ICS nodes may obtain this information is not specified in this report, although section A.3 will cover possible solutions.

A.2.4.1 Unicast Virtual Channel Connections.

An outgoing unicast connection request is generated at a Sender ADLI in response to a `gNET_create()` being issued by an AAL User. The basic request GSU is the CONNECT_RQ, shown in Figure A.2.4.1. The <ATIA, AEPI> of the desired Receiver ADLI is carried within the GSU payload, as is the <ATIA, AEPI> of the Sender ADLI. This allows connection requests to be filtered by the Receiver ADLI on the basis of the Sender ADLIs identity. The final 32 bits of the GSU payload carries the QoS field described in section 2.4.

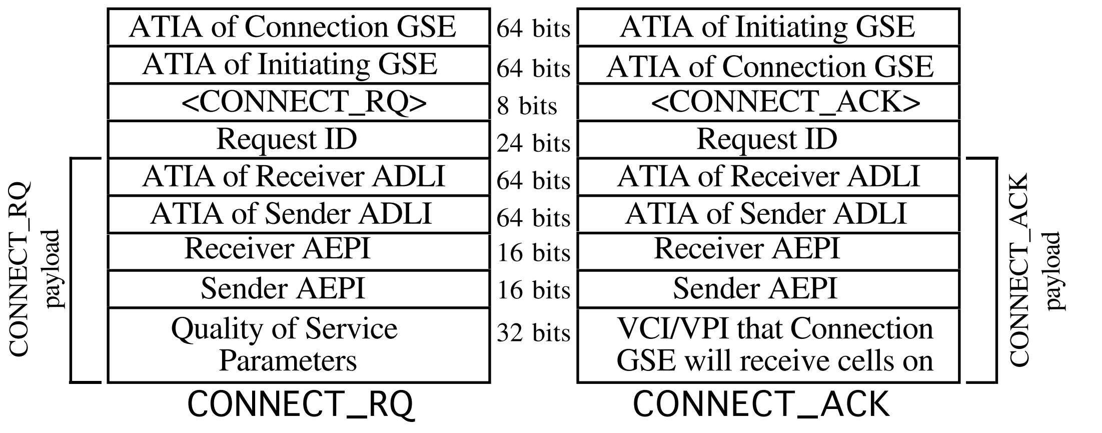


Figure A.2.4.1

Each distinct `gNET_connect()/gNET_create()` sequence to the same <ATIA, AEPI> results in a new Request ID value from the Sender ADLI. This allows ICS nodes to manage parallel virtual connections between identical <ATIA, AEPI> pairs. (If a CONNECT_RQ appears with the same Request ID field and <ATIA, AEPI> pairs as a previous one, Connection GSEs assume it to be an error recovery related retransmission, and treat it as described in section A.2.6).

Each link of a virtual connection is considered to be established when an Initiating GSE returns a CONNECT_ACK to the preceding Connection GSE. The CONNECT_ACK carries back with it the VPI/VCI that is to be associated with the virtual connection on this link. An alternative response is the CONNECT_NAK, which carries an error code indicating why the Connection GSE couldn't honour the original CONNECT_RQ. Figure A.2.4.2 shows both the VPI/VCI field format and the CONNECT_NAK.

Both ACK and NAK contain enough information for Initiating GSEs to identify the connection request being replied to - the Sender and Receiver ATIAs, and the Request ID from the CONNECT_RQ. Carrying the VPI/VCI back in the ACK specifically allows the Connection GSE on each leg to control the VPI and VCI choice, possibly based on criteria such as the requested QoS. The format of the VPI/VCI information is the same as the ATM cell header - the most significant bits first, taking the first 28 bits of the 32 bit field. The bottom 4 bits are currently reserved, and must be set to zero. Section A.8 lists the possible error codes that have been defined so far.

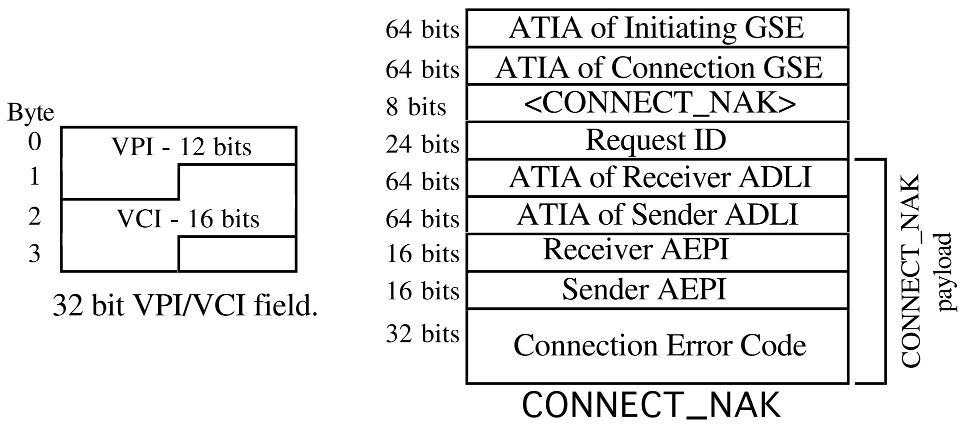


Figure A.2.4.2

Figure A.2.4.3 shows the two possible request/response sequences on a single link.

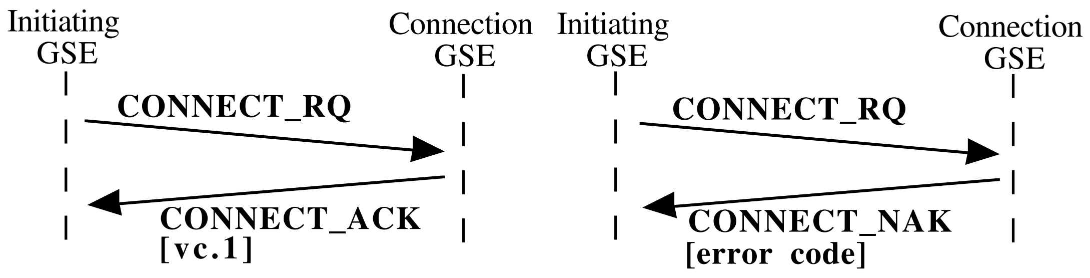


Figure A.2.4.3

A multihop connection is established by the concatenation of this basic RQ/ACK exchange. An ICS acts alternately as a Connection GSE and an Initiating GSE, as shown in Figure A.2.4.4. Acting as an Connection GSE, it accepts the CONNECT_RQ from ADLI 1. Then, acting as an Initiating GSE it propagates the CONNECT_RQ on to ADLI 2.

ADLI 2 passes back an indication that it will associate VPI/VCI pair vc.2 on its leg of this virtual connection. The ICS then specifies vc.1 as the VPI/VCI that ADLI 1 must send on for data to get through the switch and across to ADLI 2 on vc.2. When ADLI 1 receives the ACK, it may begin sending cells on vc.1 on its local fibre, which will be mapped to vc.2 on the fibre connected to ADLI 2.

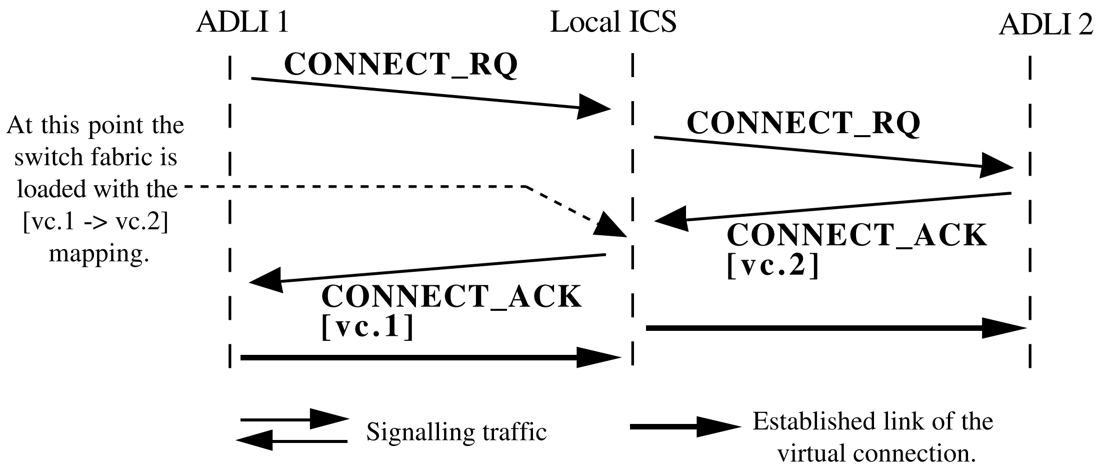

ADLI 1
Local ICS
ADLI 2
CONNECT_RQ
CONNECT_RQ
At this point the switch fabric is loaded with the [vc.1 -> vc.2] mapping.
CONNECT_ACK [vc.2]
CONNECT_ACK [vc.1]
Signalling traffic
Established link of the virtual connection.

Figure A.2.4.4

If ADLI 2 refuses the connection request for some reason, it passes back a CONNECT_NAK rather than ACK. The ICS then releases the state information it had stored regarding the request, and propagates the NAK back to ADLI 1. If the requested Quality of Service was unsupportable at a given ICS node, or some other local fault condition warrants it, the ICS itself may terminate the connection request and issue a CONNECT_NAK response. In this case the ICS doesn't propagate the CONNECT_RQ, and the intended Receiver ADLI never knows about the failed attempt.

The extension of this process to multi-hop links was outlined in section 2.5 and Figure 2.5.1.

Establishing a connection not only creates an ATM level path through the switch fabrics of each ICS, but also defines a GSU path between the two end-points. Every ICS node manages a database of all virtual connections currently active or being modified by gNET signalling activity. The database contains all the gNET information carried within a CONNECT_RQ to specify the connection, and additional information generated either locally or from the CONNECT_ACK. Figure A.2.4.5 attempts to show this relationship.

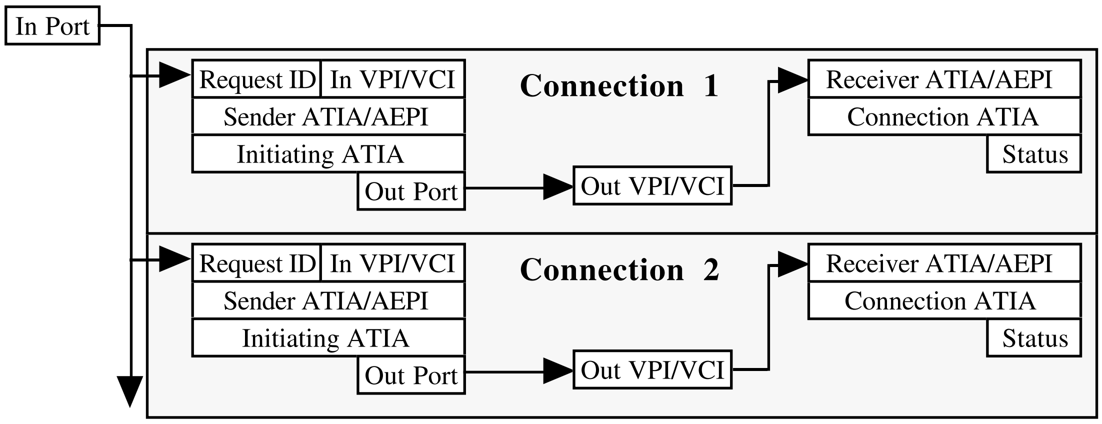


Figure A.2.4.5.

Figure A.2.4.5 shows the information indexed by incoming port, which is most useful when checking a CONNECT_RQ against pre-existing connections. The separation of the different fields is for the addition of information when a multicast connection is being supported. The ‘status’ field indicates whether the path to the indicated Receiver ADLI is active, provisionally allocated, or in the process of being deleted.

The ‘Out VPI/VCI’ field is obtained from the CONNECT_ACK that arrived at ‘Out Port’ during the connection establishment phase. The ‘In VPI/VCI’ field is filled in locally when the ICS, acting as a Connection GSE, obtains a VPI/VCI from the PVS and passes a new CONNECT_ACK out the ‘In Port’. The combination of <In Port, In VPI/VCI, Out Port, Out VPI/VCI> is copied down into the mapping tables of the underlying switch fabric in order to actually create the virtual connection through the ICS.

Finally, it is possible for a virtual connection request to be denied service by an ICS node during the correct functioning of the gNET protocol. It depends on the ICS node’s use of both confirmed and provisional resource allocations when deciding if a new connection may be supported. The following sequence of events will trigger the problem:

- A provisional allocation has been made and a CONNECT_ACK is being waited for.
- Another CONNECT_RQ arrives, requesting more resources than the ICS has available - a CONNECT_NAK is generated immediately.
- A CONNECT_NAK arrives for the original request (for whatever reason), releasing the provisionally allocated resources.

As the original connection request failed, the ICS does now have resources to support the second one. This constitutes a failure due to a race condition. For this reason an AAL User should be ready to retry a failed connection request, at least once, if the NAK carries back an Error Code suggesting the failure occured at an ICS.

A.2.4.2 Multicast connections.

From the perspective of the Sender ADLI, a multicast connection is simply a single outgoing VPI/VCI that is associated with more than one Receiver ADLI. The first virtual connection out of the Sender ADLI provides the VPI/VCI to which all subsequent leaf nodes are associated. Adding leaf nodes to a multicast VPI/VCI involves following pre-existing VPI/VCI links as far as they go towards the new node, then establishing a new unicast link for the rest of the way. The Receiver ADLI used to establish the first connection out of the Sender ADLI has no particular significance.

Three new GSUs are defined for this task, two of which are almost identical to their unicast equivalents.

The ICS management entity only needs an extension of the database shown in Figure A.2.4.5, capable of associating one incoming virtual connection with one or more outgoing virtual connections. The underlying switch fabric must be capable of accepting mapping table entries of the form:

<In Port, In VCI/VPI, Out Port, Out VPI/VCI, Out Port, Out VPI/VCI,......>

This would be taken to represent a copying action where cells coming in on a particular port and VPI/VCI would be replicated onto as many ports and VPI/VCIs as indicated. The unicast connection is simply a subset of this case.

Figure A.2.4.6 shows part of such a database. Connection 1 represents a multicast connection fanning out to 3 destinations through 2 ports of the switch. In this case the switch fabric will be required to replicate the incoming cell stream to two different ports, and with different cell headers. Connection 2 represents a multicast connection that appears to be a unicast from the perspective of the switch fabric. Although the two leaf nodes on connection 2 are not directly attached to the ICS, it must keep distinct database entries.

Connection 1 and 2 both associate more than one Receiver ATIA/AEPI with a given <VPI,VCI,port>, indicating that the cell traffic will replicate and diverge at another switch node further along the connection's path.

Receiver ADLIs are never explicitly informed that an incoming connection is a leaf node in a multicast connection. Only the basic unicast functions need to be implemented in order to be a leaf node - multicast specific functions are only required within the ICS nodes and Sender ADLIs that wish to originate a multicast connection.

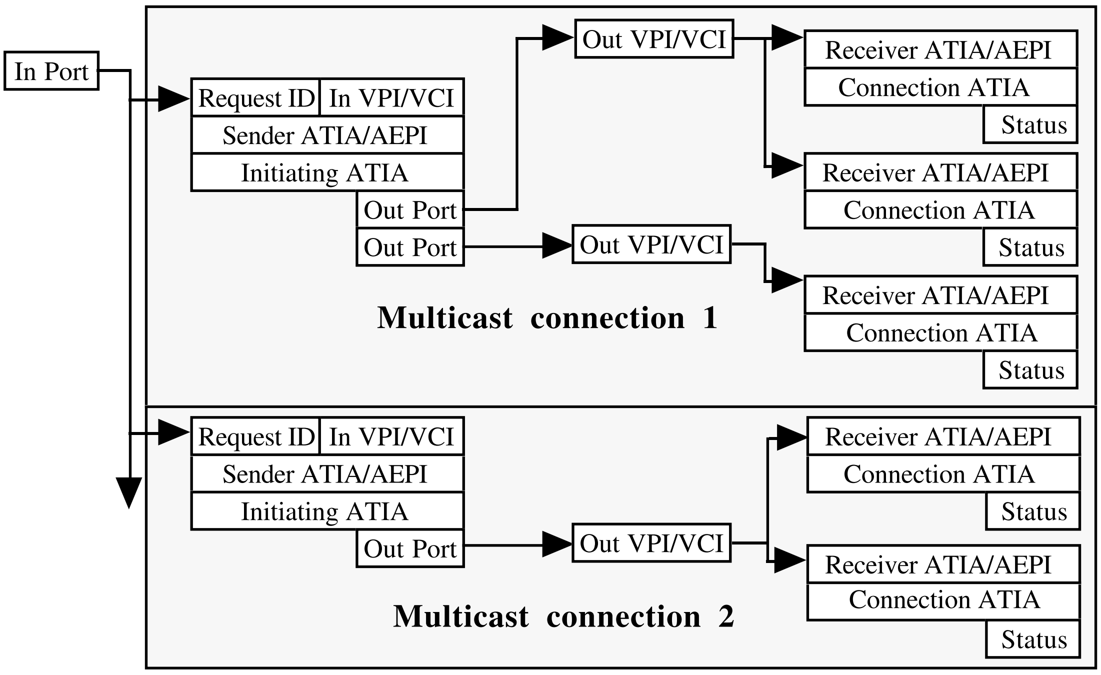


Figure A.2.4.6

On any given link an Initiating GSE uses the MULTI_ADD GSU to add a new leaf node to a pre-existing connection. Shown in figure A.2.4.7, MULTI_ADD is simply a CONNECT_RQ with the QoS field nulled. The major difference lies in how an ICS behaves when it receives a MULTI_ADD rather than a CONNECT_RQ. The connection database is searched for an entry with the same set of <In port, Initiating ATIA, Sender ATIA/AEPI, Request ID> as carried in the MULTI_ADD. This identifies the pre-existing connection being added to, and if it doesn't exist there is an error.

If the incoming link already continues on another VPI/VCI towards the Receiver ADLI specified in the MULTI_ADD, the ICS simply adds another entry to the database and propagates the MULTI_ADD. 'Connection 2' in figure A.2.4.6 is an example of this - the first connection was created with CONNECT_RQ, and the subsequent MULTI_ADD required no new route out of the switch.

ATIA of Connection GSE	64 bits	ATIA of Initiating GSE
ATIA of Initiating GSE	64 bits	ATIA of Connection GSE
<MULTI_ADD>	8 bits	<MADD_ACK>
Request ID	24 bits	Request ID
ATIA of Receiver ADLI	64 bits	ATIA of Receiver ADLI
ATIA of Sender ADLI	64 bits	ATIA of Sender ADLI
Receiver AEPI	16 bits	Receiver AEPI
Sender AEPI	16 bits	Sender AEPI
Null	32 bits	VCI/VPI that Connection GSE is receiving cells on
MULTI_ADD		MADD_ACK

Figure A.2.4.7

Ultimately the MULTI_ADD will reach an ICS where the path to the requested Receiver ADLI diverges from a pre-existing branch of the multicast connection. At this point the ICS in question copies the MULTI_ADD information into a simple unicast request (CONNECT_RQ), creating a new path to the Receiver ADLI. A database entry is added, with a 'status' field indicating that it is a multicast branch waiting for a CONNECT_ACK.

When the CONNECT_ACK arrives the underlying switch fabric is instructed to replicate cell traffic onto the specified VPI/VCI. The ICS then passes back a MADD_ACK so that preceding ICS nodes and the Sender ADLI know the request was successful. The MADD_ACK has exactly the same format as a CONNECT_ACK. If the connection request fails, a MADD_NAK is returned instead (the GSU format is identical to the CONNECT_NAK).

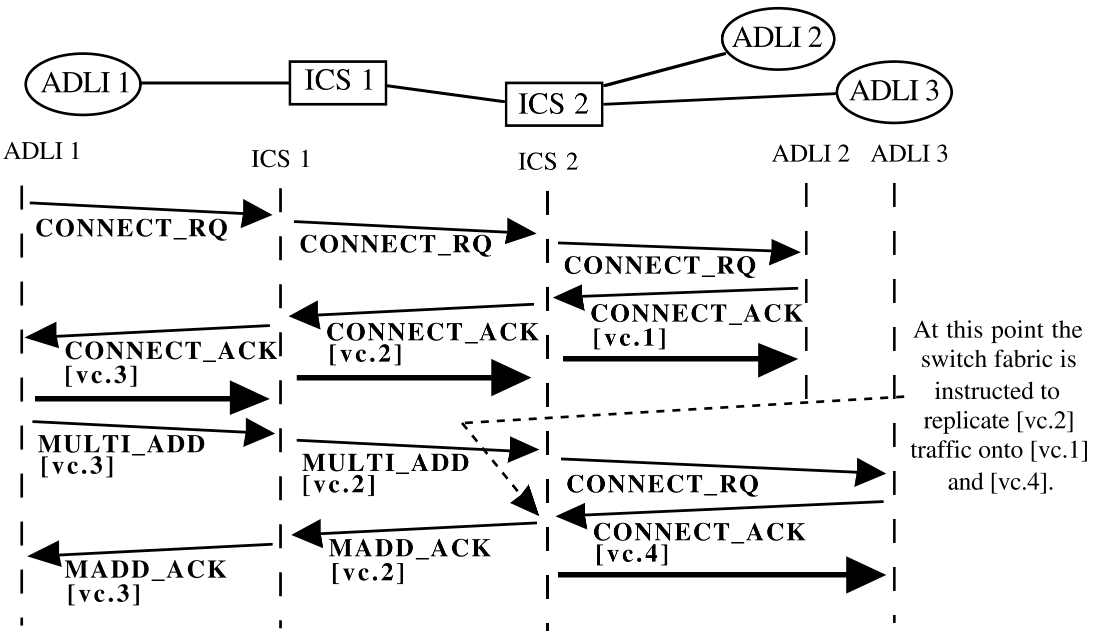


Figure A.2.4.8

Figure A.2.4.8 shows the timeline for a simple example. ADLI 1 wishes to create a multicast connection to ADLIs 2 and 3. First a standard unicast connection is established from ADLI 1 to ADLI 2. Each intervening ICS sets up unicast mappings for [vc.3->vc.2] and [vc.2->vc.1]. Then a MULTI_ADD is issued to make ADLI 3 part of the multicast group. ADLI 1 indicates that it wishes the multicast connection to be based on [vc.3]. ICS 1 recognises the next leg of the path to ADLI 3 is already covered by the connection to ADLI 2, so propagates a MULTI_ADD identifying [vc.2] as the basis of the next leg. Finally ICS 2 sees that the path to ADLI 3 diverges from that to AAL 2, and issues a unicast connection request to ADLI 3. When the ACK comes back from ADLI 3, ICS 2 informs its underlying switch fabric that traffic arriving on [vc.2] must now be multicast out on [vc.1] and [vc.4]. ICS 2 then generates a MADD_ACK back towards ADLI 1, in the same way an ICS would normally propagate back a CONNECT_ACK.

At the end of this sequence the database at ICS 1 would look like figure A.2.4.9 (showing the fields filled in with specific information from figure A.2.4.8). The database indicates that no cell copying is required of the underlying switch fabric, [vc.3] is only mapped to one outbound VPI/VCI.

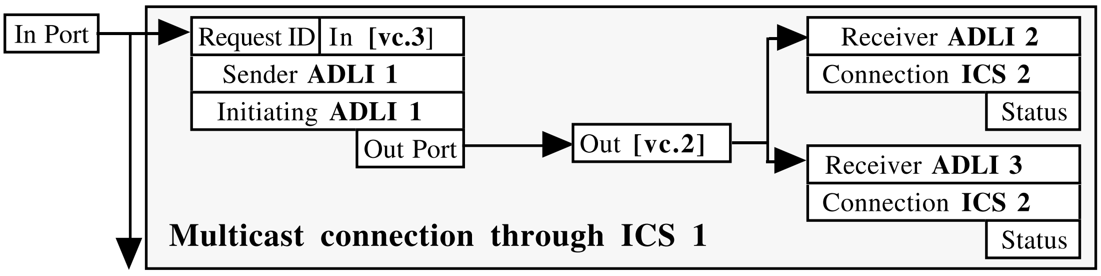


Figure A.2.4.9

The database at ICS 2 is shown in figure A.2.4.10, where for the sake of illustration it has been assumed that ADLI 2 and ADLI 3 are attached to different ports. In this case the switch fabric is instructed to copy from [vc.2] to [vc.1] and [vc.4].

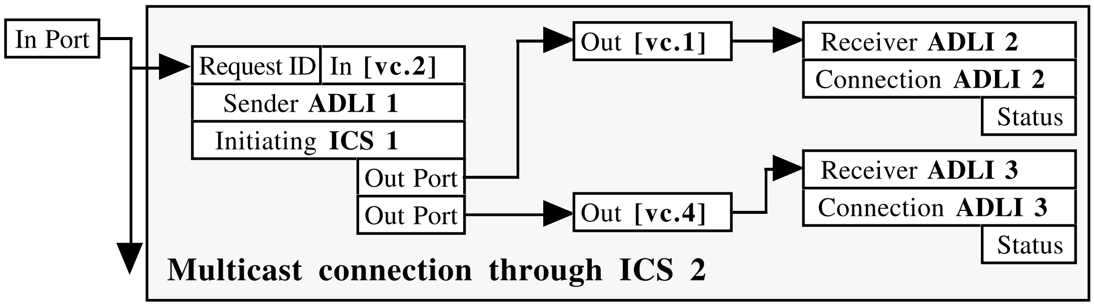


Figure A.2.4.10

An alternative scenario is shown in Figure A.2.4.11, where the destinations of the multicast group are not all directly connected to a single switch.

The first unicast link is from ADLI 1 to ADLI 2. When the MULTI_ADD intended to add ADLI 3 reaches ICS 1, it must be converted to a unicast request, as ADLI 3 is not directly connected to ICS 1. The components of the network between ICS 1 and ADLI 3 have no idea that ADLI 3 is becoming a leaf node. All ICS 1 is concerned about is that a valid CONNECT_ACK is received with a VPI/VCI that can be added to the switch fabric's multicast mapping table.

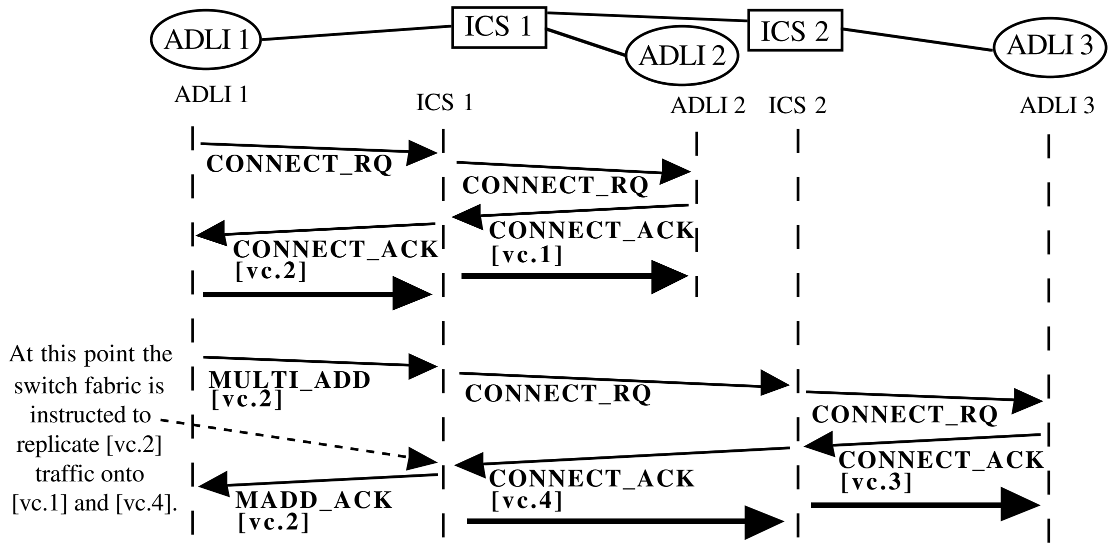


Figure A.2.4.11

As for unicast situations, the contents of the MADD_ACK or MADD_NAK are used to search the ICS database for the MULTI_ADD they are in response to. Also as for unicast, a new connection must always use a different Request ID. However, since multicast connections are based on pre-existing links - on any given leg of a virtual connection MULTI_ADDs must use the same Request ID as carried in the CONNECT_RQ that first established it.

The database for ICS 1 in this scenario will look very similar to figure A.2.4.12.

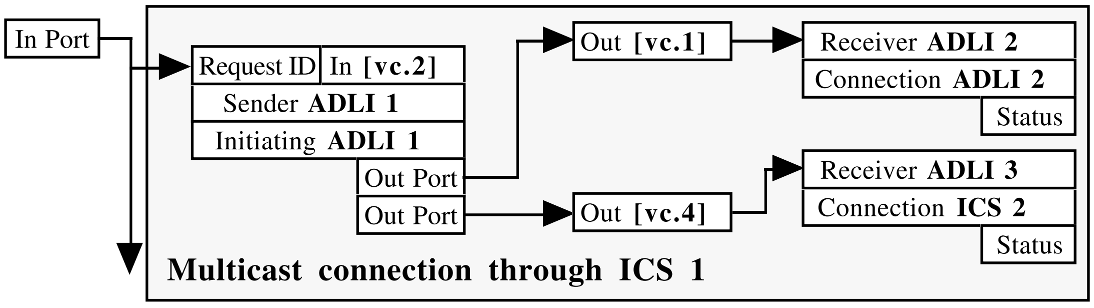


Figure A.2.4.12

The database for ICS 2 will look like a simple unicast connection.

Finally, there is the special case of leaf nodes sharing a single switch port. One of gNET's primary design goals was to allow the destination ATM node to choose whatever VPI and VCI it wished. There is no mechanism to force an ADLI to use a VPI/VCI that may already be carrying the data it needs. Consequently the local ICS must establish two unicast connections out of the same port - despite the fact that both destinations will 'see' each other's version of the same traffic.

Figure A.2.4.13 shows how this would appear. If the source material was 20Mbit/sec the shared port on switch 3 would be generating 40Mbit/sec. In most installations this limitation is not going to be a large problem - multicast groups are likely to spread over a number of switches and switch ports.

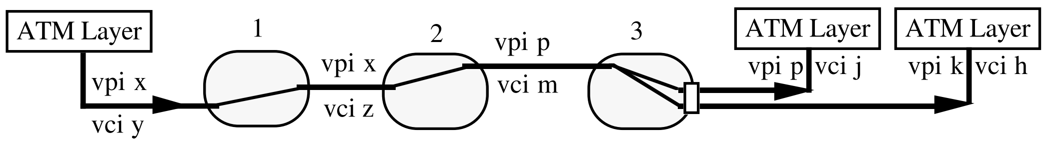


Figure A.2.4.13

A.2.4.3 Allocating a VPI/VCI to a link.

A new VPI/VCI only needs to be allocated for new unicast links. The mechanism was described in section A.2.2, but not explicitly included in section A.2.4.1's description. Figure A.2.4.14 is a repeat of the unicast connection created in figure A.2.4.4, with the PVS requests clearly shown.

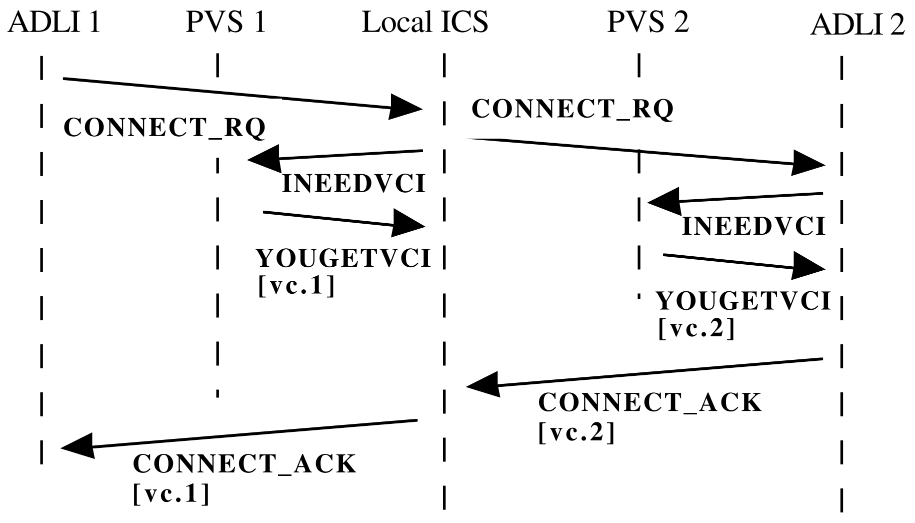


Figure A.2.4.14

PVS 1 and PVS 2 are represented as completely independent entities, although they may both be resident on the physical node providing the ICS service. In this case the GSE's are not specifically caching VPI/VCI pairs. ADLI 2 obtained [vc.2] before ACK'ing the connection request, and the ICS chose to overlap the PVS request with the propagation of the CONNECT_RQ towards ADLI 2. This technique assumes that most connection requests are ACKed, and reduces the time it takes the ICS to process and pass the returning ACK when it arrives. If ADLI 2 had NAKed the request, the ICS would have to return [vc.1] to the PVS.

A.2.4.4 Unicast and Multicast Virtual Path Connections.

When a GSE sees AAL Service type 15 in the Sender AEPI and Receiver AEPI it knows a VPC is being requested. The 2 bit VCI subset field from the Sender AEPI is used to indicate whether it is to be a subset-VPC or a full VPC. The establishment process for VPCs is virtually identical to that for VCCs, although the VPI/VCI passed back in the CONNECT_ACK is constructed and interpreted slightly differently.

gNET allows completely unrelated subset-VPCs to share VPIs on inter-switch links, provided the subsets will not overlap within a single VPI. To do this, switch fabrics must be able to shift all the VCIs in a particular VPI by a constant factor, either positive or negative, when the VPI is switched. The size of the shift at any given ICS node will be multiples of 0, ± 0x10, ± 0x100, or ± 0x1000. Not all switch fabrics, and hence ICS nodes, will be able to accommodate the VCI remapping needed to establish subset-VPCs. Such ICS nodes must immediately NAK requests for subset VPCs. When an AAL User's request for a subset-VPC is NAKed in this way, it should simply modify the AEPI to specify the entire VCI space (VCI subset type 3) and retry the request.

As for VCC requests, it is the Connection GSE that specifies where in the VPI and VCI space it expects data on the new connection to arrive. If the connection request was for a full VPC, then the CONNECT_ACK will return a new VPI to the Initiating GSE. The Initiating GSE will ensure that cells on the VPC will arrive at the Connection GSE on the VPI specified. The VCI field of the ACK must be set to zero by the Connection GSE, and can be ignored. A completely free VPI is obtained from the PVS by issuing an INEEDVCI request specifying the VCI range from 0 to 0xFFFF on a particular VPI.

Where the request is for a subset-VPC, the Connection GSE will return two pieces of information - the VPI for all cells on the new VPC, and the base value of the VCIs that will be associated with the new VPC. The Initiating GSE must then ensure that the subset-VPC's traffic is mapped to the new VPI, and that the VCIs start at the specified base value. This is achieved by configuring its switch fabric to apply the appropriate shift to the VCI field as the cells pass through.

How a Connection GSE derives the appropriate VPI and base VCI is implementation dependent. The simplest approach is to ignore the subset specification and simply allocate a completely new VPI (as for the full VPC case), specifying a base VCI of zero. This behaviour is necessary for minimal gNET functionality. There are many ways for a Connection GSE to choose to place a new subset-VPC on the same VPI as another subset-VPC between the two ICS nodes. These are beyond the scope of this report.

ICS nodes that can cope with subset-VPCs must also be able to mask off incoming VCIs. This is essential for the ICS node that is directly connected to the Sender ADLI. The mask should ensure that cells on the subset-VPC with VCIs outside of the agreed range are dropped immediately.

Establishing multicast VPCs may be achieved through ICS nodes that can multicast based on the VPI alone. As for multicast VCCs, a MULTI_ADD should be NAKed by any ICS that cannot support the required cell replication. Any implementation of gNET with subset-VPCs must be able to support multicast subset-VPCs.

A.2.5 Removing virtual connections.

There are two types of removal possible, with two different request GSUs - Sender initiated and Receiver initiated. Removing a virtual connection involves the passage of a disconnection request GSU along the path travelled by the original RQ (Sender initiated) or ACK (Receiver initiated). The first two subsections will deal with these two cases on unicast connections, followed by disconnections from a multicast group. No special attention is needed for VPCs or subset-VPCs, provided a gNET implementation is careful when releasing mappings that may be shared by other subset-VPCs.

A.2.5.1 Sender initiated removal from a unicast connection..

This occurs when the AAL User at the Sender ADLI issues a `gNET_disconnect()` to the local GSE. The response is to send an S_DISCONNECT, shown in figure A.2.5.1, to follow the path taken by the CONNECT_RQ that first established the connection. Once the request gets to the Receiver ADLI, an acknowledgment returns along the GSU path, dismantling VPI/VCI mappings at switching nodes along the way.

The S_DISCONNECT carries all the fields from the original RQ, so that Connection GSEs can identify the virtual connection being removed. The only response to an S_DISCONNECT is an S_DISCONN_ACK, which follows the reverse path of Connection GSE to Initiating GSE on each link.

ATIA of Connection GSE	64 bits	ATIA of Initiating GSE
ATIA of Initiating GSE	64 bits	ATIA of Connection GSE
<S_DISCONNECT>	8 bits	<S_DISCONN_ACK>
Request ID	24 bits	Request ID
ATIA of Receiver ADLI	64 bits	ATIA of Receiver ADLI
ATIA of Sender ADLI	64 bits	ATIA of Sender ADLI
Receiver AEPI	16 bits	Receiver AEPI
Sender AEPI	16 bits	Sender AEPI
Null	32 bits	Null
S_DISCONNECT		S_DISCONN_ACK

Figure A.2.5.1

An S_DISCONN_ACK is issued under two circumstances:

- By the Receiver ADLI, when a valid S_DISCONNECT arrives.
- By any Connection GSE receiving an S_DISCONNECT for a non-existing connection.

The second case helps shut up sources of erroneous S_DISCONNECTs, and provides support for lost-GSU-recovery (described later).

Figure A.2.5.2 shows the GSU sequence for the removal of the connection established in Figure A.2.4.4 When the S_DISCONNECT is received by an ICS, the management entity marks the virtual connection's status as 'shutting down'. A new S_DISCONNECT is then sent to the next Connection GSE used by the CONNECT_RQ. The identification of [vc.1] and [vc.2] is not explicitly carried within the GSU - it is derived from the Receiver and Sender ATIA fields, along with the Request ID.

Receipt of an S_DISCONN_ACK causes the ICS to clear the associated entry in the mapping table of its switch fabric, and issue another S_DISCONN_ACK on the next link back to the Sender ADLI. Finally it removes all state information in the management layer regarding the virtual connection. The receipt of an S_DISCONN_ACK indicates to an Initiating GSE that all links of the virtual connect between this point and the Receiver ADLI have been removed.

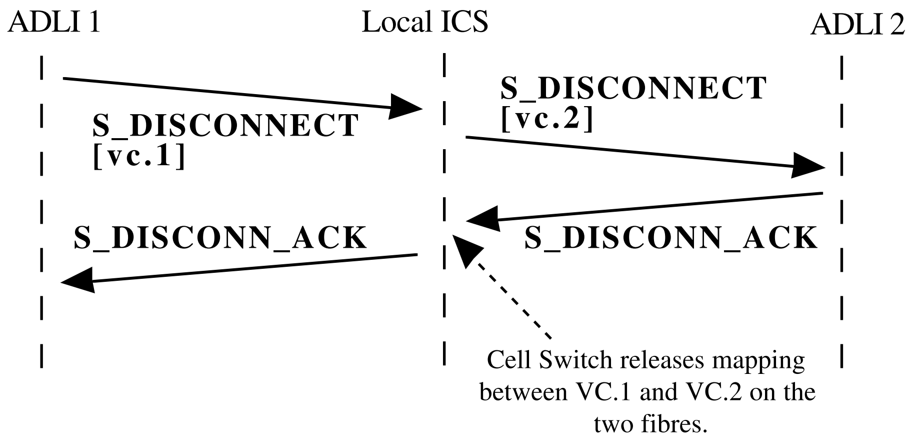


Figure A.2.5.2

A.2.5.2 Receiver initiated removal from a unicast connection..

The Receiver ADLI may also shutdown a virtual connection, using a signalling sequence almost a mirror image of the Sender initiated example. Figure A.2.5.3 shows the two GSUs.

R_DISCONNECT		R_DISCONN_ACK
ATIA of Initiating GSE	64 bits	ATIA of Connection GSE
ATIA of Connection GSE	64 bits	ATIA of Initiating GSE
<R_DISCONNECT>	8 bits	<R_DISCONN_ACK>
Request ID	24 bits	Request ID
ATIA of Receiver ADLI	64 bits	ATIA of Receiver ADLI
ATIA of Sender ADLI	64 bits	ATIA of Sender ADLI
Receiver AEPI	16 bits	Receiver AEPI
Sender AEPI	16 bits	Sender AEPI
Null	32 bits	Null

Figure A.2.5.3

Figure A.2.5.4 shows the R_DISCONNECT propagating from Connection GSE to Initiating GSE, back along the path from Receiver ADLI to Sender ADLI. When it is received at the Sender ADLI an R_DISCONN_ACK is issued, and transmissions cease on that virtual connection. As with S_DISCONNECT, intervening ICS nodes mark the virtual connection as 'shutting down' but do not actually remove the switch fabric's mapping table entry until the R_DISCONN_ACK is received travelling the other way. Once the ACK has passed, each ICS node removes all references to the particular virtual connection.

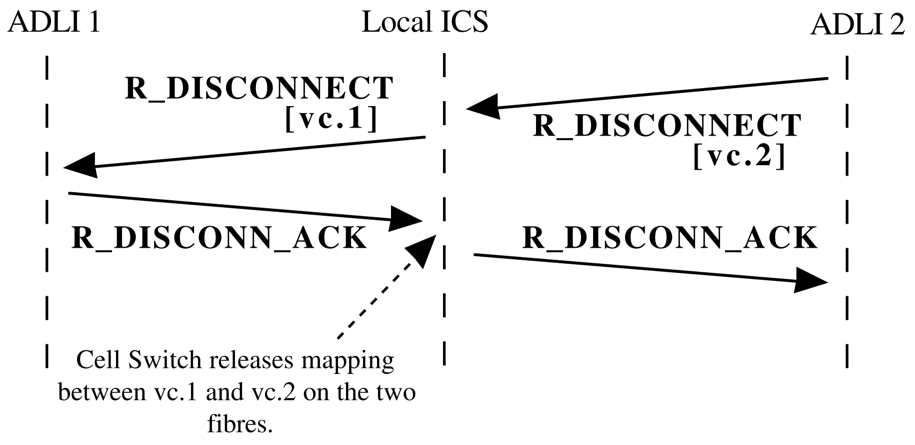


Figure A.2.5.4

A.2.5.3 Dropping leaf nodes from a multicast connection..

The only time a distinction needs to be made between multicast and unicast related signalling is when connections are being established. Once each ICS node has database entries indicating that certain destinations are part of a multicast tree, removing leaf nodes may be performed with the basic S_DISCONNECT GSU.

S_DISCONNECT handling by ICS nodes is exactly the same as for unicast connections, with the GSU being propagated along the path established by the previous MULTI_ADD and CONNECT_RQ. The difference occurs when the S_DISCONN_ACK is propagated back towards the Sender ADLI. At each ICS, receipt of the ACK causes the relevant Receiver ATIA/AEPI entry to be removed from the database. Until the last entry for a particular outgoing VPI/VCI is removed, the switch fabric's mapping table is not touched. The switch fabric's mapping table is only modified when an S_DISCONN_ACK results in a particular link being shutdown.

For example, figure A.2.5.5 shows ADLI 3 (from the multicast connection in figure A.2.4.8) being dropped. The connection being referred to is located using the Sender, Receiver, and Request ID information in the GSU, rather than specifying the actual VPI/VCI. This method ensures that the 'lost GSU' recovery mechanism described in section A.2.6 may be applied.

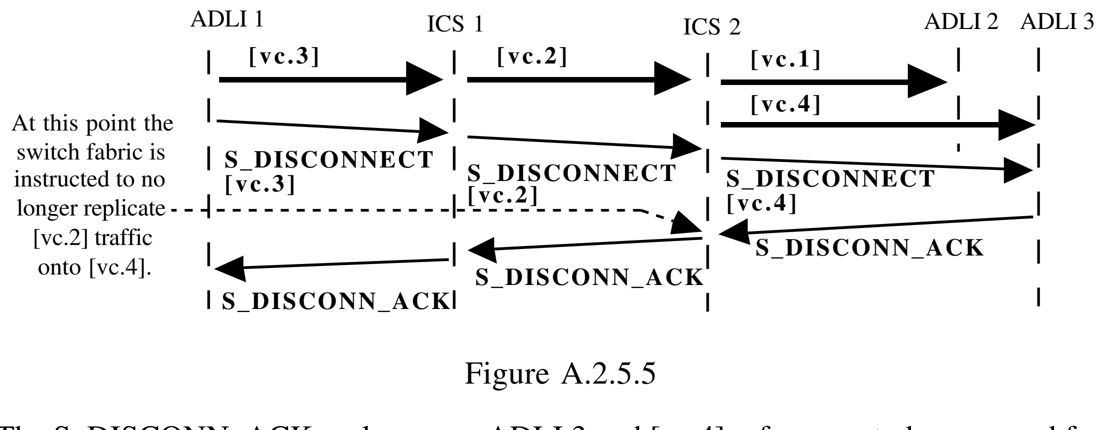


Figure A.2.5.5

The S_DISCONN_ACK reply causes ADLI 3 and [vc.4] references to be removed from the database in ICS 2, and the switch fabric updated so that it no longer replicates cells onto [vc.4]. The ACK then passes through ICS 1, causing the database entry for ADLI 3 to be removed. No modification to the mapping table on ICS 1 is required, as [vc.2] still forms a link on the path to ADLI 2.

A.2.5.4 Leaf node initiated removal.

The GSEs within Receiver ADLIs have no way of telling whether an incoming connection is a simple unicast one, or simply a unicast leaf of a multicast tree. However, issuing an R_DISCONNECT works under both situations. As in the previous section, an ICS node will only modify the mapping table in the underlying switch fabric if a disconnection results in a port and VPI/VCI no longer being needed. Aside from that, an ICS node handles an R_DISCONNECT from a multicast leaf node as though it was unicast connection.

A.2.5.5 Releasing a VPI/VCI no longer in use.

Figure A.2.5.6 expands on the connection removal in figure A.2.5.2 to include the PVS related signalling traffic. As with VPI/VCI allocation, a certain amount of asynchrony may be implemented between the generation of connection related GSUs (the S_DISCONN_ACK) and PVS related GSUs. In figure A.2.5.6 the ICS pre-emptively releases [vc.1], but it could just as well have waited until the S_DISCONN_ACK was received from ADLI 2. It was also not necessary for ADLI 2 to wait for the PVS release sequence to finish before issuing the S_DISCONN_ACK. A faster scheme could issue the S_DISCONN_ACK and *then* worry about local issues such as releasing [vc.2]. These decisions are implementation dependent.

A special problem for ICS nodes is the possibility of repeated S_DISCONNECTs (which may occur when lost GSU recovery is implemented, as described in section A.2.6). When a connection is in the ‘shutting down’ stage, care must be taken to ensure a VPI/VCI is only released once.

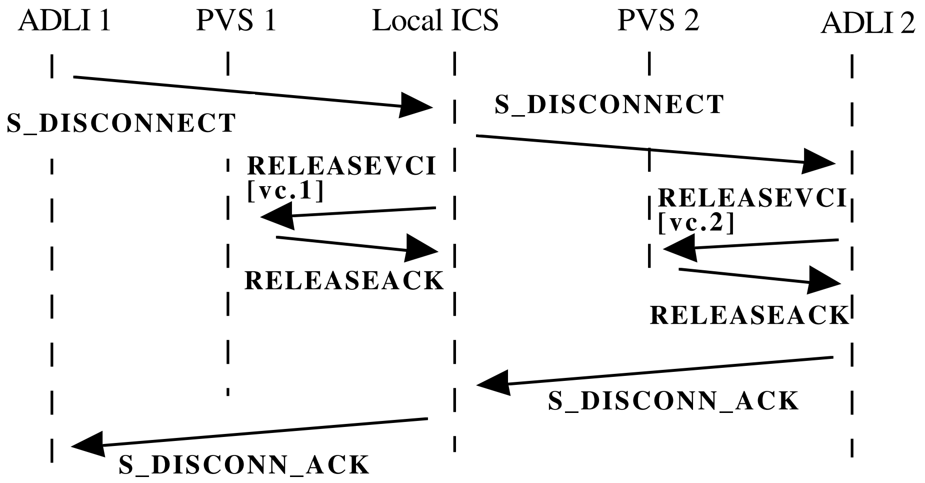


Figure A.2.5.6

A.2.6 Recovering from lost GSUs

gNET uses acknowledgment timers to cope with GSU loss during the various phases of connection establishment and removal.

A.2.6.1 Loss during connection establishment.

If a Connection GSE drops a CONNECT_RQ or MULTI_ADD, or an Initiating GSE drops an ACK or NAK, the Sender ADLI never receives a response to its initial request. The solution is to set a timer when the request is sent, and reset the timer only when an ACK or NAK arrives in response. If the timer expires, the original GSU is simply retransmitted. Error recovery is implemented only by the Sender ADLI.

Figure A.2.6.1 shows how figure A.2.4.4 would have appeared if one of the CONNECT_ACKs was lost. Since the ICS saw and processed the ACK from ADLI 2 it considers the virtual connection to be active. The retransmitted RQ is detected as such because every single field matches the database entry for the existing connection. In the most general terms, a Connection GSE that receives an RQ for a pre-existing connection must immediately regenerate an appropriate ACK.

This mechanism ensures that retransmitted RQs only need to travel as far as the link that lost the ACK or NAK.

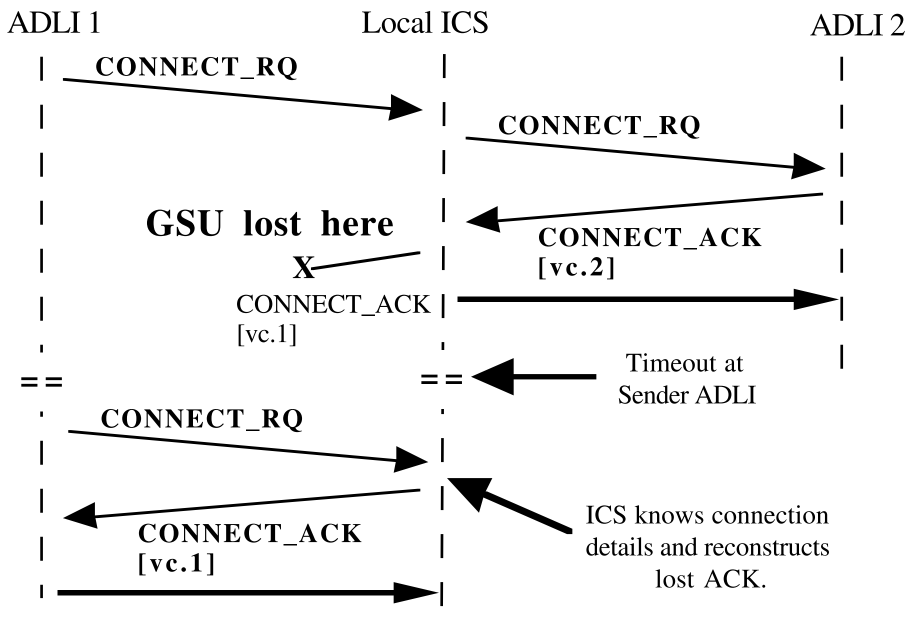


Figure A.2.6.1

Figure A.2.6.2 shows the response to the loss of an RQ. The loss means that the ICS is only holding a provisional entry for the virtual connection when the repeat RQ arrives. The ICS simply regenerates a copy of the CONNECT_RQ it passed along the first time. When the RQ finally gets past the blockage, an ACK or NAK will return and processing continues as described in section A.2.4. An important point here is that a new route for the repeat GSU is not chosen - the route allocated when the first RQ arrived continues to be used.

Both of these mechanisms together solve the more general multi-hop case. A repeat RQ, for whatever reason, will propagate until it reaches the first Connection GSE that considers the virtual connection to have already been ACKed. Lost GSU recovery only involves propagation as far as the point of failure.

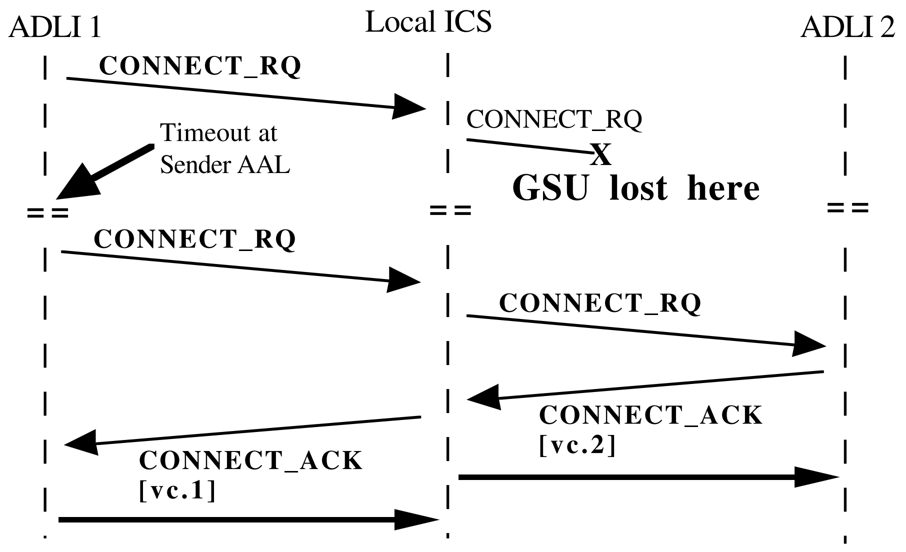


Figure A.2.6.2

If the Receiver ADLI had NAKed the request, but the NAK was lost, the repeat RQ will travel the entire path again. This scenario places the most load on GSEs along the path, as every ICS that processed the NAK will have removed all references to the original request. When the repeat RQ arrives they will treat it as an entirely new connection request, ultimately resulting in a repeat RQ arriving at the Receiver ADLI for a service it has already denied.

If the Sender ADLI does not receive an ACK or NAK within a certain number of retransmissions it stops and informs the AAL User that the connection request failed. This type of failure should also be signalled to the ADLI management, as it probably indicates some serious problem in the network.

Repeated ACKs and NAKs are to be passed onwards by ICS nodes, and ignored by ADLIs.

Finally, this mechanism is equally applicable to MULTI_ADDs, MADD_ACKs, and MADD_NAKs.

A.2.6.2 Loss during connection removal.

The recovery mechanism is similar to that for connection establishment. The Sender ADLI retransmits the S_DISCONNECT if an S_DISCONN_ACK is not received within a particular time. ICS nodes that have already marked the connection as ‘shutting down’ will pass on repeat S_DISCONNECTs, until a node is found that is willing to issue an S_DISCONN_ACK. If the original S_DISCONN_ACK was lost part way back to the Sender ADLI, the repeat S_DISCONNECT will only propagate as far as the first Connection GSE that no longer ‘remembers’ the virtual connection being referred to. This GSE will immediately reply with an S_DISCONN_ACK (as specified in section A.2.5.1), which should finally make its way back to the Sender ADLI. Figure A.2.6.3 shows this case.

Figure A.2.6.3 has the same structure as Figure A.2.6.1. Figure A.2.6.2 can similarly be used to visualise the recovery process when the S_DISCONNECT itself is lost. Both of these diagrams, with the directions of GSUs reversed, will describe the error recovery procedure for R_DISCONNECTs.

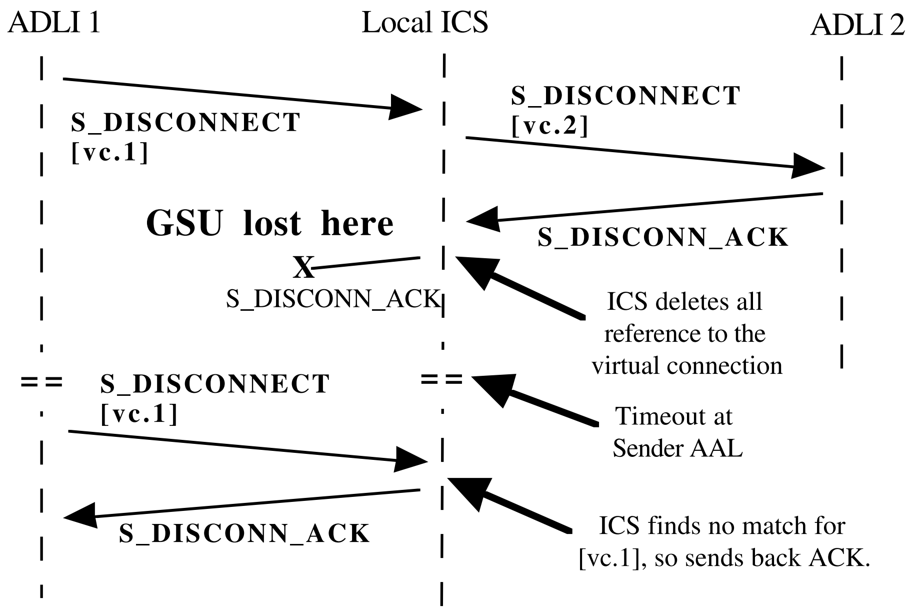


Figure A.2.6.3

If an ACK is not received within a certain number of retransmissions, they are stopped and local resources allocated to the connection are released. This type of failure should also be signalled to the ADLI management, as it probably indicates some serious problem in the network.

A.2.6.3 Clearing ICS nodes after signalling failure.

The mechanisms described in the sections A.2.6.1 and A.2.6.2 can both lead to ICS nodes being left with 'orphaned' entries for virtual connections - either in 'provisional' or 'shutting down' states. This can occur when a longterm source of cell loss causes the Sender ADLI to reach its retransmission limit without receiving an ACK.

The first line of defence is for each ICS to attach its own timer to virtual connections in the 'provisional' or 'shutting down' state. The timer's period is longer than the retransmission period of the ADLIs, and it is reset whenever a repeat request GSU is seen. If the timer ever expires, the ICS assumes the activity has failed and it releases all resources associated with the virtual connection (possibly including entries in the switch fabric mapping tables).

The second line of defence is implemented within the Sender ADLI. If a connection request fails, instead of simply ceasing the transmission of repeat CONNECT_RQs or MULTI_ADDs, the Sender ADLI enters the connection removal phase before returning a failure indication to the AAL User. The Sender ADLI begins sending S_DISCONNECTs for the same virtual connection, until one is ACKed or the retransmission limit is again reached.

The CONNECT_RQ -> S_DISCONNECT transition at the Sender ADLI works in conjunction with the timers within the ICS to cope with the most difficult scenario. If the GSU loss in figure A.2.6.1 had continued until the Sender ADLI gave up sending CONNECT_RQs, the ICS would have been left convinced the virtual connection was successfully established. Sending an S_DISCONNECT ensures the ICS removes the entry, without even being aware that the Sender ADLI considered the connection request to have failed. If the source of CONNECT_ACK loss also causes the subsequent S_DISCONN_ACKs to be lost (as in Figure A.2.6.3), the system will still converge to the desired state when the Sender ADLI gives up retransmitting S_DISCONNECTs.

Both of these mechanisms are needed to cope with losses in the multi-hop scenario. Consider the situation in Figure A.2.6.4. The connection request has been successful as far as ADLI 2 and ICS 2 are concerned, but the continual loss of ACKs between ADLI 1 and ADLI 2 leaves ADLI 1 with an entry still marked ‘provisional’. When ADLI 1 (the Sender ADLI) gives up and begins sending S_DISCONNECTs, ICS 1 marks the entry as ‘shutting down’ and starts a timer. After the first S_DISCONNECT reaches ADLI 2, and is ACKed, the virtual connection no longer exists beyond ICS 1.

However consider the possibility that all the S_DISCONN_ACKs are lost between ICS 1 and ICS 2. Eventually ADLI 1 gives up the S_DISCONNECT transmissions as well, leaving the timer in ICS 1 to perform the final removal of the virtual connection’s entry when it expires. Note that after the first S_DISCONNECT went all the way to ADLI 2, subsequent S_DISCONNECTs are ACKed by ICS 2, as it no longer has any references to the original virtual connection.

Had the Sender ADLI simply given up after too many CONNECT_RQs, ICS 1 would have removed its ‘provisional’ entry when its timer expired. However, ICS 2 and ADLI 2 would have been left believing a virtual connection had been successfully established. By sending the S_DISCONNECTs, the Sender ADLI ensures every node either explicitly remove entries for the connection, or are put into a ‘shutting down’ state that will time out and ultimately achieve the same end.

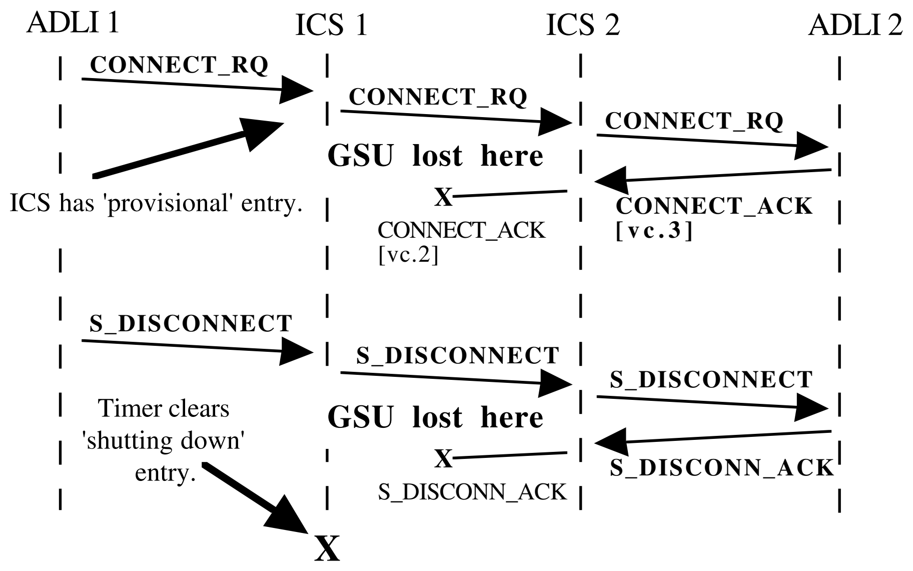


Figure A.2.6.4

Sending S_DISCONNECTs following failed CONNECT_RQs does not affect the alternative scenario where the CONNECT_RQ is being lost repeatedly. In this case ICS timers are used to ensure the system eventually releases all resources allocated to the failed ‘provisional’ connection. Figure A.2.6.5 shows this situation.

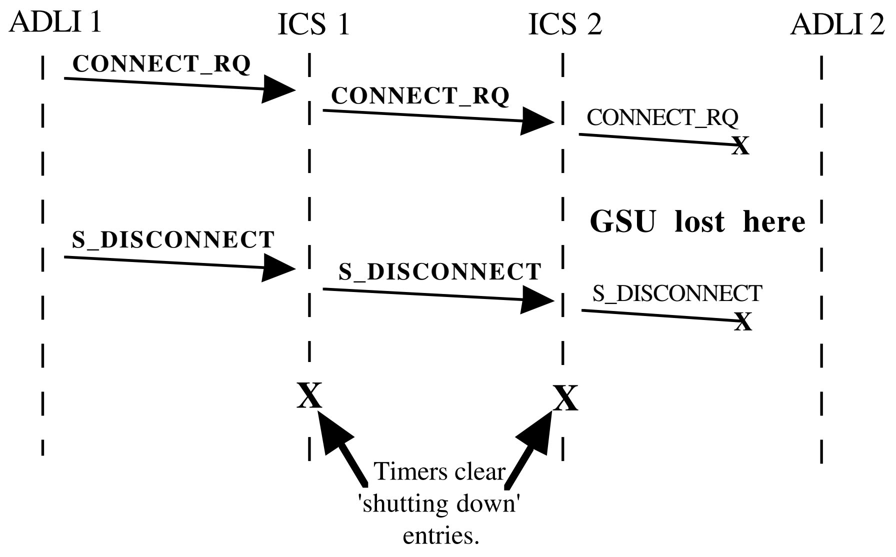


Figure A.2.6.5

In this case each ICS has a ‘provisional’ entry which gets turned directly into a ‘shutting down’ entry when the S_DISCONNECTs begin. Finally timers in all ICS nodes clear their entries when ADLI 1 stops sending.

Both figures A.2.6.4 and A.2.6.5 are worst case scenarios, which the present algorithms successfully cope with.

A.2.7 Network control of ADLI Policing Units.

Section 2.4 introduced the idea of every ADLI having a ‘policing unit’ active on each outgoing virtual connection. The policing unit is initialised when a connection is established, and may have its characteristics altered during the course of the connection. Modification while the connection is active may be initiated by any ICS node along the path, using the NEW_RATE GSU shown in figure A.2.7.1. The GSU is based on the CONNECT_ACK, with the final 32 bit field specifying the new traffic shape characteristic to be implemented by the policing unit.

ATIA of Initiating GSE	64 bits
ATIA of Connection GSE	64 bits
<NEW_RATE>	8 bits
Request ID	24 bits
ATIA of Receiver ADLI	64 bits
ATIA of Sender ADLI	64 bits
Receiver AEPI	16 bits
Sender AEPI	16 bits
New rate field	32 bits

NEW_RATE

Figure A.2.7.1

The conditions under which an ICS node might send back a NEW_RATE message depend on the QoS scheme applicable to the given virtual connection. Under some schemes, the Receiver ADLI may generate the NEW_RATE request. Under no circumstances may a NEW_RATE specify a characteristic outside the QoS allocated for that virtual connection. For example, a CBR connection (Type 1 QoS), should never be issued with a NEW_RATE requesting a variation of the connections cell rate. A Type 0 QoS connection may never be issued with a NEW_RATE below the guaranteed minimum rate.

The format of the 32 bit 'New rate' field is shown in Figure A.2.7.2, similar to that of the QoS field. A 4 bit 'type' field allows for up to 16 different styles of traffic shaping characteristics to be specified - typically these will relate to the 16 possible QoS types.

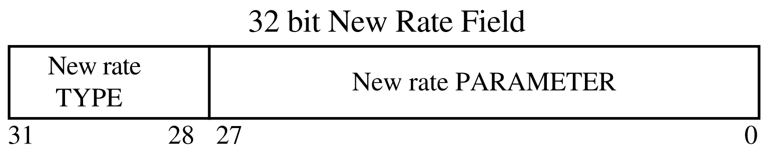


Figure A.2.7.2

The NEW_RATE GSU may be generated by any ICS along the path of a virtual connection. NEW_RATE GSUs travel backwards along the route from Connection GSE to Initiating GSE until they reach the Sender ADLI.

Figure A.2.7.3 shows a simple example where a NEW_RATE GSU is used to temporarily modify the source traffic characteristics. In the general case an ICS node is expected to release any restrictions it has imposed as soon as network conditions allow.

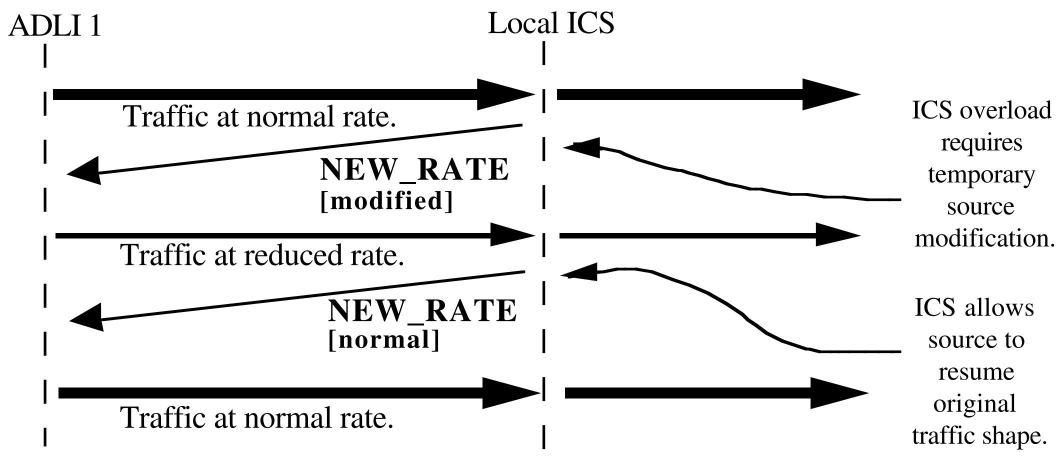


Figure A.2.7.3

A.2.7.1 Peak cell rate control.

Figure A.2.7.4 shows the Type 0 NEW_RATE GSU, which provides only a single peak cell rate value. This provides the ability to modify only the peak cell rate that a connection is entitled to send at. Such a function may be useful for Type 0 QoS system connections, where 'the network' selects a rate limit somewhere between the ranges specified in the QoS field.

N1 is a 14 bit field that represents the new peak cell rate using the same scheme as described for the Type 0 QoS parameter.

Peak rate only

TYPE=0	Unused	Peak cell rate N 1
31 28	27 14	13 0

Figure A.2.7.4

gNET does not yet define the conditions under which an ICS node would request a source to send at its guaranteed minimum rate. An ICS nodes may issue NEW_RATEs for rates in-between the peak and minimum values requested by the applications.

A Type 0 NEW_RATE will be ignored on Type 1 QoS connections (CBR).

A.2.7.2 Other traffic shaping functions.

More advanced traffic shaping functions, such as cell-spacers, may be defined in the future by ITU-T and incorporated into the policing units. The network may specify such traffic shaping in order to ensure, for example, the CDV limits required on particular Type 1 QoS connections.

Other NEW_RATE Types may be defined within gNET to cover the required characteristics.

A.3 Routing between ICS nodes.

The connection establishment phase requires that each ICS node is able to route the CONNECT_RQ to the Connection GSE that is ‘next closest’ to the Receiver ADLI. gNET does not yet specify any particular protocol to keep ICS routing tables updated. Solutions will probably draw on routing experiences within internetworking environments, such as TCP/IP.

A gNET routing protocol needs to:

- Identify what ADLIs are directly connected to particular ICS nodes.
- Advertise an ICS node as a route to a particular ATIA.

Section A.2.1 provides a solution to the problem of identifying and tracking directly connected ADLIs. Meaningful and effective sharing of this information between ICS nodes is rather more complicated.

ICS nodes will always have at least one port to which another ICS is directly attached. Their identities may be established by issuing WHOISICS and WHOISPVS requests on all ports and noting the responses.

The choice of ATIA format will be intimately tied to the routing mechanism. However, it is undefined at this stage.

In very small systems the ICS may be configured with static routing tables. However, realistic systems will implement routing management entities that use gNET to establish contact with their directly connected ICS peers. Two possibilities exist for these interconnections:

- Use gNET Signalling Units to carry routing table updates between routing entities.

- Use gNET to establish AAL5 connections between routing entities that are implemented as AAL Users.

The first mechanism would involve additional complexity in the gNET signalling entity of ICS nodes, whilst limiting routing update messages to 20 bytes (the GSU payload). The second method is the appropriate solution. It allows the routing entities to be implemented as separate functions - AAL Users registered at a particular AEPI value. Each routing entity would be free to send variable sized blocks of data across their virtual connections.

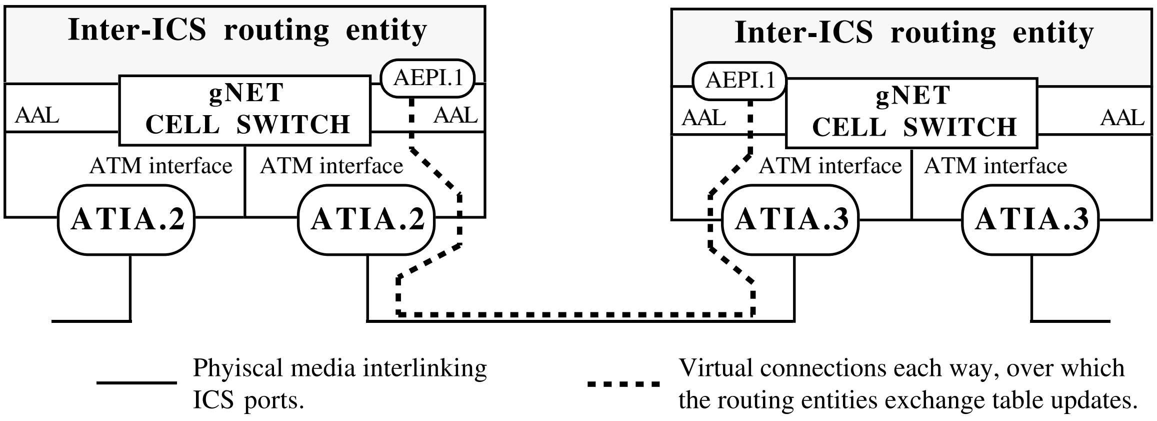


Figure A.3.1

Figure A.3.1 shows the routing entities of two directly connected ICS nodes having established two unidirectional gNET connections between each other. A globally defined AEPI is used for all Inter-ICS routing entities and pre-configured into the ICS node. The suggested AEPI is 0x5001 - Protocol ID 0x001 over an AAL5 connection (a complete list of AEPIs suggested for the support of common network services is given in section A.4).

A.4 Pre-allocated ATIA and AEPI values.

This section lists some ATIA and AEPI values that have either been defined as part of gNET, or specified while using gNET to support conventional LAN services.

ATIA	Meaning
0xFFFFFFFFFFFFFFFF	Broadcast ATIA
All other values undefined.	

The following AEPIs are suggested for gNET implementations. For common protocols these are variations on their equivalent Ethernet Type codes.

AEPI	Meaning
0x5001 .. 0x500F	Inter-ICS routing support entities.
0x5010	RFC 1483 compliant LLC interface.
0x5800	Internet Protocol (IP) interface.
0x5806	IP Address Resolution Protocol (ARP).
0x509B	AppleTalk.
0x50F3	AppleTalk Address Resolution Protocol (AARP).
0x5200	Melbourne University Video Phone - link control.
0x1200	Melbourne University Video Phone - video/audio.

0x5800, 0x5200, and 0x1200 are the only ones to have actually been applied during the work on this report. The rest are suggested allocations for future work.

A.5 Distributing the ICS, PVS, and switch fabric functions.

So far it has been assumed that the ICS function actually resides on the physical switch node. This does not have to be the case - it is sufficient merely that the ICS function is ‘visible’ to all GSEs on a given physical medium. Figure A.4.1 shows how a single ICS/PVS entity may serve more than one switch fabric and connected media.

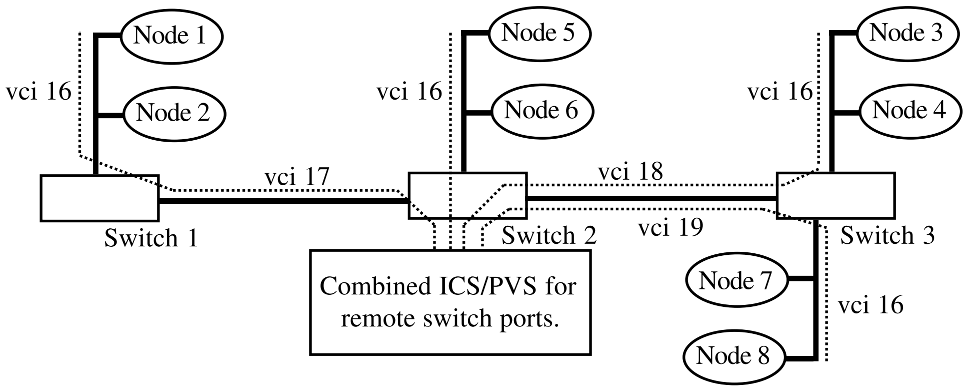


Figure A.4.1

All end-nodes connect to their local switch, and communicate on the gNET signalling channel of VPI 0, VCI 16. Switch 2 supports multiple independent ICS/PVS functions, each one ‘attached’ to a different VCI (we assume VPI 0 for all signalling traffic in this example). The main ICS function for switch 2 listens on VCI 16 of all local ports. However other ICS/PVS functions also listen on VCI 17, 18, and 19 coming in to switch 2. Switch 1 and 3 have been pre-configured to send gNET signalling traffic on their other local ports to switch 2. In each case VCI 16 is mapped to a unique VCI before reaching switch 2, so that different ICS/PVS functions ‘know’ what set of end-nodes they are serving. The mapping is bidirectional, so traffic sent from switch 2 to switch 3 on VCI 19 will be received by the GSEs in Node 7 and Node 8.

As far as all the end-nodes are concerned, the ICS and PVS functions all ‘appear’ to be on VCI 16 on their local medium and switch port. However, from the perspective of switch 2:

Node 1 and 2 are served by the ICS/PVS function on VCI 17,
Node 3 and 4 are served by the ICS/PVS function on VCI 18,
Node 5 and 6 are served by the ICS/PVS function on VCI 16,

Node 7 and 8 are served by the ICS/PVS function on VCI 19.

This scheme requires some external mechanism for each ICS entity to control the mapping tables of the switch fabrics they are meant to be serving. It may be a completely non-ATM control link (for switch fabrics that are geographically local), or an additional, fixed VCI/VPI back to simple hardware within each switch fabric that processes switch table update messages.

If switch fabrics are developed that can accept basic configuration messages over a virtual channel, then the mechanism in Figure A.4.1 may be extended to cover situations where the ICS/PVS functions are implemented many physical links away from the switches they control.

A.6 Support for other signalling systems.

‘Other signalling systems’ can describe two different scenarios - the first where a gNET based ATM LAN needs to create ATM connections through a public B-ISDN, and the second where gNET is overlaid by a more complex signalling system. This section takes a brief look at the broader issues.

A.6.1 Interworking gNET with the wider B-ISDN.

It is unlikely that a gNET based ATM network will be completely isolated from the wider ATM based B-ISDN, or even other proprietary ATM networks. One option is for the gNET network and wider ATM network to be considered separate link layers in an internetwork. Network level routers would interconnect each ATM network at its edges. This requires a router with interfaces capable of supporting both gNET and at least one foreign signalling protocol.

A preferable approach is for the boundary between networks to be a switch, allowing ATM cells to pass straight through, and only terminating the signalling protocols there. Provided the signalling protocols of both ATM networks have agreed on a VPI/VCI mapping across their boundary, a complete end-to-end ATM connection can be established.

The switch at the boundary would have to be served by a combined signalling entity, with major functions including:

- Translation between
    - ATIAs/AEPIs and equivalent identifiers on the foreign network.
    - gNET QoS specification and QoS supportable by foreign network.
- Issuing gNET connection requests when foreign connection requests come in.
- Issuing foreign connection requests when gNET connection requests come in.

A.6.2 Overlaying gNET with new signalling services.

An alternative solution to the interworking problem is to overlay gNET with the signalling system of the wider ATM network (for example Q.93b, when it is finished). Such a ‘Complex Signalling System’ (CSS) will be implemented by interposing Complex Signalling Entities (CSEs) between the AAL Users and the AAL/ATM layer services within end-nodes.

CSEs will use the underlying gNET service to establish their own signalling channels, and other virtual connections in response to user requests. Switch nodes will be controlled by CSEs performing the CSS equivalent of the ICS function.

At the very least, the underlying gNET system solves the boot-strap problem. When an ADLI locates its local ICS during boot up, it finds sufficient information for the resident CSE to immediately establish a signalling connection to the CSE on the ICS.

Figure A.6.1 shows an ADLI with a CSE overlaid and providing connection management service for AAL Users. All CSEs register themselves with their underlying gNET entity on a common AEPI. ICS nodes need to be upgraded to allow AAL Users to exist, be bound to an AEPI, and terminate virtual connections. gNET is used to create the initial connection from the end-node CSE to <ATIA of ICS, AEPI.cse>, the switch nodes CSEs (where AEPI.cse is the globally allocated AEPI for this CSS). The end-node CSE then passes its local ATIA across the connection, using a CSS specific message format, and the switch node CSE uses gNET to establish a return connection to <End-node ATIA, AEPI.cse>. From this point onwards the CSEs are now in control of the end-nodes relationship to the switch.

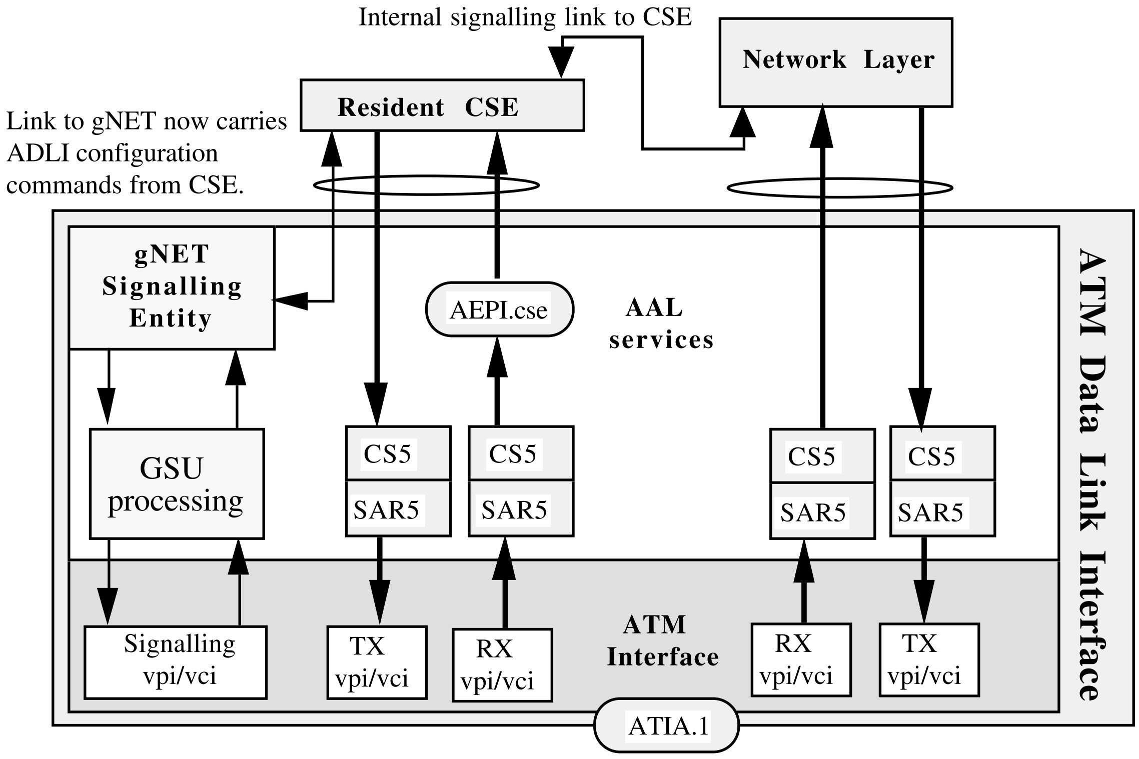


Figure A.6.1

In figure A.6.1 an arbitrary Network Layer has established a local signalling connection to the CSE, and used it to establish an ATM connection to a remote peer - this connection was established without direct communication with the local gNET signalling entity.

An overlaid CSE requires more services from the gNET signalling entity than those described in A.1. Mechanisms must be provided for the CSE to request re-configuration of data paths within the ADLI, allocation of free VPI and VCIs, and return of the local ICS node's ATIA. The CSE uses the local gNET entity to provide VPI/VCI management service, which in turn uses the underlying PVS service of the gNET LAN. This avoids replication of VPI/VCI management, and also allows other gNET services to continue to coexist with CSS controlled connections.

A.7 Many to One multicast at the AAL User level.

As mentioned in section A.1.2, an ADLI must be capable of converging multiple incoming VCCs onto a single local access point identified by <Local ATIA, Specific AEPI>. This leads to situations as shown in figure A.7.1. Once an AAL User has established a local access point, and registered an AEPI with the ADLI, the connection of multiple incoming VCCs is completely hidden. It is left up to individual implementations to provide signals to the AAL User when a new VCC is connected or removed, however all implementations must allow the AAL User to poll the ADLI and obtain a list of connection currently terminating on it. The incoming VCCs may originate from a mixture of unicast and multicast sources.

gNET does not support many-to-one multicast at the ATM level.

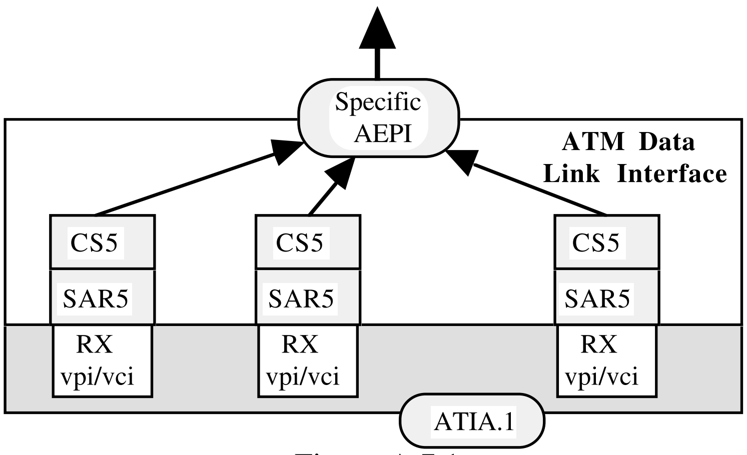


Figure A.7.1

A.8 GSU types, and error codes during connection establishment.

The following section describes some actual numerical values used during the development of a gNET testbed.

A.8.1 Numerical values for GSU Type field.

No particularly fancy or meaningful encoding scheme was applied. The 8 bit GSU Type field is currently broken into the following groups:

Connection/disconnection functions.	0x10 to 0x1F
VPI and VCI management functions.	0x20 to 0x2F
Node identification.	0x30 to 0x3F
Policing Unit management	0x40 to 0x4F

CONNECT_RQ	0x10	S_DISCONNECT	0x13
CONNECT_ACK	0x11	S_DISCONN_ACK	0x14
CONNECT_NAK	0x12	R_DISCONNECT	0x15
MULTI_ADD	0x17	R_DISCONN_ACK	0x16
MADD_ACK	0x18		
MADD_NAK	0x19		

WHOISPVS	0x20	RECLAIMRQ	0x25
IAMPVS	0x21	RELEASEVCI	0x26
INEEDVCI	0x22	RELEASEACK	0x27
YOUGETNAK	0x23		
YOUGETVCI	0x24		

WHOISICS	0x30	HEARBEAT	0x32
IAMICS	0x31		

NEW_RATE	0x40

A.8.2 Error codes returned in CONNECT_NAK.

The error condition field in the CONNECT_NAK must be filled in by the Connection GSE that decides the connection request cannot be supported. This may reflect certain error conditions at the Receiver ADLI, or at an ICS somewhere along the path. Ultimately this error indication should be passed back to the AAL User. Some error conditions may indicate significant network problems (for example, an ICS that has run out of resources to accept any connections), or problems that may be solved by retrying the connection request a little later.

The current list of error codes is:

SW_ERR_NOSLOT	0x1	An ICS had no free slot in its connection table.
SW_ERR_NOQOS	0x2	An ICS cannot support requested QoS.
SW_ERR_NOMULTIVCC	0x3	An ICS cannot support VCC multicasting.
SW_ERR_NOSUBSET	0x4	An ICS cannot support VCI shifting.
SW_ERR_NOMULTIVPC	0x5	An ICS cannot support VPC only multicasting.
SW_ERR_NOMULTICONN	0x6	The MULTI_ADD matched no current connection.
AAL_ERR_NOSLOT	0x10	Receiver ADLI has no free slot in connection table.
AAL_ERR_NOAEPI	0x11	Receiver ADLI has no matching AEPI registered.
AAL_ERR_NOVCI	0x12	Receiver ADLI failed to obtain a VPI/VCI.
AAL_ERR_DEST	0x13	This GSE cannot terminate or propagate the RQ.

The SW_ERR_NOQOS error should probably result in at least one more attempt to establish the connection. The request may have failed due to temporary resource starvation when multiple connection requests provisionally tie up resources. An SW_ERR_NOSUBSET means that the network cannot support subset-VPCs - request a full VPC instead.

The error AAL_ERR_DEST may be issued if a Receiver ADLI receives an RQ that should have gone to another ICS, or an ICS node receives an RQ for which it can establish no route.

A.9 References.

[A-1] D.C. Plummer, “An Ethernet Address Resolution Protocol”, RFC-826, November 1982.

[A-2] R. Finlayson, T. Mann, J. Mogul, M. Thiemer, “A Reverse Address Resolution Protocol”, RFC-903, June 1984.

[A-3] G.S. Sidhu, R.F. Andrews, A. B. Oppenheimer, "Inside AppleTalk", Addison-Wesley, March 1989.