Chapter 4: network layer

Welcome to Layer 3!

chapter goals:

- understand principles behind network layer services:
  - network layer service models
  - forwarding versus routing
  - how a router works
  - routing (path selection)
  - broadcast, multicast
- instantiation, implementation in the Internet
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

Network layer

- transport segment from sending to receiving host
- on sending side encapsulates segments into *datagrams*
- on receiving side, delivers segments to transport layer
- network layer protocols in *every* host, router
- router examines header fields in all IP datagrams passing through it
Two key network-layer functions

- **forwarding**: move packets from router’s input to appropriate router output
- **routing**: determine route taken by packets from source to dest.
  - *routing algorithms*

**analogy:**

- **routing**: process of planning trip from source to dest
- **forwarding**: process of getting through single interchange

Interplay between routing and forwarding

<table>
<thead>
<tr>
<th>routing algorithm</th>
<th>local forwarding table</th>
</tr>
</thead>
<tbody>
<tr>
<td>header value/output link</td>
<td>0100 3</td>
</tr>
<tr>
<td></td>
<td>0101 2</td>
</tr>
<tr>
<td></td>
<td>0111 1</td>
</tr>
<tr>
<td></td>
<td>1001 1</td>
</tr>
</tbody>
</table>

- **routing algorithm determines** end-end-path through network
- **forwarding table determines** local forwarding at this router
Connection setup

- 3rd important function in some network architectures:
  - ATM, frame relay, X.25
- before datagrams flow, two end hosts and intervening routers establish virtual connection
  - routers get involved
- network vs transport layer connection service:
  - network: between two hosts (may also involve intervening routers in case of VCs)
  - transport: between two processes

Network service model

Q: What service model for “channel” transporting datagrams from sender to receiver?

example services for individual datagrams:
- guaranteed delivery
- guaranteed delivery with less than 40 msec delay

example services for a flow of datagrams:
- in-order datagram delivery
- guaranteed minimum bandwidth to flow
- restrictions on changes in inter-packet spacing
Network layer service models:

<table>
<thead>
<tr>
<th>Network Architecture</th>
<th>Service Model</th>
<th>Guarantees?</th>
<th>Congestion feedback</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Bandwidth</td>
<td>Loss</td>
</tr>
<tr>
<td>Internet</td>
<td>Best effort</td>
<td>none</td>
<td>no</td>
</tr>
<tr>
<td>ATM</td>
<td>CBR</td>
<td>constant</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td>VBR</td>
<td>guaranteed rate</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td>ABR</td>
<td>guaranteed minimum</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td>UBR</td>
<td>none</td>
<td>no</td>
</tr>
</tbody>
</table>

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
Connection, connection-less service

- **datagram** network
  - provides network-layer *connectionless* service
- **virtual-circuit** network
  - provides network-layer *connection* service
- analogous to TCP/UDP connection-oriented / connectionless transport-layer services, but:
  - **service**: host-to-host
  - **no choice**: network provides one or the other
  - **implementation**: in network core

Virtual circuits

“source-to-dest path behaves much like telephone circuit”
- performance-wise
- network actions along source-to-dest path

- call setup, teardown for each call *before* data can flow
- each packet carries VC identifier (not destination host address)
- *every* router on source-dest path maintains “state” for each passing connection
- link, router resources (bandwidth, buffers) may be *allocated* to VC (dedicated resources = predictable service)
VC implementation

*a VC consists of:*
1. *path* from source to destination
2. *VC numbers*, one number for each link along path
3. *entries in forwarding tables* in routers along path

- packet belonging to VC carries VC number (rather than dest address)
- VC number can be changed on each link.
  - new VC number comes from forwarding table

VC forwarding table

<table>
<thead>
<tr>
<th>Incoming interface</th>
<th>Incoming VC #</th>
<th>Outgoing interface</th>
<th>Outgoing VC #</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>12</td>
<td>3</td>
<td>22</td>
</tr>
<tr>
<td>2</td>
<td>63</td>
<td>1</td>
<td>18</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>2</td>
<td>17</td>
</tr>
<tr>
<td>1</td>
<td>97</td>
<td>3</td>
<td>87</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

*VC routers maintain connection state information!*

Network Layer 4-13

Network Layer 4-14
**Virtual circuits: signaling protocols**

- used to setup, maintain, teardown VC
- used in ATM, frame-relay, X.25
- not used in today’s Internet

**Datagram networks**

- no call setup at network layer
- routers: no state about end-to-end connections
  - no network-level concept of “connection”
- packets forwarded using destination host address
Datagram forwarding table

IP destination address in arriving packet's header

4 billion IP addresses, so rather than list individual destination address list range of addresses (aggregate table entries)

Datagram forwarding table

<table>
<thead>
<tr>
<th>Destination Address Range</th>
<th>Link Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>11001000 00010111 00010000 00000000 through 11001000 00010111 00010111 11111111</td>
<td>0</td>
</tr>
<tr>
<td>11001000 00010111 00011000 00000000 through 11001000 00010111 00011000 11111111</td>
<td>1</td>
</tr>
<tr>
<td>11001000 00010111 00011001 00000000 through 11001000 00010111 00011111 11111111</td>
<td>2</td>
</tr>
<tr>
<td>otherwise</td>
<td>3</td>
</tr>
</tbody>
</table>

Q: but what happens if ranges don’t divide up so nicely?
Longest prefix matching

Longest prefix matching—when looking for forwarding table entry for given destination address, use longest address prefix that matches destination address.

<table>
<thead>
<tr>
<th>Destination Address Range</th>
<th>Link interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>11001000 00010111 00010*** ***</td>
<td>0</td>
</tr>
<tr>
<td>11001000 00010111 00011000 *******</td>
<td>1</td>
</tr>
<tr>
<td>11001000 00010111 00011*** *******</td>
<td>2</td>
</tr>
<tr>
<td>otherwise 00010111 00011*** *******</td>
<td>3</td>
</tr>
</tbody>
</table>

Examples:

- DA: 11001000 00010111 00011000 10101010
- DA: 11001000 00010111 00010110 10100001

which interface?
which interface?

Datagram or VC network: why?

**Internet (datagram)**
- data exchange among computers
  - “elastic” service, no strict timing req.
- many link types
  - different characteristics
  - uniform service difficult
- “smart” end systems (computers)
  - can adapt, perform control, error recovery
  - simple inside network, complexity at “edge”

**ATM (VC)**
- evolved from telephony
- human conversation:
  - strict timing, reliability requirements
  - need for guaranteed service
- “dumb” end systems
  - telephones
  - complexity inside network

Network Layer 4-19

Network Layer 4-20
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

**Router architecture overview**

two key router functions:
- run routing algorithms/protocol (RIP, OSPF, BGP)
- forwarding datagrams from incoming to outgoing link
Input port functions

- line termination
- link layer protocol (receive)
- lookup, forwarding
- queueing

Switching fabrics

- transfer packet from input buffer to appropriate output buffer
- switching rate: rate at which packets can be transfer from inputs to outputs
  - often measured as multiple of input/output line rate
  - N inputs: switching rate N times line rate desirable
- three types of switching fabrics

Decentralized switching:
- given datagram dest., lookup output port using forwarding table in input port memory ("match plus action")
- goal: complete input port processing at 'line speed'
- queuing: if datagrams arrive faster than forwarding rate into switch fabric

Network Layer 4-23

Network Layer 4-24
Switching via memory

**first generation routers:**
- traditional computers with switching under direct control of CPU
- packet copied to system’s memory
- speed limited by memory bandwidth (2 bus crossings per datagram)

![Diagram of switching via memory]

Switching via a bus

- datagram from input port memory to output port memory via a shared bus
- **bus contention:** switching speed limited by bus bandwidth
- 32 Gbps bus, Cisco 5600: sufficient speed for access and enterprise routers

![Diagram of switching via a bus]
Switching via interconnection network

- overcome bus bandwidth limitations
- banyan networks, crossbar, other interconnection nets initially developed to connect processors in multiprocessor
- advanced design: fragmenting datagram into fixed length cells, switch cells through the fabric.
- Cisco 12000: switches 60 Gbps through the interconnection network

Output ports

- buffering required when datagrams arrive from fabric faster than the transmission rate
- scheduling discipline chooses among queued datagrams for transmission
Output port queueing

- buffering when arrival rate via switch exceeds output line speed
- queueing (delay) and loss due to output port buffer overflow!

How much buffering?

- RFC 3439 rule of thumb: average buffering equal to “typical” RTT (say 250 msec) times link capacity C
  - e.g., C = 10 Gpbs link: 2.5 Gbit buffer
- recent recommendation: with N flows, buffering equal to

\[ \frac{\text{RTT} \cdot C}{\sqrt{N}} \]
**Input port queuing**

- fabric slower than input ports combined -> queueing may occur at input queues
  - queueing delay and loss due to input buffer overflow!
- Head-of-the-Line (HOL) blocking: queued datagram at front of queue prevents others in queue from moving forward

**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:

### IP datagram format

- **IP protocol version**
- **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><code>ver</code></td>
<td>IP protocol version</td>
</tr>
<tr>
<td><code>len</code></td>
<td>Length of header in bytes</td>
</tr>
<tr>
<td><code>type of service</code></td>
<td>Type of service, typically TCP or UDP segment</td>
</tr>
<tr>
<td><code>flags</code></td>
<td>Flags for fragmentation/reassembly</td>
</tr>
<tr>
<td><code>offset</code></td>
<td>Offset for fragmentation/reassembly</td>
</tr>
<tr>
<td><code>id</code></td>
<td>16-bit identifier</td>
</tr>
<tr>
<td><code>tos</code></td>
<td>Time to live</td>
</tr>
<tr>
<td><code>upper layer</code></td>
<td>Upper layer protocol</td>
</tr>
<tr>
<td><code>src ip</code></td>
<td>32-bit source IP address</td>
</tr>
<tr>
<td><code>dst ip</code></td>
<td>32-bit destination IP address</td>
</tr>
<tr>
<td><code>options</code></td>
<td>Options (if any)</td>
</tr>
<tr>
<td><code>data</code></td>
<td>Payload data (variable length)</td>
</tr>
</tbody>
</table>

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

Network Layer 4-33

Network Layer 4-34
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

Example:

- 4000 byte datagram
- MTU = 1500 bytes

1480 bytes in data field
offset = 1480/8

One large datagram becomes several smaller datagrams
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 00000001 00000001 00000001
223 1 1 1
```
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
  - a.k.a. "prefix"
- 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
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

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

Subnetting

- Start with a network block:
  - 172.16.0.0/16
- Decide what's needed:
  - Need so-many subnets?
  - Need so-many hosts-per-subnet?
- Determine number of bits needed to count each subnet or each host-on-a-subnet
- Move bits from host ID to network ID to according to subnet/hosts needs
  - Define corresponding subnet mask
Subnetting example

- Network block: 172.16.0/16
  - 10101100 00010000 hhhhhhhh hhhhhhh
  - Network mask:
    - 11111111 11111111 00000000 00000000
    - = 255.255.0.0
- 75 subnets needed
  - so, need enough bits to count at least 75
- \(2^7 = 128 \geq 75\), so 7 bits needed
- Move 7 leftmost host-ID bits into network ID
  - 10101100 00010000 nnnnnnnn hhhhhhh
- New subnet mask:
  - 11111111 11111111 11111111 00000000
  - = 255.255.254.0

Subnetting example – more

- Network ID bits: 16 (original block) + 7 (subnetting) = 23
  - Subnetworks: 172.16.x.0/23
    - where x varies from one subnet to the next
  - Subnet mask: 255.255.254.0
- Host ID bits: 32 – 23 = 9
  - \(2^9 – 2 = 510\) hosts per subnet

2015-10-21
IP Address Classes (the old way)

<table>
<thead>
<tr>
<th>Class</th>
<th>Subnet Prefixes</th>
<th>Address Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0 n n n n n n n n</td>
<td>1.0.0.0 to 126.0.0.0</td>
</tr>
<tr>
<td>B</td>
<td>1 0 n n n n n n n n</td>
<td>128.0.0.0 to 191.255.0.0</td>
</tr>
<tr>
<td>C</td>
<td>1 1 0 n n n n n n n</td>
<td>192.0.0.0 to 223.255.255.0</td>
</tr>
<tr>
<td>D</td>
<td>1 1 1 0 m m m m m m</td>
<td>224.0.0.0 to 239.255.255.255</td>
</tr>
<tr>
<td>E</td>
<td>1 1 1 1 e e e e e e</td>
<td>240.0.0.0 to 255.255.255.255</td>
</tr>
</tbody>
</table>

- class A: 1.0.0.0 to 126.0.0.0
- class B: 128.0.0.0 to 191.255.0.0
- class C: 192.0.0.0 to 223.255.255.0
- class D: 224.0.0.0 to 239.255.255.255
- class E: 240.0.0.0 to 255.255.255.255

Who has such addresses?

- An unusual holder of IP addresses is the university M.I.T. It has multiple addresses:
  - a class A address, 18.0.0.0
  - four class B addresses
  - a handful of class C addresses

[IP Ranges at MIT]

MIT’s IP ranges are:

```plaintext
18.*.*.*
128.30.*.*
128.31.*.*
128.52.*.*
129.55.*.*
192.52.61.*-192.52.66.*
198.125.160.*-198.125.163.*
198.125.176.*-198.125.192.*
```

The IP address for the proxy server is 18.51.1.222.
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

DHCP server: 223.1.2.5

arriving client

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

Network Layer 4-51

Network Layer 4-52
**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 is a version of the BOOTP protocol.
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
Boostrap 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

DHCP: Wireshark output (home LAN)

Message type: Boot Reply (2)
Hardware type: Ethernet
Hardware address length: 6
Hops: 0
Transaction ID: 0x6b3a11b7
Seconds elapsed: 0
Boostrap 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

ISP’s block

<table>
<thead>
<tr>
<th>Organization 0</th>
<th>11001000 00010111 00010000 00000000</th>
<th>200.23.16.0/20</th>
</tr>
</thead>
<tbody>
<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>200.23.30.0/23</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>

Network Layer 4-57
Hierarchical addressing: route aggregation

Hierarchical addressing allows efficient advertisement of routing information:

```
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

Fly-By-Night-ISP

"Send me anything with addresses beginning 200.23.16.0/20"

ISPs-R-Us

"Send me anything with addresses beginning 199.31.0.0/16"

Internet
```

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

Fly-By-Night-ISP

"Send me anything with addresses beginning 200.23.16.0/20"

ISPs-R-Us

"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)
NAT: network address translation

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 its 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>

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

Network Layer 4-65

Network Layer 4-66
NAT traversal problem

- client wants to connect to server with internal 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

![Diagram showing NAT traversal problem](diagram.png)

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

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

- For communication between IP layers on network nodes
  - Ping application can use ICMP
- "Layer 3.5" – ICMP messages are carried in IP packets
- Constant-size header
  - Varying field meanings based on the message
- Data often absent
  - ping – data field padded with arbitrary contents
  - some error messages include portions of a failed IP packet (headers, etc.)

### Some ICMP Message Types and Codes

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

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |

1st word | 2nd word | 3rd word | 4th word
Type | Code | Rest-of-Header
---|---|---

Data: 0 .. (65535-20+8) octets (limited by IP Total Length field)

Types:

0  Echo reply (to ping)
1-2  (reserved)
3  Destination Unreachable (Code value indicates why)
4  Source Quench (for congestion control)
5  Redirect Message (more information in Code)
6  Alternate Host Address
7  (reserved)
8  Echo Request (ping)
9  Router Advertisement
10  Router Solicitation
11  Time Exceeded (Code 0: TTL is 0; Code 1: Fragment Reassembly time exceeded)
12  Bad IP Header (more information in Code)
13  Timestamp
14  Timestamp Reply
15  Register
16  Domain Name Request
17  Domain Name Reply

Some other Types are also defined

Code: depends on Type, provides further information

Checksum: covers ICMP header and any data

Rest-of-Header: depends on Type and Code

---

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

Network Layer 4-74
ICMP packet reporting an error

- In this Wireshark session a machine has requested a TCP connection.
  - ...by sending a SYN packet.
- The ICMP packet is reporting that the connection has been refused.
  - Its payload is the IP and TCP headers of the packet that made the request.

ICMP as a Phishing Tool

- 2006 – phishing Trojan captures account/password information, transmits to a host using encrypted ping packets

This "innocent" ping packet...
**IPv6: motivation**

- *initial motivation:* 32-bit address space is 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

**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></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>
</table>

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

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

---

32 bits
IPv4 and IPv6 Headers Compared

<table>
<thead>
<tr>
<th>IPv4 Header</th>
<th>IPv6 Header</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>Version</td>
</tr>
<tr>
<td>IHL</td>
<td>Traffic Class</td>
</tr>
<tr>
<td>Total Length</td>
<td>Flow Label</td>
</tr>
<tr>
<td>Identification</td>
<td>Payload Length</td>
</tr>
<tr>
<td>Flags</td>
<td>Next Header</td>
</tr>
<tr>
<td>Fragment Offset</td>
<td>Hop Limit</td>
</tr>
<tr>
<td>Source Address</td>
<td>Source Address</td>
</tr>
<tr>
<td>Destination Address</td>
<td>Destination Address</td>
</tr>
<tr>
<td>Options</td>
<td>Padding</td>
</tr>
</tbody>
</table>

Legend:
- Field’s name kept from IPv4 to IPv6
- Field not kept in IPv6
- Name and position changed in IPv6
- New field in IPv6

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:
Tunneling

logical view:

physical view:

IPv4 tunnel connecting IPv6 routers