The Network Layer: Protocols
DHCP, ICMP, NAT, IPv6

CS 352, Lecture 14, Spring 2020
http://www.cs.rutgers.edu/~sn624/352

Srinivas Narayana
Course announcements

• Quiz 5 due Tuesday
  • Will be released later today

• Recording of Wednesday’s lecture available
  • See Piazza for details
  • We are working on converting video into a format that’s more accessible than WebEx’s ARF… stay tuned
Where we are: The network layer

The network layer exists on every endpoint and router.
Review of concepts

- Router components
- Input port: line termination, forwarding function
  - Forwarding: based on IP destination.
  - Longest-prefix matching
- Switching fabric: memory, bus, crossbar
- Output port: buffer management and scheduling
Poll #1

• If an ISP X owns two prefixes 128.0.0.0/16 and 128.1.0.0/16, which IP prefix can the rest of the Internet use in its forwarding rules to route data towards X?
  • (a) 128.0.0.0/8
  • (b) 128.0.0.0/24
  • (c) 128.0.0.0/15
  • (d) None of the above
Poll #2

• Suppose a router has two forwarding rules. Which port should pkt with destination IP 192.168.0.56 go out of?
  192.168.0.0/16 -> port 1
  192.168.0.0/24 -> port 2

• (a) port 1
• (b) port 2
• (c) neither port
• (d) both ports
The Internet Protocol (IP)
# IP datagram format

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ver</td>
<td>IP protocol version number (32 bits)</td>
</tr>
<tr>
<td>length</td>
<td>Header length (32 bits)</td>
</tr>
<tr>
<td>data</td>
<td>“type” of data (variable length, typically a TCP or UDP segment)</td>
</tr>
<tr>
<td>16-bit identifier</td>
<td>16-bit identifier (variable length)</td>
</tr>
<tr>
<td>flgs</td>
<td>flags</td>
</tr>
<tr>
<td>fragment offset</td>
<td>fragment offset</td>
</tr>
<tr>
<td>time to live</td>
<td>Time to live (32 bit)</td>
</tr>
<tr>
<td>upper layer</td>
<td>Upper layer protocol (to deliver payload to upper layer protocol)</td>
</tr>
<tr>
<td>header checksum</td>
<td>Header checksum</td>
</tr>
<tr>
<td>32 bit source IP</td>
<td>32 bit source IP address</td>
</tr>
<tr>
<td>32 bit destination IP</td>
<td>32 bit destination IP address</td>
</tr>
<tr>
<td>Options (if any)</td>
<td>Options (if any)</td>
</tr>
<tr>
<td>total datagram length</td>
<td>Total datagram length (bytes)</td>
</tr>
</tbody>
</table>

- **Max number remaining hops**: (decremented at each router)

- **Upper layer protocol**: to deliver payload to upper layer protocol

**E.g. timestamp, record route taken, specify list of routers to visit.**

**How much overhead with TCP?**

- 20 bytes of TCP
- 20 bytes of IP
- = 40 bytes + app layer overhead
IP Fragmentation & Reassembly

- network links have MTU (max.transfer size) - largest possible link-level frame.
  - different link types, different MTUs
- large IP datagram divided (“fragmented”) within net
  - one datagram becomes several datagrams
  - reassembled only at the destination, at the IP layer
  - IP header bits used to identify, order related fragments

fragmentation:
in: one large datagram
out: 3 smaller datagrams

reassemble
IP Fragmentation and Reassembly

**Example**
- 4000 byte datagram
- MTU = 1500 bytes
- (includes IP headers)

One large datagram becomes several smaller datagrams

<table>
<thead>
<tr>
<th>length</th>
<th>ID</th>
<th>fragflag</th>
<th>offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>4000</td>
<td>x</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

4000 byte datagram

<table>
<thead>
<tr>
<th>length</th>
<th>ID</th>
<th>fragflag</th>
<th>offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>1500</td>
<td>x</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

MTU = 1500 bytes

<table>
<thead>
<tr>
<th>length</th>
<th>ID</th>
<th>fragflag</th>
<th>offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>1500</td>
<td>x</td>
<td>1</td>
<td>185</td>
</tr>
</tbody>
</table>

(includes IP headers)

1480 bytes in data field

offset = 1480/8
Poll #3

• Which header does the IP source and destination live in?
  • (a) TCP header
  • (b) IP header
  • (c) Application-layer header
  • (d) None of the above
Dynamic Host Configuration

How does an endpoint get its IP address?
IP addresses: how to get one?

Q: How does a host get IP address?

• Hard-coded by system admin in a file
  • UNIX: /etc/network/interfaces
  • Windows: controlpanel -> network -> configuration -> tcp/ip -> properties

• DHCP: Dynamic Host Configuration Protocol: dynamically get address from as server
  • “plug-and-play”
Similar bootstrapping problems

• How does a host get its IP address?

• How does a host know its local DNS server?

• How does a host know its subnet mask?

• How does a host know which router is its “gateway” to other networks?
goal: allow host to dynamically obtain its IP address from network server when it joins network

- can renew its lease on address in use
- allows reuse of addresses (only hold address while connected/“on”)
- support for mobile users who want to join network

DHCP overview:

- host broadcasts “DHCP discover” msg [optional]
- DHCP server responds with “DHCP offer” msg [optional]
- host requests IP address: “DHCP request” msg
- DHCP server sends address: “DHCP ack” msg
DHCP client-server scenario
DHCP client-server scenario

DHCP server: 223.1.2.5

DHCP discover

Broadcast: is there a DHCP server out there?

DHCP offer

Broadcast: I'm a DHCP server! Here’s an IP address you can use

DHCP request

Broadcast: OK. I'll take that IP address!

DHCP ACK

Broadcast: OK. You've got that IP address!
DHCP Protocol

Server 1

DHCPDISCOVER

DHCPDISCOVER

DHCPOFFER

DHCPOFFER

Client

Collects replies
Selects server 2

DHCPREQUEST

DHCPREQUEST

Server 2

DHCPOFFER

DHCPOFFER

DHCPACK
DHCP: more than IP addresses

DHCP can return more than just allocated IP address on subnet:

• address of first-hop router for client to reach other subnets
  • Also called the gateway router

• name and IP address of the local DNS server

• subnet mask of the IP network the host is on
  • Useful to know whether other endpoint is inside or outside the current IP network
DHCP Relay Agents

- DHCP relay agents allow DHCP servers to handle requests from other subnets
DHCP: An example

- connecting laptop needs its IP address, addr of first-hop router, addr of DNS server: use DHCP
- DHCP request encapsulated in UDP, encapsulated in IP, encapsulated in 802.1 Ethernet
- Ethernet frame broadcast (dest: FFFFFFFF) on LAN, received at router running DHCP server
- Ethernet demuxed to IP demuxed, UDP demuxed to DHCP
DHCP: An example

- DHCP server formulates DHCP ACK containing client’s IP address, subnet mask, IP address of first-hop router for client, name & IP address of DNS server
  - encapsulation of DHCP server, frame forwarded to client, demuxing up to DHCP at client
  - client now knows its IP address, name and IP address of DNS server, IP address of its first-hop router
Poll #4

- What functions does the DHCP protocol serve?
  - (a) discovering a usable IP address when a host joins
  - (b) discovering the local DNS server address
  - (c) discovering the gateway router of the local IP network
  - (d) all of the above
Poll #5

• When a host joins a new network, how does it know the IP address of the DHCP server?
  • (1) this is a static configuration on all endpoints
  • (2) it broadcasts a message to discover DHCP servers
  • (3) neither of the above
Summary

• IP addresses don’t have to be manually configured into hosts

• DHCP allows “ignorant” hosts to receive IP addresses (and more) at start-up time

• DHCP solves important bootstrapping problems in attaching new hosts to a network
Internet Control Message Protocol (ICMP)
ICMP

- Protocol for error detection and reporting
  - tightly coupled with IP, unreliable
- ICMP messages delivered in IP packets
- ICMP functions:
  - Announce reachability and network errors
  - Announce “time exceeded” errors for IP packets
  - Announce network congestion
- ICMP assists network troubleshooting in general
ICMP message

<table>
<thead>
<tr>
<th>IP header</th>
<th>Source, Destination Address, TTL, ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICMP MSG</td>
<td>Message type, Code, Checksum, Data</td>
</tr>
</tbody>
</table>
ICMP: Internet Control Message Protocol

<table>
<thead>
<tr>
<th>Type</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>echo reply (ping)</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>dest. network unreachable</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>dest host unreachable</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>dest protocol unreachable</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>dest port unreachable</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>dest network unknown</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>dest host unknown</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>source quench (congestion control - not used)</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>echo request (ping)</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>route advertisement</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>router discovery</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>TTL expired</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>bad IP header</td>
</tr>
</tbody>
</table>

Time for an activity!
Specific uses of ICMP

- Echo request reply
  - Can be used to check if a host is alive
- Destination unreachable
  - Invalid address and/or port
- TTL expired
  - Routing loops, or too far away
Ping

• Uses ICMP echo request/reply
• Source sends ICMP echo request message to the destination address
• Destination replies with an ICMP echo reply message containing the data in the original echo request message
• Source can calculate round trip time (RTT) of packets
• If no echo reply comes back, then the destination is unreachable
Ping (cont’d)
Ping example

```
$ ping google.com
PING google.com (172.217.10.110): 56 data bytes
64 bytes from 172.217.10.110: icmp_seq=0 ttl=56 time=5.727 ms
64 bytes from 172.217.10.110: icmp_seq=1 ttl=56 time=4.701 ms
64 bytes from 172.217.10.110: icmp_seq=2 ttl=56 time=5.954 ms
64 bytes from 172.217.10.110: icmp_seq=3 ttl=56 time=11.981 ms
64 bytes from 172.217.10.110: icmp_seq=4 ttl=56 time=6.084 ms
64 bytes from 172.217.10.110: icmp_seq=5 ttl=56 time=5.829 ms
64 bytes from 172.217.10.110: icmp_seq=6 ttl=56 time=8.667 ms
64 bytes from 172.217.10.110: icmp_seq=7 ttl=56 time=8.916 ms
64 bytes from 172.217.10.110: icmp_seq=8 ttl=56 time=4.537 ms
64 bytes from 172.217.10.110: icmp_seq=9 ttl=56 time=4.980 ms
```
Traceroute

• Traceroute records the route that packets take
• A clever use of the IP TTL (time to live) field
• In general, when a router receives a packet, it decrements the TTL on that packet
• If TTL=0, router sends an ICMP time exceeded message back to sender
• traceroute progressively increases TTL of the packets it sends out
  • Every time an ICMP time exceeded message is received, record the sender’s (router’s) address
  • Repeat until the destination host is reached or an error message occurs
• If packet reaches the destination, the dest host usually sends an ICMP port unreachable
Traceroute (cont’d)

Time

TTL=1, Dest = B, port = invalid

TTL=2, Dest = B

TTL=3, Dest = B

TTL=4, Dest = B

Te (R1)

Te (R2)

Te (R3)

Pu (B)

Te = Time exceeded
Pu = Port unreachable
Traceroute example

```bash
traceroute google.com
traceroute to google.com (172.217.10.238), 64 hops max, 52 byte packets
1  vlan451-sr03-hill-nbp.runet.rutgers.net (172.25.112.1) 9.278 ms 3.210 ms 3.124 ms
2  xe-1-3-0-0-cr02-hill-nbp.runet.rutgers.net (172.29.6.65) 37.125 ms 2.912 ms 2.899 ms
3  ae1-2000-cr10-hill-nbp.runet.rutgers.net (172.29.6.42) 3.078 ms 3.086 ms 3.016 ms
4  ae5-2000-cr02-halsey-nwk.runet.rutgers.net (172.29.6.55) 3.693 ms 3.707 ms 3.793 ms
5  ae2-0-er10-halsey-ext.runet.rutgers.net (172.29.8.6) 3.699 ms 3.693 ms 3.766 ms
6  ae1-0-fw01-halsey-nwk.runet.rutgers.net (172.29.8.41) 4.019 ms 3.909 ms 3.750 ms
7  et-2-2-0-0-er10-halsey-ext.runet.rutgers.net (172.29.8.46) 4.310 ms 4.181 ms 3.948 ms
8  gateway-pni.google.com (128.6.1.114) 4.426 ms 3.901 ms 4.161 ms
9  108.170.248.65 (108.170.248.65) 5.024 ms
8  108.170.248.1 (108.170.248.1) 6.147 ms 6.165 ms
10 72.14.233.201 (72.14.233.201) 5.316 ms 5.426 ms 5.359 ms
11 lga25s59-in-f14.1e100.net (172.217.10.238) 5.410 ms 5.156 ms 5.135 ms
```
Poll #6

- What’s the IP header field that traceroute changes for each packet that it sends out to a given destination?
  - (1) IP source address
  - (2) IP source port
  - (3) IP TTL
  - (4) IP length
Network Address Translation (NAT)

How do you survive in a society where names are scarce?
NAT: Network Address Translation

All datagrams leaving local network have same source IP address: 138.76.29.7, with different source IP port numbers.
NAT: Network Address Translation

1: host 10.0.0.1 sends datagram to 128.119.40.186, 80

2: NAT router changes datagram source addr from 10.0.0.1, 3345 to 138.76.29.7, 5001, updates table

3: Reply arrives dest. address: 138.76.29.7, 5001

4: NAT router changes datagram dest addr from 138.76.29.7, 5001 to 10.0.0.1, 3345

<table>
<thead>
<tr>
<th>WAN side addr</th>
<th>LAN side addr</th>
</tr>
</thead>
<tbody>
<tr>
<td>138.76.29.7, 5001</td>
<td>10.0.0.1, 3345</td>
</tr>
</tbody>
</table>

NAT translation table

2: NAT router changes datagram source addr from 10.0.0.1, 3345 to 138.76.29.7, 5001, updates table

3: Reply arrives dest. address: 138.76.29.7, 5001

4: NAT router changes datagram dest addr from 138.76.29.7, 5001 to 10.0.0.1, 3345
NAT: Network Address Translation

- **Features:** local network uses just one IP address as far as outside world is concerned:
  - range of addresses not needed from ISP: just one IP address for all devices
  - can change addresses of devices in local network without notifying outside world
  - can change ISP without changing addresses of devices in local network
  - Devices inside local network not explicitly addressable
  - Devices inside local network invisible to the outside world (a security plus) unless the device inside connects first.

Your home WiFi router uses NATs. If you’re home, you’re behind a NAT right now.
The impact of NATs

```
[flow:352-S20]$ ifconfig en0
en0: flags=8863<UP,BROADCAST,SMART, RUNNING, SIMPLEX, MULTICAST> mtu 1500
    ether f0:18:98:1c:fc:36
    inet6 fe80::1036:7dea:82ee:e868%en0 prefixlen 64 secured scopeid 0xa
    inet 192.168.1.151 netmask 0xfffff00 broadcast 192.168.1.255
    nd6 options=201<PERFORMNUD,DAD>
    media: autoselect
    status: active
```

what's my ip address

Your IP address is 74.102.79.209 in New Brunswick, New Jersey, United States (08901)
NAT: Network Address Translation

• 16-bit port-number field:
  • 60,000 simultaneous connections with a single LAN-side address!

• NAT is controversial:
  • Routers should only work upto the network layer, not transport ports!
  • violates “end-to-end argument”
    • NAT must be taken into account by app designers
    • e.g., P2P applications like skype
  • Purists: address shortage should instead be solved by IPv6
Think about…

• How do the hosts inside the home network get their IP addresses?

• How does your home router get its externally visible IP address?
Poll #7

• The translation table at the NAT router contains
  • (1) Local IP address
  • (2) Local IP port
  • (3) Remote IP address
  • (4) Remote IP port
  • (5) All of the above
Poll #8

• Network Address Translation
  • (1) reduces the demand for fresh IPv4 addresses
  • (2) keeps internal IP addresses hidden from the outside world
  • (3) requires support from the NAT gateway router
  • (4) all of the above
Internet Protocol v6 (IPv6)
Recent Developments: IPv6

- IPv4 has limited address space (32 bits) and is running out of addresses. 32 bits are not enough!

- More devices: phones, watches, your refrigerator(!), …

- Real-time traffic and mobile users are also becoming more common

IP version 6
IPv6: Main changes from IPv4

- **Large address space:**
  - 128-bit addresses (16 bytes)
  - Allows up to $340,282,366,920,938,463,463,374,607,431,768,211,456$ unique addresses ($3.4 \times 10^{38}$)

- **Fixed length headers (40 bytes)**
  - Improves the speed of packet processing in routers

- IPv6 “options” processing happens through a separate mechanism
**IPv6 datagram format**

- **priority**: identify priority among datagrams in flow
- **flow Label**: identify datagrams in same “flow”
  (concept of “flow” left undefined)
- **next header**: identify upper layer protocol for data

<table>
<thead>
<tr>
<th>ver</th>
<th>pri</th>
<th>flow label</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>payload len</th>
<th>next hdr</th>
<th>hop limit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>source address</th>
</tr>
</thead>
<tbody>
<tr>
<td>(128 bits)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>destination address</th>
</tr>
</thead>
<tbody>
<tr>
<td>(128 bits)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>data</th>
</tr>
</thead>
</table>

32 bits
Other changes from IPv4

• **checksum**: removed entirely to reduce processing time at each hop

• **options**: allowed, but outside of header, indicated by “Next Header” field

• **ICMPv6**: new version of ICMP
  • additional message types, e.g. “Packet Too Big”
  • multicast group management functions
IPv4 vs IPv6: Can you tell the differences?

<table>
<thead>
<tr>
<th>ver</th>
<th>pri</th>
<th>flow label</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>payload len</th>
<th>next hdr</th>
<th>hop limit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>source address (128 bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>destination address (128 bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
</tr>
</tbody>
</table>

32 bits
IPv6 Flows

• Support for “flows”
  • Flows help support real-time service in the Internet
  
  • A “flow” is a number in the IPv6 header that can be used by routers to see which packets belong to the same stream
  
  • Guarantees can then be assigned to certain flows
  
• Example:
  • Packets from flow 10 should receive rapid delivery
  • Packets from flow 12 should receive reliable delivery
IPv6 Addresses

- Classless addressing/routing (similar to CIDR)

  - x = 4-bit hex number
  - contiguous 0s are compressed: 47CD::A456:0124
  - IPv6 compatible IPv4 address: ::128.64.18.87
    - First 96 bits are 0
  - Global unicast addresses start with 001....
  - 2000::/3 prefix
IPv6: Adoption

- Google: 8% of clients access services via IPv6
- NIST: 1/3 of all US government domains are IPv6 capable

- *Long (long!) time for deployment, use*
  - 20 years and counting!

- Think of application-level changes in last 20 years: WWW, Facebook, streaming media, Skype, …
- *Why?*