ARP - Address Resolution Protocol

TCP/IP class

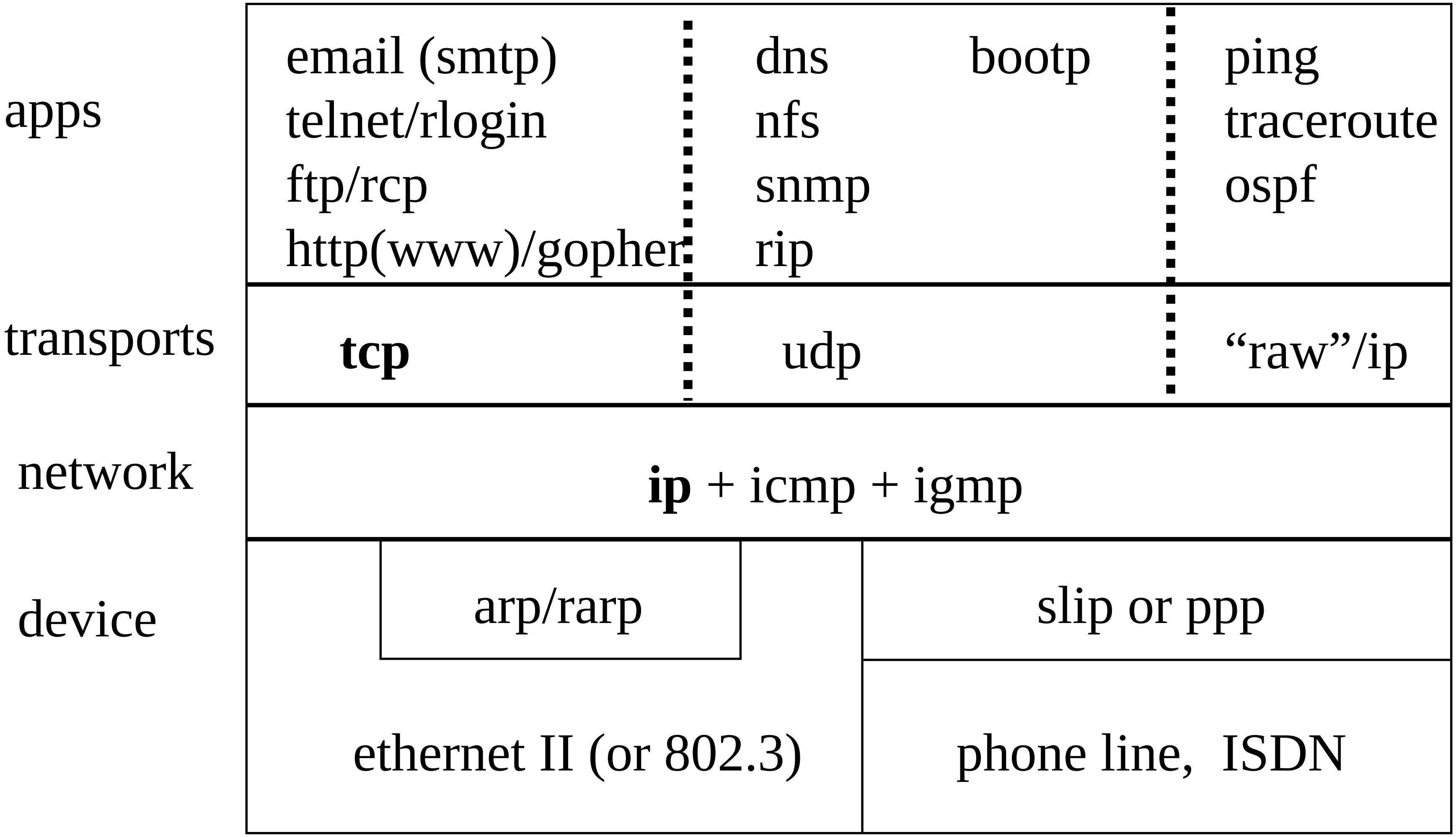
outline

- what’s the problem
- how does it work?
- format/arp dump
- variations
  - proxy arp
  - gratuitous arp
- study questions

arp - the problem?

- problem: **how does ip address get mapped to ethernet address?**
- 2 machines on same enet can only communicate if they know MAC/hw addr
- solutions:
  - configure addresses by hand (ouch!)
  - encode in IP address (48 bits in 32?)
  - use broadcast

Internet Protocols



apps

email (smtp)
telnet/rlogin
ftp/rcp
http(www)/gopher

dns
nfs
snmp
rip

bootp

ping
traceroute
ospf

transports

tcp

udp

“raw”/ip

network

ip + icmp + igmp

device

arp/rarp

ethernet II (or 802.3)

slip or ppp

phone line, ISDN

solution - arp protocol

- rfc 826
- host A, wants to resolve IP addr B,
  - send BROADCAST arp request
  - get UNICAST arp reply from B
- same link only
- ethernet (or MAC) specific, although protocol designed to be extensible
- implemented in driver, not IP

% *arp -a* (SunOs)

```
# arp -a
banshee.cs.pdx.edu (131.252.20.128) at   0:0:a7:0:2d:a0
pdx-gwy.cs.pdx.edu (131.252.20.1) at     0:0:c:0:f9:17
longshot.cs.pdx.edu (131.252.20.129) at  8:0:11:1:44:68
walt-suncs.cs.pdx.edu (131.252.21.2) at  8:0:20:e:21:25
walt-cs.cs.pdx.edu (131.252.20.2) at     8:0:20:e:21:25
connor.cs.pdx.edu (131.252.21.179) at    0:0:c0:c5:57:10
dazzler.cs.pdx.edu (131.252.21.132) at   8:0:11:1:12:82
sprite.cs.pdx.edu (131.252.21.133) at    8:0:11:1:12:e7
```

(DNS name,ip address,Ethernet address)

arp command functions

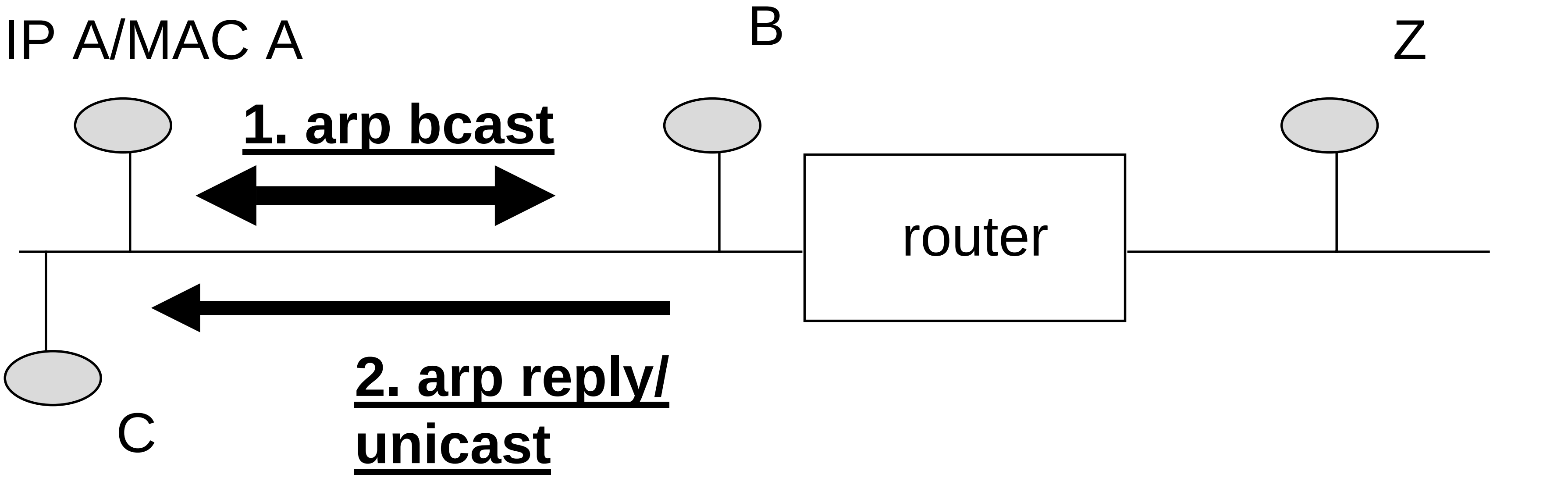
- ping someone and learn MAC address
- debugging
- delete out of date ARP entry (you changed the IP address, and you don’t want to wait, OR somebody mucked up)

refinements

- o.s. will cache arp replies in **arp cache**
  (ip , MAC, 20 minute timeout)
  - don’t need to do arp on every packet
- machine may store all arp broadcast to get sender ip/mac mapping
- recv. machines can update their cache
- why not broadcast all packets and skip arp? :->

arp protocol

1. A to B, arp request/broadcast on link
2. B to A, arp reply/unicast



ARP packet header format

<table>
<tr><th>0</th><th></th><th>16   31</th></tr>
<tr><td colspan="2">Hardware Type (1 byte)</td><td>Protocol Type (1 byte)</td></tr>
<tr><td>HLEN</td><td>PLEN</td><td>ARP Operation Code</td></tr>
<tr><td colspan="3">Sender HA (MAC) (bytes 0-3)</td></tr>
<tr><td colspan="2">Sender HA (bytes 4-5)</td><td>Sender IP Addr (0-1)</td></tr>
<tr><td colspan="2">Sender IP (2-3)</td><td>Target HA (0-1)</td></tr>
<tr><td colspan="3">Target HA (MAC) (bytes 2-5)</td></tr>
<tr><td colspan="3">Target IP Address (4 bytes)</td></tr>
</table>

header format details

- header format is not fixed, somewhat dynamic (not used though)
- hw type, ethernet == 1
- protocol type, ip = 0x800
- hwlen, 6 (MAC), plen 4 (ip)
- operation: (used by rarp too)
  - 1: arp request, 2: arp reply
  - 3: rarp request, 4: rarp reply

more details

- ◆ sender hw addr, 6 bytes
  - – the answer, if reply
- ◆ sender ip: 4 bytes
- ◆ target hw address: 6 bytes
  - – 0 in request
- ◆ target ip: 4 bytes

Arp dump - echo request/reply

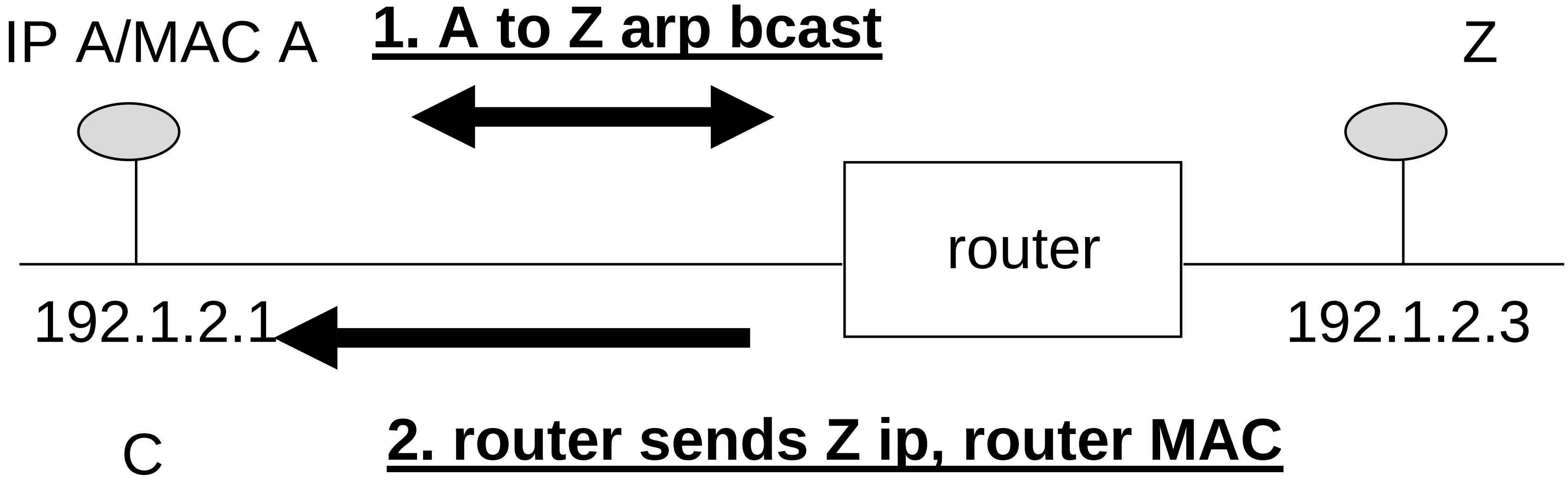
```
% etherfind -x -between bob ray arp

   60  arp              bob         ray
ff ff ff ff ff ff 00 80 f9 96 90 00 08 06
00 01 08 00 06 04 00 01 00 80 f9 96 90 00 8f b9
06 57 00 00 00 00 00 00 8f b9 06 01 00 00 00 00
00 ...

   60  arp         ray              bob
00 80 f9 96 90 00 00 00 3c 00 19 56 08 06
00 01 08 00 06 04 00 02 00 00 3c 00 19 56 8f b9
06 01 00 80 f9 96 90 00 8f b9 06 57 00 00 00 00
00 ...
```

proxy arp

- basic idea: machine A answers requests for machine B (that can’t arp for some reason), forwards packets to B somehow
  - machine A might have 2 IP addresses associated with one interface

proxy arp diagram

- if A to Z, router answers for Z



IP A/MAC A

1. A to Z arp bcast

Z

router

192.1.2.1

192.1.2.3

C

2. router sends Z ip, router MAC

proxy arp pros/cons

- pros
  - same network numbers
  - can aid dumb host that can’t arp
  - remote serial host appears on same ethernet courtesy of terminal emulator/router
- cons
  - can drive you nuts -- debugging
  - not simple and not secure

gratuitous/promiscuous arp

- **grat arp -** at boot or change of ip address, issue broadcast arp request for YOURSELF
  - unix ifconfig does this
  - detect other boxes with same IP address
  - allow recv boxes to cache your MAC addr
- **promiscuous arp -** issue bcast arp to change other’s ideas of ip/mac mapping
  - **problem: no one guaranteed to be listening**

study questions

- in the picture of the arp protocol example, assuming A requests B’s address, what roles do: 1. the router, 2. Z, 3. C play?
- if A was to telnet to Z, what arp packets would appear on the 2 links (assume ethernet)?
- is arp a link layer protocol? if so, how do you explain its encapsulation (at the same level with ip...)

even more study questions

- could arp be used to allow an attacking machine to masquerade as a attacked machine on a net? think about proxy arp too
- if two boxes with different IP network numbers are on the same net, can they ARP for each other?