Lecture 9.

Direct Datagram Forwarding:

Address Resolution Protocol (ARP)

Manual mapping

- ➔**A possibility, indeed!!**
  - ⇨Nothing contrary, in principle
    - ➔actually done in X.25, ISDN (do not support broadcast)
  - ⇨Simply keep in every host a mapping between IP address and hardware address for every IP device connected to the considered network
- ➔**drawbacks**
  - ⇨tedious
  - ⇨error prone
  - ⇨requires manual updating
    - ➔e.g. when attaching a new PC, must touch all others...

Problem statement

- ➔**Routing decision for packet X has two possible outcomes:**
  - ⇨You are arrived to the final network: go to host X
  - ⇨You are not arrived to the final network: go through router interface Y
- ➔**In both cases we have an IP address on THIS network. How can we send data to the interface?**
- ➔**Need to use physical network facilities!**

ARP

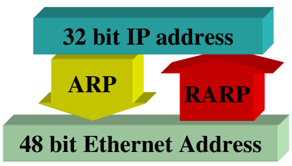
- ➔**Dynamic mapping**
  - ⇨not a concern for application & user
  - ⇨not a concern for system administrator!
- ➔**Any network layer protocol**
  - ⇨not IP-specific
- ➔**supported protocol in datalink layer**
  - ⇨not a datalink layer protocol !!!!
- ➔**Need datalink with broadcasting capability**
  - ⇨e.g. ethernet shared bus

Reaching a physical host

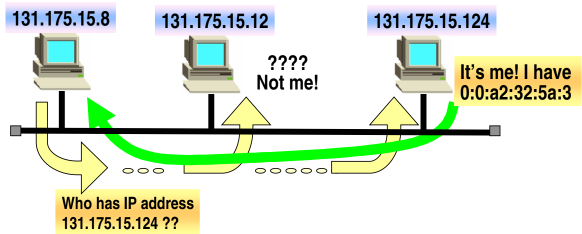
- ➔**IP addresses only make sense to TCPIP protocol suite**
- ➔**physical networks have their own hardware address**
  - ⇨e.g. 48 bits Ethernet address, 16 or 48 bits Token Ring, 16 or 48 bit FDDI, ...
  - ⇨datalink layers may provide the basis for several network layers, not only IP!

Address Resolution Protocol
RFC 826

Here described for Ethernet, but more general: designed for any datalink with broadcast capabilities



ARP idea



- ➔**Send broadcast request**
- ➔**receive unicast response**

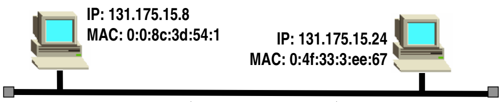
ARP cache

- **➔Avoids arp request for every IP datagram!**
  - ⇨Entry lifetime defaults to 20min
    - →deleted if not used in this time
    - →3 minutes for "incomplete" cache entries (i.e. arp requests to non existent host)
    - →it may be changed in some implementations
      - » in particularly stable (or dynamic) environments
  - ⇨***arp -a*** to display all cache entries (arp –d to delete)

try a traceroute or ping to check ARP caching!

- →First packet generally delays more
- → includes an ARP request/reply!

Sample ARP request/reply



Ethernet Packet: ARP REQUEST			Ethernet Packet: ARP reply	
FF:FF:FF:FF:FF:FF		dest MAC	00:00:8c:3d:54:01	
00:00:8c:3d:54:01		src MAC	00:4f:33:03:ee:67	
0x0806		ARP frame type	0x0806	
0x0001	0x0800	Ethernet / IP	0x0001	0x0800
0x06 0x04	0x0001	MAC=6 / IP=4 / rq=1,rpl=2	0x06 0x04	0x0002
00:00:8c:3d:54:01		src MAC	00:4f:33:03:ee:67	
131.175.15.8		src IP	131.175.15.24	
00:00:00:00:00:00		dest MAC	00:00:8c:3d:54:01	
131.175.15.24		dest IP	131.175.15.8	
checksum		Ethernet checksum	checksum	

ARP request/reply
Incapsulation in Ethernet Frame

6 bytes	6 bytes	2 bytes	28 bytes (for IP)	4 bytes
Ethernet destination address	Ethernet source address	frame type	ARP Request / Reply	CRC

- **➔Ethernet Destination Address**
  - ⇨ff:ff:ff:ff:ff:ff (broadcast) for ARP request
- **➔Ethernet Source Address**
  - ⇨of ARP requester
- **➔Frame Type**
  - ⇨ARP request/reply: 0x0806
  - ⇨RARP request/reply: 0x8035
  - ⇨IP datagram: 0x0800

Protocol demultiplexing codes!

ARP cache updating

- **➔ARP requests carry requestor IP/MAC pair**
- **➔ARP requests are broadcast**
  - ⇨thus, they MUST be read by everyone
- **➔Therefore, it comes for free, for every computer, to update its cache with requestor pair**

- **➔*Cannot do this with ARP reply, as it is unicast!***

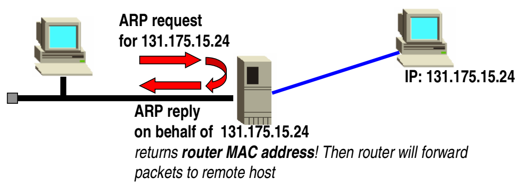
ARP request/reply format

0 – 7	8 – 15	16 – 31	
Hardware Type		Protocol Type	28 bytes
Hardware len	Protocol len	ARP operation	
Sender MAC address (bytes 0-3)			
Sender MAC address (bytes 4-5)		Sender IP address (bytes 0-1)	
Sender IP address (bytes 2-3)		Dest MAC address (bytes 0-1)	
Dest MAC address (bytes 2-5)			
Dest IP address (bytes 0-3)			

Hardware type: 1 for ethernet
Protocol type: 0x0800 for IP (0000.1000.0000.0000)
⇨ *the same of Ethernet header field carrying IP datagram!*
Hardware len = 6 bytes for ethernet
Protocol len = 4 bytes for IP
ARP operation: 1=request; 2=reply; 3/4=RARP req/reply

Proxy ARP

- **➔Device that responds to an ARP request on behalf of some other machine**
  - ⇨allows having ONE logical (IP) network composed of more physical networks
  - ⇨especially important when different techologies used (e.g. 100 PC ethernet + 2 PC dialup SLIP)



Gratuitous ARP

- ➔ **ARP request issued by an IP address and addressed to *the same* IP address!!**
  - ⇨ Clearly nobody else than ME can answer!
  - ⇨ WHY asking the network which MAC address do I have???
- ➔ **Two main reasons:**
  - ⇨ determine if another host is configured with the same IP address
    - → in this case respond occurs, and MAC address of duplicated IP address is known.
  - ⇨ Use gratuitous ARP when just changed hardware address
    - → all other hosts update their cache entries!
    - → A problem is that, despite specified in RFC, not all ARP cache implementations operate as described….

The problem

- ➔ **Bootstrapping a diskless terminal**
  - ⇨ this was the original problem in the 70s and 80s
- ➔ **Reverse ARP [RFC903]**
  - ⇨ a way to obtain an IP address starting from MAC address
- ➔ **Today problem: dynamic IP address assignment**
  - ⇨ limited pool of addresses assigned only when needed
- ➔ **RARP not sufficiently general for modern usage**
  - ⇨ BOOTP (Bootstrap Protocol - RFC 951): significant changes to RARP (a different approach)
  - ⇨ DHCP (Dynamic Host Configuration Protocol - RFC 1541): extends and replaces BOOTP

ARP: not only *this* mechanism!

- ➔ **Described mechanism for broadcast networks (e.g. based on shared media)**
- ➔ **Non applicable for non broadcast networks**
  - ⇨ in this case OTHER ARP protocols are used
    - → e.g. distributed ARP servers
    - → e.g. algorithms to map IP address in network address

RARP packet format

almost identical to ARP. Differences:

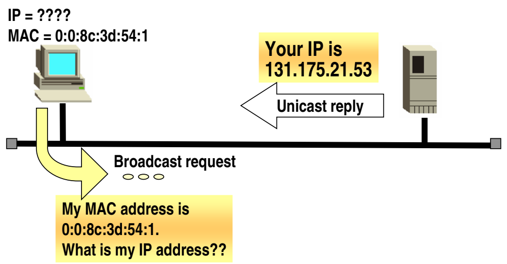
6 bytes	6 bytes	2B	28 bytes (for IP)	4 bytes
Dest addr	Src addr	ftyp: 0x 8035	RARP Request / Reply	CRC

0 – 7	8 – 15	16 – 31
Hardware Type		Protocol Type
Hardware len	Protocol len	oper: 3 (RARP req) or 4 (RARP reply)
Sender MAC address (bytes 0-3)		
Sender MAC address (bytes 4-5)		Sender IP address (bytes 0-1)
Sender IP address (bytes 2-3)		Dest MAC address (bytes 0-1)
Dest MAC address (bytes 2-5)		
Dest IP address (bytes 0-3)		

Getting an IP address:

Reverse Address Resolution Protocol (RARP)

RARP Request/reply



RARP problems

- ➔**Network traffic**
  - ⇨for reliability, multiple RARP servers need to be configured on the same Ethernet
    - →to allow bootstrap of terminals even when one server is down
  - ⇨But this implies that ALL servers simultaneously respond to RARP request
    - →contention on the Ethernet occurs
- ➔**RARP requests not forwarded by routers**
  - ⇨being hardware level broadcasts...

BOOTP parameters exchange

- ➔**Many more parameters**
  - ⇨client IP address (when static IP is assigned)
  - ⇨your IP address (when dynamic server assignment)
  - ⇨gateway IP address (bootp relay agent - router - IP)
  - ⇨server hostname
  - ⇨boot filename
- ➔**Fundamental: vendor-specific information field (64 bytes)**
  - ⇨seems a lot of space: not true!
  - ⇨DHCP uses a 312 vendor-specific field!

RARP fundamental limit

- ➔**Allows only to retrieve the IP address information**
  - ⇨and what about all the remaining full set of TCPIP configuration parameters???
    - →Netmask?
    - →name of servers, proxies, etc?
    - →other proprietary/vendor/ISP-specific info?
- ➔**This is the main reason that has driven to engineer and use BOOTP and DHCP**

Vendor specific information

format allows general information exchange

Tag 1 byte	Len 1 byte	Parameter exchanged

- ➔**E.g.: subnet mask:**
  - ⇨ tag=1, len=4, parameter=32 bit subnet mask
- ➔**e.g.: time offset:**
  - ⇨tag=2, len=4, parameter=time (seconds after midnight, jan 1 1900 UTC)
- ➔**e.g. gateway (variable item)**
  - ⇨tag=3, len=N, list of gateway IPaddr (first preferred)
- ➔**e.g. DNS server (tag 6)**

BOOTP/DHCP approach

- ➔**Requests/replies encapsulated in UDP datagrams**
  - ⇨may cross routers
  - ⇨no more dependent on physical medium
- ➔**request addressing:**
  - ⇨destination IP = 255.255.255.255
  - ⇨source IP = 0.0.0.0
  - ⇨destination port (BOOTP): 67
  - ⇨source port (BOOTP): 68
- ➔**router crossing:**
  - ⇨router configured as BOOTP relay agent
  - ⇨forwards broadcast UDP requests with destination port 67