The World According to Internet

Some Terminology

internet: collection of packet switching networks interconnected by routers

(the) Internet: “public” interconnection of networks

definition = host: computer that is attached to the network ↔ router; usually one network interface

router = gateway = intermediate system: routes packets, several interfaces

subnetwork: part of an internet (e.g., single Ethernet)

firewall: router placed between an organization’s internal internet and a connection to the external Internet, restricting packet flows to provide security.
Protocols

- rules by which active network elements communicate with each other is a protocol
- protocols = “algorithms + data structures”
  - formats of messages exchanged
  - actions taken on receipt of messages
  - how to handle errors
- hardware/operating-system independent
- real-life examples:
  - Robert’s rules for meetings
  - conversational rules (interrupts, request for retransmission, …)

What Do Protocols Do for a Living?

error control: make channel more reliable ➞ retransmission
resequencing: reorder out-of-sequence messages
flow control: avoid flooding slower receiver
congestion control: avoid flooding slower network
fragmentation: divide large message into smaller chunks to fit lower layer
multiplexing: combine several higher-layer sessions into one “channel”
addressing/naming: manage identifiers
compression: reduce data rate
privacy, authentication: even if somebody else is listening
resource allocation: bandwidth, buffers among contenders
Protocol Layering

**send side** layer $N$ takes protocol data (PDU) from layer $N + 1$, adds header, and passed to $N - 1$

**receive side** layer $N$ takes PDU from $N - 1$, strips $N$ headers, processes, and passes rest to $N + 1$

**TH (transport header):** sequence numbers, error detection, timestamp information ➞ end-to-end

**NH (network header):** source and destination address, hop counts

**L.Hn (link header):** error detection, hop-by-hop error control
Routers and Bridges

**host**: all layers

**router**: modifies data link headers/address, may touch network headers (IP options!)

**bridge**: may modify data link header

**repeater**: physical layer

IP packet maintains same source and destination addresses end-to-end, but gets many different link headers/trailers and addresses

---

Layering Considered Harmful?

- need layers to manage complexity ➔ don’t want to reinvent Ethernet-specific protocol for each application
- common functionality ➔ “ideal” network

but:

- layer $N$ may duplicate lower layer functionality (error recovery)
- different layers may need same information
- layer $N$ may need to peek into layer $N - 2$ (e.g., fragmentation)
- implementation issues: avoid copying
Internet View of the World

“anything over IP, IP over anything”

- subnetwork: ATM, Ethernet, ISDN (with PPP, SLIP)
- network layer: IP, IPng, (CLNP?)
- transport: UDP, TCP, ...
- application: http, ftp, telnet, RTP, ...
- control: RSVP
- management: SNMP
- directory: DNS

no session, presentation; also: OSI, AppleTalk over IP

---

Subnetwork Technologies

Some examples:

<table>
<thead>
<tr>
<th>technology</th>
<th>bandwidth</th>
<th>WAN, LAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATM</td>
<td>25 Mb/s … 2.4 Gb/s</td>
<td>WAN</td>
</tr>
<tr>
<td>leased line</td>
<td>56 kb/s, 1.5 Mb/s (T1), 2.0 Mb/s (E1)</td>
<td>WAN</td>
</tr>
<tr>
<td>satellite</td>
<td>2.4 kb/s … Mb/s</td>
<td>WAN</td>
</tr>
<tr>
<td>Ethernet</td>
<td>10 Mb/s, 100 Mb/s</td>
<td>LAN</td>
</tr>
<tr>
<td>Tokenring</td>
<td>4 Mb/s, 10 Mb/s</td>
<td>LAN</td>
</tr>
<tr>
<td>ISDN</td>
<td>64 kb/s</td>
<td>LAN</td>
</tr>
<tr>
<td>POTS modem</td>
<td>2.4 … 28.8 kb/s</td>
<td>LAN</td>
</tr>
</tbody>
</table>
### Refresher: Ethernet

- multiple access network
- 10 Mb/s “raw” speed (new: 100 Mb/s *Fast Ethernet*)
- media: coaxial cable, fiber, UTP-5 (unshielded twisted pair)
- cocktail party protocol: listen, transmit, back off

```plaintext
L = 1
start:
    if (nobody else transmitting)
        transmit
        if (collision detected)
            stop transmission immediately
        L *= 2
        wait random period of time (0, L) minislots; goto start
    else
        wait random period of time; goto start
```
**Ethernet Packet**

**preamble:** 7 bytes for clock synchronization

**length/frame type:** 2 bytes, < 1500; or IPv4: 0x0800; ARP: 0x0806

<table>
<thead>
<tr>
<th>7</th>
<th>1</th>
<th>6</th>
<th>6</th>
<th>2</th>
<th>46..1500</th>
<th>4 octets</th>
</tr>
</thead>
<tbody>
<tr>
<td>preamble</td>
<td>destination</td>
<td>source</td>
<td>data</td>
<td>pad</td>
<td>CRC (checksum)</td>
<td></td>
</tr>
<tr>
<td>7 x 10101010</td>
<td>MAC address</td>
<td>MAC address</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Names and Addresses**
Names, Addresses, Routes

Shoch (1979):

**Name** identifies what you want,

**Address** identifies where it is,

**Route** identifies a way to get there.

Saltzer (1982):

**Service and users**: time of day, routing, …

**Nodes**: end systems and routers

**Network attachment point**: ≥ 1 per node ➔ multihomed host vs. router

**Paths**: traversal of nodes and links

binding = (temporary) equivalence of two names

---

### Internet Names and Addresses

<table>
<thead>
<tr>
<th>example</th>
<th>organization</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAC address</td>
<td>8:0:20:72:93:18</td>
</tr>
<tr>
<td>IP address</td>
<td>132.151.1.35</td>
</tr>
<tr>
<td>Host name</td>
<td><a href="http://www.ietf.org">www.ietf.org</a></td>
</tr>
</tbody>
</table>

host name ➔ **many** to **many**
IP address ➔ **1** to **1**
MAC address ➔ **DNS,many** to **many**
The Internet Domain Name System

We’ll talk about *name resolution* later…

- host name (has IP address)
  - organization administering host
  - organization administering subnames to left

```plaintext
lupus.fokus.gmd.de
```

Anywhere from two to ∞ parts

---

### The Internet Domain Name System

2 letters: countries
3 letters: independent of geography (except edu, gov, mil)

<table>
<thead>
<tr>
<th>domain</th>
<th>usage</th>
<th>example</th>
</tr>
</thead>
<tbody>
<tr>
<td>com</td>
<td>business (global)</td>
<td>research.att.com</td>
</tr>
<tr>
<td>edu</td>
<td>U.S. higher education</td>
<td>cs.umass.edu</td>
</tr>
<tr>
<td>gov</td>
<td>U.S. non-military gov’t</td>
<td>whitehouse.gov</td>
</tr>
<tr>
<td>mil</td>
<td>U.S. military</td>
<td>arpa.mil</td>
</tr>
<tr>
<td>org</td>
<td>non-profit organization (global)</td>
<td><a href="http://www.ietf.org">www.ietf.org</a></td>
</tr>
<tr>
<td>net</td>
<td>network provider</td>
<td>nis.nsf.net</td>
</tr>
<tr>
<td>us</td>
<td>U.S. geographical</td>
<td>ietf.cnri.reston.va.us</td>
</tr>
<tr>
<td>de</td>
<td>Germany</td>
<td>fokus.gmd.de</td>
</tr>
<tr>
<td>uk</td>
<td>United Kingdom</td>
<td>cs.ucl.ac.uk</td>
</tr>
</tbody>
</table>
**IP Addresses**

Each Internet host has one or more globally unique 32-bit IP addresses, consisting of a network number and a host number:

<table>
<thead>
<tr>
<th>Network</th>
<th>Host</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>Class A</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>Class B</td>
</tr>
<tr>
<td>110</td>
<td></td>
<td>Class C</td>
</tr>
<tr>
<td>1110</td>
<td></td>
<td>Class D</td>
</tr>
</tbody>
</table>

- two-level hierarchy → three-level (later)
- an IP address identifies an *interface*, not a host!
- a host may have two or more addresses. Why?
- net 10: reserved for internal use

**IP Addresses**

- dotted decimal notation: 4 decimal integers, each specifying one byte of IP address:
  - host name: lupus.fokus.gmd.de
  - 32-bit address: 1100 0000 0010 0011 1001 0101 0011 0100
  - dotted decimal: 192.35.149.52
- loopback: 127.0.0.1 (packets never appear on network)
- own network (broadcast): hostid = 0; own host: netid = 0
- directed broadcast: hostid = all ones
- local broadcast: 255.255.255.255
### IP Address Classes

<table>
<thead>
<tr>
<th>class</th>
<th>first</th>
<th>hosts per network</th>
<th>nets</th>
<th>used (1.96)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class A</td>
<td>&lt; 128</td>
<td>16 mio.</td>
<td>128</td>
<td>92</td>
</tr>
<tr>
<td>Class B</td>
<td>128...191</td>
<td>65534</td>
<td>16384</td>
<td>5655</td>
</tr>
<tr>
<td>Class C</td>
<td>192...233</td>
<td>254</td>
<td>2 mio.</td>
<td>87924</td>
</tr>
<tr>
<td>Class D</td>
<td>224...239</td>
<td>268 mio.</td>
<td></td>
<td>dynamic</td>
</tr>
<tr>
<td>Class E</td>
<td>240...255</td>
<td>134 mio.</td>
<td></td>
<td>reserved</td>
</tr>
</tbody>
</table>

---

**Example: ifconfig**

```
ifconfig -a
le0:  flags=863<UP,BROADCAST,NOTRAILERS,RUNNING>
lnot 192.35.149.117 netmask fffffff0 broadcast 192.35.149.0
fa0:  flags=863<UP,BROADCAST,NOTRAILERS,RUNNING>
lnot 194.94.246.72 netmask fffffff0 broadcast 194.94.246.0
qaa0: flags=61<UP,NOTRAILERS,RUNNING>
lnot 193.175.134.117 netmask fffffff0
qaa1: flags=61<UP,NOTRAILERS,RUNNING>
lnot 129.26.216.231 netmask ffff0000
qaa2: flags=60<NOTRAILERS,RUNNING>
qaa3: flags=60<NOTRAILERS,RUNNING>
lo0:  flags=849<UP,LOOPBACK,RUNNING>
lnot 127.0.0.1 netmask fff00000
```
Problems with IP Addresses

- if a host moves from one network to another, its IP address changes
- currently, mostly assigned without regards to topology → too many networks ⇒ CIDR
- limited space ⇒ IPv6
- class thresholds: class C net grows beyond 254 hosts
- hard to change: hidden in lots of places
- multihomed host: path taken to host depends on destination address

CIDR: Classless Interdomain Routing

- problem: too many networks ⇒ routing table explosion
- problem: class C too small, class B too big (and scarce)
- discard class boundaries ⇒ supernetting
- ISP assigns a contiguous group of $2^n$ class C blocks
- “longest match routing” on masked address; e.g. 192.175.132.0/22

<table>
<thead>
<tr>
<th>address/mask</th>
<th>next hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.175.132.0/22</td>
<td>1</td>
</tr>
<tr>
<td>192.175.133.0/23</td>
<td>2</td>
</tr>
<tr>
<td>192.175.128.0/17</td>
<td>3</td>
</tr>
</tbody>
</table>

- e.g., all sites in Europe common prefix ⇒ only single entry in most U.S. routers
**Subnetting**

- large organizations: multiple LANs with single IP network address
- subdivide “host” part of network address ➜ subnetting

```
150.17.1.0 (mask ffffff00) 254 nodes
150.17.1.1
150.17.0.0
150.17.2.1

Network 150.17.1.0 (mask ffffff00)

150.17.1.2
150.17.1.3

<256 subnets

Network 150.17.2.0 (mask ffffff00)

150.17.2.2
150.17.2.3
```

**IP Forwarding**

```
get destination IP address D
if network(D) == directly attached network {
    ARP: D -> MAC address
    put in link layer frame
    forward
}
else
    foreach entry in routing table {
        if (D & subnet mask) == network(entry) {
            get next hop address N
            ARP: N -> MAC address
            put in link layer frame
            forward
        }
    }
}
```

➡️ IP source/destination remains same, MAC changes
IP Forwarding

128.10.0.0
193.175.132.0
193.175.132.1
193.175.132.17
193.175.132.2
anything else
193.175.132.25
193.175.132.2

128.10.0.0/16 193.175.132.1
default 193.175.132.2

GMD Fokus Network

192.35.149.0 (yellow cable)
193.175.132.0 (switched)
193.175.133.0 (switched)

188.1.132.4

modems

192.35.149.247
193.179.135.0
modems

193.175.135.0

internet

(WIN)

ATM

LAX

H

R

pepper

192.35.149.35
192.35.149.35
193.175.133.1
193.175.133.1
193.175.132.1
193.175.132.1
193.175.132.2
193.175.132.248
193.175.133.248
192.35.149.117
192.35.149.117
193.175.134.204
193.175.134.204
192.35.149.117
192.35.149.117
193.175.134.117
193.175.134.117
193.175.133.59
192.35.149.59
193.175.132.1
193.175.133.1
The Internet Protocol

IPv4 Service Model

- **datagram**: each packet is independent of all others
- **best effort**: packet may arrive *or not* after some time
IPv4 Header

<table>
<thead>
<tr>
<th>Field</th>
<th>Bits/Bytes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>version</td>
<td>4</td>
<td>Always 4</td>
</tr>
<tr>
<td>header length</td>
<td>8</td>
<td>Encapsulating upper-layer protocol</td>
</tr>
<tr>
<td>type of service</td>
<td>8</td>
<td>Permitted protocol types</td>
</tr>
<tr>
<td>precedence</td>
<td>3</td>
<td>Rarely used for traffic control, used to prioritize traffic</td>
</tr>
<tr>
<td>DSCP</td>
<td>6</td>
<td>Class of service, differentiated service, minimizes delay, maximize throughput, maximize reliability, minimize cost</td>
</tr>
<tr>
<td>flags</td>
<td>3</td>
<td>DF: Do not fragment, MF: last fragment, reserved for future use</td>
</tr>
<tr>
<td>frag offset</td>
<td>16</td>
<td>Fragment offset, multiple fragments</td>
</tr>
<tr>
<td>total length</td>
<td>16</td>
<td>Total length of data in bytes, up to 65,535 bytes</td>
</tr>
<tr>
<td>ID</td>
<td>16</td>
<td>Identifier for packet from source host, different for each packet</td>
</tr>
<tr>
<td>TTL</td>
<td>8</td>
<td>Time to live, decremented at each router, prevent loops!</td>
</tr>
<tr>
<td>protocol</td>
<td>16</td>
<td>Next higher protocol</td>
</tr>
<tr>
<td>header checksum</td>
<td>16</td>
<td>Add together 16-bit words using one’s complement, optimized for software</td>
</tr>
</tbody>
</table>

IPv4

**version:** always 4

**TOS (type of service):** precedence (3 bits) and “minimize delay”, “maximize throughput”, “maximize reliability”, “minimize cost” bits ➔ rarely used

**identifier:** identifier, different for each packet from host

**TTL:** time to live field; initialized to 64; decremented at each router ➔ drop if TTL = 0 (prevent loops!)

**protocol:** next higher protocol (TCP: 6, UDP: 17)

**header checksum:** add together 16-bit words using one’s complement ➔ optimized for software
IP Fragmentation and Reassembly

Data link protocol may limit packets < 65,536 bytes \Rightarrow transport layer packet may be too big to send in single IP packet

Original Packet

<table>
<thead>
<tr>
<th>Field</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>X</td>
</tr>
<tr>
<td>flg</td>
<td>0</td>
</tr>
<tr>
<td>offset</td>
<td>0</td>
</tr>
<tr>
<td>src</td>
<td>Y</td>
</tr>
<tr>
<td>dest</td>
<td>Z</td>
</tr>
<tr>
<td>data</td>
<td>8K</td>
</tr>
</tbody>
</table>

Yields Two Fragments

<table>
<thead>
<tr>
<th>Field</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>X</td>
</tr>
<tr>
<td>flg</td>
<td>1</td>
</tr>
<tr>
<td>offset</td>
<td>0</td>
</tr>
<tr>
<td>src</td>
<td>Y</td>
</tr>
<tr>
<td>dest</td>
<td>Z</td>
</tr>
<tr>
<td>data</td>
<td>4K</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>X</td>
</tr>
<tr>
<td>flg</td>
<td>0</td>
</tr>
<tr>
<td>offset</td>
<td>4K</td>
</tr>
<tr>
<td>src</td>
<td>Y</td>
</tr>
<tr>
<td>dest</td>
<td>Z</td>
</tr>
<tr>
<td>data</td>
<td>8K</td>
</tr>
</tbody>
</table>

- Split TPDU into fragments
- Each fragment becomes its own IP packet (routers don’t care)
- Each fragment has same identifier, source, destination address
- Fragment offset field gives offset of data from start of original packet
- More fragments (MF) flag of 0 if last (or only) fragment of packet
- Fragments reassembled only at final destination
- Routers must handle at least 576 bytes
- Do not fragment bit prevents fragmentation \Rightarrow drop + error message
- Avoid multiple fragmentation (1500 → 620) \Rightarrow MTU discovery
IP Options

Extend functionality of IP without carrying useless information:

- security and handling restrictions for military
- determine route (source route)
- record route
- record route and timestamps

(rarely used ↔ not all routers support them)

IP Record Route Option

- source creates empty list of $\leq 9$ IP addresses
- option: length, pointer, list of IP addresses
- routers note outgoing interface in list
- …and bump pointer
IP Source Route Option

- source determines path taken by packet (≤ 9 hops)
- loose: any number of hops in between
- strict: every hop; if not directly connected, discard
- same format as record route option
- router overwrites with address of outgoing interface
- must be copied to fragments
- destination should reverse route for return packets
- not too popular ➾ router performance ↓

ICMP

- used to communicate network-level error conditions and info to IP/TCP/UDP entities or user processes
- often considered part of the IP layer, but
  - IP demultiplexes up to ICMP using IP protocol field
  - ICMP messages sent within IP datagram
- ICMP contents always contain IP header and first 8 bytes of IP contents that caused ICMP error message to be generated

<table>
<thead>
<tr>
<th>20-byte standard IP header</th>
<th>8 bit ICMP type</th>
<th>8 bit ICMP code</th>
<th>16-bit checksum</th>
<th>contents of ICMP msg</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>code</td>
<td>description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>------</td>
<td>--------------------------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>echo reply (to a ping)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>destination network unreachable</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>destination host unreachable</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>destination protocol unreachable</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>destination port unreachable</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>fragmentation needed and DF set</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>destination network unknown</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>destination host unknown</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>...</td>
<td>other reasons</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>source quench (slow down)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>redirect message to host</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>echo request (ping)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>IS-ES router advertisement (new)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>ES-IS router discovery (new)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>time exceeded = TTL zero</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>IP header bad</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>0</td>
<td>address (subnet) mask request</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>0</td>
<td>address (subnet) mask reply</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**ping**

- checks if host is reachable, alive
- uses ICMP echo request/reply
- copy packet data request → reply

```
ping -s gaia.cs.umass.edu
PING gaia.cs.umass.edu: 56 data bytes
64 bytes from gaia.cs.umass.edu (128.119.40.186): icmp_seq=0 time=276 ms
64 bytes from gaia.cs.umass.edu (128.119.40.186): icmp_seq=1 time=281 ms
64 bytes from gaia.cs.umass.edu (128.119.40.186): icmp_seq=2 time=276 ms
---gaia.cs.umass.edu PING Statistics---
4 packets transmitted, 3 packets received, 25% packet loss
round-trip (ms) min/avg/max = 276/277/281
```
**traceroute**

- allows to follow path taken by packet
- send UDP to unlikely port; 'time exceeded' and 'port unreachable' ICMP replies
- can use source route (-g), but often doesn’t work

```bash
$ traceroute gaia.cs.umass.edu  
1  gmdbgate (192.35.149.248) 6 ms 2 ms 2 ms
2  188.1.132.142 (188.1.132.142) 263 ms 178 ms 188 ms
3  gmdiegate.gmd.de (192.54.35.68) 153 ms 187 ms 151 ms
4  icm-bonn-1.gmd.de (192.76.246.17) 226 ms 207 ms 242 ms
5  icm-dc-1-52/2-512k.icp.net (192.157.65.209) 320 ms 315 ms 393 ms
6  icm-mae-e-wi0-73.icp.net (198.67.131.9) 372 ms 297 ms 354 ms
7  mae-east (192.41.177.180) 456 ms 537 ms 401 ms
8  border2-hsa12-0.washington.mci.net (204.70.74.117) 529 ms 385 ms 340 ms
9  core-fddi-1.washington.mci.net (204.70.3.1) 437 ms 554 ms 581 ms
10  core-fsa1-3.NewYork.mci.net (204.70.1.6) 418 ms 547 ms 492 ms
11  core-fsa1-3.Boston.mci.net (204.70.1.2) 453 ms 595 ms 724 ms
12  border1-fddi-0.Boston.mci.net (204.70.2.34) 789 ms 404 ms 354 ms
13  nearnet.Boston.mci.net (204.70.20.6) 393 ms 323 ms 346 ms
14  mit3-gw.near.net (192.233.33.10) 340 ms 465 ms 399 ms
15  umass-gw.near.net (199.94.201.66) 557 ms 316 ms 369 ms
16  lgrc-gw.gw.umass.edu (192.80.83.1) 396 ms 309 ms 389 ms
17  cs-gw.cs.umass.edu (128.119.44.1) 276 ms 490 ms 307 ms
18  gaia.cs.umass.edu (128.119.40.186) 335 ms 317 ms 350 ms
```

**ARP: IP address → MAC address**

- for broadcast networks like Ethernet, token ring, …
- if MAC address unknown, send ARP request and hold on to packet
- ARP request → broadcast: sender IP, MAC; target IP, MAC
- *all* machines update their cache ⇒ efficiency, allow change of interface
- ARP reply → requestor: reverse source/target; fill in source MAC
- directly on Ethernet, *not* IP!
- cache ARP replies; drop after 20 minutes
ARP example

rp -a

Net to Media Table

<table>
<thead>
<tr>
<th>Device</th>
<th>IP Address</th>
<th>Mask</th>
<th>Flags</th>
<th>Phys Addr</th>
</tr>
</thead>
<tbody>
<tr>
<td>le0</td>
<td>hamlet</td>
<td>255.255.255.255</td>
<td></td>
<td>08:00:09:70:7d:16</td>
</tr>
<tr>
<td>le0</td>
<td>gaia</td>
<td>255.255.255.255</td>
<td></td>
<td>08:00:20:20:07:03</td>
</tr>
<tr>
<td>le0</td>
<td>pern</td>
<td>255.255.255.255</td>
<td></td>
<td>08:00:20:20:75:3c</td>
</tr>
<tr>
<td>le0</td>
<td>kite</td>
<td>255.255.255.255</td>
<td></td>
<td>08:00:09:92:0d:d1</td>
</tr>
<tr>
<td>le0</td>
<td>condor</td>
<td>255.255.255.255</td>
<td></td>
<td>08:00:20:1c:95:ed</td>
</tr>
</tbody>
</table>

RARP: MAC → IP address

- determine IP address at boot for diskless workstations
- remember: MAC address is unique and permanent
- host broadcasts RARP request (with its own MAC address)
- RARP server responds with reply
- allows third-party queries
- want several servers for reliability
Proxy ARP

- extend network: router fronts for H3, H4
- router answers ARP requests for H3, H4 from H1, H2 with its own hardware address
- assumes trusting relationship
- only needs to be added to single router
- only works for broadcast networks