Chapter 4: outline

4.1 introduction
4.2 virtual circuit and datagram networks
4.3 what’s inside a router
4.4 IP: Internet Protocol
   - datagram format
   - IPv4 addressing
   - ICMP
   - IPv6
4.5 routing algorithms
   - link state
   - distance vector
   - hierarchical routing
4.6 routing in the Internet
   - RIP
   - OSPF
   - BGP
4.7 broadcast and multicast routing
The Internet network layer

host, router network layer functions:

- **routing protocols**
  - path selection
  - RIP, OSPF, BGP

- **IP protocol**
  - addressing conventions
  - datagram format
  - packet handling conventions

- **ICMP protocol**
  - error reporting
  - router "signaling"

transport layer: TCP, UDP
**IP datagram format**

- **IP protocol version** number
- **header length** (bytes)
- "type" of data
- max number remaining hops (decremented at each router)
- upper layer protocol to deliver payload to

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ver</td>
<td>IP protocol version number</td>
</tr>
<tr>
<td>len</td>
<td>header length (bytes)</td>
</tr>
<tr>
<td>type of service</td>
<td>&quot;type&quot; of data</td>
</tr>
<tr>
<td>length</td>
<td>length</td>
</tr>
<tr>
<td>16-bit identifier</td>
<td>fragment offset (for fragmentation/reassembly)</td>
</tr>
<tr>
<td>flgs</td>
<td>max number remaining hops (decremented at each router)</td>
</tr>
<tr>
<td>fragment</td>
<td>remaining hops</td>
</tr>
<tr>
<td>offset</td>
<td>max number remaining hops (decremented at each router)</td>
</tr>
<tr>
<td>time to live</td>
<td>max number remaining hops (decremented at each router)</td>
</tr>
<tr>
<td>upper layer</td>
<td>max number remaining hops (decremented at each router)</td>
</tr>
<tr>
<td>checksum</td>
<td>max number remaining hops (decremented at each router)</td>
</tr>
</tbody>
</table>

- **32 bit source IP address**
- **32 bit destination IP address**
- **options (if any)**
- **data** (variable length, typically a TCP or UDP segment)

**how much overhead?**
- 20 bytes of TCP
- 20 bytes of IP
- 40 bytes + app layer overhead

=e.g. timestamp, record route taken, specify list of routers to visit.
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 final destination
  - IP header bits used to identify, order related fragments

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

reassembly
IP fragmentation, reassembly

example:
- 4000 byte datagram
- MTU = 1500 bytes

one large datagram becomes several smaller datagrams

- 1480 bytes in data field
- offset = 1480/8

Network Layer 4-5
Chapter 4: outline

4.1 introduction
4.2 virtual circuit and datagram networks
4.3 what’s inside a router
4.4 IP: Internet Protocol
   ▪ datagram format
   ▪ IPv4 addressing
   ▪ ICMP
   ▪ IPv6
4.5 routing algorithms
   ▪ link state
   ▪ distance vector
   ▪ hierarchical routing
4.6 routing in the Internet
   ▪ RIP
   ▪ OSPF
   ▪ BGP
4.7 broadcast and multicast routing
**IP addressing: introduction**

- **IP address**: 32-bit identifier for host, router *interface*
- **interface**: connection between host/router and physical link
  - router’s typically have multiple interfaces
  - host typically has one or two interfaces (e.g., wired Ethernet, wireless 802.11)
- **IP addresses associated with each interface**

\[
223.1.1.1 = 11011111 00000000 00000001 00000001
\]

Network Layer 4-7
IP addressing: introduction

Q: how are interfaces actually connected?

A: we’ll learn about that in chapter 5, 6.

A: wired Ethernet interfaces connected by Ethernet switches

For now: don’t need to worry about how one interface is connected to another (with no intervening router)

A: wireless WiFi interfaces connected by WiFi base station
Subnets

- IP address:
  - subnet part - high order bits
  - host part - low order bits

- what’s a subnet?
  - device interfaces with same subnet part of IP address
  - can physically reach each other without intervening router

Network consisting of 3 subnets

223.1.1.1
223.1.1.2
223.1.1.3
223.1.1.4
223.1.1.5

223.1.2.1
223.1.2.2
223.1.2.3

223.1.3.1
223.1.3.2
223.1.3.3
223.1.3.4

Network Layer 4-9
Subnets

**recipe**

- to determine the subnets, detach each interface from its host or router, creating islands of isolated networks
- each isolated network is called a *subnet*

Subnet mask: /24

223.1.1.0/24

223.1.1.1
223.1.1.2
223.1.1.3
223.1.1.4

223.1.2.0/24

223.1.2.1
223.1.2.2
223.1.2.9

223.1.3.0/24

223.1.3.1
223.1.3.2
223.1.3.27

223.1.3.22

Network Layer 4-10
Subnets

how many?
IP addressing: CIDR

**CIDR**: Classless InterDomain Routing
- subnet portion of address of arbitrary length
- address format: `a.b.c.d/x`, where `x` is # bits in subnet portion of address

```
11001000  00010111  00010000  00000000
```

`200.23.16.0/23`
IP addresses: how to get one?

Q: How does a *host* get IP address?

- hard-coded by system admin in a file
  - Windows: control-panel->network->configuration->tcp/ip->properties
  - UNIX: /etc/rc.config
- **DHCP**: *Dynamic Host Configuration Protocol*: dynamically get address from as server
  - “plug-and-play”
DHCP: Dynamic Host Configuration Protocol

**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 (more shortly)

**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

arriving DHCP client needs address in this network
DHCP client-server scenario

DHCP server: 223.1.2.5

**DHCP discover**
- src: 0.0.0.0, 68
- dest: 255.255.255.255, 67
- yiaddr: 0.0.0.0
- transaction ID: 654

**DHCP offer**
- src: 223.1.2.5, 67
- dest: 255.255.255.255, 68
- yiaddr: 223.1.2.4
- transaction ID: 654
- lifetime: 3600 secs

**DHCP request**
- src: 0.0.0.0, 68
- dest: 255.255.255.255, 67
- yiaddr: 223.1.2.4
- transaction ID: 655
- lifetime: 3600 secs

**DHCP ACK**
- src: 223.1.2.5, 67
- dest: 255.255.255.255, 68
- yiaddr: 223.1.2.4
- transaction ID: 655
- lifetime: 3600 secs
DHCP: more than IP addresses

DHCP can return more than just allocated IP address on subnet:
- address of first-hop router for client
- name and IP address of DNS server
- network mask (indicating network versus host portion of address)
DHCP: 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: example

- DCP server formulates DHCP ACK containing client’s IP address, 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
DHCP: Wireshark output (home LAN)

Message type: **Boot Request (1)**
Hardware type: Ethernet
Hardware address length: 6
Hops: 0
**Transaction ID:** 0x6b3a11b7
Seconds elapsed: 0
Bootp flags: 0x0000 (Unicast)
Client IP address: 0.0.0.0 (0.0.0.0)
Your (client) IP address: 0.0.0.0 (0.0.0.0)
Next server IP address: 0.0.0.0 (0.0.0.0)
Relay agent IP address: 0.0.0.0 (0.0.0.0)
**Client MAC address:** Wistron_23:68:8a (00:16:d3:23:68:8a)
Server host name not given
Boot file name not given
Magic cookie: (OK)
**Option:** (t=53,l=1) **DHCP Message Type = DHCP Request**
**Option:** (61) **Client identifier**
    Length: 7; Value: 010016D323688A;
    Hardware type: Ethernet
    Client MAC address: Wistron_23:68:8a (00:16:d3:23:68:8a)
**Option:** (t=50,l=4) **Requested IP Address = 192.168.1.101**
**Option:** (t=12,l=5) **Host Name = "nomad"**
**Option:** (55) **Parameter Request List**
    Length: 11; Value: 010F03062C2E2F1F21F92B
    1 = Subnet Mask; 15 = Domain Name
    3 = Router; 6 = Domain Name Server
    44 = NetBIOS over TCP/IP Name Server

Message type: **Boot Reply (2)**
Hardware type: Ethernet
Hardware address length: 6
Hops: 0
**Transaction ID:** 0x6b3a11b7
Seconds elapsed: 0
Bootp flags: 0x0000 (Unicast)
Client IP address: **192.168.1.101 (192.168.1.101)**
Your (client) IP address: 0.0.0.0 (0.0.0.0)
Next server IP address: **192.168.1.1 (192.168.1.1)**
Relay agent IP address: 0.0.0.0 (0.0.0.0)
Client MAC address: Wistron_23:68:8a (00:16:d3:23:68:8a)
Server host name not given
Boot file name not given
Magic cookie: (OK)
**Option:** (t=53,l=1) **DHCP Message Type = DHCP ACK**
**Option:** (t=54,l=4) **Server Identifier = 192.168.1.1**
**Option:** (t=1,l=4) **Subnet Mask = 255.255.255.0**
**Option:** (t=3,l=4) **Router = 192.168.1.1**
**Option:** (6) **Domain Name Server**
    Length: 12; Value: 445747E2445749F244574092;
    IP Address: 68.87.71.226;
    IP Address: 68.87.73.242;
    IP Address: 68.87.64.146
**Option:** (t=15,l=20) **Domain Name = "hsd1.ma.comcast.net."**
**IP addresses: how to get one?**

**Q:** how does network get subnet part of IP addr?

**A:** gets allocated portion of its provider ISP’s address space

<table>
<thead>
<tr>
<th>ISP's block</th>
<th>11001000 00010111 00010000 00000000</th>
<th>200.23.16.0/20</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organization 0</td>
<td>11001000 00010111 00010000 00000000</td>
<td>200.23.16.0/23</td>
</tr>
<tr>
<td>Organization 1</td>
<td>11001000 00010111 00010010 00000000</td>
<td>200.23.18.0/23</td>
</tr>
<tr>
<td>Organization 2</td>
<td>11001000 00010111 00010100 00000000</td>
<td>200.23.20.0/23</td>
</tr>
<tr>
<td>...</td>
<td>....</td>
<td>....</td>
</tr>
<tr>
<td>Organization 7</td>
<td>11001000 00010111 00011110 00000000</td>
<td>200.23.30.0/23</td>
</tr>
</tbody>
</table>
Hierarchical addressing allows efficient advertisement of routing information:

Organizations:
- Organization 0: 200.23.16.0/23
- Organization 1: 200.23.18.0/23
- Organization 2: 200.23.20.0/23
- Organization 7: 200.23.30.0/23

ISPs:
- Fly-By-Night-ISP
- ISPs-R-Us

Internet

“Send me anything with addresses beginning 200.23.16.0/20”
“Send me anything with addresses beginning 199.31.0.0/16”
Hierarchical addressing: more specific routes

ISPs-R-Us has a more specific route to Organization 1

- Organization 0
  - 200.23.16.0/23

- Organization 2
  - 200.23.20.0/23

- Organization 7
  - 200.23.30.0/23

- Organization 1
  - 200.23.18.0/23

- ISPs-R-Us
  - "Send me anything with addresses beginning 200.23.16.0/20"

- Fly-By-Night-ISP
  - "Send me anything with addresses beginning 199.31.0.0/16 or 200.23.18.0/23"

Internet
IP addressing: the last word...

Q: how does an ISP get block of addresses?
A: ICANN: Internet Corporation for Assigned Names and Numbers http://www.icann.org/
- allocates addresses
- manages DNS
- assigns domain names, resolves disputes
NAT: network address translation

all datagrams leaving local network have same single source NAT IP address: 138.76.29.7, different source port numbers

datagrams with source or destination in this network have 10.0.0/24 address for source, destination (as usual)
motivation: 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 net not explicitly addressable, visible by outside world (a security plus)
NAT: network address translation

**implementation:** NAT router must:

- **outgoing datagrams: replace** (source IP address, port #) of every outgoing datagram to (NAT IP address, new port #) . . . remote clients/servers will respond using (NAT IP address, new port #) as destination addr

- **remember (in NAT translation table)** every (source IP address, port #) to (NAT IP address, new port #) translation pair

- **incoming datagrams: replace** (NAT IP address, new port #) in dest fields of every incoming datagram with corresponding (source IP address, port #) stored in NAT table
**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

**NAT translation table**

<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>
<tr>
<td>……</td>
<td>……</td>
</tr>
</tbody>
</table>

Network Layer 4-28
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 process up to layer 3
  - violates end-to-end argument
    - NAT possibility must be taken into account by app designers, e.g., P2P applications
  - address shortage should instead be solved by IPv6
NAT traversal problem

- client wants to connect to server with address 10.0.0.1
  - server address 10.0.0.1 local to LAN (client can’t use it as destination addr)
  - only one externally visible NATed address: 138.76.29.7

- solution 1: statically configure NAT to forward incoming connection requests at given port to server
  - e.g., (123.76.29.7, port 2500) always forwarded to 10.0.0.1 port 25000
NAT traversal problem

- **solution 2**: Universal Plug and Play (UPnP) Internet Gateway Device (IGD) Protocol. Allows NATed host to:
  - learn public IP address (138.76.29.7)
  - add/remove port mappings (with lease times)

i.e., automate static NAT port map configuration
NAT traversal problem

- **solution 3**: relaying (used in Skype)
  - NATed client establishes connection to relay
  - external client connects to relay
  - relay bridges packets between to connections

1. connection to relay initiated by NATed host
2. connection to relay initiated by client
3. relaying established

138.76.29.7

client

10.0.0.1

NAT router

Network Layer 4-32
Chapter 4: outline

4.1 introduction
4.2 virtual circuit and datagram networks
4.3 what’s inside a router
4.4 IP: Internet Protocol
  ▪ datagram format
  ▪ IPv4 addressing
  ▪ ICMP
  ▪ IPv6
4.5 routing algorithms
  ▪ link state
  ▪ distance vector
  ▪ hierarchical routing
4.6 routing in the Internet
  ▪ RIP
  ▪ OSPF
  ▪ BGP
4.7 broadcast and multicast routing
ICMP: internet control message protocol

- used by hosts & routers to communicate network-level information
  - error reporting: unreachable host, network, port, protocol
  - echo request/reply (used by ping)
- network-layer “above” IP:
  - ICMP msgs carried in IP datagrams
- ICMP message: type, code plus first 8 bytes of IP datagram causing error

<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>
Traceroute and ICMP

- Source sends series of UDP segments to dest
  - First set has TTL = 1
  - Second set has TTL = 2, etc.
  - Unlikely port number
- When nth set of datagrams arrives to nth router:
  - Router discards datagrams
  - And sends source ICMP messages (type 11, code 0)
  - ICMP messages includes name of router & IP address
- When ICMP messages arrives, source records RTTs

Stopping criteria:
- UDP segment eventually arrives at destination host
- Destination returns ICMP “port unreachable” message (type 3, code 3)
- Source stops
IPv6: motivation

- **initial motivation**: 32-bit address space soon to be completely allocated.
- additional motivation:
  - header format helps speed processing/forwarding
  - header changes to facilitate QoS

**IPv6 datagram format:**
- fixed-length 40 byte header
- no fragmentation allowed
**IPv6 datagram format**

- **priority**: identify priority among datagrams in flow
- **flow Label**: identify datagrams in same “flow.”
  (concept of “flow” not well defined).
- **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>payload len next hdr hop limit</td>
</tr>
<tr>
<td>source address (128 bits)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>destination address (128 bits)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>data</td>
<td></td>
<td></td>
</tr>
</tbody>
</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
Transition from IPv4 to IPv6

- not all routers can be upgraded simultaneously
  - no “flag days”
  - how will network operate with mixed IPv4 and IPv6 routers?
- tunneling: IPv6 datagram carried as payload in IPv4 datagram among IPv4 routers
Tunneling

logical view:

physical view:

IPv4 tunnel connecting IPv6 routers
Tunneling

IPv4 tunnel connecting IPv6 routers

logical view:

A IPv6

B IPv6

E IPv6

F IPv6

physical view:

A IPv6

B IPv6

C IPv4

D IPv4

E IPv6

F IPv6

A-to-B: IPv6

B-to-C: IPv6 inside IPv4

Flow: X
Src: A
Dest: F

data

Flow: X
Src: A
Dest: F

data

src:B
dest: E

src:B
dest: E

flow: X
src: A
dest: F

data

E-to-F: IPv6

Network Layer 4-41