

RARP: Reverse Address Resolution Protocol

When a system with a local disk is bootstrapped it normally obtains its IP address from a configuration file that is read from a disk file. But a system without a disk, such as an X terminal or a diskless workstation, needs some other way to obtain its IP address.

Each computer on a network has a unique hardware address, assigned by the manufacturer of the network interface. The principle of RARP is for the diskless workstation to read its unique hardware address from the interface card and send an RARP request (an RARP request broadcast in a frame over the network) asking for someone to reply with the diskless system's IP address (in an RARP reply).

The format of an RARP request is identical to an ARP (Address resolution protocol) request discussed in Lecture 2. The only differences are that the frame type (in the frame header) is $(8035)_{16}$ for an RARP request or reply, and the *operation* field in the Ethernet data portion has the value 3 for an RARP request and 4 for an RARP reply.

An RARP server attached to the physical network normally provides the mapping from a hardware address to an IP address for many hosts (the diskless systems on the network). The mapping is contained in a disk file (normally */etc/ethers* on Unix systems). The reply from the server is unicast (to the sender of the RARP request).

As with ARP requests, RARP requests are sent as hardware (Ethernet) level broadcasts. This means they are not forwarded by routers. Thus the server must be present on the same physical network as the diskless workstation. To allow diskless stations to bootstrap even when the main RARP server is down, multiple RARP servers are provided on a single network. As the number of servers increases, the network traffic increases, since every server sends an RARP reply to every RARP request. The diskless system normally uses the first RARP reply that it receives. Furthermore, there is a chance that each RARP server can try to respond at about the same time, increasing the probability of collisions on an Ethernet (This is not a problem with ARP, because only a single host sends an ARP reply).

Internet Protocols:

The protocol that defines the unreliable, connectionless, delivery mechanism is called the Internet Protocol (IP). IP provides three basic definitions:

1. The IP protocol defines the basic unit of data transfer for internet software.
2. It provides the *routing* function, i.e., choosing a path over which data is to be sent.
3. IP includes a set of rules that embody the ideas of unreliable packet delivery. The rules characterize how hosts and routers should process packets, how and when error messages should be generated, and the conditions under which packets can be discarded. These form part of the Internet Control Message (ICMP) protocol suite.

The IP service is connectionless because each packet is treated independently from all others. The service is unreliable because delivery is not guaranteed, and packets may well be lost, duplicated, delayed, or delivered out of order. Finally, the service is best-effort delivery because the software makes an earnest attempt to deliver packets; unreliability arises only when the resources are exhausted or the underlying networks fail. The key idea in IP is to keep the network relatively simple and put any necessary intelligence in the end hosts. The IP layer resembles the postal system. On top of the IP layer, the end hosts implement transport layer protocols that coordinate the end-to-end delivery of data between applications.

Internet Datagrams:

The basic transfer unit in the IP layer is called an *IP datagram*. Like a typical Ethernet frame discussed in Lecture 2, a datagram is divided into header and data areas. The IP datagram travels (encapsulated) in the data portion of an Ethernet frame. The typical datagram format appears in Figure 7.3 of Comer. The fields in the datagram header are typically set by the operating system on the sending machine. Although an IP router forwards a packet based on the destination address, the header includes additional fields that facilitate the successful communication between sender and receiver:

1. **Version number (4 bits):** The 4-bit version number typically has a value of 4. Knowing the version number enables the routers and the receiving host to interpret the rest of the header correctly. Any other version of the protocol, such as IPv6, may have a different format.
2. **Header length (4 bits):** The 4-bit header length indicates the number of 4-byte words in the header. The basic IP header is 20 bytes long (five 4-byte words), but longer headers are possible when *IP options* are used. The length of the IP header is always a multiple of 32 bits.
3. **Type of service (8 bits):** The 8 bits of service (TOS) were originally included in the IP header to influence the path the packet follows through the network. For example, paths may have different performance properties, such as low-delay, high-throughput, or high-reliability. A packet from a large file transfer might be forwarded on a high-throughput

path because file transfer delays depend on the bandwidth available between the sender and the receiver. A detailed description of the TOS fields appears in Section 7.7.2 of Comer.

4. **Total length (16 bits):** The 16-bit total length field indicates the total number of bytes in the packet. An IP packet can be up to 65536 bytes long. However, most link-layer technologies cannot handle such large packets, and therefore may impose a smaller *maximum transmission unit* (MTU). For example, many LANs use Ethernet, which has an MTU of 1500 bytes. The limitation on the MTU can be handled in two ways. In the first approach, the sending machine avoids sending IP packets that exceed the MTU of the links along the path to the receiving machine. But, typically the application does not know the MTU values in advance. In the second approach, an oversized IP packet is divided into two or more fragments by the sender or a router. The fragmentation of IP packets is supported by the next three fields in the IP header - the 16 bit identifier, the 3-bit flags, and the 13-bit offset.
5. **Identification (16 bits):** The 16-bit identifier field contains a unique value for each IP packet from the sender. Before forwarding a packet to an outgoing link, an IP router checks that the packet size does not exceed the link's MTU. A packet that is too large is fragmented. Each fragment of the packet has the same identifier. This enables the destination to recognize that the fragments belong together. The fragments are reassembled *only* at the destination, and a single IP packet is delivered to a higher-layer protocol such as TCP or UDP.
6. **IP flags (3 bits):** Two of the three 1-bit flags relate to the fragmentation process; the other bit is reserved for future use. The *more fragments* bit is set to 1 for all but the last fragment of a packet. This ensures that the destination can determine whether all of the fragments have arrived. Fragmentation can be disabled by setting the *don't fragment* bit. Upon receiving an oversized packet with the *don't fragment* bit set, the router discards the packet and notifies the sender. The notification is sent using the Internet Control Message Protocol (ICMP). The *don't fragment* bit plays an important role in learning the maximum packet size permitted by the routers along the path through the network. The sender can experiment by sending packets of various sizes with the *don't fragment* bit set to see whether any of the routers along the path generates an error. The communication software on most computers imposes a de facto MTU (576 or 1500 bytes) that is small enough to avoid fragmentation inside the network.
7. **Fragment offset (13 bits):** The 13 bit fragment offset field contains the offset (in 8-byte units) of the fragment from the beginning of the original IP packet. Using 8-byte units enables the 13 bit field to identify offsets within an IP packet that is up to $2^{16}(8 \times 2^{13})$ bytes long. Together, the offset and the length enable the destination to determine what range of bytes is covered by this fragment. Once all the fragments have arrived, the destination can reassemble the packet.
8. **Time to live (TTL) (8 bits):** The 8-bit TTL field limits the number of hops in the path traversed by the packet. The sender sets the TTL field

to an initial value. Then each router along the path decrements the field by one. When the TTL reaches 0, the packet is discarded, and an ICMP error message is sent to the sender. This procedure addresses a potential robustness problem in IP routing. Sometimes a misconfigured router or a link failure can cause a temporary routing loop. As a result, a packet may repeatedly travel over the same set of links, not making any progress toward the destination. The packet may eventually reach the destination long after the two end points have stopped communicating. The arrival of such a very old packet may confuse the recipient. Discarding packets with expired TTLs avoids this problem.

9. **Protocol (8 bits):** The 8-bit protocol field identifies the higher-level protocol (TCP, UDP, ICMP) responsible for sending the IP packet. Knowing the protocol is important for the recipient of the packet to interpret the data after the IP header.
10. **Header checksum (16 bits):** The 16-bit checksum provides a way to detect if any of the bits in the header is corrupted as the packet travels through the network. Before transmitting the packet, the sender computes the 16-bit sum of the bits in the IP header and includes the result in the header checksum field. An IP packet with an incorrect header checksum is discarded by the receiving machine. Routers along the path update the header checksum as part of forwarding an IP packet to account for any changes in the header, such as decrementing the TTL field. It is important to note that the checksum applies only to the IP header and not to the contents of the packet. IP does not check whether the data was corrupted. This is the task of a high-level protocol such as TCP or UDP.
11. *Source IP address (32 bits):* The IP header includes the 32 bit address that identifies the sender of the packet. This allows the recipient or an intermediate router to contact the sender. For example, the receiving application may want to send a reply message. In addition, an intermediate router may need to know the source address to send an ICMP error message.
12. **Destination IP address (32 bits):** The 32-bit destination address is necessary to identify the intended recipient of the packet. Upon receiving a packet, a router uses the destination IP address to determine the *next hop* en route to the destination.
13. **IP options (variable length):** The final field of the IP header contains a variable-length list of optional information, ending on a 32 bit boundary. IP options can be used to exercise additional control over routing or security or to have routers add a timestamp to the packet. A description of some options appears in Section 7.8 of Comer.

Internet Routing:

Routing refers to the process of choosing a path over which to send packets, and *router* refers to the computer making the choice. We distinguish hosts from routers; host computers can be configured to act as routers, but we will assume, unless otherwise specified, that host computers do not perform the router's function of transferring packets from one network to another. We can divide routing into two forms:

1. **Direct Delivery over a single network:** To see that a host computer lies on a directly connected network, the sender has simply to extract the network id portion of the destination IP address, and this is the same as the network portion of its own IP address. In this case, the sender simply encapsulates the datagram within an Ethernet frame, maps the destination IP address to a physical address (using the ARP protocol discussed in Lecture 2, and Chapter 5 of Comer), and uses the network hardware to deliver it. No router is involved in this case.
2. **Indirect Delivery:** This is more difficult than direct delivery because the sender must identify a router to which the datagram can be sent. In this case, the sender encapsulates the datagram in a physical frame, and sends it to the nearest (in most cases a default) router. The sender then invokes an ARP request to map the router IP address (not the destination IP address) to a physical address, and sends the input datagram to the router using the physical network. Once the frame reaches the router, the software extracts the IP datagram, and the router selects the next router (next hop) en route to the next destination. The datagram is again placed in a frame and sent over the next physical network to a second router, and so on, until it can be delivered directly. To select the next hop, the router consults its routing tables. Typically, a routing table consists of ordered pair (N,R), where N is the IP address of a destination network, and R is the IP address of the next router (next hop) along the path to network N. The routing table thus contains one step along the path from router R to the destination network, the router itself is unaware of the complete path to the destination. The routing tables contain the IP addresses of the destination network, and the individual host to keep the routing tables small, and to make routing efficient. Thus, all traffic destined for the same network takes the same path.

The IP routing algorithm:

This is the Algorithm in Figure 8.3 of Comer. For ease of exposition, I have a simplified version of this Algorithm below:

RouteDatagram(Datagram, RoutingTable)

1. Extract destination IP address, D, from the datagram, and compute the network prefix, N
2. **If** N matches the network prefix of the router IP address **then**
 deliver datagram to directly destination D over that network.
 else if N matches a network IP address specified in the table **then**
 send datagram to next-hop specified in table
 else if if the table contains a default route **then**
 send datagram to the default router specified in the table
 else declare a routing error
 end if

A few points regarding this algorithm are now in order:

1. In case the network prefix N matches the network portion of the IP address of the router, the datagram is sent directly to the destination in a

physical frame, after an ARP request resolves the physical address of the destination D.

2. In the 2nd and 3rd cases, the datagram is sent to a next hop router specified in the table. An ARP request resolves the physical address of this next hop router, and the datagram is forwarded to this router in a physical frame. At all times, the IP datagram contains only the IP address of the final destination. The IP address of the next-hop router is discarded once one has resolved the physical address of this router.
3. In the final case, the router sends an ICMP message to the original sender whose IP address is specified in the IP datagram.

Figure 8.2 in Comer illustrates an example on which you can test this routing algorithm.

Subnets:

Subnetting is a technique used to allow a single network address to span multiple physical networks. Instead of dividing a 32 bit IP address into a network prefix and a host suffix, subnetting divides the address into a network portion and a local portion. The interpretation of the network portion remains the same as for networks that do not use subnetting. This network portion needs to be advertised to the rest of the internet. However, the interpretation of the local portion of an address is left up to the site. The local portion is in turn divided into subnet and host id portions. The result is a form of hierarchical addressing that also leads to corresponding hierarchical routing.

As an example, consider McMaster University, and two departments at Mac, namely the Computing & Software (CAS) and ECE departments. Let the McMaster University physical network have the class B address 128.10.0.0 (the network id is the first 2 octets of the 32 bit IP address). Also, the CAS and ECE department physical networks have the IP addresses 128.10.1.0 and 128.10.2.0 (the subnet id here is the 3rd octet of the IP address) respectively. Finally, let's say Kartik's host machine *karmarkar.cas.mcmaster.ca* has the IP address 128.10.1.1 (the host id of Kartik's machine is contained in the last octet of the IP address).

A site using subnet addressing needs to choose a 32 bit *subnet mask* for its network. Bits in the subnet mask are set to 1 if machines on the network treat the corresponding bit in the IP address as part of the subnet prefix, and 0 if they treat the bit as part of the host identifier. A subnet mask also has 1's for all bits that correspond to the network portion of the address. For the McMaster University example above, the 32 bit mask is 255.255.255.0 in dotted decimal representation. Usually, subnet masks choose contiguous bits of the IP address.

Subnet routing algorithm:

The standard IP routing algorithm mentioned above must be modified to work with subnet addresses. All hosts and routers attached to a network that uses subnet addressing use a modified algorithm known as *subnet routing*.

In the standard routing algorithm it is possible to determine the net id and the host id of the IP address from the first 3 bits which encode the address type, i.e., class A, B or C. With subnets, it is not possible to decide which

bits correspond to the network and which to the host from the address alone. Instead, the modified algorithm maintains an additional field called subnet mask in the routing table. The entry in this field specifies the subnet mask used with the network in that entry.

The unified routing algorithm is given in Figure 10.8 of Comer. I present a simplified version of this algorithm below. For simplicity, I assume that there is no default entry in the table.

Route-IP-Datagram(Datagram, Routing-Table)

1. Extract destination IP address, D, from datagram.
2. Let N be the bitwise-and of D and the subnet mask associated with the router physical networks.
3. **If** N matches the network address to which the router attaches **then** send datagram directly to the destination over the network
else
for each entry in the routing table **do**
 Let N be the bitwise-and of D and the subnet mask
 if N matches the network address field of the entry **then**
 route the datagram to the specified next hop address
 end for
end if
4. Declare a routing error.

As an example of the subnet addressing algorithm consider the network on Figure 8.2 of Comer. However, let the network addresses of the four networks be 10.0.0.0, 20.0.0.0, 130.23.0.0 and 198.72.34.0. Let the IP addresses of the router Q be 10.0.0.5 and 20.0.0.5, those of router R be 20.0.0.6 and 130.23.0.6, and those of S 130.23.0.7 and 198.72.34.7 respectively. Also let the routing table for router R be Now let the destination IP address in the datagram be

Network Number	Subnet Mask	Next Hop
20.0.0.0	255.0.0.0	Deliver Directly
130.23.0.0	255.255.0.0	Deliver Directly
10.0.0.0	255.0.0.0	20.0.0.5
198.72.34.0	255.255.255.0	130.32.0.7

130.23.0.33. Router R consults its routing table, and finds that a *bitwise and* of 130.23.0.33 and 255.255.0.0 yields 130.23.0.0. Since the router R attaches directly to this physical network, this datagram is delivered directly.

Recommended Reading

1. Chapter 6 of Comer [1], and Chapter 5 of Stevens [2] for a discussion of the RARP protocol.
2. Chapter 7 of Comer and Chapter 3 of Stevens for a discussion of the IP protocol.

3. Chapter 8 of Comer for a discussion on the routing algorithm.
4. Chapter 10 of Comer, especially Section 10.6-10.10 for subnet addressing, and Sections 10.11-10.13 for the subnet routing algorithm. Also, Section 3.4 of Stevens for subnet addressing, and Chapter 9 of Stevens for the general routing algorithm.

References

- [1] D.E. COMER, *Internetworking with TCP/IP: Principles, Protocols, and Architectures*, 4th edition, Prentice Hall, NJ, 2000.
- [2] W. RICHARD STEVENS, *TCP/IP Illustrated, Volume I: The Protocols*, Addison Wesley Professional Computing Series, 1994.