ARP Request and Reply

Documentation

Prepared by Priya Chawla, Jeff Nainaparampil, Jacob Chesley • November 26, 2014

Table of Contents

1. Introduction

1.1. What is ARP?

ARP stands for Address Resolution Protocol. In order for someone to ping an IP address to their local network, the system will need to convert an IP address into a MAC address. In order to do this, the user will need to use ARP to resolve the address.

The system stores information about what IP addresses are associated with MAC addresses in an ARP look-up table. In order to send a packet to an IP address, the system will need to use the ARP look-up table to see if it has the MAC address stored. If the value is already cached, it is not used.

Next, the system will send a broadcast packet to the network using the ARP protocol to ask who has the correct IP address. Since a broadcast packet is used, it is sent to a special MAC address that all network machines receive. Thus, any machine that has the requested IP address will reply with an ARP packet, claiming it is the IP address. This also includes the MAC address that can receive packets for that particular IP address.

2. How to implement ARP?

2.1. Step-by-Step Overview

Setup step:

Include proper headers built in linux:

```
#include <linux/if_arp.h>
#include <net/arp.h>
```

1. **Inside of the net header operations detect ARP protocol for socket buffer**:

```
struct header_ops net_header_operations = { .create = header,
};

if (skb->protocol == htons (ETH_P_ARP)) {

            memcpy(eth->h_dest, eth->h_source, dev->addr_len);
            memcpy(eth->h_source, dev->dev_addr, ETH_ALEN);
            return dev->hard_header_len;
}
```

2. **Inside of the start transmit function, look for the ARP protocol:**

```
struct arphdr *arp_header; //points to where arp header is

// check for ARP protocol, to see if skb is a type of ARP.
```

3. **Create header from socket buffer:**

```
arp_header = arp_hdr(skb);
```

4. **Once ARP request is detected, switch it to an ARP reply:**

```
arp_header->ar_op = htons(ARPOP_REPLY)
```

5. **Flip source and destination address. Using this table we calculated the sender and target hardware and protocol addresses:**

Internet Protocol (IPv4) over Ethernet ARP packet		
octet offset	**0**	**1**
0	Hardware type (HTYPE)	
2	Protocol type (PTYPE)	
4	Hardware address length (HLEN)	Protocol address length (PLEN)
6	Operation (OPER)	
8	Sender hardware address (SHA) (first 2 bytes)	
10	(next 2 bytes)	
12	(last 2 bytes)	
14	Sender protocol address (SPA) (first 2 bytes)	
16	(last 2 bytes)	
18	Target hardware address (THA) (first 2 bytes)	
20	(next 2 bytes)	
22	(last 2 bytes)	
24	Target protocol address (TPA) (first 2 bytes)	
26	(last 2 bytes)	

Source: http://en.wikipedia.org/wiki/Address_Resolution_Protocol

Sender and target hardware and protocol addresses equal 10 bytes. 6 for Hardware, 4 for protocol. The reason we use 18 for one of them and 8 for the other is because 18 bytes - 8 bytes is 10 bytes..

```
#define ARP_THA_OFFSET 18
#define ARP_SHA_OFFSET 8

char tmp[10]

memcpy(tmp, (void *)arp_header + ARP_SHA_OFFSET, 10);
memcpy((void *)arp_header + ARP_SHA_OFFSET,
(void*)arp_header + ARP_THA_OFFSET, 10);
memcpy((void *)arp_header + ARP_THA_OFFSET, tmp, 10);
```

6. **The last step is to set the ethernet address to the device address. **Ensure the rest of the socket buffer is set properly for:**

```
memcpy(eth->h_dest, dev->dev_addr, dev->addr_len);

//make sure the line below is implemented, otherwise it will
cause kernel panic
skb->dev = dev;

skb->protocol = eth_type_trans(skb, dev)
ih->check = 0;
ih->check = ip_fast_csum((unsigned char *)ih, ih->ihl);
```

The lines below send the socket buffer to the upper layers of the operating system. This is exactly like the receive handler.

```
netif_rx(skb);
return NETDEV_TX_OK;

// END the arp protocol check here with } (#2 if statement)
```

2.2. Application to the Course

1. ARP Request and Reply applies to *Lab 7: Net Device Driver*.
    a. remove os1 flag and os0 flag, that defines the NOARP:

```
// Comment these out
os0->flags |= IFF_NOARP;
os1->flags |= IFF_NOARP;
```

2. The implementation was completed by Jacob Chesley, a 4th year student in the course during the Fall of 2014.
3. The source code he implemented is archived by Professor Franco.

3. References

http://searchnetworking.techtarget.com/definition/Address-Resolution-Protocol-ARP

http://www.tummy.com/articles/networking-basics-how-arp-works/

http://en.wikipedia.org/wiki/Address_Resolution_Protocol