CHAPTER

8

Address Resolution Protocol (ARP)

Before the IP protocol can deliver a packet from a source host to the destination host, it needs to know how to deliver it to the next hop first. An IP packet can consult its routing table, as discussed in Chapter 6, to find the IP address of the next hop. But since IP uses the services of the data link layer, it needs to know the physical address of the next hop. This can be done using a protocol, called Address Resolution Protocol (ARP), which we discuss in this section.

OBJECTIVES

The chapter has several objectives:

- To make a distinction between logical address (IP address), which is used at the network layer, and physical address (MAC address), which is used at the data link layer.
- To describe how the mapping of a logical address to a physical address can be static or dynamic.
- To show how the address resolution protocol (ARP) is used to dynamically map a logical address to a physical address.
- To show that the proxy ARP can be used to create a subnetting effect.
- To discuss ATMARP, which maps the IP addresses when the underlying network is an ATM WAN.
- To show that an ARP software package can be made of five components: a cache table, queues, an output module, an input module, and a cache-control module.
- To show the pseudocode for each module used in the ARP software package.

8.1 ADDRESS MAPPING

An internet is made of a combination of physical networks connected together by internetworking devices such as routers. A packet starting from a source host may pass through several different physical networks before finally reaching the destination host.

The hosts and routers are recognized at the network level by their logical addresses. A logical address is an internetwork address. Its jurisdiction is universal. A logical address is unique universally. It is called a *logical* address because it is usually implemented in software. Every protocol that deals with interconnecting networks requires logical addresses. The logical addresses in the TCP/IP protocol suite are called **IP addresses** and are 32 bits long.

However, packets pass through physical networks to reach these hosts and routers. At the physical level, the hosts and routers are recognized by their physical addresses. A **physical address** is a local address. Its jurisdiction is a local network. It should be unique locally, but not necessarily universally. It is called a *physical* address because it is usually (but not always) implemented in hardware. Examples of physical addresses are 48-bit MAC addresses in the Ethernet protocol, which are imprinted on the NIC installed in the host or router.

The physical address and the logical address are two different identifiers. We need both of them because a physical network such as Ethernet can have two different protocols at the network layer such as IP and IPX (Novell) at the same time. Likewise, a packet at a network layer such as IP may pass through different physical networks such as Ethernet and LocalTalk (Apple).

This means that delivery of a packet to a host or a router requires two levels of addressing: logical and physical. We need to be able to map a logical address to its corresponding physical address and vice versa. These can be done using either static or dynamic mapping.

Static Mapping

Static mapping means creating a table that associates a logical address with a physical address. This table is stored in each machine on the network. Each machine that knows, for example, the IP address of another machine but not its physical address can look it up in the table. This has some limitations because physical addresses may change in the following ways:

1. A machine could change its NIC, resulting in a new physical address.
2. In some LANs, such as LocalTalk, the physical address changes every time the computer is turned on.
3. A mobile computer can move from one physical network to another, resulting in a change in its physical address.

To implement these changes, a static mapping table must be updated periodically. This overhead could affect network performance.

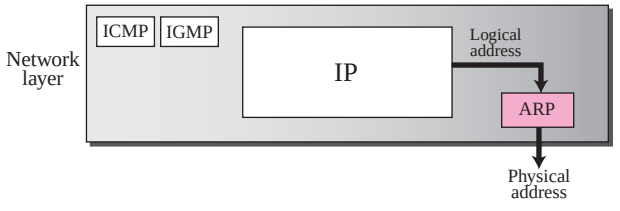
Dynamic Mapping

In **dynamic mapping**, each time a machine knows the logical address of another machine, it can use a protocol to find the physical address. Two protocols have been designed to perform dynamic mapping: **Address Resolution Protocol (ARP)** and **Reverse Address Resolution Protocol (RARP)**. ARP maps a logical address to a physical address; RARP maps a physical address to a logical address. Since RARP is replaced with another protocol and therefore deprecated, we discuss only ARP protocol in this chapter.

8.2 THE ARP PROTOCOL

Anytime a host or a router has an IP datagram to send to another host or router, it has the logical (IP) address of the receiver. But the IP datagram must be encapsulated in a frame to be able to pass through the physical network. This means that the sender needs the physical address of the receiver. A mapping corresponds a logical address to a physical address. Figure 8.1 shows the position of the ARP in the TCP/IP protocol suite. ARP accepts a logical address from the IP protocol, maps the address to the corresponding physical address and pass it to the data link layer.

Figure 8.1 *Position of ARP in TCP/IP protocol suite*



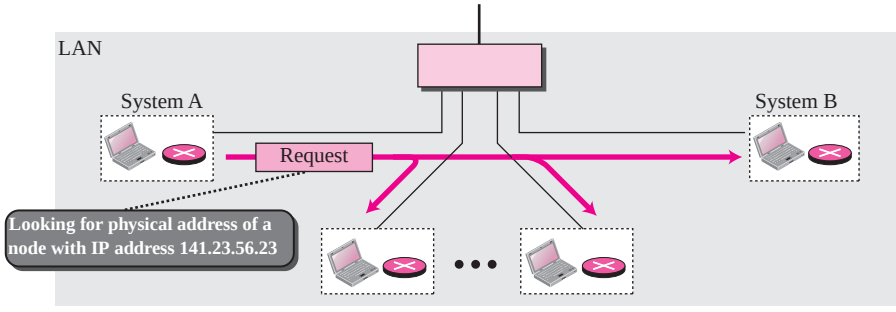
ARP associates an IP address with its physical address. On a typical physical network, such as a LAN, each device on a link is identified by a physical or station address that is usually imprinted on the NIC.

Anytime a host, or a router, needs to find the physical address of another host or router on its network, it sends an ARP query packet. The packet includes the physical and IP addresses of the sender and the IP address of the receiver. Because the sender does not know the physical address of the receiver, the query is broadcast over the network (see Figure 8.2).

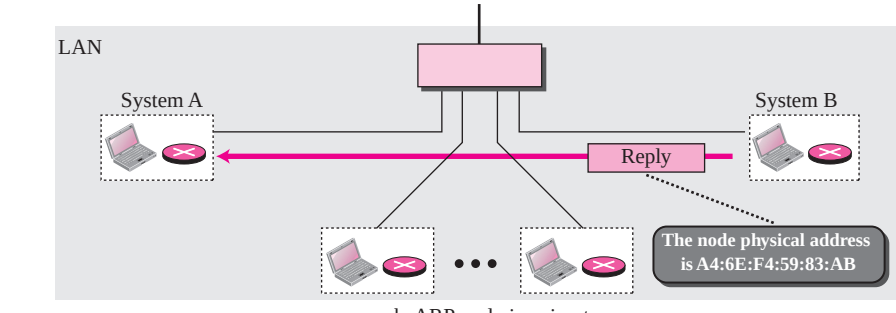
Every host or router on the network receives and processes the ARP query packet, but only the intended recipient recognizes its IP address and sends back an ARP response packet. The response packet contains the recipient's IP and physical addresses. The packet is unicast directly to the inquirer using the physical address received in the query packet.

In Figure 8.2a, the system on the left (A) has a packet that needs to be delivered to another system (B) with IP address 141.23.56.23. System A needs to pass the packet to its data link layer for the actual delivery, but it does not know the physical address of

Figure 8.2 *ARP operation*



a. ARP request is multicast



b. ARP reply is unicast

the recipient. It uses the services of ARP by asking the ARP protocol to send a broadcast ARP request packet to ask for the physical address of a system with an IP address of 141.23.56.23.

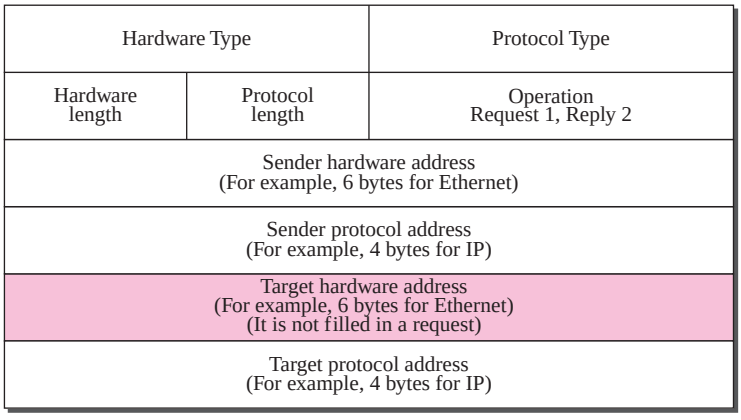
This packet is received by every system on the physical network, but only system B will answer it, as shown in Figure 8.2b. System B sends an ARP reply packet that includes its physical address. Now system A can send all the packets it has for this destination using the physical address it received.

Packet Format

Figure 8.3 shows the format of an ARP packet. The fields are as follows:

- **Hardware type.** This is a 16-bit field defining the type of the network on which ARP is running. Each LAN has been assigned an integer based on its type. For example, Ethernet is given the type 1. ARP can be used on any physical network.
- **Protocol type.** This is a 16-bit field defining the protocol. For example, the value of this field for the IPv4 protocol is 0800_{16}. ARP can be used with any higher-level protocol.
- **Hardware length.** This is an 8-bit field defining the length of the physical address in bytes. For example, for Ethernet the value is 6.
- **Protocol length.** This is an 8-bit field defining the length of the logical address in bytes. For example, for the IPv4 protocol the value is 4.

Figure 8.3 *ARP packet*



- **Operation.** This is a 16-bit field defining the type of packet. Two packet types are defined: ARP request (1), ARP reply (2).
- **Sender hardware address.** This is a variable-length field defining the physical address of the sender. For example, for Ethernet this field is 6 bytes long.
- **Sender protocol address.** This is a variable-length field defining the logical (for example, IP) address of the sender. For the IP protocol, this field is 4 bytes long.
- **Target hardware address.** This is a variable-length field defining the physical address of the target. For example, for Ethernet this field is 6 bytes long. For an ARP request message, this field is all 0s because the sender does not know the physical address of the target.
- **Target protocol address.** This is a variable-length field defining the logical (for example, IP) address of the target. For the IPv4 protocol, this field is 4 bytes long.

Encapsulation

An ARP packet is encapsulated directly into a data link frame. For example, in Figure 8.4 an ARP packet is encapsulated in an Ethernet frame. Note that the type field indicates that the data carried by the frame is an ARP packet.

Operation

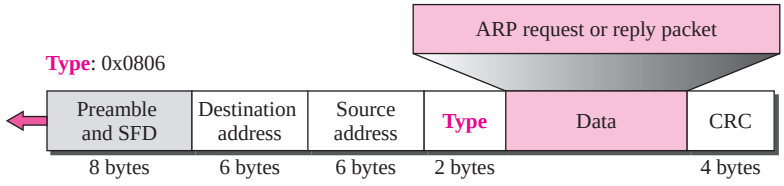
Let us see how ARP functions on a typical internet. First we describe the steps involved. Then we discuss the four cases in which a host or router needs to use ARP.

Steps Involved

These are seven steps involved in an ARP process:

1. The sender knows the IP address of the target. We will see how the sender obtains this shortly.
2. IP asks ARP to create an ARP request message, filling in the sender physical address, the sender IP address, and the target IP address. The target physical address field is filled with 0s.

Figure 8.4 *Encapsulation of ARP packet*



3. The message is passed to the data link layer where it is encapsulated in a frame using the physical address of the sender as the source address and the physical broadcast address as the destination address.
4. Every host or router receives the frame. Because the frame contains a broadcast destination address, all stations remove the message and pass it to ARP. All machines except the one targeted drop the packet. The target machine recognizes the IP address.
5. The target machine replies with an ARP reply message that contains its physical address. The message is unicast.
6. The sender receives the reply message. It now knows the physical address of the target machine.
7. The IP datagram, which carries data for the target machine, is now encapsulated in a frame and is unicast to the destination.

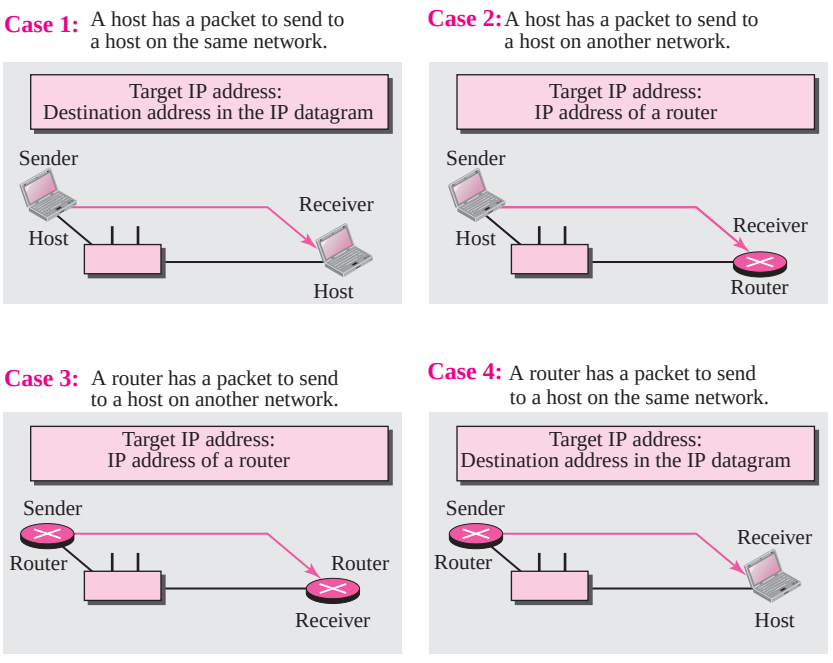
An ARP request is broadcast; an ARP reply is unicast.

Four Different Cases

The following are four different cases in which the services of ARP can be used (see Figure 8.5).

- **Case 1:** The sender is a host and wants to send a packet to another host on the same network. In this case, the logical address that must be mapped to a physical address is the destination IP address in the datagram header.
- **Case 2:** The sender is a host and wants to send a packet to another host on another network. In this case, the host looks at its routing table and finds the IP address of the next hop (router) for this destination. If it does not have a routing table, it looks for the IP address of the default router. The IP address of the router becomes the logical address that must be mapped to a physical address.
- **Case 3:** The sender is a router that has received a datagram destined for a host on another network. It checks its routing table and finds the IP address of the next router. The IP address of the next router becomes the logical address that must be mapped to a physical address.
- **Case 4:** The sender is a router that has received a datagram destined for a host in the same network. The destination IP address of the datagram becomes the logical address that must be mapped to a physical address.

Figure 8.5 *Four cases using ARP*



Example 8.1

A host with IP address 130.23.43.20 and physical address B2:34:55:10:22:10 has a packet to send to another host with IP address 130.23.43.25 and physical address A4:6E:F4:59:83:AB (which is unknown to the first host). The two hosts are on the same Ethernet network. Show the ARP request and reply packets encapsulated in Ethernet frames.

Solution

Figure 8.6 shows the ARP request and reply packets. Note that the ARP data field in this case is 28 bytes, and that the individual addresses do not fit in the 4-byte boundary. That is why we do not show the regular 4-byte boundaries for these addresses. Also note that the IP addresses are shown in hexadecimal. For information on binary or hexadecimal notation see Appendix B.

Proxy ARP

A technique called *proxy* ARP is used to create a subnetting effect. A **proxy ARP** is an ARP that acts on behalf of a set of hosts. Whenever a router running a proxy ARP receives an ARP request looking for the IP address of one of these hosts, the router sends an ARP reply announcing its own hardware (physical) address. After the router receives the actual IP packet, it sends the packet to the appropriate host or router.

Let us give an example. In Figure 8.7 the ARP installed on the right-hand host will answer only to an ARP request with a target IP address of 141.23.56.23.

Figure 8.6 *Example 8.1*

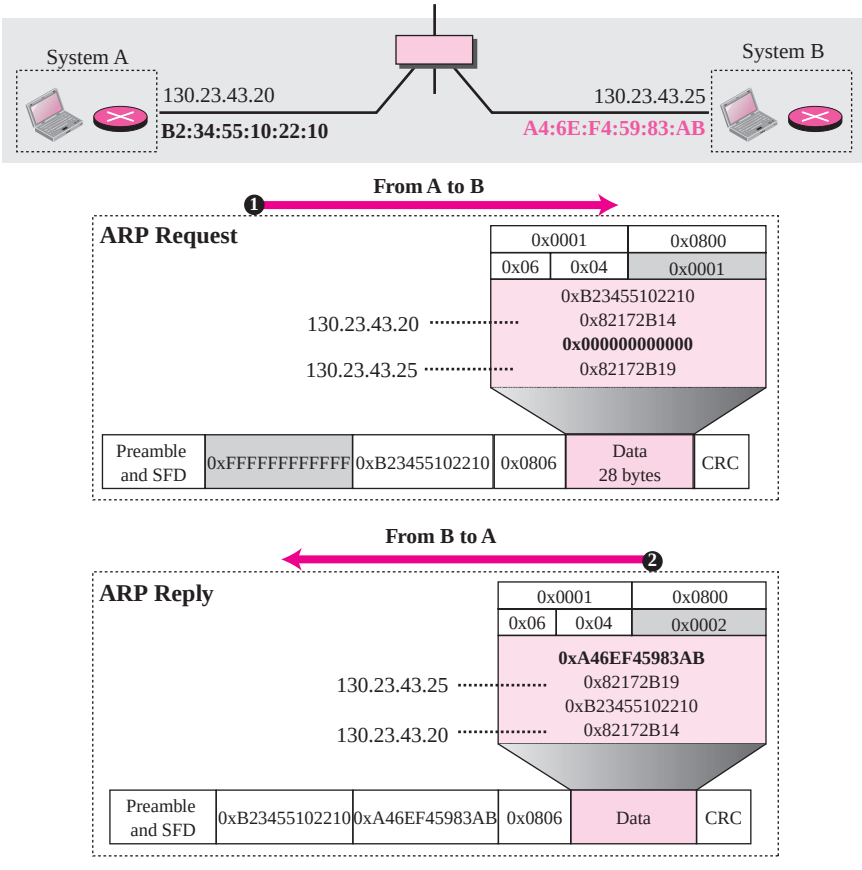
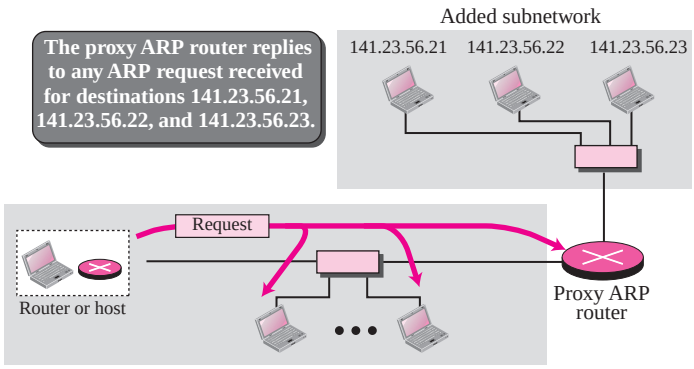


Figure 8.7 *Proxy ARP*



However, the administrator may need to create a subnet without changing the whole system to recognize subnetted addresses. One solution is to add a router running a proxy ARP. In this case, the router acts on behalf of all of the hosts installed on the subnet. When it receives an ARP request with a target IP address that matches the address of one of its protégés (141.23.56.21, 141.23.56.22, and 141.23.56.23), it sends an ARP reply and announces its hardware address as the target hardware address. When the router receives the IP packet, it sends the packet to the appropriate host.

8.3 ATMARP

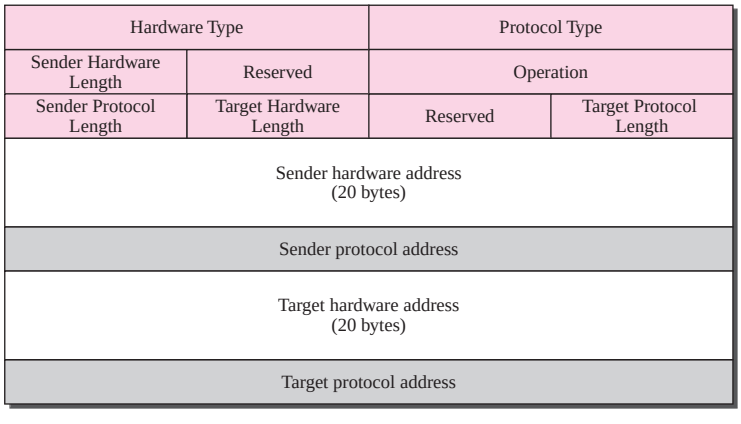
We discussed IP over ATM in Chapter 7. When IP packet are moving through an ATM WAN, a mechanism protocol is needed to find (map) the physical address of the exiting-point router in the ATM WAN given the IP address of the router. This is the same task performed by ARP on a LAN. However, there is a difference between a LAN and an ATM network. A LAN is a broadcast network (at the data link layer); ARP uses the broadcasting capability of a LAN to send (broadcast) an ARP request. An ATM network is not a broadcast network; another solution is needed to handle the task.

Packet Format

The format of an **ATMARP** packet, which is similar to the ARP packet, is shown in Figure 8.8. The fields are as follows:

- **Hardware type (HTYPE).** The 16-bit HTYPE field defines the type of the physical network. Its value is 0013_{16} for an ATM network.
- **Protocol type (PTYPE).** The 16-bit PTYPE field defines the type of the protocol. For IPv4 protocol the value is 0800_{16}.
- **Sender hardware length (SHLEN).** The 8-bit SHLEN field defines the length of the sender's physical address in bytes. For an ATM network the value is 20. Note

Figure 8.8 *ATMARP packet*



that if the binding is done across an ATM network and two levels of hardware addressing are necessary, the neighboring 8-bit **reserved field** is used to define the length of the second address.

- **Operation (OPER).** The 16-bit OPER field defines the type of the packet. Five packet types are defined as shown in Table 8.1.

Table 8.1 *OPER field*

Message	*OPER value*
Request	1
Reply	2
Inverse Request	8
Inverse Reply	9
NACK	10

- **Sender protocol length (SPLEN).** The 8-bit SPLEN field defines the length of the address in bytes. For IPv4 the value is 4 bytes.
- **Target hardware length (TLEN).** The 8-bit TLEN field defines the length of the receiver's physical address in bytes. For an ATM network the value is 20. Note that if the binding is done across an ATM network and two levels of hardware addressing are necessary, the neighboring 8-bit reserved field is used to define the length of the second address.
- **Target protocol length (TPLEN).** The 8-bit TPLEN field defines the length of the address in bytes. For IPv4 the value is 4 bytes.
- **Sender hardware address (SHA).** The variable-length SHA field defines the physical address of the sender. For ATM networks defined by the ATM Forum, the length is 20 bytes.
- **Sender protocol address (SPA).** The variable-length SPA field defines the address of the sender. For IPv4 the length is 4 bytes.
- **Target hardware address (THA).** The variable-length THA field defines the physical address of the receiver. For ATM networks defined by the ATM Forum, the length is 20 bytes. This field is left empty for request messages and filled in for reply and NACK messages.
- **Target protocol address (TPA).** The variable-length TPA field defines the address of the receiver. For IPv4 the length is 4 bytes.

ATMARP Operation

There are two methods to connect two routers on an ATM network: through a permanent virtual circuit (PVC) or through a switched virtual circuit (SVC). The operation of ATMARP depends on the connection method.

PVC Connection

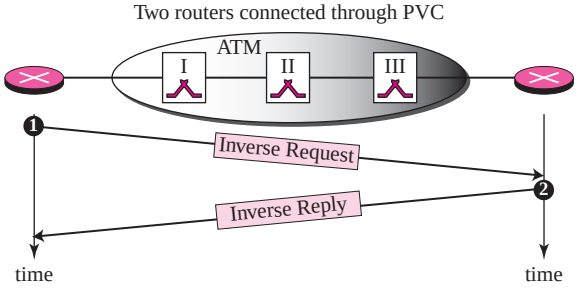
A permanent virtual circuit (PVC) connection is established between two end points by the network provider. The VPIs and VCIs are defined for the permanent connections and the values are entered in a table for each switch.

If a permanent virtual circuit is established between two routers, there is no need for an ATMARP server. However, the routers must be able to bind a physical address to an IP address. The **inverse request message** and **inverse reply message** can be used for the binding. When a PVC is established for a router, the router sends an inverse request message. The router at the other end of the connection receives the message (which contains the physical and IP address of the sender) and sends back an inverse reply message (which contains its own physical and IP address).

After the exchange, both routers add a table entry that maps the physical addresses to the PVC. Now, when a router receives an IP datagram, the table provides information so that the router can encapsulate the datagram using the virtual circuit identifier. Figure 8.9 shows the exchange of messages between two routers.

The inverse request and inverse reply messages can bind the physical address to an IP address in a PVC situation.

Figure 8.9 *Binding with PVC*



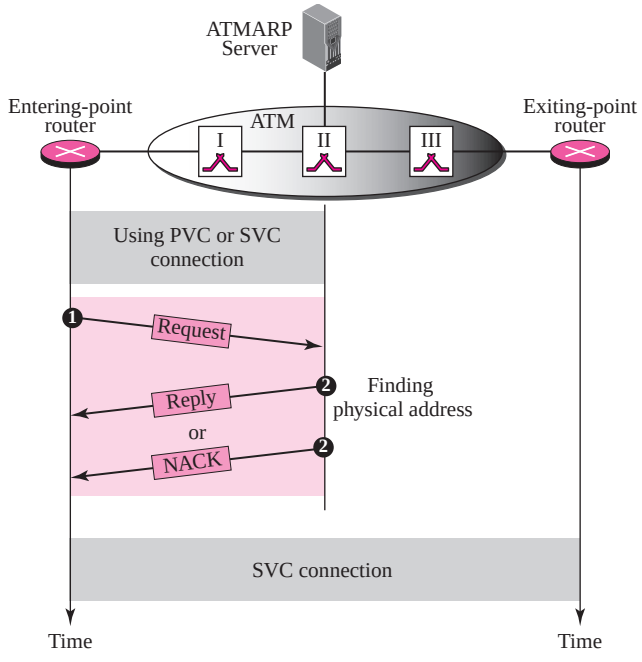
SVC Connection

In a switched virtual circuit (SVC) connection, each time a router wants to make a connection with another router (or any computer), a new virtual circuit must be established. However, the virtual circuit can be created only if the entering-point router knows the physical address of the exiting-point router (ATM does not recognize IP addresses).

To map the IP addresses to physical addresses, each router runs a client ATMARP program, but only one computer runs an ATMARP server program. To understand the difference between ARP and ATMARP, remember that ARP operates on a LAN, which is a broadcast network. An ARP client can broadcast an ARP request message and each router on the network will receive it; only the target router will respond. ATM is a nonbroadcast network; an ATMARP request cannot reach all routers connected to the network.

The process of establishing a virtual connection requires three steps: connecting to the server, receiving the physical address, and establishing the connection. Figure 8.10 shows the steps.

Figure 8.10 *Binding with ATMARP*



Connecting to the Server Normally, there is a permanent virtual circuit established between each router and the server. If there is no PVC connection between the router and the server, the server must at least know the physical address of the router to create an SVC connection just for exchanging ATMARP request and reply messages.

Receiving the Physical Address When there is a connection between the entering-point router and the server, the router sends an *ATMARP request* to the server. The server sends back an *ATMARP reply* if the physical address can be found or an *ATMARP NACK* otherwise. If the entering-point router receives a NACK, the datagram is dropped.

Establishing Virtual Circuits After the entering-point router receives the physical address of the exiting-point router, it can request an SVC between itself and the exiting-point router. The ATM network uses the two physical addresses to set up a virtual circuit which lasts until the entering-point router asks for disconnection. In this step, each switch inside the network adds an entry to its tables to enable them to route the cells carrying the IP datagram.

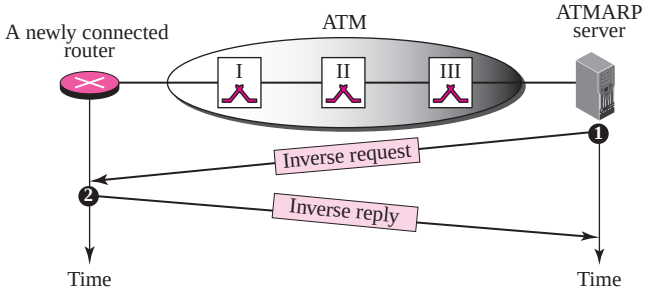
The request and reply message can be used to bind a physical address to an IP address in an SVC situation.

Building the Table

How does the ATM server build its mapping table? This is also done through the use of ATMARP and the two inverse messages (inverse request and inverse reply). When a router is connected to an ATM network for the first time and a permanent virtual connection is established between the router and the server, the server sends an inverse request message to the router. The router sends back an inverse reply message, which includes its IP address and physical address. Using these two addresses, the server creates an entry in its routing table to be used if the router becomes an exiting-point router in the future. Figure 8.11 shows the inverse operation of ATMARP.

The inverse request and inverse reply can also be used to build the server's mapping table.

Figure 8.11 *Building a table*



Logical IP Subnet (LIS)

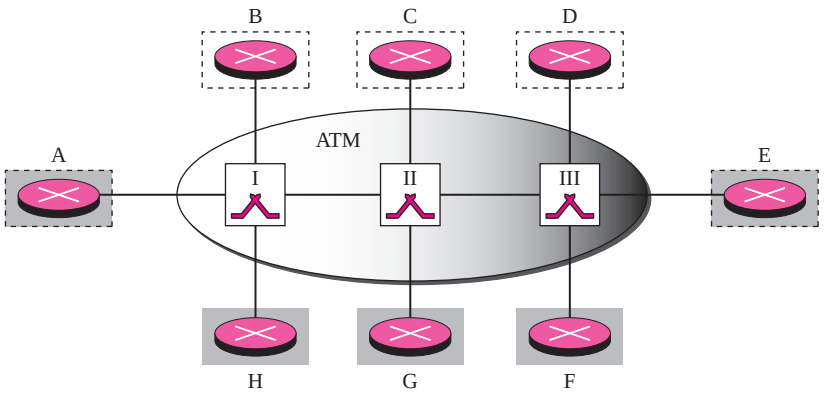
Before we leave the subject of IP over ATM, we need to discuss a concept called **logical IP subnet (LIS).** For the same reason that a large LAN can be divided into several subnets, an ATM network can be divided into logical (not physical) subnetworks. This facilitates the operation of ATMARP and other protocols (such as IGMP) that need to simulate broadcasting on an ATM network.

Routers connected to an ATM network can belong to one or more logical subnets, as shown in Figure 8.12. In the figure, routers B, C, and D belong to one logical subnet (shown by broken-line boxes); routers F, G, and H belong to another logical subnet (shown by shaded boxes). Routers A and E belong to both logical subnets. A router can communicate and send IP packets directly to a router in the same subnet; however, if it needs to send a packet to a router that belongs to another subnet, the packet must first go to a router that belongs to both subnets. For example, router B can send a packet directly to routers C and D. But a packet from B to F must first pass through A or E.

Note that routers belonging to the same logical subnet share the same prefix and subnet mask. The prefix for routers in different subnets is different.

To use ATMARP, there must be a different ATMARP server in each subnet. For example, in the above figure, we need two ATMARP servers, one for each subnet.

Figure 8.12 *LIS*



LIS allows an ATM network to be divided into several logical subnets. To use ATMARP, we need a separate server for each subnet.

8.4 ARP PACKAGE

In this section, we give an example of a simplified ARP software package. The purpose is to show the components of a hypothetical ARP package and the relationships between the components. Figure 8.13 shows these components and their interactions.

We can say that this ARP package involves five components: a **cache table,** queues, an output module, an input module, and a cache-control module. The package receives an IP datagram that needs to be encapsulated in a frame that needs the destination physical (hardware) address. If the ARP package finds this address, it delivers the IP packet and the physical address to the data link layer for transmission.

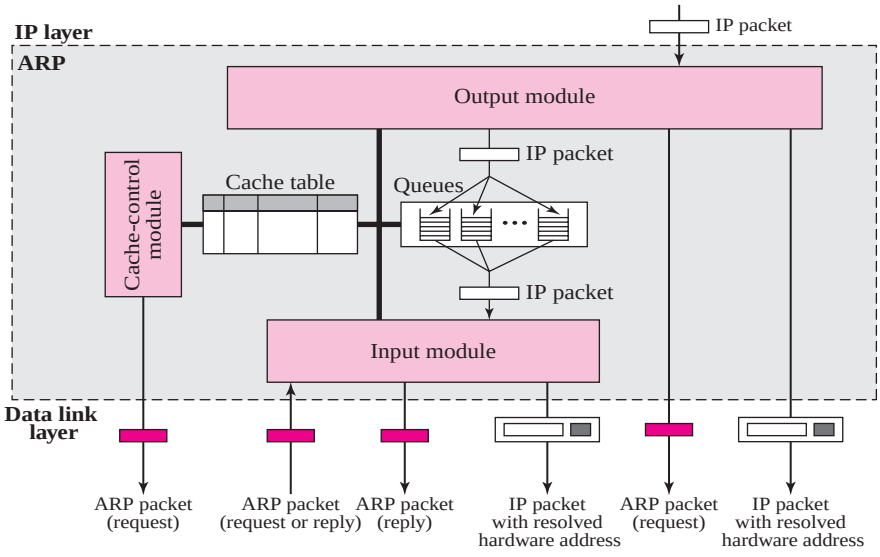
Cache Table

A sender usually has more than one IP datagram to send to the same destination. It is inefficient to use the ARP protocol for each datagram destined for the same host or router. The solution is the cache table. When a host or router receives the corresponding physical address for an IP datagram, the address can be saved in the cache table. This address can be used for the datagrams destined for the same receiver within the next few minutes. However, as space in the cache table is very limited, mappings in the cache are not retained for an unlimited time.

The cache table is implemented as an array of entries. In our package, each entry contains the following fields:

- **State.** This column shows the state of the entry. It can have one of three values: *FREE*, *PENDING*, or *RESOLVED*. The FREE state means that the time-to-live for

Figure 8.13 *ARP components*



this entry has expired. The space can be used for a new entry. The PENDING state means a request for this entry has been sent, but the reply has not yet been received. The RESOLVED state means that the entry is complete. The entry now has the physical (hardware) address of the destination. The packets waiting to be sent to this destination can use the information in this entry.

- **Hardware type.** This column is the same as the corresponding field in the ARP packet.
- **Protocol type.** This column is the same as the corresponding field in the ARP packet.
- **Hardware length.** This column is the same as the corresponding field in the ARP packet.
- **Protocol length.** This column is the same as the corresponding field in the ARP packet.
- **Interface number.** A router (or a multihomed host) can be connected to different networks, each with a different interface number. Each network can have different hardware and protocol types.
- **Queue number.** ARP uses numbered queues to enqueue the packets waiting for address resolution. Packets for the same destination are usually enqueued in the same queue.
- **Attempts.** This column shows the number of times an ARP request is sent out for this entry.
- **Time-out.** This column shows the lifetime of an entry in seconds.
- **Hardware address.** This column shows the destination hardware address. It remains empty until resolved by an ARP reply.
- **Protocol address.** This column shows the destination IP address.

Queues

Our ARP package maintains a set of queues, one for each destination, to hold the IP packets while ARP tries to resolve the hardware address. The output module sends unresolved packets into the corresponding **queue.** The input module removes a packet from a queue and sends it, with the resolved physical address, to the data link layer for transmission.

Output Module

Table 8.2 shows the output module in pseudocode.

Table 8.2 *Output Module*

```
ARP_Output_Module ( )
{
  Sleep until an IP packet is received from IP software.
  Check cache table for an entry corresponding to the
       destination of IP packet.
  If (entry is found)
  {
      If (the state is RESOLVED)
      {
        Extract the value of the hardware address from the entry.
        Send the packet and the hardware address to data
             link layer.
        Return
      } // end if
      If (the state is PENDING)
      {
        Enqueue the packet to the corresponding queue.
        Return
     }//end if
  }//end if
  If (entry is not found)
  {
     Create a cache entry with state set to PENDING and
          ATTEMPTS set to 1.
     Create a queue.
     Enqueue the packet.
     Send an ARP request.
     Return
  }//end if
} //end module
```

The output module waits for an IP packet from the IP software. The output module checks the cache table to find an entry corresponding to the destination IP address of

this packet. The destination IP address of the IP packet must match the protocol address of the entry.

If the entry is found and the state of the entry is RESOLVED, the packet along with the destination hardware address is passed to the data link layer for transmission.

If the entry is found and the state of the entry is PENDING, the packet waits until the destination hardware address is found. Because the state is PENDING, there is a queue already created for this destination. The module sends the packet to this queue.

If no entry is found, the module creates a queue and enqueues the packet. A new entry with the state of PENDING is created for this destination and the value of the ATTEMPTS field is set to 1. An ARP request packet is then broadcast.

Input Module

Table 8.3 shows the input module in pseudocode.

Table 8.3 *Input Module*

```
1  ARP_Input_Module ( )
2  {
3     Sleep until an ARP packet (request or reply) arrives.
4     Check the cache table to find the corresponding entry.
5     If (found)
6     {
7        Update the entry.
8        If (the state is PENDING)
9        {
10          While (the queue is not empty)
11          {
12             Dequeue one packet.
13             Send the packet and the hardware address.
14          }//end if
15       }//end if
16    }//end if
17    If (not found)
18    {
19       Create an entry.
20       Add the entry to the table.
21    }//end if
22    If (the packet is a request)
23    {
24       Send an ARP reply.
25    }//end if
26    Return
27 }//end module
```

The input module waits until an ARP packet (request or reply) arrives. The input module checks the cache table to find an entry corresponding to this ARP packet. The target protocol address should match the protocol address of the entry.

If the entry is found and the state of the entry is PENDING, the module updates the entry by copying the target hardware address in the packet to the hardware address field of the entry and changing the state to RESOLVED. The module also sets the value of the TIME-OUT for this entry. It then dequeues the packets from the corresponding queue, one by one, and delivers them along with the hardware address to the data link layer for transmission.

If the entry is found and the state is RESOLVED, the module still updates the entry. This is because the target hardware address could have been changed. The value of the TIME-OUT field is also reset.

If the entry is not found, the module creates a new entry and adds it to the table. The protocol requires that any information received is added to the table for future use. The state is set to RESOLVED and TIME-OUT is set.

Now the module checks to see if the arrived ARP packet is a request. If it is, the module immediately creates an ARP reply message and sends it to the sender. The ARP reply packet is created by changing the value of the operation field from request to reply and filling in the target hardware address.

Cache-Control Module

The **cache-control module** is responsible for maintaining the cache table. It periodically (for example, every 5 s) checks the cache table, entry by entry. If the state of the entry is FREE, it continues to the next entry. If the state is PENDING, the module increments the value of the attempts field by 1. It then checks the value of the attempts field. If this value is greater than the maximum number of attempts allowed, the state is changed to FREE and the corresponding queue is destroyed. However, if the number of attempts is less than the maximum, the module creates and sends another ARP request.

If the state of the entry is RESOLVED, the module decrements the value of the time-out field by the amount of time elapsed since the last check. If this value is less than or equal to zero, the state is changed to FREE and the queue is destroyed. Table 8.4 shows the cache-control module in pseudocode.

Table 8.4 *Cache-Control Module*

```
ARP_Cache_Control_Module ( )
{
    Sleep until the periodic timer matures.
    Repeat for every entry in the cache table
    {
       If (the state is FREE)
       {
          Continue.
       }//end if
        If (the state is PENDING)
        {
```

Table 8.4 *Cache-Control Module (continued)*

```
12          Increment the value of attempts by 1.
13          If (attempts greater than maximum)
14          {
15               Change the state to FREE.
16               Destroy the corresponding queue.
17          }// end if
18          else
19          {
20             Send an ARP request.
21           }//end else
22           continue.
23         }//end if
24         If (the state is RESOLVED)
25         {
26             Decrement the value of time-out.
27             If (time-out less than or equal 0)
28             {
29                Change the state to FREE.
30                Destroy the corresponding queue.
31            }//end if
32         }//end if
33       }//end repeat
34       Return.
35    }//end module
```

More Examples

In this section we show some examples of the ARP operation and the changes in the cache table. Table 8.5 shows some of the cache table fields at the start of our examples.

Table 8.5 *Original cache table used for examples*

State	*Queue*	*Attempt*	*Time-Out*	*Protocol Addr.*	*Hardware Addr.*
R	5		900	180.3.6.1	ACAE32457342
P	2	2		129.34.4.8	
P	14	5		201.11.56.7	
R	8		450	114.5.7.89	457342ACAE32
P	12	1		220.55.5.7	
F					
R	9		60	19.1.7.82	4573E3242ACA
P	18	3		188.11.8.71	

Example 8.2

The ARP output module receives an IP datagram (from the IP layer) with the destination address 114.5.7.89. It checks the cache table and finds that an entry exists for this destination with the RESOLVED state (R in the table). It extracts the hardware address, which is 457342ACAE32, and sends the packet and the address to the data link layer for transmission. The cache table remains the same.

Example 8.3

Twenty seconds later, the ARP output module receives an IP datagram (from the IP layer) with the destination address 116.1.7.22. It checks the cache table and does not find this destination in the table. The module adds an entry to the table with the state PENDING and the Attempt value 1. It creates a new queue for this destination and enqueues the packet. It then sends an ARP request to the data link layer for this destination. The new cache table is shown in Table 8.6.

Table 8.6 *Updated cache table for Example 8.3*

State	*Queue*	*Attempt*	*Time-Out*	*Protocol Addr.*	*Hardware Addr.*
R	5		900	180.3.6.1	ACAE32457342
P	2	2		129.34.4.8	
P	14	5		201.11.56.7	
R	8		450	114.5.7.89	457342ACAE32
P	12	1		220.55.5.7	
P	23	1		116.1.7.22	
R	9		60	19.1.7.82	4573E3242ACA
P	18	3		188.11.8.71	

Example 8.4

Fifteen seconds later, the ARP input module receives an ARP packet with target protocol (IP) address 188.11.8.71. The module checks the table and finds this address. It changes the state of the entry to RESOLVED and sets the time-out value to 900. The module then adds the target hardware address (E34573242ACA) to the entry. Now it accesses queue 18 and sends all the packets in this queue, one by one, to the data link layer. The new cache table is shown in Table 8.7.

Table 8.7 *Updated cache table for Example 8.4*

State	*Queue*	*Attempt*	*Time-Out*	*Protocol Addr.*	*Hardware Addr.*
R	5		900	180.3.6.1	ACAE32457342
P	2	2		129.34.4.8	
P	14	5		201.11.56.7	
R	8		450	114.5.7.89	457342ACAE32
P	12	1		220.55.5.7	
P	23	1		116.1.7.22	
R	9		60	19.1.7.82	4573E3242ACA
R	18		900	188.11.8.71	E34573242ACA

Example 8.5

Twenty-five seconds later, the cache-control module updates every entry. The time-out values for the first three resolved entries are decremented by 60. The time-out value for the last

resolved entry is decremented by 25. The state of the next-to-the last entry is changed to FREE because the time-out is zero. For each of the three pending entries, the value of the attempts field is incremented by one. After incrementing, the attempts value for one entry (the one with IP address 201.11.56.7) is more than the maximum; the state is changed to FREE, the queue is deleted, and an ICMP message is sent to the original destination (see Chapter 9). See Table 8.8.

Table 8.8 *Updated cache table for Example 8.5*

State	*Queue*	*Attempt*	*Time-Out*	*Protocol Addr.*	*Hardware Addr.*
R	5		840	180.3.6.1	ACAE32457342
P	2	3		129.34.4.8	
F					
R	8		390	114.5.7.89	457342ACAE32
P	12	2		220.55.5.7	
P	23	2		116.1.7.22	
F					
R	18		875	188.11.8.71	E34573242ACA

8.5 FURTHER READING

For more details about subjects discussed in this chapter, we recommend the following books and RFCs. The items enclosed in brackets refer to the reference list at the end of the book.

Books

Several books give thorough coverage of materials discussed in this chapter. We recommend [Com 06], [Tan 03], and [Ste 94].

RFCs

Several RFCs in particular discuss ARP including RFC 826, RFC 1029, RFC 1166, and RFC 1981.

8.6 KEY TERMS

Address Resolution Protocol (ARP)
cache table
cache-control module
dynamic mapping
encapsulation
inverse reply message
inverse request message
IP addresses
logical IP subnet (LIS)
physical address
proxy ARP
queue
reserved field
Reverse Address Resolution Protocol (RARP)
static mapping