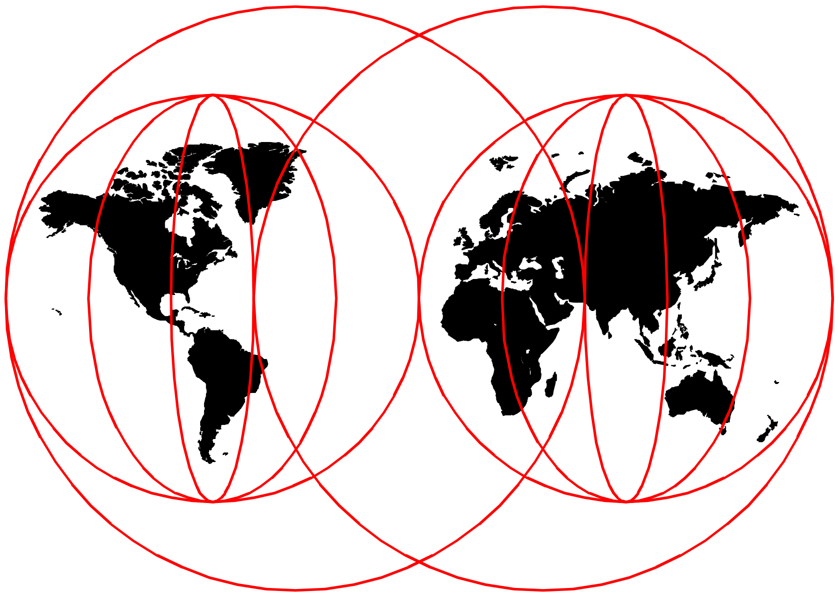
RS/6000 ATM Cookbook

Hajo Kitzhöfer, Cindy Kueck Young



International Technical Support Organization

www.redbooks.ibm.com

SG24-5525-00

SG24-5525-00

International Technical Support Organization

RS/6000 ATM Cookbook

April 2000

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix C, “Special notices” on page 277.

First Edition (April 2000)

This edition applies to AIX Version 4.3, program number 5754-C34, and supsequent releases.

This document created or updated on April 11, 2000.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. JN9B Building 003 Internal Zip 2834
11400 Burnet Road
Austin, Texas 78758-3493

Contents

Preface

The objective of this redbook is to provide material describing how to use Asynchronous Transfer Mode (ATM) technology within an RS/6000 environment.

This book describes key concepts, such as:

- Classic IP
- LAN Emulation
- HACMP scenarios

Also, day-to-day ATM problems and performance issues will be covered throughout this book.

This redbook was written for those who need to implement ATM in an RS/6000 environment and should be especially useful for network consultants, system planners, and system engineers.

Some knowledge of AIX, ATM, and TCP/IP is assumed.

The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization, Poughkeepsie Center.

Dr. Hajo Kitzhöfer is an Advisory International Technical Support Organization (ITSO) Specialist for RS/6000 SP at the Poughkeepsie Center. Before joining ITSO, he worked as an SP Specialist at the RS/6000 and AIX Competence Center, IBM Germany. He has worked at IBM for nine years. His areas of expertise include RS/6000 SP, SMP, and benchmarks. He now specializes in SP system management, SP performance tuning, and SP hardware. Hajo holds a Ph.D. degree in Electrical Engineering from the Ruhr University of Bochum (RUB).

Cindy Kueck Young is a Consulting Information Technology Specialist in IBM Advanced Technical Support. She has worked in technical support for IBM UNIX products since 1984 and currently provides RS/6000 technical marketing support for networking technologies, such as Token Ring, Ethernet, X.25, and ATM. Cindy holds a BS degree from the University of Missouri at Columbia.

Thanks to the following people for their invaluable contributions to this project:

Frank Dea
Herman Dierks
Baltazar De Leon
Lon Hall
Jim Smith
Thomas Weaver
Aruna Yedavilli
IBM Austin, U.S.A.

Comments welcome

Your comments are important to us!

We want our Redbooks to be as helpful as possible. Please send us your comments about this or other Redbooks in one of the following ways:

- Fax the evaluation form found in "IBM Redbooks review" on page 309 to the fax number shown on the form.
- Use the online evaluation form found at `http://www.redbooks.ibm.com/`
- Send your comments in an Internet note to `redbook@us.ibm.com`

Chapter 1. ATM introduction and overview

Fast packet research began at university and research labs in the early 1980s. By 1998, the CCITT (now renamed the International Telegraph and Telephone Union - Telecommunications Standardization Sector or ITU-T) adopted a cell relay approach for broadband networking standards that ultimately became what we know today as Asynchronous Transfer Mode (ATM). In 1989, the ITU-T selected a fixed length, 53-byte cell as a compromise to accommodate mixed data types on the same connection.

With cell relay networking, small packets or cells of information, fixed in size, are relayed from network node to network node until they are delivered to the final destination. This approach differs from other networking approaches, such as X.25, Frame Relay, and traditional LAN technologies, which transmit variable length packets or frames.

The ATM Forum was established in October of 1991. The ATM Forum's goal is to work with existing standard groups to define ATM specifications as quickly as possible. At the time of this writing, there are more than 500 ATM forum member companies worldwide. These members represent users, computer and network equipment manufacturers, long distance carriers, and industry researchers.

ATM is an international standard for high speed, cell relay networking. Much of the excitement surrounding ATM stems from its promise of high-speed networking, architected to transport any mixture of voice, video, and traditional computer data across the local, campus, metropolitan, and wide area networks (LANs, MANs, and WANs, respectively).

ATM is a set of rules and standards that define a communication method to satisfy the needs of current and evolving applications. The major components are end nodes, in our case RS/6000s, and network nodes, the switches. ATM defines the interfaces between these components. ATM specifies the communication layers, services, and application mapping to the ATM services on the end nodes.

What sets ATM apart from other networking methods is its design, which includes:

- Switched
- Cell-based
- Connection-oriented

Switched network - In a switched network, the full bandwidth of the physical connection is available between nodes. With shared bandwidth LANs, such as Token Ring and Ethernet, the nodes connected to a segment share the bandwidth of that segment.

Cell based - Although most network technologies are packet-based, ATM differs in that a fixed length packet 53 byte cell is specified. Of those bytes, 48 bytes are payload and 5 bytes are header. The packet size is a compromise to facilitate different application requirements. Fixed length packets in a switched network will be received at the end node at regular intervals. This allows applications that are timing-dependent.

Connection-oriented - In connection-oriented networks, the path between the end nodes is established at the time the connection is made. During a session, the connection appears to be point-to-point. In a switched connection-oriented network, cells are guaranteed to be delivered in the same sequence they are sent. Cells can be lost in the network, but a cell can never pass the cells sent before it.

ATM offers a connection-oriented, switch-based networking technology with fixed length cells. With ATM, data transfer starts after an end-to-end virtual channel connection has been established across the network to link two systems. While this underlying concept is connection-oriented, connectionless applications also may use ATM. ATM networking is based upon using a switch to provide dedicated bandwidth rather than sharing the media as with traditional LAN connections. For example, with the AIX ATM subsystem, each connection across the switch can provide throughput of up to 155 megabits per second (Mbps) rather than the network providing total throughput of 155 Mbps with users vying for a share of the capacity.

Unlike leased line links, however, this dedicated bandwidth is provided only for the duration of the connection. The use of fixed length cells makes it possible to guarantee service to time-dependent (or isochronous) applications, such as real time voice and video, and allows very fast hardware-based cell switching. The use of fixed length cells provides the predictability of circuit switching with the flexibility of packet switching.

ATM provides multiple concurrent (logical) connections over a single cable. ATM allows users to multiplex several connections simultaneously using a single physical connection. For example, IBMs current AIX ATM subsystem supports up to 2048 concurrent connections per port.

ATM pushes error control outside the network to the user Interface. The ATM networking standards capitalize upon today's high-speed, reliable communication links. Traffic is not delayed by error detection/correction at each intermediate node within the network. Performing error checking at each step in the network can have a severe impact on performance, thus slowing overall throughput. With X.25, for example, the packet is received into a buffer at each network node along the path through the network. The node performs error checking and, if necessary, requests retransmission until the packet can be sent on correctly. This requires buffers and processing intelligence at each network node.

Today, network links have become far more reliable, and errors are relatively rare. For this reason, it is much more efficient to assume error-free data and perform checking only at the end nodes. This approach ensures increased throughput.

ATM can use the network cabling currently installed. ATM can be used with voice or data grade Unshielded Twisted Pair (UTP) wiring (UTP categories 3 and 5, respectively) as well as cabling for other networking cable plans, such as Shielded Twisted Pair (STP) and fiber optic cable. This means that users will not have to face costly rewiring to adopt ATM networking technology.

ATM supports a wide range of transmission speed end environments. ATM has no inherent speed limitations. The design of the interlace and lower layers of the protocol model is the same for local area network (LAN), metropolitan area network (MAN), and wide area network (WAN) networking environments with no predefined distance limits. Only the type of data being transmitted (based upon the ATM adaptation layer service) effects the class of networking services provided by the network.

ATM enables Virtual LAN (VLAN) and logical network operations. Whereas most LAN network operations are defined by the physical LAN segments, ATM is not. For example, when two TCP/IP networks are joined with a router, their IP network addresses must be different. When a TCP/IP host (workstation) is moved to a location on the other side of the router, the host's IP address must be changed to match the new network address scheme. With ATM, TCP/IP users are no longer restricted by the physical LAN network. They are free to participate in multiple, logical IP subnets simultaneously with others interconnected by the same ATM network but not necessarily in the same geographical vicinity.

ATM is an evolution from today’s networks and applications. Networks may be public or private. Large companies or organizations may build and manage their own internal network in order to maintain full control. For smaller organizations, however, it can sometimes be beneficial to purchase network services from a network supplier. The network supplier takes responsibility for rerouting in the event of link or network outages as well as all other aspects of network maintenance.

ATM is more of an evolution because computer vendors offer products, such as switches and hubs, that will allow today's typical LAN topologies, such as Ethernet, Token Ring, and FDDI, to interoperate with ATM. This ensures users will be able to migrate to ATM, thus allowing coexistence while the evolution occurs at a pace desired by the user.

1.1 What is driving ATM?

Figure 1 shows the bandwidth requirements of both present and future applications. This rapid escalation in throughput demand is perhaps the single most important factor driving the move to ATM networking.

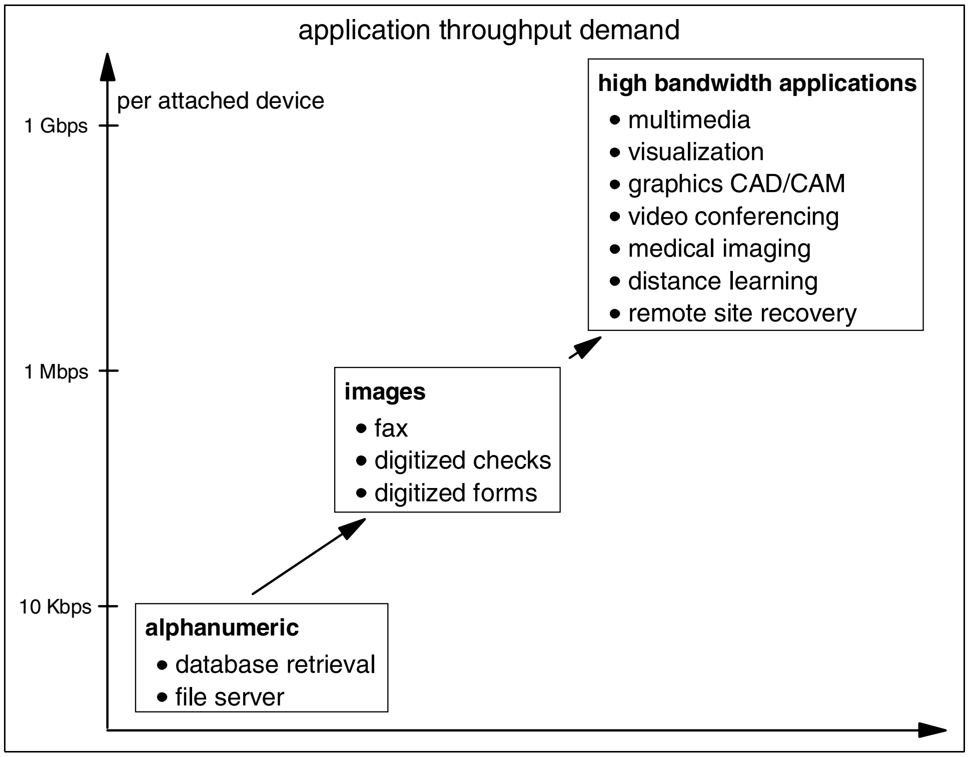


Figure 1. Application throughput demand

There are two important phenomena driving this escalation in bandwidth requirements beyond the capacity typically provided by today's local area networks (LANs) and wide area networks (WANs). First, new applications, such as video conferencing, multimedia, video serving, collaborative computing, and real time video, by their very nature, require increased bandwidth.

Second, with the widespread use of PCs and intelligent workstations, the sheer number of users being networked today is increasing rapidly, therefore, requiring more network bandwidth than before. While the network has typically been called upon to transport alphanumeric applications, which do not inherently have high bandwidth requirements, the increasing number of users on the network drives the need for higher bandwidth.

Conventional LAN technologies can transport digitized images to small work groups. But, new applications that use image files (used in bank applications) and that process electronic copies of checks rather than handling and storing original paper copies, require additional bandwidth. Other industries, such as insurance companies and hospitals, are moving forward paperless environments by using electronic file folders to hold digitized forms, photos, customer records, video tape, and test results.

1.2 Information types and requirements

The evolving world of communications has produced the desire to transmit several types of information across a single communications technology. The most common types of information are:

- Voice - Real time and/or compressed
- Video - Real time and/or compressed
- Data - Large and small transmissions
- Multimedia - Any/all of the above
- Emulated leased line - Looks like today's lines

The requirements of the different information types are radically or slightly different among themselves. The only constants are reliability and cost. Even these vary since video can tolerate an occasional lost packet, while data integrity is the holy grail of the computer industry, and cost is relative in each situation. ATM attempts to provide a network solution for all these information types. An ATM network can integrate all these data types in a single network.

The Quality of Service (QoS) for ATM defines parameters, such as cell rates, delay tolerance, discard priority, and error correction requirements. QoS is customized for the type of payload carried across the virtual channel.

For data payloads, for example, errors should usually be detected and the data retransmitted. However, for voice or real time video, retransmission would result in out-of-sequence transmission; so, it would be better for errors to be ignored. Likewise, voice and real time video require guaranteed cell delivery rate to maintain a smooth sound and image. This type of traffic is *isochronous*, or timing sensitive. In most cases, data traffic can tolerate transmission delays and uneven cell rates. Thus, a customized quality of service is provided when each virtual channel is established, and multiple payload types can be serviced by a single adapter.

Switch-based technology offers the possibility of higher aggregate bandwidth than shared media technologies, such as today's Ethernet, Token Ring, and FDDI LANs, as well as dedicated bandwidth on each connection. With ATM, each virtual channel gets the full available bandwidth; other virtual channels for other systems connecting through the local ATM switch do not compete for bandwidth as they do on shared media networks. Also, using switch technology allows a wide range of speeds and costs so that service may be customized for different connections within a single enterprise. Since each connection is dedicated, the total bandwidth in the network is dependent only on the type of switch used.

Finally, the ATM cell is always 53 octets or bytes. The header is five octets in length, and the payload, or data portion, is 48 bytes, as Figure 2 on page 7 depicts.

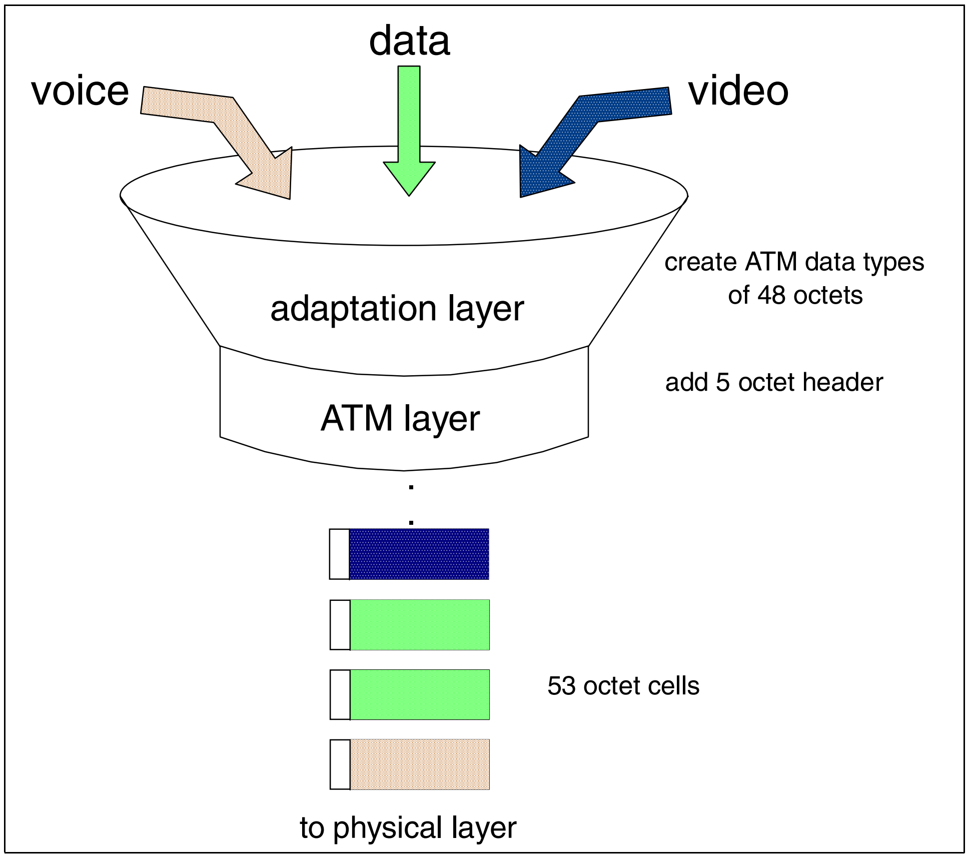


Figure 2. ATM layer model

The small cell size was chosen to allow data, voice, and video (or ATM payload) to be intermixed on the same connection. The fixed length allows easier hardware and software design and makes ATM scalable in both distance and speed. The ATM header contains virtual connection information that allows non-destructive rerouting of connections in the event link failures occur.

1.2.1 ATM adaption layer

The ATM adaption layer (AAL) provides mapping of applications to the appropriate ATM service. This layer segments and assembles the information into 48-octect (byte) payloads and then hands the payload over to the ATM layer. For incoming traffic, the AAL reverses this process; it receives the payload from the ATM layer and reassembles the 48-octet payloads into user information.

The standard groups and the ATM Forum have identified four adaptation layer types that are intended to provide service to various kinds of traffic as Figure 3 depicts. These classes of service are differentiated by requirements for timing relationship, variability of bit rate, and connection mode.

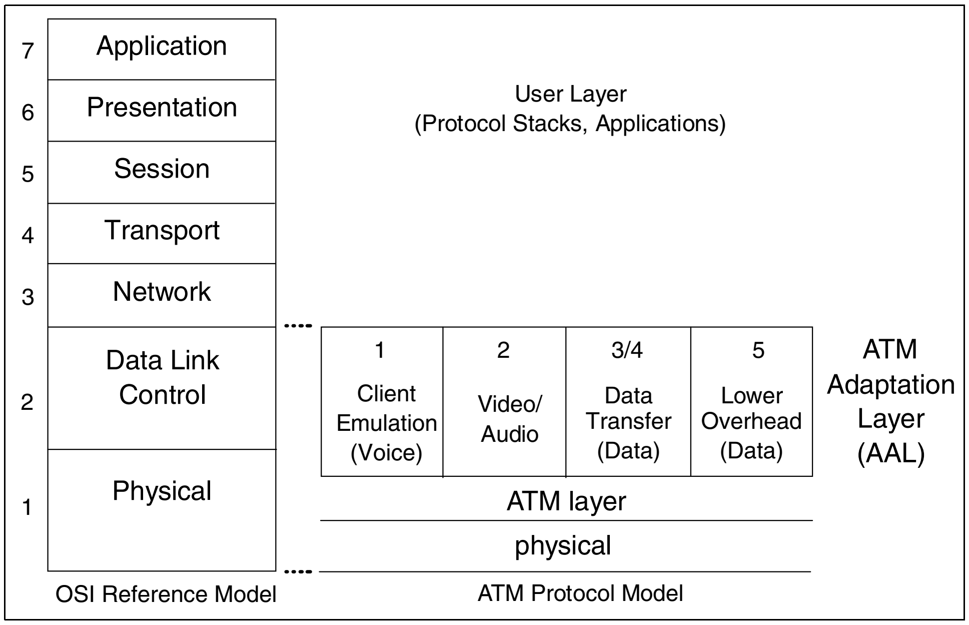


Figure 3. ATM protocol model

AAL 5, which is supported by the RISC System/6000's initial ATM subsystem, is the most widely implemented adaptation layer today. In the future, other adaptation layers may be specified to meet new requirements.

AAL 1

In AAL 1, Constant Bit Rate (CBR) services handle traffic where there is a strong timing sensitivity between the source and the destination. Examples of applications include voice traffic, constant bit rate video, and the emulation of leased lines.

AAL 2

Variable Bit Rate (VBR) timing-sensitive services handle traffic in AAL 2, where a strong timing relationship between the source and the destination is required, but the bit rate may vary. Examples of applications with such requirements include variable bit rate voice and compressed video.

AAL 3/4

Connection-oriented and connectionless VBR data transfer, handled in AAL 3/4, is a complex layer that can handle VBR data using different types of ATM connections. An example of a connection-oriented type of data transfer is a large file transfer, such as traffic on a LAN interconnection. This AAL provides no timing relationship.

AAL 5

AAL 5 is the Simple and Efficient Adaption Layer (SEAL). This layer may be viewed as a simplified version of AAL 3/4 that is designed to meet the requirements of local, high-speed LAN implementations. AAL 5 is intended for either connection-oriented or connectionless VBR services that are not timing sensitive.

ATM layer

The ATM layer adds or removes a 5 octet header to the payload or cell delivered to it. This header contains virtual path and channel identifiers and other Information about the cell. This layer multiplexes the cells into virtual connections and guarantees sequential delivery. The ATM layer passes a 53 octet cell to the physical layer. For incoming information, this process is reversed. The ATM layer removes the 5 byte header and passes the resultant 48 octet package to the AAL layer.

Physical layer

The physical layer sends and receives the 53 octet cell from the ATM Layer and creates the transmission frame for the specific physical network type. Network types include single or multi-mode fiber, shielded or unshielded twisted pair, and coax.

Errors within the data portion of the cell are not detected by the network. The end user equipment or the adaptation layer must do this. However, the network does protect against the misrouting of cells due to header errors. The Header-Error Control (HEC) field in the ATM header is checked and reconstructed at each switch within the network as defined in the NNI specification.

1.3 ATM packet ordering

Real time voice, video, and emulated leased line require Constant Bit Rate (CBR) traffic information sent and received at regular intervals. In ATM terminology, this is service class A (AAL 1 service).

Compressed voice, video, and multimedia applications can use Variable Bit Rate (VBR) traffic, an amount of information during an interval. In ATM terminology, this is service class B (AAL 2 service).

Data, the material we send around now with TCP/IP and other protocols, just has to reliably get there. In ATM terminology, this is service class C or D. Class C (AAL 3/4 service) is connection-oriented, while class D (AAL 5) allows for connectionless protocols.

How fast information is sent and received is determined by the application and how much money you are willing to spend.

To oversimplify how ATM works, let us assume that a Class A application that requires every third cell is sharing a physical link with a Class B application that requires two cells of every 10 cells and two Class C applications, C1 and C2. With this, the cells on the link might look similar to what is shown in Figure 4.

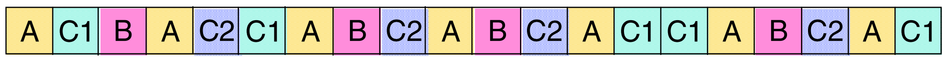


Figure 4. ATM cell distribution

1.4 Virtual channel link

Each ATM adapter provides one physical connection to the network. Each physical connection can support multiple, concurrent logical connections, Virtual Channel Connections (VCCs), to the end user machines on the network. The actual number of concurrent connections possible depends on the configuration of the local switch or hub, but the RISC System/6000 AIX ATM subsystem supports up to 1024 VCCs.

The ATM Forum initially defined virtual channels as one direction only, meaning that two VCs were required for two-way communication. VCs were defined as bi-directional in the UNI V3.0 specification. Thus, users may see switches implementing uni-directional VCs.

When an end user needs to communicate with someone, a call goes out across the network. The logical connection is formed when a virtual channel connection is established. Once the VCC is established, all traffic between these two systems flows as it is if there were a point-to-point link. This means that the first cell (for call set up) carries the destination address, but subsequent cells need not carry the address. Thus, the link is termed *connection-oriented*. In this sense, traditional LANs are *connectionless* because both the source and destination are carried in each frame. VCCs are analogous to the virtual circuit concept used in both X.25 and Frame Relay communications.

1.5 PVC and SVC

Like X.25 and Frame Relay, ATM defines two basic types of virtual channel connections:

- Permanent Virtual Channels (PVCs)
- Switched Virtual Channels (SVCs)

PVCs are semi-permanent and are set up using administrative procedures at the switches, while SVCs are temporary. SVCs are dynamically created when an end node requests a connection to another end node. A VCC is created through the ATM network, and a VP:VC pair is assigned at each end node. SVCs are set up on a call-by-call basis and established using signalling procedures. The connection is terminated when the application stops using it. The result is a VCC between two systems; this connection may be set up across multiple switches.

The concept of virtual paths is unique to ATM. A virtual path (VP) carries a logical bundle of virtual channels that share the next same hop destination. With VPs, the network node’s routing table is simplified because the VCs are switched as a group along the virtual path.

Virtual Path Identifiers (VPIs) and Virtual Channel Identifiers (VCIs) are used to distinguish different VPs and VCs. The VPI and VCI numbers are not end-to-end within the network. Rather, they have meaning only between a pair of ATM network components at each link across the path.

When a connection is established between two end nodes, VP:VCs are connected throughout the network to establish a Virtual Channel Connection, VCC. The VP:VC pairs of the two ends are now connected. Figure 5 on page 12 depicts the virtual channel and virtual path relationship.

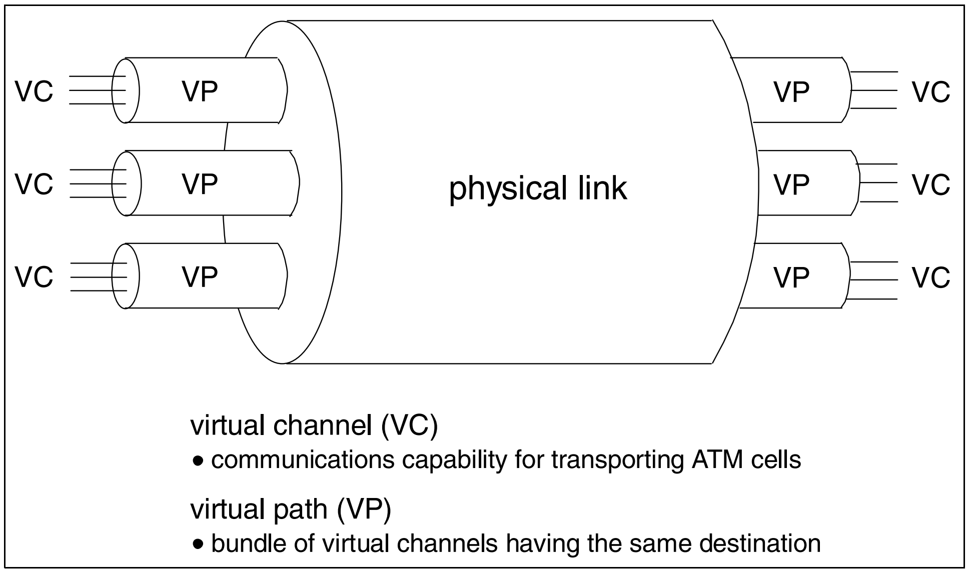

VC
VP
VC
VP
VC
VP
physical link
VP
VC
VP
VC
VP
VC
virtual channel (VC)
• communications capability for transporting ATM cells
virtual path (VP)
• bundle of virtual channels having the same destination

Figure 5. VCs and VPs on a physical link

Information flows on virtual channels. Virtual channels can be grouped together in virtual paths. A physical link can have many virtual paths, each with many virtual channels.

1.6 ATM network topology

The ATM standards define several functional interfaces, including:

- User Network Interface (UNI)
- Network Node Interface (NNI)
- Broadband - Intercarrier Interface (B-ICI)

UNI and B-ICI are analogous to X.25 and X.75, respectively. Like X.25, users may either build their own private network or subscribe to a network owned by either a long distance telephone carrier or an independent company. Figure 6 on page 13 shows an example ATM network layout.

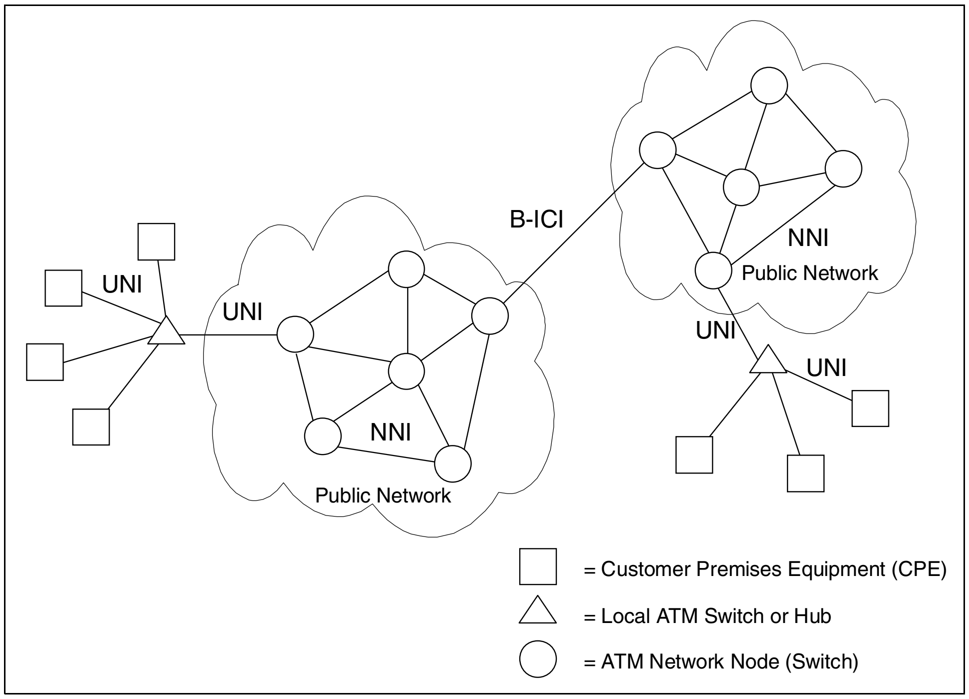

B-ICI
NNI
Public Network
UNI
UNI
UNI
UNI
UNI
NNI
Public Network
□ = Customer Premises Equipment (CPE)
△ = Local ATM Switch or Hub
○ = ATM Network Node (Switch)

Figure 6. Conceptional view of ATM

As Figure 7 on page 14 depicts, the first byte of a packet differentiates between UNI and NNI packets.

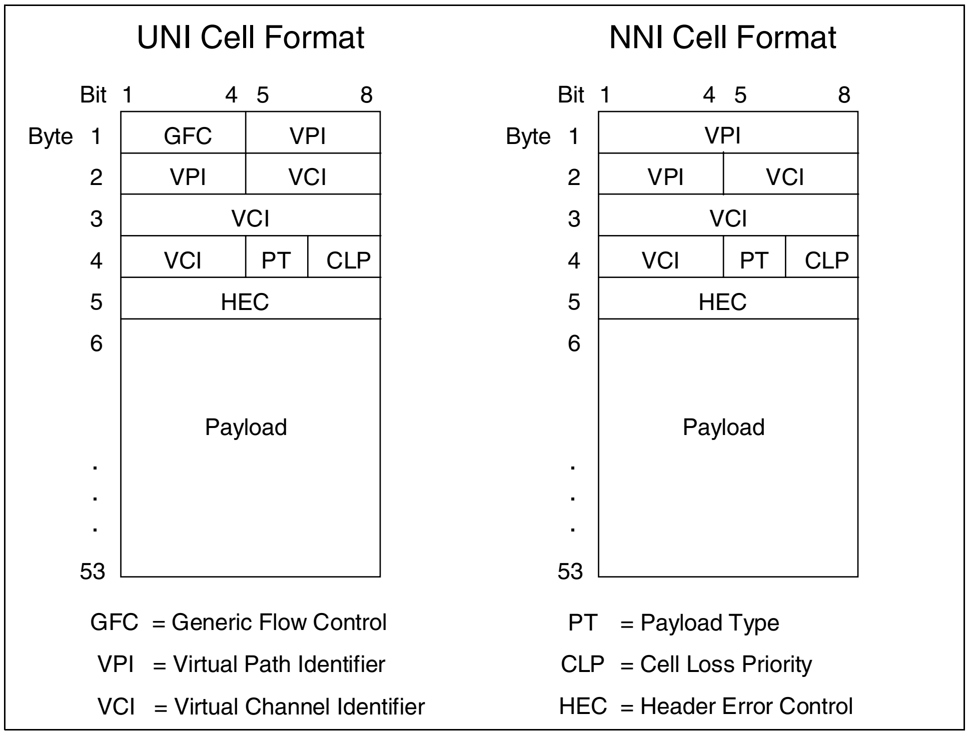

UNI Cell Format
NNI Cell Format
Bit 1 4 5 8
Byte 1 2 3 4 5 6 . . . 53
GFC
VPI
VCI
PT
CLP
HEC
Payload
GFC = Generic Flow Control
VPI = Virtual Path Identifier
VCI = Virtual Channel Identifier
PT = Payload Type
CLP = Cell Loss Priority
HEC = Header Error Control

Figure 7. Packet formats for UNI and NNI packets

1.6.1 Public and private ATM networks

Figure 8 on page 15 shows an ATM network with end nodes, switches, and physical links. An RS/6000 can be attached to a local switch or a switch belonging to an ATM network provider. RS/6000s can also be connected back-to-back as seen in Chapter 6, “Classical IP over ATM” on page 71.

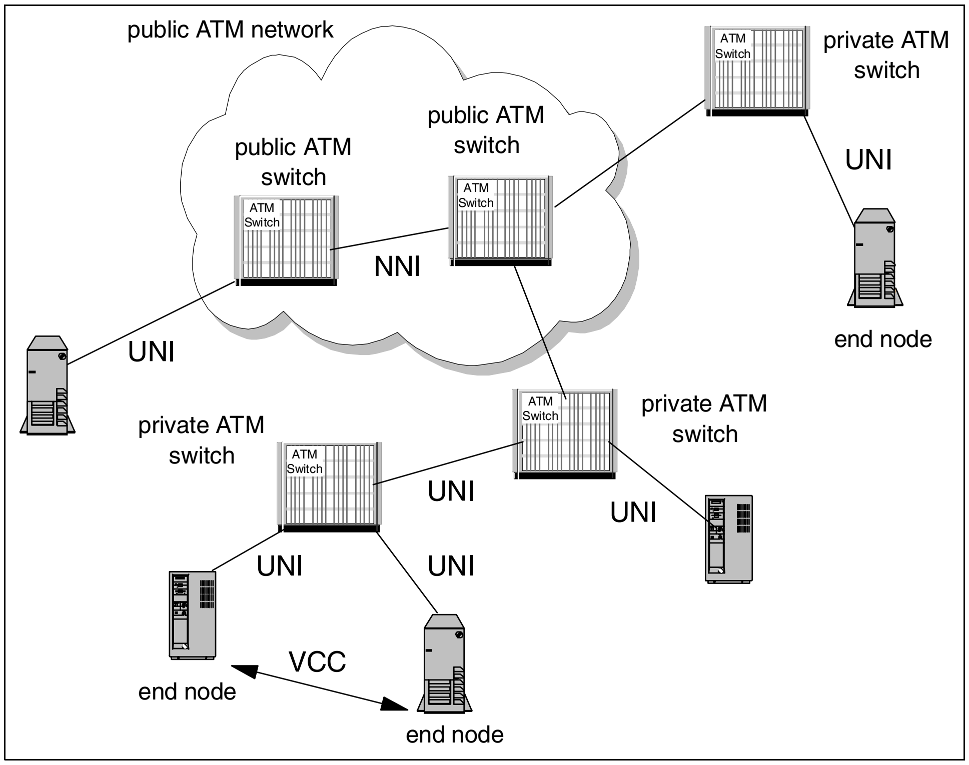


Figure 8. An ATM network

AIX supports ATM Forum UNI Specification V3.0 and V3.1. This allows RS/6000s to be end nodes connected to private or public ATM switches that conform to the same standard UNI.

1.6.2 LAN, MAN, and WAN

ATM networks can be LAN, MAN, WAN, or any combination. ATM networks are not limited by geography.

ATM defines the interfaces between end nodes and switches for private ATM networks, LANs, networks within one common carrier, MANs, and among common carriers, WANs. This means a connection can be established between end nodes anywhere in the ATM network without using routers or gateways. To the end user, it all appears to be local. In Figure 3 on page 5, all the RS/6000s can be defined in the same subnet. Connections can be established between any two nodes without using routers. The end nodes, as shown in the figure, can be dispersed throughout the world.

1.7 Application interfaces to ATM

Early ATM products typically offered one of two common application interfaces to ATM. These protocols (TCP/IP and LAN Emulation) offered directly opposite approaches to ATM, making each of them suitable for different user communities. Both protocols involve clients and servers for SVC operations. TCP/IP supports PVC operation, but LAN Emulation does not. RS/60000 supports both of these application protocols.

1.7.1 Classical IP

TCP/IP (often termed *Classical IP* or *CIP* in ATM discussions) is integral to the UNIX world; so, it was the logical choice for the first application for AIX. By enabling a direct TCP/IP interface to ATM, applications, such as NFS, DCE, and a host of database and client server applications, are also enabled for ATM.

1.7.2 LAN Emulation

Whereas Classical IP offers a direct TCP/IP interface to the ATM network, LAN Emulation (often abbreviated LANE) provides software emulation of traditional LANs (such as Ethernet and Token Ring), thereby hiding ATM from the upper-level applications. With LAN Emulation, these applications run, without change, across an ATM network, providing users a means of coexistence during the migration to ATM networking. LAN Emulation takes frames constructed for Token Ring and Ethernet LANs and converts them to ATM. LAN Emulation may run as software in an end station, or it may run in a network component, such as a bridge, converting Ethernet and Token Ring frames to ATM.

When LAN Emulation is available, AIX protocols, such as SNA, NetBios, and IPX, will be able to run transparently across ATM networks. Although LAN Emulation introduces overhead to applications, it enables a large number of applications already running on workstations and provides an excellent migration path to ATM networking. Thus, it was (logically) the first application offered on PCs and workstations where off-the-shelf software packages are the norm.

1.7.3 User programming interface

LAN Emulation and Classical IP are, primarily, for text payloads. With the extension to sockets in AIX Version 4.2, users have a programming interface that will allow direct access to ATM device drivers (previously, user applications had to go through TCP/IP first). This programming interface provides access to ATM QoS and enables the possibility of video, voice, and data applications. This programming interface is the first direct ATM interface offered by any IBM product.

1.7.4 Interoperability

Figure 9 shows the AIX implementation of sockets and Classical IP as well as the relationship they would have with LAN Emulation.

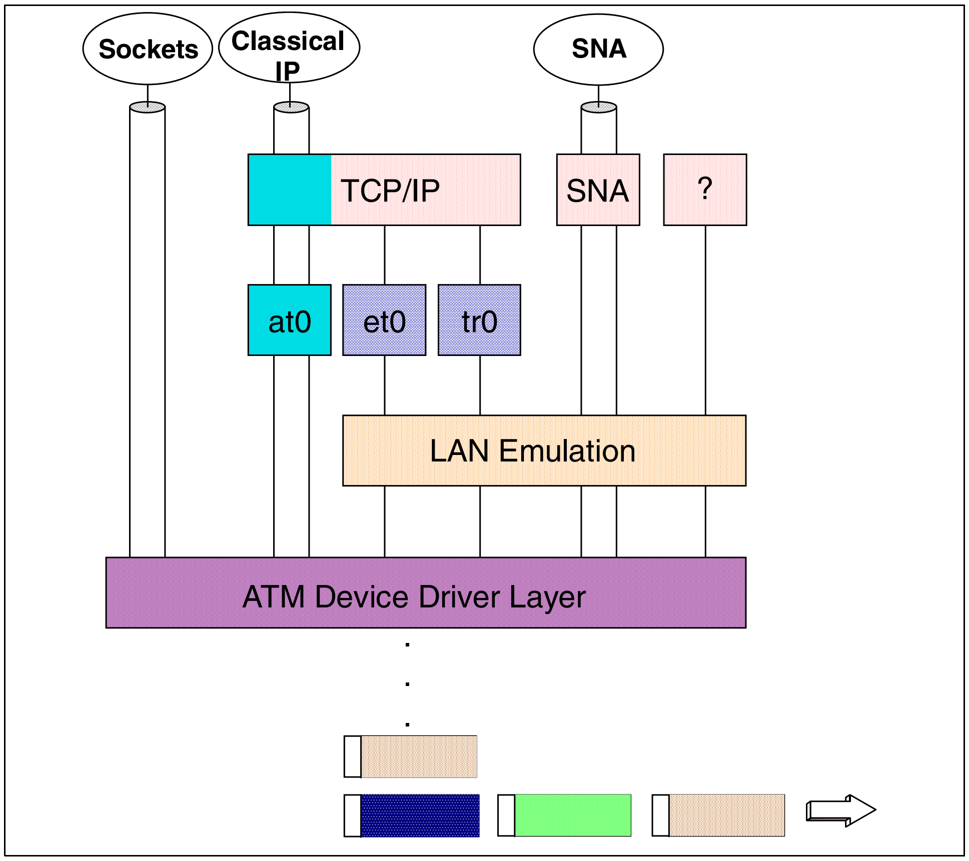


Figure 9. ATM protocol implementation

Users may run multiple protocols over a single ATM network, but, just as in a traditional LAN environment, they may only communicate directly with others running the same type of application. Classical IP and LAN Emulation are different protocols and are not compatible with each other. (This is true even if the LAN Emulation user is running a TCP/IP application written for Token Ring or Ethernet. The only way to exchange data between those two environments would be via an IP router with both a LAN and Classical IP interface.)

Socket applications are custom-written, and both system connections, via VCC, must run the same application.

1.8 TCP/IP addressing review

Three types of addresses are involved in delivering TCP/IP information to an end station on a network. A symbolic name (a host name, such as copper or brass) is for the end users' convenience. A 4-byte IP address, such as 192.168.2.3 or 192.168.2.5, has meaning in the TCP/IP V4 protocol. The 6-byte MAC (Media Access Control) address, such as 00.06.ab.12.34.03 or 08:00:5a:75:02:3d, identifies an individual network adapter and is used to deliver the information on the physical network wire.

To communicate with another node on the local TCP/IP network, the host name keyed by the user must be resolved to an IP address (for routing through the IP network), which, in turn, must be resolved to a MAC address (for delivery on the wire). Here is how it works.

- Typically, the end user's local /etc/hosts file (or file entries at the domain name server system) performs the host name to IP address translation.
- The sending system consults its local ARP cache for the MAC address that corresponds with the IP address. If not there, it sends out a request for the MAC information.

  On traditional LANs, such as Token Ring and Ethernet, this request is in the form of an ARP packet broadcast that is received by all stations in the local network. The system with the matching IP address responds with its MAC address, and the requester completes the ARP cache entry. ATM is not, however, a shared media; so, it cannot support such network broadcasts. This ARP cache update problem is solved using an ARP server system, as described in the "ATM ARP client/server environment" section of this paper. The ARP server receives and services the requests for MAC-type addressing information on the Classical IP network.

  TCP/IP systems maintain a table (the ARP cache) of the most recently used IP to MAC (or Alternate) address mappings. RS/6000 users display and manipulate the ARP cache with the `arp` command.

MAC addresses are burned into the network adapter during manufacture. They are expected to be unique in the network, and they should also, in theory, be unique in the world because the IEEE (Institute of Electrical and Electronics Engineers) assigns specific MAC address ranges to different manufacturers. Users may override the *burned-in* address via the SMIT Alternate Address field in the adapter's Change/Show Characteristics menu. There are a number of reasons for doing this. In ATM networks, the MAC address becomes part of the ATM address so that installing a new adapter on a server system can be transparent to the clients if the old MAC address is reinstated via the Alternate Address field.

1.9 ATM address construction

End stations on ATM networks are identified by a network unique, 20-byte ATM address that is constructed each time the station joins the network. In Figure 10, the ATM address for system copper is:

```
47.00.05.80.ff.e1.00.00.00.f2.15.16.45.00.06.ab.12.34.03.00
```

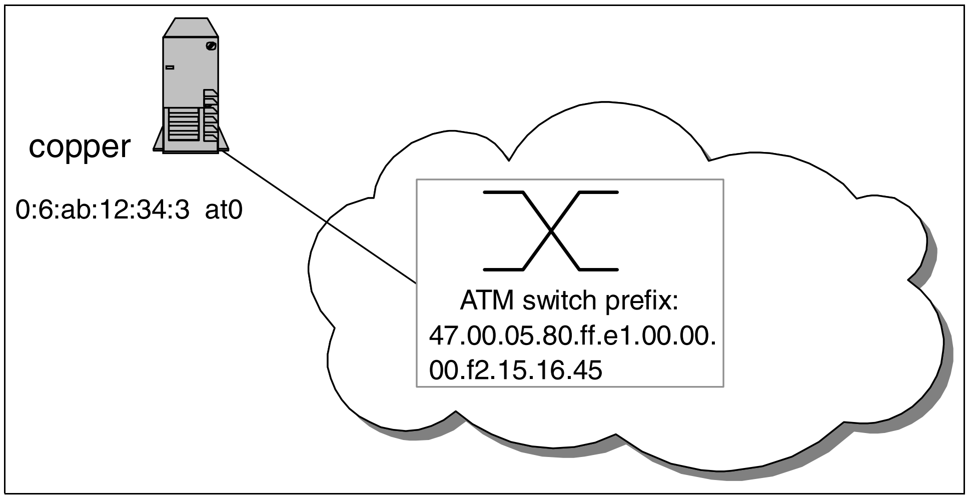


Figure 10. ATM address construction

Here is how that address is built:

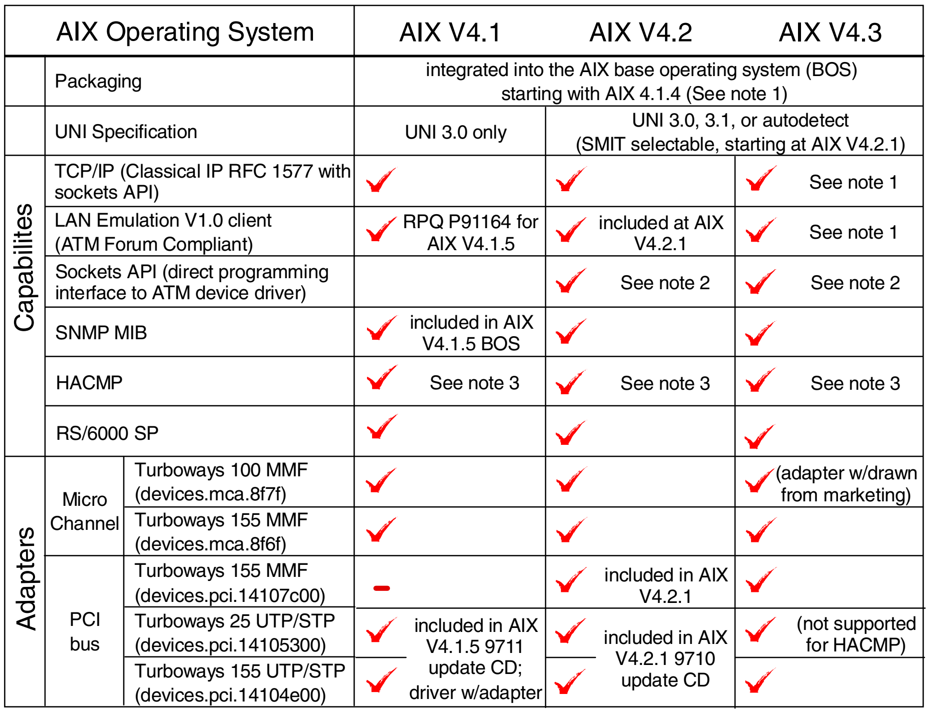
1. When copper joins the ATM network, it registers with the local switch by sending its 6-byte MAC address.
2. The switch pre-pends the unique, 13-byte switch prefix assigned it by the network administrator and returns the resultant ATM address to system copper.
3. The final byte, the *selector byte*, is assigned and used by the upper level application protocol and ignored by the switch.

For LAN Emulation, the interfaces are numbered sequentially; so, on a system configured for two LAN Emulation interfaces, the selector bytes are 0 and 1. For Classical IP, it is the at-interface number in hexadecimal. Thus, the selector byte for system steel's at1 interface on this network is 1. If it had an at12 interface, the selector byte would be 0c.

For simplicity sake, similar MAC and IP addresses are used in this redbook, with the system unique number in the last byte. System platinum, for example, has MAC address 06.ab.12.34.04 and IP address 192.168.2.4.

Chapter 2. ATM on the RS/6000

ATM support is integrated into the AIX V4 base operating system, therefore, the chart shown in Figure 11 is organized by software level. Find the desired operating system level, and the ATM support is summarized in the column beneath it.



AIX Operating System			AIX V4.1	AIX V4.2	AIX V4.3
	Packaging		integrated into the AIX base operating system (BOS) starting with AIX 4.1.4 (See note 1)		
	UNI Specification		UNI 3.0 only	UNI 3.0, 3.1, or autodetect (SMIT selectable, starting at AIX V4.2.1)	
Capabilites	TCP/IP (Classical IP RFC 1577 with sockets API)		✓	✓	✓ See note 1
Capabilites	LAN Emulation V1.0 client (ATM Forum Compliant)		✓ RPQ P91164 for AIX V4.1.5	✓ included at AIX V4.2.1	✓ See note 1
Capabilites	Sockets API (direct programming interface to ATM device driver)			✓ See note 2	✓ See note 2
Capabilites	SNMP MIB		✓ included in AIX V4.1.5 BOS	✓	✓
Capabilites	HACMP		✓ See note 3	✓ See note 3	✓ See note 3
Capabilites	RS/6000 SP		✓	✓	✓
Adapters	Micro Channel	Turboways 100 MMF (devices.mca.8f7f)	✓	✓	✓ (adapter w/drawn from marketing)
Adapters	Micro Channel	Turboways 155 MMF (devices.mca.8f6f)	✓	✓	✓
Adapters	PCI bus	Turboways 155 MMF (devices.pci.14107c00)	–	✓ included in AIX V4.2.1	✓
Adapters	PCI bus	Turboways 25 UTP/STP (devices.pci.14105300)	✓ included in AIX V4.1.5 9711 update CD; driver w/adapter	✓ included in AIX V4.2.1 9710 update CD	✓ (not supported for HACMP)
Adapters	PCI bus	Turboways 155 UTP/STP (devices.pci.14104e00)	✓	✓	✓

Figure 11. RS/6000 ATM support summary

Note

1. Support for Classical IP temporary backup ARP server (RFC 2225) was added for AIX V4.3 systems via PTF. AIX V4.3.3 includes Multiprotocol Over ATM (MPOA) client support for ATM LAN Emulation.
2. The sockets API enables video/voice applications. With it, user-written applications can specify signaling parameters and Quality of Service and can run directly over ATM AAL5 without TCP/IP. A whitepaper and sample programs are available from URL:
   `http://www.rs6000.ibm.com/resource/technology`
3. HACMP V4.1.1 (plus PTFs) and HACMP V4.2 supports Classical IP over ATM atop the Turboways 155 MMF MCA adapter on systems running AIX V4.1.4 and up. HACMP V4.2.2 and AIX V4.2.1 (and up) supports both Classical IP and ATM LAN Emulation on all three MCA and PCI bus Turboways 155 adapters. HACMP V4.3.1 adds ATM MAC address takeover. (There is no HACMP support of ATM LAN Emulation on AIX V4.1.5 or on the Turboways 25 PCI adapter.)

2.1 Supported protocols and standards

The AIX device driver is compliant with the UNI 3.0 and UNI 3.1 specifications for the User to Network Interface as defined by the ATM Forum. Classical Internet Protocol (CIP or CLIP) conforms to RFC 1577, which defines the mapping and resolution of 20 byte ATM addresses to TCP/IP addresses using ATMARP over AAL5 connections. Logical IP Subnets (LIS) are defined with one or more ARP servers (Primary and Backup) providing arp services for the set of clients sharing the same subnet address. Connectivity between members of the subnet require the establishment of a virtual circuit between the two communicating stations. Direct communication with other units outside of the Logical IP Subnet is not defined. The two stations may be on the same switch or separated by a large network of ATM switches. In an LIS, each member has the same subnet address and subnet mask and the same MTU support for ATMARP.

Backup ARP server support (defined in RFC 2225) is provided in AIX 4.3.3 and later. Configuration details are covered later in Section 6.2.2 “SMIT Configuration”.

AIX support is available for AAL5 traffic only. The Virtual Path must be zero. MCA adapters are capable of supporting 1024 virtual circuits. The device driver default is 256.

A Logical IP Subnet (LIS) client will use three circuits just to maintain connectivity with the switch and the ARP server, and additionally, for every LIS partner that may be active and in communication with this adapter. Virtual circuits can be closed and reused as needed if there are none available for outgoing calls. However, incoming calls will be rejected if there are no virtual circuits available.

In summary, these are the standards currently adhered to by the AIX ATM subsystems and adapters:

- CCITT 1.361
- ATM Forum UNI specification V3.0 and UNI V3.1
- ANSI T1S1.5/92-002R3
- IETF RFC 1577 Classical IP and ARP over ATM
- Classical IP (TCP/IP) RFC 1577, RFC 2225
- ATM LAN Emulation V 1.0 and V 2.0 client
- Multiprotocol over ATM (MPOA) client

ATM Forum specifications and RFCs are available on the Internet. (See section D.4, “Referenced Web sites” on page 282 for URLs.)

2.2 AIX ATM capabilities

At this writing, AIX supports ATM as follows:

- 100 and 155 Mbps
- AAL 5 only
- UNI 3.0, UNI 3.1, and autodetect
- Sockets API (AIX Version 4.2)
- SNMP MIB

2.3 Supported ATM adapters

The following is a list of the currently supported ATM adapters:

- F/C 2963 Turboways 155 PCI UTP ATM adapter (PCI ID 14107c00)
- F/C 2988 Turboways 155 PCI MMF (Multi-Mode Fiber) ATM adapter (PCI ID 14104e00)
- F/C 2989 Turboways155 ATM adapter (Microchannel POSID 8f67)

- F/C 2984 Turboways 100 ATM adapter (Microchannel POSID 8f7f)

Device drivers, microcode, and diagnostics are supplied with AIX 4.1.4 and later for these adapters. For more details, see Appendix B, “Detailed adapter information” on page 273.

Chapter 3. Planning

ATM on the RS/6000 requires the following components:

- AIX operating system (with the desired ATM support)
- ATM adapter and adapter cable
- Local ATM switch

For ATM LAN Emulation environments, users must also plan to include the following components:

- ATM LAN Emulation Service functions
- Upper-layer protocol and application software (for example, SNA, IPX/SPX)

See Appendix A, “Planning check list” on page 269 for a planning check list.

IBM offers Classical IP (with the TCP/IP sockets API) and ATM LAN Emulation, the most popular data protocols in today’s ATM networks. These two protocols represent opposite approaches to networking with ATM. Whereas Classical IP systems run TCP/IP directly atop ATM via an IP interface, LAN Emulation hides ATM from the upper-layer protocol, thereby providing ATM connectivity for applications written to operate over an Ethernet or Token Ring LAN. In addition, a sockets API (included in AIX Version 4.2) provides a direct programming interface to the ATM device driver.

3.1 Classical IP

ATM support for TCP/IP communications (defined in RFC 1577 "Classical IP and ARP over ATM") is included as part of the base operating system beginning with AIX Version 4.1.4. The AIX ATM support allows communication with other IBM or OEM hosts running Classical IP. RS/6000s may be configured as Classical IP ARP clients or servers. Support for a backup ARP server scheme (RFC 2225) is included beginning with AIX 4.3.3.

3.2 LAN Emulation

ATM Forum compliant ATM LAN Emulation V1.0 client software is included in the base operating system beginning with AIX Version 4.2.1. It supports TCP/IP, IBM Communications server (with LU0, LU1, LU2, LU3, LU6.2), AIX connections (with IPX/SPX, NetBEUI, and AppleTalk) Streams Data Link Programming Interface (DLPI), and AIX Generic Data Link Control. The RPQ

P91164 provides LAN Emulation support for TCP/IP and IBM Communications server on AIX Version 4.1.5 systems. ATM LAN Emulation service functions are not a part of the AIX software and must be provided elsewhere in the network. It is composed of three servers that work together but need not exist in the same place within the network:

- LAN Emulation Server (LES)
- Broadcast and Unknown Server (BUS) are required services
- LAN Emulation Configuration Server (LECS) is recommended but optional.

3.3 Multiprotocol over ATM

AIX V4.3.3 includes Multiprotocol Over ATM (MPOA) client support for standard and 802.3 Ethernet frame types. MPOA creates shortcuts to minimize IP routing hops between end systems, thereby improving performance across an ATM network.

3.4 Interoperability

Systems running Classical IP and ARP over ATM cannot communicate directly with systems running ATM LAN Emulation; these are different, incompatible protocols. TCP/IP traffic can, however, be passed between these environments via an IP router system that has an interface of each type.

3.5 Hardware considerations

Cabling types and speeds must match between the RS/6000 adapter and the local switch. The switch provides the link between the ports, thus making connections, regardless of individual port speed.

Specific ATM adapters supported and maximum adapters allowed per system varies by RS/6000 model. The large Micro Channel cards do not fit in all models. Conversely, an adapter may physically fit in a system, but IBM may not offer support for it.

Back-to-back ATM operation (without a local ATM switch) works fine, but it is limited to Classical IP PVC operations between two systems, making it most useful for self-education or as an inexpensive application development test bed. Many companies will obtain wide area network access from an external ATM network provider, but there are no direct dependencies between the provider and RS/6000 configuration. Instead, the dependency is upon configuration of the local ATM switch, which is most likely an in-house component.

3.6 System considerations

ATM enables Virtual LAN (VLAN) operations, which, in turn, pose unique considerations for RS/6000 implementers. Generically, a VLAN is a group of devices that use hardware-based switching to communicate as if they were attached to the same LAN as a single broadcast domain. LAN Emulation ELANs (emulated LANs) and Classical IP LISs (logical IP subnets) are VLAN implementations in ATM. In the non-ATM world, some LAN switches also support VLAN creation by linking ports through administrative procedures. AIX ATM users may participate in multiple VLANs simultaneously, but because they can only communicate with users defined for the same LIS or ELAN, they must be able to define multiple IP addresses and emulated LAN adapters atop a single ATM adapter.

Users can configure up to 256 TCP/IP network addresses on an AIX system running ATM. The interfaces, numbered at0 through at255, can be added in any sequence and, when multiple ATM adapters exist, may be assigned to any adapter. The ability to configure multiple IP addresses on a single adapter is unique to ATM and defined in TCP/IP RFC 1577. Users must define each TCP/IP interface on a RS/6000, regardless of the network type, for a unique IP subnet. (For example, assuming IP address 192.2.2.2 and netmask 255.255.255.0, 192.5.2.3 is for a unique IP subnet, whereas 192.2.2.3 is not.)

Although currently limited by software to 32 per port, an RS/6000 could, theoretically, participate in up to 256 ELANs. The LAN Emulation client is a logical LAN adapter (even using device names, such as ent1 or tok0) and, as in a traditional Ethernet and Token Ring LAN environment, multiple, upper-layer applications can be configured to run atop this logical adapter. It would be unusual for a user environment to require more than a couple of ELANs.

In TCP/IP environments, all information (IP addresses, client/server applications, and so on) needed to plan a traditional TCP/IP network will be needed for ATM networks. In addition, since users can define multiple IP addresses per adapter, thereby creating multiple, overlaying virtual networks, diagrams for each logical IP subnet (LIS) and emulated LAN (ELAN) should include:

- Interface number (for example, at0) and MTU size.

- Burned-in MAC adapter address (or alternate adapter address to be used).
- Host names, Internet addresses, and subnet masks.
- Domain name, nameserver, gateway/router name (if applicable).
- For Classical IP PVCs, VPI and VCI numbers configured at ATM switch.
- For Classical IP SVCs, the full (20-byte) ATM address of the ARP Server.
- For LAN Emulation, the full ATM address of the LAN Emulation Service devices unless the "well known" addresses will be used. (SVCs are required for LAN Emulation.)

TCP/IP broadcasts and ARPs are not sent across point-to-point (connection oriented) connections such as Classical IP, X.25, PPP, and SLIP. Therefore, TCP/IP client/server applications in which the client discovers the server by broadcasting will not work. This means items, such as rwhod, NIS, NCS, timed, and so forth, will not work over Classical IP, but should work over LANE. ATM does, however, provide plenty of bandwidth for high-traffic applications, such as NFS and X-Windows.

3.7 HACMP information

The latest version of HACMP supports ATM Classical IP, ATM LAN Emulation, and MAC address takeover. Support is available in various forms:

- PTFs for HACMP V4.1.1 running on AIX V4.1.4 (released in May 1996)
- HACMP V4.2 plus AIX V4.1.4
- HACMP V4.2 plus AIX V4.2
- HACMP V4.3 plus AIX V4.3.3

There are special considerations when running HACMP with ATM. ATM is a connection-oriented networking method. Do not view ATM as a drop-in replacement for connectionless methods, such as Ethernet and Token Ring. One should allow a little time to plan for HACMP in an ATM environment.

3.8 Application dependent considerations

Depending on the type of application, special ATM settings should be considered. The following sections will give some recommendations.

3.8.1 ADSM configuration

ADSM applications need special tuning considerations as servers using ADSM over ATM may encounter problems with loss of connections or incomplete backups. ADSM backups are often done on cron jobs for clients during times where system activity is minimal. Client backups are often scheduled in large groups at a particular time of day. This causes a peak loading situation for the server. This is also compounded on SP nodes where the SP Switch is also present and used for backups.

ADSM uses a privately defined parameter, TCPWindowSize, to control TCP/IP windowing. It ignores the network options parameters, *tcp_sendspace*, and *tcp_recvspace,* unless the TCPWindowSize is set to zero. For AIX 4.3.3 and later versions, it is best to set the option to zero and let the network options for tcp_sendspace and tcp_recvspace control the amount of data that is used for buffering. For earlier versions, use TCPWindowSize no greater than 128 Kbytes. For ADSM backup and restore using an ATM network adapter, the optimal value for TCPWindowSize is 64 kbytes. This is due to ATM's use of private, pre-mapped buffers and the potential to have dozens of client sessions simultaneously. Because of system loading, contention for the ATM media, and other factors, each session can tie up large amounts of kernel memory space waiting for TCP acknowledgments or sending them. With large numbers of sessions, this adds up quickly. Occasionally, when many sessions are running, connections running over ATM would be dropped due to time-outs and congestion. Several changes have been made. The number of private buffers ATM can configure has been increased for AIX 4.2 and AIX 4.3. For PCI 155 ATM adapters, the device driver can acquire up to 14,000 mbufs if they are available. A new facility has been added to AIX 4.3.3 to configure *tcp_sendspace* and *tcp_recvspace* as part of Interface Specific Network Options (ISNO). A new command, `isno`, has been created to specify network options uniquely for each network interface. To take advantage of this, set ADSM TCPWindowSize to zero.

ADSM uses TCP/IP to pace its input data coming from clients. It leaves unread data on the socket receive queue. This causes a system and TCP/IP buffering problem if too many sessions are holding unread data in mbufs on the ADSM receive sockets. ADSM must manage writing data out to disk, tape, and other media as required for backups and other activities for storage management. Due to congestion, locality of filesets, speed of devices, and error recovery actions, this doesn't always happen as fast as incoming data arrives. ADSM will leave unread data in system mbufs on the socket receive queues until they can be processed. The amount of data is limited by either the TCPWindowSize or tcp_recvspace network option (or isno - interface specific network option). For ATM, we only allow 14,000 mbufs, which is enough for 56 Mbytes of data for Classical IP and, at most, 21 Mbytes for LANE. If this space is exceeded, then ATM cannot function because it is out of buffers and cannot maintain connections to the ATM switch and LANE LES under worst case scenarios. Proper tuning prevents this situation from happening.

For every ATM adapter, backup sessions should be limited to the maximum that can be supported at a given window size for the TCPIP connections required. If you use the recommended 64 Kbytes for TCP/IP send and receive window space, you should not configure ADSM to allow more than:

For Classical IP:

(14,000 x 4096 x .9) / window size per ATM port

For LANE:

(14,000 x 1517 x .9) / window size per ATM port

Example:

An ADSM sever that uses LANE for backups exclusively with window sizes set for 64 K:

21238000 x .9 / 65535 = 291 ADSM backup sessions

An ADSM server that uses Classical IP with window sizes set for 64 K:

57344000 x .9 / 65535 = 787 ADSM backup sessions

Theoretically, these are limits to what can be supported and not have network failures on ATM due to lack of pre-mapped mbufs. Limitations may occur in what is practical to implement in that storage device, system memory, CPU utilization, or other unrelated problems may prevent reaching these goals, or performance may not be optimum with this specific number of concurrent sessions. This data is only provided as a guide. If problems do occur, then ADSM's maximum number of sessions parameter should be used to reduce the number of concurrent sessions so that flawless backups and restores occur.

When doing backups over the SP Switch, the speed of the switch is so much greater than the storage devices being used; therefore, buffers will be held constantly in the socket mbufs waiting for space to copy ship data to storage devices. ADSM must do this using TCP/IP, and this will hold more system mbufs that cannot be used until freed.

There is little advantage in using huge values for TCPWindowsize, as it just allows large numbers of mbufs to be tied up and released at the rate of the operation of the storage subsystems involved. ATM is fast enough to acquire data far faster than data can actually be moved to the devices. Unless the file set is small enough to be written entirely in storage and later written, then there is little to be gained by large window sizes.

It is possible to run ADSM over LANE ports and TCP/IP simultaneously. There are big performance gains in using TCP/IP over Classical IP due to reduced overhead for larger packet sizes. For ADSM, ATM CIP configuration should be setup for a 32 KB data size. The MTU size should be 32838 or larger, and the ATM PDU size should be eight bytes larger than the MTU size.

3.8.2 NFS

NFS servers with hundreds of simultaneously NFS clients sharing file sets over ATM may experience time-outs and loss of connections if the clients simultaneously perform remounts. This is due to a high peak demand for buffers. Code has been changed to allocate more buffers. There does not need to be more than 300 nfsd instances running simultaneously, and if this situation occurs due to the system design, it will be necessary to increase the minimum numbers for MCA small, medium, and large mbufs. For PCI, it may be necessary to increase *rx_que_size*, to avoid this situation. Beyond that setting the nfso (NFS options) nfs_device_specific_bufs=0 should eliminate the problem. Often this must be added to the `rc.net` script to be set on subsequent reboots. For SP nodes, there is usually a common configuration script that must be edited.

3.8.3 HACMP considerations

HACMP running on the server could also be affected if the buffers become so constrained that HA heartbeats are lost and HA cannot reach the network, therefore, resulting in a DSM time-out (panic). A backup network other than ATM or LANE should be implemented (TMSSA, TMSCSI, RS-232, Ethernet) to prevent loss of communications to the HACMP partition. If LANE or ATM CIP is the only path, an HACMP initiated system shutdown is possible if the number of nfsd instances is high and the maximum and minimum buffers have not been increased. Tuning can minimize the possibility of this occurring. The *failure_detection_rate* for the network interface in the HACMP topology should be set to SLOW (18 seconds). Beyond this, the ODM HACMPnim database contains three parameters that can be manually updated with odm commands. A backup of this database should always be done prior to any attempt to alter it. The cycle parameter for the specific network interface is the number of consecutive losses of heartbeats that can be missed prior to switching adapters or TCP/IP interfaces. Increasing this value reduces the sensitivity of HACMP to short duration losses of connection to the network. The hbrate parameter specifies how frequently heartbeats are sent on this interface. This is specified in milliseconds. The grace period is set to one second less than the failure_detection_rate (cycle * hbrate) and is set to 17 seconds whenever the failure rate is set to SLOW.

Switch outages or network congestion may present HACMP problems if the recovery is slow or the congestion persists. This is more likely if the LAN LES is not provided on the local switch. This is a network configuration and performance problem but may look as a problem with the local system.

HACMP 4.3.1 implemented ATM adapter MAC address swapping. TCP/IP address swapping was available in prior releases.

One should review the constraints for the maximum number of sessions for ADSM using ATM if ADSM will be running on an HACMP system. It is possible for ADSM to hold more ATM pre-mapped buffers as the system gets busier and contention for system resources occurs to the point where HACMP heartbeats cannot get through. Tuning ADSM maximum number of sessions and TCP window sizes will avoid this problem.

Chapter 4. Sample Network

Throughout this redbook, we will use the sample network depicted in Figure 12 as our reference network. This network can be built in stages as the reader proceeds through the scenario chapters. The individual scenarios will comprise only parts of our whole setup.

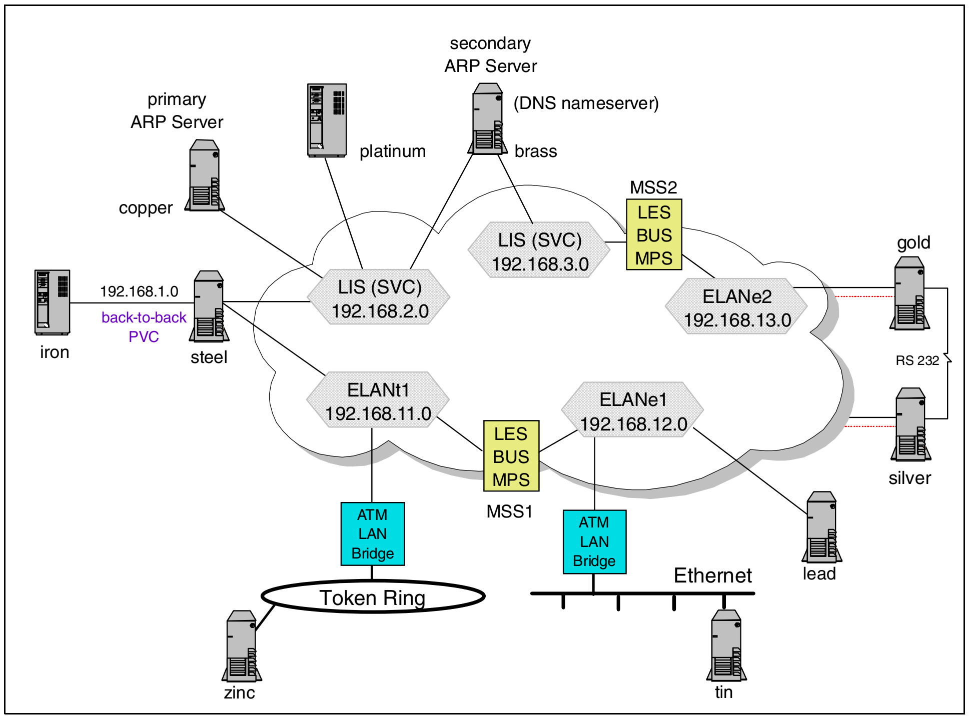


Figure 12. Sample network

Even though we tried to use a network setup that fits to all our test scenarios, we had to change IP addresses and network connections for some special setups, for example, the HACMP test described in Chapter 9, “HACMP scenarios” on page 223.

4.1 Workstations

We used ten RS/6000 systems for sample scenarios. Eight of them were PCI based systems, and two of them were Micro Channel systems. Table 1 gives on overview of our naming conventions.

Table 1. General naming conventions

System number	**System name**	**Bus system**	**Scenario**
1	iron	MCA	PVC only
2	steel	PCI	PVC, SVC, ELANt1
3	copper	PCI	SVC primary ARP server
4	platinum	MCA	RFC 1577 SVC only
5	brass	PCI	SVC, DNS nameserver, backup ARP
6	lead	PCI	ELANe1 only
7	gold	PCI	HACMP, ELANe2
8	silver	PCI	HACMP
9	zinc	PCI	ATM LAN bridge
10	tin	PCI	ATM LAN bridge

4.2 MCA versus PCI

We used mostly RS/6000 PCI based systems for the test because they have two advantages over MCA systems:

- The PCI adapters have much lower latency than the MCA adapters.
- The newer PCI based platforms also have the fastest processor speeds.

4.3 Network components

We used two ATM switches (IBM 8260-A17 with DMM, PNNI, ATM, and MSS). They provided the ATM "cloud" and LAN Emulation Service functions.

Table 2 on page 35 gives an overview of our network layout. We used different subnets for our various scenarios. The format for our scheme was:

192.168.NET#.SYS#

Table 2. Network setup

Network	Scenario	IP address	Netmask
#1	PVC	192.168.1.0	255.255.255.0
#2	SVC	192.168.2.0	255.255.255.0
#3	SVC	192.168.3.0	255.255.255.0
#11	ELANt1	192.168.11.0	255.255.255.0
#12	ELANe1	192.168.12.0	255.255.255.0
#13	ELANe2	192.168.13.0	255.255.255.0
#20	HACMP	192.168.20.0	255.255.255.0
#21	HACMP	192.168.21.0	255.255.255.0

So, system steel is 192.168.1.2 on the PVC, 192.168.2.2 on the SVC, and 192.168.11.2 on the Token Ring ELAN.

4.4 Adapter overview

Table 3 gives an overview of all adapters in our systems.

Table 3. System and adapter overview

System name	Real adapter	Scenario	IP interface	
iron	atm0	PVC	at0	
steel	atm0	PVC	at0	
	atm1	SVC	at1	
	tok0	n/a		Real Token Ring
	atm1 (tok1)	ELANt1	tr1	Token Ring ELAN
copper	atm0	SVC	at0	Primary ARP server
platinum	atm0	SVC	at0	AIX 4.2.1 (RFC 1577)
brass	atm0	SVC	at0	DNS nameserver, second ARP server
	atm0	Second SVC	at1	ARP server
lead	ent0	N/A		Real LAN

System name	Real adapter	Scenario	IP interface	
	atm0 (ent1)	ELANe1	en1	Ethernet ELAN, MPOA
gold	tty0	N/A		HA heartbeat
	atm0	ELANe2 (HA-LE)	en0	MPOA, HACMP
	atm0	HA-LE		HACMP
	atm0	HA-CIP	at0	HACMP
	atm0	HA-CIP	at1	HACMP
silver	tty0	N/A		HA heartbeat
	atm0 (ent0)	ELANe2 (HA-LE)	en0	MPOA, HACMP
	atm1 (ent1)	HA-LE		HACMP
	atm0	HA-CIP	at0	HACMP
	atm1	HA-CIP	at1	HACMP
zinc	tok0	Real T-R net	tr0	Attached via ATM LAN bridge
tin	ent0	Real Ethernet	en0	Attached via ATM LAN bridge

4.5 AIX Version and additional software

All systems were installed with AIX Version 4.3.3 except platinum, which had AIX 4.2.1 installed (see Table 3 on page 35).

For our HACMP test, we used HACMP Version 4.3.1.

Chapter 5. Configuration and attachment

In this chapter, you will find a step-by-step description of the installation process for ATM adapters. It also covers the installation of PCI and MCA adapters.

5.1 Installation considerations

For a successful ATM adapter installation, topics, such as adapter placement, which connection media is used, and which software is required, should be reviewed. These considerations should be part of the planning phase.

5.1.1 Adapter selection

ATM adapters are available in different flavors. One criteria is the I/O bus system used, either Micro Channel (MCA) or PCI. If ATM performance is an issue, a PCI system should be the preferred selection. For more information, see Chapter 11, “ATM performance considerations” on page 253.

Another criteria is the transfer speed, Currently, adapters for 100 Mbps and 155 Mbps are available.

5.1.2 Software

The appropriate software for the ATM adapter needs to be installed. The ATM device driver software is AIX version dependent; so, make sure the latest driver software is installed. The following gives an example on how to check if the device driver software is already installed:

```
$ lslpp -l | grep ATM
bos.atm.atmle              4.3.3.0  COMMITTED  ATM LAN Emulation Client
  ...
  devices.pci.14107c00.com   4.3.3.0  COMMITTED  Common ATM Adapter Software
  devices.pci.14107c00.diag  4.3.3.0  COMMITTED  PCI ATM Adapter (14107c00)
  devices.pci.14107c00.rte   4.3.3.0  COMMITTED  PCI ATM Adapter (14107c00)
```

5.1.3 Adapter placement

For MCA systems, the ATM adapter may be placed in any available Micro Channel slot. The situation is not that easy for PCI systems. Before installing any PCI ATM adapter, you should have a look at the latest *PCI Adapter Placement Reference*, SA38-0538, documentation.

If you are not immediately connecting the adapter to the network, make sure that the wrap connector is attached to the card.

5.1.4 Cabling

The connection media is also a selection criteria for an adapter. For example, 155 Mbps ATM adapters are available with Multi-Mode Fiber (MMF) connection or with Unshielded Twisted Pair (UTP) connection. It depends on which network environment your system will be connected.

5.1.4.1 UTP cable connection

A UTP cable with RJ45 connectors is used to interconnect systems or a system to a switch. The cable must have transmit and receive crossed over. Pins 1 and 2 on each end are crossed to pins 7 and 8, respectively, on the other end, as Figure 13 depicts. When both LED's are green, a signal is detected. This does not mean that the adapter is functional, but it does indicate that the cables are correctly inserted. The adapter must be configured and have an *Available* status in AIX before you can use the LED indicators.

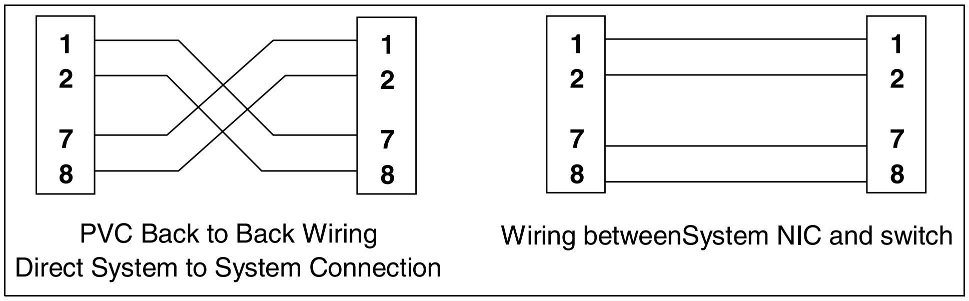

1
2
7
8
1
2
7
8
PVC Back to Back Wiring
Direct System to System Connection
1
2
7
8
1
2
7
8
Wiring betweenSystem NIC and switch

Figure 13. UTP wiring

Cable lengths up to 100 m of two pair CAT 5 wiring may be used. Avoid routing the cables across, or next to, electronic noise sources, such as electric motors, florescent lighting fixtures, and CRTs, as the induced noise can cause CRC errors and dropped cells. Figure 14 on page 39 shows two UTP RJ45 connectors.

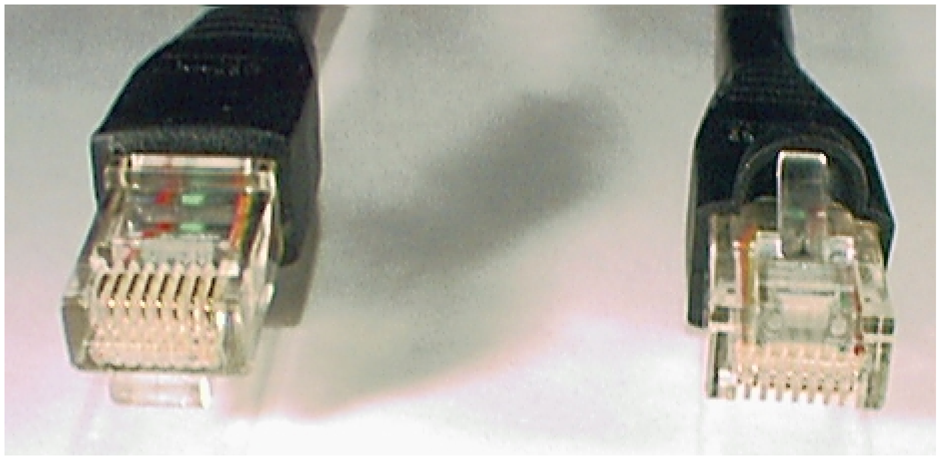


Figure 14. UTP RJ45 connectors

5.1.5 Multi-Mode fiber cabling

Using Multi-Mode fiber for the connections, the maximum transmission distances will be around 3 km. Connections between the RS/6000 ATM adapter and the switch, hub device, or, in a back-to-back configuration, a second RISC System RS/6000, are made with a 62.5 micron Multi-Mode duplex fiber optic cable that is terminated with SC-type connectors. The SC (subscriber) connector is a ANSI standard duplex connector (see Figure 15 on page 40 for more details). The other end of the fiber cable will have to have the appropriate connector for the device at that end. The RS/6000 ATM adapters support only SC-type connectors.

The RS/6000 ATM adapter has two fiber sockets, one for receiving and one for transmitting. To connect two ATM devices in the correct way, you have to connect the transmitting socket of one device to the receiving socket of the other device. In other words, you have to cross the cables.

Orientation can be confirmed by examination of the LEDs on the back of the adapter when the adapter and remote connection are configured. The adapter must be configured in AIX and available to use the LED indicators.

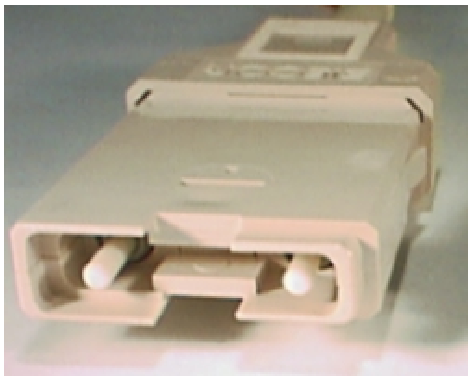
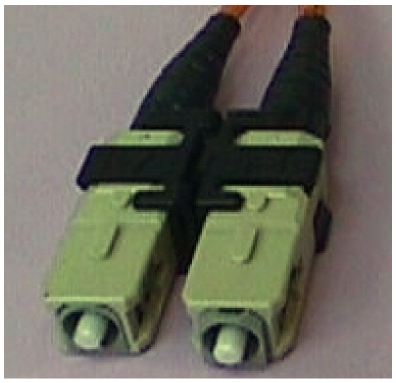
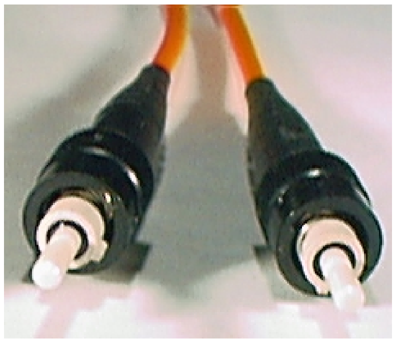
Dual	Simplex
 MIC Connector MIC = Media Interface Connector	 SC (subscriber) Connector
	 ST (straight-tipped) Connector

Figure 15. Multi-mode connectors

5.2 PCI adapter configuration

Adapters are shipped with a wrap plug that must be removed prior to the installation of the adapter. Installation instructions are shipped with the adapter. Generally, you should install the device driver from the CD-ROM after installing the adapter. After the installation of the device driver, the ATM adapters can be configured by either rebooting the system or by using the `cfgmgr` command.

5.2.1 Adapter installation check for PCI adapter

Once the adapter is installed, check if it is detected by AIX. Use:

```
# lsparent -C -k atm -H
```

You should get an output like:

```
name      status       location      description
atm0      Available    00-05         IBM PCI 155 Mbps ATM adapter
```

The third column after `Available` is system and slot dependent. This output is an indicator that the adapter is installed correctly and the microcode has been loaded onto the card. If you get an output like:

```
name      status       location      description
atm0      Defined      00-05         IBM PCI 155 Mbps ATM adapter
```

then either the device driver was not installed on the machine before the adapter was installed, or the adapter needs to be reset (reboot the machine).

5.2.2 Upgrading device driver code for PCI adapter

If there were adapters and device drivers already installed, and you are upgrading the device driver common code, or device driver, then you must shutdown all uses of ATM to prepare to use the `cfgmgr` command to load the new versions. In some cases, it may be easier to shutdown and reboot the system. However, an *ifconfig at(x)* detach on all network interfaces, including any LANE network interfaces that use the adapter, followed with *rmdev -l ent(x)* or *rmdev -l tok(x)* to remove LANE devices if any, followed then by *rmdev -l atm(x)* on all ATM ports, will clear the device driver and all ATM code from the AIX kernel; so, the `cfgmgr` command will reload with the new versions.

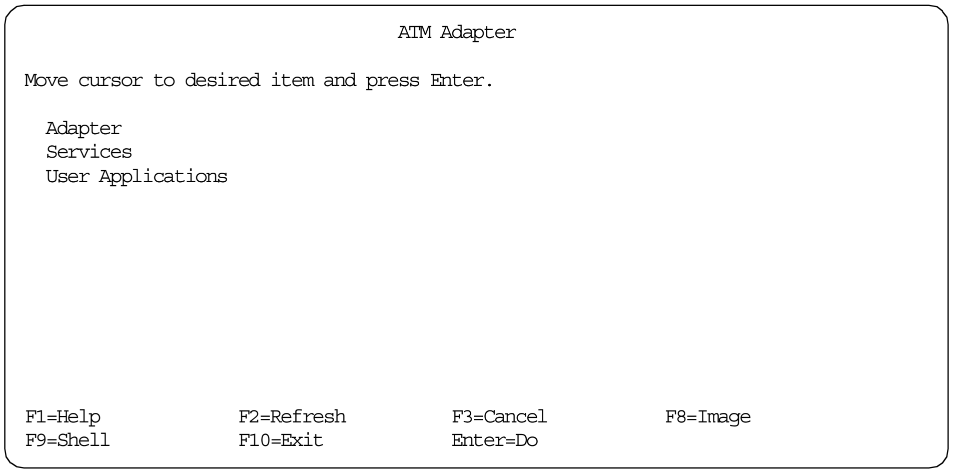
When one LED's is green a signal is detected. This does not mean that the adapter is functional, but it does indicate that the cables are correctly inserted. You may use *atmstat -d atm(x)* to examine the *Driver Flags*: status. It should not be in *Limbo* state. If there is no signal, the cables may be checked for continuity. For fiber cables, a pen light or other light source placed at one end will transmit light to the opposite end and allow a rudimentary check. While this technique does not provide qualitative data, it does help resolve swapped cables, broken cables, crimped fibers, or swapped leads.

5.2.3 Installation steps for PCI adapters

To install a PCI adapter, we issue:

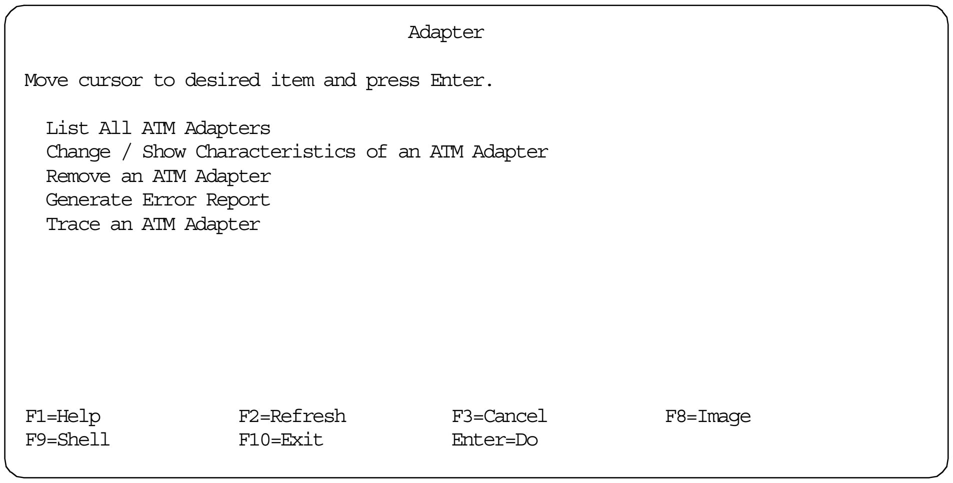
```
# smitty atm
```

Select **Adapter**.



```
                                   ATM Adapter

Move cursor to desired item and press Enter.

  Adapter
  Services
  User Applications









F1=Help             F2=Refresh          F3=Cancel           F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Select **Change / Show Characteristics of an ATM Adapter**.



```
                                    Adapter

Move cursor to desired item and press Enter.

  List All ATM Adapters
  Change / Show Characteristics of an ATM Adapter
  Remove an ATM Adapter
  Generate Error Report
  Trace an ATM Adapter







F1=Help             F2=Refresh          F3=Cancel           F8=Image
F9=Shell            F10=Exit            Enter=Do
```

You will get a screen with a list of installed adapters. Select the one you want to configure. In our example, we selected **atm0**.

```
                                    Adapter

Move cursor to desired item and press Enter.

  List All ATM Adapters
  Change / Show Characteristics of an ATM Adapter
  Remove an ATM Adapter
  Generate Error Report
  Trace an ATM Adapter


  +--------------------------------------------------------------------------+
  |                              ATM Adapter                                 |
  |                                                                          |
  | Move cursor to desired item and press Enter.                             |
  |                                                                          |
  |   atm0 Available 04-08 IBM PCI 155 Mbps ATM Adapter (14107c00)           |
  |   atm1 Available 04-02 IBM PCI 155 Mbps ATM Adapter (14107c00)           |
  |                                                                          |
  | F1=Help                 F2=Refresh              F3=Cancel                |
  | F8=Image                F10=Edit                Enter=Do                 |
  | /=Find                  n=Find Next                                      |
  +--------------------------------------------------------------------------+
```

You will get the following screen with changeable settings for the ATM adapter.

```
               Change / Show Characteristics of an ATM Adapter

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                         [Entry Fields]
  Device name                                          atm0
  Description                                          IBM PCI 155 Mbps ATM A>
  Status                                               Available
  Location                                             04-08
  Enable ALTERNATE ATM MAC address                     no                     +
  ALTERNATE ATM MAC address (12 hex digits)           [0x0]
  Software Transmit Queue Size                        [100]                   +#
  Minimum Guaranteed VCs Supported                    [32]                    +#
  Maximum number of VCs Needed                        [1024]                  +#
  SVC UNI Version                                      auto_detect            +
  Minimum 4K pre-mapped receive buffers               [0x60]                  +#
  Sonet or SDH interace                               [0]                     +#
  Provide SONET clock                                 [0]                     +#


F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

The following will describe the parameters in more detail.

Alternate ATM MAC address

This is a 12 octets parameter. The default is 0.

The ATM address is 20 octets, of which the upper 13 octets are supplied by the switch. The adapter has a burned in address for the next six octets. The last octet in the ATM address is the selector byte. It is used by LANE and CIP for selective addressing. The alternate ATM MAC address hex digits will replace the six octets of the burned in address only when the parameter *ENABLE ALTERNATE MAC ADDRESS* (use_alt_addr) is *yes*.

Software transmit queue size

The default for this parameter is 100, and the valid range is between zero and 4,096.

This parameter defines the number of mbufs waiting that the ATM device driver will queue up to send to the adapter hardware queues. The trade-off is between the kernel heap space (management table size and mbufs that are potentially tied up waiting for space on the adapter) against the need to keep the adapter busy by providing a sufficient number of queued mbufs for the current system environment. Since the parameter is only used at power-up or during configuration of the adapter, the parameter should be set to accommodate the worst case system loading scenario. The default of 100 should be sufficient for most applications.

The `atmstat` command provides two variables that can be used to tune this parameter.

The *Max Packets on S/W Transmit Queue* statistic shows what the high-water mark for the Software Transmit Queue was since the last time the statistics were cleared. This can be used to gauge how system overhead effected the ATM adapters operation. If the number remains lower than 75 percent of the maximum value, assuming that peak load periods for all applications have been encountered, then there is no need to adjust the value upwards. If the number remains less than 25 percent of the maximum value, the number should be decreased by 50 percent until the default of 100 is reached. If kernel heap space is very limited, there may be justification to reduce this value below 100, but it is not recommended.

The *S/W Transmit Queue Overflow* should be used to determine if dramatic increases for Software Transmit Queue size are needed. You should attempt to tune the Software Transmit Queue size by 10 percent to 20 percent increases if overflows occur, until either the overflow occurrences cease or the limit has been reached. Typically, overflows mean that there is significant loading of the system and resources of the system where the ATM software experiences longer delay times in processing mbufs for transmission.

Minimum guaranteed VCs supported

The default for this parameter is 32, and the valid range is between five and 1,024.

This is the number of table entries reserved at configuration time by the device driver for virtual circuits. It only specifies that memory for managing VC's be reserved. If additional virtual circuits are used in excess of this number, this requires a kernel call to allocate space per Virtual Circuit request and to free the space when the Virtual Circuit is closed. This parameter is used to minimize usage of dedicated kernel heap space. The exposure for setting it low is the chance that occasional constraints in the availability of kernel heap space may prevent virtual circuits from being allocated on the first attempt. This may become more of a problem when applications and system functions load the system heavily and kernel heap memory is at a premium. In these situations, it will become increasingly difficult to add virtual circuits beyond the minimum number specified here. Tuning this parameter to handle worst case scenarios for your system can only be based on anecdotal information about the ability of remote systems to reliably access the system via LANE or CIP during peak periods. For most systems, the default of 32 is sufficient. If there are problems accessing this system via this ATM adapter, then calculations should be done to determine how many virtual circuits are needed and then slowly adjusting this parameter closer to this value. This calculation is discussed in the next section. Unless you are experiencing severe shortages of kernel heap memory, this parameter should be set to less than 100 percent of the maximum virtual circuits needed.

Maximum number of VCs needed

The default for this parameter is 1,024, and the valid range is between five and 2048.

This parameter is used to determine how to optimize the adapters internal tables to support Virtual Circuit operation. This must be configured when the adapter is first initialized. At minimum, it should be configured to be the same, or greater than, the minimum number of VCs.

Since this is configurable only when the adapter and device driver are initialized, worst case situations must be considered. For SVC operation, ATM needs three virtual circuits for ILMI, SSCOP, and ARP server, plus one for every active Classical IP partner, three virtual circuits for every

LANE ethernet or Token Ring network, and one for every LANE network partner.

In addition, the LANE Ethernet or Token Ring configuration must have the *arp_cache_size* parameter configured to the maximum number of simultaneous connections, plus three for LANE service connections. It is defaulted to 32, which is insufficient for large networks. Leaving this at a low value causes AIX software to flush entries as attempts to communicate with more systems than arp_cache entries. The new entry must be obtained from the LES. This results in thrashing.

If there are insufficient virtual circuits, the system ATM network layers will clear existing calls whenever they need to send data, and the ARP table does not contain the IP/ATM mapping for the destination. It will establish a new connection and then update the ARP table, thus purging the old entry. If incoming call requests are received, and there are no available virtual circuits, the incoming call will be rejected. If you have trouble accessing a system from other specific systems in a large ATM network intermittently, then there is a good chance that there aren't enough virtual circuits available. The `atmstat` command will return:

- *Virtual Connections in use*
- *Maximum Virtual connections in use*
- *Virtual Connections Overflow*

Virtual Connections Overflow is a counter of the times more virtual circuits were needed and is not a valid count of the absolute maximum number of virtual circuits needed. If this is non-zero, use your calculations to revise the maximum number of VCs needed parameter. If this is insufficient, make incremental adjustments to the Maximum Number of VCs needed parameter until the problem disappears or the maximum value is reached.

Maximum Virtual Connections in use (output from `atmstat`) reports the high-water mark up to the configured maximum value. If you are experiencing intermittent problems accessing this ATM or LANE interface from the network, and `atmstat` shows that the system is not using the maximum number of VCs, then the problem is most likely system congestion and contention for kernel heap to allocate virtual circuits. In this situation, dedicated kernel heap memory is needed to allow more virtual circuits to be configured without contending for memory. Adjust the Minimum Number of VCs needed parameter closer to the reported maximum virtual connections in use value.

When LANE and CIP are used simultaneously to the same remote systems, this potentially doubles the number of virtual circuits that may be needed at a maximum. Because of the larger MTU sizes, CIP connections are more efficient and should be used in preference to LANE connections where possible. The LANE device *arp_cache_size* parameter can be tuned to reduce the number of simultaneous TCP/IP LANE partners and limit the need for VC's. This can be done if `atmstat` still reports *Virtual Connection Overflow* as a non-zero value.

SVC UNI version

The default is auto_detect, and the possible values are auto_detect, uni-3.0, and uni-3.1.

This parameter sets the ATM Forum compatibility level for the ATM User-Network Interface. Auto-detect is generally sufficient, and interoperation with most ATM switches works well with auto-detect. However, there are several switches that do not support our auto_detect efforts and need the Uni-3.1 setting. Problems of this nature usually appear when either the system or the switch is rebooted, and connections cannot be established. Use `atmvcstat` to verify that the first three virtual circuits listed are connected in these situations.

Minimum 4K pre-mapped receive buffers

The default for this parameter is 0x60 (decimal 96), and the valid range is between 0x60 and 0x200.

This parameter is used to control how many mbufs and memory are dedicated to this adapter on a continual basis. The default is 96. These buffers are used to DMA data from the adapter and send it as mbufs through the system's protocol stacks. The number of pre-mapped receive buffers grow and shrink with demand. This is a critical issue in many customer installations.

Setting the Minimum Number of 4 k Pre-Mapped Receive Buffers to a higher value will decrease the chances of running out of buffers when the application has high bursts of small packets. Since the device driver only allows the commitment of 300 receive buffers, this only reduces the symptoms for the first 300 buffers needed. It also helps to reduce the number of *Out of Rcv Buffers* errors, but not necessarily eliminate them for heavily loaded systems.

The number of allocated buffers is reported by `atmstat` as:

```
Total 4k byte Receive Buffers xxx Using: 41 Maximum zzz
```

where `xxx` is the currently allocated 4 k receive buffers for this adapter. It should never drop below the minimum 4 K pre-mapped receive buffers parameter value.

The maximum, `zzz`, is the maximum number of 4 k receive buffers that this ATM adapter is configured to acquire. If the *rx_que_size* parameter for the ATM adapter is left at the default, zero, this number will be calculated to be roughly one buffer for every 107 KB of the network parameter *thewall*. This will be rounded up to 800 buffers if the calculation falls below 800. See 5.2.4, “Changing rx_que_size” on page 48 for details of this ATM parameter. This will override the automatic calculation if not set to zero. For AIX 4.1, the maximum is fixed at 800 buffers. For the latest versions of AIX 4.2 and AIX 4.3 ATM software, the maximum number of buffers has been increased.

Applications must clear the receive queue on the socket in a timely manner. ATM has a limited number of available receive buffers. We have reduced the exposure to a poorly performing application, but eventually the supply of buffers will be exhausted. This can occur for four reasons or a combination thereof: Limited physical memory, too much demand for kernel heap memory (processes, service, or other device drivers), too much loading on the processor(s) CPU time and interrupt processing, or simply an application with long receive path lengths or poor responsiveness to received packets. System performance, hardware configuration, and software/application configuration are independent variables that interact. The interaction and tuning of these variables is beyond the scope of this redbook. To examine buffer usage on receive queues, use the `netstat -a` command periodically to see the ebb and flow of receive buffers. To examine memory, and I/O activity, use `vmstat` or the SP monitor tool.

SONET or SDH interface

This ATM adapter parameter is used to control the mode of operation of the PHY on the adapter. The default is to use the Sonet setting. Networks can be either SONET or SDH compatible. Change this only when attaching to a network using SDH.

Provide SONET clock

Most, but not all, switches provide the SONET clock to use for transmission and reception. This parameter is generally set to zero unless using direct connection PVCs, where one system must provide the clock for both, or connecting to a switch that does not provide a clock.

5.2.4 Changing rx_que_size

The default of this parameter is zero for calculations based on the *thewall* network option, and the valid range is between one and 140,000.

This is an adapter parameter used to control the maximum number of pre-mapped 4 k receive buffers. It is only changed using the `chdev` command.

If a `chdev -arx_que_size=yyy -l atm(x)` command, where `yyy` is between one and 14, 000, has been completed, the next reboot or re-initialization of the ATM device driver will use this value to configure the maximum number of receive buffers. Note that changing this parameter does not take place immediately. The *rx_que_size* is displayed by using the `lsattr -El atm(x)` command. The network option, *thewall*, must be large enough to fill the mbuf memory requests. 14,000 pre-mapped receive buffers will require at least 57 M Bytes of TCP/IP buffer space. For ADSM, plan on setting the rx_queue_size to 14,000. See section 3.8.1, “ADSM configuration” on page 29 for details.

For heavy ATM traffic, there have been systems where the kernel heap memory has been so overcommitted that ATM cannot get receive buffers. We have seen systems where the network options parameter, *thewall* , has been set too low, and there are no network buffers available to use as receive buffers. We have also seen systems that could not expand the network buffer pool to get receive buffers in a timely basis for some applications that have small packets and bursts of traffic. Additionally, some applications do not process the socket receive queue responsibly and starve the ATM device driver by holding its receive buffers. The `atmstat` command generally reports these as *Out of Rcv Buffers*.

These are four different problems associated with *Out of Rcv Buffers* errors from `atmstat`.

For heavily loaded systems, setting *rx_que_size* to a larger size via the `chdev` command will allow the ATM interface to configure more space to track and use more receive buffers. Changing the network option, *thewall* , to a larger value may also be needed to assure that there is significant space to allocate buffers in the network memory pool. The space being used comes from the kernel heap. There may also be insufficient space to satisfy all uses due to application and kernel service usage or simply insufficient real memory. If the `atmstat` report line:

```
   Total 4k byte Receive Buffers xxx Using: 41 Maximum zzz
```

shows that `xxx` is the same as `zzz`, then expanding the *rx_queue_size* is a reasonable next step. If `xxx` is less than `zzz`, this is inconclusive, as `xxx` is the current number of buffers in use and not a high-water mark. This value will contract as the buffers are not needed and are freed to the system by the ATM device driver.

New `atmstat` output lines are being released to display the high-water mark for 4 k receive buffers that were used since clearing the status last and the maximum that the device driver can currently configure:

Maximum 4k byte receive buffers used `yyy`

Maximum configurable 4k byte receive buffers `zzz`

where `yyy` is the high-water mark, and `zzz` is the maximum that the ATM device driver is configured to allocate.

If the two are equal, then some application(s) have not been reading data off of the receive sockets in a timely manner. For ATM, this is unacceptable in that the connection to the ATM switch and to the LANE LES entities require some free mbufs. HACMP heartbeats require free mbufs. Some applications, such as ADSM, use TCP/IP to pace the flow of work between client and server and can accumulate a window or more data on the receive socket. While this is not a recommended practice, it does happen. In cases like this, many applications allow options to limit the number of sessions.

If the two are not equal, but the out of receive buffers count is constantly increasing, then the problem is either network memory (controlled by the *thewall* network option parameter), kernel heap memory, real physical memory, high CPU utilization, or an application/service holding buffers.

These can be eliminated one by one. Look for applications/services holding buffers by using `netstat -an`.

Use `netstat -m` to determine if there are numerous 4 KB mbuf failures. Use `vmstat` or the AIX performance toolbox (PTX) tools to determine available memory. Again, monitoring results over long periods to determine worst case usage may require periodic capturing of information along with timestamp and process information.

Generally, a few instances of having a non-zero, Out of Rcv Buffers is not catastrophic. However, if these situations persist solidly past eight seconds, the ATM connection to the switch will be reset along with all of the virtual circuits being used. Reconnection is usually rapid, but it is disruptive to the performance. In some instances, buffers being held by applications will be returned too slowly, and interference with reconnection to the switch may occur.

5.3 Micro channel adapter configuration

The base steps for the MCA adapter configuration are the same as for PCI. As already mentioned, you should install the device driver from the CD-ROM after installing the adapter. After the installation of the device driver, the ATM adapters can be configured by either rebooting the system or by using the `cfgmgr` command.

5.3.1 Adapter installation check for MCA adapter

Once the adapter is installed, check if it is detected by AIX. Use:

```
# lsparent -C -k atm -H
```

You should get an output like:

```
name      status        location      description
atm0      Available     00-05         155 Mbps ATM Fiber Adapter
```

The third column after `Available` is system and slot dependent. This output is an indicator that the adapter is installed correctly, and the microcode has been loaded onto the card. If you get an output like:

```
name      status        location      description
atm0      Defined       00-05         155 Mbps ATM Fiber Adapter
```

then either the microcode has not been installed on the machine before the adapter was installed, or the adapter needs to be reset (reboot the machine).

5.3.2 Upgrading device driver code for MCA adapter

If there were adapters and device drivers already installed, and you are upgrading the device driver common code or device driver, then you must shutdown all uses of ATM to prepare for the `cfgmgr` command to load the new versions.

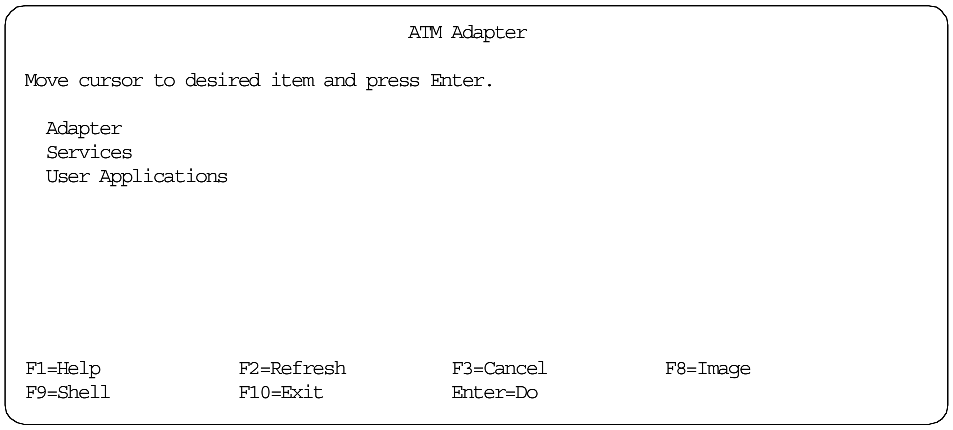
In some cases, it may be easier to shutdown and reboot the system. However, an *ifconfig at(x) detach* on all network interfaces, including any LANE network interfaces that use the adapter, followed with *rmdev -l ent(x)* or *rmdev -l tok(x)* to remove LANE devices, if any, followed then by *rmdev -l atm(x)* on all ATM ports, will clear the device driver and all AIX ATM code from the kernel; so, `cfgmgr` will reload with the new versions.

5.3.3 Installation steps for MCA adapters

The first step is to issue the following command:

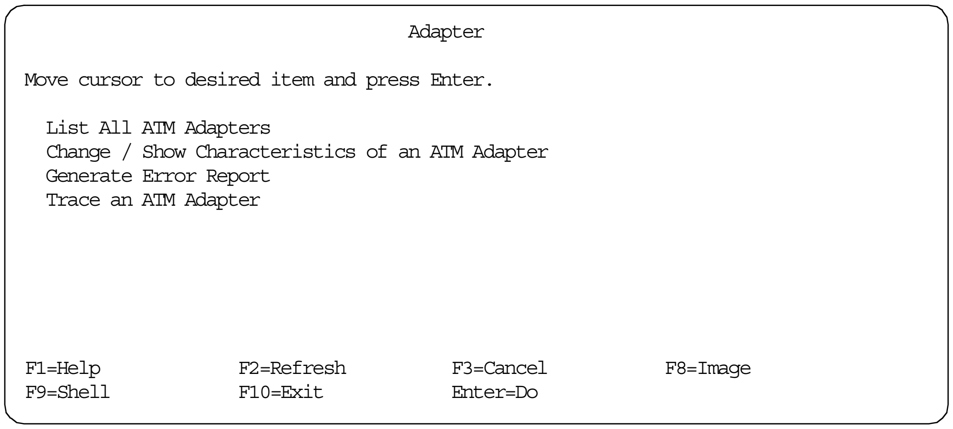
```
# smitty atm
```

Select **Adapter**.



```
                                    ATM Adapter

Move cursor to desired item and press Enter.

  Adapter
  Services
  User Applications









F1=Help             F2=Refresh          F3=Cancel           F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Select **Change / Show Characteristics of an ATM Adapter**.



```
                                    Adapter

Move cursor to desired item and press Enter.

  List All ATM Adapters
  Change / Show Characteristics of an ATM Adapter
  Generate Error Report
  Trace an ATM Adapter








F1=Help             F2=Refresh          F3=Cancel           F8=Image
F9=Shell            F10=Exit            Enter=Do
```

You will get a screen with a list of installed adapters. Select the one you want to configure. In our example, we selected **atm0**.

```
                                    Adapter

Move cursor to desired item and press Enter.

  List All ATM Adapters
  Change / Show Characteristics of an ATM Adapter
  Generate Error Report
  Trace an ATM Adapter


  +--------------------------------------------------------------------------+
  |                   100 Mbps ATM Fiber Adapter (8f7f)                      |
  |                                                                          |
  | Move cursor to desired item and press Enter.                             |
  |                                                                          |
  |   atm0 Available 00-03 155 Mbps ATM Fiber Adapter (8f67)                 |
  |   atm1 Available 00-04 155 Mbps ATM Fiber Adapter (8f67)                 |
  |                                                                          |
  | F1=Help                 F2=Refresh              F3=Cancel                |
  | F8=Image                F10=Exit                Enter=Do                 |
  | /=Find                  n=Find Next                                      |
  +--------------------------------------------------------------------------+
```

You will get the following screen with changeable settings for the ATM adapter.

```
               Change / Show Characteristics of an ATM Adapter

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[TOP]                                                        [Entry Fields]
  ATM Adapter                                              atm0
  Description                                              155 Mbps ATM Fiber Ada>
  Status                                                   Available
  Location                                                 00-03
  User Best Effort Peak Rate (kbits/sec)                  [1500]                  +#
  Enable ALTERNATE ATM MAC address                         no                     +
    ALTERNATE ATM MAC address (12 hex digits)             [0x]
  ATM Adapter Maximum PDU size (bytes)                    [9188]                  +#
  DMA bus memory width                                    [0x1000000]             +X
  Maximum Small ATM mbufs                                 [50]                    +#
  Maximum Medium ATM mbufs                                [100]                   +#
  Maximum Large ATM mbufs                                 [300]                   +#
  Maximum Huge ATM mbufs                                  [50]                    +#
  Max Transmit Block (MTB) size (kbytes)                  [64]                    +#
  Minimum Small ATM mbufs                                 [20]                    +#
  Minimum Medium ATM mbufs                                [30]                    +#
  Minimum Large ATM mbufs                                 [70]                    +#
  Minimum Huge ATM mbufs                                  [5]                     +#
  Minimum ATM MTB mbufs                                   [4]                     +#
  Minimum ATM interface buffer size                       [2048]                  +#
  Minimum Guaranteed VCs Supported                        [8]                     +#
  Maximum Number of VCs Needed                            [32]                    +#
  SVC UNI Version                                          auto_detect            +
  Software transmit queue length                          [512]                   +#
  Sonet or SDH interface                                  [0]                     +#
  Provide SONET clock                                     [0]                     +#
[BOTTOM]

F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

The following will describe the parameters in more detail:

User best effort peak rate (kbits/sec)

The default for this parameter is 1,500, and the valid range is between zero and 155,000.

This value is used to configure the adapter. A MCA ATM adapter has four queue speeds. One is programmable. This value is used to calculate the last queue speed. It is defaulted to 1,500 kbits/sec and should remain at this setting. Classical IP and LANE in AIX release 4.3 use the adapters maximum bandwidth. For AIX 4.2.5 and earlier releases of AIX, the Classical IP parameter, *Best Effort Bit Rate* in kbits/sec, must be set to 155,000 to get full use of the adapter's bandwidth. For AIX 4.2.5 and earlier releases of AIX, the LANE device parameter, *Forward/Backward Peak Rate* in (Kbits/sec), must be configured to 155,000 instead of the default 25,600 if full speed operation is needed.

The naming of this parameter is misleading. It should be eliminated from the configuration menu. Please do not change the value of this parameter. Changing this parameter can occasionally lead to ATM UCODE error log messages with sense codes ending in 0003 0202, which occur when incoming calls arrive with a low peak rate value, and the ATM microcode cannot find a low speed queue that is appropriate.

Enable ALTERNATE ATM MAC address

The default for this parameter is *no,* and the two possible values are *yes* and *no*.

This parameter is used to select between the burned in card address (six octets) and the six octet address defined in the *ALTERNATE ATM MAC* address. The default is *no*. This parameter will not affect an operation adapter. The adapter must be removed from operation, the value changed, and then restarted to use the changed value of this parameter.

ALTERNATE ATM MAC address (12 octets)

This is a 12 octets parameter. The default is 0.

The ATM MAC address consists of 20 octets. The first 13 octets are defined by the switch that the adapter is directly connected to. The next six octets are normally taken from the burned in MAC address for the adapter. When the *Enable ALTERNATE ATM MAC* address parameter is yes during the configuration of the ATM adapter, then the *ALTERNATE ATM MAC* address is used to form the ATM MAC address. This happens only at the configuration of the adapter. The remaining octet is the selector field. It is assigned to each active CIP or LANE port on the adapter.

ATM adapter maximum PDU size

The default for this parameter is 9,188 (bytes), and the valid range is between 4,096 and 60,424.

This parameter establishes the maximum PDU size for the adapter during configuration. This value must be at least eight bytes greater than the user's MTU size. For most applications, 9,188 is larger than the traffic's large buffers. However, for some applications, such as ADSM, setting this value to a larger number can reduce overhead in handling packets. For ADSM over CIP, 32 K byte data packets improve performance. If so, the recommended *ATM Adapter Maximum PDU size* is 32,846 to accommodate the headers and data.

DMA bus memory width

The default for this parameter is 0x1000000, and the valid range is between 0x800000 and 0x40000000.

This parameter is used to configure the adapters and reserve bus TCE mapping space. The ATM configuration routine does a calculation of the sum of the maximum small (256 bytes rounded up to 4 k bytes), Medium (4 k bytes), large (8 k bytes), huge (16 k bytes), and special buffers (MTB size is programmable from 32 k-1,024 k). If this sum plus eight megabytes is less than the *DMA bus memory width* parameter, the adapter is allowed to configure. If the adapter will not configure, with an error in the error log (use `errpt -a`), *ATM NOBUFS PERMANENT ERROR*, either reduce the number of buffers or increase the DMA bus memory width.

Maximum small ATM mbufs

The default for this parameter is 50, and the valid range is between 40 and 400.

Small ATM mbufs are 256 bytes. This parameter establishes the maximum number of 256 byte buffers that can be used by this ATM adapter's device driver. These buffers are not pre-allocated during configuration. There is no guarantee that mbufs will be available for use when the device driver needs more than already allocated. This parameter will take effect when the adapter is reconfigured and not during operation. To change this, remove all network and LANE interfaces using the adapter and unconfigure the adapter by using `rmdev -l atm(x)`.

Then change the parameter or parameters using SMIT. The `atmstat` output will show four parameters that can be used in tuning:

Packets Dropped - No small DMA buffers:

Small mbufs were not available for receiving incoming traffic.

Small Mbuf in Use:

Current count of small mbufs being used by device driver/applications.

Max Small Mbuf in Use:

The high water mark for use with small mbufs.

Small Mbuf overflow:

Times the small mbufs requested exceeded the maximum.

Max Small Mbuf Overflow should not be greater than zero. Periodically examine this statistic and increase the *Maximum Small ATM mbufs* to prevent the peak from being reached. Overflows for buffers of this size can result in lost receive packets, poor adapter performance, or dropped transmit packets. There is no retry for allocation of the mbuf of this size. If the atmstat *Packets Dropped - No Small DMA buffers* statistic is non-zero, and *Max Small Mbuf Overflow* is also non-zero, there may be a problem with the rate of return of buffers to the pool. Tuning this parameter to a higher value should reduce the occurrence of these problems.

The atmstat *Max Small Mbuf in Use* should not be 50 percent or less than the configured maximum for small ATM mbufs. This is not catastrophic. The device driver uses kernel heap storage to manage the device, and space could be freed up if it is not needed. This parameter should be reduced.

Packets Dropped - No Small DMA buffers indicates that request for mbufs failed due to resource constraints. If the *Small Mbuf Overflow* value is non-zero, the network option, *thewall,* may have been too small for the number of buffers needed for TCP/IP, or there may have been insufficient kernel heap memory available to allocate to communications mbufs. This can happen infrequently as part of normal operation because there are peak load conditions in the system. If this happens for long periods, it can be a serious problem and result in lost connections, loss of communication with the adapter, poor ATM performance, or application failure. The severity of the problem is determined by multiple, independent variables related to the system capacity and workload. Buffers are freed by applications for reuse at too slow of a rate. Focus on the following in the order:

- Network options:

  thewall, tcp_sendspace, tcp_recvspace, udp_sendspace, udp_recvspace

- System resources:

  Physical memory, CPU performance, processes

- Applications

  Sockets, custom parameters, receive buffer processing, instances.

Maximum Medium ATM mbufs

The default for this parameter is 400, and the valid range is between 40 and 800.

Medium ATM mbufs are 4,096 bytes. This parameter establishes the maximum number of 4,096 byte buffers that can be used by this ATM adapter's device driver. These buffers are not pre-allocated during configuration. There is no guarantee that mbufs this large will be available for use when the device driver needs more than already allocated. This parameter will take effect when the adapter is reconfigured and not during operation. To change this, remove all network and LANE interfaces using the adapter, and unconfigure the adapter by using `rmdev -l atm(x)`.

Then change the parameter or parameters using SMIT. The atmstat output will show four parameters that can be used in tuning:

Packets Dropped - No Medium DMA buffers:

Medium mbufs were not available for receiving incoming traffic.

Medium Mbuf in Use:

Count of medium mbufs being used by the device driver/applications.

Max Medium Mbuf in Use:

The high water mark for use with medium mbufs.

Medium Mbuf overflow:

Times medium mbufs requested exceeded the maximum.

Max Medium Mbuf Overflow should not be greater than zero. Periodically examine this statistics and increase the maximum medium ATM mbufs to prevent the peak from being reached. It is not catastrophic to have an overflow, as the device driver will attempt to get the next smaller size of buffer. It is inefficient. If the *Packets Dropped - No Medium DMA buffers* value is non-zero, and *Max Medium Mbuf Overflow* is also non-zero, there may be a problem with the rate of return of buffers to the pool. Tuning this parameter to a higher value should reduce the occurrence of these problems.

Max Medium Mbuf in Use should not be 50percent or less than the configured maximum for small ATM mbufs. This is not catastrophic. The device driver uses kernel heap storage to manage the device, and space could be freed up if it is not needed.

A non-zero value for *Packets Dropped - No Medium DMA buffers* indicates that request for mbufs failed due to resource constraints. If the medium mbuf overflow value is non-zero, the network option, *thewall,* may have been too small for the number of buffers needed for TCP/IP, or there may have been insufficient kernel heap memory available to allocate to communication mbufs. This can happen infrequently as part of normal operation because there are peak load conditions in the system. If this happens for long periods, it can be a serious problem and result in lost connections, loss of communication with the adapter, poor ATM performance, or application failure. The severity of the problem is determined by multiple independent variables related to the system capacity and workload. Buffers are freed by applications for reuse at too slow of a rate. Focus on the following order:

- Network options

  thewall, tcp_sendsapce, tcp_recvspace, udp_sendspace, udp_recvspace

- System resources

  Physical memory, CPU performance, processes

- Applications

  Sockets, custom parameters, receive buffer processing, instances

Maximum large ATM mbufs

The default for this parameter is 400, and the valid range is between 75 and 1,600.

Large ATM mbufs are 8,192 bytes and are the most frequently used. For AIX 4.3.2 and later releases, the device driver will allow configuration of 1,600 mbufs. Earlier releases of the device driver will only allow configuration of 800 as the maximum number of large ATM mbufs. This parameter establishes the maximum number of 8,192 byte buffers that can be used by this ATM adapter's device driver. These buffers are not pre-allocated during configuration. There is no guarantee that mbufs this large will be available for use when the device driver needs more than already allocated. This parameter will take effect when the adapter is reconfigured and not during operation. To change this, remove all network and LANE interfaces using the adapter and unconfigure the adapter by using `rmdev -l atm(x)`. Then, change the parameter or parameters using SMIT. The atmstat output will show four parameters that can be used in tuning:

Packets Dropped - No Large DMA buffers:

Large mbufs were not available for receiving incoming traffic.

Large Mbuf in Use:

Current count of large mbufs being used by device drive/applications.

Max Large Mbuf in Use:

The high water mark for uses of large mbufs.

Large Mbuf overflow:

Times large mbufs requested exceeded the maximum.

Max Large Mbuf overflow should not be greater than 0. Periodically examine this statistics and increase the *Maximum Large ATM mbufs* to prevent the peak from being reached. It is not catastrophic to have an overflow as the device driver will attempt to get the next smaller size of buffer. It is inefficient. If the *Packets Dropped - No Large DMA buffers* value is non zero and *Max Large Mbuf Overflow* is also non-zero, there may be a problem with the rate of return of buffers to the pool. Tuning this parameter to a higher value should reduce the occurrence of these problems.

Max Large Mbuf in Use should not be 50% or less than the configured maximum for Large ATM mbufs. This is not catastrophic. The device driver uses kernel heap storage to manage the device and space could be freed up if it is not needed. This parameter should be reduced.

A non-zero value for *Packets Dropped - No Large DMA buffers* indicates that request for mbufs failed due to resource constraints. If the Large Mbuf Overflow value is non-zero, the network option *thewall* may have been too small for the number of buffers needed for TCP/IP or there may have been insufficient kernel heap memory available to allocate to communications mbufs. This can happen infrequently as part of normal operation because there is peak load conditions in the system. If this happens for long periods, it can be a serious problem and result in lost connections, loss of communication with the adapter, poor ATM performance, or application failure. The severity of the problem is determined by multiple independent variables related to the system capacity and workload. Buffers are freed by applications for reuse at too slow of a rate. Focus on the following, in order:

- network options

  thewall, tcp_sendsapce, tcp_recvspace, udp_sendspace, udp_recvspace

- system resources

  physical memory, CPU performance, processes

- applications

  sockets, custom parameters, receive buffer processing, instances.

Maximum Huge ATM mbufs

The default for this parameter is 50, and the valid range is between 0 and 800.

Huge ATM mbufs are 16384 bytes. This parameter establishes the maximum number of 16384 byte buffers that can be used by this ATM adapter's device driver. These buffers are not pre-allocated during configuration. There is no guarantee that mbufs this large will be available for use when the device driver needs more than already allocated. This parameter will take effect when the adapter is reconfigured and not during operation. To change this, remove all Network and LANE interfaces using the adapter, and unconfigure the adapter by using *rmdev -l atm(x)*. Then change the parameter or parameters using smit. The atmstat output will show four parameters that can be used in tuning:

Packets Dropped - No Huge DMA buffers:

Huge mbufs were not available for receiving incoming traffic

Huge Mbuf in Use:

Current Count of Huge mbufs being used by Device Drive/applications

Max Large Mbuf in Use:

The high water mark for uses of Huge mbufs.

Huge Mbuf overflow:

Times Huge buffers requested exceeded the maximum.

Max Huge Mbuf overflow should not be greater than 0. Periodically examine this statistics and increase the Maximum Huge ATM mbufs to prevent the peak from being reached. It is not catastrophic to have an overflow as the device driver will attempt to get the next smaller size of buffer.

Max Huge Mbuf in Use should not be 50% or less than the configured maximum for Huge ATM mbufs. This is not catastrophic. The device driver uses kernel heap storage to manage the device and space could be freed up if it is not needed. This parameter should be reduced.

Maximum ATM MTB mbufs

The default for this parameter is 4, and the valid range is between 0 and 400.

These are the largest ATM mbufs. They are variable size (32k -1024k bytes) depending upon the setting for *Max Transmit Block size*. The default is 64K bytes. This size is generally used with large MTU and PDU sizes configured. This parameter should be tuned if applications that use device special buffers and large block sizes are to be executed frequently or large MTU sizes greater than 16k bytes are configured. If the *MTB Mbuf overflow* count is non-zero, then increase the value to eliminate the problem. If the MTB Mbuf in Use count is low relative to the maximum, reduce the maximum value.

Max Transmit Block (MTB) size

The default for this parameter is 64 KB, and the valid range is between 32 and 1024.

This sets the size of the Maximum Transmit Buffers. The size must be set greater than 16K bytes.

Minimum Small ATM mbufs

The default for this parameter is 20, and the valid range is between 10 and 200.

This is the number of Small (256 byte) mbufs (sml_highwater) that are allocated when the ATM Device Driver is configured. These are private buffers for ATM and not available for other devices or kernel processes. The ATM device driver will not free these mbufs until the device driver is terminated.

This parameter must be less than or equal to the *Maximum Small ATM mbufs* (max_sml_bufs). It can be set to as low as 10. It is best to tune this value by periodically examining the *atmstat* statistics: *Max Small Mbuf in Use*, and *Packets Dropped - No small DMA buffer*.

If packets are being dropped, and the *Max Small Mbuf in Use* is less than max_sml_bufs, then increase the number of small mbufs (sml_highwater) initially configured. The device driver for this adapter must be removed and reconfigured before changes take effect. All network interfaces, and LANE devices using the adapter must be shutdown prior to reconfiguring the adapter. Alternatively, rebooting the system, will reconfigure the device. Pre-allocating the buffers will avoid some situations where there is insufficient kernel heap memory or insufficient memory reserved for system mbufs during peak workload or network traffic.

Minimum Medium ATM mbufs

The default for this parameter is 30, and the valid range is between 20 and 300.

This is the number of medium (4096 byte) mbufs (med_highwater) that are allocated when the ATM Device Driver is configured. These are private buffers for ATM and not available for other devices or kernel processes. The ATM device driver will not free these mbufs until the device driver is terminated.

This parameter must be less than or equal to the *Maximum Medium ATM mbufs* (max_med_bufs). It can be set to as low as 20. It is best to tune this value by periodically examining the *atmstat* statistics: *Max Medium Mbufs in Use*, and *Packets Dropped - No medium DMA buffer*.

If packets are being dropped, and the *Max Medium Mbuf in Use* is less than max_med_bufs, increase the number of Medium mbufs (med_highwater) initially configured. The device driver for this adapter must be removed and reconfigured before changes take effect. All Network Interfaces, and LANE devices using the adapter must be shutdown prior to reconfiguring the adapter. Alternatively, rebooting the system will reconfigure the device. Pre-allocating the buffers will avoid some situations where there is insufficient kernel heap storage or insufficient memory reserved for system mbufs during peak workload or network traffic.

Minimum Large ATM mbufs

The default for this parameter is 70, and the valid range is between 65 and 800.

This is the number of large (8192 byte) mbufs (lrg_highwater) that are allocated when the ATM Device Driver is configured. These are private buffers for ATM and not available for other devices or kernel processes. The ATM device driver will not free these mbufs until the device driver is terminated. This is the most frequently used buffer size and usually the first to reach maximum usage and overflow.

This parameter must be less than or equal to the *Maximum Large ATM mbufs* (max_lrg_bufs). It can be set to as low as 65. It is best to tune this value by periodically examining the *atmstat* statistics: *Max Large Mbufs in Use*, and *Packets Dropped - No large DMA buffer*.

If packets are being dropped, and the *Max Large Mbuf in Use* is less than *max_lrg_bufs*, increase the number of Large mbufs (lrg_highwater) initially configured. The device driver for this adapter must be removed and reconfigured before changes take effect. All Network Interfaces, and LANE devices using the adapter must be shutdown prior to reconfiguring the adapter. Alternatively, rebooting the system will reconfigure the device. Pre-allocating the buffers will avoid some situations where there is insufficient kernel heap storage or insufficient memory reserved for system mbufs during peak workload or network traffic.

Minimum Huge ATM mbufs

The default for this parameter is 5, and the valid range is between 4 and 30.

This is the number of huge (16384 byte) mbufs (hug_highwater) that are allocated when the ATM Device Driver is configured. These are private buffers for ATM and not available for other devices or kernel processes. The ATM device driver will not free these mbufs until the device driver is terminated.

This parameter must be less than or equal to the *Maximum Huge ATM mbufs* (max_hug_bufs). It can be set to as low as 4. This parameter has very little feedback for tuning. This should be tuned to allow the best performance for any application using special socket buffers and huge MTU sizes.

Minimum ATM MTB mbufs

The default for this parameter is 4, and the valid range is between 0 and 300.

This is the number of MTB (32K-1024 byte) mbufs (spec_highwater) that are allocated when the ATM Device Driver is configured. These are private buffers for ATM and not available for other devices or kernel processes. The ATM device driver will not free these mbufs until the device driver is terminated.

This parameter must be less than or equal to the *Maximum MTB ATM mbufs* (max_spec_bufs). It can be set to as low as 4. This parameter has very little feedback for tuning. This should be tuned to allow the best performance for any application using special socket buffers and MTU sizes greater than 16k bytes.

Minimum ATM interface buffer size

The default for this parameter is 2048, and the valid range is between 2048 and 65536.

This ATM adapter supports the use of interface buffers., the default is 2048. The socket layer will attempt to obtain an interface buffer and copy data directly into a buffer that has already been prepared for DMA by the device driver. Interface buffers are not used for transmits with lengths less than this value.

Minimum Guaranteed VCs Supported

The default for this parameter is 8, and the valid range is between 8 and 1023.

This is the number of virtual circuits that is guaranteed to be supported, the default is 8. It appears as min_vc in the output from *lsattr -El atm(x)*. This parameter reserves Virtual Circuit table entries at configuration time by the device driver. It only specifies that memory for managing VC's be reserved. If additional virtual circuits are used in excess of this number, this requires a kernel call to allocate space per Virtual Circuit request and to free the space when the Virtual Circuit is closed. This parameter is used to minimize usage of dedicated kernel heap space. The exposure for setting it low is the chance that occasional constraints in the availability of kernel heap space may prevent virtual circuits from being allocated on the first attempt. This may become more of a problem when applications and system functions load the system heavily and kernel heap memory is at a premium. In these situations, it will become increasingly difficult to add virtual circuits beyond the minimum number specified here. Tuning this parameter to handle worst case scenarios for your system can only be based on anecdotal information about the ability of remote systems to reliably access the system via LANE or CIP during peak periods. For most systems, the default of 8 is sufficient. If there are problems accessing this system via this ATM adapter, then calculations should be done to determine how many virtual circuits are needed and slowly adjusting this parameter closer to this value. This calculation is discussed in the next Section. Unless you are experiencing severe shortages of kernel heap memory, This parameter should be set to less than 100% of the maximum virtual circuits needed.

This parameter should always be less than or equal to the *Maximum Number of VCs Needed* (max_vc).

Maximum Number of VCs Needed

The default for this parameter is 256, and the valid range is between 8 and 1023.

This parameter is used to determine how optimize the adapters internal tables to support Virtual Circuit operation, the default is 256. This must be configured when the adapter is first initialized. At minimum, it should be configured to be the same or greater than the Minimum Number of VCs.

Since this is a configurable only when the adapter and device driver is initialized, worst case situations must be considered. For SVC operation, ATM needs three virtual circuits for ILMI, SSCOP and ARP server, plus one for every active Classical IP partner, three virtual circuits for every LANE Ethernet or Token Ring network, and one for every LANE network partner.

In addition, the LANE ethernet or Token Ring configuration must have the *arp_cache_size* parameter configured to the maximum number of simultaneous connections, plus three for LANE service connections. It is defaulted to 32, which is insufficient for large networks. Leaving this at a low value causes AIX software to flush entries as attempts to communicate with more systems than arp_cache entries. The new entry must be obtained from the LES. This results in trashing.

If there are insufficient virtual circuits, the system ATM network layers will clear existing calls whenever it needs to send data and the ARP table does not contain the IP/ATM mapping for the destination. It will establish a new connection and then update the ARP table, purging the old entry. If incoming call requests are received, and there are no available virtual circuits, the incoming call will be rejected. If you have trouble accessing a system from specific other systems in a Large ATM network intermittently, then there is a good chance that there aren't enough virtual circuits available. The *atmstat* command will return: *Virtual Connections in use*, and *Maximum Virtual connections in use*, and *Virtual Connections Overflow*.

Virtual Connections Overflow is a counter of the times more virtual circuits were needed, and not a valid count of the absolute maximum number of virtual circuits needed. If this is non-zero, use your calculations to revise the Maximum Number of VCs needed parameter. If this is insufficient, make incremental adjustments to the Maximum Number of VCs needed parameter until the problem disappears or the maximum value is reached.

Maximum Virtual Connections in use, output from atmstat, reports the high-water mark up to the configured maximum value. If you are experiencing intermittent problems accessing this ATM or LANE interface from the network, and atmstat shows that the system in not using the maximum number of VCs, then the problem is most likely system congestion and contention for kernel heap to allocate virtual circuits. In this situation, dedicated kernel heap memory is needed to allow more virtual circuits to be configured without contending for memory. Adjust the Minimum Number of VCs needed parameter closer to the reported Maximum Virtual Connections in use value.

When LANE and CIP are used simultaneously to the same remote systems, it potentially doubles the number of virtual circuits that may be needed at a maximum. Because of the larger MTU sizes, CIP connections are more efficient and should be used in preference to LANE connections where possible. The LANE device *arp_cache_size* parameter can be tuned to reduce the number of simultaneous TCP/IP LANE partners and limit the need for VC's. This can be done if atmstat still reports Virtual Connection Overflow as a non-zero value.

SVC UNI Version

The default is auto_detect, possible values are auto_detect, uni-3.0, and uni-3.1.

This parameter sets the ATM Forum compatibility level for the ATM User-Network Interface. Auto-detect is generally sufficient. Interoperation with most ATM switches works well with auto-detect. However, there are several switches that do not support our *auto_detect* efforts and need the uni-3.1 setting. Problems of this nature usually appear when either the system or the switch is rebooted and connections cannot be established. Use the *atmvcstat* to verify that the first three virtual circuits listed are connected in these situations.

Software transmit queue length

The default for this parameter is 512, and the valid range is between 0 and 2048.

This parameter (sw_queue) defines the number of mbufs that the ATM device driver will queue up waiting to send to the adapter hardware queues. The trade-off is between the kernel heap space (management table size and mbufs that are potentially tied up waiting for space on the adapter) against the need to keep the adapter busy by providing a sufficient number of queued mbufs for the current system environment. Since the parameter is only used at power-up or during configuration of the adapter, the parameter should be set to accommodate the worst case system loading. The default of 512 should be sufficient for most applications.

The *atmstat* command provides two variables that can be used to tune this parameter.

- *Max Packets on S/W Transmit Queue*

  This shows what the high-water mark for the Software Transmit Queue was since the last time the statistics were cleared. This can be used to gauge how system overhead effected the ATM adapters operation. If the number remains lower than 75% of the maximum value, assuming that peak load periods for all applications have been encountered, then there is no need to adjust the value upwards. If the number remains less than 25% of the maximum value, the number should be decreased by 50% until the default of 100 is reached. If kernel heap space is very limited, there may be justification to reduce this value below 100, but it is not recommended.

- *S/W Transmit Queue Overflow*

  This parameter should be used to determine if dramatic increases for Software Transmit Queue size are needed. You should attempt to tune the Software Transmit Queue size by 10%-20% increases if overflows occur, until either the overflow occurrences cease or the limit has been reached. Typically overflows mean that there is significant loading of the system and resources of the system where the ATM software experiences longer delay times in processing mbufs for transmission.

Sonet or SDH interface

This ATM adapter parameter is used to control the mode of operation of the PHY on the adapter. The default is to use the Sonet setting. Networks can be either SONET or SDH compatible. Change this only if when attaching to a network using SDH.

Provide SONET clock

Most, but not all, switches provide the clock to use for transmission and reception. This parameter is generally set to a 0 unless using direct connection PVC's, where one system must provide the clock for both, or connecting to a switch that does not provide a clock.

5.3.4 Verify the settings

After you have modified any of your adapter settings it always wise to check these modifications. One way of doing this is by issuing the following command:

```
# lsattr -El atm0
```

You will get a screen similar to the following.

```
# lsattr -El atm0
dma_mem        0x800000    N/A                                       False
regmem         0x1be08000  Bus Memory address of Adapter Registers     False
virtmem        0x1be10000  Bus Memory address of Adapter Virtual Memory False
busintr        13          Bus Interrupt Level                       False
intr_priority  3           Interrupt Priority                        False
use_alt_addr   no          Enable ALTERNATE ATM MAC address          True
alt_addr       0x0         ALTERNATE ATM MAC address (12 hex digits)    True
sw_txq_size    100         Software Transmit Queue size              True
rx_que_size    0           N/A                                       True
max_vc         1024        Maximum Number of VCs Needed              True
min_vc         32          Minimum Guaranteed VCs Supported          True
rv_buf4k_min   0x30        Minimum 4K-byte pre-mapped receive buffers  True
interface_type 0           Sonet or SDH interface                    True
adapter_clock  0           Provide SONET Clock                       True
uni_vers       auto_detect N/A                                       True
#
```

Check all the displayed attributes and make sure that these settings are correct.

Chapter 6. Classical IP over ATM

Classical IP is the common term for the TCP/IP operations over ATM defined by the Internet Engineering Task Force (IETF) RFCs 1577 and 2225. Each ATM end station is known by a unique, 20-byte ATM Address. The RFCs define the mapping and resolution of the ATM addresses to TCP/IP addresses using ATMARP over AAL 5 connections.

In traditional LANs, TCP/IP subnets are usually geographically limited by physical location and wiring. In ATM networks, these physical restrictions are removed. In fact, as explained in section 3.6, “System considerations” on page 27 users can participate in multiple IP subnets (up to 256) on a single ATM adapter.

Virtual channel types

End stations attached to an ATM switch may be connected via PVCs or SVCs. The benefit of running PVCs in a normal switched ATM environment is that end stations are, essentially, always linked because the end stations dedicate a virtual channel connection (a particular VPI:VCI pair) between the two systems. When the two stations need to communicate, there is no call setup delay as with SVCs because this logical point-to-point connection is always up and available. Intervening switches along the path between the two end stations must also be configured with VPI:VCI pairs identifying the unique PVC connection. A PVC provides two IP end stations with a dedicated communication path across the network.

With SVCs, the communication path between the two systems is set up ad hoc across the switch(s) each time a new connection is needed. Classical IP end systems join a Logical IP Subnets (LIS) which is based upon an ARPclient/server relationship model. LIS members should have the same IP subnet address, network mask, and TCP/IP maximum transmission unit (MTU) size. One end station (or a network device) acts as the ARP server providing address resolution protocol services for the clients within the LIS. The ARP server keeps a table mapping IP addresses and their corresponding ATM addresses for all clients systems in the LIS. LIS stations may be connected to the same switch or geographically separated by a large network of ATM switches. Support for backup ARP servers, defined in RFC 2225, is included starting with AIX V4.3.3. Direct communication with end stations outside the LIS is not defined -- however, this is TCP/IP, so an end station with interfaces on two LISs can act as an IP router and pass traffic between the LISs.

Virtual channel utilization

RS/6000 Micro Channel and PCI ATM adapters support up to 1024 and 2048 virtual channels (VCs), respectively. (The adapter notes in Chapter 5, “Configuration and attachment” on page 37 explain AIX ATM device driver minimum, maximum and default values). When planning an ATM environment, it is helpful to know how AIX utilizes VCs.

Users should expect some number of VCs to always be in use for administrative and background purposes. The exact number varies depending upon the configuration. The ATM adapter to switch connection, for example, takes two PVCs (one for ILMI and the other for signaling).

One VC is allocated for each user-configured PVC connection. For SVCs, one VC is maintained for the connection to the ARP server and then one VC is opened for each new connection to another LIS member. Switched virtual channels between LIS members are set up, closed, and reused as needed. If all available virtual channels are in use, outgoing calls will fail and incoming calls will be rejected until virtual channels are freed and are again available for reuse.

Consider this VC utilization example. Suppose a system has one ATM adapter (atm0) and three PVC connections to other end stations are defined across the at0 interface. In addition, this end station is an ARP client on at1 and at2 with an active connection to one other LIS member on each. How many VCs would be used? The answer is nine. Here is the breakdown:

2	Connection to the switch (two PVCs per ATM adapter)
3	One per PVC defined (these are tied up whether currently in use or not)
2	One per ARP server (our system has two)
2	One for each active connection to another LIS member our system has one per LIS)
___	
9	Total VCs in use

In this example, seven VCs would always be tied up. The switch and ARP server connections are always maintained and the user-configured PVC connections are reserved whether or not they are currently in use. So, any variation in the number of VCs in use will come from changes in currently active connections to other LIS members as VCs are opened, closed, and reused.

The *General Statistics* section of the *atmstat* command provides valuable information about virtual connection usage. It details the current number of VCs in use, the maximum available, and any overflow requests.

6.1 Back-to-back (PVC) scenario

Although ATM workstations are normally attached to an ATM switch, it is possible to connect and operate two RS/6000s running Classical IP over a PVC and cabled in a back-to-back fashion. This non-switch setup is quite useful as a test environment when RS/6000 adapters, cabling, and software are available, but an ATM switch is not, allowing users hands-on experience with ATM and Classical IP PVC installation, configuration steps, and operations.

It operates under these restrictions:

- connects two systems only; a switch is required to interconnect more systems
- possible in PVC environments only; SVCs rely upon a switch
- can run Classical IP operation only; LAN Emulation requires SVCs
- systems must be configured identically; there is no switch to provide speed and interoperability services between the stations. The systems must have matching adapters (same speed and cable type) and should agree on software options such as same frame size, VPI:VCI pair, and UNI level.
- high speed performance can be achieved, but it should not be viewed as an equivalent to a production environment because factors such as switch services or delays are excluded.

In a switched environment, however, adapter speeds and VPI:VCI pair values need not match because the intervening ATM switches along the path provide these interoperability services.

6.1.1 Overview

In this scenario, systems iron and steel will be linked back-to-back, as Figure 16 on page 74 depicts. Although the two systems have different bus architectures (iron is Micro Channel and steel is PCI) they can be linked in this fashion as long as their adapters are the same speed and cable type.

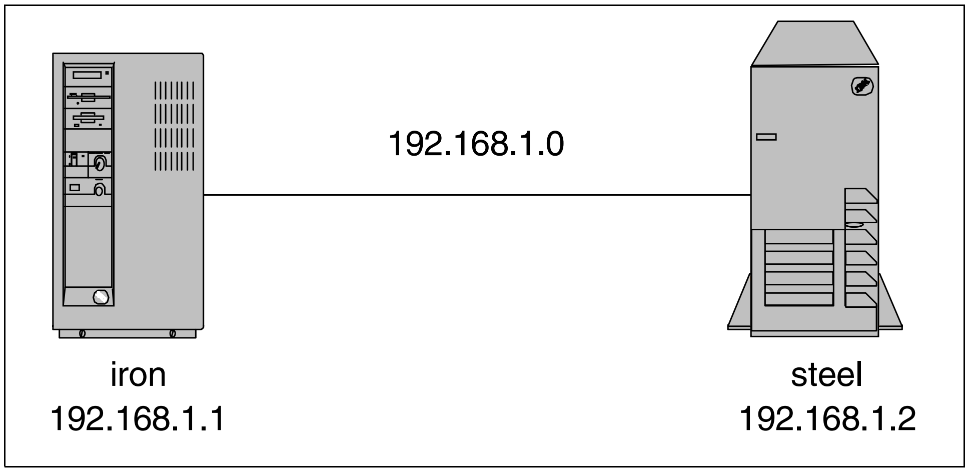


Figure 16. Back-to-back connection

In this case, each system has a Turboways 155 Mbps Multi-Mode fiber (MMF) adapter. Both adapters have an SC fiber connector, so we need a single MMF cable with SC connectors on either end. Plug the fiber (or specially wired UTP) cable ends directly into the adapters in both systems.

Note

The transmit and receive signals must be crossed between the two systems. For multimode fiber, attach the cables so that transmit on one system flows into receive on the other and vice versa. If the cable has a shroud forcing the two connectors to be plugged in one way only, this reversal has probably been done already by the cable manufacturer. For unshielded twisted pair, use a special crossover cable, wired as described in 5.1.4.1, “UTP cable connection” on page 38.

The Turboways 155 UTP and MMF ATM adapters use SONET clocking. This function is normally provided by the ATM switch but, since there is no switch in this configuration, one system must be configured to provide the clocking.

Systems iron and steel are each running AIX V 4.3.3, but any software level that supports PVC and Classical IP operations would work. The VPI:VCI values set in SMIT must be the same on both systems. It is recommended, but not mandatory, that matching UNI levels are specified. Leave the other parameters at the defaults.

6.1.2 SMIT configuration

Assume interface/adapters are *Available* and properly connected to the ATM network as described in Chapter 5, “Configuration and attachment” on page 37. We will be using the at0 interface on device atm0 on both systems. The basic configuration steps shown here also apply to PVCs in an ATM switch environment except the VPI:VCI pair are provided by the switch administrator and the SONET clock will probably remain off at the end station.

6.1.2.1 Check the adapter configuration

Because we are running 155 Mbps ATM adapters, we must set SONET clocking on one system. In the Change/Show Characteristics of an ATM Adapter SMIT menu (fastpath smit chg_atm) set the Provide SONET clock parameter on system steel to 1. On system iron, this parameter should be off (set to zero). For more details see 5.2.3, “Installation steps for PCI adapters” on page 41 and 5.3.3, “Installation steps for MCA adapters” on page 51.

The difference is:

	system iron	**system steel**
`Provide SONET clock`	`[0]`	`[1]`

Note

This step is not necessary on Turboways 100 and 25 Mbps ATM adapters because they do not use SONET clocking.

6.1.2.2 Add the IP interface

Add and configure the IP interface (at0) on both systems using the following menu progression.

```
smitty
  Communications Applications and Services
    TCP/IP
      Further Configuration
        Network Interfaces
          Network Interface Selection
            Add a Network Interface
              Add an ATM Network Interface
```

or use the fastpath: `smit mkinetat`

```
                         Add an ATM Network Interface

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                  system iron     system steel

* INTERNET ADDRESS (dotted decimal)               [192.168.1.1]   [192.168.1.2]
  Network MASK (hexadecimal or dotted decimal)    [255.255.255.0] [255.255.255.0]
  Network Interface                               [at0]           [at0]
  Connection Type                                  pcv             pvc
  ATM Server Address                              []              []
  Alternate Device                                [atm0]          [atm0]
  Idle Timer                                      []              []
  Best Effort Bit Rate (UBR) in Kbits/sec         []              []
* ACTIVATE the Interface after Creating it?        yes             yes
```

Internet Address	As for any TPC/IP interface the Internet Address should be unique to this interface and entered in dotted decimal form.
Network Mask	The Network Mask for this interface. It is an optional field.
Network Interface	Specify at0 as the name of the network interface you are adding. The name for ATM Classical IP network interfaces are atN, where N is a decimal number (for example, at1). With ATM, multiple IP interfaces may be configured on one device, so while other IP interfaces normally carry the same number as the device, there is no one-to-one relationship for ATM interfaces. For example, the at1 and at2 interfaces can be configured on top of the atm1 device, while at3 an at6 might be configured for atm0. Use the Alternate Device field to make sure the interface and device names are specified according to your own needs. A valid interface name is required for this field.
Connection Type	Specify PVC for this back-to-back configuration. Other valid options include svc_s for the ARP server on an SVC and svc_c for the ARP client on the SVC connection. The ARP clients must specify the ATM 20-byte address of the client's designated ARP server in the ATM Server Address field.

ATM Server Address	Leave this field blank for back-to-back configurations. This field contains the 20-byte ATM Address of the ARP server used in SVC environments. It is required for ARP client (svc_c) systems.
Alternate Device	This is the device name (atm0 or atm1) associated with this interface. In our scenario, the device is atm0 for both systems. If you are configuring multiple interfaces on a single device, you must enter a value in this field. By default, the at0 interface is on the atm0 device and the at1 interface is on the atm1 device. This field is optional if you are configuring only one interface on a device, but we recommend eliminating the guesswork and system assumptions by entering a device name for every interface.
Idle Timer	Leave this field blank because it is not used with PVC connections. Specify a decimal number representing the minutes allowed on a temporary virtual connection on this network interface before the connection is invalidated. The default is 60 minutes.
Best Effort Bit Rate (UBR) in Kbits/sec	Leave this field blank because it does not apply to PVC connections. It specifies the bit rate for the Best Effort connections that are initiated from the configuring interface. This network interface supports the Classical IP over ATM connections. All of the outgoing SVC connections will be set as Best Effort connections with the specified bit rate as the forward and backward peak rate. When this attribute is 0 (by default), the network interface uses the peak rate supported by the associated device as the Best Effort bit rate. Specify a decimal integer in K bits per second. For example, the value 155000 represents 155M bits per second.
Activate the Interface After Creating it?	If yes, the added interface will be activated.

Note:

The TCP/IP mtu size for the interfaces should match, so leave it at the default value (9180). See 6.4.5, “Configuring the TCP/IP MTU size” on page 106 for details.

6.1.2.3 Configure the PVC(s) for IP over ATM

Add and configure the IP interface (at0) on both systems using the following menu progression.

```
smitty
  Communications Applications and Services
    TCP/IP
      Further Configuration
        Network Interfaces
          Network Interface Selection
            PVCs for IP over an ATM Network
              Add a PVC for IP over an ATM Network
```

or use the fastpath: `smit mkatmpvc`

```
                     Add a PVC for IP over an ATM Network

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                   system iron  and steel


                                                     [Entry Fields]
  PVC Description (Optional)                       [iron/steel bk2bk PVC]
* VPI:VCI                                          [0:44]
  Network Interface                                [at0]
  Destination IP Address                           []
  Automatically Discover Destination IP Address     yes                +
  LLC Encapsulation                                 yes                +


F1=Help F2=Refresh F3=Cancel F4=List F5=Reset F6=Command F7=Edit F8=Image
F9=Shell F10=Exit Enter=Do
```

PVC Description	This field is optional, but recommended. You may enter any text you wish.

VPI:VCI	This is the most important field on the SMIT screen. Specify 0:44 on both systems as the VPI (virtual path identifier) and the VCI (virtual connection identifier) for this back-to-back PVC connection scenario. In a switched environment, the ATM switch administrator normally tells the end user which values have been configured for the PVC connection. This pair of decimal integers must be separated by a : (colon) character (for example, 0:82). In AIX, the VPI is always 0 and VCI number is limited by the maximum number of virtual connections supported by the device which is 1023 for Micro Channel and 2047 for PCI adapters. VCIs 0 through 31 are reserved, so you should not use them.
Network Interface	Specify the network interface associated with this PVC (at0 in our scenario). If you defined the Destination IP address for the PVC, this field is optional.
Destination IP Address	Leave this field blank. It specifies the IP address of the destination host for this PVC. The address format is the standard IP address format, 4-bytes, each byte separated by a . (period). This field is optional because the ATM network interface has the capability of obtaining the remote IP address by using the INverseARP protocol. If you want the network interface to obtain the remote IP address for the PVC connection, do not enter a value in this field. If you do not enter a value in this field, make sure you define the Network Interface, and make sure that both sides of the PVC connection support the ATM ARP protocol and LLC/SNAP encapsulation.
Automatically Discover Destination IP Address	Specify whether the ATM ARP protocol will be supported on this PVC. The default value is yes. Select no for compatibility if the remote host does not support it. If you specify no, you must define the Destination IP address for this PVC.

LLC Encapsulation	Specify whether LLC/SNAP encapsulation will be supported on this PVC. The default value is yes. Select no for compatibility if the remote host does not support it. If you specify no, the ATM ARP protocol cannot be supported so the Destination IP address of this PVC must be defined.

6.1.3 Test the connection

The systems should now be able to communicate with each other via TCP/IP over an ATM PVC. Test the connection using ping or telnet. If using a system name rather than IP address, enter the remote system in your /etc/hosts table. (For additional information see section 6.5.1. Name Resolution Tips) On system steel, for example, key one of the following commands to send three ping packets.

```
ping -c 3 iron   -or-

ping -c 3 192.168.1.1
```

```
PING iron: (192.168.1.1): 56 data bytes
64 bytes from 192.168.1.1: icmp_seq=0 ttl=255 time=1 ms
64 bytes from 192.168.1.1: icmp_seq=1 ttl=255 time=1 ms
64 bytes from 192.168.1.1: icmp_seq=2 ttl=255 time=1 ms

----iron PING Statistics----
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 1/1/1 ms
```

The *arp -t atm -a arp* table output for the successful connection is as follows:

```
PVC
   VPI:VCI  Handle        IP Address          llc     arp
   0:44     3             iron(192.168.1.1)   True    True
```

The `ifconfig at0` output is as follows:

```
at0: flags=8000861<UP,NOTRAILERS,RUNNING,SIMPLEX>
          inet 192.168.1.2 netmask 0xffffff00
```

The *netstat -in* output for the at0 interface should like like this:

```
Name   Mtu    Network    Address        Ipkts  Ierrs  Opkts  Oerrs  Coll
lo0    16896  Link#1                      290  0      290    0      0
lo0    16896  127        loopback         290  0      290    0      0
at0    9180   Link#2     0.6.ab.12.33.2     4  0        4    0      0
at0    9190   192.168.1  192.168.1.2        4  0        4    0      0
```

6.2 ARP server (SVC) scenario

Whereas systems on a LAN are identified by MAC address, ATM end stations are uniquely identified by their 20-byte ATM address. Therefore, the TCP/IP (Classical IP) ARP server maintains a table mapping IP addresses and their corresponding ATM addresses rather than MAC addresses. Each active system also maintains a local cache of recently used information, but the ARP server is the authority on the logical IP subnet (LIS) maintaining a master list of all systems who have joined the LIS.

How does it get the information? It comes from the systems themselves. One-by-one, as end stations join the LIS, they register their ATM and IP addresses with the ARP server. Then, when other members of the LIS need ATM addresses for systems not in their local cache, they request it from the ARP server, Figure 17 on page 82 depicts such an environment.

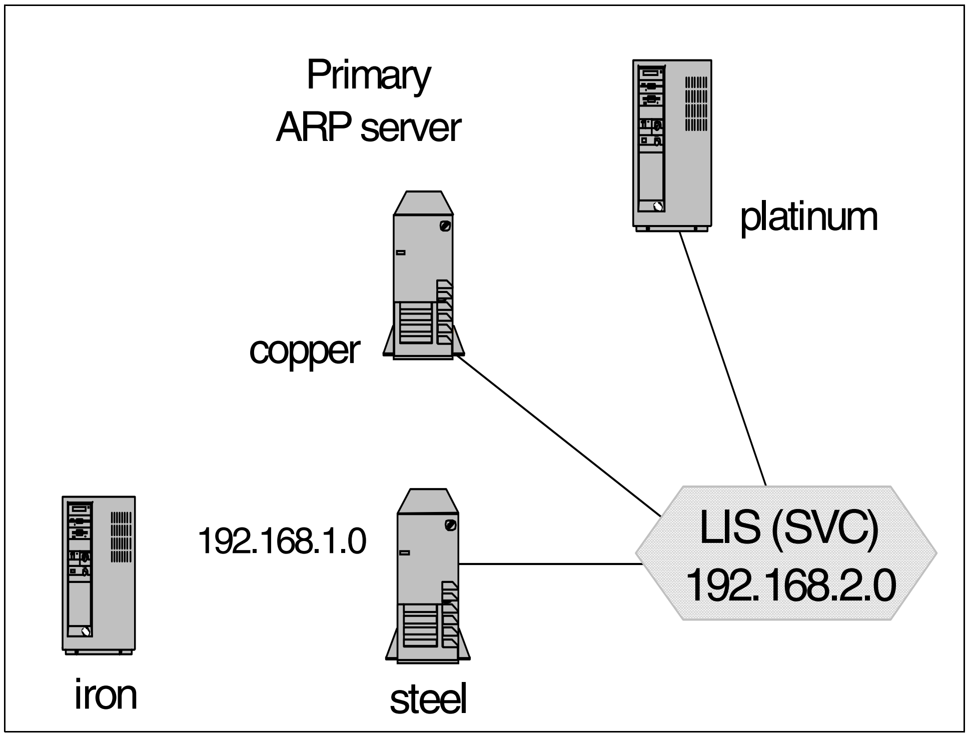


Figure 17. ARP server scenario

How do AIX systems know the role they play in the LIS? It is part of the SMIT configuration. Systems are configured as either svc_s or svc_c (ARP servers or clients). For client systems, the 20-byte ATM address of the ARP server system must be entered in SMIT so, if system copper is the ARP server for the 192.168.2.0 LIS, steel, platinum and brass are configured as svc_c with copper's ATM address listed as the ARP server.

6.2.1 Overview

In this scenario, systems steel, platinum, and copper are connected to a real ATM switched network and will be configured as members of the 192.168.2.0 LIS. System steel's atm0 adapter was connected to iron in previous the back-to-back PVC scenario, so steel's second ATM adapter (atm1) is connected to the ATM network switch in this one. ATM switches and network devices often provide ARP server function for attached end stations. In our environment, however, system copper will be configured as the ARP server.

Systems copper and steel are running AIX V 4.3.3. Platinum will be used as a non-RFC 2225 system in the next scenario and is running AIX V 4.2.1.

Note

In this simple example, all three systems have the same ATM address prefix, indicating they are attached to a single ATM switch. They could, however, interoperate just a well if they were connected to different ATM switches in a fully enmeshed ATM network. In this case, the unique 13-byte ATM address prefixes on each box would signal the end user that the systems are attached to different switches.

6.2.2 SMIT configuration

We will be following the naming and addressing conventions outlined in Chapter 4, “Sample Network” on page 33, configuring interface at1 on adapter atm1 on system steel and at0 on adapter atm0 on the other two.

6.2.2.1 Check the adapter configuration

Verify that the interface and adapters are *Available* and properly connected to the ATM network as described in Chapter 5, “Configuration and attachment” on page 37.

6.2.2.2 Add the ARP IP interfaces

Add and configure the IP interface on both systems using the following menu progression. Configure system platinum like steel, using the IP address 192.168.2.4 on interface at0 and adapter atm0.

```
smit
  Communications Applications and Services
    TCP/IP
      Further Configuration
        Network Interfaces
          Network Interface Selection
            Add a Network Interface
              Add an ATM Network Interface
```

or use the fastpath: `smit mkinetat`

```
                           Add an ATM Network Interface

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                               system copper   system steel

* INTERNET ADDRESS (dotted decimal)            [192.168.2.3]   [192.168.2.2]
  Network MASK (hexadecimal or dotted decimal) [255.255.255.0] [255.255.255.0]
  Network Interface                            [at0]           [at1]
  Connection Type                               svc_s           svc_c
  ATM Server Address                           []              [47.0.5.80.ff.e1.0.0.>
  Alternate Device                             [atm0]          [atm1]
  Idle Timer                                   []              []
  Best Effort Bit Rate (UBR) in Kbits/sec      []              []
* ACTIVATE the Interface after Creating it?     yes             yes
```

The greater than sign (>) indicates that the ATM Server Address for system copper continues off the screen. The AIX V4.3.3 help text for the *ATM Server Address* field provides this additional information:

```
Specify the 20-byte ATM address of the designated ARP server for this
Logical IP Subnet (LIS) in hexadecimal numbers, using a period as the
delimiter between bytes. You can omit the leading zero of each byte. For
example, 0.1.2.3.4.5.6.7.8.9.a.b.c.d.e.f.10.11.12.13 is a valid address.

Normally, the server's ATM address can be obtained from the server after
the server is registered with the switch.

You can enter a list of ARP server addresses with a comma as a delimiter.
The first entry is considered to be the Primary ARP server. Others are
considered to be Secondary ARP servers.
```

Note

It is interesting to note that the Classical IP ARP scheme is designed with a dependency on only one address; that of the ARP server. Client system addresses are not fixed in fact, clients do not even know their own ATM address until after they have joined the network. This gives ATM network administrators freedom to move clients within the network.

If an end station is moved to another ATM switch, the resultant change in their ATM address has no effect on operations. The only LIS requirement is that the network can deliver packets for the ARP server using the 20-byte ATM address specified in SMIT.

6.2.3 Test the connection

You should now be able to communicate between LIS members. Test the connection using ping or telnet. If using a system name rather than IP address, enter the remote system in your /etc/hosts table (For additional information see section 6.4.1, “Name resolution tips” on page 93). On system copper, for example, key one of the following commands to send three ping packets.

```
ping -c 3 steel
```

-or-

```
ping -c 3 192.168.2.2
```

```
PING iron: (192.168.2.2): 56 data bytes
64 bytes from 192.168.2.2: icmp_seq=0 ttl=255 time=1 ms
64 bytes from 192.168.2.2: icmp_seq=1 ttl=255 time=1 ms
64 bytes from 192.168.2.2: icmp_seq=2 ttl=255 time=1 ms

----iron PING Statistics----
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 1/1/1 ms
```

View the local ATM ARP (address resolution protocol) cache using the *arp -t atm -a* command. (This is also a good way to learn a system's ATM address after registration with the switch as mentioned in the help text). The ATM ARP table on system copper might look something like this:

```
# arp -t atm -a

SVC - at0 on device atm0 -
==========================
at0(192.168.2.3) 47.0.5.80.ff.e1.0.0.0.f2.15.16.45.0.6.ab.12.34.3.0
 IP Addr    VPI:VCI Handle ATM Address
platinum(192.168.2.4) 0:102 8
47.0.5.80.ff.e1.0.0.0.f2.15.16.45.0.6.ab.12.34.4.0

steel(192.168.2.2) 0:584 7
47.0.5.80.ff.e1.0.0.0.f2.15.16.45.0.6.ab.12.34.2.1

?(192.168.2.8) 0:101 6 47.0.5.80.ff.e1.0.0.0.f2.15.16.45.0.6.ab.12.34.8.0
```

Note

If executing this command on a client system, expect to see a subset of this server's table, with the first line identifying the local system. On systems with multiple interfaces, the command output groups the information by interface.

Notice that leading zeros in the command output are truncated, as explained in the help text, so that 47.00.05.80 becomes 47.0.5.80. Also, notice the host names are filled in for all but the last system listed. This is a cosmetic issue, not a problem affecting LIS operations. The arp command tries to resolve the IP address to a host name (generally by consulting the local /etc/hosts file or domain name server) for the end users convenience. If it cannot resolve, a "?" is posted in the output. When name resolution services are unavailable, the user will notice a delay in command while the name resolution attempt times out. Eliminate the name resolution step, and the resultant delay, by adding the n flag, making the command syntax *arp -t atm -an*.

Note

The ARP client/server concept applies to ATM Classical IP SVC environments only; ARP tables are not used in Classical IP PVC or ATM LAN Emulation operations.

6.2.4 Interoperability with the back-to-back scenario

You can enable a traffic flow between these two scenarios by configuring system steel as an IP router and setting up routes properly on the other systems.

System steel has a presence on each network and will pass the traffic between the two IP subnets if the ipforwarding option is enabled using the *no* command. Check the current value using the command:

```
no -a | grep ipfor
```

Set ipforwarding on using the command:

```
no - o ipforwarding=1
```

Set this forwarding value permanently by placing it anywhere in the /etc/rc.net startup file.

System steel knows about the two network because it has a direct presence on both. The other systems (iron, platinum, copper, etc) must specify a route through steel to the other network. They could each specify a network route. However, in this simple example of just two IP subnets, each system could list steel at their default route and traffic would flow as desired.

For more information, see 6.4.4, “IP routing in the sample environment” on page 102.

6.3 Backup ARP server (SVC) scenario

Prior to the AIX V4.3.3 announcement, support for a backup Classical IP ARP server, the most notable feature of RFC 2225, was added as an AIX enhancement via PTF. Because RFC 2225 enhances the ATM Classical IP ARP support as originally defined in RFC 1577, it is sometimes referred to as RFC 1577+.

The last Figure 17 on page 82 illustrates a sample LIS environment. System copper, the ARP server, plays a key role in the LIS operation. But, what happens if copper suddenly becomes unavailable; if it malfunctions or is unplugged or rebooted? Existing connections will continue for a time but, when local ARP cache entries expire, LIS connectivity is lost until the ARP server is restored. RFC 1577 actually introduced this single point of failure because it specified that ATM classical IP client systems could designate one and only one classical IP ARP server per logical IP subnet (LIS). RFC 2225 remedies this with a backup ARP server scheme for ARP clients.

RFC 2225 designers were challenged to create a backup ARP server facility, while preserving interoperability with existing RFC 1577 end stations. Actually, RFC 2225 defines the backup facility but it does not define how systems rejoins the primary ARP server. AIX V4.3.3 RFC 2225 secondary ARP server protects interoperability because it serves as a temporary measure, not as a permanent replacement. AIX V4.3.3 supports the assumption that networks have a mixture of RFC 1577 and 2225 boxes, so the goal is to provide a temporary backup scheme only; resuming the primary configuration just as soon as possible.

Comparison of RFC and 2225

Here is how it works:

1. **ARP server definition**

Both ARP client types list the ATM address of the ARP server in SMIT. (If AIX V4.3.3 is installed, it does RFC 2225 -- it is not anything that is turned on and off -- it is always on!).

- RFC 1577 clients define a single ARP server.
- RFC 2225 ARP clients define a primary ARP server address, optionally followed by one or more secondary ARP servers. The backup ARP server address(es) are listed in the SMIT ARP Server address field as described in the AIX V 4.3.3 help menu. (The actual number of servers is limited by the size of the SMIT screen field). Most AIX users will find a primary and a couple of secondaries sufficient for their environment.

2. **Client LIS registration**

   Both ARP client types initiate contact with the server. However, one key RFC 2225 design feature is that the initiator of the LIS registration process is reversed.

   - the RFC 1577 ARP server initiates the LIS registration procedure
   - the RFC 2225 ARP client initiates the LIS registration process, re-registering with that server every 15 minutes.

3. **ARP server failure**

   Both ARP client types detect the server failure when the server fails to respond or they are notified of a loss of connection to the server (such as the virtual channel connection has been closed). Communication with other LIS members may continue for a time but, once the local ARP cache entries expire, LIS connectivity is lost until the primary is restored.

   - During the outage, RFC 1577 clients continue to try to make contact with the ARP server. They are, however, essentially out of commission on the LIS until their ARP server is restored.
   - RFC 2225 ARP clients begin attempts to register with each address in the backup list until successful. Clients should be able re-establish communication with other LIS members running RFC 2225. Note, at this time the current LIS is only a subset of the original LIS, since RFC1577 clients are not participants. AIX RFC 2225 clients will keep trying the primary every 15 minutes and will rejoin the original LIS when their next registration attempt is successful.

4. **ARP server restoration**

   During the outage, both client types continue to try to contact the primary ARP server so as clients detect that the primary server is back in service, all operations for both types of clients is also restored.

Some network devices, like the IBM 8210 Nways Multiprotocol Switched Services (MSS) Server, take this backup ARP server a step further in that they implement a server-to-server protocol to achieve a sort of load balancing of client requests among a group of ARP servers spread across multiple MSS devices. In this scenario, RFC 2225 clients on a failed ARP server will try to reestablish contact with a backup. Once the primary server recovers, the original distributed ARP server balancing scheme is restored.

6.3.1 Overview

This scenario updates the last one. System copper is still the ARP server, but system brass was added as the backup ARP server. System platinum runs AIX V 4.2.1, so it cannot participate in a backup server environment. System steel will be updated to list brass as the secondary ARP server.

Secondary ARP server selection

Systems that are NOT active members of the LIS are the best candidates for secondary ARP servers because they generally participate in the LIS only when they are acting as the LIS ARP server. This is due, in part, to the secondary's SMIT configuration; it is identical to the primary's. The bigger factor, however, is not ATM but rather TCP/IP routing algorithms.

In the sample environment, system brass is actually an active participant in the 192.168.3.0 LIS, so it is a good choice for secondary ARP server role in the 192.168.2.0 LIS (see Figure 18 on page 90 for an example). The connection type field in the SMIT configuration menu is svc_s for both the primary (copper) and secondary (brass) LIS ARP servers. ARP clients join the LIS and build their local ARP caches based upon information exchanged with the server. As a server, brass awaits client contact and subsequent registration for LIS operations. This will only happen when the primary ARP server (copper) has failed. Until that time, brass literally has no one with which to talk.

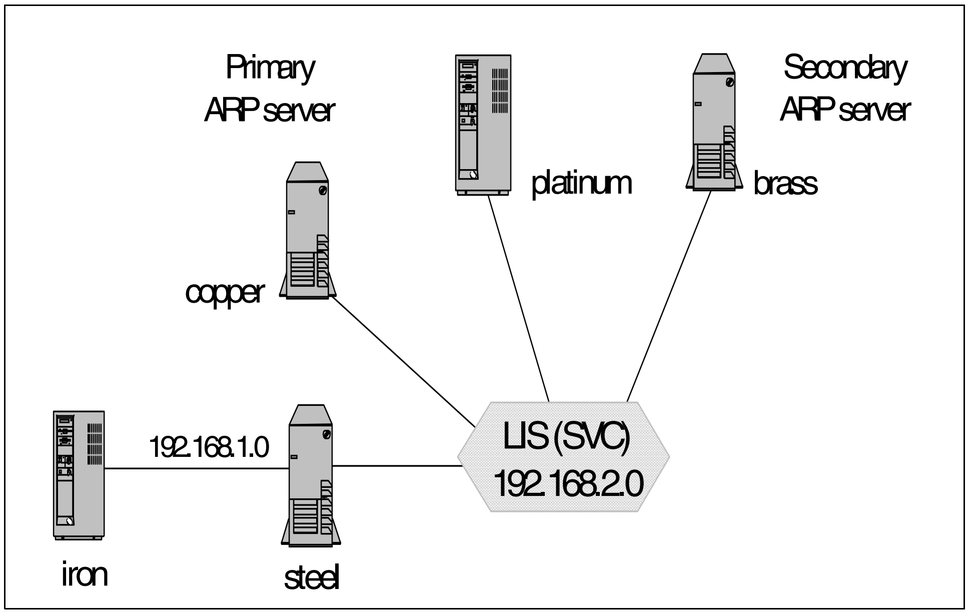


Figure 18. Backup ARP server scenario

As an alternative one could configure another CIP interface for the same LIS; perhaps at2 with an IP address of 192.168.2.9. Now brass's at0 interface (IP address 192.168.2.5) can sit and wait for copper to fail, while the at2 interface is the active presence in the LIS. With at2's unique ATM and IP addresses, the ATM LIS part of the situation is resolved.

Unfortunately, the TCP/IP routing table provides the next challenge. With two IP interfaces on the same IP subnetwork, users need two routing table entries for the local network. The routing table only allows one and, even if two were posted, only one would be used. For a more detailed explanation, see 6.4.3, “Configuring multiple AIX IP interfaces on the same IP subnet” on page 97. As with many TCP/IP situations, one might be able to work around this in some way; perhaps by carefully selecting the routing table interface or by using a subnet mask to alter the local system's interpretation of the network. But, when all is said and done, it simply might not be worth the effort.

Thus, the secondary ARP server recommendation stands. Select an end station with no on-going involvement in the LIS.

6.3.2 SMIT configuration

This scenario picks up where the last one took off, so it assumes the previous scenario setup was just successfully completed.

6.3.2.1 Check the adapter configuration

Verify that the interface and adapters are *Available* and properly connected to the ATM network as described in Chapter 5, “Configuration and attachment” on page 37.

6.3.2.2 Detaching existing IP interfaces

We will modify the at1 IP interface configuration on system steel; so it is a good idea to detach it first with the following command:

```
ifconfig at1 detach
```

If system brass is an existing system, you should do this for the interface at0 (the interface we are using our example) also.

6.3.2.3 Configure the systems for backup ARP service

Configure brass as an ARP server and add brass as backup ARP server on system steel. Use the smit commands as listed in the following menu progression. If brass is a new system, used these parameters to complete the "Add ATM Interface" menu as shown in 6.2.2.2, “Add the ARP IP interfaces” on page 83.

```
smit
  Communications Applications and Services
    TCP/IP
      Further Configuration
        Network Interfaces
          Network Interface Selection
            Change / Show Characteristcs of a Network Interface
```

or use the fastpath: `smit chinet`

```
                        Change / Show an ATM Network Interface

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                 system brass    system steel

  Network Interface                              [at0]           [at1]
  INTERNET ADDRESS (dotted decimal)              [192.168.2.5]   [192.168.2.2]
  Network MASK (hexadecimal or dotted decimal)   [255.255.255.0] [255.255.255.0]
  Connection Type                                 svc_s           svc_c
  ATM Server Address                             []              [47.0.5.80.ff.e1.0.0.>
  Alternate Device                               [atm0]          [atm1]
  Idle Timer                                     []              []
  Best Effort Bit Rate (UBR) in Kbits/sec        []              []
  Current STATE                                   up              up
```

It is not necessary to reboot the systems after this change.

System brass is now configured as an ARP server. The ATM addresses are too long to properly show this as a single long line in the SMIT menu and below. However, system steel's ATM s erver address field should specify the addresses for copper (the primary) then brass (the secondary) separated with a comma like this:

```
47.0.5.80.ff.e1.0.0.0.f2.15.16.45.0.6.ab.12.34.3.0,47.0.5.80.ff.e1.0.0.0.f2.15.16.45.0.6
.ab.12.34.5.0
```

6.3.3 Test the connection

Use the *ping* command to verify system steel's continued participation in the LIS. Assuming that was successful, try testing the backup server scheme. If a downlevel AIX system, like system platinium in the scenario, is available, observe what happens to it by running the *arp -t atm -a* command during the test.

1. Execute the *arp -t atm -a* command on systems steel, platinum, and brass (Redirect the output to a file to save a copy of the starting point output).
2. Now shutdown system copper, the primary ARP server.
3. Execute the *arp* command every two or three minutes on each system until you observe a change in the output.

The output on the three systems should show:

- on steel, brass is the new ARP server. And, in a larger environment, entries for other end stations joining brass's LIS (perhaps others from the failed LIS) begin to appear.
- on brass, entries for steel and any other stations that have registered on this LIS.
- on platinum, the ARP server connection has been lost when the VPI:VCI values are zero.

Status Information

The following *lsdev* command output provides a handy snapshot summary of the ATM situation (adapters and Classical IP interfaces) on system steel.

```
# lsdev -Cc adapter
# lsdev -Cc adapter
scsi0  Available 04-A0 Wide SCSI I/O Controller
scsi1  Available 04-B0 Wide SCSI I/O Controller
iga0   Available 04-03 GXT110P Graphics Adapter
sa0    Available 01-D0 Standard I/O Serial Port 1
sa1    Available 01-E0 Standard I/O Serial Port 2
sioka0 Available 01-G0 Keyboard Adapter
sioma0 Available 01-H0 Mouse Adapter
fda0   Available 01-I0 Standard I/O Diskette Adapter
atm0   Available 04-08 IBM PCI 155 Mbps ATM Adapter (14107c00)
ppa0   Available 01-C0 Standard I/O Parallel Port Adapter
paud0  Available 01-F0 Ultimedia Integrated Audio
atm1   Available 04-02 IBM PCI 155 Mbps ATM Adapter (14107c00)
tok0   Available 04-07 IBM PCI Tokenring Adapter (14101800)
```

```
# lsdev -Cc if
```

```
# lsdev -Cc if
lo0 Available Loopback Network Interface
tr0 Available Token Ring Network Interface
at0 Available ATM Network Interface
at1 Available ATM Network Interface
```

6.4 Advanced TCP/IP considerations

In this section will discuss more advanced TCP/IP topics like Domain Name Services (DNS), IP routing, and multiple IP interfaces on one ATM adapter.

6.4.1 Name resolution tips

Testing new TCP/IP connections on systems that are already a part of a name service domain can be challenging. Testers often find it easier to type host names than IP addresses, but they want a short term name resolution method; they do not want to ask the DNS administrator to make a temporary name resolution table change.

6.4.1.1 Local /etc/host

There are two alternatives in this situation:

- specify IP addresses in all commands (thereby avoiding name resolution altogether)
- alter the name resolution order via the /etc/netsvc.conf file (picking up the temporary values in local /etc/hosts table)

The *netsvc.conf* file overrides the default and the */etc/irs.conf* behavior of name resolution on your system, providing a handy temporary name resolution workaround. For a test environment, enter the temporary host name and IP address in the /etc/hosts file, then create the /etc/netsvc.conf configuration file and specify the desired ordering. The following entry accommodates both the temporary (local) and permanent DNS (bind) name resolution users by specifying the following search order:

```
hosts=local,bind
```

To return to the default behavior, delete the *netsvc.conf* file or reverse the search order it specifies. This file recognizes several name resolution services including bind, local, nis, bind4, bind6, local4, local6, nis4, and nis6. For additional information, see *AIX Commands Reference, Volume 1 - 4,* GC23-2376.

6.4.1.2 Domain Name Service across ATM

Symptom: During boot of system running ATM, the LEDs get stuck on 538 or 581 for about 10 minutes.

Cause: This happens when an ATM interface (such as at0 or at1) or ELAN interface (such as en0 or tr0) is defined as the default interface to the remote DNS nameserver. ATM has a lot of initial setup to do to join the LIS or ELAN (register with the ARP or LAN Emulation Server) so name server contact must be made later in the bringup process than with traditional LAN networks. The LEDs are "stuck" while the attempt to contact the nameserver times out.

This behavior is not an ATM software defect; rather, it is a result of running TCP/IP nameservice across a connection-oriented network. A similar situation occurs with X.25 when IP to NUA translation tables try to build before connection to the nameserver is established.

Workaround: Delay nameserver contact until the system is fully connected to the ATM network.

Edit the /etc/rc.net file, adding two lines with *export* and *unset* of NSORDER to remove actual name server resolution action. This normally runs at config rule time, and the *atmsvcd* daemons are not available yet. Insert the two lines marked with *ADD* around the existing hostid and uname lines:

```
ADD ==> export NSORDER="local"
        /usr/sbin/hostid `hostname` >>$LOGFILE 2>&1
        /bin/uname -S`hostname|sed 's/\..*$//'` >>$LOGFILE 2>&1
ADD ==> unset NSORDER
```

6.4.2 Multiple Classical IP interfaces on one ATM adapter

On most networks, there is a one-to-one correspondence between adapter names and network interface names. For example, Token Ring adapter tok0 corresponds to interface tr0, adapter tok1 corresponds to interface tr1, and so on. Similarly, Ethernet adapter ent0 corresponds to interface en0 (for Ethernet Version 2) and et0 (for IEEE 802.3), and adapter ent1 corresponds to interface en1 and et1.

In the case of ATM, however, an end station with just one adapter can still be part of multiple Logical IP Subnetworks and LAN Emulation ELANs. Because multiple IP interfaces can be associated with a single ATM device (adapter) each interface must be specifically added and a device name must be specifically assigned to it in SMIT.

6.4.2.1 Multiple network interfaces on the same ATM adapter

System steel, in our sample environment, provides an excellent example of the unique relationship of ATM adapters and network interfaces. System steel has two ATM adapters (atm0 and atm1) and is running three IP interfaces (at0, at1, and tr1) atop them.

In Chapter 5, “Configuration and attachment” on page 37 steel's atm0 adapter and the at0 interface are used for a back-to-back TCP/IP PVC connection with system iron. Next, the atm1 adapter and at1 interface are configured to participate in a Classical IP LIS. Although this sample environment only shows two, we could add several more "at" interfaces, each configured independently on atm0 and atm1.

System steel's capabilities are again expanded in Chapter 7, “ATM LAN Emulation” on page 111 when we configure a Token Ring LEC (tok1) and tr1 interface atop the atm1 adapter. We could also configure addition LECs -- both Ethernet and Token Ring -- atop this atm1 adapter. In our environment, LAN Emulation cannot run on the atm0 adapter because it is a back-to-back connection. If it were attached to an ATM switch, however, LECs could be configured on it.

In all cases, the linkage between the interfaces and the real ATM adapter is specified in the SMIT menu, summarized as follows.

In this, the Classical IP menu, the network interface field default values range from at0 through at7. The Alternative Device field values relate to the number of real ATM adapters installed on the system. In the case of system steel in our sample environment, possible values are atm0 and atm1.

```
 Add an ATM Network Interface

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                 <------- System "steel" ------->
* INTERNET ADDRESS (dotted decimal)             [192.168.1.2]      [192.168.2.2]
  Network MASK (hexadecimal or dotted decimal)  [255.255.255.0]    [255.255.255.0]
  Network Interface                             [at0]              [at1]
  Connection Type                                pvc                svc_c
  ATM Server Address                            []
  Alternate Device                              [atm0]             [atm1]
  Idle Timer                                    []
  Best Effort Bit Rate (UBR) in Kbits/sec       []
* ACTIVATE the Interface after Creating it?      yes                yes
```

In these two menus, *tok1* is created on *atm1*, then *tr1* is configured for *tok1*:

```
                              Add a Token Ring ATM LE Client

Type or select values in entry fields.
Press Enter AFTER making all desired changes.
                                                   (Creates tok1 on System "steel")
  Local LE Client's LAN MAC Address (dotted hex)      [0.6.ab.12.34.2]
  Automatic Configuration via LECS                     Yes
    If No, enter the LES ATM Address (dotted hex)     []
    If Yes, enter the LECS ATM Address (dotted hex)   []
  Local ATM Device Name                               [atm1]
  Emulated LAN Type                                    Token Ring
  Maximum Frame Size (bytes)                           Unspecified
  Emulated LAN Name                                   [ELANt1]
  Force Emulated LAN Name                              No
  Enable Forum MPOA and LANE-2 functions               No
  MPOA Primary Auto Configurator                       No
```

```
                             Add a Token-Ring Network Interface

Type or select values in entry fields.
Press Enter AFTER making all desired changes.
                                                    (Creates tr1 on tok1 on "steel")
* INTERNET ADDRESS (dotted decimal)                   [192.168.11.2]
  Network MASK (hexadecimal or dotted decimal)        [255.255.255.0]
  Network Interface                                    tr1
* ACTIVATE the Interface after Creating it?            yes
  Use Address Resolution Protocol (ARP)?               yes
  Enable Hardware LOOPBACK Mode?                       no
  BROADCAST ADDRESS (dotted decimal)                  []
  Confine BROADCAST to LOCAL Token-Ring?               no
```

6.4.2.2 More than eight interfaces

The maximum number of interfaces allowed under AIX for any network type is defined by the ifsize option of the no command. The default is ifsize=8. This means 8 Ethernets, 8 Token Rings, 8 FDDI, 8 ATM, etc. When you change the ifsize you change the maximum number of interfaces allowed for all network types.

IP interfaces configured on ATM LAN Emulation Clients count against the number of interfaces for that LAN network type. For example, if tr0 and tr1 are configured on real Token Ring devices tok0 and tok1, and tr2 is the IP interface on the ATM LAN Emulation Client tok2, three of the default maximum eight Token Ring interfaces have been configured.

With the default, valid ATM interfaces are at0 - at7. AIX supports 256 ATM interfaces on one RS/6000. If you intend to have more than 8 interfaces on an RS/6000, you must change the ifsize. You change the *ifsize* by:

```
no -o ifsize=16
```

In this example, the maximum number of ATM interfaces is set to 16. The number must be equal or greater than the number of interfaces you intend to configure on an ATM adapter. Valid ATM interfaces are now at0 - at15.

The new ifsize value is only valid until the next time the system is booted. After that, the ifsize reverts to the default. To avoid this, add the *no -o ifsize=xxx* in the */etc/rc.net* file before the section of the file that starts the interfaces. Do not add it at the end where the other no commands are shown.

6.4.3 Configuring multiple AIX IP interfaces on the same IP subnet

Occasionally, users ask if they can provide greater availability and performance by adding a second network adapter to a particular machine. For example, they may want to have two Token Ring adapters attached to the same physical network, so they can "double network capacity" or perform a sort of "load balancing" across the two adapters. The implications of this design tactic apply to both ATM and non-ATM environments.

While it is possible to have more than one interface on the same network, this is not generally recommended. So, for the sake of simplicity, we have historically told users that *IP interfaces on the same host SHOULD not belong to the same IP subnet*.

This answer is still correct but, over the years, TCP/IP has been changed and has become more tolerant of such things. As a result, it is more technically accurate today to tell users *It may work if both interfaces are on the same* *physical network, but it probably will not accomplish what you want it to accomplish.*

Why? While AIX TCP/IP SMIT menus allow users to define two interfaces for the same subnet, TCP/IP traffic handling (routing) may not meet the users design needs. For example, users often want to use this for a sort of "load balancing" but it usually does this only halfway. Incoming traffic is generally handled as desired because the interface is specified by the IP address. Outgoing traffic, on the other hand, is NOT balanced because is funneled out one interface -- the interface specified for the local network in the IP routing table. This behavior applies to TCP/IP on all sorts of networks from traditional LANs (Ethernet and Token Ring) to Classical IP.

In the section 6.3.2.3, “Configure the systems for backup ARP service” on page 91 6.3.1 we hinted at the possibility of configuring a second Classical IP interface on system brass. We initially planned to demonstrate brass as a backup server (svc_s) but quickly realized it was not as easy as it appeared on the surface. The client-server relationship of ATM Classical IP adds complications to the scenario that do not exist in other LAN networks. For example, the primary ARP server cannot talk with another server because the secondary does not register with the primary. As a result, primary does not have the secondary's ARP entry so neither the ARP clients nor the primary server can communicate with the secondary.

So, in Figure 19 on page 99, we ended up illustrating a second ARP client LIS connection on system brass.

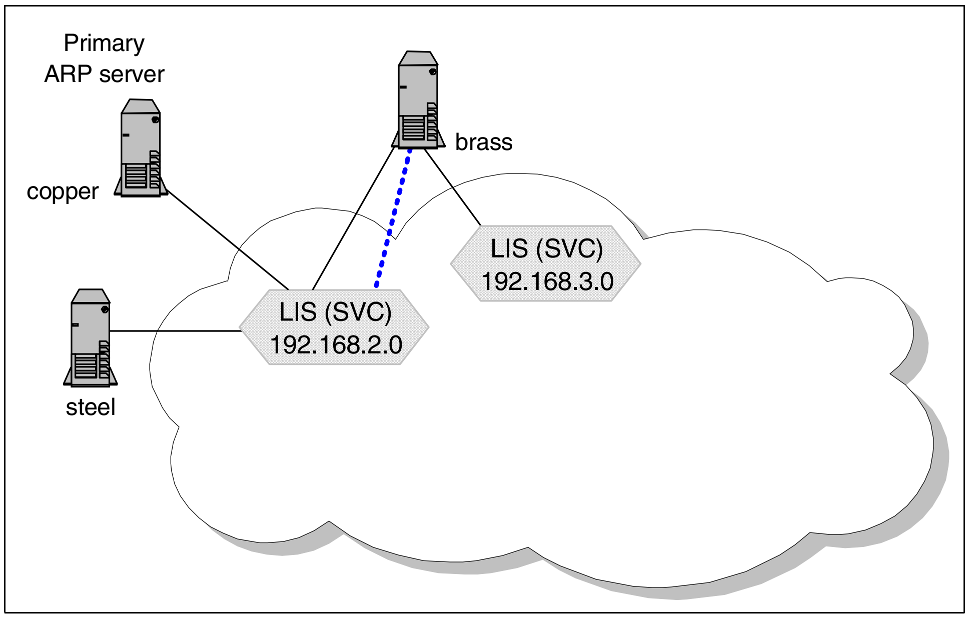


Figure 19. Multiple IP interfaces on the same ATM adapter

In the figure, system copper is the primary ARP server and systems brass and steel have ARP client interfaces. The pertinent information on these interfaces follows:

System	Type	Adapter	Interface	IP Address	MAC/selector byte
brass	svc_c	atm0	at0	192.168.2.5	0.6.ab.12.34.5.0
	svc_c	atm0	at2	192.168.2.9	0.6.ab.12.34.5.2
copper	svc_s	atm0	at0	192.168.2.3	0.6.ab.12.34.3.0
steel	svc_c	atm1	at1	192.168.2.2	0.6.ab.12.34.2.1

System brass actually has a third interface, at1 (192.168.3.5) on atm0, which is part of the 192.168.3.0 LIS. To simplify the example, the command output, provided for reference at the end of this section, will only show information pertaining to the 192.168.2.0 LIS.

On systems with two interfaces on one IP subnetwork (like brass) outgoing traffic goes out only one of the two interfaces. Why? Because traffic for the local network is addressed using the ARP cache information and routed through the local interface route (marked with a "U") in the routing table. In the command output, notice the arp cache entry for both interfaces on system brass. The system names like *brass* and *brass_again* come from the /etc/hosts files each system.

This IP routing behavior applies to ATM and non-ATM networks. In the case of system brass, at0 and at2 actually share a single adapter so the attempt to "load balance across adapters" doesn't apply as it would on a system configured with two LAN adapters of the same type.

A word of caution

Do not assume this will work in all environments. Some TCP stacks are picky about the socket pairs matching and will misinterpret the connection.

Here is the command output showing the routing table and arp caches in the sample environment pictured in Figure 19 on page 99:

On system brass:

```
netstat -in:
Name  Mtu   Network     Address          Ipkts Ierrs Opkts Oerrs Coll
lo0   16896 link#1                         240     0   240     0    0
lo0   16896 127         127.0.0.1          340     0   340     0    0
at0   9180  link#2      0.6.ab.12.34.5     158     0   195     0    0
at0   9180  192.168.2   192.168.2.5        158     0   195     0    0
at2   9180  link#3      0.6.ab.12.34.5      65     0    23     0    0
at2   9180  192.168.2   192.168.2.9         65     0    23     0    0
```

```
netstat -rn:
Routing tables
Destination      Gateway        Flags   Refs   Use   If  PMTU  Exp Groups
Netmasks:(0) 0 ff00
(0) 0 ffff ff00
Route Tree for Protocol Family 2:
default          192.168.2.2       UG    2     251   at0   -   -
127              127.0.0.1         U     0       0   lo0   -   -
192.168.2/24     192.168.2.5       U     2     292   at0   -   -

arp -t atm -a:
SVC - at0 on device atm0 -
==========================
at0(192.168.2.5)    47.0.5.80.ff.e1.0.0.0.f2.15.16.45.0.6.ab.12.34.5.0
IP Addr                  VPI:VCI Handle ATM Address
  copper(192.168.2.3)       0:103    7
47.0.5.80.ff.e1.0.0.0.f2.15.16.45.0.6.ab.12.34.3.0
  steel(192.168.2.2)        0:102    4
47.0.5.80.ff.e1.0.0.0.f2.15.16.45.0.6.ab.12.34.2.1

SVC - at2 on device atm0 -
==========================
at2(192.168.2.9)    47.0.5.80.ff.e1.0.0.0.f2.15.16.45.0.6.ab.12.34.9.2
IP Addr                  VPI:VCI Handle ATM Address
  copper(192.168.2.3)       0:384    5
47.0.5.80.ff.e1.0.0.0.f2.15.16.45.0.6.ab.12.34.3.0
  steel(192.168.2.2)        0:379    6
47.0.5.80.ff.e1.0.0.0.f2.15.16.45.0.6.ab.12.34.2.1
```

On system copper (the ARP server):

```
arp -t atm -a:
SVC - at0 on device atm0 -
==========================
at0(192.168.2.3)  47.0.5.80.ff.e1.0.0.0.f2.15.16.45.0.6.ab.12.34.3.0
IP Addr                  VPI:VCI Handle ATM Address
  steel(192.168.2.2)        0:94    11
47.0.80.ff.e1.0.0.0.f2.15.16.45.0.6.ab.12.34.2.1
  brass(192.168.2.5)        0:97    12
47.0.5.80.ff.e1.0.0.0.f2.15.16.45.0.6.ab.12.34.5.0
  brass_again(192.168.2.9)  0:95     8
47.0.5.80.ff.e1.0.0.0.f2.15.16.45.0.6.ab.12.34.9.2
```

On system steel:

```
arp -t atm -a:
VC - at1 on device atm1 -
```

```
========================
at1(192.168.2.2)  47.0.5.80.ff.e1.0.0.0.f2.15.16.45.0.6.ab.12.34.2.1
IP Addr                  VPI:VCI Handle ATM Address
  brass(192.168.2.5)        0:166    4
47.0.5.80.ff.e1.0.0.0.f2.15.16.45.0.6.ab.12.34.5.0
  copper(192.168.2.3)       0:124    3
47.0.5.80.ff.e1.0.0.0.f2.15.16.45.0.6.ab.12.34.3.0
  brass_again(192.168.3.9)  0:167    5
47.0.5.80.ff.e1.0.0.0.f2.15.16.45.0.6.ab.12.34.9.2
```

6.4.4 IP routing in the sample environment

All of the systems in the sample network are running TCP/IP. Those networks configured in this chapter are running Classical IP and ones which will be set up in the Chapter 7, “ATM LAN Emulation” on page 111 are running TCP/IP atop LAN Emulation. Interoperability between these separate networks requires two key, properly positioned and configured routing components:

- IP routers
- IP hosts

As in all TCP/IP environments, these separate IP subnetworks are linked via IP routers. In large networked environments, dedicated IP routers may provide this function. In our environment, we've used the IBM 8210 MSS device and systems steel and brass which are multi-homed RS/6000 hosts that participate in two or more networks to provide the linkage. These hosts are configured to forward traffic as described in section 6.2.3, “Test the connection” on page 85.

Figure 20 on page 103 shows a way to set up the routing in our sample network. (The ATM configuration is identical to the one described in Chapter 4, “Sample Network” on page 33). IP routing tables on both routers and hosts must be configured to pass traffic to the next hop along the path to the destination system. For example, system gold cannot communicate with system copper directly; it must pass the datagrams to MSS2 who must know how to pass it to brass who will then transmit the datagrams on the 192.168.2.0 network.

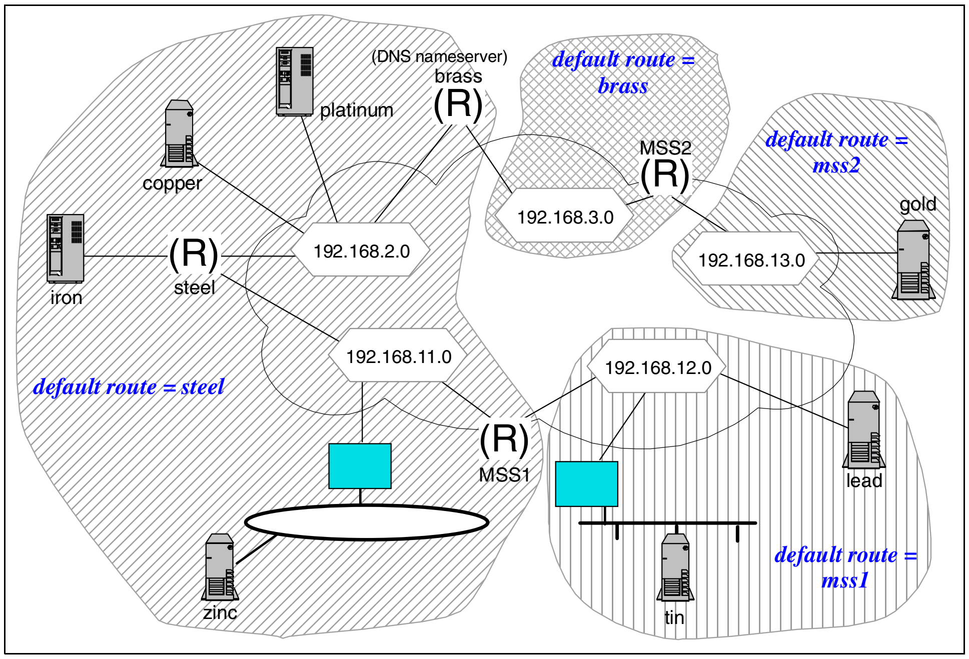


Figure 20. IP routing in a Classical IP environment

Because system steel has a direct interface on three networks, all of the systems on those networks specify system steel as the default route. System gold specifies its default route through MSS2 which, in turn, default routes through system brass. Systems lead and tin, point to MSS1 as the default route. Here is a summary of the routing set up at the RS/6000s. The network routes are for those situations where a system cannot be reached via the default route.

System	Route Type	Gateway (or Router)	Destination Address
iron	default	steel (192.168.1.2)	
steel	default	MSS1	
	net	brass (192.168.2.5)	192.168.3.0
	net	brass (192.168.2.5)	192.168.13.0
copper/platinum	default	steel (192.168.2.2)	
	net	brass (192.168.2.5)	192.168.3.0
	net	brass (192.168.2.5)	192.168.13.0
brass	default	steel (192.168.2.2)	
	net	MSS2	192.168.13.0
gold	default	MSS2	
zinc	default	steel (192.168.11.2)	
	net	MSS1	192.168.12.0
lead/tin	default	MSS1	

Routes to directly attached networks will be added automatically when you configure the network interface. In the sample network, we used netmask 255.255.255.0.

Note

Since Classical IP over ATM as defined in RFCs 1577 and 2225 do not support protocols that resolve addresses using broadcast, routed and gated will not work across the ATM subnet. Static network routes must be

The default route is generally set in the *Minimum Configuration & Startup* menu or via the route command. There is only one default route per host. Here is how either system platinum or copper might set its default route to steel. Remember that steel has three IP addresses, so be sure to specify the IP address (or symbolic name) that matches the subnetwork of your system. Follow this menu progression.

```
smit
  Communications Applications and Services
    TCP/IP
      Minimum Configuration & Startup
```

or use the fastpath: `smit mktcpip`

```
                         Minimum Configuration & Startup

  To Delete existing configuration data, please use Further Configuration menus

 Type or select values in entry fields.
 Press Enter AFTER making all desired changes.

 [TOP]                                                      [Entry Fields]
 * HOSTNAME                                              [copper]
 * Internet ADDRESS (dotted decimal)                     [192.168.2.3]
   Network MASK (dotted decimal)                         [255.255.255.0]
 * Network INTERFACE                                      at0
   NAMESERVER
            Internet ADDRESS (dotted decimal)            []
            DOMAIN Name                                  []
   Default GATEWAY Address                               [192.168.2.2]
           (dotted decimal or symbolic name)
   Connection Type                                        svc_c              +
   ATM Server Address                                    [39.99.99.99.99.99.99.0>
   Alternate Device                                      []
   Idle Timer                                            [60]
   Best Effort Bit Rate (UBR) in Kbits/sec               [0]
   START TCP/IP daemons Now                               no                 +
 [BOTTOM]

 F1=Help             F2=Refresh          F3=Cancel           F4=List
 F5=Reset            F6=Command          F7=Edit             F8=Image
 F9=Shell            F10=Exit            Enter=Do
```

To set up one of the network routes on system platinum or copper, follow this menu progression

```
smit
  Communications Applications and Services
    TCP/IP
      Further Configuration
        Static Routes
          Add Static Route
```

or use the fastpath: `smit mkroute`

```
 Add Static Route

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
  Destination TYPE                                   net
* DESTINATION Address                               [192.168.3]
  (dotted decimal or symbolic name)
* Default GATEWAY Address                           [192.168.2.5]
  (dotted decimal or symbolic name)
* METRIC (number of hops to destination gateway)    [1]
  Network MASK (hexadecimal or dotted decimal)      [255.255.255.0]
```

To verify the routes, enter the following command on each system:

```
netstat -rn
```

The routing table on system copper should look like the following:

```
Routing tables
Destination      Gateway       Flags  Refs     Use   If   PMTU  Exp  Groups

Route Tree for Protocol Family 2 (Internet):
default          192.168.2.2   UG       2     1081   at0     -    -
127/8            127.0.0.1     U        5      276   lo0     -    -
192.168.3/24     192.168.2.5   U        1      119   at0     -    -
192.168.13/24    192.168.2.5   U        0       56   at0     -    -
```

6.4.5 Configuring the TCP/IP MTU size

A TCP/IP datagram or packet (Maximum Transmission Unit or MTU) size is set for each IP interface. Minimum, maximum and default values vary depending upon the network type. In the case of ATM, the default is 9180, but possible values range from 1 to 64K (65,536) bytes.

If you are setting up a Micro Channel bus RS/6000 for ATM, it is a good idea to try to match the MTU size with the target system. For additional information see 6.4.6, “TCP/IP MTU and ATM PDU relationship” on page 107.

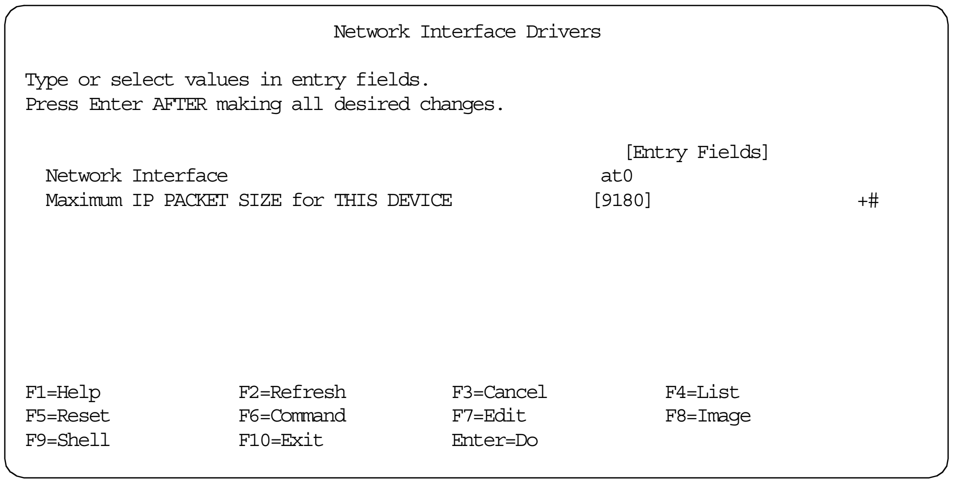
To determine the MTU size, issue the following command:

```
netstat -in
Name  Mtu   Network     Address            Ipkts  Ierrs  Opkts   Oerrs  Coll
lo0   16896 link#1                          1818      0   1818      0     0
lo0   16896 127         127.0.0.1           1818      0   1818      0     0
lo0   16896 ::1                             1818      0   1818      0     0
tr0   1492  link#2      10.0.5a.a8.c7.5d   33027      0   3815      0     0
tr0   1492  9.19.169    9.19.169.34        33027      0   3815      0     0
at0   9180  link#3      0.6.ab.12.33.2      2450      0   2450      0     0
at0   9180  192.168.1   192.168.1.1         2450      0   2450      0     0
```

To change the MTU size for an IP interface, use the following SMIT menu progression.

```
smitty
  Communications Applications and Services
    TCP/IP
      Further Configuration
        Network Interfaces
          Network Interface Drivers
```

or use the fastpath: `smit chif`



```
                              Network Interface Drivers

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                          [Entry Fields]
  Network Interface                                      at0
  Maximum IP PACKET SIZE for THIS DEVICE                [9180]                 +#






F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

6.4.6 TCP/IP MTU and ATM PDU relationship

Question often asked are:

- When I increase the ATM Classical IP *MAX IP PACKET SIZE* in SMIT, is the *ATM ADAPTER PDU SIZE* increased accordingly?

- Our SPs seem to randomly lose the connection to the ARP server (which is also an SP system). The connection is only restored by disabling/enabling the 8265 ports to which the SPs are connected. It appears the user increased the max IP packet size from 9180 to an advised optimum of 60416, but did not change the ATM adapter PDU size from the 9188 default.

  Is this a valid configuration?

The answer to these questions is:

Development says your configuration is invalid and it is quite likely causing your seemingly "random" problems on your Micro Channel bus SP systems. Here are the details.

The *MAX IP PACKET SIZE* in SMIT is the familiar TCP/IP MTU size for the IP interface, so I'll use the term MTU in this explanation. The user sets the MTU and PDU values in SMIT -- there is no internal logic to automatically detect and scale the values based upon user changes -- whatever values are entered in SMIT are the values used.

On PCI bus ATM Adapters

There is no adapter max PDU setting, so you need not worry about the balancing the IP MTU and adapter PDU values.

On Micro Channel bus ATM Adapters

The adapter PDU size should be at least 8 bytes larger than the IP MTU. (Your configuration is invalid because the MTU is larger than the PDU.) Ideally then, the PDU is the MTU value + 8. The AIX V 4.2.1 system defaults reflect this ideal match in that the MTU is 9180 and the PDU is 9188.

So what happens in your situation where the MTU is set larger than the PDU? Those datagrams that fit into the PDU will most likely succeed while those surpassing the PDU size will fail. Pings and most telnets would probably succeed. However, this isn't 100% predictable because there are things going on under the covers trying to create full datagrams. The Nagle Algorithm RFC 896, for example, tries to create full datagrams by concatenating small datagrams.

The PDU value translates to adapter memory reserved for buffer space. Running a PDU size that is excessively larger than the MTU works but it is a waste of adapter memory. Remember, however, that you can configure multiple IP interfaces on a single ATM adapter so, if you're running different

MTU sizes for each interface, make sure the PDU setting is at least 8 bytes larger than the largest MTU to be run on that adapter.

Chapter 7. ATM LAN Emulation

ATM LAN Emulation (LANE) provides a way for Ethernet and Token Ring LAN MAC-layer protocols and their higher-layer protocols and applications to transparently communicate with devices across an ATM network. AIX ATM LAN Emulation Clients emulate real Ethernet (*ent*) and Token Ring (*tok*) devices; so, legacy Ethernet and Token Ring applications continue to run unchanged and are configured as though ATM were not involved. It is intended to give workstations a path for gradual migration to ATM while protecting current investments in Token Ring and Ethernet LAN hardware and applications.

The challenge of emulating traditional LAN operations stems from the inherent differences in the way ATM and legacy LANs work. ATM is a connection-oriented networking method. A call to a specific ATM address goes out across the network. Once the ATM connection between two systems has been established, communications are, essentially, point-to-point. Frame Relay, X.25, and point-to-point connections, such as SLIP and PPP, are also connection-oriented networking methods.

Legacy LANs, on the other hand, operate in a connectionless fashion. Each data frame contains the source and destination hardware (MAC) address for independent delivery across the LAN. All attached stations monitor the traffic on the network. When a station recognizes its own MAC address in the destination field, it copies the transmission from the LAN. With LANs, all stations hear the transmissions, while with ATM, only two stations do.

LAN Emulation software mimics Ethernet and Token Ring operation, therefore, allowing applications written for these LANs to operate across an ATM network. This means the LAN Emulation software exists between the application and the ATM network. The application thinks it is talking to a real Ethernet or Token Ring LAN; so, LAN Emulation software must exhibit the behavior of the real LAN. It must also convert the data to a form acceptable for delivery across an ATM network. The result is a logical, emulated LAN (ELAN) that is software-defined and independent of the physical network setup.

Figure 21 on page 112 provides a conceptual view of ATM LAN Emulation. Two components make up an individual Token Ring or Ethernet ELAN:

- LAN Emulation Clients (such as the RS/6000)
- LAN Emulation Services (provided within the network)

A proxy LAN Emulation Client (a feature of some LAN bridges) provides ATM network access and interoperability for legacy LAN-attached devices. It provides a LAN Emulation Client presence for devices that have no knowledge of LANE or of ATM network.

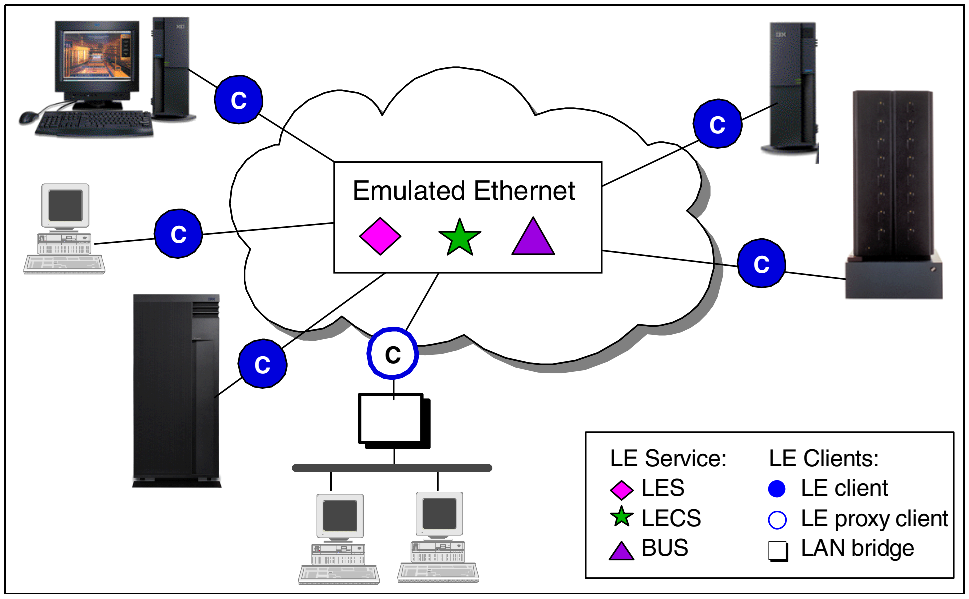


Figure 21. Conceptual view of ATM LAN Emulation

AIX and LAN Emulation

ATM Forum compliant ATM LAN Emulation V1.0 Client software is included in the base operating system beginning with AIX V 4.2.1. ATM LAN Emulation Service functions are not a part of the AIX software and must be provided elsewhere in the network. These services are composed of three servers that work together but need not exist in the same place within the network:

- **LAN Emulation Server** (LES) is a required service that handles registration and resolves MAC and routing addresses to ATM addresses.
- **Broadcast and Unknown Server** (BUS) is a required service that handles all broadcast and multicast requests.
- **LAN Emulation Configuration Server** (LECS) is an optional service. It can automatically configure LAN Emulation Clients. For example, this server can provide parameters, such as the LAN Emulation Server's ATM address. With LECS, network administrators centrally configure and maintain a pool of configuration parameters that are issued upon request to clients systems joining the ELAN. This means that parameters need not

be hard-coded for every potential client station, therefore, making it particularly useful in the proxy LE client environments. Due to its convenience and flexibility, a LAN Emulation Configuration Server is recommended.

In summary, AIX ATM LAN Emulation:

- Hides ATM from the upper-level application; the application creates LAN frames that are then split into ATM cells. This frame conversion may occur at an end station or a network component, such as an ATM LAN bridge.
- Provides a migration and coexistence path; LAN applications run unchanged enabling protocols, such as SNA, IPX, and NetBios.
- Uses a server (the LES) to maintain a dynamic table linking 20-byte ATM and MAC addresses.
- Supports SVC operations only and requires an ATM switch network connection.
- Emulates Ethernet and Token Ring networks only.

Application protocols

Once a system joins an ELAN, protocols, such as TCP/IP, SNA, and IPX, will be able to run transparently across ATM networks. Users configure the application in SMIT just as if the LANE Emulation Client is a real Ethernet or Token Ring LAN interface. Although LAN Emulation introduces overhead, it enables a large number of applications already running on workstations and provides an excellent migration path to ATM networking. With LAN Emulation, these applications run, without change, across an ATM network providing users a means of coexistence during the migration to ATM networking. LAN Emulation may run as software in an end station, or it may run in a network component, such as a bridge, converting Ethernet and Token Ring frames to ATM.

Communicating with real LANs

Systems attached to real Token Ring or Ethernet LANs are unaware of the ATM network but can be linked into the ATM ELAN via network devices, such as an IBM 2216 ATM LAN bridge. The proxy LAN Emulation Client is the key component of such devices. It provides a LAN Emulation Client presence for legacy LAN-attached devices that have no knowledge of LAN Emulation for the ATM network. The scenarios in this chapter will show such interoperability with LAN-attached systems *zin*c and *tin*.

Interoperability with Classical IP environments

TCP/IP traffic is linked between ELANs and Classical IP environments via an IP router. MPOA will be introduced in 7.3, “MPOA scenario” on page 132 as a way to improve performance in a routed TCP/IP environment. It is important to remember that all virtual LANs (ATM VLANs are Classical IP LISs and LAN Emulation ELANs) operate independently of each other; so, interoperability is handled in the same way as it would if they were different, physical networks.

Virtual Channel utilization

The general ATM adapter VC utilization (two PVCs per ATM adapter) was described in Chapter 6, “Classical IP over ATM” on page 71. LAN Emulation requires four VCs per LE Client. It generally takes one regular VC and one point-to-multipoint VC to maintain connections to the LES and to the BUS, and one VC is needed for each connection to another ELAN member, but allocate a few extra VCs because, under some circumstances, a second VC per connection may be used for a short period of time.

Allow up to twelve VCs for a system with one ATM adapter, an Ethernet and Token Ring ELAN, and one active connection to another ELAN member. Here is the breakdown:

2	Connection to the switch (two PVCs per ATM adapter)
8	Four per LAN Emulation Client (two for the LES connection and two for the BUS connection)
1 - 2	One (sometimes two) for each active connection to another ELAN member
____	
11 - 12	Total VCs in use

The switch, LES, and BUS connections are always maintained; so, in this example, ten VCs would always be tied up. Therefore, any variation in the number of VCs in use will come from changes in currently active connections to other ELAN members as VCs are opened, closed, and reused.

The LAN Emulation ARP (LE_ARP) cache

Most AIXers are very familiar with the TCP/IP arp cache, which is used to address IP datagrams for delivery to systems on the local IP subnet. This arp table maps IP and MAC (hardware) addresses for systems on the local IP subnetwork and is displayed and manipulated using the `arp` command. On an Ethernet-attached RS/6000, for example, the local ARP cache contains IP-to-MAC mappings for host nodes and routers on the local Ethernet IP subnetwork. With Classical IP, the 20-byte ATM address is used to deliver information within the LIS; so, the arp cache maps IP and ATM (rather than MAC) addresses. As mentioned previously, the Classical IP entries in the ARP cache are displayed and maintained using the `arp -t atm -a` command.

LAN Emulation hides the ATM network from the upper-level TCP/IP protocol; so, the TCP/IP arp cache contains the same sort of IP-to-MAC address mappings appropriate for delivery on a real Ethernet or Token Ring LAN segment. Therefore, another ARP cache, the LE_ARP cache, located on the LAN Emulation Server (LES), provides the further mapping of MAC to 20-byte ATM addresses. Systems joining the ELAN register with this server. It, in turn, provides MAC-to-ATM address information for those LECs contacting other ELAN members. Each LEC builds and maintains its own LE_ARP cache of the current BUS address and any remote clients it contacts.

The mechanics of two ARP caches are quite similar. AIX TCP/IP systems maintain a local ARP cache, and entries are posted and expire or are deleted based upon the table size and other system configuration parameters. The LE_ARP cache works in much the same way. An LE_ARP entry for a recently active connection is kept on a RS/6000 LE client until it either expires due to non-use or is purged to free ARP cache table space for new connections.

The LE client SMIT or *chdev* values (arp_aging, ARP_cache_size, ARP_response_timeout, etc) relate to the local LE_ARP cache. (Section 7.5, “Configuration recommendations” on page 153 provides a complete listing of parameters). Use them to set the maximum number of table entries, the non-use timeout, and so forth. By default, the table holds 32 entries. The arrival of entry number 33 triggers the deletion of the entry that has been unused for the longest time. Therefore, if a user regularly communicates with 50 systems, it would be wise to increase the *arp_cache_size* so that the information need not be continually relearned. At this time, there is no user tool for displaying or manipulating LE_ARP cache information.

7.1 Token Ring ELAN scenario

The scenario in Figure 22 on page 116 adds yet another ATM function to system steel. In Chapter 6, “Classical IP over ATM” on page 71, it had a back-to-back PVC connection to system iron and participated as an ARP client in an SVC LIS environment. System steel already has a real Token Ring network adapter; so, the ATM Emulated LAN (ELAN) Client will be tok1 on atm1.

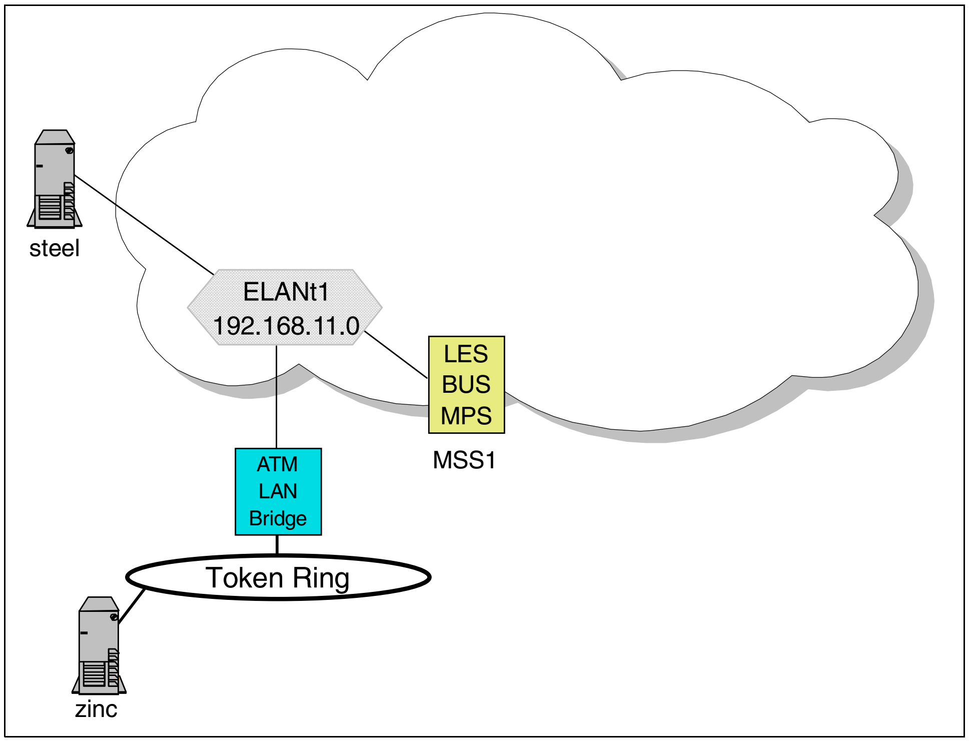


Figure 22. Token Ring LAN Emulation scenario

7.1.1 Token Ring LAN Emulation overview

In this scenario, system steel will join a Token Ring emulated LAN we named ELANt1. System steel's second ATM adapter (atm1) which was used in the Classical IP SVC scenarios, will also be used in this scenario.

The IBM 8210 Nways Multiprotocol Switched Services (MSS) server is a multifunction ATM network device we have named MSS1. It will provide ATM LAN Emulation Services (LES, BUS, and LECS) for this ELAN. Later, it will also act as an IP router linking IP traffic with the Ethernet ELAN scenario.

Once steel has joined the ELAN, and an IP interface has been configured, it will be able to communicate with other ELAN members running TCP/IP. In our simple environment, steel can communicate with system *zinc* because it runs TCP/IP, and the intervening ATM LAN bridge provides proxy LEC services for it. Just as in a real LAN, systems can only talk if they are running matching protocols. So, if zinc runs SNA or IPX, then steel must run the same protocol.

7.1.2 SMIT configuration for Token Ring

We will configure system steel's LAN Emulation Client for the Token Ring ELAN following the naming and addressing conventions outlined in Chapter 4, “Sample Network” on page 33. The Token Ring LAN Emulation Client on system steel is tok1, configured on atm1. The IP interface, configured on the tok1 LAN client, is tr1.

We assume the atm1 adapter is *Available* and properly connected to the ATM network as described in Chapter 5, “Configuration and attachment” on page 37.

Make sure the LAN Emulation Client (LEC) software (fileset bos.atm.atmle) is installed on your system. Installing the latest available updates for LANE is also recommended. Check for the LANE software using the following command:

```
# lslpp -l bos.atm.atmle
```

If it is not installed, the output is:

```
lslpp: 0504-132  Fileset bos.atm.atmle not installed.
```

If it is installed, the output will be similar to the following:

```
Fileset                   Level    State       Description
------------------------------------------------------------------------
Path: /usr/lib/objrepos
  bos.atm.atmle    4.3.3.2  COMMITTED  ATM LAN Emulation Client Support
Path: /etc/objrepos
  bos.atm.atmle    4.3.3.0  COMMITTED  ATM LAN Emulation Client Support
```

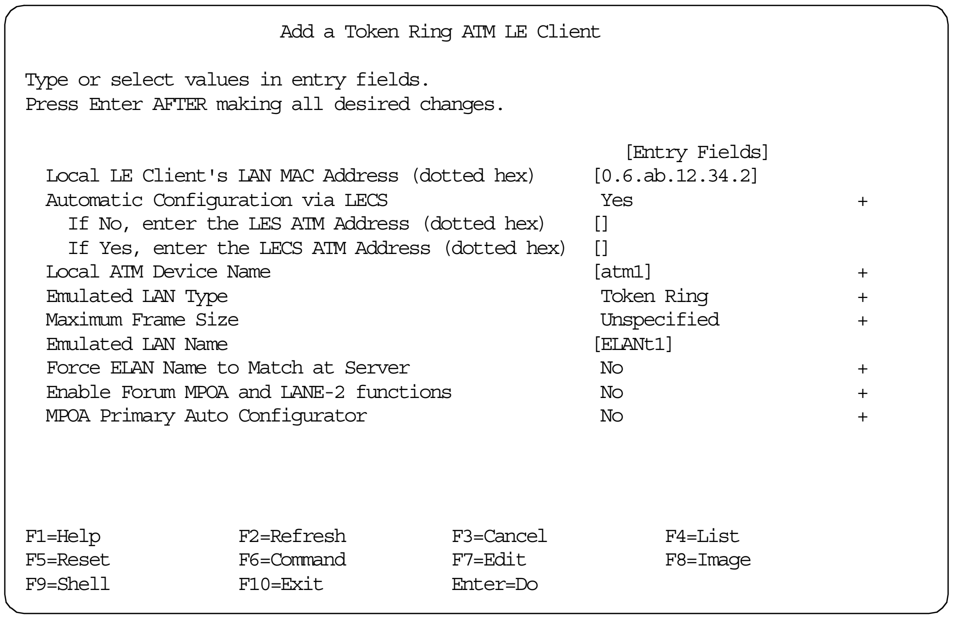
The numbers to the right of the fileset name are the fileset level. If the level numbers are different, your software level is the higher of the two. For instance, this command output shows 4.3.3.2 for the usr part and 4.3.3.0 for the root part; so, the system is running level 4.3.3.2.

7.1.2.1 Add a Token Ring LAN Emulation Client

Add the LAN Emulation Client (*tok1*) using the following menu progression:

```
smit
  Devices
    Communication
      ATM Adapter
        Services
          ATM LAN Emulation
            Add an ATM LE Client
              Add a Token Ring ATM LE Client
```

or use the fastpath: `smit atmle_panel`



```
                         Add a Token Ring ATM LE Client

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                        [Entry Fields]
  Local LE Client's LAN MAC Address (dotted hex)      [0.6.ab.12.34.2]
  Automatic Configuration via LECS                     Yes                    +
    If No, enter the LES ATM Address (dotted hex)     []
    If Yes, enter the LECS ATM Address (dotted hex)   []
  Local ATM Device Name                               [atm1]                  +
  Emulated LAN Type                                    Token Ring             +
  Maximum Frame Size                                   Unspecified            +
  Emulated LAN Name                                   [ELANt1]
  Force ELAN Name to Match at Server                   No                     +
  Enable Forum MPOA and LANE-2 functions               No                     +
  MPOA Primary Auto Configurator                       No                     +




F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

The following text describes each changeable parameter in more detail:

Local LE Client's LAN MAC	The unicast LAN MAC address of this system in the Emulated Ethernet or Token Ring LAN environment. It represents this LE Client and will be registered with the LE server (ATM Forum LE Client parameter C6). This 6-byte address must be entered as hexadecimal numbers using a . (period) as the delimiter between bytes. Leading zeros of each byte may be omitted. Examples of valid MAC addresses include: 2.60.8C.2C.D2.DC Ethernet 10.0.5A.4F.4B.C4 Token Ring Note: Just as duplicate MAC addresses are not allowed on real LANs, no two LE clients can have the same MAC address in the same ELAN. An entry is required in this field.

Automatic Config via LECS	Specify Yes (automatic configuration) to receive configuration information from the LAN Emulation Configuration Server (LECS) located in device MSS1. The default value is No (manual configuration).

= No (default)
The LE client's configuration parameters (including the ATM address) will be manually configured in the SMIT menus. Set this to No, and then specify the 20-byte ATM address of the LAN Emulation Server (LES) in the LES ATM Address field.

= Yes
Select Yes if the LAN Emulation Configuration Server (LECS) will provide configuration settings for this LE client. Parameters include the LE Server ATM address as well as any additional configuration parameters provided by the LECS. A setting of Yes means the ATM address of the LECS can be discovered automatically. Discovery is attempted, until successful, in the following standardized hierarchy:

- LECS 20-byte ATM address (from this SMIT menu)
- ILMI Discovery (a Get of the service registry returns a list of potentials that AIX tries in order) on AIX V4.3.2 or higher systems
- ATM Forum's new style and original well-known LECS addresses:
  C5.00.79.00.00.00.00.00.00.00.00.00.00.
  A0.3E.00.00.01.00
  (only if MPOA and LANE-2 enabled)
  47.00.79.00.00.00.00.00.00.00.00.00.00.
  A0.3E.00.00.01.00
- ATM Forum's well-known PVC address:
  decimal VPI=0,VCI=17 (which is VPI=0, VCI=11 in hex)

The LECS values override any configuration values specified by the user.

LES ATM Address	Leave this field empty for this scenario. If doing manual LE client configuration (without the aid of an LECS) use this field for the ATM address of the LE ARP server (ATM Forum LE Client parameter C9). This 20-byte address must be entered as hexadecimal numbers using a . (period) as the delimiter between bytes. Leading zeros of each byte may be omitted. For example, 39.11.ff.22.99.99.99.0.0.0.0.1.49.10.0.5a.68.0.a.1 If the "Automatic Configuration via LECS" field is set to No, you must specify the ATM address of the LES. If it was set to Yes, it may be left blank because the LES ATM address will be obtained from the LECS.
LECS ATM Address	Leave this field blank for this scenario. Used to specify a specific 20-byte ATM address of the LECS on the network.
Local ATM Device Name	Specify `atm1` for this scenario. This is the ATM adapter for this LE Client. There are defaults, but we recommend making a specific entry.
Emulated LAN Type	This value is set based upon the client type being added and cannot be changed in SMIT. Values are Ethernet/IEEE 802.3 for Ethernet LE clients and Token Ring for Token Ring LE clients.
Maximum Frame Size	Leave this at Unspecified, the default, for this scenario. Users can set one of the predefined frame sizes or choose "Unspecified" meaning the LECS will assign the maximum frame Size for this client. Valid values are as follows: Unspecified (for either Ethernet or Token Ring clients) 1516 (for Ethernet) 4544 (for 4 Mbps Token Rings) 18190 (for 16 Mbps Token Rings)

Emulated LAN Name	Specify ELANt1 for this scenario. This is the network administrator-defined name of the ELAN. If the ELAN name can be obtained from the LECS automatically, this field may be left blank. Otherwise, you must enter in the ELAN name to which this LE client will belong. The ELAN name can be obtained from your network administrator. The ELAN name is an SNMPv2 string of 1-32 characters, and blanks can be inserted within the name by keying a "~" character wherever a blank character is desired.
Force Emulated LAN Name	Leave this field at the default (No) for this scenario. This field determines whether AIX checks to ensure the ELAN name sent in a join request matches the one returned in the join response. If this option is set to Yes and the names do not match, the LE client will not configure -- the driver enters the limbo state and moves to the next LECS address in the list to try again. The name match must be exact and is case sensitive. The default is No, meaning the LE Client will configure even if the name returned does not match that in the request.
Enable Forum MPOA and LANE-2 functions	Leave this field at the default (No) for this scenario. Select Yes if this LE client will be using MPOA. The default is No, meaning this LE client will NOT be using MPOA.
MPOA Primary Auto Configurator	Leave this field at the default for this scenario. Select Yes if the LE client will get MPOA configuration information from the LECS. Select No if the LE client will NOT get MPOA configuration information from the LECS. To use this option, the Enable Forum MPOA and LANE-2 functions field must be set to Yes.

Entries are required for the local LE client's LAN MAC address and possibly the LES ATM address or LECS ATM address depending on the support provided at the server. If the server accepts the "well-known ATM address" for LECS or registers its LECS address with ILMI, then automatic configuration via LECS can be set to "Yes," and the LES and LECS ATM address fields are left blank. If the server does not support the "well-known ATM address" for LECS, or does not register with ILMI, an ATM address must be entered for either LES (manual configuration) or LECS (automatic configuration). All other configuration attribute values are optional.

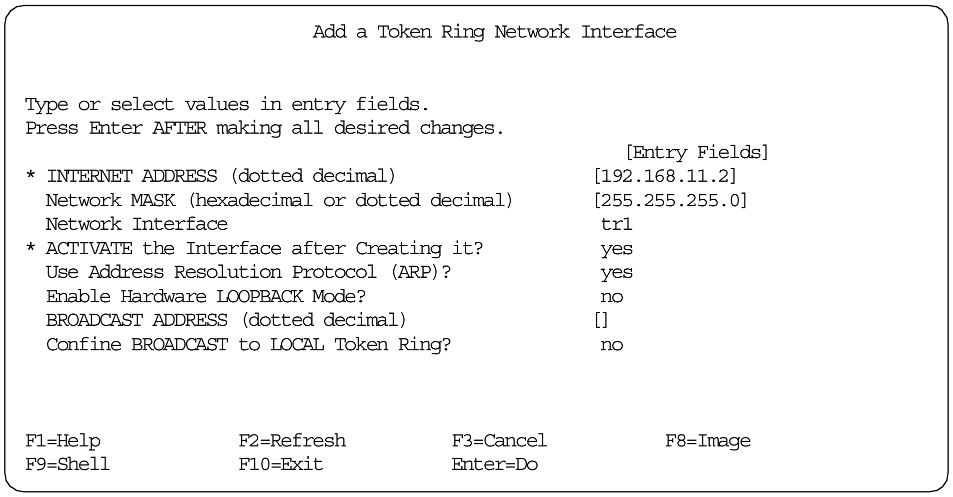
Additional parameters are listed and explained in the Change/Show an ATM LE Client SMIT screen (see 7.3.2, “SMIT configuration for a LAN Emulation Client” on page 141) and in 7.4.2, “ELAN naming tips” on page 149 and 7.4.3, “Additional LAN Emulation device driver parameters” on page 150.

7.1.2.2 Configure the upper layer application

Once the LE client has been created, the user must configure the desired upper layer application. In this case, add the Token Ring IP interface (*tr1*), which will run atop the LAN Emulation Client (*tok1*) using the following menu progression. The menu should be familiar, as it is the same one used to configure a real Token Ring LAN attachment:

```
smit
  Communications Applications and Services
    TCP/IP
      Further Configuration
        Network Interfaces
          Network Interface Selection
            Add a Network Interface
              Add a Token Ring Network Interface
```

or use the fastpath: `smit mkinet`



```
                        Add a Token Ring Network Interface


Type or select values in entry fields.
Press Enter AFTER making all desired changes.
                                                         [Entry Fields]
* INTERNET ADDRESS (dotted decimal)                   [192.168.11.2]
  Network MASK (hexadecimal or dotted decimal)        [255.255.255.0]
  Network Interface                                    tr1
* ACTIVATE the Interface after Creating it?            yes
  Use Address Resolution Protocol (ARP)?               yes
  Enable Hardware LOOPBACK Mode?                       no
  BROADCAST ADDRESS (dotted decimal)                  []
  Confine BROADCAST to LOCAL Token Ring?               no



F1=Help             F2=Refresh          F3=Cancel           F8=Image
F9=Shell            F10=Exit            Enter=Do
```

7.1.3 Test the Token Ring connection

System steel should now be able to communicate with other systems configured for TCP/IP on ELANt1; so, try the *ping* and *telnet* tests described in the Classical IP section.

However, even if you cannot test with TCP/IP commands (perhaps because no other TCP/IP systems are immediately known), you can verify that steel has successfully joined the ELAN using the `tokstat -d tok1` command. Check the driver flags under the *General Statistics* section of the command output. A successful ELAN join will show:

```
Driver Flags: Up Broadcast Running
```

The driver flags are particularly informative:

- Up — The LE client has been opened.
- Running — The LE client is connected with the LAN Emulation Servers and is operational.
- Limbo — Network recovery mode. The LE client lost contact with one or more LAN Emulation Servers.
- Dead — A hard failure has occurred. The LE client is no longer operational.

Status Information:

The following `lsdev` command output provides a handy snapshot summary of ATM (adapters, Classical IP, LAN Emulation, and TCP/IP interfaces) on system steel.

```
# lsdev -Cc adapter
scsi0  Available 04-A0 Wide SCSI I/O Controller
sa0    Available 01-D0 Standard I/O Serial Port 1
sa1    Available 01-E0 Standard I/O Serial Port 2
sioka0 Available 01-G0 Keyboard Adapter
sioma0 Available 01-H0 Mouse Adapter
fda0   Available 01-I0 Standard I/O Diskette Adapter
atm0   Available 04-07 IBM PCI 155 Mbps ATM Adapter (14107c00)
atm1   Available 04-08 IBM PCI 155 Mbps ATM Adapter (14107c00)
supp0  Available 01-A0 Mini-Service Processor
ppa0   Available 01-C0 Standard I/O Parallel Port Adapter
tok0   Available 04-06 IBM PCI Tokenring Adapter (14101800)
tok1   Available       ATM LAN Emulation Client (Token Ring)
#
```

```
# lsdev -Cc if
lo0 Available  Loopback Network Interface
tr0 Available  Token Ring Network Interface
at0 Available  ATM Network Interface
at1 Available  ATM Network Interface
tr1 Available  Token Ring Network Interface
#
```

Chapter 8, “Troubleshooting within LAN Emulation environments” on page 155 will give you more details on how to tackle the problem if your test shows the communication is not working properly.

7.2 Ethernet ELAN scenario

The scenario in Figure 23 on page 125 shows how system lead participates in an Ethernet ELAN. System lead has one ATM adapter (*atm0*) and one Ethernet (*ent0*) adapter; so, the ATM Emulated LAN (ELAN) client will be ent1 on atm0.

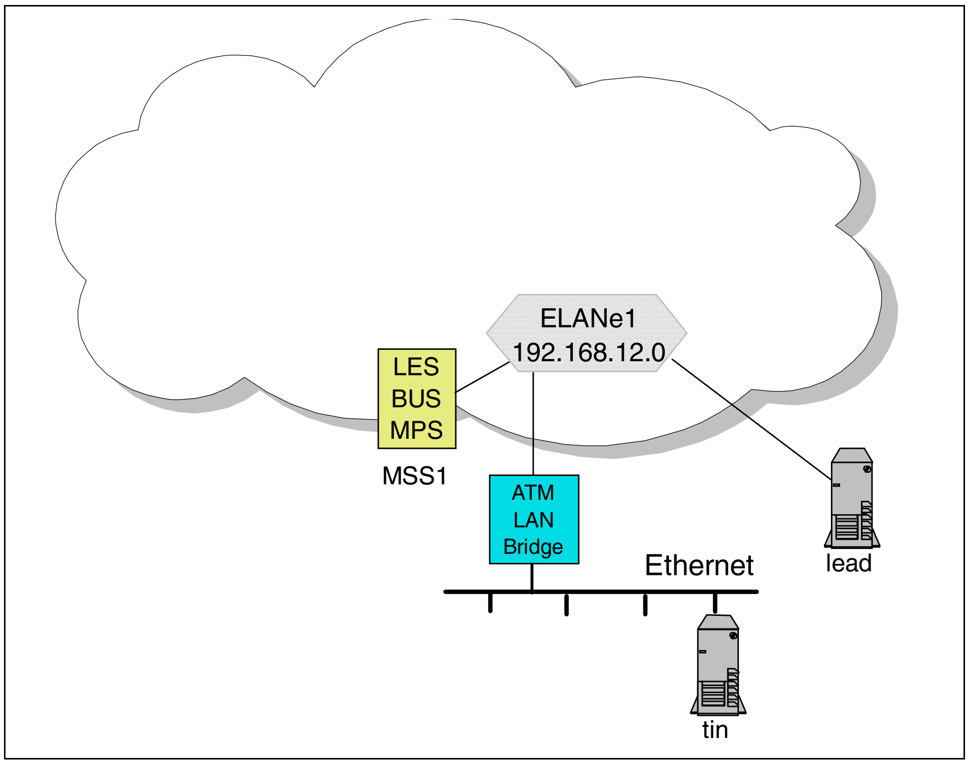


Figure 23. Ethernet LAN Emulation scenario

7.2.1 Ethernet LAN Emulation overview

In this scenario, system lead will be configured to join an Ethernet emulated LAN, that we named ELANe1, using its atm0 adapter.

We will again use MSS1 (the IBM 8210) for the ATM LAN Emulation services (LES, BUS, and LECS) and will later use it as an IP router linking IP traffic with the Token Ring (ELANt1) scenario in section 7.1.1, “Token Ring LAN Emulation overview” on page 116.

Once lead has joined the ELAN, and an IP interface has been configured, it will be able to communicate with other ELAN members running TCP/IP. In our simple environment, lead can communicate with system tin because it runs TCP/IP, and the intervening ATM LAN bridge provides proxy LEC services for it. Just as in a real LAN, systems can only talk if they are running matching protocols. So, if *tin* runs SNA or IPX, then *lead* must run the same protocol.

7.2.2 SMIT configuration for Ethernet

We will configure system lead's LAN Emulation Client for the Ethernet ELAN, that we named ELANe1, following the naming and addressing conventions outlined in Chapter 4, “Sample Network” on page 33. The Ethernet LAN Emulation Client on system lead is ent1 configured on atm0. The IP interface, configured on the ent1 LANE client, is en1.

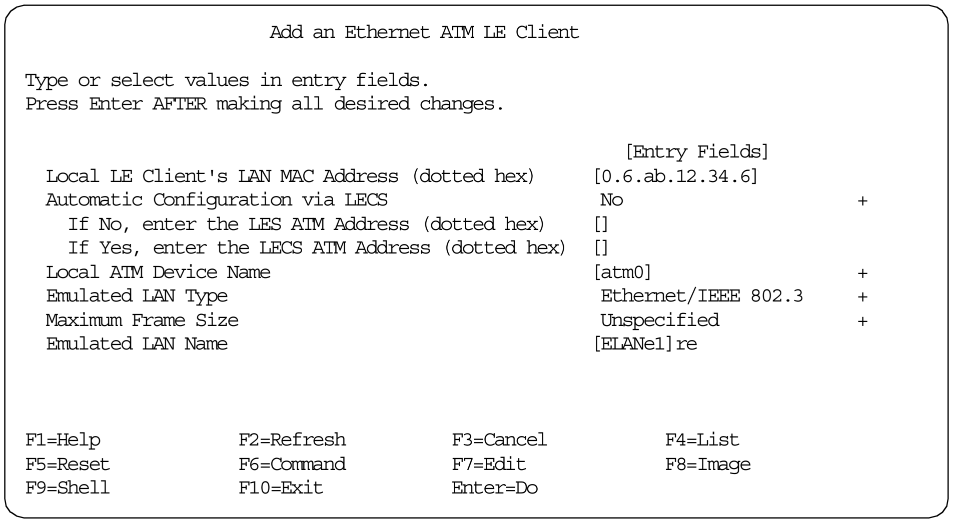
We assume the ATM LAN Emulation software is installed as described in 7.1.2, “SMIT configuration for Token Ring” on page 117, and that the atm0 adapter is *Available* and properly connected to the ATM network as described in Chapter 5, “Configuration and attachment” on page 37.

7.2.2.1 Add an Ethernet LAN Emulation Client

Add the LAN Emulation Client (ent1) using the following menu progression:

```
smit
  Devices
    Communication
      ATM Adapter
        Services
          ATM LAN Emulation
            Add an ATM LE Client
              Add an Ethernet ATM LE Client
```

or use the fastpath: `smit atmle_panel`



```
                        Add an Ethernet ATM LE Client

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                        [Entry Fields]
  Local LE Client's LAN MAC Address (dotted hex)      [0.6.ab.12.34.6]
  Automatic Configuration via LECS                     No                    +
    If No, enter the LES ATM Address (dotted hex)     []
    If Yes, enter the LECS ATM Address (dotted hex)   []
  Local ATM Device Name                               [atm0]                 +
  Emulated LAN Type                                    Ethernet/IEEE 802.3   +
  Maximum Frame Size                                   Unspecified           +
  Emulated LAN Name                                   [ELANe1]re



F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Most of the changeable parameters have the same meaning as described in Section 7.1.2, “SMIT configuration for Token Ring” on page 117, but for the sake of completeness, we decided to repeat them here. Only the *Maximum Frame Size* parameter is different between the Token Ring and the Ethernet description.

Local LE Client's LAN MAC	This is the MAC address that will be used by the Ethernet LEC in the ELAN environment. It is six octets long with a "." between each octet. You can use any value for the MAC address. The only restriction is that no two LE Clients can have the same MAC address in the same ELAN. The following is an example MAC address: 2.60.8C.2C.D2.DC Ethernet 10.0.5A.4F.4B.C4 Token Ring An entry is required in this field.
Automatic Config via LECS	Specify Yes (automatic configuration) to receive configuration information from the LAN Emulation Configuration Server (LECS) located in device MSS1. The default value is No (manual configuration). = No (default) The LE client's configuration parameters (including the ATM address) will be manually configured in the SMIT menus. Set this to No, and then specify the 20-byte ATM address of the LAN Emulation Server (LES) in the LES ATM address field. = Yes Select Yes if the LAN Emulation Configuration Server (LECS) will provide configuration settings for this LE client. Parameters include the LE server ATM address as well as any additional configuration parameters provided by the LECS. A setting of Yes means the ATM address of the LECS can be discovered automatically. Discovery is attempted, until successful, in the following standardized hierarchy:

	- LECS 20-byte ATM address (from this SMIT menu) - ILMI Discovery (a Get of the service registry returns a list of potentials that AIX tries in order) on AIX V4.3.2 or higher systems - ATM Forum's new style and original well-known LECS addresses: C5.00.79.00.00.00.00.00.00.00.00.00.00.00. A0.3E.00.00.01.00 (only if MPOA and LANE-2 enabled) 47.00.79.00.00.00.00.00.00.00.00.00.00.00. A0.3E.00.00.01.00 - ATM Forum's well-known PVC address: decimal VPI=0,VCI=17 (which is VPI=0, VCI=11 in hex) The LECS values override any configuration values specified by the user.
LES ATM Address	Leave this field empty for this scenario. If doing manual LE client configuration (without the aid of an LECS) use this field for the ATM address of the LE ARP server (ATM Forum LE client parameter C9). This 20-byte address must be entered as hexadecimal numbers using a . (period) as the delimiter between bytes. Leading zeros of each byte may be omitted. For example, 39.11.ff.22.99.99.99.0.0.0.0.1.49.10.0.5a.68.0.a.1 If the "Automatic Configuration via LECS" field is set to No, you must specify the ATM address of the LES. If it was set to Yes, it may be left blank because the LES ATM address will be obtained from the LECS.
LECS ATM Address	Leave this field blank for this scenario. Used to specify a specific 20-byte ATM address of the LECS on the network.
Local ATM Device Name	Specify `atm1` for this scenario. This is the ATM adapter for this LE client. There are defaults, but we recommend making a specific entry.
Emulated LAN Type	This value is set based upon the client type being added and cannot be changed in SMIT. Values are Ethernet/IEEE 802.3 for Ethernet LE clients and Token Ring for Token Ring LE clients.

Maximum Frame Size	Unspecified - The LECS/LES will assign what the max frame size should be. 1516 - The max frame size will be 1516.
Emulated LAN Name	Specify ELANt1 for this scenario. This is the network administrator-defined name of the ELAN. If the ELAN name can be obtained from the LECS automatically, this field may be left blank. Otherwise, you must enter in the ELAN name to which this LE client will belong. The ELAN name can be obtained from your network administrator. The ELAN name is an SNMPv2 string of 1-32 characters, and blanks can be inserted within the name by keying a "~" character wherever a blank character is desired.
Force Emulated LAN Name	Leave this field at the default (No) for this scenario. This field determines whether AIX checks to ensure the ELAN name sent in a join request matches the one returned in the join response. If this option is set to Yes, and the names do not match, the LE client will not configure -- the driver enters the limbo state and moves to the next LECS address in the list to try again. The name match must be exact and is case sensitive. The default is No, meaning the LE client will configure even if the name returned does not match that in the request.
Enable Forum MPOA and LANE-2 functions	Leave this field at the default (No) for this scenario. Select Yes if this LE client will be using MPOA. The default is No, meaning this LE client will NOT be using MPOA.
MPOA Primary Auto Configurator	Leave this field at the default for this scenario. Select Yes if the LE client will get MPOA configuration information from the LECS. Select No if the LE client will NOT get MPOA configuration information from the LECS. To use this option, the Enable Forum MPOA and LANE-2 functions field must be set to Yes.

Entries are required for the local LE client's LAN MAC address and possibly the LES ATM address or LECS ATM address depending on the support provided at the server. If the server accepts the "well-known ATM address" for

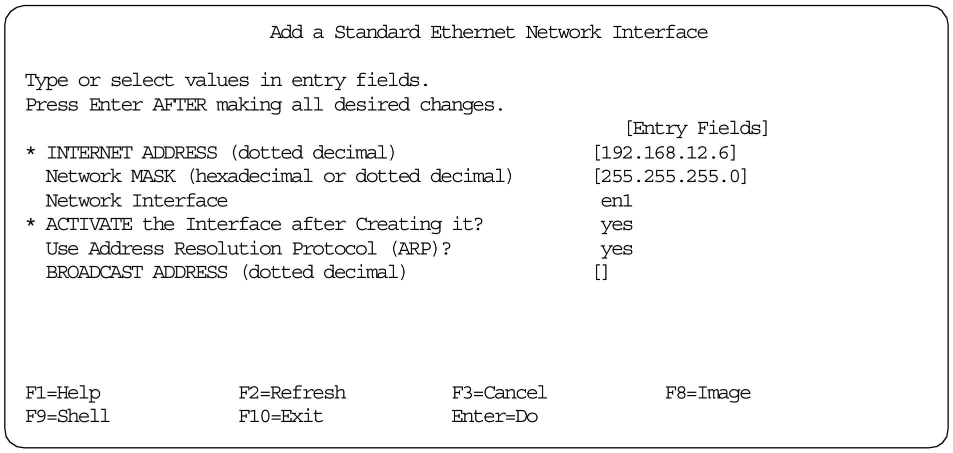
LECS or registers its LECS address with ILMI, then automatic configuration via LECS, can be set to "Yes," and the LES and LECS ATM address fields are left blank. If the server does not support the "well-known ATM address" for LECS, or does not register with ILMI, an ATM address must be entered for either LES (manual configuration) or LECS (automatic configuration). All other configuration attribute values are optional.

Additional parameters are listed and explained in the Change/Show an ATM LE Client SMIT screen (see 7.3.2, “SMIT configuration for a LAN Emulation Client” on page 141) and in 7.4.2, “ELAN naming tips” on page 149 and 7.4.3, “Additional LAN Emulation device driver parameters” on page 150.

Once the LE client has been created, the user must configure the desired upper-layer application. In this case, add the standard Ethernet IP interface (en1), which will run atop the LAN Emulation Client (ent1) using the following menu progression. The menu should be familiar, as it is the same one used to configure a real Ethernet LAN attachment:

```
smit
  Communications Applications and Services
    TCP/IP
      Further Configuration
        Network Interfaces
          Network Interface Selection
            Add a Network Interface
              Add an Ethernet Network Interface
```

or use the fastpath: `smit mkinet`



```
                    Add a Standard Ethernet Network Interface

Type or select values in entry fields.
Press Enter AFTER making all desired changes.
                                                          [Entry Fields]
* INTERNET ADDRESS (dotted decimal)                    [192.168.12.6]
  Network MASK (hexadecimal or dotted decimal)         [255.255.255.0]
  Network Interface                                     en1
* ACTIVATE the Interface after Creating it?             yes
  Use Address Resolution Protocol (ARP)?                yes
  BROADCAST ADDRESS (dotted decimal)                   []




F1=Help             F2=Refresh          F3=Cancel           F8=Image
F9=Shell            F10=Exit            Enter=Do
```

7.2.3 Test the Ethernet connection

System lead should now be able to communicate with other systems configured for TCP/IP on ELANe1; so, try the *ping* and *telnet* tests described in the Classical IP section.

However, even if you cannot test with TCP/IP commands (perhaps because no other TCP/IP systems are immediately known), you can verify that *lead* has successfully joined the ELAN by using the `entstat -d ent1` command. Check the Driver Flags under the General Statistics section of the command output. A successful ELAN join will show:

```
Driver Flags: Up Broadcast Running
              Simples AlternateAddress
```

Driver flags are explained in 7.1.3, “Test the Token Ring connection” on page 123, but an unsuccessful join looks something like this:

```
Driver Flags: Up Broadcast Simplex Limbo
              Linbo AlternateAddress
```

Status Information:

The following `lsdev` command output provides a handy snapshot summary of ATM (adapters, LAN Emulation, and TCP/IP interfaces) on system lead.

```
# lsdev -Cc adapter
scsi0  Available 04-A0 Wide SCSI I/O Controller
iga0   Available 04-03 GXT110P Graphics Adapter
sa0    Available 01-D0 Standard I/O Serial Port 1
sa1    Available 01-E0 Standard I/O Serial Port 2
sioka0 Available 01-G0 Keyboard Adapter
sioma0 Available 01-H0 Mouse Adapter
fda0   Available 01-I0 Standard I/O Diskette Adapter
ent0   Available 04-C0 IBM PCI Ethernet Adapter (22100020)
atm0   Available 04-08 IBM PCI 155 Mbps ATM Adapter (14107c00)
supp0  Available 01-A0 Mini-Service Processor
ent1   Available       ATM LAN Emulation Client (Ethernet)
#
```

```
# lsdev -Cc if
en0 Available  Standard Ethernet Network Interface
et0 Defined    IEEE 802.3 Ethernet Network Interface
lo0 Available  Loopback Network Interface
en1 Available  Standard Ethernet Network Interface
et1 Available  IEEE 802.3 Ethernet Network Interface
#
```

Chapter 8, “Troubleshooting within LAN Emulation environments” on page 155, will give you more details how to tackle the problem if your test shows the communication is not working properly.

7.3 MPOA scenario

The AIX ATM Multiprotocol Over ATM (MPOA) client support builds on the existing ATM LAN Emulation environment. Users choose to take this incremental step to get "More Performance Outta ATM"! MPOA minimizes IP routing hops between end systems, thereby improving performance across an ATM network.

MPOA involves a client (an end station, such as an RS/6000) and server (IP router) logical component model. Although MPOA could apply in a broader environment, AIX MPOA currently focuses entirely on IP. So, for now, MPOA means only ATM LAN Emulation and TCP/IP.

Figure 24 shows the traditional routed IP path between systems 192.168.12.6 and 192.16.13.7. Even though these Class C network numbers appear to be sequential, this does not reflect the physical network layout, and IP packets must actually be relayed through four intermediary IP routers.

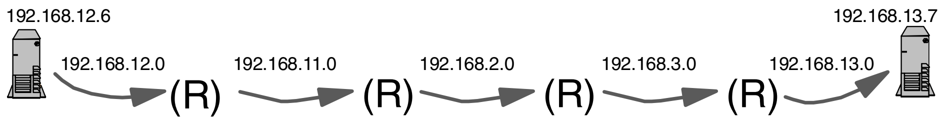


Figure 24. Traditional routed IP path

Figure 25 on page 133 reveals that this routed IP path is actually between systems lead and gold in our sample ATM environment, and the IP routers are a combination of IBM 8210 MSSs and multi-homed RS/6000s configured to route IP traffic between two virtual LANs. These networks are those that have been created in Chapter 6, “Classical IP over ATM” on page 71. Now, we are combining these individual networks to demonstrate the purpose and benefits of MPOA. Figure 25 on page 133 has some hidden complexities; so, here is a quick review of the stages in the routed path.

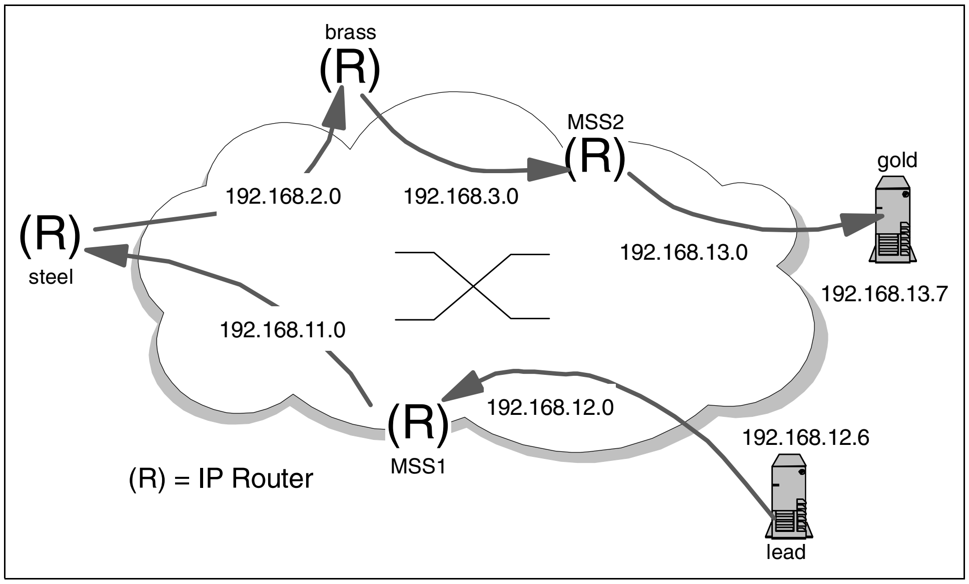


Figure 25. Routed IP path between system lead and gold

System *lead* is attached to IP network 102.168.12.0, which is on ELANe1. Figure 26 on page 134 shows the Ethernet and Token Ring ELANs (ELANe1 and ELANt1) that were created earlier in this chapter. The MSS1 device provides the IP routing function that links traffic between these two networks.

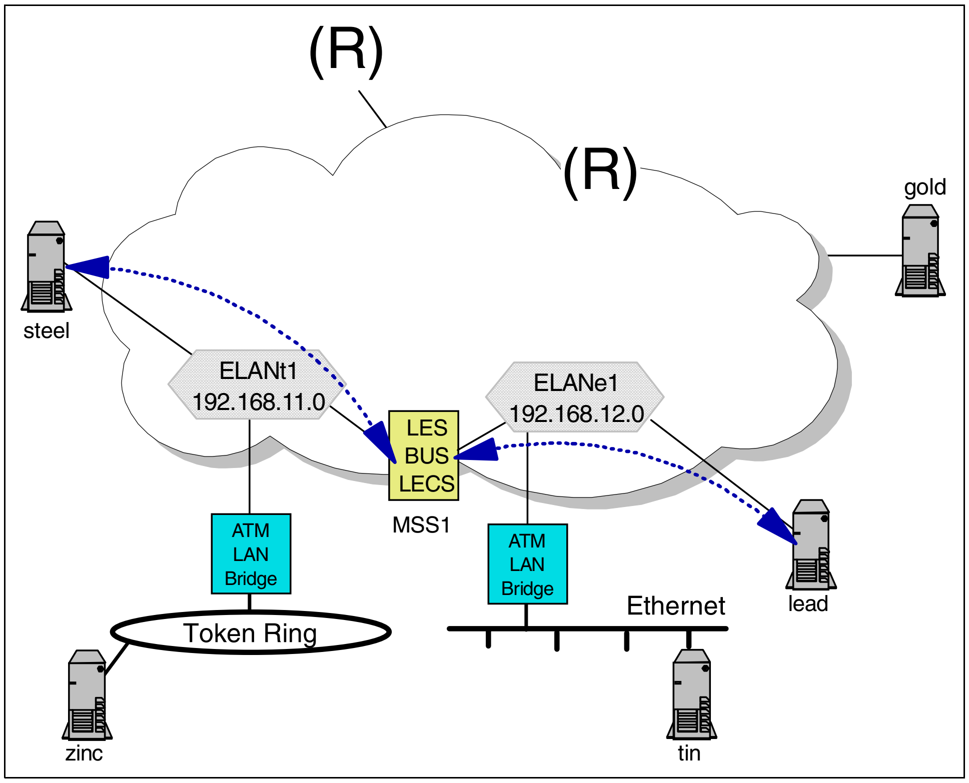


Figure 26. IP routing in the LAN Emulation portion

System *steel* is a multi-homed host that participates in three virtual LANs: ELANt1, the PVC link with system *iron* (not shown), and the 192.168.2.0 Classical IP LIS, which will be added in the next figure. Thus, system *steel* provides the link between the LAN Emulation and Classical IP portions of the network as shown in Figure 27 on page 135.

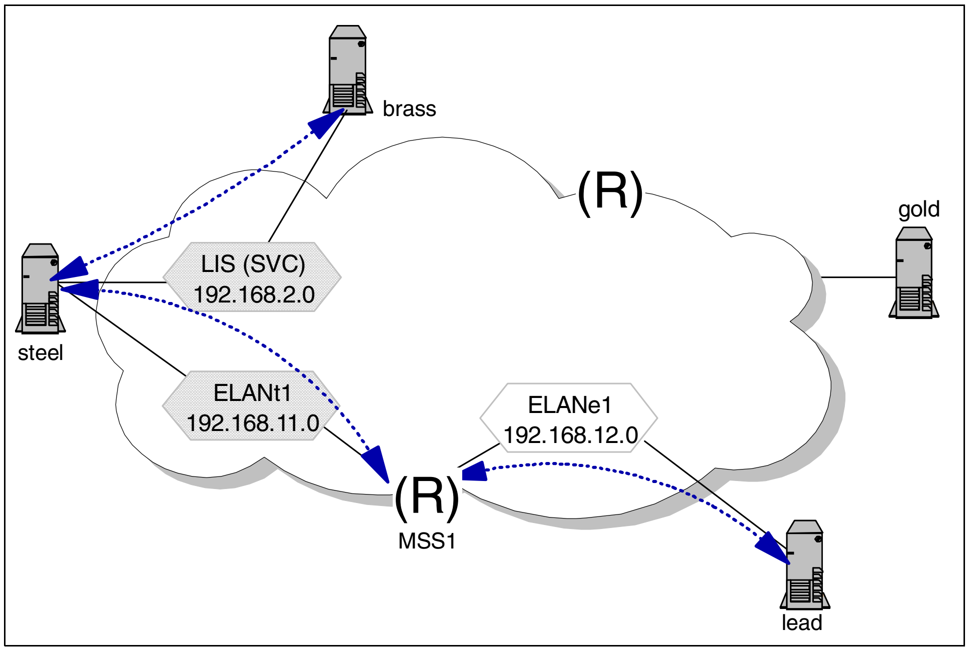


Figure 27. IP routing in the Classical IP portion

Figure 28 on page 136 finishes the routed path. It shows system brass, our backup ARP server from the Classical IP RFC 2225 section, configured as an ARP client and is now routing traffic between the two Classical IP LISs. MSS2 has a presence on 192.168.3.0 LIS and ELANe2 and provides the IP routing for this last leg of the path.

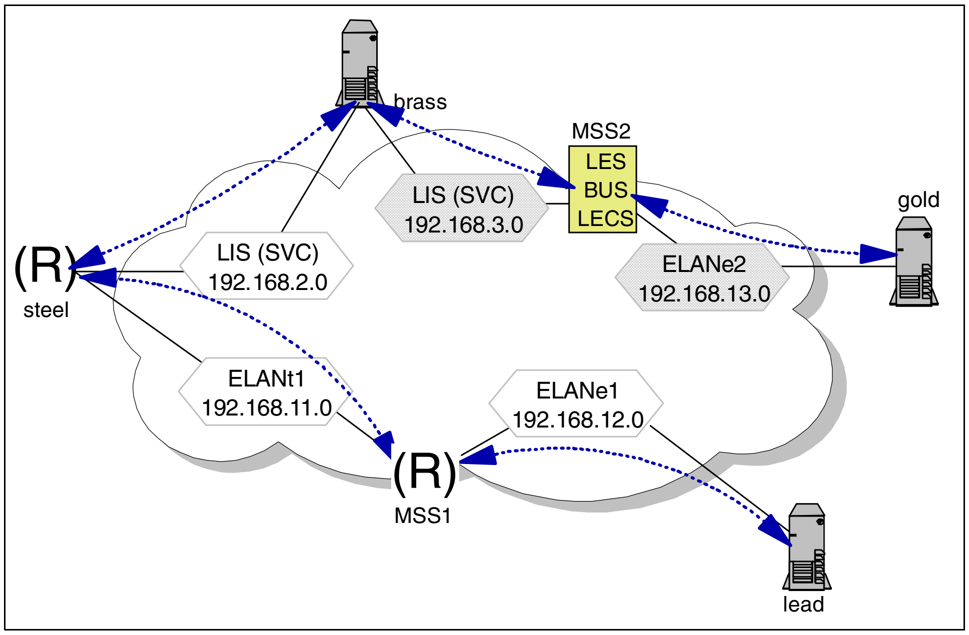


Figure 28. IP routing between systems lead and gold

So, how can MPOA help in this sample network? Remember that the ATM networks we have defined (ELANs and LISs) are virtual; they are defined through software, not LAN wiring segments. Since systems *lead* and *gold* both have unique, 20-byte ATM address identities on the ATM network, they should, ideally, be able to talk to each other directly, skipping the extra hops dictated by this IP routing scheme. MPOA's goal is to do just that, to provide a direct communication link between lead and gold as shown in Figure 29 on page 137.

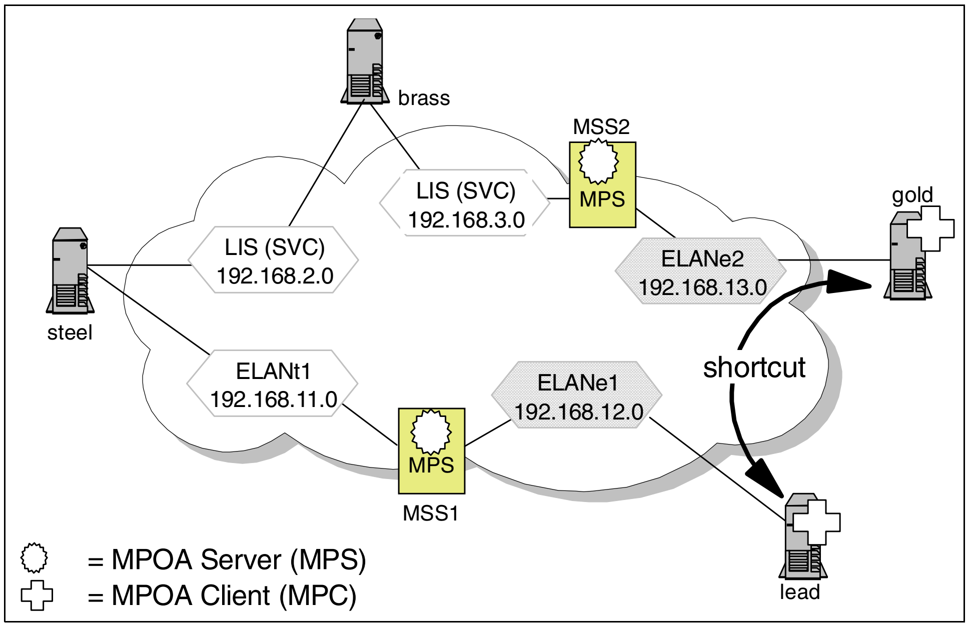


Figure 29. Introducing MPOA components

MPOA clients (MPCs) and servers (MPSs) work together to eliminate these extra routing hops via "shortcuts". An MPOA client exchanges information with its nearby router (the MPOA server) to set up a shortcut path to the target as shown in the figure. Since both lead and gold have MPOA client functions, the MPOA servers drop out of the data path, and the short cut path that is ultimately established looks like Figure 30 on page 138. If one workstation implements MPOA client, and the other does not, the shortcut is implemented between the MPOA client and the MPOA-aware device nearest the target. (For more on this, see 7.4.4, “Communicating with non-MPC end stations” on page 153).

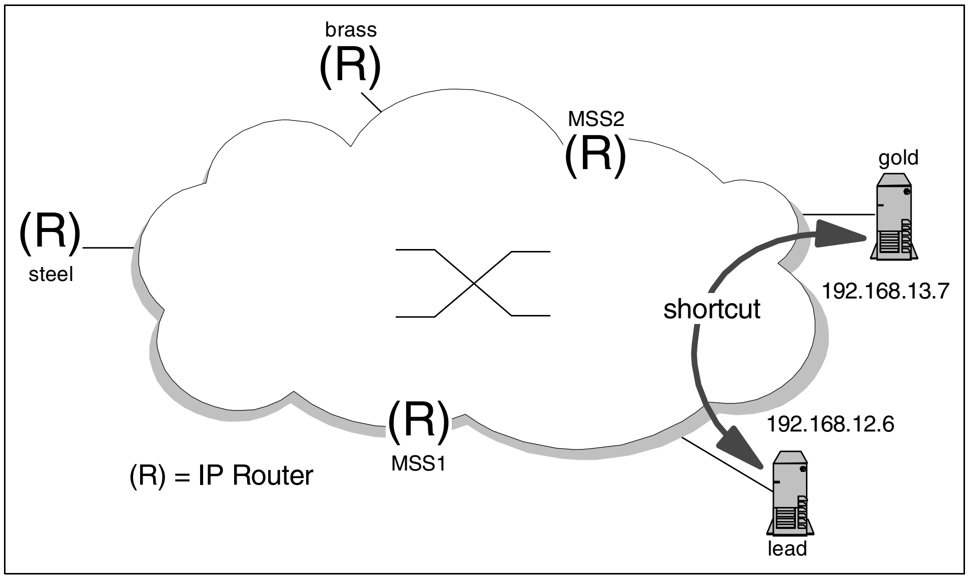


Figure 30. MPOA communication shortcut

MPOA servers maintain complete knowledge of the MAC and internetworking layer topologies for the subnetworks they serve. They exchange information with other MPS and MPCs.

MPOA clients register their ATM and IP addresses with the MPS. They maintain a local cache of mapping information learned from ATM packets and from requests made with the MPS. The cached information represents either the IP address of the client itself or, if the client is an edge device or router, the IP addresses reachable through the client.

The actual steps involved in the MPOA shortcut setup, illustrated in Figure 31 on page 139, are as follows. This flow is generic; so, the names MPC1, MPC2, MPS1, and MPS2 are system lead, gold, MSS1, and MSS2, respectively.

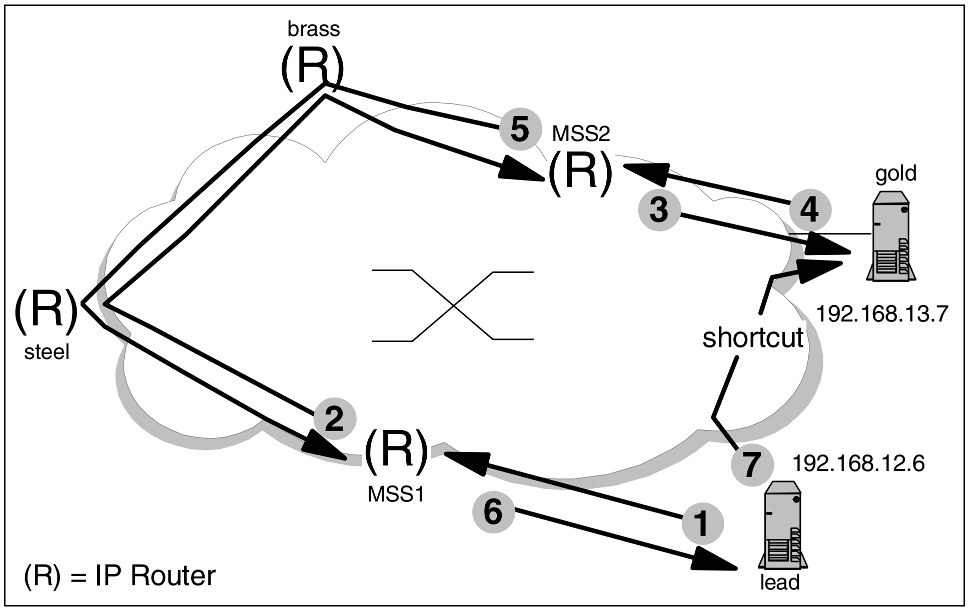


Figure 31. Flow of MPOA shortcut setup

1. MPOA Resolution Request (sent from MPC1 to MPS1)
2. NHRP Resolution Request (sent from MPS1 to MPS2)
3. MPOA Cache Imposition Request (sent from MPS2 to MPC2)
4. MPOA Cache Imposition Reply (sent from MPC2 to MPS2)
5. NHRP Resolution Reply (sent from MPS2 to MPS1)
6. MPOA Resolution Reply (sent from MPS1 to MPC1)
7. Shortcut is set up between MPC1 and MPC2

Many MPOA publications also reference Next Hop Resolution Protocol or NHRP. Although the shortcut flow shows that MPOA clients send NHRP like messages, they are not NHRP clients. NHRP is also a client-server implementation. It is related to MPOA, in that NHRP server functionality is deployed at network routers/switches, which are likely to be MPOA servers also. This allows the MPOA servers to talk to each other. NHRP client functionality is usually deployed in inexpensive edge devices, such as LAN switches with ATM uplinks, rather than user workstations. Contrary to what the name implies, NHRP (pronounced "nerp") is not a routing protocol; it is an address resolution protocol. While RFC 1577 and 2255 ARP servers map ATM and IP addresses for a single Classical IP logical IP subnet, NHRP servers maintain a cache of IP and ATM addresses for multiple LISs on a single ATM network. NHRP is not inherently restricted to TCP/IP or ATM but, like MPOA, most of the work done so far is in this environment.

Note

Our sample environment only shows the networks we have created for our ATM scenarios. A real network would probably be more complex; so, the return path might be the same, or if routing protocols are different, it might take a different path. The exact route is unimportant; the key concept is the relay through and around the intermediary routers.

Also, MPOA can be used to improve performance for end stations that are both directly attached to the ATM network and indirectly attached on a legacy LAN. Although our examples show all ATM VLANs, intervening networks in the communication path may be combinations of real LANs.

AIX and MPOA

AIX Version 4.3.3 now includes enhancements for Multiprotocol Over ATM (MPOA) clients. The initial support focus is for TCP/IP over standard and IEEE 802.3 Ethernet frame types.

To add MPOA capabilities to an existing ELAN environment, the RS/6000 needs to run AIX V 4.3.3 or above and be enabled for MPOA in SMIT. An MPOA server must be added to the existing LAN Emulation network services (LES, BUS, and LECS).

7.3.1 Overview

This scenario builds on the Ethernet ELAN scenario shown in 7.2.1, “Ethernet LAN Emulation overview” on page 125. As shown in Figure 29 on page 137, systems *lead* and *gold* will implement MPC, and the two MSS systems will provide the MPS support required to set up a direct shortcut between these systems on ELANe1 and ELANe2.

We show the configuration steps for systems lead and gold and will again use MSS1 (the IBM 8210) for the ATM LAN Emulation Services (LES, BUS, and LECS).

Once the shortcut has been established, systems *lead* and *gold* will communicate directly.

7.3.2 SMIT configuration for a LAN Emulation Client

We will configure systems *lead* and *gold* LAN Emulation Clients for MPOA following the naming and addressing conventions outlined in Chapter 4, “Sample Network” on page 33. The Ethernet LAN Emulation Client on system *lead* is ent1 configured on atm0. The IP interface, configured on the ent1 LAN Emulation Client, is en1. On system gold, the en0 IP interface is configured on LAN Emulation Client ent0 on the atm0 ATM device.

We assume the ATM LAN Emulation and MPOA software is installed as described in 7.2.2, “SMIT configuration for Ethernet” on page 126, and the atm0 adapter is *Available* and properly connected to the ATM network as described in Chapter 5, “Configuration and attachment” on page 37. We also assume that the LAN Emulation Client is properly configured on both systems.

Note

The steps listed in 7.3.2, “SMIT configuration for a LAN Emulation Client” on page 141 should be done only one time for the entire system and must be completed prior to configuring the LE client for MPOA. These changes will enable MPOA for all ELAN interfaces on the system.

Once the system is successfully configured and participating in an ATM LAN Emulation ELAN (emulated LAN), enabling MPOA client operations is a matter of following the SMIT menu progression to add the client followed by setting a single parameter in the LE client’s configuration to enable it for MPOA.

7.3.2.1 Add the MPOA client

Add the MPOA client for system lead and gold LAN Emulation Clients (ent1 and ent0, respectively) for MPOA automatic configuration by using the following menu progression.

```
smit
  Devices
    Communication
      ATM Adapter
        Services
          Multi-Protocol Over ATM (MPOA)
            Add an MPOA Client
```

or use the fastpath: `smit mpoa_panel`

```
                          Add an MPOA Client

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                     [Entry Fields]
  Automatic Configuration via LECS                  No                    +
  Shortcut Setup Frame Count                       [10]                   +#
  Shortcut Setup Frame Time (seconds)              [1]                    +#
  Initial Request Retry Time (seconds)             [5]                    +#
  Maximum Request Retry Time (seconds)             [40]                   +#
  Failed request retry Hold Down Time (seconds)    [160]                  +#
  VCC Inactivity Timeout value (minutes)           [20]                   +#




F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

The default parameters can be left for our environment.

Auto Config via LEC/LECS	Specifies whether the MPOA client is to be automatically configured via LAN Emulation Configuration Server (LECS). Select "Yes" if a primary LE client will be used to obtain the MPOA configuration attributes, which will override any manual or default values. The default value is "No" (manual configuration). The attribute values are: Yes - Auto configuration No - Manual configuration
Shortcut Setup Frame Count	This attribute is used in conjunction with Shortcut Setup Frame Time to determine when to establish a shortcut path. Once the MPC has forwarded at least the number of packets specified here (default is 10) to the same target within the period of time specified in Shortcut Setup Frame Time (default is one second), the MPC attempts to create a shortcut VCC. This attribute correlates to ATM Forum MPC Configuration parameter MPC-p1. The default value is 10 packets.

Shortcut Setup Frame Time	This attribute is used in conjunction with Shortcut Setup Frame Count to determine when to establish a shortcut path. Once the MPC has forwarded at least that number of packets to the same target within the period specified in this parameter, the MPC attempts to create a shortcut VCC. This attribute correlates to ATM Forum MPC Configuration parameter MPC-p2. The default value is one second.
Request Retry Time	Specifies the length of time to wait before sending the first retry of a request that does not receive a response. This attribute correlates to ATM Forum MPC Configuration parameter MPC-p4. The default value is five seconds.
Max Request Retry Time	Specifies the maximum length of time to wait when retrying requests that have not received a response. Each retry duration after the initial retry is doubled until the retry duration reaches this Maximum Request Retry Time. All subsequent retries will wait this maximum value. This attribute correlates to ATM Forum MPC Configuration parameter MPC-p5. The default value is 40 seconds.
Failed resolution request retry Hold Down Time	Specifies the length of time to wait before reinitiating a failed address resolution attempt. This value is normally set to a value greater than Maximum Request Retry Time and correlates to ATM Forum MPC Configuration parameter MPC-p6. The default value is 160 seconds.
VCC Inactivity Timeout value	Timeout value specifies the maximum length of time to keep a shortcut VCC enabled when there is no send or receive activity on that VCC. The default value is 20 minutes.

7.3.2.2 Update the LAN Emulation Client

Update the LAN Emulation Client configuration on systems *lead* and *gold* for MPOA support using the following menu progression:

```
smit
  Devices
    Communication
      ATM Adapter
    Services
      ATM LAN Emulation
        Change / Show an ATM LE Client
```

or use the fastpath: `smit atmle_panel`

```
                                        Change / Show an ATM LE Client

Type or select values in entry fields.
Press Enter AFTER making all desired changes.
                                                         system "lead"    systrm "gold"
  LE Client Device Name                                  ent1             ent0
  Local LE Client's LAN MAC Address (dotted hex)        [0.6.ab.12.34.6]  [0.6.ab.12.34.7]
  Automatic Configuration via LECS                       No
    If No, enter the LES ATM Address (dotted hex)       [47.00.05.80.ff.> [47.00.05.80.f>
    If Yes, enter the LECS ATM Address (dotted hex)     []
  Local ATM Device Name                                 [atm0]            [atm0]
  Emulated LAN Type                                      Ethernet/IEEE802.3
  Maximum Frame Size (bytes)                             Unspecified
  Emulated LAN Name                                     [ELANe1]          [ELANe2]
  Force Emulated LAN Name                                No
  Control Timeout (seconds)                             [120]
  Maximum Configuration Retries                         [1]
  Data Direct VCC Timeout Period (seconds)              [1200]
  Expected ARP Response Time (seconds)                  [1]
  Maximum Retry LE_ARP_REQUEST Count                    [1]
  ARP Cache Size                                        [32]
  ARP Cache Aging Time (seconds)                        [300]
  Forward Delay Time (seconds)                          [15]
  Flush Timeout (seconds)                               [4]
  Path Switching Delay (seconds)                        [6]
  Connection Completion (Ready) Timer (seconds)         [4]
  Maximum Connection Completion (Ready) Retries         [2]
  Maximum Unknown Frame Time (seconds)                  [1]
  Maximum Unknown Frame Count                           [1]
  Initialization Failsafe Timeout (seconds)             [0]
  Forward and Backward Peak Rate (Kbits/sec)            [155000]
  Maximum Queued Frames                                 [60]
  Hold data VC's on loss of LES/BUS                      No
  Enable Forum MPOA and LANE-2 functions                 Yes              Yes
  MPOA Primary Auto Configurator                         No               No
  Apply change to DATABASE only                          No
```

Enable MPOA by changing the two MPOA parameters (enable forum MPOA and LANE-2 functions and MPOA Primary Auto Configuration) to *Yes*. The menu should reflect the original LE client customized configuration, and the remaining parameters should be okay as they are.

The parameters not previously described are explained in section 7.4.3, “Additional LAN Emulation device driver parameters” on page 150.

7.3.3 Test the connection

The two systems should now establish a direct shortcut VCC connection; so, add a route to the remote station and try a flood ping. The shortcut should get set up once the threshold has been reached. This can be verified by issuing the `mpcstat -a` command.

Here is an example of `mpcstat` output:

```
$ mpcstat -a
=========================================================================

MPOA CLIENT STATISTICS (mpc0) :
=========================================================================
MPC State: Operational
MPC ATM Address: 47000580ffe1000000f215164508005a99a814ff on device atm0

-------------------------------------------------------------------------

MPC CONFIGURATION
-------------------------------------------------------------------------
Shortcut Setup Frame Count                 : 10
Shortcut Setup Frame Time (sec)            : 1
Initial Request Retry Time (sec)           : 5
Maximum Request Retry Time (sec)           : 40
Failed Request Retry Hold Down Time (sec)  : 160
VCC Inactivity Time (min)                  : 20

-------------------------------------------------------------------------
MPC EGRESS CACHE
    L3 Address       Shortcut Address                            State
-------------------------------------------------------------------------
00  192.168.12.1     47000580ffe1000000f21516450004ac47ad91ff    Active


ElanID (hex)                 : b
ElanType                     : Ethernet
TagValue (hex)               : 1
CacheId (hex)                : 20
Seconds before Expiration    : 2353
Frames Received              : 3577
Frames Discarded             : 0


-------------------------------------------------------------------------
MPC INGRESS CACHE
    L3 Address       Shortcut Address                            State
-------------------------------------------------------------------------
```

```
00  192.168.13.7      47000580ffe1000000f21516450004ac47ad91ff    Resolved


TagValue (hex)              : 1
Seconds before Expiration   : 1153
Frames Forwared             : 3547
Resolution Reqs Sent        : 1
Succesful Res Repl Recvd    : 1
Resolution Naks Recvd       : 0
VC Handle for this Shortcut : 12
VC State                    : Operational
VC Call Failures            : 0
Atm device used for this VC : atm0

01  192.168.12.1      0000000000000000000000000000000000000000 Flow Detect

Seconds before Expiration   : -687
Frames Forwared             : 0
Resolution Reqs Sent        : 0
Succesful Res Repl Recvd    : 0
Resolution Naks Recvd       : 0
VC Handle for this Shortcut : 0
VC State                    : Idle
VC Call Failures            : 0
Atm device used for this VC : atm0


-------------------------------------------------------------------------
MPS LIST
LEC Name MPS ATM Address MPS MAC Address
-------------------------------------------------------------------------
00 ent0 47000580ffe1000000f2151645000629b9ae8d02 000629b9ae8d

-------------------------------------------------------------------------

MPC STATISTICS
-------------------------------------------------------------------------
Transmitted Frames        : 3574        Received Frames            : 3577
Transmitted Octets        : 343104      Received Octets            : 343392

Res Req Xmit (total)      : 1           Imposition Req Recvd       : 2
Res Req Xmit (Refresh)    : 1
Res Req Xmit (Re-xmit)    : 0
Res Req Xmit (Timouts)    : 0
Resolution Repl Success   : 1
Res Repl Discarded        : 0
Resolution Naks Recvd     : 0           Imposition Naks Sent       : 0
```

```
    Adm Prohibited        : 0              Adm Prohibited        : 0
    Insuff Resources      : 0              Insuff Resources      : 0
    No Binding            : 0              No Binding            : 0
    Binding Not Unique    : 0              Binding Not Unique    : 0
    No Cache Resources    : 0              No Cache Resources    : 0
    No Shortcut Resources : 0              No Shortcut Resources : 0
    No Cache or S-cut Res : 0              No Cache or S-cur Res : 0
    Unsupported Protocol  : 0              Unsupported Protocol  : 0
    Unsupported Mac Enc   : 0              Unsupported Mac Enc   : 0
    Unspecified           : 0              Unspecified           : 0

Inactv I-cache Deletions  : 0          E-Cache entries Aged-out  : 0

MPS Purges Received       : 0          Dataplane Purges Recvd    : 0
MPS Purges Discarded      : 0          Dataplane Purges Xmitted  : 0
MPS Triggers Received     : 0          Dataplane Purge Re-xmit   : 0
MPS Triggers Discarded    : 0          Dataplane Purge Timeouts  : 0
MPS Keep Alives Received  : 45
MPS Keep Alives Discarded : 0          Frames Returned to LEC    : 55
```

You can verify that *steel* has successfully joined the ELAN using the `entstat -d ent1` command. Check the driver flags under the *General Statistics* section of the command output. A successful ELAN join will show:

```
Driver Flags: Up Broadcast Running
```

Driver flags are explained in 7.1.3, “Test the Token Ring connection” on page 123, but an unsuccessful join looks something like this

```
Driver Flags: Up Broadcast Simplex Limbo
```

Status Information:

The following is the new `lsdev` command output that summarizes ATM (adapters, LAN Emulation, MPOA, and TCP/IP interfaces) on system *lead*.

```
# lsdev -Cc adapter
scsi0  Available 04-A0 Wide SCSI I/O Controller
scsi1  Available 04-B0 Wide SCSI I/O Controller
iga0   Available 04-03 GXT110P Graphics Adapter
sa0    Available 01-D0 Standard I/O Serial Port 1
sa1    Available 01-E0 Standard I/O Serial Port 2
sioka0 Available 01-G0 Keyboard Adapter
sioma0 Available 01-H0 Mouse Adapter
fda0   Available 01-I0 Standard I/O Diskette Adapter
ent0   Available 04-C0 IBM PCI Ethernet Adapter (22100020)
atm0   Available 04-08 IBM PCI 155 Mbps ATM Adapter (14107c00)
supp0  Available 01-A0 Mini-Service Processor
siota0 Available 01-B0 Tablet Adapter
ppa0   Available 01-C0 Standard I/O Parallel Port Adapter
paud0  Available 01-F0 Ultimedia Integrated Audio
atm1   Available 04-02 IBM PCI 155 Mbps ATM Adapter (14107c00)
ent1   Available       ATM LAN Emulation Client (Ethernet)
tok0   Available 04-07 IBM PCI Tokenring Adapter (14101800)
tok1   Available       ATM LAN Emulation Client (Token Ring)
#
```

```
# lsdev -Cc if
en0 Available  Standard Ethernet Network Interface
et0 Defined    IEEE 802.3 Ethernet Network Interface
lo0 Available  Loopback Network Interface
tr0 Available  Token Ring Network Interface
en1 Available  Standard Ethernet Network Interface
et1 Defined    IEEE 802.3 Ethernet Network Interface
at0 Available  ATM Network Interface
at1 Available  ATM Network Interface
tr1 Defined    Token Ring Network Interface
```

7.4 Advanced LAN Emulation and MPOA considerations

The following sections provide details on LANE and MPOA issues and advanced configuration parameters.

7.4.1 Duplicate MAC addresses on an ELAN

The Local LE client's LAN MAC address is a required SMIT field, and the configured values are expected to be unique on each emulated LAN just as they are on real LANs. In our sample environment, we defined a standard MAC address and then varied it using a unique system number. Users can also enter the real MAC address of the ATM adapter in this field, as ATM adapter manufacturers assign unique addresses that would also be unique on the Token Ring or Ethernet ELAN.

Nonetheless, there are times in the normal course of events when duplicate values might exist, but recovery can occur without user or administrator intervention.

When a remote LE Client is subjected to a switch/router load balancing swap or is physically moved to a different proxy, for instance, there is a chance for its LAN MAC address to be correlated with two different 20-byte ATM addresses. As long as this is a transient situation, and not an ELAN configuration error, the protocol allows for learning the eventual new singular address correlation, and data flow will adjust accordingly. If the ELAN is configured incorrectly with duplicate LAN MAC addresses, the LE Client that attempts to add the duplicate will generally fail to join the ELAN with status value 4 (duplicate lan destination registration).

7.4.2 ELAN naming tips

The *Emulated ELAN Name* SMIT parameter is the name of the Emulated LAN this LE client wishes to join. (It is the AIX *elan_name* attribute and corresponds with ATM Forum LE Client parameter C5.) Specify an SNMPv2 DisplayString of 1-32 characters or, if unused, leave the parameter blank. See RFC1213 for a definition of an SNMPv2 DisplayString.

When attempting to join an ELAN, this operator configured name may need to match the LECS/LES server value exactly. Some servers can alias the ELAN name, thus allowing the operator to specify a logical name that correlates to the actual name. Other servers may require an exact name to be specified.

Previous versions of AIX LANE would accept any *elan_name* from the server, even when configured differently by the operator. However, with multiple LECS/LES now possible, it is desirable that only the ELAN identified by the network administrator is joined.

Use the *force_elan_name* attribute (Force Emulated LAN Name in SMIT) to insure that the name you have specified will be the only ELAN joined. If no *elan_name* attribute is configured at the AIX LE Client, or if the *force_elan_name* attribute is disabled, the server can stipulate whatever elan_name is available.

Blanks may now be inserted within an *elan_name* by typing a tilde (~) character whenever a blank character is desired. This allows a network administrator to specify an ELAN name with imbedded blanks as in the default of some servers. Any tilde character that occupies the first character position of the elan_name is interpreted as a tilde. This means a name may start with a tilde, but all remaining tilde characters are converted to blanks.

The *Force Emulated LAN Name* parameter (the AIX *force_elan_name* attribute) determines whether the Emulated LAN Name returned from the LECS or LES server must exactly match the name entered in SMIT. The default value, *No*, allows the server to specify the ELAN name.

Select *Yes* if the SMIT name field must perfectly match the server configuration and join parameters. This allows a specific ELAN to be joined when multiple LECS and LES servers are available on the network. If this option is set to *Yes* and the names do not match, the LE Client will not configure; the driver enters the LIMBO state and moves to the next LECS address in the list to try again.

The name match must be exact and is case sensitive. For instance, we used the name ELANt1 in the sample scenario. This SMIT entry does not match a server parameter of elant1, ELANt1, or ELANe1; so, the ELAN join would fail. If we had configured the client to look for identical names, the *tokstat* (or *enstat*) output would reflect the failure with driver flag *Limbo*.

7.4.3 Additional LAN Emulation device driver parameters

The ATM LANE driver supports the following additional LE client configuration parameters. Current values can be viewed via the `lsattr` command and can be set using the *chdev* command or in the SMIT menu (fastpath atmle_panel).

arp_aging_time	Specifies the maximum timeout period (in seconds) that the LE Client will maintain an LE_ARP cache entry without verification (ATM Forum LE client parameter C17). The default value is 300 seconds (five minutes).
arp_cache_size	Specifies the maximum number of LE_ARP cache entries that will be held by the LE client before removing the least recently used entry. The default value is 32 entries.
arp_response_timeo ut	Specifies the maximum timeout period (in seconds) for LE_ARP exchanges (ATM Forum LE client parameter C20). The default value is one second.
control_timeout	Specifies the maximum timeout period (in seconds) for most request/response control frame interaction. The default value is 120 seconds (two minutes).

failsafe_time	Specifies the maximum timeout period (in seconds) that the LE client will attempt to recover from a network outage. A value of zero indicates that attempts to recover should not stop unless a non-recoverable error is encountered. The default value is zero (unlimited).
flush_timeout	Specifies the maximum timeout period (in seconds) for FLUSH request/response exchanges (ATM Forum LE client parameter C21). The default value is four seconds.
fwd_delay_time	Specifies the maximum timeout period (in seconds) that the LE Client will maintain an entry for a non-local MAC address in its LE_ARP cache without verification when the Topology Change flag is true (ATM Forum LE client parameter C18). The default value is 15 seconds.
max_arp_retries	Specifies the maximum number of times an LE_ARP request can be retried (ATM Forum LE Client parameter C13). The default value is 1.
max_config_retries	Specifies the number of times a configuration control frame, such as LE_JOIN_REQUEST should be retried using a duration of control_timeout seconds between retries. The default is one.
max_queued_frames	Specifies the maximum number of outbound packets that will be held for transmission per LE_ARP cache entry. This queueing occurs when the Maximum Unknown Frame Count (max_unknown_fct) has been reached or when flushing previously transmitted packets while switching to a new virtual channel. The default value is 60 packets.
max_rdy_retries	Specifies the maximum number of READY_QUERY packets sent in response to an incoming call that has not yet received data or a READY_IND packet. The default value is two retries.

max_unknown_fct	Specifies the maximum number of frames for a given unicast LAN MAC address that may be sent to the Broadcast and Unknown Server (BUS) within time period Maximum Unknown Frame Time (max_unknown_ftm) (ATM Forum LE Client parameter C10). The default value is one.
max_unknown_ftm	Specifies the maximum timeout period (in seconds) that a given unicast LAN address may be sent to the Broadcast and Unknown Server (BUS). The LE Client will send no more than Maximum Unknown Frame Count (max_unknown_fct) packets to a given unicast LAN destination within this timeout period (ATM Forum LE client parameter C11). The default value is one second.
path_sw_delay	Specifies the maximum timeout period (in seconds) that frames sent on any path in the network will take to be delivered (ATM Forum LE client parameter C22). The default value is six seconds.
peak_rate	Specifies the forward and backward peak bit rate in K-bits per second that will be used by this LE Client to set up virtual channels. It is generally best to specify a value that is compatible with the lowest speed remote device that you expect this LE Client to be communicating with. Higher values may cause congestion in the network. The default value is 155,000 K-bits per second and is adjusted to the actual speed of the adapter for known adapters.
ready_timeout	Specifies the maximum timeout period (in seconds) in which data or a READY_IND message is expected from a calling party (ATM Forum LE Client parameter C28). The default value is four seconds.
soft_restart	Specifies whether active data VCs are to be maintained during connection loss of ELAN services such as the LE ARP Server (LES) or Broadcast and Unknown Server (BUS). Normal ATM Forum operation forces a disconnect of data VCs when LES/BUS connections are lost. This option to maintain active data VCs may be advantageous when server backup capabilities are available. The default value is No.

vcc_activity_timeout	Specifies the maximum timeout period (in seconds) for inactive Data Direct VCCs. Any switched Data Direct VCC that does not transmit or receive data frames in the timeout period is terminated (ATM Forum LE Client parameter C12). The default value is 1200 seconds (20 minutes).

7.4.4 Communicating with non-MPC end stations

In the real world, not all ATM attached end stations have MPOA Client (MPC) support, and not all destination systems are even ATM end stations. What happens, for example, if sample environment systems gold and tin need to communicate? System *gold* is an MPC end station, but *tin* is not. In fact, system *tin* is not even aware of the ATM network because it is attached to the real Ethernet LAN segment.

In this case, the MPS sets up a shortcut to the closest possible MPC in the path to the end station. Depending upon component capabilities in the sample environment, this might be the ATM LAN Bridge, the MSS1 (ATM device), or even system steel. Regardless of the ultimate shortcut end point, this shortcut negotiation and establishment is transparent to the LANE user in system *gold*.

7.5 Configuration recommendations

The *arp_cache_size* must always be set up by using SMIT panels to change the LANE device's parameter. The default of 32 is sufficient only for smaller networks where you will have no more than 32 other LANE MAC addresses communicating simultaneously with this LANE device. If the number of active communicating devices exceeds the *arp_cache_size*, entries in the arp must be flushed to make room for entries for outgoing traffic. Incoming VC setups will fail because of insufficient space in the arp_cache.

The *max_vc* for the adapter parameter must also be large enough to support all virtual circuits needed. LANE will have at least three virtual circuits for the LECS, LES, and LES. LANE also needs one virtual circuit per MAC address on the LANE segment. If there are 100 LANE entities that are to be simultaneously accessed, then LANE will need an *arp_cache* size of 100 and will need at least 103 Virtual Circuits from the ATM device driver and signaling daemon.

MPOA secondary LES servers can be registered by filling in the LES ATM address with the primary address and then following it with a comma and then the secondary addresses. The *Enable Forum MPOA* and *LANE-2* functions must be set to yes.

For AIX 4.2.1 and earlier releases, the default for Forward/Backward Peak Rate (kbits/sec) was set to 25,600. This speed does not take advantage of full ATM speed and should be changed to 155,000. For AIX 4.3 releases, this was changed to pickup the maximum bandwidth of the adapter and should only be changed if it is necessary to transmit at slower speeds.

Chapter 8. Troubleshooting within LAN Emulation environments

We assume that you have configured a LAN Emulation Client as described in section 7.1.2, “SMIT configuration for Token Ring” on page 117 or in section 7.2.2, “SMIT configuration for Ethernet” on page 126. If your test shows the communication is not working between the communication partners, you must do trouble shooting to determine the cause of the failure.

8.1 Checking the state of an LE Client

There are two commands you can use to check the status of a LEC:

```
entstat -d entX
tokstat -d tokX
```

When running the above commands substitute `X` with the number of your LEC. For Ethernet LECs, you use the `entstat` command. For Token Ring LECs, you use the *tokstat* command. If you are unable to communicate to other systems using your LEC, you need to verify the problem is with the LEC or is a network/resource issue. You can check the state of your LEC by looking at the `Driver Flags` section from the output of `entstat` or `tokstat`. For example, if you wanted to check the status of LEC tok2, you could run the following command:

```
tokstat -d tok2 | grep "Driver Flags"
```

The output will have one of the below flags set:

- `Running`
- `Limbo`
- `Dead`

The `Running` flag indicates that the LEC is operational. This means the problem is either lack of resources or that the LEC is unable to resolve/connect to the destination system. Looking at resource failures and taking a sniffer/iptrace trace should reveal the cause of the problem.

The `Limbo` flag indicates that the LEC failed/lost a connection to an LE Service. If the LEC was previously operational, then it lost its connection to an LE Service. If LEC remains in the *Limbo* state, then it was unable to reestablish its connection. This could either be a resource problem or a problem communicating with the switch. Looking at resource failures and taking a sniffer/iptrace trace should reveal the cause of the problem.

The `Dead` flag indicates that the LEC has suffered a hard failure. The only way to recover from this state is put the adapter in a defined state and then make it available again. For example, if tok2 is in the *Dead* state, you would do the following:

```
ifconfig tr2 down
ifconfig tr2 detach
rmdev -l tr2
rmdev -l tok2
mkdev -l tok2
chdev -l tr2 -a state=up
```

If the adapter is still in the *Dead* state, taking a kernel trace while the adapter is being brought back up should indicate the cause of the problem.

8.2 Checking for resource failures

You can check for resource failures on an ATM adapter by running the `atmstat` command. For example, if you wanted to see if there were any resource failures on atm0, you could run the following command:

```
atmstat -d atm0
```

8.2.1 Checking for resource failures on PCI adapters

The IBM Turboways 155 ATM PCI adapter has the ability to dynamically allocate buffers as needed. It also has a VCC upper limit of 2048. If this upper limit is reached, you can only reduce the number of simultaneous connections to other systems or spread the load onto another adapter. Resource failures on this adapter are usually not an issue; however, because of the high number of VCCs possible and the adapters ability to dynamically allocate resources as needed. You can check for VCC usage by running the following command:

```
atmstat -d atm0 | grep "Virtual Connections"
```

You will get output similar to the following:

```
Virtual Connections in use: 10
Max Virtual Connections in use: 64
Virtual Connections Overflow: 0
```

`Virtual Connections in use` indicates the number of VCCs currently in use.

The above output shows that we are currently using 10 VCCs.

`Max Virtual Connections in use` indicates the maximum number of VCCs that have been used at any given time since the adapter has been up. In general, you want to set the number pre-allocated VCCs to this value to reduce the amount of time spent allocating VCCs. The above output shows that, at some point, atm0 used 64 VCCs. We can run the following command to check the number of pre-allocated VCCs:

```
lsattr -El atm0 -a min_vc
```

The output for atm0 shows the following:

```
min_vc 32 Minimum Guaranteed VCs Supported True
```

The atm0 adapter currently has 32 VCCs pre-allocated. We want to increase this to 64 to meet the usual demand. You can accomplish this by running the following command:

```
chdev -l atm0 -a min_vc=64
```

You only increase this value if 64 VCCs is the usual demand. If the system does not usually use 64 VCCs, you do not want to increase this value. You can get a sense of the usual demand by resetting the statistics periodically throughout the day and looking at the *Max Virtual Connections in use*. If the max keeps hitting around 64 throughout the day, then, in our case, this is the usual demand.

If it does not, then you want to set it to the value that you are seeing as the usual demand.

The *Virtual Connections Overflow* indicates the number of times a VCC needed to be created, but the maximum number of allowed VCCs had been reached. Our output shows zero; so, we did not have any overflows. If we did have an overflow, we would want to check the current maximum with the following command:

```
lsattr -El atm0 -a max_vc
```

The output for our atm0 shows the following:

```
max_vc 1024 Maximum Number of VCs Needed True
```

So, our allowed maximum VCCs is currently set to 1024 VCCs. This value can be increased up to the upper limit of 2048. The value of max_vc should be larger than min_vc. If the overflow was caused by a temporary spike, then just increase max_vc until you longer get overflows. If the overflow was cause by the usual demand on the system, you want to increase the min_vc to the usual demand value and increase max_vc to what the maximum VCC demand will be for this system. For example, if we had a temporary spike of

1256 VCCs, we want to increase max_vc to 1256. You can do this by running the following command:

```
chdev -l atm0 -a max_vc=1256
```

If 1256 VCCs was the usual demand, then we would want to increase both min_vc and max_vc as follows:

```
chdev -l atm0 -a min_vc=1256
chdev -l atm0 -a max_vc=1400
```

8.2.2 Checking for resource failures on MCA adapters

The IBM Turboways 155 ATM MCA adapter has several different buffer sizes, each of which have their own buffer limits. Each buffer size has a set number of pre-allocated buffers and a maximum number of buffers that can be allocated. There is also a VCC upper limit of 1024. If this upper limit is reached, you can only reduce the number of simultaneous connections to other systems or spread the load onto another adapter. Resource failures can occur if the maximum number of buffers for a given buffer size is not large enough to meet the demand on the system.

You can use the `atmstat` command to determine if you have a buffer resource failure. For example, if you wanted to check if atm0 is having a buffer resource failure, you could run the following command:

```
atmstat -d atm0 | grep overflow
```

The output for our atm0 shows the following:

```
Small Mbuf overflow: 0
Medium Mbuf overflow: 0
Large Mbuf overflow: 0
Huge Mbuf overflow: 0
MTB Mbuf overflow: 0
```

The output shows that there are no overflows. This means that we are never hitting the maximum number of buffers we allowed. If any of the above values had an overflow, you would want to increase the maximum number of buffers for the buffer size that had an overflow until you are no longer getting overflows. For example, say we were getting a `Large Mbuf overflow` on atm0. We want to increase the value of max_lrg_bufs in roughly 10 percent increments until we are no longer getting overflows. We can use the following command to check the current value of max_lrg_bufs on atm0:

```
lsattr -El atm0 -a max_lrg_bufs
```

The output for our atm0 is as follows:

```
max_lrg_bufs 400 Maximum Large ATM mbufs True
```

We want to increase this value by 10 percent; so, we run the following command:

```
chdev -l atm0 -a max_lrg_bufs=440
```

If the overflows continue we will run the above command again, increasing the size by another 10 percent until we no longer get overflows.

Another resource failure can occur when buffers are not available at a given time. This can happen if a large amount of data is being transmitted or received. You can use `atmstat` to determine if you are having an mbuf availability problem. For example, we could run the following command on our atm0 adapter:

```
atmstat -d atm0 | grep "No "
```

The output for our atm0 is as follows:

```
No mbuf Errors: 0
Packets Dropped - No small DMA buffer: 0
Packets Dropped - No medium DMA buffer: 0
Packets Dropped - No large DMA buffer: 0
Receive Aborted - No Adapter Receive Buffer: 0
Transmit Attempted - No small DMA buffer: 0
Transmit Attempted - No medium DMA buffer: 0
Transmit Attempted - No large DMA buffer: 0
Transmit Attempted - No MTB DMA buffer: 0
Transmit Attempted - No Adapter Transmit Buffer: 0
```

If there was a problem getting an mbuf for any buffer size `No mbuf Errors` will be non-zero. The output above has a zero value for `No mbuf Errors`; so, we are not having mbuf allocation problems. If it was non-zero, we would look at the fields below to determine which buffer size had a problem getting an mbuf. For example, let us say that atm0 had the following output:

```
No mbuf Errors: 1500
Packets Dropped - No small DMA buffer: 0
Packets Dropped - No medium DMA buffer: 600
Packets Dropped - No large DMA buffer: 0
Receive Aborted - No Adapter Receive Buffer: 0
Transmit Attempted - No small DMA buffer: 0
Transmit Attempted - No medium DMA buffer: 900
Transmit Attempted - No large DMA buffer: 0
Transmit Attempted - No MTB DMA buffer: 0
Transmit Attempted - No Adapter Transmit Buffer: 0
```

From the above, we can see that we had problems getting medium mbufs while trying to transmit and receive. During large transfers, a few “No mbuf Errors” are normal. If the increase is large in a short time frame, however, it can impact system performance and should be tuned. To fix the problem above, we want to increase the number of pre-allocated, medium buffers. We want to increase the number of pre-allocated buffers in 10 percent increments until we no longer get “No mbuf Errors”. We first check the current size of pre-allocated medium mbufs by running the the following command:

```
lsattr -El atm0 -a med_highwater
```

The output for atm0 is as follows:

```
med_highwater 30 Minimum Medium ATM mbufs True
```

The atm0 adapter currently has 30 medium buffers pre-allocated. We increase the value by 10 percent running the following command:

```
chdev -l atm0 -a med_highwater=33
```

If the problem persists, you will run the above command again, increasing the size again by 10 percent until the problem is resolved. The value of *max_med_bufs* must be larger than med_highwater. If the value of *med_highwater* exceeds *max_med_bufs,* you must increase *max_med_bufs* so that it is larger.

Another resource failure source is an insufficient number of VCCs. You can check your VCC usage by running the following command:

```
atmstat -d atm0 | grep "Virtual Connections"
```

You will get output similar to the following:

```
Virtual Connections in use: 10
Max Virtual Connections in use: 64
Virtual Connections Overflow: 0
```

`Virtual Connections in use` indicates the number of VCCs currently in use. The above output shows that we are currently using 10 VCCs.

`Max Virtual Connections in use` indicates the maximum number of VCCs that have been used at any given time since the adapter has been up. In general, you want to set the number of pre-allocated VCCs to this value to reduce the amount of time spent allocating VCCs. The above output shows that, at some point, atm0 used 64 VCCs. We can run the following command to check the number of pre-allocated VCCs:

```
lsattr -El atm0 -a min_vc
```

The output for atm0 shows the following:

```
min_vc 16 Minimum Guaranteed VCs Supported True
```

The atm0 adapter currently has 16 VCCs pre-allocated. We want to increase this to 64 to meet the usual demand. This can be accomplished by running the following command:

```
chdev -l atm0 -a min_vc=64
```

Only increase this value if 64 VCCs is the usual demand. If the system does not usually use 64 VCCs, you do not want to increase this value. You can get a sense of the usual demand by resetting the statistics periodically throughout the day and looking at the *Max Virtual Connections in use*. If the max keeps hitting around 64 throughout the day (as in our case), then this is the usual demand. If it does not, then you want to set it to the value that you are seeing as the usual demand.

The *Virtual Connections Overflow* indicates the number of times a VCC needed to be created, but the maximum number of allowed VCCs had been reached. Our output shows zero; so, we did not have any overflows. If we did have an overflow, we would want to check the current maximum with the following command:

```
lsattr -El atm0 -a max_vc
```

The output for our atm0 shows the following:

```
max_vc 256 Maximum Number of VCs Needed True
```

So, our allowed maximum VCCs is currently set to 256 VCCs. This value can be increased up to the upper limit of 1024. The value of *max_vc* should be larger than *min_vc*. If the overflow was caused by a temporary spike, then just increase *max_vc* until you no longer get overflows. If the overflow was caused by the usual demand on the system, you want to increase the *min_vc* to the usual demand value and increase max_vc to what the maximum VCC demand will be for this system. For example, if we had a temporary spike of 956 VCCs, we would want to increase *max_vc* to 956. You can do this by running the following command:

```
chdev -l atm0 -a max_vc=956
```

If 956 VCCs was the usual demand, then we would want to increase both *min_vc* and *max_vc* as follows:

```
chdev -l atm0 -a min_vc=956
chdev -l atm0 -a max_vc=1000
```

8.3 Taking a sniffer/iptrace

Some problems require looking at the communication between the LEC and other components in the network. Normally, a sniffer is required between the LEC and the switch to resolve these problems. The CIP interfaces on AIX, however, have been modified to act like a sniffer on the network. Taking an iptrace on a CIP interface will capture all the CIP traffic but will also capture all ILMI, SSCOP, UNI (Q.2931), MPOA, and LANE traffic on the adapter. For the majority of LANE problems, this means that an iptrace on a CIP interface can be used to debug problems instead of a sniffer. The CIP interface will not, however, register packets that are dropped by the driver due to lack of resources or due to the packet being bad/malformed. In these cases, only a sniffer trace can resolve the problem. If you do not use CIP on the adapter you wish to take an iptrace on, you can use a debug CIP interface.

8.3.1 Creating a debug CIP interface

A debug CIP interface is a CIP interface created for the sole purpose of taking iptraces on an ATM adapter that does not already have a CIP interface. You create a debug CIP interface the same way you create a real CIP interface. The only difference is that the debug interface uses made-up values rather than real values. The values used can be any that do not conflict with your current network. Below are the steps required to create a debug CIP interface:

```
smitty inet
  Add a Network Interface
    Add an ATM Network Interface
```

```
                        Add an ATM Network Interface

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

* INTERNET ADDRESS (dotted decimal)                        []
  Network MASK (hexadecimal or dotted decimal)             []
  Network Interface                                        []
  Connection Type                                          svc
  ATM Server Address                                       []
  Alternate Device                                         []
  Idle Timer                                               []
  Best Effort Bit Rate (UBR) in Kbits/sec                  []
* ACTIVATE the Interface after Creating it?                 yes



F1=Help          F2=Refresh          F3=Cancel          F4=List
F5=Reset         F6=Command          F7=Edit            F8=Image
F9=Shell         F10=Exit            Enter=Do
```

Below is an explanation of the values to use for the previous SMIT panel:

INTERNET ADDRESS (dotted decimal)	Use any TCP/IP address that will not interfere with your other, real TCP/IP networks. For example, use the address 1.2.3.4.
Network MASK (hexadecimal or dotted decimal)	Any subnet mask will work as long it does not interfere with your other, real TCP/IP networks. For example, use 255.255.255.0.
Network Interface	If you have other CIP interfaces on this system use, an interface that is not in use. If you do not have any CIP interface configured, just use at0.
Connection Type	Leave the connection type as the default svc_s.
ATM Server Address	Leave this field blank.
Alternate Device	Enter in the ATM adapter on which the LE client is being created. For example, if the LE client is to be created on atm0, then enter atm0 in this field.
Idle Timer	Leave this field blank.
Best Effort Bit Rate (UBR) in Kbits/sec	Leave this field blank.
ACTIVATE the Interface after Creating it?	Leave this as the default yes.

8.3.2 Taking an iptrace

When debugging LANE problems, there are several ways to take an iptrace to capture the problem. The least useful is an iptrace on the LEC interface. This trace only shows what is being passed up/down to the LEC. This type of trace is only useful if you are trying to debug a protocol issue.

Below are the procedures for taking an iptrace on the LEC interface en2:

```
rm /tmp/iptrc.bin
startsrc -s iptrace -a "-i en2 /tmp/iptrc.bin"
```

Reproduce the problem:

```
stopsrc -s iptrace
```

The path, */tmp/iptrc.bin,* can be replaced with whatever file name and path you wish to use. iptrace will append to any existing iptrace binary. That is the purpose of removing any existing binaries with the `rm` command before hand. When taking the iptrace replace en2 with LEC interface you need to trace.

Another way to take an iptrace is on the CIP interface. As mentioned before, an iptrace on the CIP interface will capture all ILMI, SSCOP, UNI (Q.2931), MPOA, and LANE traffic on the adapter. This is the type of iptrace that is usually taken to debug LANE problems. Most LANE problems are UNI related and not LEC related. This type of trace will capture any LEC issues as well as any UNI issues. Below are the procedures for taking an iptrace on the CIP interface at0:

```
rm /tmp/iptrc.bin
startsrc -s iptrace -a "-i at0 /tmp/iptrc.bin"
```

Reproduce the problem:

```
stopsrc -s iptrace
```

The path, */tmp/iptrc.bin,* can be replaced with whatever file name and path you wish to use. iptrace will append to any existing iptrace binary. That is the purpose of removing any existing binaries with the `rm` command before hand. When taking the iptrace, replace at0 with CIP interface you need to trace.

The last way to take an iptrace is to take a wide-open iptrace. A wide-open iptrace captures traffic on all iptraces. This type of iptrace is rarely taken because it captures a large amount of data. The large amount of data makes the iptrace difficult to analyze. In some cases, however, it necessary to see what the adapter is receiving/transmitting and what is being handed up/down to the LEC. In such cases, you would want to take a wide-open iptrace. Below are the procedures for taking a wide-open iptrace:

```
rm /tmp/iptrc.bin
startsrc -s iptrace -a "/tmp/iptrc.bin"
```

Reproduce the problem:

```
stopsrc -s iptrace
```

The path, */tmp/iptrc.bin,* can be replaced with whatever file name and path you wish to use. iptrace will append to any existing iptrace binary. That is the purpose of removing any existing binaries with the `rm` command before hand.

8.3.3 Generating an ipreport

When you take an iptrace, it creates a binary file. To analyze the `iptrace` data, you need to create a readable report. You can create a human readable report by running `ipreport` on the generated iptrace binary. The following command will create a readable report:

```
ipreport -srn /tmp/iptrc.bin > /tmp/iptrc.rpt
```

Replace the path, `/tmp/iptrc.bin`, with whatever filename and path you used in your `iptrace` command. The path, `/tmp/iptrc.bin`, can be replaced with whatever filename and path you wish to use. The `/tmp/iptrc.bin` file contains the readable report. There are other flags that you can use with `ipreport` besides `-srn`. The flag, `-srn`, is the one most frequently used when analyzing ATM iptraces. From more information on ipreport flags, refer to the man pages for `ipreport`.

8.4 Taking a kernel trace

Some problems require taking a kernel on the system that is having the problem. The following procedures can be used to take a LANE kernel trace:

```
rm /tmp/trc.bin
trace -a -j"339,3a1,3a2" -L 4096000 -T 2048000 -o /tmp/trc.bin
```

Reproduce the problem:

```
trcstop
```

The `-j` flag restricts tracing to only the specified trace hooks. The above trace hooks are as follows:

```
339 - ATM Driver
3A1 - LANE Normal Path
3A2 - LANE Error Path
```

The `-L` and `-T` flags specify the size of the output file and internal buffer. The above will create up to a 4 MB trace file. The value used for the `-T` flag should be half the value of the -L flag.

The `-o` flag specifies the output path. You can replace `/tmp/trc.bin` with the path/filename that you wish to use. For more information on `trace`, refer to the man pages.

The `trace` command creates a binary file. To generate a readable report, run the following command:

```
trcrpt /tmp/trc.bin > /tmp/trc.rpt
```

Replace the path, */tmp/trc.bin,* with the path used in the `trace` command. You can replace */tmp/trc.rpt* with the path/filename that you wish to use. The trc.rpt file contains the readable report.

The `trcrpt` command also has several formatting options that can be used when generating the report. For information on these options, refer to the man pages.

8.5 LEC initialization failures

If the status of your LEC shows “Limbo” (see 8.1, “Checking the state of an LE Client” on page 155), then the LEC has failed to establish/reestablish its initialization into the ELAN.

The process the LEC goes through to initialize itself into an ELAN is known as *LEC Initialization*. The flow of a typical LEC initialization is as follows:

```
    |                                    |
    |     Point-to-Point SVC Created     |
    | ---------------------------------> |
    |                                    | L
    |        LE_Configure Request        |
    | ---------------------------------> | E
    |                                    |
    |        LE_Configure Response       | C
    | <--------------------------------- |
    |                                    | S
    |    Point-to-Point SVC Released     |
    | ---------------------------------> |
    |                                    |
    |
    |
    |                                    |
    |    Point-to-Point SVC Created      |
    | ---------------------------------> |
  L |                                    |
    |          LE_JOIN Request           |
  E | ---------------------------------> |
    |                                    |
  C |   Point-to-Multipoint SVC Created  | L
    | <--------------------------------- |
    |                                    | E
    |          LE_JOIN Response          |
    | <--------------------------------- | S
    |                                    |
    |  LE_ARP Request(FF.FF.FF.FF.FF.FF) |
    | --------------------------------> |
    |                                   |
    |          LE_ARP Response          |
    | <-------------------------------- |
    |                                   |
    |
    |
    |                                   |
    |     Point-to-Point SVC Created    | B
    | --------------------------------> |
    |                                   | U
    |   Point-to-Multipoint SVC Created |
    | <-------------------------------- | S
    |                                   |
```

8.5.1 LEC initialization tracing

You can see LEC Initialization process by taking a CIP interface iptrace. The CIP interface must be configured on the same adapter as the LEC of interest (see 8.3, "Taking a sniffer/iptrace" on page 162" for more information). Below are the procedures for capturing the LEC Initialization using CIP interface at1 and LEC tok2:

Bring down child devices and LEC

```
ifconfig tr2 down
ifconfig tr2 detach
rmdev -l tr2
rmdev -l tok2
```

Start the iptrace

```
rm /tmp/iptrc.bin
startsrc -s iptrace -a "-i at1 /tmp/iptrc.bin"
```

Bring up the LEC and child devices

```
mkdev -l tok2
chdev -l tr2 -a state=up
sleep 20
Stop the iptrace
stopsrc -s iptrace
```

If you are not using TCP/IP on the LEC replace all the tr2 commands with equivalent commands to bring down/up the application that is running on the LEC. You can replace */tmp/iptrc.bin* with whatever path/filename you wish to use. Also replace at1 and tok2 with the CIP interface and LEC you are using.

The sleep is provided to allow environments with multiple LECS enough time to try additional LECS if necessary. If the time is too short to capture the attempts off all the LECS returned increase the sleep time.

Once the have an iptrace of the LEC Initialization process you can generate a human readable report using ipreport. Below are the procedures for generating an ipreport:

```
ipreport -srn /tmp/iptrc.bin > /tmp/iptrc.rpt
```

For more information on generating a report see “Generating an ipreport”.

8.5.2 Normal LEC initialization

The following is the ipreport output of a typical LEC Initialization. It is provide as a guide to help you analyze the LEC Initialization process from the output of ipreport. The ordering of some packets may vary but does not affect the LEC Initialization process. Comments have been added to the ipreport to explain each packet and provide any relevant information.

Below is an ILMI query to the Service Registry. If the LECS has its address in the Service Registry the LEC will get the address back in the Response. If the address is not Registered the LEC will get a response with a value of NULL. This query only occurs in AIX 4.3.2 and higher.

```
Packet Number 1
ATM: ====( 45 bytes transmitted on interface at1 )== 12:19:45.233879040
ATM: VC = 0.16
ILMI: Get Next Request PDU
ILMI: Request ID   = 65286
ILMI: Error Status = 0(Success)
ILMI: Object ID    =
1.3.6.1.4.1.353.2.8.1.1.3.0.0(atmfSrvcRegATMAddress)
ILMI:     Value    = <Null>
```

Below is the response and we have a value filled in so we will use this address to connect to the LECS. If the address was NULL we would try and connect to the LECS using the Well-Known LECS ATM Address.

```
Packet Number 2
ATM: ====( 85 bytes received on interface at1 )== 12:19:45.239394304
ATM: VC = 0.16
ILMI: Get Response PDU
ILMI: Request ID   = 65286
ILMI: Error Status = 0(Success)
ILMI: Object ID    = 1.3.6.1.4.1.353.2.8.1.1.3.0.10.1.3.
ILMI:                6.1.4.1.353.1.5.1.1
ILMI:                (atmfSrvcRegATMAddress)
```

```
ILMI:     Value    = 47 0 5 80 ff e1 0 0 0 f2 f 28 f7 0 20 48 f 28 f7 10
```

There can be more than one LECS that the LE Client can use. The LE Client will keep querying until it gets all the addresses. The next query below returned NULL so there was only one LECS address register.

```
Packet Number 3
ATM: ====( 57 bytes transmitted on interface at1 )== 12:19:45.240221696
ATM: VC = 0.16
ILMI: Get Next Request PDU
ILMI: Request ID   = 65286
ILMI: Error Status = 0(Success)
ILMI: Object ID    = 1.3.6.1.4.1.353.2.8.1.1.3.0.10.1.3.
ILMI:                6.1.4.1.353.1.5.1.1
ILMI:                (atmfSrvcRegATMAddress)
ILMI:     Value    = <Null>
```

Below is the response of the ILMI query and it returned a value of NULL. This means there are no more LECS ATM address in the Service Registry.

```
Packet Number 4
ATM: ====( 57 bytes received on interface at1 )== 12:19:45.244584728
ATM: VC = 0.16
ILMI: Get Response PDU
ILMI: Request ID   = 65286
ILMI: Error Status = 2(No such Name)
ILMI: Object ID    = 1.3.6.1.4.1.353.2.8.1.1.3.0.10.1.3.
ILMI:                6.1.4.1.353.1.5.1.1
ILMI:                (atmfSrvcRegATMAddress)
ILMI:     Value    = <Null>
```

Below is the start of the Point-to-Point SVC setup to the LECS. The setup is done to the ATM Address returned in the ILMI Get Response. Also we want to make note the Call Reference Value for this connection is 10.

```
Packet Number 5
ATM: ====( 116 bytes transmitted on interface at1 )===
12:19:45.248495616
ATM: VC = 0.5
Q2931: 05(Setup)
Q2931: Call Ref Value  = 10
Q2931: Message Len     = 103
Q2931:
Q2931: IE Element      = 58(AAL Parameters)
Q2931:   AAL Type      = AAL type 5
Q2931:   Fd Max SDU sz = 1516
Q2931:   Bk Max SDU sz = 1516
Q2931:   SSCS Type     = Null
```

```
Q2931: IE Element      = 59(ATM Traffic Descriptor)
Q2931:   Fwd Peak Cell Rate (CLP : 0+1)
Q2931:                 = 413334 cells/sec (175253616 bps)
Q2931:   Bkwd Peak Cell Rate(CLP : 0+1)
Q2931:                 = 413333 cells/sec (175253192 bps)
Q2931:   Best Eff Ind  = 190
Q2931: IE Element      = 5e(Broadband Bearer Capability)
Q2931:   Bearer Class  = BCOB-X
Q2931:   Traffic Type  = No Indication
Q2931:   Susc. to clip = Susceptible to Clipping
Q2931:   Conn Type     = Point-to-Point
Q2931: IE Element      = 5f(Broadband Low Layer Information)
Q2931: IE Element      = 70(Called Party Number)
Q2931:   Type of Num   = 1 (International)
Q2931:   Addressing    = 2 (ATM Endsystem Address)
Q2931:   Address       =
Q2931:      47 0 5 80 ff e1 0 0 0 f2 f 28 f7 0 20 48 f 28 f7 10
Q2931: IE Element      = 6c(Calling Party Number)
Q2931:   Type of Num   = 1 (International)
Q2931:   Addressing    = 2 (ATM Endsystem Address)
Q2931:   Address       =
Q2931:      47 0 5 80 ff e1 0 0 0 f2 f 28 f7 8 0 5a 99 99 fc 4
Q2931: IE Element      = 5c(QoS Parameter)
Q2931:   Coding Std    = 00(ITU-T Standardized)
Q2931:   QoS Class     = QoS Class 0(Unspecified)
Q2931:   QoS Class     = QoS Class 0(Unspecified)
```

SVC setup continues. We know this “Call Proceeding” message belongs to our SVC Setup to the LECS because it has the same Call Reference Value as the Setup call. From the below information, we now know that the SVC being setup will have a VCC of 0:83 (VPI:VCI).

```
Packet Number 6
ATM: ====( 24 bytes received on interface at1 )== 12:19:45.259151360
ATM: VC = 0.5
Q2931: 02(Call Proceeding)
Q2931: Call Ref Value  = 10
Q2931: Message Len     = 9
Q2931:
Q2931: IE Element      = 5a(Connection ID)
Q2931:   VPI           = 0
Q2931:   VCI           = 83
```

SVC setup continues. Again the Call Reference Value is 10 so we know this “Connect” message is tied with the LECS SVC Setup.

```
Packet Number 7
ATM: ====( 36 bytes received on interface at1 )== 12:19:45.271764992
```

```
ATM: VC = 0.5
Q2931: 07(Connect)
Q2931: Call Ref Value  = 10
Q2931: Message Len     = 22
Q2931:
Q2931: IE Element      = 5a(Connection ID)
Q2931:   VPI           = 0
Q2931:   VCI           = 83
Q2931: IE Element      = 5f(Broadband Low Layer Information)
```

The LEC has sent a “Connect Acknowledge” back so at this point the SVC Setup to the LECS is complete. We now have an SVC connection to the LECS with a VCC of 0:83 and a Call Reference Value of 10.

```
Packet Number 8
ATM: ====( 16 bytes transmitted on interface at1 )== 12:19:45.272572416
ATM: VC = 0.5
Q2931: 0F(Connect Acknowledge)
Q2931: Call Ref Value  = 10
Q2931: Message Len     = 0
```

Now that we have an SVC Connection to the LECS we expect the LE Client to send an LE_CONFIGURE Request to the LECS. Continuing through the trace we see the LE_CONFIGURE Request sent to the LECS. We make note that the ELAN Name SZ is zero. This means that an ELAN name was not specified so we are relying on the LECS to assign us to the correct ELAN. You will also notice that the below message was sent on VCC 0:83 which is the VCC that was setup to the LECS.

```
Packet Number 9
ATM: ====( 108 bytes transmitted on interface at1 )== 12:19:45.305894912
ATM: VC = 0.83
LANE CTRL CONFIGURE_REQUEST
LANE: Status        = 0
LANE: Trans ID      = 0
LANE: Request ID    = 0
LANE: Src LAN Dest:
LANE:   Tag         = 1
LANE:   MAC Addr    = 8 0 5a 99 99 fc
LANE: Tar LAN Dest:
LANE:   Tag         = 0
LANE:   MAC Addr    = 0 0 0 0 0 0
LANE: Src ATM ADDR  = 47 0 5 80 ff e1 0 0 0 f2 f 28 f7 8 0 5a 99 99 fc 4
LANE: Tar ATM ADDR  = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
LANE: ELAN Name SZ  = 0
```

Our Configure Request has been sent so we should expect to see a Configure Response back from the LECS. We received the below Configure Response from the LECS. Looking at the Status field we see that it is zero which is correct. If a non-zero status code was returned this means that an error occurred. Also we want to make note of the “ELAN Name” and “Tar ATM ADDR” in this packet. We see that the ELAN Name is “t_lan0”. This is the ELAN that LEC will now attempt to join. Also the “Tar ATM ADDR” contains the ATM Address of the LES. This is the address the LEC will use to connect to the LES.

```
Packet Number 10
ATM: ====( 108 bytes received on interface at1 )== 12:19:45.310941696
ATM: VC = 0.83
LANE CTRL CONFIGURE_RESPONSE
LANE: Status        = 0
LANE: Trans ID      = 0
LANE: Request ID    = 0
LANE: Src LAN Dest:
LANE:   Tag         = 1
LANE:   MAC Addr    = 8 0 5a 99 99 fc
LANE: Tar LAN Dest:
LANE:   Tag         = 0
LANE:   MAC Addr    = 0 0 0 0 0 0
LANE: Src ATM ADDR  = 47 0 5 80 ff e1 0 0 0 f2 f 28 f7 8 0 5a 99 99 fc 4
LANE: Tar ATM ADDR  = 47 0 5 80 ff e1 0 0 0 f2 f 28 f7 0 20 48 f 28 f7 f0
LANE: ELAN Name SZ  = 6
LANE: ELAN Name     = t_lan0
```

Point-to-Point SVC Setup to the LES using the ATM address returned in the LE_CONFIGURE Response. We want to make note that the Call Reference for# this connection is 11.

```
Packet Number 11
ATM: ====( 116 bytes transmitted on interface at1 )== 12:19:45.311970304
ATM: VC = 0.5
Q2931: 05(Setup)
Q2931: Call Ref Value  = 11
Q2931: Message Len     = 103
Q2931:
Q2931: IE Element      = 58(AAL Parameters)
Q2931:   AAL Type      = AAL type 5
Q2931:   Fd Max SDU sz = 1516
Q2931:   Bk Max SDU sz = 1516
Q2931:   SSCS Type     = Null
Q2931: IE Element      = 59(ATM Traffic Descriptor)
Q2931:   Fwd Peak Cell Rate (CLP : 0+1)
Q2931:                 = 413334 cells/sec (175253616 bps)
```

```
Q2931:   Bkwd Peak Cell Rate(CLP : 0+1)
Q2931:                 = 413333 cells/sec (175253192 bps)
Q2931:   Best Eff Ind  = 190
Q2931: IE Element      = 5e(Broadband Bearer Capability)
Q2931:   Bearer Class  = BCOB-X
Q2931:   Traffic Type  = No Indication
Q2931:   Susc. to clip = Susceptible to Clipping
Q2931:   Conn Type     = Point-to-Point
Q2931: IE Element      = 5f(Broadband Low Layer Information)
Q2931: IE Element      = 70(Called Party Number)
Q2931:   Type of Num   = 1 (International)
Q2931:   Addressing    = 2 (ATM Endsystem Address)
Q2931:   Address       =
Q2931:      47 0 5 80 ff e1 0 0 0 f2 f 28 f7 0 20 48 f 28 f7 f0
Q2931: IE Element      = 6c(Calling Party Number)
Q2931:   Type of Num   = 1 (International)
Q2931:   Addressing    = 2 (ATM Endsystem Address)
Q2931:   Address       =
Q2931:      47 0 5 80 ff e1 0 0 0 f2 f 28 f7 8 0 5a 99 99 fc 4
Q2931: IE Element      = 5c(QoS Parameter)
Q2931:   Coding Std    = 00(ITU-T Standardized)
Q2931:   QoS Class     = QoS Class 0(Unspecified)
Q2931:   QoS Class     = QoS Class 0(Unspecified)
```

We expect to see a Call Proceeding next but we see a Release of Call Reference number 10. Looking back at our notes we see that is our SVC connection to the LECS. This is normal. Once the LEC has its configuration information it no longer needs to talk the LECS so the connection is release.

```
Packet Number 12
ATM: ====( 20 bytes transmitted on interface at1 )== 12:19:45.312769280
ATM: VC = 0.5
Q2931: 4D(Release)
Q2931: Call Ref Value  = 10
Q2931: Message Len     = 6
Q2931:
Q2931: IE Element      = 8(Cause)
Q2931: Cause Value     = 31(Normal, Unspecified)
```

The SVC Setup to the LES continues. We make note that the VCC will be 0:84.

```
Packet Number 13
ATM: ====( 24 bytes received on interface at1 )== 12:19:45.322357248
ATM: VC = 0.5
Q2931: 02(Call Proceeding)
Q2931: Call Ref Value  = 11
Q2931: Message Len     = 9
```

```
Q2931:
Q2931: IE Element      = 5a(Connection ID)
Q2931:   VPI           = 0
Q2931:   VCI           = 84
```

We receive the expected “Release Complete” for the Release message we sent for the LECS connection. We know this by matching up the Call Reference numbers again. The SVC to the LECS is now gone.

```
Packet Number 14
ATM: ====( 20 bytes received on interface at1 )== 12:19:45.337934336
ATM: VC = 0.5
Q2931: 5A(Release Complete)
Q2931: Call Ref Value  = 10
Q2931: Message Len     = 6
Q2931:
Q2931: IE Element      = 8(Cause)
Q2931: Cause Value     = 31(Normal, Unspecified)
```

SVC setup to the LES continues.

```
Packet Number 15
ATM: ====( 36 bytes received on interface at1 )== 12:19:45.343265792
ATM: VC = 0.5
Q2931: 07(Connect)
Q2931: Call Ref Value  = 11
Q2931: Message Len     = 22
Q2931:
Q2931: IE Element      = 5a(Connection ID)
Q2931:   VPI           = 0
Q2931:   VCI           = 84
Q2931: IE Element      = 5f(Broadband Low Layer Information)
```

The LEC has sent a “Connect Acknowledge” back so at this point the SVC Setup to the LES is complete. We now have a SVC connection to the LES with a VCC of 0:84 and a Call Reference Value of 11.

```
Packet Number 16
ATM: ====( 16 bytes transmitted on interface at1 )=== 12:19:45.344071168
ATM: VC = 0.5
Q2931: 0F(Connect Acknowledge)
Q2931: Call Ref Value  = 11
Q2931: Message Len     = 0
```

Now that we have an SVC connection to the LES we expect our LE Client to send a Join Request to the LES for ELAN t_lan0. The LEC will send its LAN MAC address and its ATM address in this request. You will also notice that the below Join Request is sent on VCC 0:84. This is the VCC that was setup to the LES.

```
Packet Number 17
ATM: ====( 108 bytes transmitted on interface at1 )== 12:19:45.377415680
ATM: VC = 0.84
LANE CTRL JOIN_REQUEST
LANE: Status        = 0
LANE: Trans ID      = 1
LANE: Request ID    = 0
LANE: Src LAN Dest:
LANE:   Tag         = 1
LANE:   MAC Addr    = 8 0 5a 99 99 fc
LANE: Tar LAN Dest:
LANE:   Tag         = 0
LANE:   MAC Addr    = 0 0 0 0 0 0
LANE: Src ATM ADDR  = 47 0 5 80 ff e1 0 0 0 f2 f 28 f7 8 0 5a 99 99 fc 4
LANE: Tar ATM ADDR  = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
LANE: ELAN Name SZ  = 6
LANE: ELAN Name     = t_lan0
```

Before receiving a response to our Join Request the LES will setup a Uni-directional SVC back to the LEC for distributing information. Looking at the Calling Party Address we see that the below Setup is indead coming from the LES. Also looking at the “Conn Type” we notice that this is a Point-to-Multipoint connection whereas, the other connections have been Point-to-Point connections. We also make note that the Call Reference Value is 1578.

```
Packet Number 18
ATM: ====( 132 bytes received on interface at1 )== 12:19:45.389979392
ATM: VC = 0.5
Q2931: 05(Setup)
Q2931: Call Ref Value  = 1578
Q2931: Message Len     = 118
Q2931:
Q2931: IE Element      = 59(ATM Traffic Descriptor)
Q2931:   Fwd Peak Cell Rate (CLP : 0+1)
Q2931:                 = 353207 cells/sec (149759768 bps)
Q2931:   Bkwd Peak Cell Rate(CLP : 0+1)
Q2931:                 = 0 cells/sec (0 bps)
Q2931:   Best Eff Ind  = 190
Q2931: IE Element      = 5e(Broadband Bearer Capability)
Q2931:   Bearer Class  = BCOB-X
Q2931:   Traffic Type  = No Indication
Q2931:   Susc. to clip = Susceptible to Clipping
Q2931:   Conn Type     = Point-to-Multipoint
```

```
Q2931: IE Element      = 70(Called Party Number)
Q2931:   Type of Num   = 1 (International)
Q2931:   Addressing    = 2 (ATM Endsystem Address)
Q2931:   Address       =
Q2931:      47 0 5 80 ff e1 0 0 0 f2 f 28 f7 8 0 5a 99 99 fc 4
Q2931: IE Element      = 5c(QoS Parameter)
Q2931:   Coding Std    = 00(ITU-T Standardized)
Q2931:   QoS Class     = QoS Class 0(Unspecified)
Q2931:   QoS Class     = QoS Class 0(Unspecified)
Q2931: IE Element      = 5a(Connection ID)
Q2931:   VPI           = 0
Q2931:   VCI           = 85
Q2931: IE Element      = 58(AAL Parameters)
Q2931:   AAL Type      = AAL type 5
Q2931:   Fd Max SDU sz = 1516
Q2931:   Bk Max SDU sz = 1516
Q2931:   SSCS Type     = Null
Q2931: IE Element      = 54(ATM Traffic Descriptor)
Q2931: IE Element      = 6c(Calling Party Number)
Q2931:   Type of Num   = 1 (International)
Q2931:   Addressing    = 2 (ATM Endsystem Address)
Q2931:   Address       =
Q2931:      0 5 80 ff e1 0 0 0 f2 f 28 f7 0 20 48 f 28 f7 f0
Q2931: IE Element      = 5f(Broadband Low Layer Information)
```

SVC setup from the LES to the LEC continues. Again we use the Call Reference Numbers to tie the messages together. We also make note that the SVC will have a VCC of 0:85.

```
Packet Number 19
ATM: ====( 32 bytes transmitted on interface at1 )== 12:19:45.391526912
ATM: VC = 0.5
Q2931: 02(Call Proceeding)
Q2931: Call Ref Value  = 1578
Q2931: Message Len     = 16
Q2931:
Q2931: IE Element      = 5a(Connection ID)
Q2931:   VPI           = 0
Q2931:   VCI           = 85
Q2931: IE Element      = 54(ATM Traffic Descriptor)
```

SVC setup from the LES to the LEC continues.

```
Packet Number 20
ATM: ====( 32 bytes transmitted on interface at1 )== 12:19:45.393064704
ATM: VC = 0.5
Q2931: 07(Connect)
Q2931: Call Ref Value  = 1578
```

```
Q2931: Message Len     = 16
Q2931:
Q2931: IE Element      = 5a(Connection ID)
Q2931:   VPI           = 0
Q2931:   VCI           = 85
Q2931: IE Element      = 54(ATM Traffic Descriptor)
```

The LEC has received a “Connect Acknowledge”; so, at this point, the SVC Setup from the LES is complete. We now have an SVC connection from the LES with a VCC of 0:85 and a Call Reference Value of 1578.

```
Packet Number 22
ATM: ====( 16 bytes received on interface at1 )== 12:19:45.401818368
ATM: VC = 0.5
Q2931: 0F(Connect Acknowledge)
Q2931: Call Ref Value  = 1578
Q2931: Message Len     = 0
```

Now that the LES has setup its Uni-directional SVC to our LEC we expect the LES to send the LEC a Join Response to the LEC’s Join Request. Looking further in the trace we see the Join Response. We make note that the Status field is zero. A non-zero value for this field indicates that an error occurred.

```
Packet Number 23
ATM: ====( 108 bytes received on interface at1 )== 12:19:45.404785664
ATM: VC = 0.84
LANE CTRL JOIN_RESPONSE
LANE: Status        = 0
LANE: Trans ID      = 1
LANE: Request ID    = 95
LANE: Src LAN Dest:
LANE:   Tag         = 1
LANE:   MAC Addr    = 8 0 5a 99 99 fc
LANE: Tar LAN Dest:
LANE:   Tag         = 0
LANE:   MAC Addr    = 0 0 0 0 0 0
LANE: Src ATM ADDR  = 47 0 5 80 ff e1 0 0 0 f2 f 28 f7 8 0 5a 99 99 fc 4
LANE: Tar ATM ADDR  = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
LANE: ELAN Name SZ  = 6
LANE: ELAN Name     = t_lan0
```

Now that we have joined the ELAN successfully we need to make a connection to the BUS. The ATM address of the BUS is determined by doing an LE_ARP Request for a LAN MAC address of FF.FF.FF.FF.FF.FF. The LE_ARP Response will contain the ATM of the BUS.

```
Packet Number 24
```

```
ATM: ====( 108 bytes transmitted on interface at1 )===
12:19:45.404856064
ATM: VC = 0.84
LANE CTRL ARP_REQUEST
LANE: Status        = 0
LANE: Trans ID      = 2
LANE: Request ID    = 95
LANE: Src LAN Dest:
LANE:   Tag         = 1
LANE:   MAC Addr    = 8 0 5a 99 99 fc
LANE: Tar LAN Dest:
LANE:   Tag         = 1
LANE:   MAC Addr    = ff ff ff ff ff ff
LANE: Src ATM ADDR  = 47 0 5 80 ff e1 0 0 0 f2 f 28 f7 8 0 5a 99 99 fc 4
LANE: Tar ATM ADDR  = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Below, we see the LE_ARP response to LEC's request. The `Tar ATM ADDR` field contains the ATM address of the BUS. The LEC will now use this address to set up a connection to the BUS:

```
Packet Number 25
ATM: ====( 108 bytes received on interface at1 )== 12:19:45.407246336
ATM: VC = 0.84
LANE CTRL ARP_RESPONSE
LANE: Status        = 0
LANE: Trans ID      = 2
LANE: Request ID    = 95
LANE: Src LAN Dest:
LANE:   Tag         = 1
LANE:   MAC Addr    = 8 0 5a 99 99 fc
LANE: Tar LAN Dest:
LANE:   Tag         = 1
LANE:   MAC Addr    = ff ff ff ff ff ff
LANE: Src ATM ADDR  = 47 0 5 80 ff e1 0 0 0 f2 f 28 f7 8 0 5a 99 99 fc 4
LANE: Tar ATM ADDR  = 47 0 5 80 ff e1 0 0 0 f2 f 28 f7 0 20 48 f 28 f7 f0
```

As expected, the next thing we see is an SVC setup to the BUS. We know this is to the BUS by looking at the called party address. This address matches with the one returned in the LE_ARP response. We make note that the call reference number is 12 for this SVC setup:

```
Packet Number 27
ATM: ====( 116 bytes transmitted on interface at1 )== 12:19:45.408858880
ATM: VC = 0.5
Q2931: 05(Setup)
Q2931: Call Ref Value  = 12
Q2931: Message Len     = 103
```

```
Q2931:
Q2931: IE Element      = 58(AAL Parameters)
Q2931:   AAL Type      = AAL type 5
Q2931:   Fd Max SDU sz = 18190
Q2931:   Bk Max SDU sz = 18190
Q2931:   SSCS Type     = Null
Q2931: IE Element      = 59(ATM Traffic Descriptor)
Q2931:   Fwd Peak Cell Rate (CLP : 0+1)
Q2931:                 = 413334 cells/sec (175253616 bps)
Q2931:   Bkwd Peak Cell Rate(CLP : 0+1)
Q2931:                 = 413333 cells/sec (175253192 bps)
Q2931:   Best Eff Ind  = 190
Q2931: IE Element      = 5e(Broadband Bearer Capability)
Q2931:   Bearer Class  = BCOB-X
Q2931:   Traffic Type  = No Indication
Q2931:   Susc. to clip = Susceptible to Clipping
Q2931:   Conn Type     = Point-to-Point
Q2931: IE Element      = 5f(Broadband Low Layer Information)
Q2931: IE Element      = 70(Called Party Number)
Q2931:   Type of Num   = 1 (International)
Q2931:   Addressing    = 2 (ATM Endsystem Address)
Q2931:   Address       =
Q2931:      47 0 5 80 ff e1 0 0 0 f2 f 28 f7 0 20 48 f 28 f7 f0
Q2931: IE Element      = 6c(Calling Party Number)
Q2931:   Type of Num   = 1 (International)
Q2931:   Addressing    = 2 (ATM Endsystem Address)
Q2931:   Address       =
Q2931:      47 0 5 80 ff e1 0 0 0 f2 f 28 f7 8 0 5a 99 99 fc 4
Q2931: IE Element      = 5c(QoS Parameter)
Q2931:   Coding Std    = 00(ITU-T Standardized)
Q2931:   QoS Class     = QoS Class 0(Unspecified)
Q2931:   QoS Class     = QoS Class 0(Unspecified)
```

SVC setup to the BUS continues. We make note that the VCC will be 0:86

```
Packet Number 28
ATM: ====( 24 bytes received on interface at1 )== 12:19:45.419407872
ATM: VC = 0.5
Q2931: 02(Call Proceeding)
Q2931: Call Ref Value  = 12
Q2931: Message Len     = 9
Q2931:
Q2931: IE Element      = 5a(Connection ID)
Q2931:   VPI           = 0
Q2931:   VCI           = 86
```

SVC setup to the BUS continues.

```
Packet Number 31
ATM: ====( 36 bytes received on interface at1 )== 12:19:45.455242752
ATM: VC = 0.5
Q2931: 07(Connect)
Q2931: Call Ref Value  = 12
Q2931: Message Len     = 22
Q2931:
Q2931: IE Element      = 5a(Connection ID)
Q2931:   VPI           = 0
Q2931:   VCI           = 86
Q2931: IE Element      = 5f(Broadband Low Layer Information)
```

After the SVC setup to the BUS is complete, we expect the BUS to set up a uni-directional SVC back to the LEC. The previous SVC does not have to complete before this happens, as we see below. The BUS has started its SVC back to the LEC before the SVC from the LEC to the BUS is complete. We make note that the call reference value is 1580:

```
Packet Number 32
ATM: ====( 132 bytes received on interface at1 )== 12:19:45.455311360
ATM: VC = 0.5
Q2931: 05(Setup)
Q2931: Call Ref Value  = 1580
Q2931: Message Len     = 118
Q2931:
Q2931: IE Element      = 59(ATM Traffic Descriptor)
Q2931:   Fwd Peak Cell Rate (CLP : 0+1)
Q2931:                 = 353207 cells/sec (149759768 bps)
Q2931:   Bkwd Peak Cell Rate(CLP : 0+1)
Q2931:                 = 0 cells/sec (0 bps)
Q2931:   Best Eff Ind  = 190
Q2931: IE Element      = 5e(Broadband Bearer Capability)
Q2931:   Bearer Class  = BCOB-X
Q2931:   Traffic Type  = No Indication
Q2931:   Susc. to clip = Susceptible to Clipping
Q2931:   Conn Type     = Point-to-Multipoint
Q2931: IE Element      = 70(Called Party Number)
Q2931:   Type of Num   = 1 (International)
Q2931:   Addressing    = 2 (ATM Endsystem Address)
Q2931:   Address       =
Q2931:      47 0 5 80 ff e1 0 0 0 f2 f 28 f7 8 0 5a 99 99 fc 4
Q2931: IE Element      = 5c(QoS Parameter)
Q2931:   Coding Std    = 00(ITU-T Standardized)
Q2931:   QoS Class     = QoS Class 0(Unspecified)
Q2931:   QoS Class     = QoS Class 0(Unspecified)
Q2931: IE Element      = 5a(Connection ID)
Q2931:   VPI           = 0
```

```
Q2931:   VCI           = 87
Q2931: IE Element      = 58(AAL Parameters)
Q2931:   AAL Type      = AAL type 5
Q2931:   Fd Max SDU sz = 18190
Q2931:   Bk Max SDU sz = 18190
Q2931:   SSCS Type     = Null
Q2931: IE Element      = 54(ATM Traffic Descriptor)
Q2931: IE Element      = 6c(Calling Party Number)
Q2931:   Type of Num   = 1 (International)
Q2931:   Addressing    = 2 (ATM Endsystem Address)
Q2931:   Address       =
Q2931:      0 5 80 ff e1 0 0 0 f2 f 28 f7 0 20 48 f 28 f7 f0
Q2931: IE Element      = 5f(Broadband Low Layer Information)
```

The LEC has sent a “Connect Acknowledge” back; so, at this point the SVC setup to the BUS is complete. We now have an SVC connection to the BUS with a VCC of 0:86 and a call reference value of 12:

```
Packet Number 33
ATM: ====( 16 bytes transmitted on interface at1 )== 12:19:45.456079616
ATM: VC = 0.5
Q2931: 0F(Connect Acknowledge)
Q2931: Call Ref Value  = 12
Q2931: Message Len     = 0
```

SVC from the BUS to the LEC continues. We notice that the VCC will be 0:87 for this SVC:

```
Packet Number 34
ATM: ====( 32 bytes transmitted on interface at1 )== 12:19:45.457731584
ATM: VC = 0.5
Q2931: 02(Call Proceeding)
Q2931: Call Ref Value  = 1580
Q2931: Message Len     = 16
Q2931:
Q2931: IE Element      = 5a(Connection ID)
Q2931:   VPI           = 0
Q2931:   VCI           = 87
Q2931: IE Element      = 54(ATM Traffic Descriptor)
```

SVC setup from BUS to the LEC continues:

```
Packet Number 35
ATM: ====( 32 bytes transmitted on interface at1 )== 12:19:45.490214912
ATM: VC = 0.5
Q2931: 07(Connect)
Q2931: Call Ref Value  = 1580
Q2931: Message Len     = 16
```

```
Q2931:
Q2931: IE Element      = 5a(Connection ID)
Q2931:   VPI           = 0
Q2931:   VCI           = 87
Q2931: IE Element      = 54(ATM Traffic Descriptor)
```

The LEC has received a “Connect Acknowledge”; so, at this point, the SVC setup from the BUS is complete. We now have a SVC connection from the BUS with a VCC of 0:87 and a call reference value of 1580:

```
Packet Number 36
ATM: ====( 16 bytes received on interface at1 )== 12:19:45.496945152
ATM: VC = 0.5
Q2931: 0F(Connect Acknowledge)
Q2931: Call Ref Value  = 1580
Q2931: Message Len     = 0
```

The LEC has completed initialization successfully after the BUS establishes its SVC to the LEC. At this point, the LEC now has four SVC connections in the LANE environment. The connections are as follows:

```
    VCC                 Description                  Components
   ======  ====================================  ==============
    0:84    Bi-directional Point-to-Point          LEC <==> LES
    0:85    Uni-directoinal Point-to-Multipoint    LEC <==O LES
    0:86    Bi-directional Point-to-Point          LEC <==> BUS
    0:87    Uni-directoinal Point-to-Multipoint    LEC <==O BUS
```

These four connections must be maintained while the LE Client is a member of the ELAN. If any connection is lost, the other connections will be released, and the LEC initialization processes must be repeated for the LEC to become operational again.

8.5.3 Examples of LEC initialization failures

When entstat or tokstat shows that the status of your ATM adapter is in limbo, this usually means you are having an LEC initialization/ reinitialization failure. To resolve these failures you need to first determine what type of failure you are having. The majority of the problems can be determined by looking at the output of entstat/tokstat and looking at an iptrace of the LEC initialization process. The following sections provide examples of LEC Intialization failures. Each example will contain an explanation of what was seen in the iptrace and the entstat/tokstat output that identified the problem.

8.5.3.1 LECS ATM address failure

The LEC tok2 has just been configured with a TCP/IP interface tr2. When we try to ping other systems in the same subnet, we get 100 percent packet loss. We run the following command to check the status of the LEC:

```
tokstat -d tok2 | grep "Driver Flags"
```

The command results in the following output:

```
Driver Flags: Up Broadcast Limbo
```

The tokstat output shows that the LEC is currently in a Limbo state. The LEC is not operational; so, we want to see how far it got in the initialzation process. We run the following command to get the LANE specific statistics:

```
tokstat -d tok2 | grep -p "Specific Statistics"
```

Below is the resulting output of the above command:

```
ATM LAN Emulation Specific Statistics:
--------------------------------------
Emulated LAN Name: tok16_1
Local ATM Device Name: atm0
Local LAN MAC Address:
00.04.ac.ad.1e.f5
Local ATM Address:
47.00.05.80.ff.e1.00.00.00.f2.15.16.45.00.04.ac.ad.1e.f5.04
Auto Config With LECS:
Yes
LECS ATM Address:
c5.00.79.00.00.00.00.00.00.00.00.00.00.a0.3e.00.00.01.00
LES ATM Address:
00.00.00.00.00.00.00.00.00.00.00.00.00.00.00.00.00.00.00.00
General Errors: 26                 Address Deregistrations: 0
Control Timeout (sec): 120         LE_ARP Rsp Timeout (sec): 1
Max Unknown Frame Count: 1         Flush Timeout (sec): 4
Max Unknown Frame Time (sec): 1    Path Switch Delay (sec): 6
VCC Activity Timeout (sec): 1200   VCC Avg Rate (Kbps): 155000
Maximum LE_ARP Retries: 1          VCC Peak Rate (Kbps): 155000
LE_ARP Aging Timeout (sec): 300    Connect Complete Time (sec): 4
LE_ARP Forward Timeout (sec): 15   Maximum Frame Size: Unspecified
```

We have a valid local ATM address; so, switch communication is working. Auto Config is set to "Yes"; so, we will be getting our information from the LECS. The LES ATM address is all zeros; so, this tells us that we had a problem communicating with the LECS or LES. The following commands are run to gather an iptrace of the LEC initialziation process:

Bring down child devices and LEC:

```
ifconfig tr2 down
ifconfig tr2 detach
rmdev -l tr2
rmdev -l tok2
```

Start the iptrace:

```
rm /tmp/iptrc.bin
startsrc -s iptrace -a "-i at0 /tmp/iptrc.bin"
```

Bring up the LEC and child devices:

```
mkdev -l tok2
chdev -l tr2 -a state=up
sleep 20
```

Stop the iptrace:

```
stopsrc -s iptrace
```

Generate a readable report:

```
ipreport -srn /tmp/iptrc.bin > /tmp/iptrc.rpt
```

The following are the first few packets of the ipreport output that was generated (iptrc.rpt). Comments have been added to explain what is being seen in the report.

Below is an ILMI query to the service registry. If the LECS has its address in the service registry, the LEC will get the address back in the response. If the address is not registered, the LEC will get a response with a value of NULL. This query only occurs in AIX 4.3.2 and higher:

```
Packet Number 1
ATM: ====( 45 bytes transmitted on interface at0 )== 10:28:03.962131428
ATM: VC = 0.16
ILMI: Get Next Request PDU
ILMI: Request ID   = 65286
ILMI: Error Status = 0(Success)
ILMI: Object ID    =
1.3.6.1.4.1.353.2.8.1.1.3.0.0(atmfSrvcRegATMAddress)
ILMI:     Value    = <Null>
```

The first ILMI query returned a NULL response; so, there are no LECS ATM addresses registered in the service registry. This means the LECS will have to be configured with the well-known ATM addresses or the well-known PVC:

```
Packet Number 2
```

```
ATM: ====( 45 bytes received on interface at0 )== 10:28:03.966424887
ATM: VC = 0.16
ILMI: Get Response PDU
ILMI: Request ID   = 65286
ILMI: Error Status = 2(No such Name)
ILMI: Object ID    =
1.3.6.1.4.1.353.2.8.1.1.3.0.0(atmfSrvcRegATMAddress)
ILMI:     Value    = <Null>
```

Below is the start of the point-to-point SVC setup to the LECS. By looking at the called party ATM address we can see that the setup is done to the LECS new-style, well-known ATM address. Also, we want to make note that the call reference value for this connection is 103:

```
Packet Number 3
ATM: ====( 116 bytes transmitted on interface at0 )===
10:28:03.977034875
ATM: VC = 0.5
Q2931: 05(Setup)
Q2931: Call Ref Value  = 1038
Q2931: Message Len     = 103
Q2931:
Q2931: IE Element      = 58(AAL Parameters)
Q2931:   AAL Type      = AAL type 5
Q2931:   Fd Max SDU sz = 1516
Q2931:   Bk Max SDU sz = 1516
Q2931:   SSCS Type     = Null
Q2931: IE Element      = 59(ATM Traffic Descriptor)
Q2931:   Fwd Peak Cell Rate (CLP : 0+1)
Q2931:                 = 413334 cells/sec (175253616 bps)
Q2931:   Bkwd Peak Cell Rate(CLP : 0+1)
Q2931:                 = 413333 cells/sec (175253192 bps)
Q2931:   Best Eff Ind  = 190
Q2931: IE Element      = 5e(Broadband Bearer Capability)
Q2931:   Bearer Class  = BCOB-X
Q2931:   Traffic Type  = No Indication
Q2931:   Susc. to clip = Susceptible to Clipping
Q2931:   Conn Type     = Point-to-Point
Q2931: IE Element      = 5f(Broadband Low Layer Information)
Q2931: IE Element      = 70(Called Party Number)
Q2931:   Type of Num   = 1 (International)
Q2931:   Addressing    = 2 (ATM Endsystem Address)
Q2931:   Address       =
Q2931:      c5 0 79 0 0 0 0 0 0 0 0 0 0 0 a0 3e 0 0 1 0
Q2931: IE Element      = 6c(Calling Party Number)
Q2931:   Type of Num   = 1 (International)
Q2931:   Addressing    = 2 (ATM Endsystem Address)
```

```
Q2931:   Address       =
Q2931:      47 0 5 80 ff e1 0 0 0 f2 15 16 45 0 4 ac ad 1e f5 4
Q2931: IE Element      = 5c(QoS Parameter)
Q2931:   Coding Std    = 00(ITU-T Standardized)
Q2931:   QoS Class     = QoS Class 0(Unspecified)
Q2931:   QoS Class     = QoS Class 0(Unspecified)
```

The response back is a release with a reason of “Unallocated Number”. This means that the ATM address is valid, but there is no device registered with that ATM address. This tells us there is no LECS currently configured with the new-style, well-known address:

```
Packet Number 4
ATM: ====( 20 bytes received on interface at0 )== 10:28:03.980772201
ATM: VC = 0.5
Q2931: 5A(Release Complete)
Q2931: Call Ref Value  = 1038
Q2931: Message Len     = 7
Q2931:
Q2931: IE Element      = 8(Cause)
Q2931: Cause Value     = 1(Unallocated Number)
Q2931:    IE Element   = 4(Unknown)
```

Looking at the called party ATM address, we can see that this is an SVC setup to LECS old-style, well-known address:

```
Packet Number 5
ATM: ====( 116 bytes transmitted on interface at0 )== 10:28:04.972608527
ATM: VC = 0.5
09 03 00 04 0f 05 80 00 67 58 80 00 09 05 8c 05 ec 81 05 ec
84 00 59 80 00 09 84 06 4e 96 85 06 4e 95 be 5e 80 00 03 10
80 80 5f 80 00 09 6b 40 80 80 00 a0 3e 00 01 70 80 00 15 82
47 00 79 00 00 00 00 00 00 00 00 00 00 00 a0 3e 00 00 01 00
6c 80 00 16 02 80 47 00 05 80 ff e1 00 00 00 f2 15 16 45 00
04 ac ad 1e f5 04 5c 80 00 02 00 00 08 00 4e ed
Q2931: 05(Setup)
Q2931: Call Ref Value  = 1039
Q2931: Message Len     = 103
Q2931:
Q2931: IE Element      = 58(AAL Parameters)
Q2931:   AAL Type      = AAL type 5
Q2931:   Fd Max SDU sz = 1516
Q2931:   Bk Max SDU sz = 1516
Q2931:   SSCS Type     = Null
Q2931: IE Element      = 59(ATM Traffic Descriptor)
Q2931:   Fwd Peak Cell Rate (CLP : 0+1)
Q2931:                 = 413334 cells/sec (175253616 bps)
Q2931:   Bkwd Peak Cell Rate(CLP : 0+1)
```

```
Q2931:                 = 413333 cells/sec (175253192 bps)
Q2931:   Best Eff Ind  = 190
Q2931: IE Element      = 5e(Broadband Bearer Capability)
Q2931:   Bearer Class  = BCOB-X
Q2931:   Traffic Type  = No Indication
Q2931:   Susc. to clip = Susceptible to Clipping
Q2931:   Conn Type     = Point-to-Point
Q2931: IE Element      = 5f(Broadband Low Layer Information)
Q2931: IE Element      = 70(Called Party Number)
Q2931:   Type of Num   = 1 (International)
Q2931:   Addressing    = 2 (ATM Endsystem Address)
Q2931:   Address       =
Q2931:      47 0 79 0 0 0 0 0 0 0 0 0 0 0 a0 3e 0 0 1 0
Q2931: IE Element      = 6c(Calling Party Number)
Q2931:   Type of Num   = 1 (International)
Q2931:   Addressing    = 2 (ATM Endsystem Address)
Q2931:   Address       =
Q2931:      47 0 5 80 ff e1 0 0 0 f2 15 16 45 0 4 ac ad 1e f5 4
Q2931: IE Element      = 5c(QoS Parameter)
Q2931:   Coding Std    = 00(ITU-T Standardized)
Q2931:   QoS Class     = QoS Class 0(Unspecified)
Q2931:   QoS Class     = QoS Class 0(Unspecified)
```

The response back is a release with a reason of “Unallocated Number”. This means that the ATM address is valid, but there is no device registered with that ATM address. This tells us there is no LECS currently configured with the old-style, well-known address:

```
Packet Number 6
ATM: ====( 20 bytes received on interface at0 )== 10:28:04.976357419
ATM: VC = 0.5
Q2931: 5A(Release Complete)
Q2931: Call Ref Value  = 1039
Q2931: Message Len     = 7
Q2931:
Q2931: IE Element      = 8(Cause)
Q2931: Cause Value     = 1(Unallocated Number)
Q2931:    IE Element   = 4(Unknown)
```

From the above iptrace, we can see that there are no LECS ATM addresses in the ILMI Service Registry, no LECS configured with the LECS well-known ATM address, and no LECS configured with the well-known PVC. We know there is no well-known PVC because the LECS would have sent a configure request across it if it had existed.

So, the problem is that the LEC has been configured so that it can Auto Config using the LECS, but the LECS ATM address cannot be obtained automatically. To resolve the problem, the LECS configuration needs to be fixed so that the LEC can find the LECS by one of the following means:

1. ILMI query of the service registry
2. LECS well-known ATM address
3. LECS well-known PVC

Another solution would be to enter in the ATM address of the LECS or turn off Auto Config and enter in the LES ATM address.

8.5.3.2 LECS configure request failure

The LEC tok2 has just been configured with a TCP/IP interface tr2. When we try to ping other systems in the same subnet, we get 100 percent packet loss. We then run the following command to check the status of the LEC:

```
tokstat -d tok2 | grep "Driver Flags"
```

The command results in the following output:

```
Driver Flags: Up Broadcast Limbo
```

The tokstat output shows that the LEC is currently in a Limbo state. The LEC is not operational; so, we want to see how far it got in the initialization process. We run the following command to get the LANE specific statistics:

```
tokstat -d tok2 | grep -p "Specific Statistics"
```

Below is the resulting output of the above command:

```
ATM LAN Emulation Specific Statistics:
--------------------------------------
Emulated LAN Name: tok16_1
Local ATM Device Name: atm0
Local LAN MAC Address:
00.04.ac.ad.1e.f5
Local ATM Address:
47.00.05.80.ff.e1.00.00.00.f2.15.16.45.00.04.ac.ad.1e.f5.04
Auto Config With LECS:
Yes
LECS ATM Address:
47.00.79.00.00.00.00.00.00.00.00.00.00.00.a0.3e.00.00.01.00
LES ATM Address:
00.00.00.00.00.00.00.00.00.00.00.00.00.00.00.00.00.00.00.00
General Errors: 11                 Address Deregistrations: 0
Control Timeout (sec): 120         LE_ARP Rsp Timeout (sec): 1
Max Unknown Frame Count: 1         Flush Timeout (sec): 4
Max Unknown Frame Time (sec): 1    Path Switch Delay (sec): 6
VCC Activity Timeout (sec): 1200   VCC Avg Rate (Kbps): 155000
```

```
Maximum LE_ARP Retries: 1          VCC Peak Rate (Kbps): 155000
LE_ARP Aging Timeout (sec): 300    Connect Complete Time (sec): 4
LE_ARP Forward Timeout (sec): 15   Maximum Frame Size: Unspecified
```

We have a valid local ATM address; so, switch communication is working. Auto Config is set to “Yes”; so, we will be getting our information from the LECS. The LES ATM address is all zeros; so, this tells us that we had a problem communicating with the LECS or LES. The following commands are run to gather an iptrace of the LEC initialziation process:

Bring down child devices and LEC:

```
ifconfig tr2 down
ifconfig tr2 detach
rmdev -l tr2
rmdev -l tok2
```

Start the iptrace:

```
rm /tmp/iptrc.bin
startsrc -s iptrace -a “-i at0 /tmp/iptrc.bin”
```

Bring up the LEC and child devices:

```
mkdev -l tok2
chdev -l tr2 -a state=up
sleep 20
```

Stop the iptrace:

```
stopsrc -s iptrace
```

Generate a readable report:

```
ipreport -srn /tmp/iptrc.bin > /tmp/iptrc.rpt
```

The following are the first few packets of the ipreport output that was generated (iptrc.rpt). Comments have been added to explain what is being seen in the ipreport.

Below is an ILMI query to the service registry. If the LECS has its address in the service registry, the LEC will get the address back in the response. If the address is not registered, the LEC will get a response with a value of NULL. This query only occurs in AIX 4.3.2 and higher:

```
Packet Number 1
ATM: ====( 45 bytes transmitted on interface at0 )== 13:19:23.181965190
ATM: VC = 0.16
ILMI: Get Next Request PDU
ILMI: Request ID   = 65286
```

```
ILMI: Error Status = 0(Success)
ILMI: Object ID    =
1.3.6.1.4.1.353.2.8.1.1.3.0.0(atmfSrvcRegATMAddress)
ILMI:     Value    = <Null>
```

Below is the response, and we have a value filled in; so, we will use this address to connect to the LECS. If the address was NULL, we would try and connect to the LECS using the well-known LECS ATM address:

```
Packet Number 2
ATM: ====( 77 bytes received on interface at0 )== 13:19:23.185508303
ATM: VC = 0.16
ILMI: Get Response PDU
ILMI: Request ID   = 65286
ILMI: Error Status = 0(Success)
ILMI: Object ID    = 1.3.6.1.4.1.353.2.8.1.1.3.0.10.1.3.
ILMI:                6.1.4.1.353.1.5.1.1
ILMI:                (atmfSrvcRegATMAddress)
ILMI:     Value    = 47 0 5 80 ff e1 0 0 0 f2 15 16 45 0 6 29 b9 ae 8d 0
```

There can be more than one LECS that the LE Client can use. The LE Client will keep querying until it gets all the addresses. The next query below returned NULL; so, there was only one LECS address register:

```
Packet Number 3
ATM: ====( 57 bytes transmitted on interface at0 )== 13:19:23.185711479
ATM: VC = 0.16
ILMI: Get Next Request PDU
ILMI: Request ID   = 65286
ILMI: Error Status = 0(Success)
ILMI: Object ID    = 1.3.6.1.4.1.353.2.8.1.1.3.0.10.1.3.
ILMI:                6.1.4.1.353.1.5.1.1
ILMI:                (atmfSrvcRegATMAddress)
ILMI:     Value    = <Null>
```

Below is the response of the ILMI query, and it returned a value of NULL. This means there are no more LECS ATM addresses in the service registry:

```
Packet Number 4
ATM: ====( 57 bytes received on interface at0 )== 13:19:23.190056576
ATM: VC = 0.16
ILMI: Get Response PDU
ILMI: Request ID   = 65286
ILMI: Error Status = 2(No such Name)
ILMI: Object ID    = 1.3.6.1.4.1.353.2.8.1.1.3.0.10.1.3.
ILMI:                6.1.4.1.353.1.5.1.1
ILMI:                (atmfSrvcRegATMAddress)
ILMI:     Value    = <Null>
```

Below is the start of the point-to-point SVC setup to the LECS. The setup is done to the ATM address returned in the ILMI Get response. Also, we want # to make note that the call reference value for this connection is 103:

```
Packet Number 5
ATM: ====( 116 bytes transmitted on interface at0 )===
13:19:23.200605361
ATM: VC = 0.5
Q2931: 05(Setup)
Q2931: Call Ref Value  = 189
Q2931: Message Len     = 103
Q2931:
Q2931: IE Element      = 58(AAL Parameters)
Q2931:   AAL Type      = AAL type 5
Q2931:   Fd Max SDU sz = 1516
Q2931:   Bk Max SDU sz = 1516
Q2931:   SSCS Type     = Null
Q2931: IE Element      = 59(ATM Traffic Descriptor)
Q2931:   Fwd Peak Cell Rate (CLP : 0+1)
Q2931:                 = 413334 cells/sec (175253616 bps)
Q2931:   Bkwd Peak Cell Rate(CLP : 0+1)
Q2931:                 = 413333 cells/sec (175253192 bps)
Q2931:   Best Eff Ind  = 190
Q2931: IE Element      = 5e(Broadband Bearer Capability)
Q2931:   Bearer Class  = BCOB-X
Q2931:   Traffic Type  = No Indication
Q2931:   Susc. to clip = Susceptible to Clipping
Q2931:   Conn Type     = Point-to-Point
Q2931: IE Element      = 5f(Broadband Low Layer Information)
Q2931: IE Element      = 70(Called Party Number)
Q2931:   Type of Num   = 1 (International)
Q2931:   Addressing    = 2 (ATM Endsystem Address)
Q2931:   Address       =
Q2931:      47 0 5 80 ff e1 0 0 0 f2 15 16 45 0 6 29 b9 ae 8d 0
Q2931: IE Element      = 6c(Calling Party Number)
Q2931:   Type of Num   = 1 (International)
Q2931:   Addressing    = 2 (ATM Endsystem Address)
Q2931:   Address       =
Q2931:      47 0 5 80 ff e1 0 0 0 f2 15 16 45 0 4 ac ad 1e f5 4
Q2931: IE Element      = 5c(QoS Parameter)
Q2931:   Coding Std    = 00(ITU-T Standardized)
Q2931:   QoS Class     = QoS Class 0(Unspecified)
Q2931:   QoS Class     = QoS Class 0(Unspecified)
```

SVC setup continues. We know this “Call Proceeding” message belongs to our SVC Setup to the LECS because it has the same call reference value as the setup call. From the below information, we now know that the SVC being set up will have a VCC of 0:199 (VPI:VCI):

```
Packet Number 6
ATM: ====( 24 bytes received on interface at0 )== 13:19:23.205554061
ATM: VC = 0.5
Q2931: 02(Call Proceeding)
Q2931: Call Ref Value  = 189
Q2931: Message Len     = 9
Q2931:
Q2931: IE Element      = 5a(Connection ID)
Q2931:   VPI           = 0
Q2931:   VCI           = 199
```

SVC setup continues. Again, the call reference value is 189; so, we know this “Connect” message is tied with the LECS SVC setup:

```
Packet Number 7
ATM: ====( 24 bytes received on interface at0 )== 13:19:23.220085347
ATM: VC = 0.5
Q2931: 07(Connect)
Q2931: Call Ref Value  = 189
Q2931: Message Len     = 9
Q2931:
Q2931: IE Element      = 5a(Connection ID)
Q2931:   VPI           = 0
Q2931:   VCI           = 199
```

The LEC has sent a “Connect Acknowledge” back; so, at this point, the SVC setup to the LECS is complete. We now have an SVC connection to the LECS with a VCC of 0:199 and a call reference value of 189:

```
Packet Number 8
ATM: ====( 16 bytes transmitted on interface at0 )=== 13:19:23.220250356
ATM: VC = 0.5
Q2931: 0F(Connect Acknowledge)
Q2931: Call Ref Value  = 189
Q2931: Message Len     = 0
```

The configure request is sent to the LECS:

```
Packet Number 9
ATM: ====( 108 bytes transmitted on interface at0 )===
13:19:23.230486809
ATM: VC = 0.199
LANE CTRL CONFIGURE_REQUEST
LANE: Status        = 0
LANE: Trans ID      = 0
LANE: Request ID    = 0
```

```
LANE: Src LAN Dest:
LANE:   Tag         = 1
LANE:   MAC Addr    = 0 4 ac ad 1e f5
LANE: Tar LAN Dest:
LANE:   Tag         = 0
LANE:   MAC Addr    = 0 0 0 0 0 0
LANE: Src ATM ADDR  = 47 0 5 80 ff e1 0 0 0 f2 15 16 45 0 4 ac ad 1e f5 4
LANE: Tar ATM ADDR  = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
LANE: ELAN Name SZ  = 7
LANE: ELAN Name     = tok16_1
```

Our configure request has been sent; so, we should expect to see a configure response back from the LECS. We received the below configure response from the LECS. Looking up the status code in the ATM forum's LANE specification, we find that this means the client is not recognized:

```
Packet Number 10
ATM: ====( 108 bytes received on interface at0 )== 13:19:23.230829622
ATM: VC = 0.199
LANE CTRL CONFIGURE_RESPONSE
LANE: Status        = 20
LANE: Trans ID      = 0
LANE: Request ID    = 0
LANE: Src LAN Dest:
LANE:   Tag         = 1
LANE:   MAC Addr    = 0 4 ac ad 1e f5
LANE: Tar LAN Dest:
LANE:   Tag         = 0
LANE:   MAC Addr    = 0 0 0 0 0 0
LANE: Src ATM ADDR  = 47 0 5 80 ff e1 0 0 0 f2 15 16 45 0 4 ac ad 1e f5 4
LANE: Tar ATM ADDR  = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
LANE: ELAN Name SZ  = 7
LANE: ELAN Name     = tok16_1
```

We see a release with a call reference value of 189. Looking back at our notes, we see that is our SVC connection to the LECS. This is normal. Once the LEC has its configuration information, it no longer needs to talk the LECS; so, the connection is released:

```
Packet Number 11
ATM: ====( 20 bytes transmitted on interface at0 )== 13:19:24.192202366
ATM: VC = 0.5
Q2931: 4D(Release)
Q2931: Call Ref Value  = 189
Q2931: Message Len     = 6
Q2931:
Q2931: IE Element      = 8(Cause)
```

```
Q2931: Cause Value     = 31(Normal, Unspecified)
```

We were unable to configure using the LECS returned by ILMI; so, we check if there is LECS using the new-style, well-known ATM address that we can use. Now, set up LECS using the new-style, well-known ATM address.

```
Packet Number 12
ATM: ====( 116 bytes transmitted on interface at0 )== 13:19:24.192505011
ATM: VC = 0.5
Q2931: 05(Setup)
Q2931: Call Ref Value  = 188
Q2931: Message Len     = 103
Q2931:
Q2931: IE Element      = 58(AAL Parameters)
Q2931:   AAL Type      = AAL type 5
Q2931:   Fd Max SDU sz = 1516
Q2931:   Bk Max SDU sz = 1516
Q2931:   SSCS Type     = Null
Q2931: IE Element      = 59(ATM Traffic Descriptor)
Q2931:   Fwd Peak Cell Rate (CLP : 0+1)
Q2931:                 = 413334 cells/sec (175253616 bps)
Q2931:   Bkwd Peak Cell Rate(CLP : 0+1)
Q2931:                 = 413333 cells/sec (175253192 bps)
Q2931:   Best Eff Ind  = 190
Q2931: IE Element      = 5e(Broadband Bearer Capability)
Q2931:   Bearer Class  = BCOB-X
Q2931:   Traffic Type  = No Indication
Q2931:   Susc. to clip = Susceptible to Clipping
Q2931:   Conn Type     = Point-to-Point
Q2931: IE Element      = 5f(Broadband Low Layer Information)
Q2931: IE Element      = 70(Called Party Number)
Q2931:   Type of Num   = 1 (International)
Q2931:   Addressing    = 2 (ATM Endsystem Address)
Q2931:   Address       =
Q2931:      c5 0 79 0 0 0 0 0 0 0 0 0 0 0 a0 3e 0 0 1 0
Q2931: IE Element      = 6c(Calling Party Number)
Q2931:   Type of Num   = 1 (International)
Q2931:   Addressing    = 2 (ATM Endsystem Address)
Q2931:   Address       =
Q2931:      47 0 5 80 ff e1 0 0 0 f2 15 16 45 0 4 ac ad 1e f5 4
Q2931: IE Element      = 5c(QoS Parameter)
Q2931:   Coding Std    = 00(ITU-T Standardized)
Q2931:   QoS Class     = QoS Class 0(Unspecified)
Q2931:   QoS Class     = QoS Class 0(Unspecified)
```

We receive the expected “Release Complete” for the release message we sent for the LECS connection. We know this by matching up the call reference numbers again. The SVC to the LECS is now gone:

```
Packet Number 13
ATM: ====( 20 bytes received on interface at0 )== 13:19:24.196761917
ATM: VC = 0.5
Q2931: 5A(Release Complete)
Q2931: Call Ref Value  = 189
Q2931: Message Len     = 6
Q2931:
Q2931: IE Element      = 8(Cause)
Q2931: Cause Value     = 31(Normal, Unspecified)
```

The response back for our setup is a release with a reason of “Unallocated Number”. This means that the ATM address is valid, but there is no device registered with that ATM address. This tells us there is no LECS currently configured with the new-style, well-known address:

```
Packet Number 14
ATM: ====( 20 bytes received on interface at0 )== 13:19:24.200507073
ATM: VC = 0.5
Q2931: 5A(Release Complete)
Q2931: Call Ref Value  = 188
Q2931: Message Len     = 7
Q2931:
Q2931: IE Element      = 8(Cause)
Q2931: Cause Value     = 1(Unallocated Number)
Q2931:    IE Element   = 4(Unknown)
```

Now check if there is an LECS using the old-style, well-known address:

```
Packet Number 15
ATM: ====( 116 bytes transmitted on interface at0 )== 13:19:26.192769078
ATM: VC = 0.5
Q2931: 05(Setup)
Q2931: Call Ref Value  = 187
Q2931: Message Len     = 103
Q2931:
Q2931: IE Element      = 58(AAL Parameters)
Q2931:   AAL Type      = AAL type 5
Q2931:   Fd Max SDU sz = 1516
Q2931:   Bk Max SDU sz = 1516
Q2931:   SSCS Type     = Null
Q2931: IE Element      = 59(ATM Traffic Descriptor)
Q2931:   Fwd Peak Cell Rate (CLP : 0+1)
Q2931:                 = 413334 cells/sec (175253616 bps)
Q2931:   Bkwd Peak Cell Rate(CLP : 0+1)
```

```
Q2931:                 = 413333 cells/sec (175253192 bps)
Q2931:   Best Eff Ind  = 190
Q2931: IE Element      = 5e(Broadband Bearer Capability)
Q2931:   Bearer Class  = BCOB-X
Q2931:   Traffic Type  = No Indication
Q2931:   Susc. to clip = Susceptible to Clipping
Q2931:   Conn Type     = Point-to-Point
Q2931: IE Element      = 5f(Broadband Low Layer Information)
Q2931: IE Element      = 70(Called Party Number)
Q2931:   Type of Num   = 1 (International)
Q2931:   Addressing    = 2 (ATM Endsystem Address)
Q2931:   Address       =
Q2931:      47 0 79 0 0 0 0 0 0 0 0 0 0 0 a0 3e 0 0 1 0
Q2931: IE Element      = 6c(Calling Party Number)
Q2931:   Type of Num   = 1 (International)
Q2931:   Addressing    = 2 (ATM Endsystem Address)
Q2931:   Address       =
Q2931:      47 0 5 80 ff e1 0 0 0 f2 15 16 45 0 4 ac ad 1e f5 4
Q2931: IE Element      = 5c(QoS Parameter)
Q2931:   Coding Std    = 00(ITU-T Standardized)
Q2931:   QoS Class     = QoS Class 0(Unspecified)
Q2931:   QoS Class     = QoS Class 0(Unspecified)
```

LECS was found by using the old-style, well-known address. The SVC setup proceeds for the Call Reference Value 187, and the VCC is 0:201:

```
Packet Number 16
ATM: ====( 24 bytes received on interface at0 )== 13:19:26.197794306
ATM: VC = 0.5
Q2931: 02(Call Proceeding)
Q2931: Call Ref Value  = 187
Q2931: Message Len     = 9
Q2931:
Q2931: IE Element      = 5a(Connection ID)
Q2931:   VPI           = 0
Q2931:   VCI           = 201
```

The SVC setup for call reference value 187 continues:

```
Packet Number 17
ATM: ====( 24 bytes received on interface at0 )== 13:19:26.210410232
ATM: VC = 0.5
Q2931: 07(Connect)
Q2931: Call Ref Value  = 187
Q2931: Message Len     = 9
Q2931:
Q2931: IE Element      = 5a(Connection ID)
Q2931:   VPI           = 0
```

```
Q2931:   VCI           = 201
```

The SVC setup to LECS using old-style, well-known address is complete. The call reference value is 187, and the VCC is 0:201:

```
Packet Number 18
ATM: ====( 16 bytes transmitted on interface at0 )== 13:19:26.210616204
ATM: VC = 0.5
Q2931: 0F(Connect Acknowledge)
Q2931: Call Ref Value  = 187
Q2931: Message Len     = 0
```

The configure request is sent over VCC 0:201. The configuration request is for elan_name tok16_1:

```
Packet Number 19
ATM: ====( 108 bytes transmitted on interface at0 )===
13:19:26.220853452
ATM: VC = 0.201
LANE CTRL CONFIGURE_REQUEST
LANE: Status        = 0
LANE: Trans ID      = 0
LANE: Request ID    = 0
LANE: Src LAN Dest:
LANE:   Tag         = 1
LANE:   MAC Addr    = 0 4 ac ad 1e f5
LANE: Tar LAN Dest:
LANE:   Tag         = 0
LANE:   MAC Addr    = 0 0 0 0 0 0
LANE: Src ATM ADDR  = 47 0 5 80 ff e1 0 0 0 f2 15 16 45 0 4 ac ad 1e f5 4
LANE: Tar ATM ADDR  = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
LANE: ELAN Name SZ  = 7
LANE: ELAN Name     = tok16_1
```

We received a configure response with a status of 20. Looking up the status code in the ATM forum’s LANE specification, we find that this means the client is not recognized:

```
Packet Number 20
ATM: ====( 108 bytes received on interface at0 )== 13:19:26.226483199
ATM: VC = 0.201
LANE CTRL CONFIGURE_RESPONSE
LANE: Status        = 20
LANE: Trans ID      = 0
LANE: Request ID    = 0
LANE: Src LAN Dest:
LANE:   Tag         = 1
LANE:   MAC Addr    = 0 4 ac ad 1e f5
```

```
LANE: Tar LAN Dest:
LANE:   Tag         = 0
LANE:   MAC Addr    = 0 0 0 0 0 0
LANE: Src ATM ADDR  = 47 0 5 80 ff e1 0 0 0 f2 15 16 45 0 4 ac ad 1e f5 4
LANE: Tar ATM ADDR  = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
LANE: ELAN Name SZ  = 7
LANE: ELAN Name     = tok16_1
```

The configure request failed to LECS using old-style, well-known address; so, we release the connection:

```
Packet Number 21
ATM: ====( 20 bytes transmitted on interface at0 )== 13:19:29.192883992
ATM: VC = 0.5
Q2931: 4D(Release)
Q2931: Call Ref Value  = 187
Q2931: Message Len     = 6
Q2931:
Q2931: IE Element      = 8(Cause)
Q2931: Cause Value     = 31(Normal, Unspecified)
```

The release of VCC 0:201 is compete:

```
Packet Number 22
ATM: ====( 20 bytes received on interface at0 )== 13:19:29.197138994
ATM: VC = 0.5
Q2931: 5A(Release Complete)
Q2931: Call Ref Value  = 187
Q2931: Message Len     = 6
Q2931:
Q2931: IE Element      = 8(Cause)
Q2931: Cause Value     = 31(Normal, Unspecified)
```

The above iptrace shows the LEC made a configure request to two LECS. One LECS was obtained by doing an ILMI query of the service registry and the other by making a connection to the LECS old-style, well-known address. In both cases, the configure request was rejected because the LEC was not recognized by the LECS. To resolve this problem, the systems administrator needs to check the LECS configuration to see why it is rejecting the LEC. The request only contains a LAN MAC address, a SRC ATM Address, and an ELAN Name. Looking at the LECS, there are no policies to reject the LEC based on LAN or ATM addresses. The only thing left can be that the problem is the ELAN Name specified. Either the ELAN name we specified was not correct, or the LECS does have the ELAN name, tok16_1, in its database. We ask the LECS administrator to see if the ELAN name is configured in the LECS, and it is not. The administrator adds tok16_1 to the LECS ELAN name policy, and the problem is resolved.

8.5.3.3 PDU/Max frame size failure

The LEC tok2 has just been configured with a TCP/IP interface tr2. When we try to ping other systems in the same subnet, we get 100 percent packet loss. We run the following command to check the status of the LEC:

```
tokstat -d tok2 | grep “Driver Flags”
```

The command results in the following output:

```
Driver Flags: Up Broadcast Limbo
```

The tokstat output shows that the LEC is currently in a Limbo state. The LEC is not operational; so, we want to see how far it got in the initialzation process. We run the following command to get the LANE specific statistics:

```
tokstat -d tok2 | grep -p “Specific Statistics”
```

Below is the resulting output of the above command:

```
ATM LAN Emulation Specific Statistics:
--------------------------------------
Emulated LAN Name: tok16_1
Local ATM Device Name: atm0
Local LAN MAC Address:
08.00.5a.99.a8.14
Local ATM Address:
47.00.05.80.ff.e1.00.00.00.f2.15.16.45.08.00.5a.99.a8.14.03
Auto Config With LECS:
Yes
LECS ATM Address:
00.00.00.11.00.00.00.00.00.00.00.00.00.00.00.00.00.00.00.00
LES ATM Address:
00.00.00.00.00.00.00.00.00.00.00.00.00.00.00.00.00.00.00.00
General Errors: 17                  Address Deregistrations: 0
Control Timeout (sec): 120          LE_ARP Rsp Timeout (sec): 1
Max Unknown Frame Count: 1          Flush Timeout (sec): 4
Max Unknown Frame Time (sec): 1     Path Switch Delay (sec): 6
VCC Activity Timeout (sec): 1200    VCC Avg Rate (Kbps): 155000
Maximum LE_ARP Retries: 1           VCC Peak Rate (Kbps): 155000
LE_ARP Aging Timeout (sec): 300     Connect Complete Time (sec): 4
LE_ARP Forward Timeout (sec): 15    Maximum Frame Size: Unspecified
```

We have a valid, local ATM address; so, switch communication is working. Auto Config is set to “Yes”; so, we will be getting our information from the LECS. The LES ATM address is all zeros; so, this tells us that we had a problem communicating with the LECS or LES. The following commands are run to gather an iptrace of the LEC Initialziation process:

Bring down child devices and LEC:

```
ifconfig tr2 down
ifconfig tr2 detach
rmdev -l tr2
rmdev -l tok2
```

Start the iptrace:

```
rm /tmp/iptrc.bin
startsrc -s iptrace -a "-i at0 /tmp/iptrc.bin"
```

Bring up the LEC and child devices:

```
mkdev -l tok2
chdev -l tr2 -a state=up
sleep 20
```

Stop the iptrace:

```
stopsrc -s iptrace
```

Generate a readable report:

```
ipreport -srn /tmp/iptrc.bin > /tmp/iptrc.rpt
```

The following are the first few packets of the ipreport output that was generated (iptrc.rpt). Comments have been added to explain what is being seen in the ipreport.

Below is an ILMI query to the service registry. If the LECS has its address in the service registry the LEC will get the address back in the response. If the address is not registered, the LEC will get a response with a value of NULL. This query only occurs in AIX 4.3.2 and higher:

```
Packet Number 1
ATM: ====( 45 bytes transmitted on interface at0 )== 16:06:41.079202560
ATM: VC = 0.16
ILMI: Get Next Request PDU
ILMI: Request ID   = 65286
ILMI: Error Status = 0(Success)
ILMI: Object ID    =
1.3.6.1.4.1.353.2.8.1.1.3.0.0(atmfSrvcRegATMAddress)
ILMI:     Value    = <Null>
```

Below is the response, and we have a value filled in; so, we will use this address to connect to the LECS. If the address was NULL, we would try and connect to the LECS using the well-known LECS ATM address:

```
Packet Number 2
```

```
ATM: ====( 77 bytes received on interface at0 )== 16:06:41.083205120
ATM: VC = 0.16
ILMI: Get Response PDU
ILMI: Request ID   = 65286
ILMI: Error Status = 0(Success)
ILMI: Object ID    = 1.3.6.1.4.1.353.2.8.1.1.3.0.10.1.3.
ILMI:                6.1.4.1.353.1.5.1.1
ILMI:                (atmfSrvcRegATMAddress)
ILMI:     Value    = 47 0 5 80 ff e1 0 0 0 f2 15 16 45 0 6 29 b9 ae 8d 0
```

There can be more than one LECS that the LE Client can use. The LE Client will keep querying until it gets all the addresses. The next query below returned NULL; so, there was only one LECS address register:

```
Packet Number 3
ATM: ====( 57 bytes transmitted on interface at0 )== 16:06:41.083673856
ATM: VC = 0.16
ILMI: Get Next Request PDU
ILMI: Request ID   = 65286
ILMI: Error Status = 0(Success)
ILMI: Object ID    = 1.3.6.1.4.1.353.2.8.1.1.3.0.10.1.3.
ILMI:                6.1.4.1.353.1.5.1.1
ILMI:                (atmfSrvcRegATMAddress)
ILMI:     Value    = <Null>
```

Below is the response of the ILMI query, and it returned a value of NULL. This means there are no more LECS ATM addresses in the service registry:

```
Packet Number 4
ATM: ====( 57 bytes received on interface at0 )== 16:06:41.088693248
ATM: VC = 0.16
ILMI: Get Response PDU
ILMI: Request ID   = 65286
ILMI: Error Status = 2(No such Name)
ILMI: Object ID    = 1.3.6.1.4.1.353.2.8.1.1.3.0.10.1.3.
ILMI:                6.1.4.1.353.1.5.1.1
ILMI:                (atmfSrvcRegATMAddress)
ILMI:     Value    = <Null>
```

Below is the start of the point-to-point SVC setup to the LECS. The setup is done to the ATM address returned in the ILMI Get response. Also, we want to make note that the call reference value for this connection is 128:

```
Packet Number 5
ATM: ====( 116 bytes transmitted on interface at0 )== 16:06:41.089971584
ATM: VC = 0.5
Q2931: 05(Setup)
Q2931: Call Ref Value  = 128
```

```
Q2931: Message Len     = 103
Q2931:
Q2931: IE Element      = 58(AAL Parameters)
Q2931:   AAL Type      = AAL type 5
Q2931:   Fd Max SDU sz = 1516
Q2931:   Bk Max SDU sz = 1516
Q2931:   SSCS Type     = Null
Q2931: IE Element      = 59(ATM Traffic Descriptor)
Q2931:   Fwd Peak Cell Rate (CLP : 0+1)
Q2931:                 = 413334 cells/sec (175253616 bps)
Q2931:   Bkwd Peak Cell Rate(CLP : 0+1)
Q2931:                 = 413333 cells/sec (175253192 bps)
Q2931:   Best Eff Ind  = 190
Q2931: IE Element      = 5e(Broadband Bearer Capability)
Q2931:   Bearer Class  = BCOB-X
Q2931:   Traffic Type  = No Indication
Q2931:   Susc. to clip = Susceptible to Clipping
Q2931:   Conn Type     = Point-to-Point
Q2931: IE Element      = 5f(Broadband Low Layer Information)
Q2931: IE Element      = 70(Called Party Number)
Q2931:   Type of Num   = 1 (International)
Q2931:   Addressing    = 2 (ATM Endsystem Address)
Q2931:   Address       =
Q2931:      47 0 5 80 ff e1 0 0 0 f2 15 16 45 0 6 29 b9 ae 8d 0
Q2931: IE Element      = 6c(Calling Party Number)
Q2931:   Type of Num   = 1 (International)
Q2931:   Addressing    = 2 (ATM Endsystem Address)
Q2931:   Address       =
Q2931:      47 0 5 80 ff e1 0 0 0 f2 15 16 45 8 0 5a 99 a8 14 3
Q2931: IE Element      = 5c(QoS Parameter)
Q2931:   Coding Std    = 00(ITU-T Standardized)
Q2931:   QoS Class     = QoS Class 0(Unspecified)
Q2931:   QoS Class     = QoS Class 0(Unspecified)
```

SVC setup continues. We know this “Call Proceeding” message belongs to our SVC Setup to the LECS because it has the same call reference value as the setup call. From the below information, we now know that the SVC being set up will have a VCC of 0:847 (VPI:VCI):

```
Packet Number 6
ATM: ====( 24 bytes received on interface at0 )== 16:06:41.095608320
ATM: VC = 0.5
Q2931: 02(Call Proceeding)
Q2931: Call Ref Value  = 128
Q2931: Message Len     = 9
Q2931:
Q2931: IE Element      = 5a(Connection ID)
```

```
Q2931:   VPI           = 0
Q2931:   VCI           = 847
```

The SVC setup continues. Again, the call reference value is 128; so, we know this “Connect” message is tied with the LECS SVC setup:

```
Packet Number 7
ATM: ====( 24 bytes received on interface at0 )== 16:06:41.110634496
ATM: VC = 0.5
Q2931: 07(Connect)
Q2931: Call Ref Value  = 128
Q2931: Message Len     = 9
Q2931:
Q2931: IE Element      = 5a(Connection ID)
Q2931:   VPI           = 0
Q2931:   VCI           = 847
```

The LEC has sent a “Connect Acknowledge” back; so, at this point, the SVC setup to the LECS is complete. We now have an SVC connection to the LECS with a VCC of 0:847 and a call reference value of 128:

```
Packet Number 8
ATM: ====( 16 bytes transmitted on interface at0 )== 16:06:41.111094144
ATM: VC = 0.5
Q2931: 0F(Connect Acknowledge)
Q2931: Call Ref Value  = 128
Q2931: Message Len     = 0
```

There is a configure request over VCC 0:847 to join ELAN tok16_1:

```
Packet Number 9
ATM: ====( 108 bytes transmitted on interface at0 )== 16:06:41.145653504
ATM: VC = 0.847
LANE CTRL CONFIGURE_REQUEST
LANE: Status        = 0
LANE: Trans ID      = 0
LANE: Request ID    = 0
LANE: Src LAN Dest:
LANE:   Tag         = 1
LANE:   MAC Addr    = 8 0 5a 99 a8 14
LANE: Tar LAN Dest:
LANE:   Tag         = 0
LANE:   MAC Addr    = 0 0 0 0 0 0
LANE: Src ATM ADDR  = 47 0 5 80 ff e1 0 0 0 f2 15 16 45 8 0 5a 99 a8 14 3
LANE: Tar ATM ADDR  = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
LANE: ELAN Name SZ  = 7
LANE: ELAN Name     = tok16_1
```

We get a configure response with a status of zero; so, the configure request succeeded. The response contains a target ATM address of `47 0 5 80 ff e1 0 0 0 f2 15 16 45 0 6 29 b9 ae 8d 40`. This is the ATM address of the LES to which a connection needs to be established.

```
Packet Number 10
ATM: ====( 108 bytes received on interface at0 )== 16:06:41.145699200
ATM: VC = 0.847
LANE CTRL CONFIGURE_RESPONSE
LANE: Status        = 0
LANE: Trans ID      = 0
LANE: Request ID    = 0
LANE: Src LAN Dest:
LANE:   Tag         = 1
LANE:   MAC Addr    = 8 0 5a 99 a8 14
LANE: Tar LAN Dest:
LANE:   Tag         = 0
LANE:   MAC Addr    = 0 0 0 0 0 0
LANE: Src ATM ADDR  = 47 0 5 80 ff e1 0 0 0 f2 15 16 45 8 0 5a 99 a8 14 3
LANE: Tar ATM ADDR  = 47 0 5 80 ff e1 0 0 0 f2 15 16 45 0 6 29 b9 ae 8d 40
LANE: ELAN Name SZ  = 7
LANE: ELAN Name     = tok16_1
```

Looking at the called party address, we see that this is an SVC setup to the LES. The call reference value is 129:

```
Packet Number 11
ATM: ====( 116 bytes transmitted on interface at0 )== 16:06:41.146338048
ATM: VC = 0.5
Q2931: 05(Setup)
Q2931: Call Ref Value  = 129
Q2931: Message Len     = 103
Q2931:
Q2931: IE Element      = 58(AAL Parameters)
Q2931:   AAL Type      = AAL type 5
Q2931:   Fd Max SDU sz = 1516
Q2931:   Bk Max SDU sz = 1516
Q2931:   SSCS Type     = Null
Q2931: IE Element      = 59(ATM Traffic Descriptor)
Q2931:   Fwd Peak Cell Rate (CLP : 0+1)
Q2931:                 = 413334 cells/sec (175253616 bps)
Q2931:   Bkwd Peak Cell Rate(CLP : 0+1)
Q2931:                 = 413333 cells/sec (175253192 bps)
Q2931:   Best Eff Ind  = 190
Q2931: IE Element      = 5e(Broadband Bearer Capability)
Q2931:   Bearer Class  = BCOB-X
Q2931:   Traffic Type  = No Indication
```

```
Q2931:   Susc. to clip = Susceptible to Clipping
Q2931:   Conn Type     = Point-to-Point
Q2931: IE Element      = 5f(Broadband Low Layer Information)
Q2931: IE Element      = 70(Called Party Number)
Q2931:   Type of Num   = 1 (International)
Q2931:   Addressing    = 2 (ATM Endsystem Address)
Q2931:   Address       =
Q2931:      47 0 5 80 ff e1 0 0 0 f2 15 16 45 0 6 29 b9 ae 8d 40
Q2931: IE Element      = 6c(Calling Party Number)
Q2931:   Type of Num   = 1 (International)
Q2931:   Addressing    = 2 (ATM Endsystem Address)
Q2931:   Address       =
Q2931:      47 0 5 80 ff e1 0 0 0 f2 15 16 45 8 0 5a 99 a8 14 3
Q2931: IE Element      = 5c(QoS Parameter)
Q2931:   Coding Std    = 00(ITU-T Standardized)
Q2931:   QoS Class     = QoS Class 0(Unspecified)
Q2931:   QoS Class     = QoS Class 0(Unspecified)
```

Looking at the call reference value, we know that this is the SVC connection to LECS. We have our config information; so, the connection is released:

```
Packet Number 12
ATM: ====( 20 bytes transmitted on interface at0 )== 16:06:41.146790784
ATM: VC = 0.5
Q2931: 4D(Release)
Q2931: Call Ref Value  = 128
Q2931: Message Len     = 6
Q2931:
Q2931: IE Element      = 8(Cause)
Q2931: Cause Value     = 31(Normal, Unspecified)
```

The call reference value is 129; so, this is our SVC setup to the LES. The VCC will be 0:848:

```
Packet Number 13
ATM: ====( 24 bytes received on interface at0 )== 16:06:41.152329728
ATM: VC = 0.5
Q2931: 02(Call Proceeding)
Q2931: Call Ref Value  = 129
Q2931: Message Len     = 9
Q2931:
Q2931: IE Element      = 5a(Connection ID)
Q2931:   VPI           = 0
Q2931:   VCI           = 848
```

The SVC connection to the LECS is now released:

```
Packet Number 14
```

```
ATM: ====( 20 bytes received on interface at0 )== 16:06:41.156369664
ATM: VC = 0.5
Q2931: 5A(Release Complete)
Q2931: Call Ref Value  = 128
Q2931: Message Len     = 6
Q2931:
Q2931: IE Element      = 8(Cause)
Q2931: Cause Value     = 31(Normal, Unspecified)
SVC setup to LES continues.
Packet Number 15
ATM: ====( 24 bytes received on interface at0 )== 16:06:41.162593536
ATM: VC = 0.5
Q2931: 07(Connect)
Q2931: Call Ref Value  = 129
Q2931: Message Len     = 9
Q2931:
Q2931: IE Element      = 5a(Connection ID)
Q2931:   VPI           = 0
Q2931:   VCI           = 848
```

The SVC setup to the LES is complete. We now have an SVC with a call reference value of 129 and a VCC 0:848:

```
Packet Number 16
ATM: ====( 16 bytes transmitted on interface at0 )== 16:06:41.163038080
ATM: VC = 0.5
Q2931: 0F(Connect Acknowledge)
Q2931: Call Ref Value  = 129
Q2931: Message Len     = 0
```

Now that we have an SVC connection to the LES we expect our LE client to send a join request to the LES for ELAN tok16_1. The LEC will send its LAN MAC address and its ATM address in this request. You will also notice that the below join request is sent on VCC 0.848. This is the VCC that was set up to the LES:

```
Packet Number 17
ATM: ====( 108 bytes transmitted on interface at0 )== 16:06:41.196040704
ATM: VC = 0.848
LANE CTRL JOIN_REQUEST
LANE: Status        = 0
LANE: Trans ID      = 1
LANE: Request ID    = 0
LANE: Src LAN Dest:
LANE:   Tag         = 1
LANE:   MAC Addr    = 8 0 5a 99 a8 14
LANE: Tar LAN Dest:
```

```
LANE:   Tag          = 0
LANE:   MAC Addr     = 0 0 0 0 0 0
LANE: Src ATM ADDR  = 47 0 5 80 ff e1 0 0 0 f2 15 16 45 8 0 5a 99 a8 14 3
LANE: Tar ATM ADDR  = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
LANE: ELAN Name SZ  = 7
LANE: ELAN Name     = tok16_1
```

Before receiving a response to our join request the LES will set up a uni-directional SVC back to the LEC for distributing information. Looking at the calling party address, we see that the below setup is indeed coming from the LES. We also make note that the call reference value is 41258:

```
Packet Number 18
ATM: ====( 132 bytes received on interface at0 )== 16:06:41.203034368
ATM: VC = 0.5
Q2931: 05(Setup)
Q2931: Call Ref Value  = 41258
Q2931: Message Len     = 119
Q2931:
Q2931: IE Element      = 58(AAL Parameters)
Q2931:   AAL Type      = AAL type 5
Q2931:   Fd Max SDU sz = 1516
Q2931:   Bk Max SDU sz = 0
Q2931:   SSCS Type     = Null
Q2931: IE Element      = 5f(Broadband Low Layer Information)
Q2931: IE Element      = 70(Called Party Number)
Q2931:   Type of Num   = 1 (International)
Q2931:   Addressing    = 2 (ATM Endsystem Address)
Q2931:   Address       =
Q2931:      47 0 5 80 ff e1 0 0 0 f2 15 16 45 8 0 5a 99 a8 14 3
Q2931: IE Element      = 6c(Calling Party Number)
Q2931:   Type of Num   = 1 (International)
Q2931:   Addressing    = 2 (ATM Endsystem Address)
Q2931:   Address       =
Q2931:      47 0 5 80 ff e1 0 0 0 f2 15 16 45 0 6 29 b9 ae 8d 40
Q2931: IE Element      = 54(ATM Traffic Descriptor)
Q2931: IE Element      = 59(ATM Traffic Descriptor)
Q2931:   Fwd Peak Cell Rate (CLP : 0+1)
Q2931:                 = 365566 cells/sec (154999984 bps)
Q2931:   Bkwd Peak Cell Rate(CLP : 0+1)
Q2931:                 = 0 cells/sec (0 bps)
Q2931:   Best Eff Ind  = 190
Q2931: IE Element      = 5e(Broadband Bearer Capability)
Q2931:   Bearer Class  = BCOB-X
Q2931:   Traffic Type  = No Indication
Q2931:   Susc. to clip = Susceptible to Clipping
Q2931:   Conn Type     = Point-to-Multipoint
```

```
Q2931: IE Element      = 5c(QoS Parameter)
Q2931:   Coding Std    = 00(ITU-T Standardized)
Q2931:   QoS Class     = QoS Class 0(Unspecified)
Q2931:   QoS Class     = QoS Class 0(Unspecified)
Q2931: IE Element      = 5a(Connection ID)
Q2931:   VPI           = 0
Q2931:   VCI           = 849
```

The SVC setup from LES to LEC continues. The VCC will be 0:849:

```
Packet Number 19
ATM: ====( 32 bytes transmitted on interface at0 )== 16:06:41.203934080
ATM: VC = 0.5
Q2931: 02(Call Proceeding)
Q2931: Call Ref Value  = 41258
Q2931: Message Len     = 16
Q2931:
Q2931: IE Element      = 5a(Connection ID)
Q2931:   VPI           = 0
Q2931:   VCI           = 849
Q2931: IE Element      = 54(ATM Traffic Descriptor)
```

The SVC setup from LES to LEC continues:

```
Packet Number 20
ATM: ====( 32 bytes transmitted on interface at0 )== 16:06:41.204887936
ATM: VC = 0.5
Q2931: 07(Connect)
Q2931: Call Ref Value  = 41258
Q2931: Message Len     = 16
Q2931:
Q2931: IE Element      = 5a(Connection ID)
Q2931:   VPI           = 0
Q2931:   VCI           = 849
Q2931: IE Element      = 54(ATM Traffic Descriptor)
```

The SVC setup from LES to LEC is complete. The call reference value is 41258, and the VCC is 0:849:

```
Packet Number 21
ATM: ====( 16 bytes received on interface at0 )== 16:06:41.210683904
ATM: VC = 0.5
Q2931: 0F(Connect Acknowledge)
Q2931: Call Ref Value  = 41258
Q2931: Message Len     = 0
```

Now that the LES has set up its uni-directional SVC to our LEC, we expect the LES to send the LEC a join response to the LEC’s join request. Looking further in the trace we see the Join Response. We make note that the Status field is zero. A non-zero value for this field indicates that an error occurred:

```
Packet Number 22
ATM: ====( 117 bytes received on interface at0 )== 16:06:41.212888448
ATM: VC = 0.848
LANE CTRL JOIN_RESPONSE
LANE: Status        = 0
LANE: Trans ID      = 1
LANE: Request ID    = 3
LANE: Src LAN Dest:
LANE:   Tag         = 1
LANE:   MAC Addr    = 8 0 5a 99 a8 14
LANE: Tar LAN Dest:
LANE:   Tag         = 0
LANE:   MAC Addr    = 0 0 0 0 0 0
LANE: Src ATM ADDR  = 47 0 5 80 ff e1 0 0 0 f2 15 16 45 8 0 5a 99 a8 14 3
LANE: Tar ATM ADDR  = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
LANE: ELAN Name SZ  = 7
LANE: ELAN Name     = tok16_1
LANE:
LANE: Num TLVs      = 1
LANE: TLV Item      = ELAN-ID
LANE: TLV Length    = 4
LANE: TLV Value     = 0 0 0 28
```

At this point, we are expecting the LEC to send an LE_ARP for FF.FF.FF.FF.FF.FF to determine the BUS ATM address. Instead, we see that the LEC has sent a release request for its multipoint connection from the LES:

```
Packet Number 23
ATM: ====( 20 bytes transmitted on interface at0 )== 16:06:42.080637568
ATM: VC = 0.5
Q2931: 4D(Release)
Q2931: Call Ref Value  = 41258
Q2931: Message Len     = 6
Q2931:
Q2931: IE Element      = 8(Cause)
Q2931: Cause Value     = 31(Normal, Unspecified)
```

The LEC has now sent a release request of its point-to-point connection to the LES:

```
Packet Number 24
ATM: ====( 20 bytes transmitted on interface at0 )== 16:06:42.081357568
ATM: VC = 0.5
Q2931: 4D(Release)
```

```
Q2931: Call Ref Value  = 129
Q2931: Message Len     = 6
Q2931:
Q2931: IE Element      = 8(Cause)
Q2931: Cause Value     = 31(Normal, Unspecified)
```

The LES multipoint connection is released:

```
Packet Number 25
ATM: ====( 20 bytes received on interface at0 )== 16:06:42.086891904
ATM: VC = 0.5
Q2931: 5A(Release Complete)
Q2931: Call Ref Value  = 41258
Q2931: Message Len     = 6
Q2931:
Q2931: IE Element      = 8(Cause)
Q2931: Cause Value     = 31(Normal, Unspecified)
```

The point-to-point connection to the LES is released:

```
Packet Number 26
ATM: ====( 20 bytes received on interface at0 )== 16:06:42.091556864
ATM: VC = 0.5
Q2931: 5A(Release Complete)
Q2931: Call Ref Value  = 129
Q2931: Message Len     = 6
Q2931:
Q2931: IE Element      = 8(Cause)
Q2931: Cause Value     = 31(Normal, Unspecified)
```

From the above trace, we see that after receiving the join response from the LES, the LEC released its connections to the LES. At this point, we are expecting the LEC to send an LE_ARP to determine the ATM address of the BUS. The join response has a status code zero; so, this means the problem is with the LEC. Since the response was okay, this means the LEC did not like a value returned in the response, or it had a problem transitioning to the next stage of the LEC Intialization process. The only way to resolve this problem is to see what is happening internally with the LEC. To do this, we need to take a kernel trace capturing the problem. A kernel trace, however, requires access to the source code to analyze. For these types of problems, you want to contact AIX support to have the kernel trace analyzed. The LEC is most likely failing, however, because it is hitting some error condition, most error conditions are usually logged in the errlog. Before resorting to a kernel trace, we want to look at the errlog to see if there are any relevant errors. We run the following command to look at the current errlog entries:

```
errpt -a | more
```

Looking at the errpt, we find the following error occurs every time we try and configure the LEC:

```
---------------------------------------------------------------------
LABEL:          ATMLE_DEV_MTU_SIZE
IDENTIFIER:     76FB2F5C

Date/Time:       Tue Feb 29 17:00:54
Sequence Number: 574
Machine Id:      000044344800
Node Id:         lanebox
Class:           S
Type:            PERM
Resource Name:   tok2
Description
ERROR ALLOCATING MEMORY
Probable Causes
SOFTWARE DEVICE DRIVER
INTERNAL SYSTEM RESOURCE NOT AVAILABLE
TRANSMIT/RECEIVE BUFFERS
Failure Causes
SOFTWARE PROGRAM
Recommended Actions
REVIEW LINK CONFIGURATION DETAIL DATA
INCREASE SIZE OF DRIVER BUFFER
CONTACT LAN ADMINISTRATOR RESPONSIBLE FOR THIS LAN
CONTACT ATM NETWORK ADMINISTRATOR
Detail Data
FILE NAME
line: 1124 file: atmle_proto.c
SOURCE ADDRESS
0800 5A99 A814
CLIENT ID
0000 0003
SENSE DATA
18190
SENSE DATA
9188
```

The above errpt entry indicates that a device mtu was too small. Looking at the sense data, we see values of 18190 and 9188. The first value is the max frame size of the ELAN the LEC is trying to join. The second value is the ATM drivers mtu.

So, the problem is that the SDU for the ATM device is too small. The current SDU is 9188, but an SDU size of 18190 is needed to join into the ELAN tok16_1. In order to fix the problem, we need to increase the adapters PDU size. The PDU needs to be 18190 + 8 (AAL5 Header) = 18198.

We run the following command to change the adapters PDU:

Bring down child devices:

```
ifconfig tr2 down
ifconfig tr2 detach
rmdev -l tr2
rmdev -l tok2
```

Change the PDU size:

```
chdev -l atm0 -a pdu=18198
```

Bring up the child devices:

```
mkdev -l tok2
chdev -l tr2 -a state=up
```

After changing the PDU size, the LEC is able to initialize into the ELAN successfully.

8.5.3.4 LES join failure

The LEC tok2 has just been configured with a TCP/IP interface tr2. When we try to ping other systems in the same subnet, we get 100 percent packet loss. We run the following command to check the status of the LEC:

```
tokstat -d tok2 | grep "Driver Flags"
```

The command results in the following output:

```
Driver Flags: Up Broadcast Limbo
```

The tokstat output shows that the LEC is currently in a Limbo state. The LEC is not operational so we want to see how far it got in the Initialzation process. We run the following command to get the LANE specific statistics:

```
tokstat -d tok2 | grep -p "Specific Statistics"
```

Below is the resulting output of the above command:

```
ATM LAN Emulation Specific Statistics:
--------------------------------------
Emulated LAN Name: tok4_1
Local ATM Device Name: atm0
Local LAN MAC Address:
00.04.ac.ad.1e.f5
Local ATM Address:
```

```
47.00.05.80.ff.e1.00.00.00.f2.15.16.45.00.04.ac.ad.1e.f5.04
Auto Config With LECS:
Yes
LECS ATM Address:
47.00.05.80.ff.e1.00.00.00.f2.15.16.45.00.06.29.b9.ae.8d.00
LES ATM Address:
47.00.05.80.ff.e1.00.00.00.f2.15.16.45.00.06.29.b9.ae.8d.30
General Errors: 114                 Address Deregistrations: 0
Control Timeout (sec): 120          LE_ARP Rsp Timeout (sec): 1
Max Unknown Frame Count: 1          Flush Timeout (sec): 4
Max Unknown Frame Time (sec): 1     Path Switch Delay (sec): 6
VCC Activity Timeout (sec): 1200    VCC Avg Rate (Kbps): 155000
Maximum LE_ARP Retries: 1           VCC Peak Rate (Kbps): 155000
LE_ARP Aging Timeout (sec): 300     Connect Complete Time (sec): 4
LE_ARP Forward Timeout (sec): 15    Maximum Frame Size: 4544
```

We have a valid local ATM address; so, switch communication is working. Auto Config is set to “Yes”; so, we will be getting our information from the LECS. We have a valid LES ATM address so this tells us that the LECS configure phase completed, and we had a failure during the join phase or the BUS connect phase. The following commands are run to gather an iptrace of the LEC Initialziation process:

Bring down child devices and LEC:

```
ifconfig tr2 down
ifconfig tr2 detach
rmdev -l tr2
rmdev -l tok2
```

Start the iptrace:

```
rm /tmp/iptrc.bin
startsrc -s iptrace -a “-i at0 /tmp/iptrc.bin”
```

Bring up the LEC and child devices:

```
mkdev -l tok2
chdev -l tr2 -a state=up
sleep 20
```

Stop the iptrace:

```
stopsrc -s iptrace
```

Generate a readable report:

```
ipreport -srn /tmp/iptrc.bin > /tmp/iptrc.rpt
```

The following is the first few packets of the ipreport output that was generated (iptrc.rpt). Comments have been added to explain what is being seen in the ipreport.

Below is an ILMI query to the service registry. If the LECS has its address in the service registry, the LEC will get the address back in the response. If the address is not registered, the LEC will get a response with a value of NULL. This query only occurs in AIX 4.3.2 and higher:

```
Packet Number 1
ATM: ====( 45 bytes transmitted on interface at0 )== 17:41:41.892010972
ATM: VC = 0.16
ILMI: Get Next Request PDU
ILMI: Request ID   = 65286
ILMI: Error Status = 0(Success)
ILMI: Object ID    =
1.3.6.1.4.1.353.2.8.1.1.3.0.0(atmfSrvcRegATMAddress)
ILMI:     Value    = <Null>
```

Below is the response, and we have a value filled in; so, we will use this address to connect to the LECS. If the address was NULL, we would try and connect to the LECS using the well-known LECS ATM address:

```
Packet Number 2
ATM: ====( 77 bytes received on interface at0 )== 17:41:41.895581240
ATM: VC = 0.16
ILMI: Get Response PDU
ILMI: Request ID   = 65286
ILMI: Error Status = 0(Success)
ILMI: Object ID    = 1.3.6.1.4.1.353.2.8.1.1.3.0.10.1.3.
ILMI:                6.1.4.1.353.1.5.1.1
ILMI:                (atmfSrvcRegATMAddress)
ILMI:     Value    = 47 0 5 80 ff e1 0 0 0 f2 15 16 45 0 6 29 b9 ae 8d 0
```

There can be more than one LECS that the LE client can use. The LE client will keep querying until it gets all the addresses. The next query below returned NULL; so, there was only one LECS address register:

```
Packet Number 3
ATM: ====( 57 bytes transmitted on interface at0 )== 17:41:41.895755791
ATM: VC = 0.16
ILMI: Get Next Request PDU
ILMI: Request ID   = 65286
ILMI: Error Status = 0(Success)
ILMI: Object ID    = 1.3.6.1.4.1.353.2.8.1.1.3.0.10.1.3.
ILMI:                6.1.4.1.353.1.5.1.1
ILMI:                (atmfSrvcRegATMAddress)
ILMI:     Value    = <Null>
```

Below is the response of the ILMI query, and it returned a value of NULL. This means there are no more LECS ATM addresses in the service registry:

```
Packet Number 4
ATM: ====( 57 bytes received on interface at0 )== 17:41:41.900198645
ATM: VC = 0.16
ILMI: Get Response PDU
ILMI: Request ID   = 65286
ILMI: Error Status = 2(No such Name)
ILMI: Object ID    = 1.3.6.1.4.1.353.2.8.1.1.3.0.10.1.3.
ILMI:                6.1.4.1.353.1.5.1.1
ILMI:                (atmfSrvcRegATMAddress)
ILMI:     Value    = <Null>
```

Below is the start of the point-to-point SVC setup to the LECS. The setup is done to the ATM address returned in the ILMI Get response. Also, we want to make note that the call reference value for this connection is 516:

```
Packet Number 5
ATM: ====( 116 bytes transmitted on interface at0 )== 17:41:41.910717864
ATM: VC = 0.5
Q2931: 05(Setup)
Q2931: Call Ref Value  = 516
Q2931: Message Len     = 103
Q2931:
Q2931: IE Element      = 58(AAL Parameters)
Q2931:   AAL Type      = AAL type 5
Q2931:   Fd Max SDU sz = 1516
Q2931:   Bk Max SDU sz = 1516
Q2931:   SSCS Type     = Null
Q2931: IE Element      = 59(ATM Traffic Descriptor)
Q2931:   Fwd Peak Cell Rate (CLP : 0+1)
Q2931:                 = 413334 cells/sec (175253616 bps)
Q2931:   Bkwd Peak Cell Rate(CLP : 0+1)
Q2931:                 = 413333 cells/sec (175253192 bps)
Q2931:   Best Eff Ind  = 190
Q2931: IE Element      = 5e(Broadband Bearer Capability)
Q2931:   Bearer Class  = BCOB-X
Q2931:   Traffic Type  = No Indication
Q2931:   Susc. to clip = Susceptible to Clipping
Q2931:   Conn Type     = Point-to-Point
Q2931: IE Element      = 5f(Broadband Low Layer Information)
Q2931: IE Element      = 70(Called Party Number)
Q2931:   Type of Num   = 1 (International)
Q2931:   Addressing    = 2 (ATM Endsystem Address)
Q2931:   Address       =
Q2931:      47 0 5 80 ff e1 0 0 0 f2 15 16 45 0 6 29 b9 ae 8d 0
Q2931: IE Element      = 6c(Calling Party Number)
```

```
Q2931:   Type of Num   = 1 (International)
Q2931:   Addressing    = 2 (ATM Endsystem Address)
Q2931:   Address       =
Q2931:      47 0 5 80 ff e1 0 0 0 f2 15 16 45 0 4 ac ad 1e f5 4
Q2931: IE Element      = 5c(QoS Parameter)
Q2931:   Coding Std    = 00(ITU-T Standardized)
Q2931:   QoS Class     = QoS Class 0(Unspecified)
Q2931:   QoS Class     = QoS Class 0(Unspecified)
```

SVC setup continues. We know this “Call Proceeding” message belongs to our SVC setup to the LECS because it has the same call reference value as the setup call. From the below information we now know that the SVC being set up will have a VCC of 0:530 (VPI:VCI):

```
Packet Number 6
ATM: ====( 24 bytes received on interface at0 )== 17:41:41.915803357
ATM: VC = 0.5
Q2931: 02(Call Proceeding)
Q2931: Call Ref Value  = 516
Q2931: Message Len     = 9
Q2931:
Q2931: IE Element      = 5a(Connection ID)
Q2931:   VPI           = 0
Q2931:   VCI           = 530
```

The SVC setup continues. Again, the call reference value is 516; so, we know this “Connect” message is tied with the LECS SVC setup:

```
Packet Number 7
ATM: ====( 24 bytes received on interface at0 )== 17:41:41.927831390
ATM: VC = 0.5
Q2931: 07(Connect)
Q2931: Call Ref Value  = 516
Q2931: Message Len     = 9
Q2931:
Q2931: IE Element      = 5a(Connection ID)
Q2931:   VPI           = 0
Q2931:   VCI           = 530
```

The LEC has sent a “Connect Acknowledge” back; so, at this point, the SVC setup to the LECS is complete. We now have an SVC connection to the LECS with a VCC of 0:530 and a call reference value of 516:

```
Packet Number 8
ATM: ====( 16 bytes transmitted on interface at0 )== 17:41:41.927990929
ATM: VC = 0.5
Q2931: 0F(Connect Acknowledge)
Q2931: Call Ref Value  = 516
```

```
Q2931: Message Len     = 0
```

Now that we have an SVC connection to the LECS, we expect the LE client to send an LE_CONFIGURE request to the LECS. Continuing through the trace, we see the LE_CONFIGURE request sent to the LECS. We make note that the ELAN name is tok4_1. This means that the LEC is requesting configuration for ELAN tok4_1. Depending the LECS policies, it should try and configure LEC on this specific ELAN. You will also notice that the below message was sent on VCC 0:530, which is the VCC that was set up to the LECS:

```
Packet Number 9
ATM: ====( 108 bytes transmitted on interface at0 )== 17:41:41.938224997
ATM: VC = 0.530
LANE CTRL CONFIGURE_REQUEST
LANE: Status        = 0
LANE: Trans ID      = 0
LANE: Request ID    = 0
LANE: Src LAN Dest:
LANE:   Tag         = 1
LANE:   MAC Addr    = 0 4 ac ad 1e f5
LANE: Tar LAN Dest:
LANE:   Tag         = 0
LANE:   MAC Addr    = 0 0 0 0 0 0
LANE: Src ATM ADDR  = 47 0 5 80 ff e1 0 0 0 f2 15 16 45 0 4 ac ad 1e f5 4
LANE: Tar ATM ADDR  = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
LANE: ELAN Name SZ  = 6
LANE: ELAN Name     = tok4_1
```

Our configure request has been sent; so, we should expect to see a configure response back from the LECS. We received the below configure response from the LECS. Looking at the Status field, we see that it is zero, which is correct. If a non-zero status code was returned, this means that an error occurred. Also, we want to make note of “Tar ATM ADDR” in this packet. We see that the “Tar ATM ADDR” contains the ATM address of the LES. This is the address the LEC will use to connect to the LES:

```
Packet Number 10
ATM: ====( 108 bytes received on interface at0 )== 17:41:41.938575231
ATM: VC = 0.530
LANE CTRL CONFIGURE_RESPONSE
LANE: Status        = 0
LANE: Trans ID      = 0
LANE: Request ID    = 0
LANE: Src LAN Dest:
LANE:   Tag         = 1
LANE:   MAC Addr    = 0 4 ac ad 1e f5
```

```
LANE: Tar LAN Dest:
LANE:   Tag         = 0
LANE:   MAC Addr    = 0 0 0 0 0 0
LANE: Src ATM ADDR  = 47 0 5 80 ff e1 0 0 0 f2 15 16 45 0 4 ac ad 1e f5 4
LANE: Tar ATM ADDR  = 47 0 5 80 ff e1 0 0 0 f2 15 16 45 0 6 29 b9 ae 8d 30
LANE: ELAN Name SZ  = 6
LANE: ELAN Name     = tok4_1
```

Point-to-point SVC setup to the LES using the ATM address returned in the LE_CONFIGURE response. We want to make note that the call reference for this connection is 515:

```
Packet Number 11
ATM: ====( 116 bytes transmitted on interface at0 )== 17:41:41.948955609
ATM: VC = 0.5
Q2931: 05(Setup)
Q2931: Call Ref Value  = 515
Q2931: Message Len     = 103
Q2931:
Q2931: IE Element      = 58(AAL Parameters)
Q2931:   AAL Type      = AAL type 5
Q2931:   Fd Max SDU sz = 1516
Q2931:   Bk Max SDU sz = 1516
Q2931:   SSCS Type     = Null
Q2931: IE Element      = 59(ATM Traffic Descriptor)
Q2931:   Fwd Peak Cell Rate (CLP : 0+1)
Q2931:                 = 413334 cells/sec (175253616 bps)
Q2931:   Bkwd Peak Cell Rate(CLP : 0+1)
Q2931:                 = 413333 cells/sec (175253192 bps)
Q2931:   Best Eff Ind  = 190
Q2931: IE Element      = 5e(Broadband Bearer Capability)
Q2931:   Bearer Class  = BCOB-X
Q2931:   Traffic Type  = No Indication
Q2931:   Susc. to clip = Susceptible to Clipping
Q2931:   Conn Type     = Point-to-Point
Q2931: IE Element      = 5f(Broadband Low Layer Information)
Q2931: IE Element      = 70(Called Party Number)
Q2931:   Type of Num   = 1 (International)
Q2931:   Addressing    = 2 (ATM Endsystem Address)
Q2931:   Address       =
Q2931:      47 0 5 80 ff e1 0 0 0 f2 15 16 45 0 6 29 b9 ae 8d 30
Q2931: IE Element      = 6c(Calling Party Number)
Q2931:   Type of Num   = 1 (International)
Q2931:   Addressing    = 2 (ATM Endsystem Address)
Q2931:   Address       =
Q2931:      47 0 5 80 ff e1 0 0 0 f2 15 16 45 0 4 ac ad 1e f5 4
Q2931: IE Element      = 5c(QoS Parameter)
```

```
Q2931:   Coding Std    = 00(ITU-T Standardized)
Q2931:   QoS Class     = QoS Class 0(Unspecified)
Q2931:   QoS Class     = QoS Class 0(Unspecified)
```

The SVC setup to the LES continues. We make note that the VCC will be 0:531:

```
Packet Number 12
ATM: ====( 24 bytes received on interface at0 )== 17:41:41.953852214
ATM: VC = 0.5
Q2931: 02(Call Proceeding)
Q2931: Call Ref Value  = 515
Q2931: Message Len     = 9
Q2931:
Q2931: IE Element      = 5a(Connection ID)
Q2931:   VPI           = 0
Q2931:   VCI           = 531
```

The SVC setup to the LES continues:

```
Packet Number 13
ATM: ====( 24 bytes received on interface at0 )== 17:41:41.967862425
ATM: VC = 0.5
Q2931: 07(Connect)
Q2931: Call Ref Value  = 515
Q2931: Message Len     = 9
Q2931:
Q2931: IE Element      = 5a(Connection ID)
Q2931:   VPI           = 0
Q2931:   VCI           = 531
```

The LEC has sent a “Connect Acknowledge” back; so, at this point, the SVC setup to the LES is complete. We now have a SVC connection to the LES with a VCC of 0:531 and a call reference value of 515:

```
Packet Number 14
ATM: ====( 16 bytes transmitted on interface at0 )== 17:41:41.968013844
ATM: VC = 0.5
Q2931: 0F(Connect Acknowledge)
Q2931: Call Ref Value  = 515
Q2931: Message Len     = 0
```

Now that we have an SVC connection to the LES, we expect our LE client to send a join request to the LES for ELAN tok4_1. The LEC will send its LAN MAC address and its ATM address in this request. You will also notice that the below join request is sent on VCC 0:531. This is the VCC that was # setup to the LES:

```
Packet Number 15
```

```
ATM: ====( 108 bytes transmitted on interface at0 )== 17:41:41.978230491
ATM: VC = 0.531
LANE CTRL JOIN_REQUEST
LANE: Status        = 0
LANE: Trans ID      = 1
LANE: Request ID    = 0
LANE: Src LAN Dest:
LANE:   Tag         = 1
LANE:   MAC Addr    = 0 4 ac ad 1e f5
LANE: Tar LAN Dest:
LANE:   Tag         = 0
LANE:   MAC Addr    = 0 0 0 0 0 0
LANE: Src ATM ADDR  = 47 0 5 80 ff e1 0 0 0 f2 15 16 45 0 4 ac ad 1e f5 4
LANE: Tar ATM ADDR  = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
LANE: ELAN Name SZ  = 6
LANE: ELAN Name     = tok4_1
```

We see a release with a call reference value of 516. Looking back at our notes, we see that is our SVC connection to the LECS. This is normal. Once the LEC has its configuration information, it no longer needs to talk the LECS; so, the connection is released:

```
Packet Number 16
ATM: ====( 20 bytes transmitted on interface at0 )== 17:41:41.978527376
ATM: VC = 0.5
Q2931: 4D(Release)
Q2931: Call Ref Value  = 516
Q2931: Message Len     = 6
Q2931:
Q2931: IE Element      = 8(Cause)
Q2931: Cause Value     = 31(Normal, Unspecified)
```

Now that the LES has set up its uni-directional SVC to our LEC, we expect the LES to send the LEC a join response to the LEC's join request. Looking further in the trace, we see the join response. We make note that the Status field is 4. A zero value indicates a successful join. Looking up a status code of 4 in the ATM forum LANE specification, we find this error indicates a duplicate LAN MAC address:

```
Packet Number 17
ATM: ====( 108 bytes received on interface at0 )== 17:41:41.978730168
ATM: VC = 0.531
LANE CTRL JOIN_RESPONSE
LANE: Status        = 4
LANE: Trans ID      = 1
LANE: Request ID    = 0
LANE: Src LAN Dest:
```

```
LANE:   Tag         = 1
LANE:   MAC Addr    = 0 4 ac ad 1e f5
LANE: Tar LAN Dest:
LANE:   Tag         = 0
LANE:   MAC Addr    = 0 0 0 0 0 0
LANE: Src ATM ADDR  = 47 0 5 80 ff e1 0 0 0 f2 15 16 45 0 4 ac ad 1e f5 4
LANE: Tar ATM ADDR  = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
LANE: ELAN Name SZ  = 6
LANE: ELAN Name     = tok4_1
```

We receive the expected “Release Complete” for the release message we sent for the LECS connection. We know this by matching up the call reference numbers again. The SVC to the LECS is now gone:

```
Packet Number 18
ATM: ====( 20 bytes received on interface at0 )== 17:41:41.989112474
ATM: VC = 0.5
Q2931: 5A(Release Complete)
Q2931: Call Ref Value  = 516
Q2931: Message Len     = 6
Q2931:
Q2931: IE Element      = 8(Cause)
Q2931: Cause Value     = 31(Normal, Unspecified)
```

Looking at the call reference value, we know that this is a release of the SVC from the LEC to LES:

```
Packet Number 19
ATM: ====( 20 bytes transmitted on interface at0 )== 17:41:42.902269280
ATM: VC = 0.5
Q2931: 4D(Release)
Q2931: Call Ref Value  = 515
Q2931: Message Len     = 6
Q2931:
Q2931: IE Element      = 8(Cause)
Q2931: Cause Value     = 31(Normal, Unspecified)
```

The release of the SVC from the LEC to LES is now complete:

```
Packet Number 20
ATM: ====( 20 bytes received on interface at0 )== 17:41:42.906491777
ATM: VC = 0.5
Q2931: 5A(Release Complete)
Q2931: Call Ref Value  = 515
Q2931: Message Len     = 6
Q2931:
Q2931: IE Element      = 8(Cause)
Q2931: Cause Value     = 31(Normal, Unspecified)
```

From the above iptrace, we see that the LEC received a join response with a status code of 4, and it released its connections in the ELAN. A status code of 4 indicates that the LEC tried to join into the ELAN with a LAN MAC address that another LEC had already registered. Changing the LAN MAC address to a unique value in the ELAN resolved the issue, and the LEC was operational.

Chapter 9. HACMP scenarios

Typical customer environments, where HACMP and ATM would be used together, have the following characteristics:

- The server must be highly available.
- The server is accessed across an ATM network via TCP/IP.

Clients requiring access to the server can be users or applications running on other systems. If data is associated with the applications on the server, the disk holding the data will be shared between the systems in the HACMP cluster. Unlike a typical HACMP customer situation, we did not make any applications on these servers highly available, nor did we make any data highly available; however, this is not necessary to demonstrate HACMP working with ATM. For this purpose, making an ATM IP address highly available, and moving the IP address to other adapters and hosts upon adapter or host failure, is necessary.

If you are new to either ATM or HACMP, the following overviews describe key concepts important to our test environment. If you are new to both areas, read both.

9.1 HACMP for the ATM expert

HACMP provides availability of an application and its data by using redundant hardware. The simplest implementation, hot standby, provides a machine that will take over the IP address, the application data, if any, and restart the application after the server fails as Figure 32 on page 224 depicts.

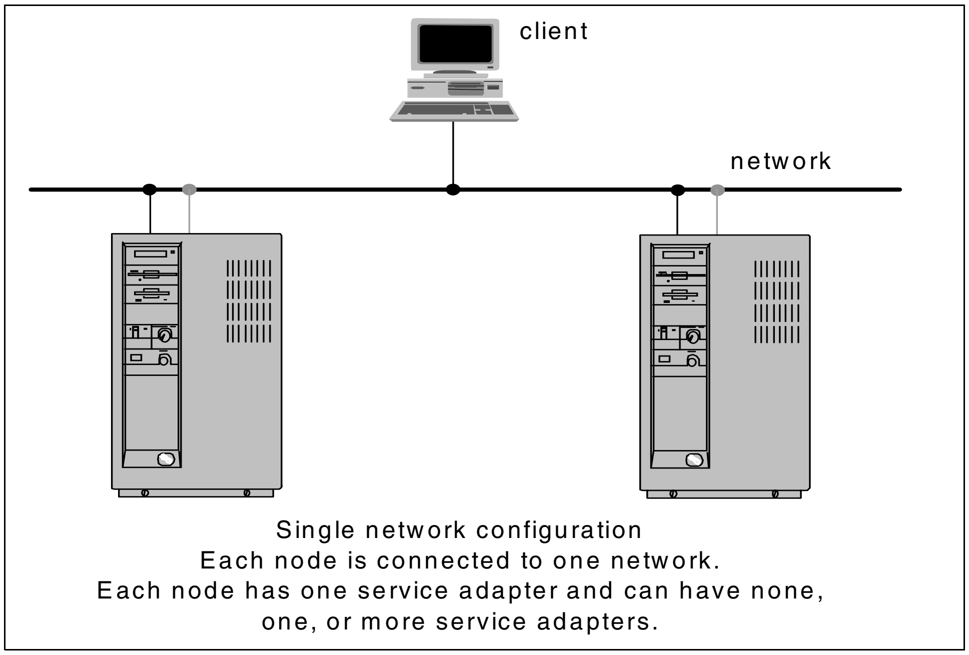

client
network
Single network configuration
Each node is connected to one network.
Each node has one service adapter and can have none,
one, or more service adapters.

Figure 32. Minimum HACMP configuration

Each machine sends heartbeats to, and listens for heartbeats from, the other system. Normally, these heartbeats are sent over at least three paths:

- A non-TCP/IP path (typically an RS-232 connection)
- A TCP/IP path over the network
- A TCP/IP path over a different IP subnet on the same physical network

These heartbeat paths provide the information necessary to determine if a system, network adapter, or network has failed. In fact, this is all HACMP detects by default. HACMP does not, by default, take any action for a network failure. For a network adapter failure, it will move the highly-available IP address (the address by which clients access the application) to another network adapter on the host. For a server failure, HACMP will move the IP address and any shared disk to the hot standby system and then restart the application. So, from a client's perspective, for example, a telnet session to the server, the client will lose access to the server, but after a short time, will be able to reconnect to the same IP address and access the same data even though, in reality, they may be logging onto a different server.

HACMP for AIX supports both the Classical IP and LAN Emulation forms of the ATM protocol. Beginning with version 4.3.1, HACMP supports hardware address takeover on ATM adapters attached to the same switch. This support is already available for Token Ring, Ethernet, and FDDI adapters, and alleviates the communications problems that could occur with adapter swap and IP address takeover for ATM adapters. Hardware address takeover is supported for Classical IP and ATM LAN Emulation.

HACMP can move both an IP address (also called IP Address Takeover or IPAT for short) and the hardware address, also known as the MAC address (Media Access Control address) from one adapter to another.

With traditional LANs, the physical network address is the MAC address. For ATM, the physical network address is the 20-byte ATM address that includes a 6-byte MAC address.

The ability to move the hardware address from one adapter to another is an important feature. Each host has an ARP table listing IP addresses and corresponding hardware addresses for each adapter recently contacted on the LAN. When a host sends an IP message to another system on the LAN, it converts the IP address to a hardware address using its ARP table and then sends it to the hardware address. Thus, when HACMP moves an IP address from one adapter to another, moving the hardware address ensures that other systems on the LAN have correct ARP tables.

If MAC address takeover is not used, we must use a mechanism to ensure that other systems on the LAN have ARP table entries updated when an IPAT occurs. For networks incapable of IPAT, there are two options for handling the situation:

- Pinging all the clients after an IPAT to update their ATM ARP cache
- Making the client's cluster aware

The first option is configured by editing the /usr/sbin/cluster/etc/clinfo.rc file on the HACMP servers and adding the clients to the PING_CLIENT_LIST variable. The second option requires the following steps on the client:

1. Install the HACMP client code
2. Edit the */usr/sbin/cluster/etc/clhosts* file to list the adapters on the HACMP servers
3. Start the clinfo daemon

Upon a change in the cluster, the clinfo daemon on the client updates its ARP cache. This daemon can also be configured to run other scripts when a change to the cluster occurs.

9.2 ATM for the HACMP expert

When compared to traditional LANs, such as Ethernet and Token Ring, ATM has some unique characteristics that can have a direct effect upon an HACMP environment. The current implementation of HACMP cannot exploit all of the special features of ATM.

AIX supports the most popular data protocols in ATM networks today: Classical IP, which is TCP/IP communications as defined in RFCs 1577 and 1755 Classical IP and ARP over ATM and ATM Forum LAN Emulation V1.0 client.These two protocols represent opposite approaches to networking with ATM. Whereas, Classical IP systems run TCP/IP directly atop ATM via an IP interface, LAN Emulation hides ATM from the upper-layer protocol, thereby providing ATM connectivity for applications written to operate over an Ethernet or Token Ring LAN. Two application programming interfaces (APIs) are available for ATM. The TCP/IP sockets programming interface is available for Classical IP environments, and a sockets API, which is included in AIX beginning with version 4.2, provides a direct programming interface to the ATM device driver.

Systems running Classical IP cannot communicate directly with systems running ATM LAN Emulation; these are different, incompatible protocols. TCP/IP traffic can, however, be passed between these environments via an IP router system that has an interface of each type.

ATM support for Classical IP is included in the base operating system (BOS) beginning with AIX V 4.1.4. Support for ATM LAN Emulation Client is available for AIX V 4.1.5 systems via I-listed (limited availability) RPQ P91164 but is included in AIX beginning with version 4.2.1. For ATM LAN Emulation environments, RS/6000 users must make sure that the LAN Emulation Service functions are provided within the network.

9.2.1 ATM switch-based design

ATM networks are built by interconnecting ATM switch devices, such as IBMs 8260 or 8285 ATM switches, in a mesh-like fashion. User end stations, such as the RS/6000, are attached to the network at their local switch. The local switch passes traffic out to the network for delivery to end stations attached elsewhere in the network. Because the exact design and placement of the switches varies from network-to-network, ATM networks are typically depicted as clouds.

The local switch administrator sets up ports for end stations, such as the RS/6000. Since most switch and end station configuration values must match, the administrator is key in determining the values that should be entered in the RS/6000 SMIT menus.

9.2.2 ATM connection types

Each ATM adapter provides one physical connection to the network which, in turn, supports multiple, concurrent logical connections (virtual channel connections or VCCs) to the other end user stations on the network. The RS/6000 supports up to 1023 of these virtual channels (essentially, simultaneous login sessions) per adapter. The ATM Forum defines two basic types of ATM virtual channel connections, Permanent Virtual Circuits (PVCs) and Switched Virtual Circuits (SVCs). PVCs are semi-permanent that are set up using administrative procedures at the switches, while SVCs are logical connections set up anew using signalling procedures when a new call is placed. PVCs are analogous to leased line connections, where SVCs are similar to the circuit switching done for a traditional telephone call.

Once the VCC has been established across the network, a point-to-point style connection is up, and communication is bidirectional, much like the traditional telephone call. ATM Classical IP can run on PVCs or SVCs. ATM LAN Emulation requires SVCs. While it is possible to create a test setup without a switch (by cabling directly between two RS/6000 ATM adapters), it is limited to two systems configured to run TCP/IP on a PVC. Because PVCs are not flexible enough to "move" with a failure, IBM's HACMP software is designed for use with SVCs.

In a PVC environment, the local end station need only know the unique VPI:VCI (virtual path and virtual channel identifier) pair that has been preconfigured to lead from the local switch to the remote end station. With SVCs, on the other hand, the remote end station is located based upon its unique, 20-byte ATM address. SVC calls are routed through the ATM switches within the network much like telephone calls are set up based upon the digits in the telephone number.

9.2.3 SVC client/server design

Both ATM Classical IP and LAN Emulation have been designed to operate in a client/server fashion. In the case of Classical IP, RS/6000s may be configured as Classical IP clients or Classical IP ARP servers, with one ARP server for each IP subnet. The end station configured as a Classical IP ARP server maintains a master ARP table that maps IP addresses to ATM addresses for all end stations currently running on that logical IP subnet (LIS). When client stations need to contact others on the subnet, the client sends the IP address of the remote client to the Classical IP ARP server who, in turn, responds with the ATM address for that remote system. This information is cached on the local system so that communication between the two clients can be direct. The ARP server revalidates its ARP cache information every 15 minutes. Clients verify their cached information every 20 minutes; so, inactive connections are removed from the ARP caches within 20 minutes.

ATM ARP serving applies only to SVC operations. In a PVC connection, the path connecting the two systems is preconfigured at the intervening switches. Thus, the end stations need only know the VPI:VCI pair that represents the connection to the remote end station.

The current Classical IP RFCs specify that there can be one and only one ARP server per LIS. A newly drafted RFC (often called RFC 1577+) will define a backup server configuration.

9.2.4 Multi-interface capability

A LIS defines one IP subnet, but RS/6000s running ATM can, in theory, configure up to 256 Classical IP interfaces, thereby participating in up to 256 LISs. The interfaces, numbered at0 through at255, can be added in any sequence, and when multiple ATM adapters exist, the interfaces may be assigned to any adapter. In practice, however, most people implement just a few addresses. If you need more than eight Classical IP interfaces, you will need to increase the *ifsize* value which is set and viewed using the `no` command. The default value is 8, meaning that up to eight interfaces of any given network type are allowed on a system. This value would, for example, need to be increased before an ATM user could successfully configure the ninth Classical IP interface. Do not set this value abnormally large, as it causes additional resources to be reserved for all network types on your system, not just for ATM.

Generally speaking, all IP interfaces on an RS/6000, regardless of the network type, should be configured for a unique IP subnet.

9.2.5 ATM address assembly

ATM end stations, such as the RS/6000, are identified on the network by a 20-byte ATM network address. This ATM address is assembled each time an end station joins the network; for RS/6000s, this would normally be at system boot. Joining the network is a two-step registration process: First to the local switch, then to the Classical IP ARP server as follows:

1. The end station sends its 6-byte MAC address to the local switch. The switch pre-pends its 13-byte identification prefix and returns this number

as the ATM address for this end station. The last ATM address byte, termed the selector byte, is for end station application usage and is ignored by the switch.

The end station now knows its own ATM address.

2. Now the end station knows its own ATM and IP addresses, and it sends both to the ARP Server. (On RS/6000 end stations, when you configure the ATM interface, you enter the ARP server's ATM address in a SMIT menu; Section 9.4.2, “ATM setup” on page 234 shows how to do this.)

   The ARP server now lists this end station in its master ARP table.

Because the ATM address is assembled new each time the end station registers, client end stations in an SVC ATM environment can be moved and reattached to a different switch within the network without disrupting the network and without changing the client configuration. The only address that is hard-coded into end station configuration is that of the Classical IP ARP server. As the MAC address of this system's adapter is a part of its own ATM address, a change in ATM adapters or server could pose a potential problem.

Note

A wholesale configuration change at the client systems can be avoided by simply using the alternate address in SMIT to reinstate the MAC address of the original adapter.

9.2.6 Application considerations

TCP/IP broadcasts and ARP broadcasts are not sent across point-to-point (connection-oriented) connections such as Classical IP, X.25, PPP, and SLIP. Therefore, TCP/IP client/server applications in which the client discovers the server by broadcasting will not work. This means items, such as rwhod, NIS, NCS, timed, and so forth, will not work over Classical IP.

Because ATM is a connection-oriented method, it cannot do hardware address takeover in HACMP in the way in which this is done with traditional LANs. This is not a limitation of our HACMP or ATM implementation; it is a matter of ATM architecture. In Classical IP environments, one system cannot listen in on the network and assume another system's identity by using its MAC address, as is possible with connectionless methods, such as Ethernet and Token Ring. Therefore, do not view ATM as a drop-in replacement for connectionless methods.

9.3 Lab setup

We developed a HACMP test scenario that was used for:

- HACMP and ATM Classical IP and
- HACMP and ATM LAN Emulation

Both setup tests for ATM on the RS/6000 required the following components:

- ATM adapter and appropriate adapter cable
- Local ATM switch
- AIX operating system with ATM support

Figure 33 shows the test environment.

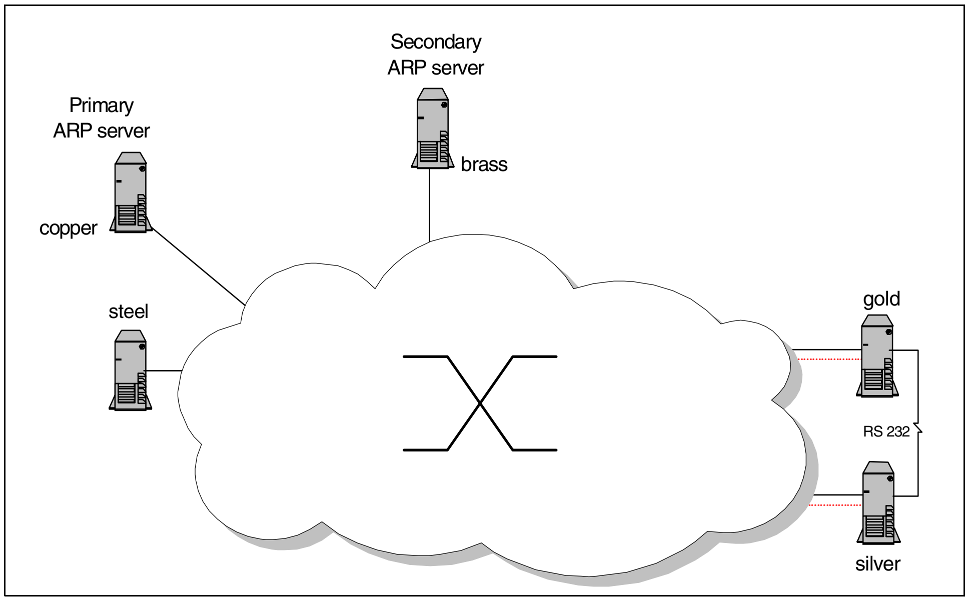


Figure 33. HACMP test environment

Also, two ARP servers are required for our tests (see 9.4.1, “HACMP configuration restrictions for Classical IP” on page 233 for details). We will only show the configuration for the primary ARP server. See Chapter 6, “Classical IP over ATM” on page 71 for the appropriate information about the configuration of a secondary ARP server.

9.3.1 Hardware requirements

Both network components and workstations were needed for the test environment.

9.3.1.1 Network components

The following components were used in the network.

- Two ATM switches (IBM 8260-A17 with DMM, PNNI, ATM, and MSS). They provided the ATM "cloud" and LAN Emulation Service functions.

ATM LAN Emulation is designed as a client/server operation. AIX provides an ATM LAN Emulation V1.0 client (only); so, IBM MSS (Multiprotocol Switched Services) provided the required ATM LAN Emulation Service functions.

9.3.1.2 Workstations

All RS/6000 systems in our configuration were models F50 with Turboways 155 PCI UTP ATM adapter (#2963). Each HACMP node had two of these ATM adapters.

Command output on one system showed the following key adapters and interfaces. Notice the ELANs appear as real adapters, but the slot information is missing from the location column:

```
# lsdev -Cc adapter
...
ent0   Available 04-C0 IBM PCI Ethernet Adapter (22100020)
atm0   Available 04-08 IBM PCI 155 Mbps ATM Adapter (14107c00)
...
atm1   Available 04-02 IBM PCI 155 Mbps ATM Adapter (14107c00)
ent1   Available       ATM LAN Emulation Client (Ethernet)
tok0   Available 04-07 IBM PCI Tokenring Adapter (14101800)
tok1   Available       ATM LAN Emulation Client (Token Ring)
#
```

9.3.2 Software requirements

HACMP V 4.3.1 and either AIX V 4.3.3 Base Operating system (which includes software support for ATM Classical IP and LAN Emulation) was used.

The following screens show the installed devices drivers for the ATM PCI adapter:

```
$ lslpp -l | grep ATM
bos.atm.atmle             4.3.3.0  COMMITTED  ATM LAN Emulation Client
  ...
  devices.pci.14107c00.com   4.3.3.0  COMMITTED  Common ATM Adapter Software
  devices.pci.14107c00.diag  4.3.3.0  COMMITTED  PCI ATM Adapter (14107c00)
  devices.pci.14107c00.rte   4.3.3.0  COMMITTED  PCI ATM Adapter (14107c00)
```

We used HACMP 4.3.1 for our tests. The following screen lists the installed software components:

```
# lslpp -l | grep cluster

  cluster.base.client.lib    4.3.1.0  COMMITTED  HACMP Base Client Libraries
  cluster.base.client.rte    4.3.1.0  COMMITTED  HACMP Base Client Runtime
  cluster.base.client.utils  4.3.1.0  COMMITTED  HACMP Base Client Utilities
  cluster.base.server.diag   4.3.1.0  COMMITTED  HACMP Base Server Diags
  cluster.base.server.events
  cluster.base.server.rte    4.3.1.0  COMMITTED  HACMP Base Server Runtime
  cluster.base.server.utils  4.3.1.0  COMMITTED  HACMP Base Server Utilities
  cluster.cspoc.cmds         4.3.1.0  COMMITTED  HACMP CSPOC Commands
  cluster.cspoc.dsh          4.3.1.0  COMMITTED  HACMP CSPOC dsh and perl
  cluster.cspoc.rte          4.3.1.0  COMMITTED  HACMP CSPOC Runtime Commands
  cluster.msg.en_US.client   4.3.1.0  COMMITTED  HACMP Client Messages - U.S.
  cluster.msg.en_US.cspoc    4.3.1.0  COMMITTED  HACMP CSPOC Messages - U.S.
  cluster.msg.en_US.server   4.3.1.0  COMMITTED  HACMP Server Messages - U.S.
  cluster.vsm.server         4.3.1.0  COMMITTED  HACMP Visual System Management
  cluster.base.client.rte    4.3.1.0  COMMITTED  HACMP Base Client Runtime
  cluster.base.server.rte    4.3.1.0  COMMITTED  HACMP Base Server Runtime
  cluster.base.server.utils  4.3.1.0  COMMITTED  HACMP Base Server Utilities
#
```

9.3.3 Naming conventions

The following naming and IP addressing conventions, shown in Table 4, were used in the test environment.

Table 4. Naming conventions for HACMP tests

Network Type	Machine Name	Boot Address	Service Address	Standby Address
Classical IP	gold	192.168.20.1	192.168.20.2	192.168.21.1
	silver	192.168.20.3	192.168.20.4	192.168.21.3
LAN Emulation Token Ring	gold	192.168.20.101	192.168.20.102	192.168.21.101
	silver	192.168.20.103	192.168.20.104	192.168.21.103

Network Type	Machine Name	Boot Address	Service Address	Standby Address
LAN Emulation Ethernet	gold	192.168.20.201	192.168.20.202	192.168.21.201
	silver	192.168.20.203	192.168.20.204	192.168.21.203

9.4 HACMP and ATM Classical IP

Because ATM is a connection-oriented technology, and IP is datagram-oriented technology, mapping IP addresses over ATM Classical IP is complex. For PVC-type ATM connections, each IP station must be manually configured. SVC-type ATM networks are dynamic and are divided into logical IP subnets (LIS). An LIS is similar to a traditional LAN segment.

As said before, Classical IP implements the TCP/IP protocols directly on top of the ATM protocol. ATM Classical IP networks are significantly different in functionality and methodology than more traditional networks, such as Token Ring and Ethernet. TCP/IP does not operate the same on TCP/IP networks using Classical IP. One key difference is that ATM does not support a network broadcast.

9.4.1 HACMP configuration restrictions for Classical IP

ATM Classical IP networks must be defined to HACMP as private networks because ATM Classical IP does not support broadcasting.

There are two basic components in configuring ATM Classical IP networks:

- Configuring the ATM Classical IP ARP servers
- Configuring the ATM Classical IP ARP clients

ATM Classical IP networks require the use of an ARP server to handle resolutions from ATM hardware addresses to IP addresses. One ARP server exists for each defined subnetwork. Thus, two ARP servers are required for each ATM Classical IP network configured in an HACMP cluster; one for the service subnet and one for the standby subnet. The ATM ARP client must be configured to include the following information about the ATM ARP server:

- The MAC address of the ATM ARP server
- The ATM ARP server and ATM ARP client must be on the same subnet.

An ATM ARP server can be an HACMP for AIX client, but it cannot be an HACMP server because ATM ARP clients hard code the ATM link address of the ARP server (including the MAC address). Adapter swaps and IPAT, therefore, do not work on HACMP nodes that are also ATM ARP servers.

ATM can support multiple interfaces per ATM adapter. Adapters used in HACMP servers, however, should use only one interface per adapter. ATM switches can be used as ATM ARP servers only if the switch automatically updates its ARP cache after an HACMP adapter event.

Thus, ATM networks must be configured as switched virtual circuits through an ATM switch, with the ATM ARP server configured on a system that is not a member of the cluster.

9.4.2 ATM setup

This section describes the configuration for the Classical IP scenario. Systems gold and silver have two ATM adapters; so, they have two connections to the switch (see Figure 34 on page 235 for details). Notice that the IBM 8260 ATM switch is represented by a cloud and its 13-byte switch prefix is 39.09.85.11.11.11.11.11.11.11.11.01.01. Because an end station's 20-byte ATM address is comprised of the A*TM switch prefix + the ATM adapter MAC address + the Classical IP at interface number*, the addresses for system gold's two Classical IP interfaces are as follows:

(at0) 39.09.85.11.11.11.11.11.11.11.11.01.01.08.00.5a.99.98.8b.00

(at1) 39.09.85.11.11.11.11.11.11.11.11.01.01.08.00.5a.99.91.f9.01

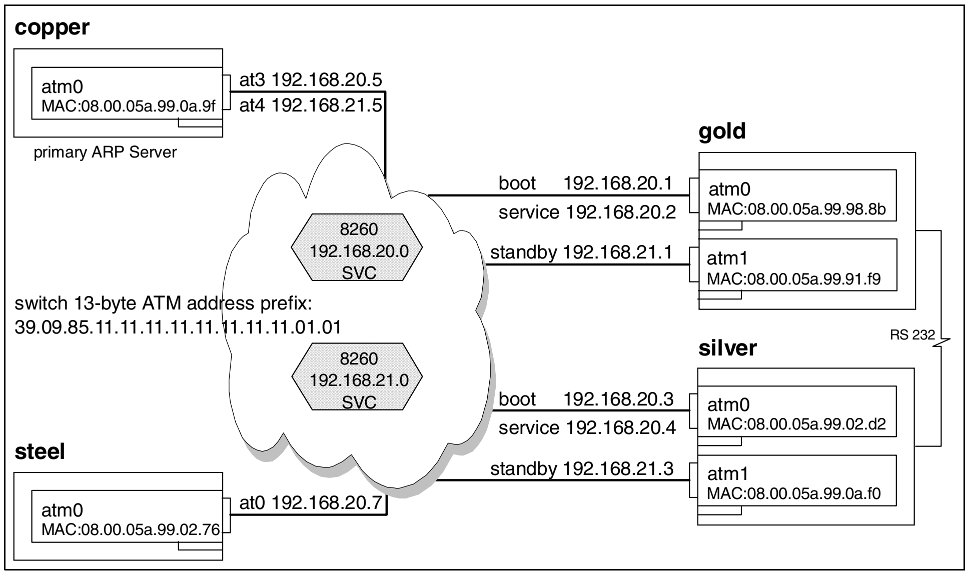


Figure 34. HACMP for ATM Classical IP

Notice that the last byte (termed the ATM address selector byte) matches the IP interface number. It is 00 for at0, 01 for at1, and so forth. For details on ATM addresses and address assembly, see “9.1, “HACMP for the ATM expert” on page 223.

For HACMP, these IP interfaces must be on different adapters to handle adapter failures. In non-HACMP environments, however, several Classical IP interfaces may be configured on one adapter and those interfaces need not be in numerical order. In those cases, the first 19 bytes of the 20-byte ATM address would be identical for all interfaces; only the last byte, the selector byte, would distinguish the interface. System *mickey* provides an example of this with the following interfaces configured on one adapter:

(at3) 39.09.85.11.11.11.11.11.11.11.11.01.01.08.00.5a.99.0a.9f.03

(at4) 39.09.85.11.11.11.11.11.11.11.11.01.01.08.00.5a.99.0a.9f.04

ATM SMIT configuration is, basically, a two step procedure. First, each adapter must be *available* and configured to match the switch port configuration. Second, TCP/IP interfaces are added for each adapter.

```
                  Change / Show Characteristics of an ATM Adapter

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[TOP]                                                    [Entry Fields]
  ATM Adapter                                          atm0
  Description                                          155 Mbps ATM Fiber Ada>
  Status                                               Available
  Location                                             00-03
  User Best Effort Peak Rate (kbits/sec)              [1500]                  +#
  Enable ALTERNATE ATM MAC address                     no                      +
  ALTERNATE ATM MAC address (12 hex digits)           [0x]
  ATM Adapter Maximum PDU size (bytes)                [9188]                  +#
  DMA bus memory width                                [0x1000000]             +X
  Maximum Small ATM mbufs                             [50]                    +#
  Maximum Medium ATM mbufs                            [100]                   +#
  Maximum Large ATM mbufs                             [300]                   +#
  Maximum Huge ATM mbufs                              [50]                    +#
  Max Transmit Block (MTB) size (kbytes)              [64]                    +#
  Minimum Small ATM mbufs                             [20]                    +#
  Minimum Medium ATM mbufs                            [30]                    +#
  Minimum Large ATM mbufs                             [70]                    +#
  Minimum Huge ATM mbufs                              [5]                     +#
  Minimum ATM MTB mbufs                               [4]                     +#
  Minimum ATM interface buffer size                   [2048]                  +#
  Minimum Guaranteed VCs Supported                    [8]                     +#
  Maximum Number of VCs Needed                        [32]                    +#
  SVC UNI Version                                      auto_detect             +
  Software transmit queue length                      [512]                   +#
  Sonet or SDH interface                              [0]                     +#
  Provide SONET clock                                 [0]                     +#
[BOTTOM]

F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

This is the SMIT information for system gold's atm0 adapter. Most users can accept the defaults just as we did, but key fields are:

Best Effort Peak Rate:

Used to slow the ATM transmission rate. This default value specifies the full 155 Mbps.

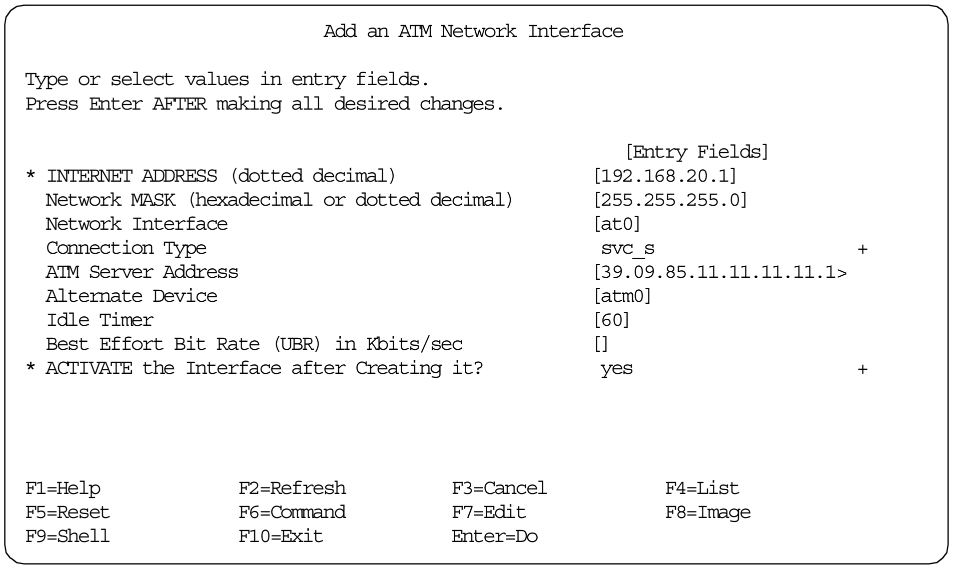
Enable alternate address:

Used to override the burned-in MAC address. User-defined values must be formatted 12 hexadecimal digits following the pre-filled "0x" shown in the menu. This should not be used on HACMP servers.

SVC UNI version

Options are auto_detect, UNI3.0, and UNI3.1. This is the ATM Forum defined "User-Network Interface" specification version and must match the switch setting.

The following screen shows the Classical IP interface configuration. (Fastpath: *smitty mkinetat*.)



```
                              Add an ATM Network Interface

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                        [Entry Fields]
* INTERNET ADDRESS (dotted decimal)                  [192.168.20.1]
  Network MASK (hexadecimal or dotted decimal)       [255.255.255.0]
  Network Interface                                  [at0]
  Connection Type                                     svc_s                  +
  ATM Server Address                                 [39.09.85.11.11.11.11.1>
  Alternate Device                                   [atm0]
  Idle Timer                                         [60]
  Best Effort Bit Rate (UBR) in Kbits/sec            []
* ACTIVATE the Interface after Creating it?           yes                    +




F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

This is the SMIT information for system gold's at0 interface.

Network interface and alternate device (meaning the adapter):

These fields are not pre-filled with values and, although default values are assumed in some instances, it is safer when the appropriate interface and adapter values are keyed in.

Connection type:

Options are svc_c for SVC ARP clients, svc_s for SVC ARP servers, and pvc for permanent virtual circuits. Client systems must list the ARP server's 20-byte ATM address. For ARP servers, set the connection type to svc_s and leave the address field blank. With PVC connections, there is no client-server relationship; so, ATM addresses are not needed in SMIT.

ATM server address:

While it is okay to omit a leading zero in a byte of the ATM address (for example, the 0a.9f.00 shown above may be typed as a.9f.0), any ATM address must be typed with dots between each byte. If the dots are omitted, the address will not be interpreted correctly.

The ATM ARP server's ATM address is a part of each client’s system configuration; so, replacing or changing the ATM adapter in the ARP server machine could alter its ATM address. If this occurs, use the Alternate Address field in the adapter SMIT menu to reinstate the original address on the ARP server.

Configuring an ATM network requires the configuration of its ARP server. The following screen shows the setup for our primary ARP server named *steel*.

```
                            Add an ATM Network Interface

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
* INTERNET ADDRESS (dotted decimal)                 [192.168.20.5]
  Network MASK (hexadecimal or dotted decimal)      [255.255.255.0]
  Network Interface                                 [at3]
  Connection Type                                    svc_s                +
  ATM Server Address                                [39.09.85.11.11.11.11.1>
  Alternate Device                                  [atm0]
  Idle Timer                                        [60]
  Best Effort Bit Rate (UBR) in Kbits/sec           []
* ACTIVATE the Interface after Creating it?          yes                  +



F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

9.4.3 HACMP setup

Once the ATM network, including ARP server, ATM switch, and ATM network adapters on the systems, is setup and working, the HACMP configuration is just like any other HACMP configuration with one exception; the ATM interfaces are defined as "private" networks because Classical IP does not support broadcast. This is documented in the HACMP manuals. So, we configured the cluster, two nodes (gold and silver), and eight network adapters on two HACMP "networks": *atm_net* consisting of the physical ATM network, and *rs232_net* as the RS-232 connection between the machines. The following output from the `cllsif` command shows the HACMP network configuration:

```
Adapter        Type    Network   Net Type Attribute Node    IP Address
------------------------------------------------------------------------
gold_boot      boot    atm_net   atm      private   gold    192.168.20.1
gold_service   service atm_net   atm      private   gold    192.168.20.2
gold_stdby     standby atm_net   atm      private   gold    192.168.21.1
gold_tty0      service rs232_net rs232    serial    gold    /dev/tty0

silver_boot    boot    atm_net   atm      private   silver  192.168.20.1
silver_service service atm_net   atm      private   silver  192.168.20.2
silver_stdby   standby atm_net   atm      private   silver  192.168.21.1
silver_tty0    service rs232_net rs232    serial    silver  /dev/tty0
```

We set up a mutual takeover environment with two cascading resource groups, gold_rg and silver_rg, whose only resources were the service IP addresses, gold_service and silver_service, respectively. The priority chain for each resource group was *gold, silver* for *gold_rg* and *silver, gold* for *silver_rg*. So, when gold and silver are booted, the interfaces will have their IP addresses set to the boot and standby IP addresses. Once HACMP is started, the IP addresses will change. The boot address will change to the service address. Figure 34 on page 235 shows the IP addresses for each interface. Only one IP address will be on the interface at any one time. When atm0 fails on gold, the service address will move to the at1 interface. If system gold crashes, gold's service IP address will be placed on silver's standby interface, at1. Also, if system silver fails, silver's service IP address will be placed on gold's standby interface, at1.

9.4.4 Tools/testing/recovery

We developed several shell scripts and used certain commands repeatedly for testing that we are including here for your convenience.

Commands

The `arp -t atm -a` command, which displays the arp cache for the ATM networks on a system, was perhaps the most helpful ATM command in our environment. This command, along with the knowledge of the ATM end station registration process (described in 9.2, “ATM for the HACMP expert” on page 226), helped verify the status of the ATM ARP client connections and to pinpoint problem areas. All clients should have properly formatted ATM addresses, and the IP network address should match that of the entire LIS. You should not, for example, see a system defined for the 192.168.20.0 network registered with the 192.168.21.0 LIS ARP server. We saw this after a failover, but this was fixed when the clinfo daemon ran a script to ping all clients.

When a client system is first brought up, the `arp` command should show an entry for the local interface and for the ARP server system. The following shows the output from system *copper*:

```
copper # arp -t atm -a

SVC - at0 on device atm0 -
====================

at0(192.168.13.7)       39.9.85.11.11.11.11.11.11.11.11.1.1.8.0.5a.99.2.76.0
        IP Addr          VPI:KI Handle ATM Address
        steel_20(192.168.20.5)  0:73    3
39.9.85.11.11.11.11.11.11.11.11.1.1.8.0.5a.99.a.9f.3
```

Once HACMP starts on systems gold and silver, systems copper and silver have the following ARP cache entries:

```
copper # arp -t atm -a

SVC - at0 on device atm0 -
====================

at0(192.168.13.7)       39.9.85.11.11.11.11.11.11.11.11.1.1.8.0.5a.99.2.76.0
        IP Addr          VPI:KI Handle ATM Address
        gold_service(192.168.20.2)      0:85 4
39.9.85.11.11.11.11.11.11.11.11.1.1.8.0.5a.99.a.f0.0
        silver_service(192.168.21.4)    0:86 5
39.9.85.11.11.11.11.11.11.11.11.1.1.8.0.5a.99.83.a0.0
        steel_20(192.168.20.5)  0:73 3
39.9.85.11.11.11.11.11.11.11.11.1.1.8.0.5a.99.a.9f.3
```

```
gold # arp -t atm -a

SVC - at0 on device atm0 -
====================

at0(192.168.20.4)           39.9.85.11.11.11.11.11.11.11.11.1.1.8.0.5a.99.83.a0.0
        IP Addr          VPI:KI Handle ATM Address
        gold_service(192.168.20.2)      0:413 4
39.9.85.11.11.11.11.11.11.11.11.1.1.8.0.5a.99.a.f0.0
        steel_20(192.168.20.5)          0:409 3
39.9.85.11.11.11.11.11.11.11.11.1.1.8.0.5a.99.a.9f.3

SVC - at1 on device atm1 -
====================

at1(192.168.21.3)           39.9.85.11.11.11.11.11.11.11.11.1.1.8.0.5a.99.91.f9.1
        IP Addr          VPI:KI Handle ATM Address
        gold_stby(192.168.21.1)         0:110 4
39.9.85.11.11.11.11.11.11.11.11.1.1.8.0.5a.99.2.d2.1
        steel_21(192.168.21.5)          0:102 3
39.9.85.11.11.11.11.11.11.11.11.1.1.8.0.5a.99.a.9f.4
```

The ARP server’s arp cache shows its local interface and entries for all of the clients that have joined the LIS. The following entry is from system *steel*:

```
steel # arp -t atm -a

SVC - at3 on device atm0 -
====================

at3(192.168.20.5)           39.9.85.11.11.11.11.11.11.11.11.1.1.8.0.5a.99.a.9f.3
        IP Addr          VPI:KI Handle ATM Address
        gold_service(192.168.20.2)      0:100 8
39.9.85.11.11.11.11.11.11.11.11.1.1.8.0.5a.99.a.f0.0
        silver_service(192.168.20.4)    0:101 9
39.9.85.11.11.11.11.11.11.11.11.1.1.8.0.5a.99.83.a0.0
        copper(192.168.20.7)             0:86 7
39.9.85.11.11.11.11.11.11.11.11.1.1.8.0.5a.99.2.76.0

SVC - at4 on device atm0 -
====================

at4(192.168.21.5)           39.9.85.11.11.11.11.11.11.11.11.1.1.8.0.5a.99.a.9f.4
        IP Addr          VPI:KI Handle ATM Address
        gold_stby(192.168.21.1)         0:94 11
39.9.85.11.11.11.11.11.11.11.11.1.1.8.0.5a.99.2.d2.1
        silver_stby(192.168.21.3)       0:97 12
39.9.85.11.11.11.11.11.11.11.11.1.1.8.0.5a.99.91.f9.1
```

The HACMP ARP client systems will reflect the status of local interfaces, and the clinfo daemon will react to failures by pinging the ARP server, which causes the server to refresh the ARP cache information. The following is from system *gold*:

```
gold # arp -t atm -a

SVC - at0 on device atm0 -
===================

at0(192.168.20.2)           39.9.85.11.11.11.11.11.11.11.11.1.1.8.0.5a.99.a.f0.0
        IP Addr          VPI:KI Handle ATM Address
        silver_service(192.168.20.5)    0:284 4
39.9.85.11.11.11.11.11.11.11.11.1.1.8.0.5a.99.83.a0.0
        steel_20(192.168.20.3)          0:280 3
39.9.85.11.11.11.11.11.11.11.11.1.1.8.0.5a.99.a.9f.3
        copper(192.168.20.7)             0:282 5
39.9.85.11.11.11.11.11.11.11.11.1.1.8.0.5a.99.2.76.0

SVC - at1 on device atm0 -
===================

at1(192.168.21.1)           39.9.85.11.11.11.11.11.11.11.11.1.1.8.0.5a.99.2.d2.1
        IP Addr          VPI:KI Handle ATM Address
        silver_stby(192.168.21.3)       0:174 4
39.9.85.11.11.11.11.11.11.11.11.1.1.8.0.5a.99.91.f9.1
        steel_21(192.168.21.5)          0:124 3
39.9.85.11.11.11.11.11.11.11.11.1.1.8.0.5a.99.a.9f.4
```

The `netstat -i` and `lscfg -vl atm0 | grep` network address commands also provide the information necessary to determine which IP address was on each adapter by cross referencing the network MAC address.

Recovery

We tested network adapter and host failure and recovery. For the network adapter failure, we disconnected the cable from the adapter with the service IP address. For host failures, we performed a graceful takeover and a system crash test.

We found a graceful takeover took 43 seconds to move the IP address from one machine to the other. Moving the IP address from one adapter to another on the same machine took about 25 seconds in our environment. These times will vary depending upon the system load, the speed of the processor, and the HACMP failure detection rate.

Custom scripts

To provide a quick test of communications, we used the following script (this one specifically for system gold):

```
#!/bin/ksh
# pingtest - a script to test pinging to the adapters in the cluster
# This script should be modified for each machine on which it is run

for i in gold_boot gold_service gold_stby silver_boot silver_service \
    silver_stby steel_20 steel_21 copper
do
        ping -cl $i > /tmp/$$.out &
        sleep 1
        grep    " 0% packet loss" /tmp/$$.out > /dev/null
        if [ $? = 0 ] then
                print "$i \t is available"
        else
                print "$i \t is unavailable"
        fi
        rm /tMP/$$.out
done
```

```
#!/bin/ksh
# clearhalogs - a script to clear out the /tmp/hacmp.out, and errlog (for
# cluster messages only) and the /usr/adm/cluster/log files on all systems
# in the cluster
export PATH=$PATH:/usr/sbin/cluster/utilities:/usr/sbin/cluster/diag
# Get a good IP address for the HACMP servers: clgetaddr is a HACMP utility
goldIP=`clgetaddr gold`
silverIP=`clgetaddr silver`
print "Clearing logs for gold"
        rsh $goldIP "cat /dev/null > /tmp/hacmp.out"
        rsh $goldIP "cat /dev/null > /usr/adm/cluster.log"
        rsh $goldIP "errclear -j AA8AB241 0"
print "Clearing logs for silver"
        rsh $silverIP "cat /dev/null > /tmp/hacmp.out"
        rsh $silverIP "cat /dev/null > /usr/adm/cluster.log"
        rsh $silverIP "errclear -j AA8AB241 0"
```

```
#!/bin/ksh
# get.net.info - a script to gather info on interfaces and arp
# chache for gold, silver, steel, copper.
# This script requires an /.rhost file on the hosts and my not work
# if there are network problems

print "Checking gold"
goldIP=`clgetaddr gold`
rsh $goldIP /usr/sbin/ifconfig at0
rsh $goldIP /usr/sbin/ifconfig at1
rsh $goldIP /usr/sbin/cluster/pingtest
rsh $goldIP /usr/sbin/arp -t atm -a

print "Checking silver"
silverIP=`clgetaddr silver`
rsh $silverIP /usr/sbin/ifconfig at0
rsh $silverIP /usr/sbin/ifconfig at1
rsh $silverIP /usr/sbin/cluster/pingtest
rsh $silverIP /usr/sbin/arp -t atm -a

print "Checking steel"
rsh steel /usr/sbin/cluster/pingtest
rsh steel /usr/sbin/arp -t atm -a

print "Checking copper"
rsh copper /usr/sbin/cluster/pingtest
rsh copper /usr/sbin/arp -t atm -a
```

9.4.5 Summary

HACMP supports ATM for IPAT on a Classical IP network; thus, a server that is accessed via TCP/IP over ATM can be made highly available. The ATM ARP server remains as a single point of failure, though it appears to be possible to implement HACMP on the ARP server, provided the other systems on the network make appropriate changes to their ATM configuration upon a IPAT on the ARP server.

9.5 HACMP and ATM LAN Emulation

As mentioned before, HACMP now supports IP address takeover in ATM LAN Emulation environments, including emulated Token Ring and Ethernet local area networks (ELANs.) This support is provided in HACMP V4.2.2 for use with AIX levels 4.2.1 and 4.3 systems.

Figure 35 shows a basic hot standby configuration. System steel is a client system that is not part of the HACMP cluster. It is directly attached to the ATM network. Such a cluster would also have a non-IP heartbeat path and shared disk for highly-available data requirements.

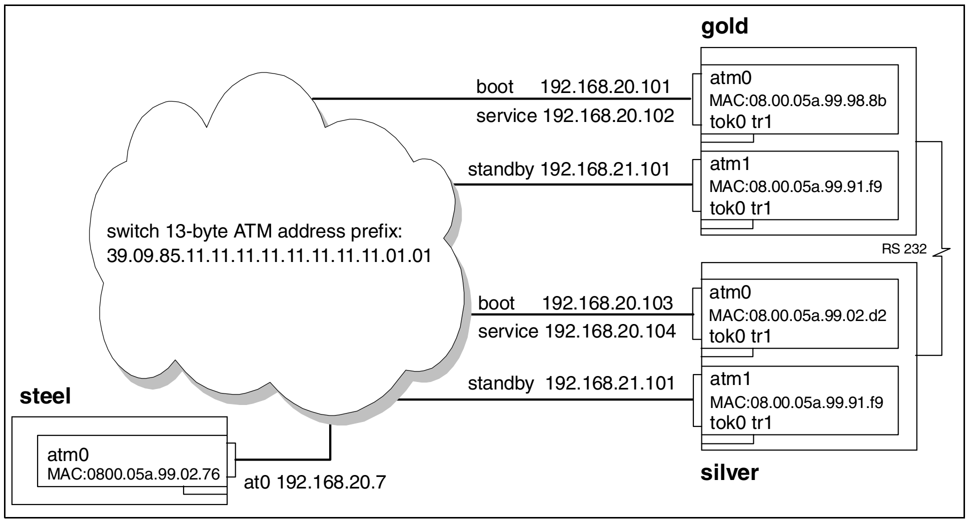


Figure 35. Basic ATM LAN Emulation

In the diagram, the HACMP service IP address, 192.168.20.102, is the only IP address accessed by clients on the network and is moved from adapter to adapter upon adapter or node failure. In a mutual takeover environment, the other machine would also have a service IP address.

The general steps to set this up are as follows:

1. Set up RS/6000 and network hardware
2. Install AIX and HACMP software
3. Configure ATM LAN Emulation networking (using boot/standby addresses)
4. Configure HACMP

While it is possible to have more than one emulated Token Ring or Ethernet LAN (ELAN) on one physical ATM adapter, HACMP only provides IP address takeover for one emulated LAN per adapter. HACMP would be unable to move the IP address from a failed service adapter to the standby adapter on the same failed physical ATM adapter.

9.5.1 Testing overview

Tests were run for both Ethernet and Token Ring emulated LANs and for both Micro Channel (MCA) and PCI bus systems. The following HACMP tests were successfully completed:

- Failure and recovery of a standby adapter.

- Failure of a service adapter and movement of the service IP address to a standby adapter.
- Failure of a service adapter and movement of the service IP address to a standby adapter. Then, restore the service adapter and fail the standby adapter and movement of the service IP address back to the service adapter.
- Start up the cluster with standby adapters disconnected.
- Restore all standby adapters after the cluster is started with them initially disconnected.
- Start up the cluster with all service adapters disconnected and establishment of the service IP addresses on the standby adapters.
- Failure and recovery of both service and standby adapters on a single node.
- Failure and recovery of the ATM network on all nodes.
- Node failures with cascading resource groups and NFS mounted file systems.
- Node shutdowns (graceful shutdown with takeover) with cascading resource groups and NFS mounted file systems.
- Node failures with rotating resource groups and NFS mounted file systems.
- Node shutdowns (graceful shutdown with takeover) with rotating resource groups and NFS mounted file.

9.5.2 Test results

ATM LAN Emulation works properly with HACMP. Specific lessons learned include:

- The ATM network definition in HACMP must be set to "Private,” even though the network is used to connect clients.

All AIX ATM interfaces should recover automatically when the network is restored after failure. If the local switch goes away, the `atmstat` command should show the driver is in *Limbo* state. When the switch comes back, the RS/6000 should recognize this, recover, and reregister with the switch. If this does not happen, it is an AIX software problem that should be reported. There have been various problems with AIX interface recovery after the loss of the network connection, and users may see these things when they are doing HACMP implementation and fail over testing.

Chapter 10. Interface specific network options

Many RS/6000 systems have multiple network interfaces combining traditional and high speed TCP/IP interfaces on a single system. Up to version 4.3.3, AIX provides a single set of system-wide values, set using the `no` command, for the key IP interface network tuning parameters, thus making it impossible to tune a system that has widely differing network adapter interfaces. Although some applications, such as ADSM, provide proprietary tuning methods (overriding both the system defaults set by the `no` command and any ISNOs), most applications are still dependent on a “one size fits all” set of system-wide parameters that, again, does not work well on systems with disparate network interface speeds.

As new, high-speed (100 Mbps or more) network options became available, AIX system administrators learned these TCP/IP interfaces must be specially tuned to achieve good, high speed performance.

Therefore, system administrators faced performance trade-off decisions; perhaps tuning for one TCP/IP network interface while sacrificing performance on the other, or balancing the two with options that were not totally suited for either interface type. Ideally, system administrators should be able to tune for each TCP/IP interface individually for best performance. This feature is now included in AIX V 4.3.3.

Starting with Version 4.3.3 AIX offers a feature called Interface Specific Network Options (ISNO) which allows IP network interfaces to be custom tuned for the best performance. Values set for an individual interface take precedence over the system-wide values set with the network option (`no`) command. The feature is enabled (the default) or disabled for the whole system with the `no` command *use_isno* option. This single point ISNO disable option is included as a diagnostic tool to eliminate potential tuning errors if the system administrator needs to isolate performance problems.

10.1 Implementation overview

The AIX V4.3.3 programmers and performance analysts should note that the ISNO values will not show up in the socket; meaning, they cannot be read by getsockopt() until after the TCP connection is made. The interface this socket actually will be using is not known until the connection is complete, so the socket reflects that the system has no defaults. Once the connection is accepted, ISNO values are put into the socket.

There are five new parameters:

- rfc1323,
- tcp_nodelay,
- tcp_sendspace,
- tcp_recvspace, and
- tcp_mssdflt

These have been added for each supported network interface. When set for a specific interface, these values override the corresponding no option values set for the system. These parameters are available for all of the mainstream TCP/IP interfaces we checked -- Token Ring, FDDI, 10/100 Ethernet, and Gigabit Ethernet -- except the css# IP interface on the SP Switch. As a simple workaround, SP Switch users can set the tuning options appropriate for the switch using the system-wide `no` command, then use the ISNOs to set the values needed for the other system interfaces. ATM is supported and will work correctly with AIX V 4.3.3 and a software update.

Note

These options are set for the TCP/IP interface such as en0 or tr0, not for the network adapter interfaces, such as ent0 or tok0.

10.2 How to use the new options

The five new ISNO parameters cannot be displayed or changed using SMIT. The following are commands that can be used to first verify system and interface support and then set and verify the new values.

1. Verify general system and interface support using the `no` and `lsattr` commands.

   Make sure the use_isno option is enabled when using the following command or a variation:

   ```
   $ no -a | grep isno
   use_isno=1
   ```

   Make sure the interface supports the five new ISNOs when using the `lsattr -El` command:

   ```
   $ lsattr -E -l en0 -H
    attribute       value     description
   rfc1323                    N/A
   tcp_nodelay                N/A
   tcp_sendspace              N/A
   ```

```
tcp_recvspace              N/A
tcp_mssdflt               N/A
```

2. Set the interface specific values using either the `ifconfig` or `chdev` command. The `ifconfig` command sets values temporarily so that it is good for testing. The `chdev` command alters the ODM; so, custom values return after system reboots.

   For example, to set the *tcp_recvspace* and *tcp_sendspace* to 64 K and enable *tcp_nodelay,* use one of the following methods:

```
$ ifconfig en0 tcp_recvspace 65536 tcp_sendspace 65536 tcp_nodelay 1
$ chdev -l en0 -a tcp_recvspace=65536 -a tcp_sendspace=65536 -a \
  tcp_nodelay=1
```

   Or, assuming the `no` command reports a *rfc1323=1* global value, user root can turn rfc1323 off for all connections over en0 with the following commands:

```
ifconfig en0 rfc1323 0
chdev -l en0 -a rfc1323=0
```

   Verify the settings using the `ifconfig` or `lsattr` command:

```
$ ifconfig en0
en0: flags=e080863<UP,BROADCAST,NOTRAILERS,RUNNING,SIMPLEX,MULTICAST,GROUPRT,64BIT>
     inet 9.19.161.100 netmask 0xffffff00 broadcast 9.19.161.255
     tcp_sendspace 65536 tcp_recvspace 65536 tcp_nodelay 1 rfc1323 0
$ lsattr -El en0
rfc1323           0          N/A                   True
tcp_nodelay       1          N/A                   True
tcp_sendspace     65536      N/A                   True
tcp_recvspace     65536      N/A                   True
tcp_mssdflt                  N/A                   True
```

Note

In our simple tests, we were pleased to see that changes made via `chdev` were reflected in the ifconfig output. Specifically, our test machine was already set to 64 K send and receive space; so, we used `ifconfig` to set a 16 K value. We ran *ifconfig en0* to verify that setting. Next, we set the 64 K value using `chdev`, executed the `ifconfig en0` command, and discovered that the `ifconfig` output reflected the new 64 K value we'd just set using `chdev`.

10.3 References for the ISNO

More information about the interface specific network options (ISNO) can be found in the following:

- *AIX Version 4.3 Differences Guide,* SG24-2014
- *AIX Commands Reference, Volume 1 - 4,* GC23-2376

These are particularly useful for the `ifconfig` and `no` command pages. The following is an excerpt from the `ifconfig` command page. Here are two clarifications to the command page text:

1. The main purpose of RFC1323 is to allow TCP to increase its window size larger than 64 K bytes (controlled by tcp_recvspace) for large MTU adapters. Without RFC 1323, a 64 K MTU adapter could only have one packet outstanding, which results in very poor performance.
2. The tcp_sendspace option only affects send space buffering in the kernel. The tcp_recvspace value on the receiver system is the only value that affects TCP maximum window size.

In AIX Version 4.3.3 and later versions, the following network options, commonly known as ISNO (Interface Specific Network Options), can be configured on a per interface basis:

rfc1323 [0 I 1]	Enables or disables TCP enhancements as specified by RFC 1323, TCP “Extensions for High Performance”. A value of 1 specifies that all TCP connections using this interface will attempt to negotiate the RFC enhancements. A value of 0 disables rfc1323 for all connections using this interface. The SOCKETS application can override this ISNO and global behavior on individual TCP connections with the setsockopt subroutine.
-rfc1323	Removes the use of ISNO for rfc1323 for this network. A SOCKETS application can override the global behavior on individual TCP connections using the setsockopt subroutine.
tcp_mssdflt Number	Sets the default maximum segment size used in communicating with remote networks. If communicating over this interface, a socket uses Number as the value of the default maximum segment size.
-tcp_mssdflt	Removes the use of ISNO for tcp_mssdflt. The global value, manipulated via /usr/sbin/no, is used instead.

tcp_recvspace	Specifies the default socket buffer size for interface sockets receiving data. The buffer size affects the window size used by TCP. (See the `no` command for more information.)
-tcp_recvspace	Removes the use of ISNO for tcp_recvspace. The global value is used instead.
tcp_sendspace	Specifies the default socket buffer size for interface sockets sending data. The buffer size affects the window size used by TCP. (See the `no` command for more information.)
-tcp_sendspace	Removes the use of ISNO for tcp_sendspace. The global value is used instead.
tcp_nodelay [0 | 1]	Specifies that sockets using TCP over this interface follow the Nagle algorithm when sending data. By default, TCP follows the Nagle algorithm.
-tcp_nodelay	Removes the use of ISNO for the tcp_nodelay option.

Chapter 11. ATM performance considerations

IBMs first ATM-related announcement was for the RS/6000 in April 1994. Since then, RS/6000 hardware and AIX software have steadily evolved in their overall features, capabilities, and performance. As IBM customers have implemented ATM at various points in this timeline, it's probably safe to assume that the majority of RS/6000s running ATM today share some, but not all, of the characteristics of the ideal configuration. If performance is a concern in an existing ATM environment, a comparison against this optimal configuration should help in planning the attack for improvement. Those designing a new RS/6000 for attachment to an ATM network will want to choose a configuration as close to the ideal as possible.

11.1 The optimal configuration

The ideal RS/6000 platform for ATM performance has these general characteristics:

- The newest generation PCI bus RS/6000 model
- With properly-placed ATM 155 PCI bus adapter(s)
- Runs the latest AIX software level
- Runs Classical IP
- Runs a batch-style file transfer application (similar to TCP/IP's FTP)
- Uses the largest data unit size (for example, TCP/IP maximum transmission unit or MTU) possible
- Has been properly tuned for performance

11.2 Line speed versus data throughput

Most end users do not distinguish between line speed and data throughput. For example, if the adapter transmits at 100 Mbps, they assume that the throughput speed of their data is 100 Mbps. In reality, line speed (the raw transmission speed of the adapter) is quite different than the actual rate at which user data is exchanged across the network. All networking methods and applications impose overhead, and the impact varies with each situation. So, while the transmission speed of the adapter may be 100 Mbps, once all the protocol and application overhead for packaging and delivering the user information across the network has been considered, the final throughput speed of the end user's information is always less than the raw transmission speed of the adapter.

In the case of the Turboways 155 ATM adapter, for example, the raw network speed is 155 Megabits per second (Mbps). The speed is reduced by:

- SONET OC-3 signaling overhead
- ATM protocol cell header
- Communications protocol (for example, TCP/IP) packet header

First, SONET OC-3's 5.3 Mbps signaling overhead reduces the cell throughput to 149.7 Mbps. Next, ATM packs information in 53 byte cells. Each cell has a five byte ATM header, leaving 48 bytes for a payload of data and/or communications protocol overhead. Thus, assuming full cells, the five byte header consumes 9.4 percent of the available 149.7 Mbps bandwidth, thus reducing the maximum unidirectional transmission speed to 135.6 Mbps.

Note

The header is always five bytes; the percentage of bandwidth consumed by this overhead is higher if cells are partially filled.

Finally, the TCP/IP or UDP/IP protocol overhead for each packet is about 54 bytes; so, user-level payload transmission will not exceed 135 Mbps and will typically be 133 Mbps (16.6 Megabytes per second) for Classical IP. With ATM LAN Emulation Ethernet transmissions, packets are limited to 1500 bytes; so, seven packets must be created to send the same amount of user data as one 9180 Classical IP packet. As a result, the 54 byte TCP/IP overhead per packet consumes from 0.5 percent (with Classical IP) to 3.6 percent (with Ethernet LAN Emulation) of the network bandwidth. Transmissions of 120 to 125 Mbps are typical for 1500 byte MTUs on the PCI ATM adapters.

Figure 36 on page 255 illustrates the TCP/IP packet and ATM cell header overhead associated with the FTP of a 15,000 byte file over a Classical IP interface. In this simple example, two TCP/IP packets, each with a 54 byte header, are created for transmission across ATM. These data packets are then broken into 48 byte cells, each with a five byte ATM header. The SONET overhead is not depicted; it is incurred when the cells are transferred across the ATM network.

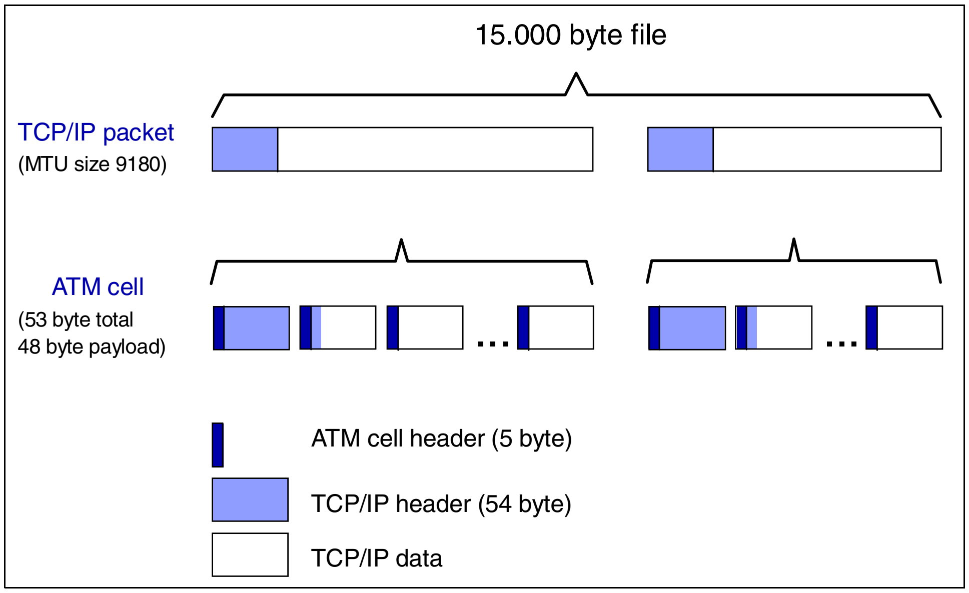


Figure 36. TCP/IP packet and ATM cell overhead for FTP of a file

Eleven 1500-byte TCP/IP packets would be created if the same file were transmitted over an ATM LAN Emulation Ethernet emulated LAN. This means nine extra 54 byte headers (totaling 486 bytes or 3888 bits) are transmitted as data across ATM for the Ethernet LAN Emulation environment. If ATM LAN Emulation must be used, a Token Ring environment would be a better choice (see 11.4.3, “Data transmission unit size” on page 259 for more information).

11.3 Test workload types

The *netperf* (network performance) application provides two basic types of tests for TCP and UDP:

- Stream test
- Request/response test (abbreviated as R/R)

In the stream test, one system continuously sends, and the other continuously receives. This is similar to normal FTP or rcp operations, except it is a memory-to-memory test; so, disk I/O (input/output) is not involved. Such a streaming test measures how fast the RS/6000 can stream the data across the network.

The R/R tests simulate client-server transactions; the client sends a request to the server, the server receives it and then responds. These ping-pong workloads typically involve small packets so that they are not as sensitive to MTU sizes as the streaming workloads. (For example, the processing overhead for a 512 byte message is the same with 1500 and 9180 byte MTUs because neither packet is filled). Therefore, R/R workload is affected most by a network round trip; so, netperf transaction tests measure the latency (or round trip) of the network. Tests of multiple sessions (for example, eight concurrent R/R sessions) can measure the capacity bottleneck. Single session R/R tests measure the latency for various send/receive sizes, and the multi-session R/R tests measure more the capacity because the test can saturate the CPU, the adapter, or the network.

CPU speed, adapter latency, and network speed are key factors in R/R environments. The RS/6000 PCI systems are better suited for these workloads for two reasons:

- The PCI adapters have much lower latency than the MCA adapters.
- The newer PCI based platforms also have the fastest processor speeds.

In large networks, the speed of the network components (routers, switches) also impact round trip latency.

Streaming tests can measure various types of TCP/IP traffic flows. In one-way or unidirectional traffic tests, system A sends to system B. With two-way or bidirectional traffic tests, system A sends to B, and B also sends to A over two separate TCP sessions.

Traditional, half-duplex LAN adapters can send and receive data but cannot do so simultaneously. In high-traffic streaming environments, enabling full duplex mode (so the adapter can send and receive simultaneously) improves performance because transmissions need not wait for traffic going the other direction. For example, 10/100 Ethernet adapter in 100 Mbps full duplex mode can stream unidirectional traffic at about 95 Mbps or bidirectional traffic at about 180 Mbps (the total of the two sessions). In 100 Mbps, half-duplex mode, the unidirectional stream is about 86 Mbps, but because the link is saturated, the bidirectional stream is also about 85 Mbps. Full-duplex performance is generally considered most important for busy server systems that need to send and receive data concurrently.

Even the unidirectional stream is lower (86 Mbps) because the TCP acknowledgment packets coming back across the half-duplex link must compete with data packets flowing the other way. When the media is in full-duplex mode, these acknowledgments can come back without contention, therefore, resulting in 95 Mbps throughput.

The netperf program was modified for bidirectional streaming to measure performance on full-duplex capable networks and network adapters. RS/6000 adapters that support full-duplex mode are ATM (it is always in this mode), Token Ring, 10/100 Ethernet (RJ45 twisted pair only, with full duplex configured), and Gigabit Ethernet.

11.4 Key performance considerations

Several software and hardware factors affect ATM performance to varying degrees.

Software

- AIX operating system level
- Upper layer protocols and applications (custom-written via sockets API, Classical IP, or LAN Emulation)
- Data transmission unit size

Hardware

- Adapter type and ATM network attachment capabilities/restrictions
- RS/6000 model and bus type
- Number of adapters and placement
- Disk read speed

Performance Tuning

- Upper layer protocols and application
- Overall system

11.4.1 AIX operating system level

Performance improvements are being continually added to AIX so that, as a rule of thumb, the newer the AIX software level, the better it performs, especially on SMP models. For example, the October 5, 1998 AIX V 4.3.2 announcement (U.S. letter 298-359) announced specific driver changes to improve TCP/IP performance with both the IBM Turboways 155 PCI ATM adapters (both the multimode fiber and UTP copper versions) and the IBM 10/100 Mbps Ethernet PCI adapter. In addition, AIX V 4.3.2 contains various TCP/IP enhancements that collectively improve performance for all environments, regardless of the physical network type. Specific changes have also been made recently to improve performance on SMP systems.

11.4.2 Upper layer protocols and applications

Starting with AIX V4.2.1, users may choose from three ATM upper-layer protocols, which are listed here in order of efficiency:

1. Custom-written directly to the ATM device driver
2. Classical IP
3. Token Ring LAN Emulation (running MTUs larger than 1500)
4. Ethernet LAN Emulation

With the extension to sockets in AIX V 4.2, users have an application programming interface (API) directly to the ATM's device driver. (Previously, user-written sockets applications had to go through TCP/IP first.) This programming interface provides access to ATM Quality of Service (QoS) and enables the possibility of video, voice, and data applications.

A user-written application using the socket's interface to write directly to the ATM device driver can have the highest performance potential because it can be custom written with performance in mind. (Of course, a poorly written application can also be a performance disaster; so, it depends on the programmer.) However, few RS/6000 administrators have the luxury of writing custom applications. Instead, they run third party-written applications, and the ATM protocol choice is normally either Classical IP or LAN Emulation.

Classical IP and LAN Emulation are two common ATM application interfaces that offer directly opposite approaches to ATM. Whereas, Classical IP offers a direct TCP/IP interface to the ATM network, thereby enabling applications, such as NFS, DCE, and a host of database and client server applications. LAN Emulation (often abbreviated LANE) provides software emulation of traditional Ethernet and Token Ring LANs to hide ATM from the upper-level application. With LAN Emulation, applications, such as SNA, NetBios, and IPX, run without change across an ATM network, thus providing users a means of coexistence during the migration to ATM networking. LAN Emulation takes frames constructed for Token Ring and Ethernet LANs and packages them for delivery across an ATM network.

Rule of thumb

Wherever possible, use Classical IP.

If LAN Emulation is required, remember that LAN Emulation and Classical IP can operate concurrently on one system; so, use them in combination. For example, if LAN Emulation is required for interoperation with non-ATM PC workstations, take advantage of Classical IP's faster performance and lower CPU utilization for nightly AIX-to-AIX or ADSM backups.

11.4.3 Data transmission unit size

Although LAN Emulation introduces another layer of overhead to applications, the more significant performance factor is packet size. Generally speaking, the larger the packet or data unit size, the better the performance. (This was illustrated in Figure 1 and will also be discussed further in the "Performance Tuning" topic.)

Classical IP specifies a 9180 default MTU (TCP/IP maximum transmission unit or packet) size, which results in good performance and low CPU usage. (The maximum MTU size is 65527.) With LAN Emulation, the packet size is limited to that specified by a real Ethernet or Token Ring LAN. The valid maximum IP packet size range for Ethernet is 60 to 1500 bytes. For 4 Mbps Token Rings, the range is 60 to 4056 bytes, and for 16 Mbps Token Ring, it is 60 to 17960 bytes. Therefore, as a rule of thumb, when possible, use Token Ring ELANs (LAN Emulation emulated LANs) rather than Ethernet. The Ethernet 1500 byte packet size will consume more system CPU resources to stream at the same rate because it must now send seven 1500 byte packets through the protocol and driver stack to deliver the same amount of user data as one 9180 byte ATM Classical IP packet.

Also, consider the nature of any applications running atop Classical IP or LAN Emulation. Generally speaking, batch, file transfer-style applications will perform better than highly-interactive applications that generate small packets because they make the most efficient use of the large packet size. Backup software, such as ADSM, adds additional processing overhead and must often handle small files, which reduces overall throughput. In most cases, ADSM will be the performance bottleneck, not the ATM network.

11.4.4 Adapter type

RS/6000 supports ATM 155 Mbps adapters for both Micro Channel (MCA) and Peripheral Component Interconnect (PCI) bus architectures. Table 5 on page 260 shows that both 155 Mbps adapter types will stream at around 133 Mbps (or roughly 16 MBps) using Classical IP, but the differences for Ethernet LAN Emulation are significant.

The table indicates that the ATM 155 PCI adapter, running roughly 120 Mbps, is the better choice for Ethernet LAN Emulation environments. While the ATM 155 MCA adapter performed well for Classical IP, the Ethernet 1500 byte MTU size used for LAN Emulation shows an adapter-related performance problem; it can only send/receive data at 66 Mbps because the speed of the processor on the adapter limits the data transfer speed for these small packet sizes.

Table 5. TCP Streaming throughput (in Mbps)

	Turboways 155 MCA	**Turboways 155 PCI**
Classical IP (MTU 9180)	133	133
LAN Emulation (MTU 1500)	66	120

Although Table 5 shows Ethernet results, one would expect better LAN Emulation performance for Token Ring ELANs configured for larger MTU sizes (the default is 1492), but a 4 K (4056) MTU is common, and the value can be set as high as 17960 bytes. For high MTU values, be sure to increase the ATM adapter PDU size. (See 11.4.8, “ATM performance tuning” on page 263 for more information.)

11.4.5 RS/6000 model, bus, and adapter placement guidelines

The number of adapter slots available is a clear limitation, but the speed of the system CPU and the number of CPUs will further determine how many adapters can be used in a particular model. Here are general hardware-specific considerations and recommendations for both MCA and PCI bus models:

- Use PCI bus architecture systems for ATM LAN Emulation environments. An MCA system with an ATM 155 MCA adapter will only stream data at 66 Mbps. Therefore, a PCI bus system is a much better choice.
- Carefully select SMP and SP MCA models if multiple ATM adapters are needed. The MCA SMP J40, J50, R50 (and SP node equivalents) systems scale poorly as adapters are added. This is not a MCA problem but rather a design limitation on the I/O controller chip on these systems. As a result, J40, J50, and R50 SMP models are not recommended for I/O intensive applications.
- Balance the I/O load across the buses on multi-bus models. The R50 has two MCA buses, and the J50 has an optional MCA expansion bus. For best performance, split the I/O load by putting the hard disk with the data to be transferred on one MCA bus and the ATM adapter on the other.
- Avoid placing high-speed adapters on a “secondary” PCI system bus. The F40 and H10 models have several slots on a secondary PCI bus. This secondary bus is bridged onto the primary PCI bus. For good performance, medium speed PCI adapters such as 10/100 Ethernet, FDDI, and ATM, should be placed on the primary bus.

Avoid using the F40 and H10 models as well as the 43P model 240. Choose the F50 and H50 models instead.

- Use S7A/S80, F50, H50, and H70 platforms for high-performance network attachment. These systems have good PCI bus I/O, and the adapter performance scales well as adapters are added to the system unit. The 43P models 140 and 150, and the 43P model 260 2-way, are also good performers.
- Use the PCI nodes that are based on the RS/6000 model H50 for SP systems. On SP wide nodes, avoid placing the ATM adapters on the secondary bus.

11.4.6 Maximum number of adapters and placement guidelines

Adapter placement is a critical factor in performance. Consult the latest *PCI Adapter Placement Reference*, SA38-0538, for detailed information on adapter limits and bus slot placement. As a rule of thumb, assume that the “medium speed” adapters (Fast Ethernet, ATM 155, and FDDI) will consume one CPU per adapter when streaming at media speed. This is true for Fast Ethernet and ATM LAN Emulation. Classical IP and FDDI actually use less CPU when configured for large packet sizes, but this is still a good guideline.

Using this performance sizing guideline, plan to use only one ATM or Fast Ethernet adapter in machines, such as the E30 or 43P models 140 or 150, two to three adapters on the two-way 43P model 260, and four adapters on F50 and H50 models. The newer H70 is a faster machine; so, it will be able to support up to six of these adapters.

On S7A/S80 models, generally plan for one of these adapters per CPU. In lab tests, a 262 MHz 12-way system with AIX V4.3.2 was able to run ten Fast Ethernets at 95 Mbps or fourteen ATM adapters (Classical IP) at 133 Mbps per adapter. This might seem backwards since it allows more adapters running at a higher speed, but it goes back to the MTU size concepts represented in Figure 36 on page 255. The 100 Mbps Fast Ethernet connection has more per-packet overhead than ATM because it sends more 1500 byte packets to transmit the same amount of data as ATM can transmit in one 9180 byte packet.

While ATM LAN Emulation tests on the S80 model had not been run at the time of this writing, conservative ATM LAN Emulation planners should estimate Ethernet ELANs to be able to support ten Fast Ethernets adapters. Ethernet LAN Emulation Clients are expected to perform within this Fast Ethernet range because they share the per-packet overhead caused by the1500 byte MTU. Also, while the single adapter performance reported in

Table 5 on page 260 is better than Fast Ethernet, multi-adapter Ethernet LAN Emulation performance is not expected to double or triple when two and three adapters are tested. Finally, the LAN Emulation driver overhead may actually result in fewer adapters supported. So, be conservative in estimates until ATM LAN Emulation performance results have been published for the S80.

11.4.7 Disk read speed

The speed of the hard disk can significantly limit the throughput for a high speed network adapter because the network transmission is delayed while data to be transferred is read from the local hard disk. Disk drives from the 1996 to 1997 time frame, for example, will stream data at 6 to 7 Megabytes per second (MBps). Older drives/controllers may run at only 2 to 4 MBps. In 1998 and 1999, fast/wide and fast/wide-ultra2 SCSIs on the newest RS/6000s will stream data at 12 to 14 MBps. Actual performance varies depending upon the specific disk and disk controller. Therefore, if disk performance is crucial, research disk options carefully.

As mentioned previously, ATM Classical IP can run at 133 Mbps, which equates to 16.6 MBps. So, if the disk in a Classical IP environment is running at 6 MBps, the disk will be the limiting factor. Striping the data across two or three disks will increase data read or write speed and will, in turn, improve performance over the network. For client-server applications, such as FTP, disk performance on both the client and the server affects throughput; the system with the slower disk will be the overall performance bottleneck.

Disk read speed can make a significant, but predictable, difference in the time it takes to complete a job. To illustrate this, suppose a user has 50 Gigabytes (50 GB) of information that must be transferred twice a week. The raw time calculations (excluding any other application, protocol, or network overhead) for disks running at the high end of these disk speeds (4, 7, and 12 MBps, respectively) are provided in the Table 6 on page 263.

Network speeds are typically measured in Megabits per second (Mbps), whereas disk sizes and speeds are measured in Megabytes per second (MBps). To make comparison easier, divide the ATM network speed (155 Mbps) by eight (there are eight bits per byte) for an equivalent ATM network speed of 19.3 MBps. The Classical IP throughput number of 16 MBps is a good gauge of transfer time because it compensates for ATM protocol (roughly 11 percent) and TCP/IP packet overhead.

Continuing with the example, fifty Gigabytes is 50,000 Megabytes. Assuming no reliance on disk speed, this data could be transferred by Classical IP at roughly 16 MBps in 52 minutes (.87 hours). In the Table 6, notice the impact of the disk speed on the time it takes to complete the job.

Table 6. Disk speed impact on throughput

Data Amount	**Disk Speed**	**Time (in seconds)**	**Time (in hours)**
50000 MB	4 MBps	12,500	3.47
50000 MB	7 MBps	7,143	1.98
50000 MB	12 MBps	4,167	1.16

This simple example assumes an ideal Classical IP environment, where throughput was calculated with no other system or network load. But, the overall lesson is that network speed, disk speed, and the application selection can have a significant affect on the overall throughput of user data.

11.4.8 ATM performance tuning

The default ATM-specific configuration values are conservative, sacrificing performance for the sake of interoperability. Three levels of tuning are available affecting:

- The overall adapter
- The ATM protocol (Classical IP or LAN Emulation)
- The specific interface

The adapter, Classical IP, and ATM LAN Emulation parameters are similar, but implementation and the SMIT screen wording differ slightly between PCI and MCA bus systems.

Refer to the *AIX Performance Tuning Guide*, SR28-5930, for complete information on tuning ATM. The following provides are two key considerations for Micro Channel adapters. (The guide describes MCA adapter buffer tuning.)

If a TCP/IP MTU larger than the default is implemented on Micro Channel systems, increase the ATM Adapter Maximum PDU size (bytes) to accommodate it. The PDU should be eight bytes larger than the MTU so that the SMIT default (SMIT fastpath is: *chg_atm*) value of 9188 is perfect for the default 9180 MTU. (On systems prior to AIX V 4.2.1, the default is also ideal, but the eight-byte difference is hidden; so, the values match.) The PDU value translates to adapter memory reserved for buffer space. Running a PDU size that is excessively larger than the MTU works, but it is a waste of adapter memory. Since multiple IP interfaces can be configured on a single ATM adapter, make sure the PDU setting is at least eight bytes larger than the largest MTU to be run on that adapter. There is no such PDU parameter for PCI bus adapters.

On systems running the ATM Forum’s User-Network Interface (UNI) version 3.1, the MCA 155 ATM adapter’s “Best Effort Peak Rate” parameter in the same menu sets the user definable rate queue for Best Effort connections and, as such, defines the maximum data rate for the adapter. The default of 1500 Kbps equates to 1.5 Mbps and actually throttles transmission speed of the adapter. This value is intended to slow the transmission speed in environments where, for example, the local system always communicates with ATM systems attached at 25 Mbps. To request full bandwidth, change the parameter to 155,000 (meaning 155 Mbps), which is the maximum value. This Best Effort value is for traffic, such as Classical IP and ATM LAN Emulation, that does not require quality of service and bandwidth reservation. There are four best effort rate queues; three predefined values accommodate 25 Mbps, 100 Mbps, 155 Mbps, and one that is user definable in SMIT. The adapter will automatically make connections using the fastest of the four queues that does not exceed the requested (user-specified) rate. There are no PDU size or Best Effort Peak parameters for the PCI bus adapters.

There are Best Effort Rate fields in both Classical IP and LAN Emulation client SMIT menus; so, values can be set independently on an interface-by-interface basis. It is important that control is independent because Classical IP connections are not affected even though ATM LAN Emulation may be interoperating with real Ethernet or Token Ring LAN-attached devices that throttle at 10 or 15 Mbps.

On both MCA and PCI bus systems, the “Best Effort Bit Rate (UBR) in Kbits/sec” option sets the maximum rate for the Classical IP interface. The SMIT (SMIT fastpath is: *chinet*) default value of zero essentially means this interface will follow the Best Effort Peak Rate set for the adapter. Any non-zero value up to the adapter rate can be set to limit the throughput for this interface.

The “Forward/Backward Peak Rate (Kbits/sec)” in the *smitty name_atmle_ls* menu performs a similar function for ATM LAN Emulation. In this case, the MCA default value (155,000) specifies full bandwidth, but the PCI bus default of 25,600 (25.6 Mbps), chosen for interoperability with 25 Mbps ATM attached systems, severely restricts the throughput on a 155 Mbps adapter.

Furthermore, *Maximum Frame Size (bytes)* works with the upper-level protocol’s data unit value (for example, TCP/IP’s MTU, SNA’s rdu, etc.) to define the maximum frame size on the ELAN. For Token Ring, valid frame size values are: “Unspecified”, 1516, 4544, and 18190. For Ethernet, frame size choices are “Unspecified” and 1516.

11.4.9 TCP/IP performance tuning

Protocols, such as TCP/IP, were originally designed to run on traditional Ethernet and Token Ring local area networks running in the 4 to 16 Mbps speed range. Therefore, TCP/IP administrators must always tune for performance to take advantage of network speeds exceeding 40 Mbps. Also, remember to tune the operating system, ATM parameters, and any third party applications (such as database) that may be running atop the upper-level protocol (TCP/IP in most cases). Key TCP/IP tuning parameters, which are viewed and set using the AIX `no` (network options) command, include *tcp_sendspace*, *tcp_recvspace*, *udp_sendspace*, *udp_recvspace*, *sb_max*, and *rfc1323*.

Up until AIX Version 4.3.3, systems have a single set of global network options that are viewed with the `no -a` command and used by applications such as *ftp* and *rcp,* that do not have built-in tuning options. This “one size fits all" does not work well with widely differing networks. Ideally, these values could be set independently for each interface, enabling fine tuning for each network type. This feature is now supported starting with AIX Version 4.3.3.

Users must adjust the global network options for best performance for all AIX systems up to, and including, AIX V 4.3.2. If the machine has a wide variety of adapter types, this may be a compromise to find values that will work well with all the networks.

Performance tuning often means making trade-offs and balancing the needs of key applications because parameters affect all TCP/IP communications on the RS/6000, not just the ATM interface. For example, tuning to optimize performance for a high-speed ATM network interface may mean sacrificing performance on a 10 Mbps Ethernet interface on the same system.

Consider an RS/6000 SP node with a 10 Mbps Ethernet interface and a very high-speed SP Switch adapter. The Ethernet interface performs best at the system defaults, but the SP Switch adapter performs best with RFC 1323 enabled, a 64 K MTU size, and the *tcp_sendspace* and *tcp_recvspace* to very large values. Setting the send and receive space global network options to very large values affects both the Ethernet and SP Switch adapter interfaces; it can cause TCP to overrun the transmit queue on the small (1500 byte MTU) Ethernet adapter. Setting tcp_sendspace to 250,000 bytes means four packets for the 64 K MTU adapter, but it also means 166 packets for a 1500 byte MTU adapter. In this scenario, several concurrent sessions might overrun the transmit queue on the Ethernet adapter.

As a guideline, set the *tcp_sendspace* and *tcp_recvspace* to roughly ten times the adapter MTU size. This setting allows ten MTU size packets to be in transit to the receiver (or buffered at the receiver) for good pipelining of packets through the network. For Ethernet or ATM LAN Emulation with a 1,500 byte MTU size, the guideline of 15,000 bytes means the AIX default of 16,384 is perfect. Under the guideline, the ATM default 9,180 MTU becomes 91,800 bytes. However, since the maximum supported non-RFC1323 window size is 64 K, using this maximum (65,536) *tcp_sendspace* and *tcp_recvspace* value works well.

Generally speaking, the *sb_max* network option should be set to at least twice the *tcp_recvspace* value. This ensures sufficient buffer space for the incoming data. The *sb_max* value defines actual buffer space; so, it needs to be larger than *tcp_recvspace*. If a 1,100 byte packet is put into a 2 K buffer, the system allocates space based on the buffer size used, not the amount of data in the buffer.

Normally, administrators should enable the *rfc1323* option for adapters that support large MTU sizes. A word of caution: These adapter can perform poorly unless the TCP rfc1323 option is enabled. When this option is enabled, the TCP 16 bit maximum window size (64 K or, specifically, 65,536 bytes) is extended to 32 bits, thus enabling much larger window sizes. Administrators are cautioned because of the following situation. Suppose the system is configured for ATM's maximum MTU value (65,527), and the *tcp_recvspace* is 65536. In this scenario, TCP sends only one packet to the receiver, then it stops and waits for an acknowledgment (ACK) packet from the target system. This results in very poor performance. When *rfc1323* is enabled, and *tcp_recvspace* and *tcp_sendspace* are set to 655,360 (about ten times the MTU size), performance is great. RFC1323 cannot always be used. (For example, when the receiving system does not support it.) For best performance in these non-RFC 1323 ATM environments, leave the default MTU size (9180) or, at least do not set it larger than 16 K.

11.4.10 UDP performance tuning

The maximum size of a UDP packet is 65,536 bytes minus the UDP header. UDP is a datagram protocol; so, the entire message (datagram) must be copied into the kernel on a "send" call as one atomic operation. Therefore, the *udp_sendspace* value must be greater than, or equal to, the data size. The IP protocol fragments datagrams into smaller packets, as necessary, based upon the MTU size of the interface. On an Ethernet interface, for example, IP fragments an 8 K datagram into 1,500 byte packets. Because UDP implements no flow control, all packets given to UPD are passed to IP (where they may be fragmented) and then placed directly on the device driver transmit queue. There is no accounting for buffers on the transmit queue in AIX with respect to consuming *udp_sendspace*; so, make sure the *udp_sendspace* is large enough to handle checks done on the send call.

On the receive side, the incoming datagram (or datagram fragment) is normally received into a buffer that is large enough to hold the largest possible packet from this device. Ethernet, for example, uses 2 K (2,048 byte) receive buffers; so, even if the incoming packet is maximum MTU size of 1,500 bytes, only 73 percent of the buffer is utilized. IP queues incoming fragments until the full UDP datagram is reassembled and then passes it to UDP. UDP puts the incoming datagram on the receiver's socket. Should the total buffer space in use on this socket exceed *udp_recvspace*, the entire datagram will be discarded. This shows up in `netstat -s` command output as "dropped due to full socket buffers" errors. Because the communication subsystem accounts for buffers used rather than the contents of the buffers, users must consider this when setting the *udp_recvspace* value. The Ethernet 8 K datagram described in the previous example is fragmented into six packets that would use six 2,048-byte receive buffers. Thus, the total amount of socket buffer consumed by this one 8 K datagram is 6*2048=12,288 bytes. This is why the *udp_recvspace* must be adjusted depending upon the incoming buffering efficiency. The efficiency varies depending upon the datagram size and device driver. If the application generates a 64 byte datagram, a 2 K buffer is consumed for each.

Users must also consider the number of datagrams that may be queued onto this one socket. For example, NFS server receives all client UDP packets at on, e well known socket. If the queue depth of this socket were 30 packets, then 368,640 bytes (the result of 30 * 12,288) of the udp_recvspace are consumed for NFS systems running 8K datagrams. (Note that NFS Version 3 allows up to 32 K datagrams.)

In all cases, *sb_max* must be set to be greater than, or equal to, *udp_sendspace*/*udp_recvspace*.

In summary, it is wise to set *udp_sendspace* to 65,536. The best *udp_recvspace* setting is harder to compute, as it varies by network adapter type, UDP sizes, and number of datagrams queued to the socket. Since packets exceeding the receive space are discarded, set udp_recvspace to a larger value. For applications, such as NFS that use a well known port (socket), this number must be much larger. The udp_recvspace parameter is really the only "flow control" tuning knob available for UDP. For example, it prevents all the mbufs from being consumed by an UDP application that blasts out UDP packets at a high rate. Use the `netstat -s` command to monitor the *dropped due to full socket buffers* statistic and increase the *udp_recvspace* if there is a high ratio of these errors to the total UDP packets received.

Use this ratio to determine whether tuning is needed for NFS. Increase *udp_sendspace* and the number of nfsd daemons (via the *nfs_max_threads* nfso option). If packets are still dropped, increase the NFS server UDP socket buffer size (via the *nfs_socketsize* nfso option). Note that this size must be less than *sb_max*. Increase it to compensate.

11.4.11 Applying this information to other networking methods

Much of the AIX operating system, RS/6000 hardware, and performance tuning information applies directly to other medium-speed networking methods, such as FDDI and Fast Ethernet. Proper adapter placement and RS/6000 model selection cannot be stressed too much; it is equally critical for all high-speed adapters.

Some of the receive buffer sizes used by various drivers are:

- **Ethernet**: 2 K (4 K buffers for Gigabit "Jumbo Frames")
- **Token Ring**: 2 K for interfaces using the default MTU1492
- **FDDI**: 4 K (Note that it will take two 4 K buffers to receive a maximum size FDDI frame of 4,352 bytes. PCI FDDI provides a SMIT option to use 8 K receive buffers.)
- **ATM**: 8 K buffers for MCA and 4 K buffers for PCI adapters

While Classical IP and ATM LAN Emulation are unique to ATM, the lessons demonstrated about application selection also apply in a general fashion. Many of the key network protocol differences have been discussed here. The network protocol overhead and frame size limitations are a part of the protocol architecture and, as such, are a part of the performance equation that must be accepted because they cannot be changed without changing the networking method itself.

Appendix A. Planning check list

Table 7. Planning check list

Activity	Y/N	Task Owner	Due Date	Comments
Verify that the ATM adapter is supported in the intended RS/6000.				
Define key applications. Ensure that no ATM restrictions apply.				
Determine the hardware and software needed. Include port speeds and connectors on the ATM switch so that matching adapters and cables can be ordered.				
For LAN Emulation, ensure that the appropriate LAN Emulation Services (LES, BUS, and LECS) are available within the network. For MPOA, ensure an MPOA server exists within the network.				
Create a Virtual LAN topology diagram for each Logical IP Subnet and Emulated LAN. Include information about interface and adapter numbers and about MAC, ATM, and upper-layer protocol addresses.				

Activity	Y/N	Task Owner	Due Date	Comments
Create hard copies of the SMIT menus to be used as a worksheet for ATM configuration. (The SMIT Image (F8) function provides a quick screen save to the smit.log file.)				
Fill in the SMIT configuration worksheets as much as possible. (There will likely be gaps that must be filled in by the local switch administrator.) Use these worksheets for SMIT configuration of the adapter and during the installation and configuration process.				
Identify the person responsible for local switch configuration. Determine values needed to complete the SMIT configuration worksheets (for example, virtual channel types, values, and adapter MAC addresses).				
Determine which tests will be run when the system is configured to verify a working connection across ATM.				

Activity	Y/N	Task Owner	Due Date	Comments
Consider performance. Determine what performance tuning will be done. If using LAN Emulation as a method of increasing bandwidth to a busy server system, verify that the network is designed to meet this goal.				
Arrange for someone with switch configuration authority to be available during system installation to work out any configuration mismatches.				
Once a successful connection has been made, leave a hard copy of the SMIT parameters with the system administrator.				

Appendix B. Detailed adapter information

The following sections cover more technical details about the currently supported ATM adapters.

B.1 Turboways 155 PCI MMF ATM adapter F/C 2988

The TURBOWAYS 155 PCI Multi-Mode Fiber (MMF) Asynchronous Transfer Mode (ATM) adapter provides the interface between the ATM 155 Mbit/sec fiber-optics network and the PCI Bus in your system.

Table 8. PCI MMF ATM adapter F/C 2988

Item	Description
FRU number	21H3890
Bus architecture	PCI 2.0
PCI operation	5V, 32 bit, 33 MHz, Bus master
Card type	Half
Connector	ANSI specified SC duplex
Wrap plug	21H3547 or 16G5609
Cables	62.5 micron, multi-mode fiber-optic, customer provided
EMC	Class B FCC
ATM	Clocking local or remote source, SDH, or SONET
ATM operation	No microcode, 2048 VCs, segmentation/reassembly on card

B.2 Turboways 155 PCI UTP ATM adapter F/C 2963

The TURBOWAYS 155 PCI Unshielded Twisted-Pair (UTP) Asynchronous Transfer Mode (ATM) adapter provides the interface between the ATM 155 Mbit/sec UTP network and the PCI Bus in your system.

Table 9. PCI UTPATM adapter F/C 2963

Item	Description
FRU number	21H7977
Bus architecture	PCI 2.0
PCI operation	5V, 32 bit, 33 MHz, Bus master
Card type	Half
Connector	RJ-45
Wrap plug	Supplied with adapter (RJ-45 Pins 1-7, 2-8)
Cables	UTP (Cat 5) or STP up to 100 meters, customer provided
EMC	Class B FCC
ATM	Clocking local or remote source, SDH, or SONET
ATM operation	No microcode, 2048 VCs, segmentation/reassembly on card

B.3 Turboways155 ATM adapter F/C 2989

The TURBOWAYS 155 Asynchronous Transfer Mode (ATM) adapter provides the interface between the ATM 155 Mbit/sec fiber-optics network and the PCI Bus in your system.

Table 10. Turboways 155 ATM adapter F/C 2989

Item	**Description**
FRU number	21H7043
Bus architecture	Micro channel 80 Mbytes/second
MCA operation	5V, 32 bit, 40 MHz, Bus master
Card type	Card type 5
Connector	ANSI specified SC duplex
Wrap plug	21H3547 or 16G5609
Cables	62.5 micron, multi-mode, fiber-optic duplex, customer provided
EMC	Class A FCC
ATM	Clocking local or remote source, SDH, or SONET
ATM operation	No microcode, 2048 VCs, segmentation/reassembly on card

Appendix C. Special notices

This publication is intended to help customers and system engineers install, customize and use ATM support of the RS/6000. The information in this publication is not intended as the specification of any programming interfaces that are provided by AIX. See the PUBLICATIONS section of the IBM Programming Announcement for AIX for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM ("vendor") products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no

AIX	APPN
AS/400	CT
IBM	Netfinity
Micro Channel	RISC System/6000
RS/6000	SP
System/390	400

Appendix D. Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

D.1 IBM Redbooks

For information on ordering these publications see “How to get IBM Redbooks” on page 285.

- *AIX Version 4.3 Differences Guide,* SG24-2014
- *Asynchronous Transfer Mode (ATM) Technical Overview*, SG24-4625
- *ATM Configuration Examples*, SG24-2126
- *Internetworking over ATM: An Introduction*, SG24-4699
- *RS/6000 and Asynchronous Transfer Mode*, SG24-4796

D.2 IBM Redbooks collections

Redbooks are also available on the following CD-ROMs. Click the CD-ROMs button at http://www.redbooks.ibm.com/ for information about all the CD-ROMs offered, updates and formats.

CD-ROM Title	**Collection Kit Number**
System/390 Redbooks Collection	SK2T-2177
Networking and Systems Management Redbooks Collection	SK2T-6022
Transaction Processing and Data Management Redbooks Collection	SK2T-8038
Lotus Redbooks Collection	SK2T-8039
Tivoli Redbooks Collection	SK2T-8044
AS/400 Redbooks Collection	SK2T-2849
Netfinity Hardware and Software Redbooks Collection	SK2T-8046
RS/6000 Redbooks Collection (BkMgr)	SK2T-8040
RS/6000 Redbooks Collection (PDF Format)	SK2T-8043
Application Development Redbooks Collection	SK2T-8037
IBM Enterprise Storage and Systems Management Solutions	SK3T-3694

D.3 Other resources

These publications are also relevant as further information sources:

- *AIX Commands Reference, Volume 1 - 4,* GC23-2376

- *AIX Performance Tuning Guide*, SR28-5930
- *HACMP V4.3 AIX: Administration Guide*, SC23-4279
- *HACMP V4.3: Concepts and Facilities*, SC23-4276
- *HANFS V4.2.2 AIX: Installation and Administration Guide*, SC23-1946
- *PCI Adapter Placement Reference*, SA38-0538

D.4 Referenced Web sites

These Web sites are also relevant as further information sources:

- `http://www.atmforum.com/`
  ATM Forum
- `http://www.rfc-editor.org/rfc.html`
  RFC information
- `ftp://ftp.isi.edu/`
  RFC download source
- `http://www.rs6000.ibm.com/resource/hardware_docs/`
  RS/6000 hardware documentation
- `http://www.rs6000.ibm.com/doc_link/en_US/a_doc_lib/aixbman/prftungd/toc.htm`
  AIX Versions 3.2 and 4 Performance Tuning Guide
- `http://www.rs6000.ibm.com/resource/technology/atmsocks/atmnew.html`
  ATM Extensions to the Socket Programming Interface in AIX 4.2
- `http://www.networking.ibm.com/per/perprod.html`
  Raleigh ATM Performance Web site

IBM internal sites:

- `http://wtscpok.itso.ibm.com/~redweb/SG245120/abstract.html`
  RS/6000 Systems Handbook, SG24-5120
- `http://rs6knet.sl.dfw.ibm.com/atm.html`
  ATM Site Washington Systems Center
- `http://rs6knet.sl.dfw.ibm.com/ATM/best_atm_performer.html`
  Whitepaper: *Best Performing RS/6000 for ATM Networks*
- `http://rs6knet.sl.dfw.ibm.com/ATM/atmchart.pdf`
  RS/6000 ATM Support Summary
- `http://rs6knet.sl.dfw.ibm.com/ATM/HA_LANE.pdf`
  HACMP and ATM LAN Emulation Test

- `http://rs6knet.sl.dfw.ibm.com/ATM/HACMP-ATM.pdf`
  HACMP and ATM Classical IP Testing Report
- `http://rs6knet.sl.dfw.ibm.com/ATM/atmlane_article.pdf`
  ATM LAN Emulation in AIX

How to get IBM Redbooks

This section explains how both customers and IBM employees can find out about IBM Redbooks, redpieces, and CD-ROMs. A form for ordering books and CD-ROMs by fax or e-mail is also provided.

- **Redbooks Web Site** `http://www.redbooks.ibm.com/`

  Search for, view, download, or order hardcopy/CD-ROM Redbooks from the Redbooks Web site. Also read redpieces and download additional materials (code samples or diskette/CD-ROM images) from this Redbooks site.

  Redpieces are Redbooks in progress; not all Redbooks become redpieces and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

- **E-mail Orders**

  Send orders by e-mail including information from the IBM Redbooks fax order form to:

	e-mail address
In United States or Canada	pubscan@us.ibm.com
Outside North America	Contact information is in the "How to Order" section at this site: `http://www.elink.ibmlink.ibm.com/pbl/pbl`

- **Telephone Orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	Country coordinator phone number is in the "How to Order" section at this site: `http://www.elink.ibmlink.ibm.com/pbl/pbl`

- **Fax Orders**

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	Fax phone number is in the "How to Order" section at this site: `http://www.elink.ibmlink.ibm.com/pbl/pbl`

This information was current at the time of publication, but is continually subject to change. The latest information may be found at the Redbooks Web site.

IBM Intranet for Employees

IBM employees may register for information on workshops, residencies, and Redbooks by accessing the IBM Intranet Web site at `http://w3.itso.ibm.com/` and clicking the ITSO Mailing List button. Look in the Materials repository for workshops, presentations, papers, and Web pages developed and written by the ITSO technical professionals; click the Additional Materials button. Employees may access `MyNews` at `http://w3.ibm.com/` for redbook, residency, and workshop announcements.

IBM Redbooks fax order form

Please send me the following:

Title	**Order Number**	**Quantity**

First name	Last name	
Company		
Address		
City	Postal code	Country
Telephone number	Telefax number	VAT number

☐ Invoice to customer number

☐ Credit card number

Credit card expiration date	Card issued to	Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

Glossary

Anisochronous. Literally, “not equal”. Used in the communications engineering context to mean bit streams of unequal rates (this causes a problem in time division multiplexing systems).

Asynchronous. Any two events that are not tied together exactly in time are said to be asynchronous.

Asynchronous Transfer Mode (ATM). A transfer mode in which the information is organized into cells. It is asynchronous in the sense that the recurrence of cells containing information from an individual user is not necessarily periodic.

ATM Peer-to-Peer Connection. A virtual channel connection (VCC) or a virtual path connection (VPC).

ATM User-to-User Connection. An association established at the ATM layer to support communication between two or more ATM service users (that is, between two or more next higher layer entities or between two or more ATM entities). The communication over an ATM layer connection may be either bidirectional or unidirectional. The same virtual channel identifier (VCI) is used for both directions of a connection at an interface.

ATM Layer Link. A section of an ATM layer connection between two active ATM layer entities (ATM entities).

ATM Link. A virtual path link (VPL) or a virtual channel link (VCL).

ATM Traffic Descriptor. A generic list of traffic parameters that can be used to capture the intrinsic traffic characteristics of a requested ATM connection.

Broadcast. A generic list of traffic parameters that can be used to capture the intrinsic traffic characteristics of a requested ATM connection.

Cell. ATM layer protocol data unit.

Cell Header. ATM layer protocol control information.

Connection Admission Control (CAC). The set of actions taken by the network at the call setup phase for re-allocation can be accommodated). Routing is part of connection admission control actions.

Congestion Control. The set of actions taken to relieve congestion by limiting its spread and duration.

Connectionless Service. A service that allows the transfer of information between service users without the need for end-to-end call establishment procedures.

Constant Bit Rate Service. A type of telecommunication service characterized by a service bit rate specified by a constant value.

Framed Interface. An interface where the serial bit stream is segmented into periodic physical frames. Each frame is divided by a fixed partition into an overhead and an information payload portion.

General Broadcast Signaling Virtual Channel. A virtual channel independent of service profiles and used for broadcast signaling.

Heterochronous. If two bit streams have nominally different bit rates, they are said to be heterochronous.

Isochronous. Literally “in the same time”. An sochronous bit stream is one that goes at a constant rate. The term isochronous is often used colloquially to mean “digitally encoded voice”. The term is not often used in the world of data communications but is a common term in the voice communications and engineering context.

Jitter. Undesirable variations in the arrival time of a transmitted digital signal.

Mesochronous. The Greek prefix “meso” means “middle”. If two isochronous bit streams have no precisely controlled relationship, but have exactly the same bit rate, they are called mesochronous.

If two bit streams start at some point of origin as synchronous streams and arrive at some destination by different routes or networking schemes, they will be mesochronous at the point of arrival.

Meta-Signaling. The procedure for establishing, checking and releasing signaling virtual channels.

Multipoint-to-Point Connection. A multipoint-to-point connection consists of a simple tree topology considered as a root node connected to many leaves. A multipoint-to-point connection has zero bandwidth from the root node to the leaf nodes and a non-zero return bandwidth from the leaf nodes to the root node.

Multipoint-to-Multipoint Connection. A multipoint-to-multipoint is a collection of ATM VC or VP links and their associated endpoint nodes. Any information sent on the connection by a node is received by all of the other nodes. A receiving endpoint node cannot distinguish which other endpoint sent the information unless some higher layer information is present for the purpose.

Network Node Interface. The interface between two ATM switches.

Network Parameter Control. The set of actions taken by the network to monitor and control traffic at the Inter-Network Node interface to protect network resources from malicious, as well as unintentional, misbehavior by detecting violations of negotiated parameters and taking appropriate actions.

OA&M. Operations, Administration, and Maintenance

OA&M Cell. A cell that contains ATM LM information. It does not form part of the upper layer information transfer.

Packet. In data communication, a sequence of binary digits, including data and control signals, that is transmitted and switched as a composite whole. Synonymous with data frame.

In ATM, "An information block identified by a label at layer 3 of the OSI reference model."

Packet Transfer Mode (PTM). A transfer mode in which the transmission and switching functions are achieved by packet oriented techniques, so as to dynamically share network transmission and switching resources between a multiplicity of connections.

Physical Connection. The ability of two connectors to mate and make electrical contact. In a network, devices that are physically connected can communicate only if they share the same protocol. See also *logical connection*.

Physical Layer. In the Open Systems Interconnection reference model, the layer that provides the mechanical, electrical, functional, and procedural means to establish, maintain, and release physical connections over the transmission medium.

Plesiochronous. Literally "nearly the same". If two bit streams have nominally the same bit rate, but are controlled by different clocks, they will have bit rates that are nearly the same, but not quite.

Point-to-Multipoint Connection. A point-to-multipoint connection is a collection of associated ATM VC or VP links and associated endpoint nodes with the following properties:

1. One ATM link, called the root link, serves as the root in a simple tree topology. When the root node sends information, all of the remaining nodes on the connection, called leaf nodes, receive copies of the information.

2. Each of the leaf nodes on the connection can send information directly to the root node. The root node cannot distinguish which leaf node is sending the information without additional (higher layer) information.

3. Leaf nodes cannot communicate directly with each other.

ATM Forum Phase 1 signaling does not support traffic (except OA&M cells) sent from a leaf to the root.

Port. (1) An access point for data entry or exit. (2) A connector on a device to which cables for other devices such as display stations and printers are attached. Synonymous with socket.

Protocol. (1) A set of semantic and syntactic rules that determines the behavior of functional units in achieving communication. (2) In SNA, the meanings of, and the sequencing rules for, requests and responses used for managing the network, transferring data, and synchronizing the states of network components. (3) A specification for the format and relative timing of information exchanged between communicating parties.

Protocol Control Information. Information exchanges between corresponding entities, using a lower layer connection to coordinate their joint operation.

Protocol Data Unit (PDU). A unit of data specified in a layer protocol and consisting of protocol control information and layer user data.

Segment. A single ATM link or group of interconnected ATM links of an ATM connection.

Selective Broadcast Signaling Virtual Channel. A virtual channel allocated to a service profile and used for broadcast signaling.

Signaling Virtual Channel. A virtual channel for transporting signaling information.

Sublayer. A logical sub-division of a layer.

Switched Connection. A connection established by signaling.

Synchronous. Literally “locked together”. When two bit streams are said to be synchronous, it is meant that they are controlled by the same clock and are in the same phase.

Telephone Twisted Pair. One or more twisted pairs of copper wire in the unshielded voice-grade cable commonly used to connect a telephone to its wall jack. Also referred to as “unshielded twisted pair”.

Unassigned Cell. A cell identified by a standardized virtual path identifier (VPI) and virtual channel identifier (VCI) value, which has been generated and does not carry information from an application using the ATM layer service.

Unshielded Twisted Pair (UTP). See *telephone twisted pair*.

Usage Parameter Control. The set of actions, taken by the network to monitor and control traffic at the User Network Interface, to protect network resources from malicious, as well as unintentional, misbehavior by detecting violations of negotiated parameters and taking appropriate actions.

Virtual Channel (VC). A concept used to describe unidirectional transport of ATM cells associated by a common unique identifier value.

Virtual Channel Connection. A concatenation of virtual channel links that extends between two points where the adaptation layer is accessed.

Virtual Channel Link. A means of unidirectional transport of ATM cells between a point where a virtual channel identifier value is assigned and the point where that value is translated or removed.

Virtual Path. A concept used to describe the unidirectional transport of ATM cells belonging to virtual channels that are associated by a common identifier value.

Virtual Path Connection. A concatenation of virtual path links that extends between the point where the virtual channel identifier values are assigned and the point where those values are translated or removed.

Virtual Path Link. The group of virtual channel links, identified by a common value of the virtual path identifier, between the point where the VPI value is assigned and the point where the VPI value is translated or removed.

Virtual Path Switch. A network element that connects VPLs. It translates VPI (not VCI) values and is directed by control plane functions. It relays the cells of the VP.

Virtual Path Terminator. A system that unbundles the VCs of a VP for independent processing of each VC.

VLAN. (Virtual LAN) Generically, a group of devices that use hardware-based switching to communicate as if they were attached to the same LAN as a single broadcast domain. ELANs and LISs are VLAN implementations in ATM networks. In the non-ATM world, some LAN switches also support VLAN creation by linking ports through administrative procedures.

Abbreviations and acronyms

AAL	ATM Adaptation Layer
ABR	Available Bit Rate
ACM	Address Complete Message
ACT	Activity Bit
ADPCM	Adaptive Differential Pulse Code Modulation
AFI	Authority and Format Identifier.
AHFG	ATM-attached Host Functional Group (MPOA SWG)
Ai	Signalling ID Assigned by Exchange A
AII	Active Input Interface (used in UNI PMD specs for Copper/Fiber)
AIM	ATM Inverse Multiplexer AIR Additive Increase Rate
AIS	Alarm Indication Signal (UNI Fault Management)
AIS-E	Alarm Indication Signal - External
AMI	Alternate Mark Inversion
ANI	Automatic Number Identification
ANM	Answer Message
ANSI	American National Standards Institute
AOI	Active Output Interface (Used in UNI PMD specs for Copper/FIber)
API	Application Programming Interface
APPN	Advanced Peer to Peer Network
ARE	All Routes Explorer
ARP	Address Resolution Protocol
ARQ	Automated Repeat Request
ASE	Application Service Element
ASIC	Application Specific Integrated Circuit
ASN	Abstract Syntax Notation
ASN.1	Abstract Syntax Notation One
ASP	Abstract Service Primitive
ATD	Asynchronous Time Division
ATE	ATM Terminating Equipment (SONET)
ATM	Asynchronous Transfer Mode
ATS	Abstract Test Suite
ATM	Asynchronous Transfer Mode
ATMARP	ATM Address Resolution Protocol
AUU	ATM User-to-User
AVSSCS	Audio-Visual Service Specific Convergence Supplier (ATM Forum)
B-HLI	Broadband High-Layer Information
B-ICI	Broadband Inter Carrier Interface
B-ICI SAAL	B-ICI signalling ATM Adaptation Layer

B-ISDN	Broadband Integrated Services Digital Network
B-ISUP	Broadband ISDN User¢s Part
B-ICI	Broadband Intercarrier Interface
B-ISSI	Broadband Inter-Switching System Interface
B-LLI	Broadband Low Layer Information
B-NT	Broadband Network Termination
B-TE	Broadband Terminal Equipment
BBC	Braodband Bearer Capability
BCBDS	Broadband Connectionless Data Bearer Service
BCD	Binary Coded Decimal
BCOB	Broadband Class of Bearer
BECN	Backward Explicit Congestion Notification
BER	Basic Encoding Rules (ASN-1)
BER	Bit Error Rate (link quality specification/testing)
BGP	Border Gateway Protocol
BGT	Broadcast and Group Translators
Bi	Signalling ID assigned by Exchange B
BIP	Bit Interleaved Parity (for example, SONET BIP-8 for path error monitoring)
BIPV	Bit Interleaved Parity Violation
BIS	Border Intermediate System (ATM Forum, PNNI SWG)
BISDN	Broadband -Integrated Services Digital Network
BISSI	Broadband Inter Switching System Interface
BN	Bridge Number
BOM	Beginning of Message
BOOTP	Bootstrap Protocol
BPDU	Bridge Protocol Data Unit
BPP	Bridge Port Pair
BPS	Bits per second
BSS	Broadband Switching System
BSVC	Broadcast Switched Virtual Connections
BT	Burst Tolerance
BT	Begin Tag
BUS	Broadcast and Unknown Server
BW	Bandwidth
CA	Cell Arrival
CAC	Connection Admission Control
CACCAC	Connection Admission Control
CBDS	Connectionless Broadband Data Service
CBR	Constant Bit Rate
CBR interactive	Constant Bit Rate interactive

CBR Non-interactive	Constant Bit Rate non-interactive
CC	Continuity Cell
CCITT	Consultative Committee on International Telephone and Telegraph
CCR	Current Cell Rate
CCS	Common Channel Signalling
CCSS7	Common Channel Signalling System 7
CDT	Cell Delay Tolerance
CDV	Cell Delay Variation
CDVT	Cell Delay Variation Tolerance
CEI	Connection Endpoint Identifier
CER	Cell Error Ratio
CES	Circuit Emulation Service
CI	Congestion Indicator
CIDR	Classless Inter-Domain Routing
CIP	Carrier Identification Parameter
CIR	Committed Information Rate
CL	Connectionless
CLNAP	Connectionless Network Access Protocol
CLNP	Connectionless Network Protocol
CLNS	Connectionless Network Service
CLP	Cell Loss Priority
CLR	Cell Loss Ratio
CLS	Connectionless Server
CLSF	Connectionless Service Function
CME	Component Management Entity
CMI	Coded Mark Inversion
CMISE	Common Management Information Service Element
CMIP	Common Management Interface Protocol
CMR	Cell Misinsertion Rate
CN	Copy Network
CNM	Customer Network Management
CNR	Complex Node Representation (ATM Forum, PNNI SWG)
CO	Connection Oriented
COD	Connection Oriented Data
COM	Continuation of Message
COS	Class of Service
CP	Connection Processor
CPCS	Common Part Convergence Sublayer
CPE	Customer Premise Equipment
CPG	Call Progress Message
CPN	Customer Premises Network
CPN	Calling Party Number
CPI	Common Part Indicator
CRC	Cyclic Redundance Check
CRCG	Common Routing Connection Group
CRF(VC)	Virtual Channel Connection Related

	Function (related to UPC/UNI 3.0)
CRF(VP)	Virtual Path Connection Related Function (related to UPC/UNI 3.0)
Crankback IE	Crankback -Information Element
CRS	Cell Relay Service
CS	Convergence Sublayer (as in CS_PDU)
CS	Carrier Selection
CS1	Capability Set One
CS 2	Capability Set Two
CSI	Convergence Sublayer Indication
CSPDN	Circuit Switched Public Data Network
CSR	Cell Missequenced Ratio
CSU	Channel Service Unit
CTD	Cell Transfer Delay
CTV	Cell Tolerance Variation
DA	Destination MAC address
DA	Destination Address
DCC	Data Country Code
DCE	Data Communication Equipment
DD	Depacketization Delay
DLC	Data Link Control
DES	Destination End System
DLCI	Data Link Connection Identifier
DLL	Data Link Layer, layer2 of the ISO model
DMDD	Distributed Multiplexing Distributed Demultiplexing
DN	Distribution Network
DQDB	Distributed Queue Dual Bus
DS	Distributed Single Layer Test Method
DS-0	Digital Signal, Level O
DS-1	Digital Signal, Level 1
DS-2	Digital Signal, Level 2
DS-3	Digital Signal, Level 3
DS3 PLCP	Physical Layer Convergence Protocol
DSE	Distributed Single Layer Embedded Test Method
DSID	Destination Signalling Identifier
DSS2	Setup Digital Subscriber Signalling #2
DSU	Data Service Unit
DSX	Digital Signal Cross-Connect
DTE	Data Terminal Equipment
DTL IE	DTL -Information Element
DXI	Data Exchange Interface
EDFG	Edge Device Functional Group (ATM Forum, MPOA SWG)
EFCI	Explicit Forward Congestion Indication
ELAN	Emulated LAN (ATM Forum LANE)
EMI	Electromagnetic Interference

EOM	End of Message
EPRCA	Proportional Rate Control Algorithm (ATM Forum)
ESI	End Station Identifier
ETAG	End Tag
ETE	End-to-End
EXM	Exit Message
FC	Feedback Control
FCS	Fast Circuit Switching
FCS	Frame Check Sequence
FCVC	Flow Controlled Virtual Circuit
FDDI	Fiber Distributed Data Interface
FEA	Functional Entity Action (UNI 3.0, C.3.2.3)
FEBE	Far End Block Error (SONET)
FEC	Forward Error Correction
FECN	Forward Explicit Congestion Notification
FERF	Far End Receive Failure
FG	Functional Group (ATM Forum, MPOA SWG)
FUNI	Frame Based Use-to-Network Interface (ATM Forum)
FRS	Frame Relay Service
FUNI	Frame User Network Interface
GAP	Generic Address Parameter
GCID	Global Call Identifier
GCID IE	Global Call Identifier-Information Element
GCRA	Generic Cell Rate Algorithm
GFC	Generic Flow Control
GRC	Generic Reference Configuration
HBFG	Host Behavior Functional Group (ATM Forum, MPOA SWG)
HDB3	High Density Bipolar 3
HDLC	High Level Data Link Control
HEC	Header Error Control
HEC	Header Error Check
HEL	Header Extension Length
HLPI	Higher Layer Protocol Identifier
HOL	Head of Line
IAA	Initial Address Acknowledgment
IAM	Initial Address Message
IAR	Initial Address Reject
IASG	Internetwork Address Sub-Group (ATM Forum, MPOA SWG)
IBM	International Business Machines
IBSG	Internetwork Broadcast Sub-Group (ATM Forum, MPOA SWG)
IBUFG	Internetwork Broadcast/Unknown Functional-Group (ATM Forum, MPOA SWG)
IC	Initial Cell Rate

ICD	International Code Designator
ICFG	IASG Coordination Function Group (ATM Forum, MPOA SWG)
ICMP	Internet Control Message Protocol
IDU	Interface Data Unit
IE	Information Element
IEC	Interexchange Carrier
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IISP	Interim Inter-Switch Protocol akd. P-NNI Phase 0
ILMI	Interim Link Management Interface
ILMI	Interim Local Management Interface
IOP	Interoperability
IP	Internet Protocol
Ipng	Internet Protocol Next Generation
IPX	Novell Internetwork Packet Exchange
ISO	International Organization for Standardization
ITSO	International Technical Support Organization
ITU	International Telecommunications Union
IUT	Implementation Under Test
IWF	Interworking Function
IWU	Interworking Unit
JPEG	Joint Photographic Experts Group
LAN	Local Area Network
LANE	Local Area Network Emulation (ATM Forum)
LAPD	Link Access Procedure D
LB	Leaky Bucket
LCD	Loss of Cell Delineation
LCT	Last Compliance Time (used in GCRA definition)
LD	LAN Destination
LE	LAN Emulation (also, LANE)
LE ARP	LAN Emulation Address Resolution Protocol
LEC	LAN Emulation Client
LEC	Local Exchange Carrier
LECID	LAN Emulation Client Identifier
LECS	LAN Emulation Configuration Server
LES	LAN Emulation Server
LGN	Logical Group Node (ATM Forum, PNNI)
LIJP	Leaf Initiated Join Parameter
LIS	Logical IP Subnet (rfc 1577)
LIV	Link Integrity Verification
LLATMI	Lower Layer ATM Interface
LLC	Logical Link Control
LLC/SNAP	Logical Link Control/Subnetwork Access Protocol

LMI	Layer Management Interface
LOC	Loss of Cell Delineation
LOF	Loss of Frame (UNI Fault Management)
LOP	Loss of Pointer (UNI Fault Management)
LOS	Loss of Signal (UNI Fault Management)
LSB	Least Significant Bit
LSR	Leaf Setup Request
LTE	Line Terminating Equipment (SONET)
LTLT	Lower Tester
LTHLTH	Length Field
LUNI	LANE UNI (ATM Forum, see LANE)
MAMA	Maintenance and Adaptation
MAC	Medium Access Control
MAN	Metropolitan Area Network
MARS	Multicast Address Resolution Service (Draft IETF - IPATM)
MBS	Maximum Burst Size
MCR	Minimum Cell Rate
MCTD	Mean Cell Transfer Delay
ME	Mapping Entity
MIB	Management Information Base
MID	Message Identifier
MIN	Multistage Interconnection Networks
MIR	Maximum Information Rate
MMF	Multimode Fiberoptic Cable
MPEG	Motion Picture Experts Group
MPOA	Multiprotocol over ATM (ATM Forum)
MRCS	Multirate Circuit Switching
MS	Meta Signalling
MSAP	Management Service Access Point
MSB	Most Significant Bit
MSN	Monitoring Cell Sequence Number
MSVC	Meta-signalling Virtual Channel
MT	Message Type
MTP	Message Transfer Part
MTU	Message Transfer Unit
N-ISDN	Narrowband Integrated Services Digital Network
NBMA	Non-Broadcast Multiple Access
NDIS	Network Driver Interface Specification
NE	Network Element
NETBIOS	Network Basic Input/Output System
NEXT	Near End Crosstalk
NHRP	Next Hop Routing Protocol (from IETF ROLC WG)
NHS	Next Hop Server
NIU	Network Interface Unit
NLPID	Network Layer Protocol Identifier
NMS	Network Management System

NNI	Network-to-Network Interface
NP	Network Performance
NPC	Network Parameter Control
NRM	Network Resource Management
NSAP	Network Service Access Point
NSAPA	Network Service Access Point Address
NSP	Network Service Provider
NSR	Nonsource Routed
NT	Network Termination
OAM	Operations, Administration, and Maintenance
OCD	Out-of-Cell Delineation (UNI 3.0 section 2.1.2.2.2)
ODI	Open Data-Link Interface
OLI	Originating Line Information
OOF	Out of Frame
OPCR	Original Program Clock Reference
OSI	Open systems Interconnection
OSID	Origination Signalling Identifier
OSPF	Open Shortest Path First
OUI	Organization Unique Identifier
OUI	Organizational Unit Identifier
P-NNI	Private Network to Network Interface
PAD	Packet Assembler and Disassembler
PBX	Private Branch eXchange
PC	Priority Control
PC	Protocol Control
PCM	Pulse Code Modulation
PCO	Point of Control and Observation
PCR	Peak Cell Rate (UNI 3.0)
PCR	Program Clock Reference
PCVS	Point to Point Switched Virtual Connections
PD	Packetization Delay
PDH	Plesiochronous Digital Hierarchy
PDU	Packet Data Unit
PDU	Protocol Data Unit
PGL	Peer Group Leader (ATM Forum, PNNI)
PHY	Physical Layer of the OSI Model
PHY	Physical Layer
PICS	Protocol Implementation Conformance Statement
PID	Protocol Identifier Governing Connection Types
PIXIT	Protocol Implementation eXtra Information for Testing
PL-OU	Physical Layer Overhead Unit (UNI physical layer frame definition)
PL	Physical Layer
PLL	Phase Locked Loop

PLCP	Physical Layer Convergence Protocol
PM	Physical Medium
PMD	Physical Layer Dependent Sublayer
PMP	Point-to-Mulitpoint (UNI 3.0)
PNNI	Private Network Node Interface (ATM Forum, PNNI SWG)
PNNI	Private Network-to-Network Interface (ATM Forum, PNNI SWG)
POH	Path Overhead
POI	Path Overhead Indicator
PT	Payload Type
PTE	Path Terminating Equipment SONET)
PTI	Payload Type Identifier
PTSE	PNNI Topology State Element (ATM Forum, PNNI)
PTSP	PNNI Topology State Packet (ATM Forum, PNNI)
PVC	Permanent Virtual Circuit
PVCC	Permanent Virtual Channel Connection
PVPC	Permanent Virtual Path Connection
QD	Queuing Delay
QOS	Quality of Service
QPSX	Queue Packet and Synchronous Circuit Exchange
RAI	Remote Alarm Indication
RBOC	Regional Bell Operating Company
RC	Routing Control
RD	Route Descriptor
RDF	Rate Decrease Factor
RD	Remote Defect Identification (UNI Fault Management)
RDI	Remote Defect Indication
REL	Release Message
RFC	Request For Comment (Document Series)
RFI	Radio Frequency Interference
R	Routing Information
RII	Routing Information Indicator
RIP	Routing Information Protocol
RISC	Reduced Instruction Set Computing
RLC	Release Complete
RM	Resource Management
	ROLC Routing Over Large Clouds
RSFG	Route Server Functional Group (ATM Forum, MPOA SWG)
RSVP (protocol)	Resource Reservation Protocol
RT	Routing Type
RTS	Residual Time Stamp
SA	Source MAC address
SA	Source Address
SAAL	Signalling ATM Adaptation Layer
SAP	Service Access Point
SAR	Segmentation and Reassembly
SCCP	Signalling Connection and Control Part
SCP	Service Control Point
SCR	Sustainable Cell Rate (UNI 3.0)
SDH	Synchronous Digital Hierarchy
SDU	Service Data Unit (as in AAL SDU)
SE	Switching Element

SEAL	Simple and Efficient Adaptation Layer
SECB	Severely Errored Cell Block
SF	Switching Fabric
SGM	Segmentation Message
SID	Signalling Identifier
SIPP	SMDS Interface Protocol
SIR	Sustained Information Rate
SMC	Sleep Mode Connection
SMDS	Switched Multimegabit Data Services
SMF	Single Mode Fiber
SN	Sequence Number
SNA	Systems Network Architecture
SNAP	Sub Network Access Protocol
SNDCF	Subnetwork Dependent Convergence Function (ATM Forum, MPOA SWG)
SNI	Subscriber Network Interface
SNMP	Simple Network Management Protocol
SOH	Section Overhead
SONET	Synchronous Optical Network
SPID	Service Protocol Identifier
SPTS	Single Program Transport Stream
SR	Source Routing (Bridging)
SRF	Specifically Routed Frame
SRT	Source Routing Transparent
SRTS	Synchronous Residual Time Stamp
SSCF	Service Specific Coordination Function
SSCOP	Service Specific Connection Oriented Protocol
SSCS	Service Specific Convergence Sublayer
ST	Segment Type
STE	Spanning Tree Explorer
STM	Synchronous Transfer Mode
STM1	Synchronous Transport Mode 1 -- 155 mbits/sec
STP	Signalling Transfer Point
STP	Shielded Twisted Pair Cable
STS	Synchronous Time Stamps
STS-3c	Synchronous Transport System-Level 3 Concatenated
SUT	System Under Test
SVC	Switched Virtual Circuit
SVCI	Switched Virtual Circuit Identifier
SVP	Switched Virtual Path
SWG	Subworking Group
T1S1	ANSI T1 Subcommittee
TAT	Theoretical Arrival Time (used in GCRA definition)

TAXI	Transparent Asynchronous Transmitter/receiver Interface
TB	Transparent Bridging
TC	Transaction Capabilities
TC	Transmission Convergence
TCAP	Transaction Capabilities Applications Part
TCI	Test Cell Input TCO Test Cell Output
TCP	Transmission Control Protocol
TCP	Test Coordination Procedure
TCP/IP	Transmission Control Program/Internet Protocol
TCS	Transmission Convergence Sublayer
TDJ	Transfer Delay Jitter
TDM	Time Division Multiplexing
TE	Terminal Equipment
TIG	Topology Information Group (ATM Forum, PNNI)
TLV	Type/ Length/ Value
TM	Traffic Management
TM	SWG Traffic Management Subworking Group
TMP	Test Management Protocol
TNS	Transit Network Selection
TPCC	Third Party Call Control
TS	Traffic Shaping
TS	Time Stamp
TS	Transport Stream
TS	Time Slot
TSAP	Transport Service Access Point
TUC	Total User Cell count
TUCD	Total User Cell Difference
UBR	Unspecified Bit Rate
UDP	User Datagram Protocol
UME	UNI Management Entity (used in ILMI definition)
UNI	User Network Interface
UPC	Usage Parameter Control
UT	Upper Tester
UTOPIA	Universal Test and Operations PHY Interface for ATM
UTP	Unshielded Twisted Pair Cable
VBR	Variable Bit Rate
VBR Delay Sensitive	Variable Bit Rate Delay Sensitive
VBR Delay Tolerant	Variable Bit Rate Delay Tolerant
VBR Non-interactive	Variable Bit Rate Non-interactive
VC	Virtual Channel (Virtual Circuit)
VC-Multiplexing	Virtual Channel -Multiplexing
VCC	Virtual Channel Connections
VCI	Virtual Connection Identifier

VCL	Virtual Channel Link (UNI 3.0)
VINCE	Vendor Independent Network Control Entity
VLAN	Virtual Local Area Network
VP	Virtual Path
VP/VC	Virtual Path, Virtual Circuit
VPC	Virtual Path Connection
VPCI/VCI	Virtual Path Connection Identifier/Virtual Channel Identifier
VPI	Virtual Path Identifier
VPL	Virtual Path Link (UNI 3)
VPT	Virtual Path Terminator (UNI 3)
VS	Virtual Scheduling
WAN	Wide Area Network
XNS	Xerox Network Systems
XTP	eXpress Transport Protocol

Index

IBM Redbooks review

Your feedback is valued by the Redbook authors. In particular we are interested in situations where a Redbook "made the difference" in a task or problem you encountered. Using one of the following methods, **please review the Redbook, addressing value, subject matter, structure, depth and quality as appropriate.**

- Use the online **Contact us** review redbook form found at `http://www.redbooks.ibm.com/`
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to `redbook@us.ibm.com`

Document Number **Redbook Title**	SG24-5525-00 RS/6000 ATM Cookbook
Review	
What other subjects would you like to see IBM Redbooks address?	
Please rate your overall satisfaction:	O Very Good O Good O Average O Poor
Please identify yourself as belonging to one of the following groups:	O Customer O Business Partner O Solution Developer O IBM, Lotus or Tivoli Employee O None of the above
Your email address: The data you provide here may be used to provide you with information from IBM or our business partners about our products, services or activities.	 O Please do not use the information collected here for future marketing or promotional contacts or other communications beyond the scope of this transaction.
Questions about IBM's privacy policy?	The following link explains how we protect your personal information. `http://www.ibm.com/privacy/yourprivacy/`

SG24-5525-00
Printed in the U.S.A.

RS/6000 ATM Cookbook

SG24-5525-00