

## *Classical IP and LAN Emulation Protocols*

---

## 6

ATM is a connection-oriented network protocol, which means that a connection must be established between two communicating entities before data transfer can begin. IP is inherently connectionless. The implementation on the host must therefore reconcile the differences in these two paradigms.

There are two standardized, commonly used ways of doing this: Classical IP, standardized in RFC 1577, and LAN Emulation standardized in the LAN Emulation 1.0 specification from the ATM Forum. The SunATM architecture supports both of these methods. Some of the key ideas of these two methods are discussed in later sections of this chapter.

Both methods allow IP to run transparently over the ATM interface. Thus IP itself sees the ATM interface just as it sees any traditional network interface. Every SunATM interface has a subnet IP address. During the process of startup of an ATM interface, appropriate modules and drivers are plumbed. All the TCP/IP and UDP/IP applications run without modifications over these modules, and all the utilities associated with the network interfaces also run without modification and display similar results (for example, `netstat`, `ifconfig` utilities, etc.), with one exception. Because of the different plumbing of the ATM modules, the `plumb` and `unplumb` options of `ifconfig` will not work on ATM interfaces; the `atmplumb(1M)` command must be used instead. IP treats the ATM interface as a subnet, choosing the interface used to send a packet out based on the IP address of the destination and on the IP address and netmask of the interface itself.

The transparency to IP is enabled in different ways in Classical IP and LAN Emulation. Those differences will be discussed in later sections of this chapter.

SunATM signalling conforms to the UNI specification of the ATM Forum. Both versions 3.0 and 3.1 of that specification are supported. This signalling, called Q.2931, runs on top of QSAAL and uses VC 5 for signalling as specified in the Forum specification.

## 6.1 ATM Addresses and Address Registration

UNI signalling uses ATM addresses for signalling. Every ATM interface will have an ATM address in addition to its IP address.

ATM addresses, like NSAP addresses, are 20 octets long. The End System Identifier (ESI) field within the ATM address is a unique 6 octet value; this can be the IEEE hardware MAC address conventionally associated with every network interface. The Selector field is one octet long. The 13 octets that make up the rest of the ATM address are called the Network Prefix, and should be derived from the ATM switch fabric to which the interface is connected. Every ATM switch fabric is configured with a 13 octet prefix.

On a SunATM host, the prefix associated with the local switch fabric is represented by the variable prefix. Its value will be obtained by the system at configuration time.

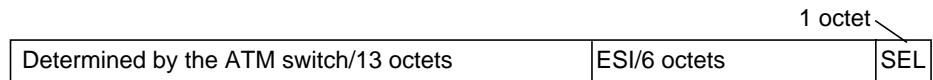


Figure 6-1 ATM Address Fields

The UNI specification specifies the Interim Local Management Interface (ILMI) service interface for a client to learn and register its ATM address. The ILMI service interface is based on the use of SNMP over AAL5. In the SunATM software package, ILMI service is provided by an address registration daemon, `ilmid`.

### 6.1.1 ATM Address Registration Daemon (ILMID)

Address registration with a switch is controlled by `ilmid`. When an ATM interface is brought up at boot time, `ilmid` is also started. `ilmid` then begins an exchange of messages with the switch: relaying local address information (the 7 octet ESI and selector) to the switch, and receiving the 13 octet network prefix information from the switch.

The default local address that is registered with the switch at boot time consists of the network prefix provided by the switch, the MAC address assigned to the local interface, and the default selector for that interface (usually 0). Additional addresses may be registered in two different ways. `aarsetup(1M)` and `lanesetup(1M)` register additional local addresses that may appear in `aarconfig(4)` and `laneconfig(4)`, respectively. There is also a user program, `atmreg(1M)`, that may be used to register addresses, to de-register addresses, and to check the status of any address.

## 6.2 *Classical Internet Protocol*

The major task required for ATM to work transparently under IP is resolving an IP address to an ATM address and establishing the connection to that destination. Classical IP does this via a database of IP/ATM address pairs that is either provided by an ATM ARP server which is accessible to all hosts on the subnet, or maintained locally in each host.

### 6.2.1 *ATM Address Resolution*

Traditional TCP/IP and UDP/IP applications use IP addresses for communicating to a destination. In order for these applications to run as before, there is a need to resolve these IP addresses into ATM addresses. The ATM address is then used in signalling to establish an ATM connection to the destination. An ATM connection in turn is represented by a VPI/VCI. The host must use this returned VPI/VCI to send packets to the destination representing the ATM connection.

ATM address resolution, also called ATM ARP, follows RFC 1577, the classic draft that describes the ATM ARP process.

RFC 1577 is based upon the existence of an ATM ARP server on every subnet. Every client of the subnet communicates with the ATM ARP server to derive an ATM address of the destination from the IP address of the destination. The ATM ARP server holds the IP to ATM address information for all hosts in the ATM subnet. It is likely that initial ATM configurations will not rely on dynamic ATM address resolution since it requires the presence of an ATM ARP server on every subnet. Also, there are no specified standards for providing redundant ATM ARP servers for a subnet. As specified, the ATM ARP server would constitute a single point of failure in the system. From a practical standpoint, however, early configurations may take the course of having the IP

to ATM address database in every system, thus avoiding the IP to ATM address resolution step altogether. The RFC requires the use of a router to pass data between subnets.

SunATM software facilitates this by providing ATM utilities that will allow configurations to specify IP to ATM addresses in `/etc/aarconfig` files. The `aarsetup` program uses the information in `/etc/aarconfig` to create IP to ATM address resolution tables. Dynamic entries into a server's resolution table are also supported.

Table 5-1 on page 5-2 shows the format of the `/etc/aarconfig` file for specifying the IP to ATM address. It is important for the file to be consistent on all systems in the subnet. See Section 5.2.1, "Editing the `/etc/aarconfig` File," on page 5-4.

### ***6.2.2 ATM ARP Address Resolution Tables***

Depending on the `aarconfig` file, the Classical IP software will run as either a server or a client. As a server, the Classical IP software is responsible for handling ATM ARP requests originating from its clients. An ATM server has to be configured for each subnet. The ATM ARP server code conforms to RFC 1577: clients send ATM ARP requests to the server to resolve a destination IP address to an ATM address. The server then replies to ATM ARP requests by sending an ATM ARP response. If the server does not have the IP to ATM address entry, then it replies with NAK.

The file `/etc/aarconfig` is also used by the ATM ARP server. All the IP to ATM address entries specified in the file will be entered into a kernel resident table by the ATM ARP setup program, `aarsetup`. Additional entries in the kernel table will be added dynamically using the inverse ARP process. When a client connects to the server, the server will send an inverse ARP request back to the client to obtain the client's IP address. When a response is received, an entry will be created for that client. The Classical IP software will also respond to client ARP requests. The software looks up a kernel IP to ATM address entry and responds to an ATM ARP request with either an ATM ARP reply or ATM ARP NAK (if there is no entry in the table). Note that an ATM ARP client uses the virtual channel (VC) specified in the `/etc/aarconfig` file to communicate with the server; or, if an ATM address is specified, it establishes an switched virtual circuit (SVC) connection to communicate with the server.

---

While dynamic entries in the ARP server's table make network administration less complex, it also creates a security problem. Any host may register with the ARP server and, therefore, gains access to the subnet. To resolve this issue, a list of hosts or networks may optionally be provided with *a* entries in the server's `/etc/aarconfig` file. If no *a* entries appear, any host will be allowed to connect to the server. If any *a* entries exist, only those hosts whose addresses match those specified will be allowed to connect.

Although the *a* entry requires a complete ATM address, multiple addresses can be referenced in a single entry using the provided wildcards. See Section 5.2.2, "Using Variables in the `/etc/aarconfig` File," on page 5-8, for more information about this feature.

The advantage to having an ATM ARP server in the subnet is that it represents a known source for all address resolutions. It is the only host which a client must know about to have IP addresses resolved to ATM connections, and it allows for access control in the ATM network.

When the `/etc/aarconfig` file has been modified on a system, it is necessary to rerun `aarsetup`.

---

**Note** – For better caching, all clients have the option of adding to their configuration file the IP to ATM address information for other clients. This can benefit clients who communicate frequently because it eliminates having to go through the ATM ARP server for IP to ATM address resolution.

---

If a host has multiple SunATM cards, the host may be a server for one IP subnet and a client for another. This is handled transparently by `aarsetup`.

## 6.3 LAN Emulation

As described in previous sections, Classical IP provides its own (IP to ATM) address resolution mechanism which corresponds to and replaces ARP, thus allowing IP-based applications to run transparently over ATM. A shortcoming of Classical IP, and a primary reason it must replace the traditional ARP, is that it does not support broadcast messages.

Because ATM is a connection-oriented protocol (unlike ethernet), implementing broadcast is much more difficult. The only host that receives a message is the host to which the message is addressed, and a call must be established to that host before the message can be sent.

Local Area Network (LAN) Emulation, as standardized by the ATM Forum, provides mechanisms to send broadcast messages in an ATM environment. Given this capability, LAN Emulation is also able to work transparently with ARP, as well as IP. IP and ARP may send broadcast messages over the ATM interface, and thus resolve IP addresses to MAC addresses; messages are then sent to the LAN Emulation driver, which has its own address resolution protocol (similar to that of Classical IP) to resolve the medium access control (MAC) address to an ATM address and connection.

The SunATM 2.1 software implements the client side of the LAN Emulation standard. In order to use LAN Emulation in an environment, several LAN Emulation services must also exist in the emulated LAN. These services, called the LAN Emulation Configuration Server (LECS), the LAN Emulation Server (LES), and the Broadcast and Unknown Address Server (BUS), are generally provided in an ATM switch. An overview of the functions of these servers is provided in the following sections.

### **6.3.1 LAN Emulation Services**

#### **6.3.1.1 LAN Emulation Configuration Server**

This server is contacted first by a host interface when the host is brought up on the emulated LAN. Its address is generally a well-known address specified by the LAN Emulation standard which is coded into the host software; thus no input from the user is required to establish this connection. When contacted by a host wishing to join its emulated LAN, the LECS replies with configuration parameters for the emulated LAN, as well as the address of the LES.

#### **6.3.1.2 LAN Emulation Server**

The second step in joining an emulated LAN is to make a connection to the LAN Emulation Server. After receiving the LES address from the LECS, a host will establish a connection to the LES. The LES may add the host to a point-to-multipoint call which is maintained by the LES with connections to every host in the emulated LAN. This point-to-multipoint connection, if created by the LES, is used to send control information to each host on the emulated LAN.

---

The LES acts as the ATM ARP server. Since IP and ARP work with MAC addresses, an additional address resolution step is required to convert a MAC address to the corresponding ATM address, which is used to make a connection to the target host; this resolution step is provided by the LES.

### ***6.3.1.3 Broadcast and Unknown Address Server***

The final step in joining an emulated LAN is to make a connection to the BUS. The ATM address of the BUS is obtained by sending a LAN Emulation ARP request to the LES for the broadcast address. Once established, this connection is used to send broadcast messages to the BUS, which will add the client to a point-to-multipoint call including all hosts on the emulated LAN. Thus when a broadcast message (such as an IP ARP request) is received by the LAN Emulation host from its upper layers, it sends that message to the BUS, which forwards it to all hosts in the emulated LAN. Just as in the case of ethernet, the correct host responds to the sender, and thus the IP address is resolved to a MAC address.

### ***6.3.2 Resolving an IP Address to an ATM Connection***

The entire process from the time IP sends a message addressed to an IP address to the arrival of that message at the appropriate destination was hinted at in the above descriptions of the LAN Emulation servers. To demonstrate how those pieces work together during the actual transmission of a message, the process is described below, assuming that none of the needed addresses have been previously resolved and cached. The two hosts involved are referred to as the source (the system who wishes to send a message) and the target (the system to which the message is addressed).

1. IP has a message to transmit, and only knows the IP address of the target system. It first sends a message to ARP, to resolve the IP address to a MAC address.
2. ARP creates a broadcast request for the MAC address corresponding to the given IP address, which it sends to the LAN Emulation driver.
3. The LAN Emulation driver recognizes that this message has a broadcast address, and sends it to the BUS, which forwards the message to every host on the emulated LAN.

4. The message is received on each host, and sent up to ARP by the LAN Emulation driver.
5. On the target, ARP recognizes the IP address as its own and sends a response with its MAC address (addressed to the source's MAC address) down to the LAN Emulation driver.
6. The LAN Emulation driver sends an LE ARP request to the LES to resolve the source's MAC address to its ATM address.
7. The LES responds with the requested ATM address, and the target host sets up an ATM connection to the source host, over which it sends the IP ARP response.
8. The LAN Emulation driver on the source receives the IP ARP response message and sends it up to ARP. ARP then inserts the MAC address into the original message and sends it back down to the LAN Emulation driver.
9. The LAN Emulation driver then must send an LE ARP request to the LES to resolve the MAC address in the message from ARP to an ATM address. When it receives an LE ARP response, it then sees that it has a connection to that address (established by the target to return the IP ARP response), and sends the original IP message to the target over that connection.

### 6.3.3 LAN Emulation Connections

As should be somewhat obvious from the preceding discussion, there will be several connections established at all times when a host is a member of an emulated LAN. The following table outlines the various LAN Emulation-related connections that should be expected on a LAN Emulation client (LEC).

---

**Note** – The command `qccstat(1M)` may be used to view all existing connections for a given interface.

---

*Table 6-1* LAN Emulation Connections

---

<b>Connection</b>	<b>Comments</b>
LEC → LECS	This connection is not required to remain open after the initial join of the emulated LAN, and thus may time out after a host has joined the LAN.
LEC → LES	Point-to-point connection over which the host may send LE ARP requests and receive responses from the LES.
LES → LEC	Point-to-multipoint connection over which the LES may send administrative information to all hosts. Hosts may not send on this connection.
LEC → BUS	Point-to-point connection over which the host may send broadcast messages to the BUS. A limited amount of data is also allowed on this connection.
BUS → LEC	Point-to-multipoint connection over which the BUS sends broadcast messages. Hosts may not send on this connection.

---

## 6.4 ATM and SNMP

Two of the ATM standards supported by the SunATM software, the User Network Interface (UNI) and LAN Emulation (LANE) specifications, include definitions of SNMP-style Management Information Bases (MIBs) relevant to those standards. These MIBs are referred to as the ATM Forum (ATMF) and LAN Emulation (LANE) MIBs, respectively.

The ATM SNMP daemon (`atmsnmpd`) handles requests for information in both MIBs, as well as the system MIBs, from SNMP-based network management systems (such as the SunNet Manager program), and from `ilmid`, when it is required, for SNMP requests coming from the switch.

If you configured your system (using the `atmadmin` configuration program) to start `atmsnmpd` as the default SNMP agent in the system then `atmsnmpd` will listen to UDP port 161 for SNMP traffic. Otherwise, `atmsnmpd` will be started with the `-n` option, meaning that `atmsnmpd` will not listen on any UDP port for SNMP traffic, but it can still be used by `ilmid`.

There is one caveat associated with the use of `atmsnmpd` running as an agent on a system. Since the SNMP protocol is defined to use a single UDP port number, only one SNMP agent, such as `atmsnmpd`, may run on a system at a time. Most SNMP agent daemons, including `atmsnmpd`, allow an alternate port number to be specified, but this will limit the accessibility of that agent to network managers such as the SunNet Manager program. Depending on your requirements, you may wish to run `atmsnmpd` on an alternate port. The `atmsnmpd` daemon is started in the `/etc/rc2.d/S00sunatm` script; the `-p` option with an alternate port number may be added to this script.

A new feature of the SunATM 2.1 software is the use of `atmsnmpd` as a forwarding agent. If configured as a forwarding agent, `atmsnmpd` will forward SNMP requests for unknown MIBs to the port specified with the `forward` option, `-f`. This allows a system to have two SNMP agents respond to requests received over the SNMP port. Figure 6-2 illustrates the required configuration. In order to set up this example configuration, `atmsnmpd` must be started with the parameter `-f 1000` and `other_snmpd` must be started so that it listens on port 1000.

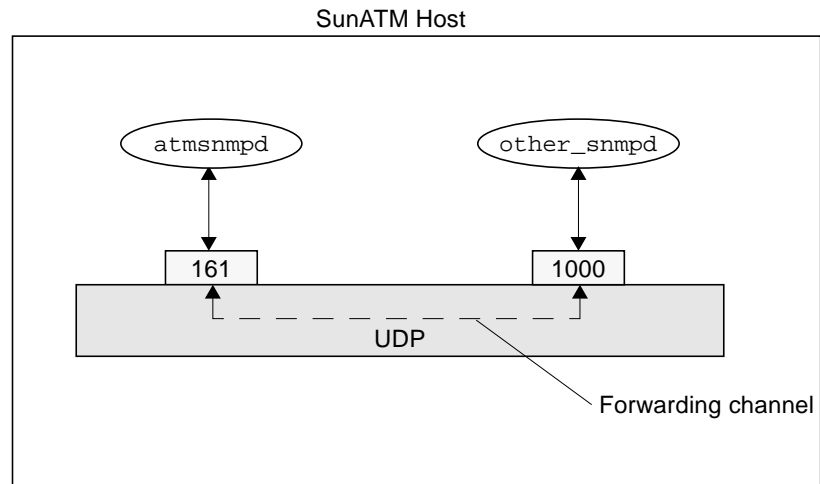


Figure 6-2 Using atmsnmpd as a Forwarding Agent

**Note** – If no port is specified to forward unknown requests, `atmsnmpd` will respond with a “No Such Name” error to requests for MIBs which it does not support. If a forwarding port is specified, `atmsnmpd` will instead forward the request to the specified port. Responses received from the agent running on the forwarding port will be sent to the requesting SNMP manager with no modification. If the agent does not respond, then `atmsnmpd` will not send any response back.

Appendix D, “Managing SunATM Interfaces with SNMP,” provides more information about using `atmsnmpd` to manage and monitor the SunATM interfaces with a network manager such as the SunNet Manager program.

