LCA 2006

Layer 2, Mac Addresses, ARP, and Duplicate IP Detection

http://marc.merlins.org/linux/talks/Layer2&ARP/

Marc MERLIN
marc_soft@merlins.org

Shoo, you don't want to be here :)

- The talk is about a kernel patch, but I'm planning to explain how the basic IP/networking pieces actually work instead of diving into kernel internals
- I will throw stuff at you
- The other speakers are most likely smarter than me
- Or you could stay here to make correct me, make fun of me, and if you stay until the end, tell me that my patch sucks :)
- Or you will also know to answer two interview questions I might ask you if you apply for a job at google.

What is Layer 2?

- It's above the physical layer (10b2/10bT/100bT/1000bT), FDDI, ATM, etc..)
- It is below IP, Appletalk, Decnet, IPX
- Token Ring is one example, as is carrier pigeon, but most of us use Ethernet now
- IEEE 802.3 replaces the original ethernet
- Can be more complex (VLANs/802.1q / 802.11 ...)
- source MAC / dest MAC / protocol type
- See /etc/protocols (ICMP:1, TCP: 6, UDP: 17)

Switching Ethernet

- A switch is a multi port bridge, it takes an incoming packet, and looks at the destination MAC address
- It decides what port to send the traffic to by looking at its CAM table (MAC to port # mapping)
- A switch does **NOT** do ARP to route ethernet frames
- A layer 2 switch does not even know what TCP/IP or ARP are. If it did, how would it route non IP traffic?
- Ethernet and TCP/IP were not designed with switches in mind, switches are supposed to be transparent. This means they do switching with no protocol support

Problems with Ethernet Switching

- How is the CAM table built?
- What if the destination MAC is not in the CAM table?
- How big do you think the CAM table is?
- What if it overflows? → you turn the water off :)
- What if a MAC address shows up on 2 ports?

ARP

- ARP is the address resolution protocol for IP
- Before you can send an IP packet, you need to build a frame with a destination ethernet address. If you don't know what the destination MAC address is, you use ARP to get it
- You broadcast a packet at layer 2 asking who has a certain IP, and what MAC address it belongs to
- The owner of the IP (if present) sends back a unicast packet to you with the answer (this means only you get the result of that arp query)

ARP

```
gandalf:~# arp 10.2.251.248
Address                  HWtype  HWaddress           Flags Mask            Iface
10.2.251.248             ether   00:11:25:84:EF:AD   C                     eth0
gandalf:~# arp -d 10.2.251.248
gandalf:~# arp 10.2.251.248
Address                  HWtype  HWaddress           Flags Mask            Iface
10.2.251.248                     (incomplete)                              eth0
gandalf:~# ping -c 1 10.2.251.248
PING 10.2.251.248 (10.2.251.248) 56(84) bytes of data.
64 bytes from 10.2.251.248: icmp_seq=1 ttl=64 time=0.838 ms

--- 10.2.251.248 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.838/0.838/0.838/0.000 ms
gandalf:~# arp 10.2.251.248
Address                  HWtype  HWaddress           Flags Mask            Iface
10.2.251.248             ether   00:11:25:84:EF:AD   C                     eth0

0:d:60:b1:d5:d4 ff:ff:ff:ff:ff:ff 0806 42: arp who-has 10.2.251.248 tell 10.2.251.242
0:11:25:84:ef:ad 0:d:60:b1:d5:d4 0806 60: arp reply 10.2.251.248 is-at 0:11:25:84:ef:ad
0:11:25:84:ef:ad 0:d:60:b1:d5:d4 0806 60: arp who-has 10.2.251.242 tell 10.2.251.248
0:d:60:b1:d5:d4 0:11:25:84:ef:ad 0806 42: arp reply 10.2.251.242 is-at 0:d:60:b1:d5:d4
```

RARP

- RARP is reverse ARP
- It was used among other things by diskless workstations to request their IP address: they would send a RARP packet with their MAC address, and expect an IP in return (from a server with a static entry in /etc/ethers and running rarpd)
- RARP is effectively obsolete and been replaced by dhcpd

MAC conflicts

- In theory, all ethernet interfaces have a different MAC address, it's guaranteed to be unique
- In theory, theory and practice are the same
- In practice, they're not :)
- Some older computers with built in ethernet (Sun and Silicon Graphics for instance) had the MAC address set at the factory on the mainboard, and sometimes there were mistakes and dupes
- Someone could have changed their MAC address to match yours, maybe with evil intentions

IP conflicts

- In theory (too) IPs are set to be unique, except:
  - Some users like to play sysadmins, and you will eventually find yourself talking to them with a baseball bat, explaining why it's bad to take over the mail server or gateway IP :)
  - DHCP hands out unique IPs until you get a subtle failure with failover and dynamic hosts
  - A thought dead host with a reallocated IP comes back to life and gets plugged in
  - A backup gets restored on more than one machine, with the original IP, or along with the backed up host

Avoiding IP conflicts

- Just use DHCP everywhere, optionally with pseudo-static leases set by MAC address. You will have to renumber your hosts one day, and you'll thank yourself.
- In almost all cases, DHCP is good about not allocating the same IP twice, it pings one before giving it out, in case it was in use, but not in its own pool
- DHCP failover: serve two ranges (active-active) or trust the failover solution (active-passive)
- Check if your IP is in use before using it with arping. Some distros (like RH) will ARP for your IP and wait for no reply before using it themselves (the original appletalk had this builtin in the protocol)

When all else fails: detecting conflicts

- So, you should now be immune to IP conflicts, right? Mmmh, for the most part, just like you should never see two hosts with the same MAC address.
- But what if you do get a conflict?
- How do switches fill their CAM table? They notice MAC/port pairings on inbound packets and save them
- For TCP/IP, the first ARP request happens to send an IP/MAC combo for the sender, and it is conveniently broadcast to the entire layer 2.
- So, all hosts can take note of that IP/MAC mapping, and check if the IP or MAC address happens to match their own

Implementation Details

- If both the IP and MAC address in the packet match yours, it is most likely a layer 2 loop that reflects your traffic back to you, so we ignore these
- Where can this be done?
- For me, the simplest place was in the kernel, where ARP packets get already processed
- You can do that in userland too, as my patch was rejected mostly on the grounds that since it can be done in userspace, it shouldn't be in the kernel.
- It would be nice in the kernel though, as it is small, and helps everyone, on all distros, to get that extra protection
- Additional kernel load would be inferior to a process getting all traffic through a raw socket, and analyzing ARPs

Who else does this?

- I obviously wasn't the first to come up with this idea: at least Windows, Irix, and Solaris do this too
- Some, like Irix, will send a gratuitous ARP reply to the other host with the same IP so as to give a clue to the other machine that there is an IP conflict between you and it
- Most hosts syslog the problem, and display a message on the console, or even on the graphical terminal since it's a problem to take care of right away
- If I recall correctly, windows will even disable its interface when it sees a conflict. Use this feature as you see fit :)

Kernel Patch

- http://marc.merlins.org/linux/arppatch/
- Basically, we hook in the ARP processing code, and if we receive a broadcast ARP request, we look at the from MAC address and IP, and compare them to ours
- If they match, we syslog a relevant message

```
gandalf:~# ifconfig eth0 10.2.250.252 netmask 255.255.0.0 broadcast 10.2.255.255
gandalf:~# tcpdump -e -n -i eth0 ether host 0:d:93:34:a1:38 or host 10.2.250.252
23:50:38.047488 0:d:93:34:a1:38 ff:ff:ff:ff:ff:ff 0806 60: arp who-has 10.2.251.254 tell
10.2.250.252

Jan 25 23:50:38 gandalf kernel: Uh Oh, MAC address 00:0D:93:34:A1:38 claims to have our
IP address (10.2.250.252) (duplicate IP conflict likely)
```

Gratuitous ARP/IP takeover

- In an attempt to be unobtrusive, the patch doesn't send gratuitous ARPs, as it doesn't generate any traffic (by default, without extra checks, it would generate a gratuitous ARP loop between two machines fighting for the same IP anyway)
- However, sometimes you want to take over another machine's IP (IP takeover). This is where tools like fake and send_arp come in (courtesy of our Oz friend Horms)
- While there are usually few legit reasons to do so, you can also receive another machine's traffic by taking over its MAC address (just send outgoing packets with its MAC and most switches will send you traffic for it later)

What's missing / TODO

- Get some version of this patch in the kernel :)
- Alternatively, write a user space version, and get most distros to pick it up and run it by default (harder, and will generate a litte more load)

Questions

Are you a good sysadmin,
or programmer?

Need a job?

(Silicon Valley, Sydney, Santa Monica, Zürich,
New York, Dublin, Tokyo, India, and others)

(4 of my coworkers came from LCA 2005)

Email: marc_jo@merlins.org