Reverse Address Resolution Protocol (RARP)

Mr. Rahul Sehrawat

Department of Information Technology

Dronacharya College of Engineering , Gurgaon, Haryana

rsehrawat75@gmail.com

Mr. Pankaj Gupta

Department of Information Technology

Dronacharya College of Engineering , Gurgaon, Haryana

im4upankaj@gmail.com

Mr. Ravishu Yadav

Department of Information Technology

Dronacharya College of Engineering , Gurgaon, Haryana

yadavravishu@gmail.com

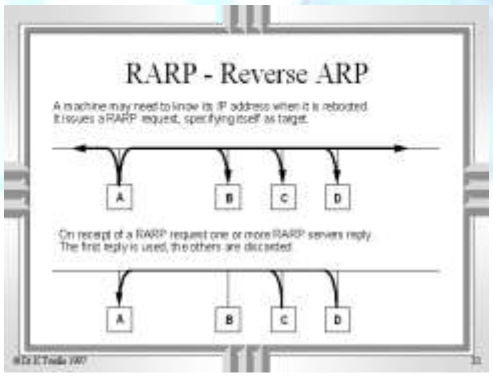
Abstract-- The Reverse Address Resolution Protocol (RARP) is an obsolete computer networking protocol used by a client computer to request its Internet Protocol (IPv4) address from a computer network, when all it has available is its Link Layer or hardware address, such as a MAC address. The client broadcasts the request, and does not need prior knowledge of the network topology or the identities of servers capable of fulfilling its request. RARP is described in Internet Engineering Task Force (IETF) publication RFC 903. It has been rendered obsolete by the Bootstrap Protocol (BOOTP) and the modern Dynamic Host Configuration Protocol (DHCP), which both support a much greater feature set than RARP. This research paper gives a brief introduction about Reverse Address Resolution Protocol (RARP) , Timing RARP Transactions ,Primary And Backup RARP Servers.

Introduction

We now know that physical network addresses are both low-level and hardware dependent, and we understand that each machine using TCP/IP is assigned one or more 32-bit IP addresses that are independent of the machine's hardware addresses. Application programs always use the IP address when specifying a destination. Because hosts and routers must use a physical address to transmit a datagram across an underlying hardware network; they rely on address resolution schemes like ARP to map between an IP address and an equivalent hardware address. Usually, a computer's IP address is kept on its secondary storage, where the operating system finds it at startup. The question arises, "How does a machine without a permanently attached disk determine its IP address?" The problem is critical for workstations that store files on a remote server or for small embedded systems because such machines need an IP address before they can use standard TCP m file transfer protocols to obtain their initial boot image. This paper explores the question of how to obtain an IP address, and describes a low-level protocol that such machines can use before they boot from a remote file server. This extends the discussion of bootstrapping, and considers popular alternatives to the protocol presented here. Because an operating system image that has a specific IP address bound into the code cannot be used on multiple computers, designers usually try to avoid compiling a machine's IP address in the operating system code or support software. In particular, the bootstrap code often found in Read Only Memory (ROM) is usually built so the same image can run on many machines. When such code starts execution, it uses the network to contact a server and obtain the computer's IP address. The bootstrap procedure sounds paradoxical: a machine communicates with a re- mote server to obtain an address needed for communication. The paradox is only imagined, however, because the machine does know how to communicate. It can use its physical address to communicate over a single network. Thus, the machine must resort to physical network addressing temporarily in the same way that operating systems use physical memory addressing to set up page tables for virtual addressing. Once a machine knows its IP address, it can communicate across an internet. The idea behind finding an IP address is simple: a machine that needs to know its address sends a request to a server? on another machine, and waits until the server sends a response. We assume the server has access to a disk where it keeps a database of internet addresses. In the request, the machine that needs to know its internet address must uniquely identify itself, so the server can look up the correct internet address and send a reply. Both the machine that issues the request and the server that responds use physical network addresses during their brief communication. How does the requester know the physical address of a server? Usually, it does not - it simply broadcasts the request to all machines on the local network. One or more servers respond. Whenever a machine broadcasts a request for an address, it must uniquely identify itself. What information can be included in its request that will uniquely identify the machine? Any unique hardware identification suffices (e.g., the CPU serial number). However, the identification should be something that an executing program can obtain easily. Unfortunately, the length or format of CPU-specific information may vary among processor models, and we would like to devise a server that accepts requests from all machines on the physical network using a single format. Furthermore, engineers who design bootstrap code attempt to create a single software image that can execute on an arbitrary processor, and each processor model may have a slightly different set of instructions for obtaining a serial number.

Reverse Address Resolution Protocol (RARP)

The designers of TCP/IP protocols realized that there is another piece of uniquely identifying information readily available, namely, the machine's physical network ad- dress. Using the physical address as a unique identification has two advantages. Because a host obtains its physical addresses from the network interface hardware, such addresses are always available and do not have to be bound into the bootstrap code. Because the identifying information depends on the network and not on the CPU vendor or model all machines on a given network will supply uniform unique identifiers. Thus, the problem becomes the reverse of address resolution: given a physical network address, devise a scheme that will allow a server to map it into an internet address. The TCP n P protocol that allows a computer to obtain its IP address from a server is known as the Reverse Address Resolution Protocol (RARP). In practice, the RARP message sent to request an internet address is a little more general than what we have outlined above: it allows a machine to request the IP address of a third party as easily as its own. It also allows for multiple physical net- work types. Like an ARP message, a RARP message is sent from one machine to another en- capsulated in the data portion of a network frame. For example, an Ethernet frame carrying a RARP request has the usual preamble, Ethernet source and destination ad- dresses, and packet type fields in front of the frame. The frame type contains the value 8035,, to identify the contents of the frame as a RARP message. The data portion of the frame contains the 28-octet RARP message. Figure 1.1 illustrates how a host uses RARP. The sender broadcasts a RARP re- quest that specifies itself as both the sender and target machine, and supplies its physical network address in the target hardware address field. All computers on the network receive the request, but only those authorized to supply the RARP service process the request and send a reply; such computers are known informally as RARP servers. For RARP to succeed, the network must contain at least one RARP server.



RARP - Reverse ARP

A machine may need to know its IP address when it is rebooted. It issues a RARP request, specifying itself as target.

On receipt of a RARP request one or more RARP servers reply. The first reply is used, the others are discarded.

Fig1.1 (a) Machine A broadcasts a RARP request specifying itself as a target, and (b) those machines authorized to supply the RAW service (C and D) reply directly to A.

Servers answer requests by filling in the target protocol address field, changing the message type from request to reply, and sending the reply back directly to the machine making the request. The original machine receives replies from all RARP servers, even though only the first is needed. Keep in mind that all communication between the computer seeking its IP address and the server supplying it must be carried out using only the physical network. Furthermore, the protocol allows a host to ask about an arbitrary target. Thus, the sender supplies its hardware address separate from the target hardware address, and the server is careful to send the reply to the sender's hardware address. On an Ethernet, having a field for the sender's hardware address may seem redundant because the information is also contained in the Ethernet frame header. However, not all Ethernet hardware provides the operating system with access to the physical frame header.

Timing RARP Transactions

Like any communication on a best-effort delivery network, RARP requests and responses are susceptible to loss (including discard by the network interface if the CRC indicates that the frame was corrupted). Because RARP uses the physical network directly, no other protocol software will time the response or retransmit the request; RARP software must handle these tasks. In general, RARP is used only on local area networks like the Ethernet, where the probability of failure is low. If a network has only one RARP server, however, that machine may not be able to handle the load, so packets may be dropped. Some computers that rely on RARP to boot choose to retry indefinitely until they receive a response. Other implementations announce failure after only a few tries to avoid flooding the network with unnecessary broadcast traffic (e.g., in case the server is unavailable). On an Ethernet, network failure is less likely than server overload. Making RARP software retransmit quickly may have the unwanted effect of flooding a congested server with more traffic. Using a large delay ensures that servers have ample time to satisfy the request and return an answer.

Primary and Backup RARP Servers

The chief advantage of having several computers function as RARP servers is that it makes the system more reliable. If one server is down or too heavily loaded to respond, another answers the request. Thus, it is highly likely that the service will be available. The chief disadvantage of using many servers is that when a machine broad- casts a RARP request, the network becomes overloaded because all servers attempt to respond. On an Ethernet, for example, using multiple RARP servers makes the probability of collision high. How can the RAW service be arranged to keep it available and reliable without incurring the cost of multiple, simultaneous replies? There are at least two possibilities, and they both involve delaying responses. In the first solution, each machine that makes RARP requests is assigned a primary server. Under normal circumstances, only the machine's primary server responds to its RARP request. All non-primary servers receive the request but merely record its arrival time. If the primary server is unavailable the original machine will timeout waiting for a response and then rebroadcast the re- quest. Whenever a non-primary server receives a second copy of a RARP request within a short time of the fist, it responds. The second solution uses a similar scheme but attempts to avoid having all non-primary servers transmit responses simultaneously. Each non-primary machine that receives a request computes a random delay and then sends a response. Under normal circumstances, the primary server responds immediately and successive responses are delayed, so there is low probability that several responses arrive at the same time. When the primary server is unavailable, the requesting machine experiences a small de-lay before receiving a reply. By choosing delays carefully, the designer can ensure that requesting machines do not rebroadcast before they receive an answer.

Conclusion

At system startup, a computer that does not have permanent storage must contact a server to find its IP address before it can communicate using TCP/IP. This chapter examined the RARP protocol that uses physical network addressing to obtain the machine's internet address. The RARP mechanism supplies the target machine's physical hardware address to uniquely identify the processor and broadcasts the RARP request. Servers on the network receive the message, look up the mapping in a table (presumably from secondary storage), and reply to the sender. Once a machine obtains its IP address, it stores the address in memory and does not use RARP again until it reboots.

REFERENCES

[1]Finlayson, et. al. [RFC 9031] [2] internetworking with TCP/IP vol1 Douglas E. Comer [89-93][3] TCP/IP The protocols W. Richard Stevens.[4] Dr. K. Twidle