8-1 ADDRESS MAPPING

The 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 disadvantages:

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.

8-2 ADDRESS MAPPING

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. 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*

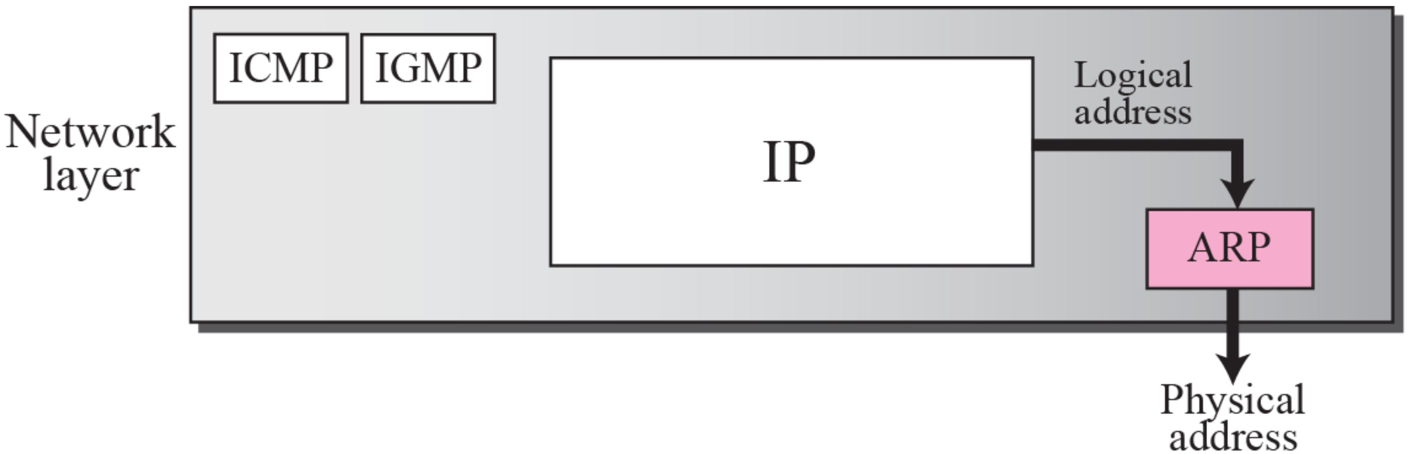
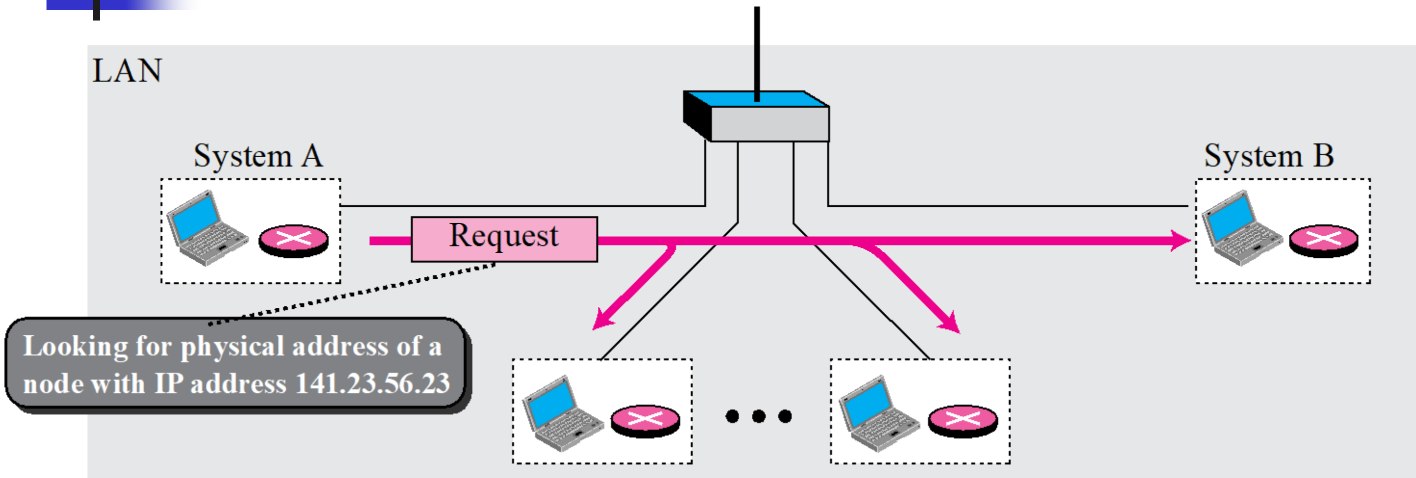
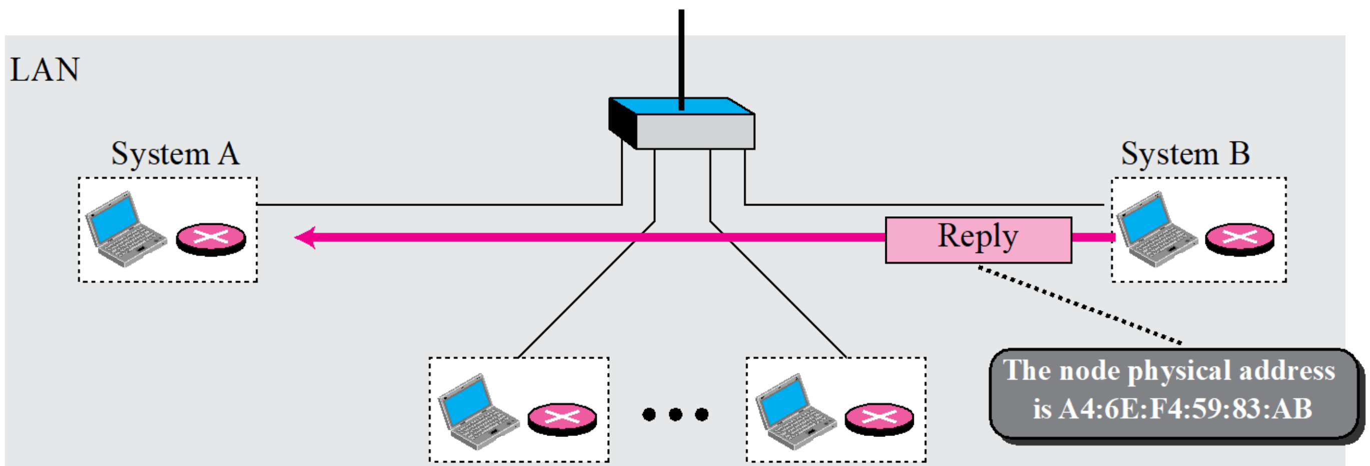


Figure 8.2 *ARP operation*



a. ARP request is broadcast

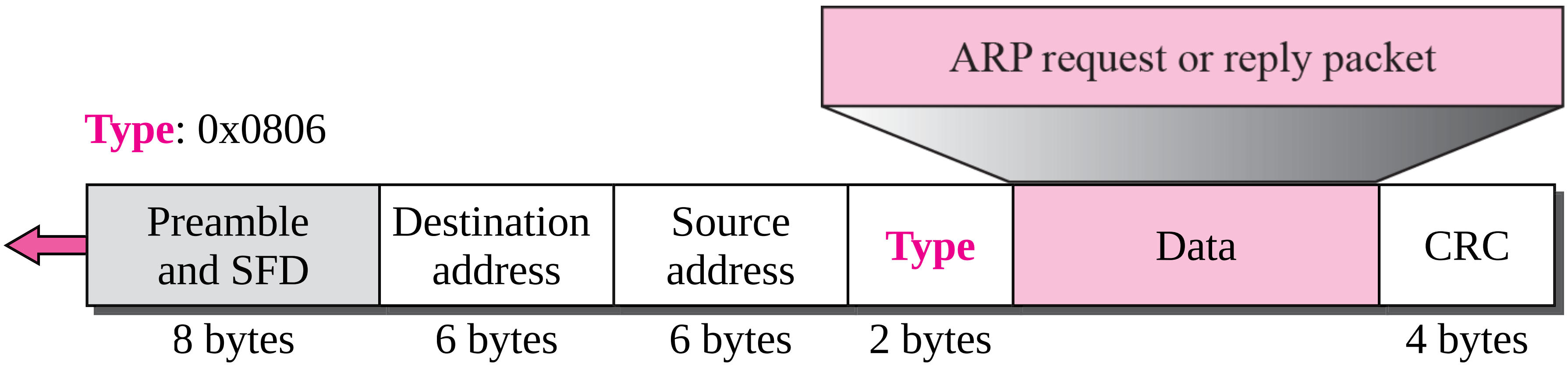


b. ARP reply is unicast

Figure 8.3 *ARP packet*

<table>
<tr><td colspan="2">Hardware Type</td><td>Protocol Type</td></tr>
<tr><td>Hardware length</td><td>Protocol length</td><td>Operation
Request 1, Reply 2</td></tr>
<tr><td colspan="3">Sender hardware address
(For example, 6 bytes for Ethernet)</td></tr>
<tr><td colspan="3">Sender protocol address
(For example, 4 bytes for IP)</td></tr>
<tr><td colspan="3">Target hardware address
(For example, 6 bytes for Ethernet)
(It is not filled in a request)</td></tr>
<tr><td colspan="3">Target protocol address
(For example, 4 bytes for IP)</td></tr>
</table>

Figure 8.4 *Encapsulation of ARP packet*

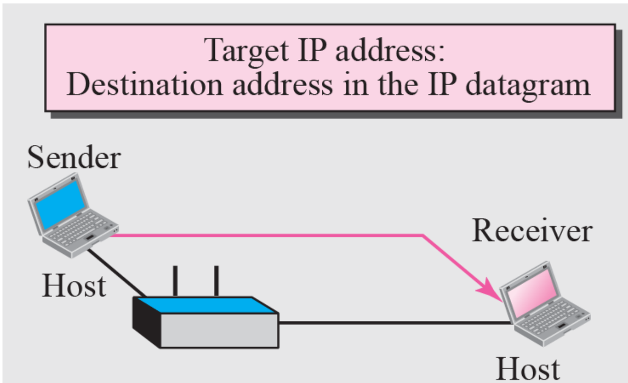


Note

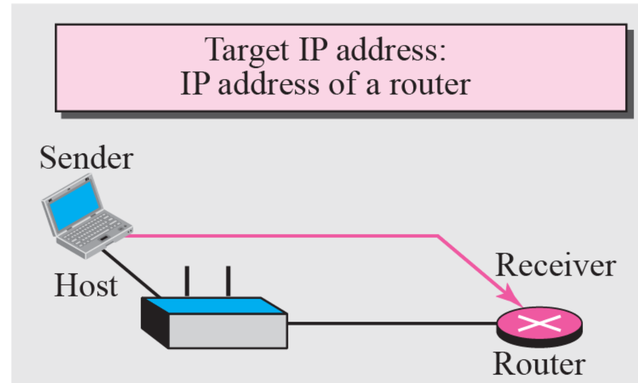
An ARP request is broadcast;
an ARP reply is unicast.

Figure 8.5 *Four cases using ARP*

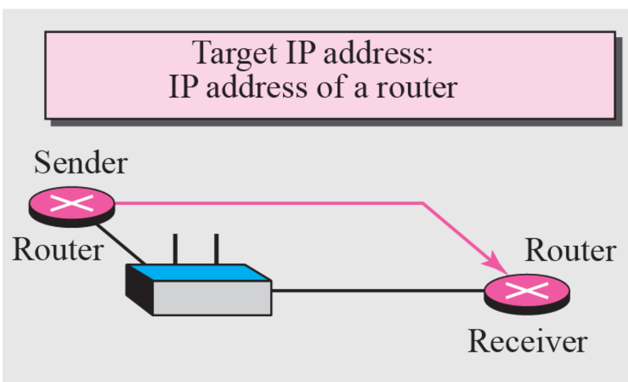
Case 1: A host has a packet to send to a host on the same network.



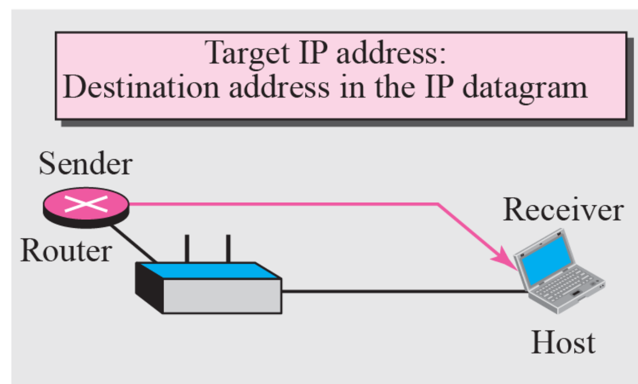
Case 2: A host has a packet to send to a host on another network.



Case 3: A router has a packet to send to a host on another network.



Case 4: A router has a packet to send to a host on the same network.



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. 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.

Figure 8.6 *Example 8.1*

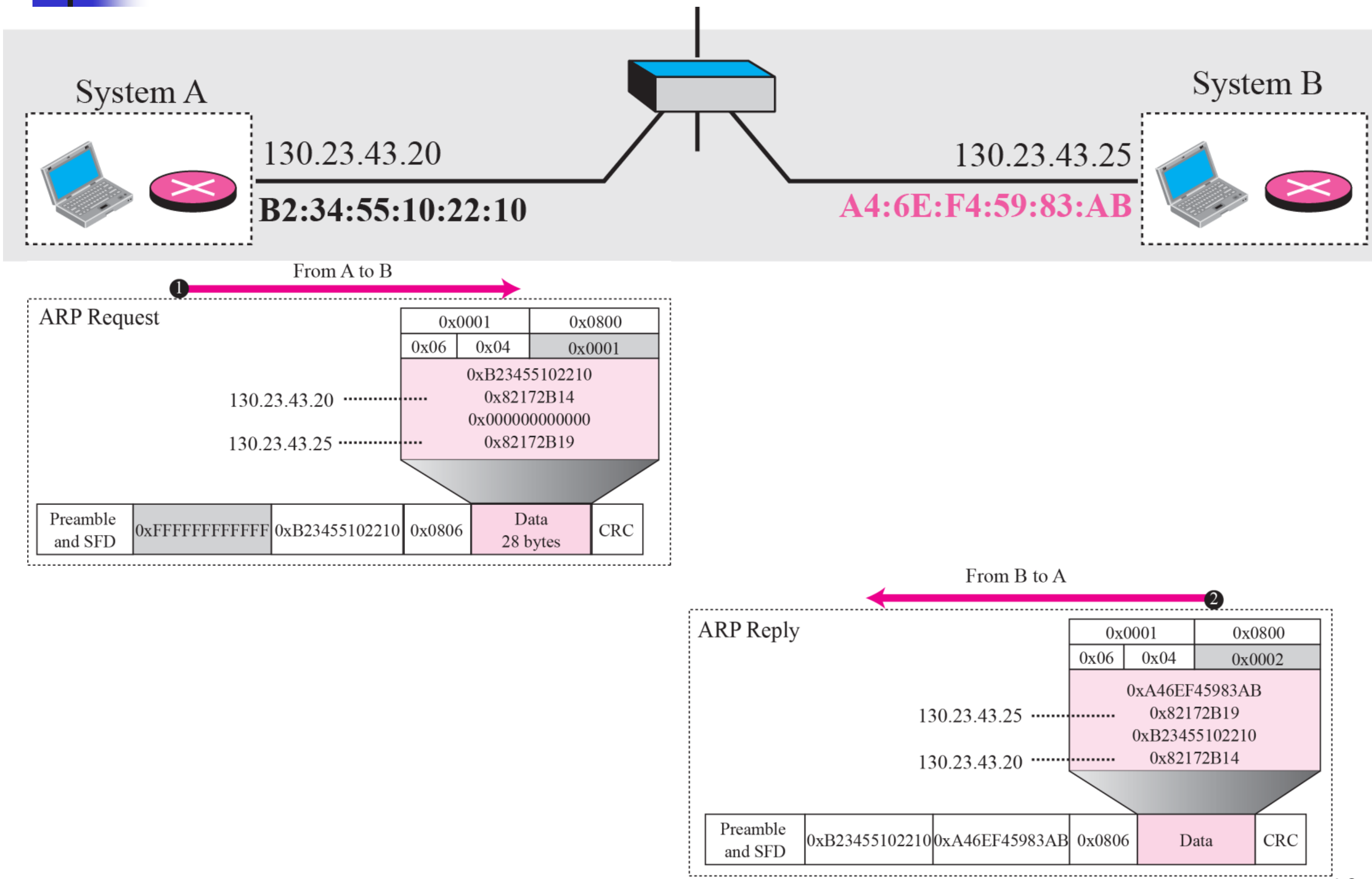


Figure 8.7 *Proxy ARP*

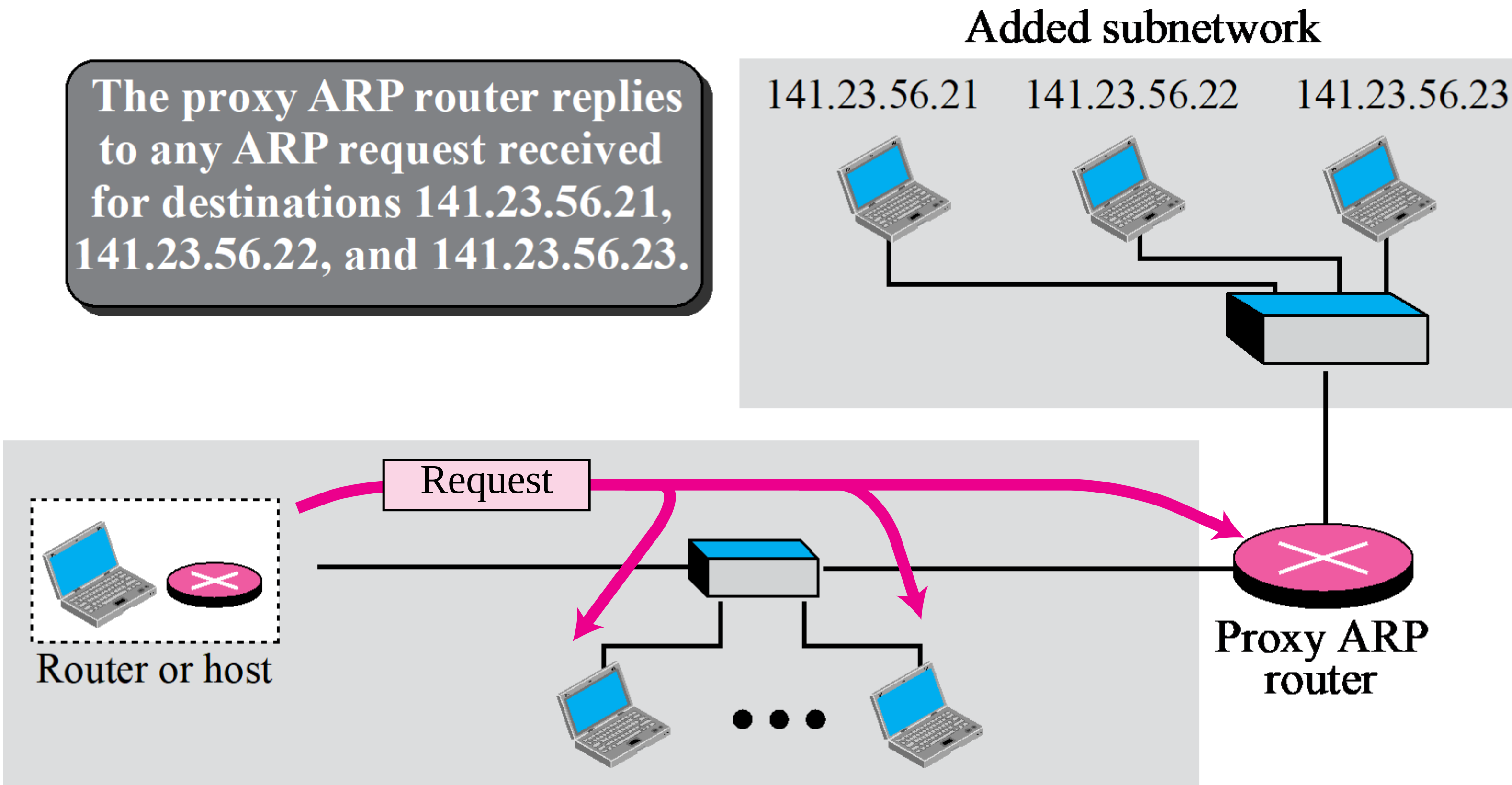


Figure 8.13 *ARP components*

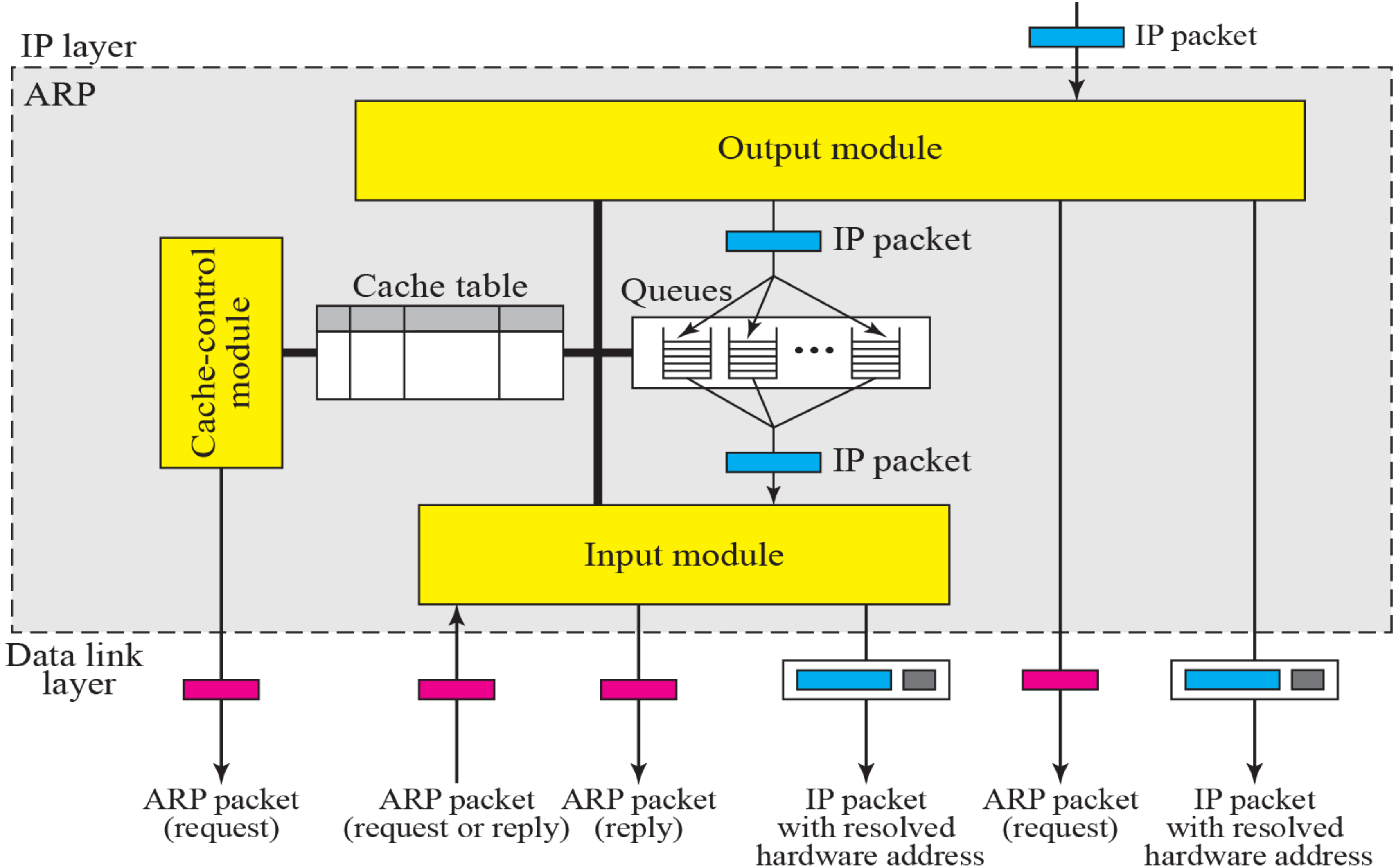


Table 8.2 *Output Module*

```
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
```

Table 8.3 *Input Module*

```
ARP_Input_Module ( )
{
   Sleep until an ARP packet (request or reply) arrives.
   Check the cache table to find the corresponding entry.
   If (found)
   {
      Update the entry.
      If (the state is PENDING)
      {
         While (the queue is not empty)
         {
            Dequeue one packet.
            Send the packet and the hardware address.
         }//end if
      }//end if
   }//end if
   If (not found)
   {
      Create an entry.
      Add the entry to the table.
   }//end if
   If (the packet is a request)
   {
      Send an ARP reply.
   }//end if
   Return
}//end module
```

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)*

```
            Increment the value of attempts by 1.
            If (attempts greater than maximum)
            {
                Change the state to FREE.
                Destroy the corresponding queue.
            }// end if
            else
            {
              Send an ARP request.
            }//end else
            continue.
          }//end if
          If (the state is RESOLVED)
          {
              Decrement the value of time-out.
              If (time-out less than or equal 0)
              {
                 Change the state to FREE.
                 Destroy the corresponding queue.
              }//end if
          }//end if
      }//end repeat
      Return.
   }//end module
```

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