5

Mapping Internet Addresses To Physical Addresses (ARP)

5.1 Introduction

We described the TCP/IP address scheme in which each host is assigned a 32-bit address, and said that an internet behaves like a virtual network, using only the assigned addresses when sending and receiving packets. We also reviewed several network hardware technologies, and noted that two machines on a given physical network can communicate *only if they know each other's physical network address.* What we have not mentioned is how a host or a router maps an IP address to the correct physical address when it needs to send a packet across a physical net. This chapter considers that mapping, showing how it is implemented for the two most common physical network address schemes.

5.2 The Address Resolution Problem

Consider two machines *A* and *B* that connect to the same physical network. Each has an assigned IP address I_A and I_B and a physical address P_A and P_B. The goal is to devise low-level software that hides physical addresses and allows higher-level programs to work only with internet addresses. Ultimately, however, communication must be carried out by physical networks using whatever physical address scheme the underlying network hardware supplies. Suppose machine *A* wants to send a packet to machine *B* across a physical network to which they both attach, but *A* has only *B*'s internet address I_B. The question arises: how does *A* map that address to *B's* physical address, P_B?

Address mapping must be performed at each step along a path from the original source to the ultimate destination. In particular, two cases arise. First, at the last step of delivering a packet, the packet must be sent across one physical network to its final destination. The computer sending the packet must map the final destination's Internet address to the destination's physical address. Second, at any point along the path from the source to the destination other than the final step, the packet must be sent to an intermediate router. Thus, the sender must map the intermediate router's Internet address to a physical address.

The problem of mapping high-level addresses to physical addresses is known as the *address resolution problem* and has been solved in several ways. Some protocol suites keep tables in each machine that contain pairs of high-level and physical addresses. Others solve the problem by encoding hardware addresses in high-level addresses. Using either approach exclusively makes high-level addressing awkward at best. This chapter discusses two techniques for address resolution used by TCP/IP protocols and shows when each is appropriate.

5.3 Two Types Of Physical Addresses

There are two basic types of physical addresses, exemplified by the Ethernet, which has large, fixed physical addresses, and proNET, which has small, easily configured physical addresses. Address resolution is difficult for Ethernet-like networks, but easy for networks like proNET. We will consider the easy case first.

5.4 Resolution Through Direct Mapping

Consider a proNET token ring network. Recall from Chapter 2 that proNET uses small integers for physical addresses and allows the user to choose a hardware address when installing an interface board in a computer. The key to making address resolution easy with such network hardware lies in observing that as long as one has the freedom to choose both IP and physical addresses, they can be selected such that parts of them are the same. Typically, one assigns IP addresses with the hostid portion equal to 1, 2, 3, and so on, and then, when installing network interface hardware, selects a physical address that corresponds to the IP address. For example, the system administrator would select physical address 3 for a computer with the IP address 192.5.48.3 because 192.5.48.3 is a class *C* address with the host portion equal to 3.

For networks like proNET, computing a physical address from an IP address is trivial. The computation consists of extracting the host portion of the IP address. Extraction is computationally efficient on most architectures because it requires only a few machine instructions. The mapping is easy to maintain because it can be performed without reference to external data. Finally, new computers can be added to the network without changing existing assignments or recompiling code.

Conceptually, choosing a numbering scheme that makes address resolution efficient means selecting a function f that maps IP addresses to physical addresses. The designer may be able to select a physical address numbering scheme as well, depending on the hardware. Resolving IP address I_A means computing

$$P_A = f(I_A)$$

We want the computation of f to be efficient. If the set of physical addresses is constrained, it may be possible to arrange efficient mappings other than the one given in the example above. For instance, when using IP over a connection-oriented network such as ATM, one cannot choose physical addresses. On such networks, one or more computers (servers) store pairs of addresses, where each pair contains an Internet address and the corresponding physical address. Typically, such servers store the pairs in a table in memory to speed searching. To guarantee efficient address resolution in such cases, software can use a conventional hash function to search the table. Exercise 5.1 suggests a related alternative.

5.5 Resolution Through Dynamic Binding

To understand why address resolution is difficult for some networks, consider Ethernet technology. Recall from Chapter 2 that each Ethernet interface is assigned a 48-bit physical address when the device is manufactured. As a consequence, when hardware fails and requires that an Ethernet interface be replaced, the machine's physical address changes. Furthermore, because the Ethernet address is 48 bits long, there is no hope it can be encoded in a 32-bit IP address†.

Designers of TCP/IP protocols found a creative solution to the address resolution problem for networks like the Ethernet that have broadcast capability. The solution allows new hosts or routers to be added to the network without recompiling code, and does not require maintenance of a centralized database. To avoid maintaining a table of mappings, the designers chose to use a low-level protocol to bind addresses dynamically. Termed the *Address Resolution Protocol* (*ARP*), the protocol provides a mechanism that is both reasonably efficient and easy to maintain.

As Figure 5.1 shows, the idea behind dynamic resolution with ARP is simple: when host A wants to resolve IP address I_B, it broadcasts a special packet that asks the host with IP address I_B to respond with its physical address, P_B. All hosts, including B, receive the request, but only host B recognizes its IP address and sends a reply that contains its physical address. When A receives the reply, it uses the physical address to send the internet packet directly to B. We can summarize:

---

†Because direct mapping is more convenient and efficient than dynamic binding, the next generation of IP is being designed to allow 48-bit hardware addresses to be encoded in IP addresses.

The Address Resolution Protocol, ARP, allows a host to find the physical address of a target host on the same physical network, given only the target's IP address.

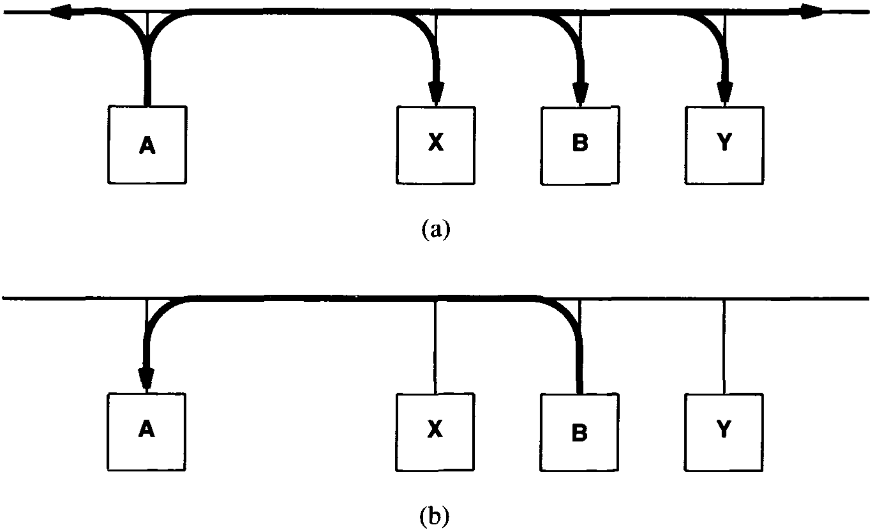

Figure 5.1 The ARP protocol. To determine P_B, B's physical address, from I_B, its IP address, (a) host A broadcasts an ARP request containing I_B to all machines on the net, and (b) host B responds with an ARP reply that contains the pair (I_B, P_B).

5.6 The Address Resolution Cache

It may seem silly that for A to send a packet to B it first sends a broadcast that reaches B. Or it may seem even sillier that A broadcasts the question, "how can I reach you?" instead of just broadcasting the packet it wants to deliver. But there is an important reason for the exchange. Broadcasting is far too expensive to be used every time one machine needs to transmit a packet to another because every machine on the network must receive and process the broadcast packet.

5.7 ARP Cache Timeout

To reduce communication costs, computers that use ARP maintain a cache of recently acquired IP-to-physical address bindings. That is, whenever a computer sends an ARP request and receives an ARP reply, it saves the IP address and corresponding hardware address information in its cache for successive lookups. When transmitting a packet, a computer always looks in its cache for a binding before sending an ARP request. If it finds the desired binding in its ARP cache, the computer need not broadcast on the network. Thus, when two computers on a network communicate, they begin with an ARP request and response, and then repeatedly transfer packets without using ARP for each one. Experience shows that because most network communication involves more than one packet transfer, even a small cache is worthwhile.

The ARP cache provides an example of *soft state*, a technique commonly used in network protocols. The name describes a situation in which information can become ''stale'' without warning. In the case of ARP, consider two computers, *A* and *B*, both connected to an Ethernet. Assume *A* has sent an ARP request, and *B* has replied. Further assume that after the exchange *B* crashes. Computer *A* will not receive any notification of the crash. Moreover, because it already has address binding information for *B* in its ARP cache, computer *A* will continue to send packets to *B*. The Ethernet hardware provides no indication that *B* is not on-line because Ethernet does not have guaranteed delivery. Thus, *A* has no way of knowing when information in its ARP cache has become incorrect.

To accommodate soft state, responsibility for correctness lies with the owner of the information. Typically, protocols that implement soft state use timers, with the state information being deleted when the timer expires. For example, whenever address binding information is placed in an ARP cache, the protocol requires a timer to be set, with a typical timeout being 20 minutes. When the timer expires, the information must be removed. After removal there are two possibilities. If no further packets are sent to the destination, nothing occurs. If a packet must be sent to the destination and there is no binding present in the cache, the computer follows the normal procedure of broadcasting an ARP request and obtaining the binding. If the destination is still reachable, the binding will again be placed in the ARP cache. If not, the sender will discover that the destination is off-line.

The use of soft state in ARP has advantages and disadvantages. The chief advantage arises from autonomy. First, a computer can determine when information in its ARP cache should be revalidated independent of other computers. Second, a sender does not need successful communication with the receiver or a third party to determine that a binding has become invalid; if a target does not respond to an ARP request, the sender will declare the target to be down. Third, the scheme does not rely on network hardware to provide reliable transfer. The chief disadvantage of soft state arises from delay — if the timer interval is *N* seconds, a sender may not detect that a receiver has crashed until *N* seconds elapse.

5.8 ARP Refinements

Several refinements of ARP have been included in the protocol. First, observe that if host *A* is about to use ARP because it needs to send to *B*, there is a high probability that host *B* will need to send to *A* in the near future. To anticipate *B*'s need and avoid extra network traffic, *A* includes its IP-to-physical address binding when sending *B* a request. *B* extracts *A*'s binding from the request, saves the binding in its ARP cache, and then sends a reply to *A*. Second, notice that because *A* broadcasts its initial request, all machines on the network receive it and can extract and update *A*'s IP-to-physical address binding in their cache. Third, when a computer has its host interface replaced, (e.g., because the hardware has failed) its physical address changes. Other computers on the net that have stored a binding in their ARP cache need to be informed so they can change the entry. The computer can notify others of a new address by sending an ARP broadcast when it boots.

The following rule summarizes refinements:

> *The sender's IP-to-physical address binding is included in every ARP broadcast; receivers update the IP-to-physical address binding information in their cache before processing an ARP packet.*

5.9 Relationship Of ARP To Other Protocols

ARP provides one possible mechanism to map from IP addresses to physical addresses; we have already seen that some network technologies do not need it. The point is that ARP would be completely unnecessary if we could make all network hardware recognize IP addresses. Thus, ARP merely imposes a new address scheme on top of whatever low-level address mechanism the hardware uses. The idea can be summarized:

> *ARP is a low-level protocol that hides the underlying network physical addressing, permitting one to assign an arbitrary IP address to every machine. We think of ARP as part of the physical network system, and not as part of the internet protocols.*

5.10 ARP Implementation

Functionally, ARP is divided into two parts. The first part maps an IP address to a physical address when sending a packet, and the second part answers requests from other machines. Address resolution for outgoing packets seems straightforward, but small details complicate an implementation. Given a destination IP address the software consults its ARP cache to see if it knows the mapping from IP address to physical address.

If it does, the software extracts the physical address, places the data in a frame using that address, and sends the frame. If it does not know the mapping, the software must broadcast an ARP request and wait for a reply.

Broadcasting an ARP request to find an address mapping can become complex. The target machine can be down or just too busy to accept the request. If so, the sender may not receive a reply or the reply may be delayed. Because the Ethernet is a best-effort delivery system, the initial ARP broadcast request can also be lost (in which case the sender should retransmit, at least once). Meanwhile, the host must store the original outgoing packet so it can be sent once the address has been resolved†. In fact, the host must decide whether to allow other application programs to proceed while it processes an ARP request (most do). If so, the software must handle the case where an application generates additional ARP requests for the same address without broadcasting multiple requests for a given target.

Finally, consider the case where machine *A* has obtained a binding for machine *B*, but then *B*'s hardware fails and is replaced. Although *B*'s address has changed, *A*'s cached binding has not, so *A* uses a nonexistent hardware address, making successful reception impossible. This case shows why it is important to have ARP software treat its table of bindings as a cache and remove entries after a fixed period. Of course, the timer for an entry in the cache must be reset whenever an ARP broadcast arrives containing the binding (but it is not reset when the entry is used to send a packet).

The second part of the ARP code handles ARP packets that arrive from the network. When an ARP packet arrives, the software first extracts the sender's IP address and hardware address pair, and examines the local cache to see if it already has an entry for the sender. If a cache entry exists for the given IP address, the handler updates that entry by overwriting the physical address with the physical address obtained from the packet. The receiver then processes the rest of the ARP packet.

A receiver must handle two types of incoming ARP packets. If an ARP request arrives, the receiving machine must see if it is the target of the request (i.e., some other machine has broadcast a request for the receiver's physical address). If so, the ARP software forms a reply by supplying its physical hardware address, and sends the reply directly back to the requester. The receiver also adds the sender's address pair to its cache if the pair is not already present. If the IP address mentioned in the ARP request does not match the local IP address, the packet is requesting a mapping for some other machine on the network and can be ignored.

The other interesting case occurs when an ARP reply arrives. Depending on the implementation, the handler may need to create a cache entry, or the entry may have been created when the request was generated. In any case, once the cache has been updated, the receiver tries to match the reply with a previously issued request. Usually, replies arrive in response to a request, which was generated because the machine has a packet to deliver. Between the time a machine broadcasts its ARP request and receives the reply, application programs or higher-level protocols may generate additional requests for the same address; the software must remember that it has already sent a request and not send more. Usually, ARP software places the additional packets on a queue. Once the reply arrives and the address binding is known, the ARP software re-

---

†If the delay is significant, the host may choose to discard the outgoing packet(s).

moves packets from the queue, places each packet in a frame, and uses the address binding to fill in the physical destination address. If it did not previously issue a request for the IP address in the reply, the machine updates the sender's entry in its cache, and then simply stops processing the packet.

5.11 ARP Encapsulation And Identification

When ARP messages travel from one machine to another, they must be carried in physical frames. Figure 5.2 shows that the ARP message is carried in the data portion of a frame.

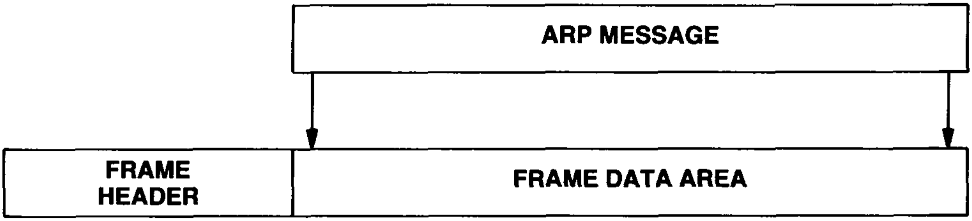

Figure 5.2 An ARP message encapsulated in a physical network frame.

To identify the frame as carrying an ARP message, the sender assigns a special value to the type field in the frame header, and places the ARP message in the frame's data field. When a frame arrives at a computer, the network software uses the frame type to determine its contents. In most technologies, a single type value is used for all frames that carry an ARP message — network software in the receiver must further examine the ARP message to distinguish between ARP requests and ARP replies. For example, on an Ethernet, frames carrying ARP messages have a type field of 0806_{16}. This is a standard value assigned by the authority for Ethernet; other network hardware technologies use other values.

5.12 ARP Protocol Format

Unlike most protocols, the data in ARP packets does not have a fixed-format header. Instead, to make ARP useful for a variety of network technologies, the length of fields that contain addresses depend on the type of network. However, to make it possible to interpret an arbitrary ARP message, the header includes fixed fields near the beginning that specify the lengths of the addresses found in succeeding fields. In fact, the ARP message format is general enough to allow it to be used with arbitrary physical addresses and arbitrary protocol addresses. The example in Figure 5.3 shows the 28-octet ARP message format used on Ethernet hardware (where physical addresses are

48-bits or 6 octets long), when resolving IP protocol addresses (which are 4 octets long).

Figure 5.3 shows an ARP message with 4 octets per line, a format that is standard throughout this text. Unfortunately, unlike most of the remaining protocols, the variable-length fields in ARP packets do not align neatly on 32-bit boundaries, making the diagram difficult to read. For example, the sender's hardware address, labeled *SENDER HA*, occupies 6 contiguous octets, so it spans two lines in the diagram.

<table>
<tr><th>0</th><th>8</th><th>16</th><th>24 31</th></tr>
<tr><td colspan="2">HARDWARE TYPE</td><td colspan="2">PROTOCOL TYPE</td></tr>
<tr><td>HLEN</td><td>PLEN</td><td colspan="2">OPERATION</td></tr>
<tr><td colspan="4">SENDER HA (octets 0-3)</td></tr>
<tr><td colspan="2">SENDER HA (octets 4-5)</td><td colspan="2">SENDER IP (octets 0-1)</td></tr>
<tr><td colspan="2">SENDER IP (octets 2-3)</td><td colspan="2">TARGET HA (octets 0-1)</td></tr>
<tr><td colspan="4">TARGET HA (octets 2-5)</td></tr>
<tr><td colspan="4">TARGET IP (octets 0-3)</td></tr>
</table>

Figure 5.3 An example of the ARP/RARP message format when used for IP-to-Ethernet address resolution. The length of fields depends on the hardware and protocol address lengths, which are 6 octets for an Ethernet address and 4 octets for an IP address.

Field *HARDWARE TYPE* specifies a hardware interface type for which the sender seeks an answer; it contains the value *1* for Ethernet. Similarly, field *PROTOCOL TYPE* specifies the type of high-level protocol address the sender has supplied; it contains 0800_{16} for IP addresses. Field *OPERATION* specifies an ARP request (*1*), ARP response (*2*), RARP† request (*3*), or RARP response (*4*). Fields *HLEN* and *PLEN* allow ARP to be used with arbitrary networks because they specify the length of the hardware address and the length of the high-level protocol address. The sender supplies its hardware address and IP address, if known, in fields *SENDER HA* and *SENDER IP*.

When making a request, the sender also supplies the target hardware address (RARP) or target IP address (ARP), using fields *TARGET HA* or *TARGET IP*. Before the target machine responds, it fills in the missing addresses, swaps the target and sender pairs, and changes the operation to a reply. Thus, a reply carries the IP and hardware addresses of the original requester, as well as the IP and hardware addresses of the machine for which a binding was sought.

---

†The next chapter describes RARP, another protocol that uses the same message format.

5.13 Summary

IP addresses are assigned independent of a machine's physical hardware address. To send an internet packet across a physical net from one computer to another, the network software must map the IP address into a physical hardware address and use the hardware address to transmit the frame. If hardware addresses are smaller than IP addresses, a direct mapping can be established by having the machine's physical address encoded in its IP address. Otherwise, the mapping must be performed dynamically. The Address Resolution Protocol (ARP) performs dynamic address resolution, using only the low-level network communication system. ARP permits machines to resolve addresses without keeping a permanent record of bindings.

A machine uses ARP to find the hardware address of another machine by broadcasting an ARP request. The request contains the IP address of the machine for which a hardware address is needed. All machines on a network receive an ARP request. If the request matches a machine's IP address, the machine responds by sending a reply that contains the needed hardware address. Replies are directed to one machine; they are not broadcast.

To make ARP efficient, each machine caches IP-to-physical address bindings. Because internet traffic tends to consist of a sequence of interactions between pairs of machines, the cache eliminates most ARP broadcast requests.

FOR FURTHER STUDY

The address resolution protocol used here is given by Plummer [RFC 826] and has become a TCP/IP internet protocol standard. Dalal and Printis [1981] describes the relationship between Ethernet and IP addresses, and Clark [RFC 814] discusses addresses and bindings in general. Parr [RFC 1029] discusses fault tolerant address resolution. Kirkpatrick and Recker [RFC 1166] specifies values used to identify network frames in the Internet Numbers document. Volume 2 of this text presents an example ARP implementation, and discusses the caching policy.

EXERCISES

5.1 Given a small set of physical addresses (positive integers), can you find a function *f* and an assignment of IP addresses such that *f* maps the IP addresses 1-to-1 onto the physical addresses and computing *f* is efficient? (Hint: look at the literature on perfect hashing).

5.2 In what special cases does a host connected to an Ethernet not need to use ARP or an ARP cache before transmitting an IP datagram?