Chapter 4
Network Layer

A note on the use of these ppt slides:
We’re making these slides freely available to all (faculty, students, readers). They’re in PowerPoint form so you see the animations; and can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a lot of work on our part. In return for use, we only ask the following:
- If you use these slides (e.g., in a class) that you mention their source (after all, we’d like people to use our book!)
- If you post any slides on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

Thanks and enjoy! JFK/KWR

Computer Networking: A Top Down Approach
6th edition
Jim Kurose, Keith Ross
Addison-Wesley
March 2012

© All material copyright 1996-2012
J.F Kurose and K.W. Ross, All Rights Reserved
Chapter 4: network layer

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

Routing algorithm determines end-end-path through network. Forwarding table determines local forwarding at this router.

Value in arriving packet’s header.

Routing algorithm

<table>
<thead>
<tr>
<th>header value</th>
<th>output link</th>
</tr>
</thead>
<tbody>
<tr>
<td>0100</td>
<td>3</td>
</tr>
<tr>
<td>0101</td>
<td>2</td>
</tr>
<tr>
<td>0111</td>
<td>2</td>
</tr>
<tr>
<td>1001</td>
<td>1</td>
</tr>
</tbody>
</table>
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 rate</td>
<td>yes</td>
</tr>
<tr>
<td>ATM</td>
<td>VBR</td>
<td>guaranteed rate</td>
<td>yes</td>
</tr>
<tr>
<td>ATM</td>
<td>ABR</td>
<td>guaranteed minimum</td>
<td>no</td>
</tr>
<tr>
<td>ATM</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**

**forwarding table in northwest router:**

<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!**
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

Routing algorithm

Local forwarding table

<table>
<thead>
<tr>
<th>dest address</th>
<th>output link</th>
</tr>
</thead>
<tbody>
<tr>
<td>address-range 1</td>
<td>3</td>
</tr>
<tr>
<td>address-range 2</td>
<td>2</td>
</tr>
<tr>
<td>address-range 3</td>
<td>2</td>
</tr>
<tr>
<td>address-range 4</td>
<td>1</td>
</tr>
</tbody>
</table>

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

IP destination address in arriving packet’s header
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

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</td>
<td>3</td>
</tr>
</tbody>
</table>

Examples:

- DA: 11001000 00010111 00010110 10100001 10100001 10100001 which interface?
- DA: 11001000 00010111 00011000 10101010 10101010 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*
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
Router architecture overview

two key router functions:

- run routing algorithms/protocol (RIP, OSPF, BGP)
- **forwarding** datagrams from incoming to outgoing link

![Diagram of router architecture](image)
Network Layer 4-23

Input port functions

- **physical layer**: bit-level reception
- **data link layer**: e.g., Ethernet

**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
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
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)
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
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 Gbps 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

```
output port contention: only one red datagram can be transferred.
lower red packet is blocked
```

```
one packet time later: green packet experiences HOL blocking
```

Network Layer 4-31
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"
# IP datagram format

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP protocol version number</td>
<td>IP protocol version number (bytes)</td>
</tr>
<tr>
<td>header length</td>
<td>“type” of data</td>
</tr>
<tr>
<td>max number remaining hops</td>
<td>(decremented at each router)</td>
</tr>
<tr>
<td>upper layer protocol to deliver payload to</td>
<td>options (if any)</td>
</tr>
</tbody>
</table>

- **32 bit source IP address**
- **32 bit destination IP address**
- **type of service**
- **time to live**
- **upper layer protocol**
- **flags**
- **fragment offset**
- **checksum**
- **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

<table>
<thead>
<tr>
<th>how much overhead?</th>
</tr>
</thead>
<tbody>
<tr>
<td>20 bytes of TCP</td>
</tr>
<tr>
<td>20 bytes of IP</td>
</tr>
<tr>
<td>= 40 bytes + app layer overhead</td>
</tr>
</tbody>
</table>
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
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

---

<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>

<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>

<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>

<table>
<thead>
<tr>
<th>length</th>
<th>ID</th>
<th>fragflag</th>
<th>offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>1040</td>
<td>=x</td>
<td>=0</td>
<td>=370</td>
</tr>
</tbody>
</table>

Network Layer 4-36
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 00000001 00000001 00000001
          223   1    1    1
```

Network Layer 4-38
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*
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?

Network Layer 4-42
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 a 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

arriving DHCP client needs address in this network

Network Layer 4-46
DHCP client-server scenario

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 server: 223.1.2.5
arriving client

Network Layer 4-47
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)
- 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)
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: (50,l=4) Requested IP Address = 192.168.1.101
Option: (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: (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>Organization 0</th>
<th>Organization 1</th>
<th>Organization 2</th>
<th>...</th>
<th>Organization 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>11001000 00010111 00010000 00000000</td>
<td>200.23.16.0/23</td>
<td>11001000 00010111 00010010 00000000</td>
<td>200.23.18.0/23</td>
<td>...</td>
<td>11001000 00010111 00011110 00000000</td>
</tr>
</tbody>
</table>
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

Send me anything with addresses beginning **200.23.16.0/20**

Send me anything with addresses beginning **199.31.0.0/16** or **200.23.18.0/23**
**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/](http://www.icann.org/)
- allocates addresses
- manages DNS
- assigns domain names, resolves disputes
**NAT: network address translation**

- **rest of Internet**
- **local network** (e.g., home network) 10.0.0/24
- **138.76.29.7**
- **10.0.0.1**
- **10.0.0.2**
- **10.0.0.3**

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.
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

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>

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

Network Layer 4-59
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
Network Layer 4-61

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
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 $n$th set of datagrams arrives to $n$th 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>payload len</td>
<td>next hdr</td>
<td>hop limit</td>
</tr>
</tbody>
</table>

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

IPv6
A IPv6
B
IPv6
E IPv6
F

physical view:

IPv6
A IPv6
B IPv4
C IPv4
D IPv4
E IPv6
F IPv6

IPv4 tunnel connecting IPv6 routers
Tunneling

physical view:

A
IPv6

flow: X
src: A
dest: F
data

B
IPv6

A-to-B:
IPv6

B-to-C:
IPv6 inside
IPv4

C
IPv4

E-to-F:
IPv6

D
IPv4

src:B
dest: E

Flow: X
Src: A
Dest: F
data

E
IPv6

src:B
dest: E

Flow: X
Src: A
Dest: F
data

F
IPv6

Network Layer 4-72
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
Interplay between routing, forwarding

- Routing algorithm determines end-end-path through network.
- Forwarding table determines local forwarding at this router.

<table>
<thead>
<tr>
<th>Dest Address</th>
<th>Output Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>address-range 1</td>
<td>3</td>
</tr>
<tr>
<td>address-range 2</td>
<td>2</td>
</tr>
<tr>
<td>address-range 3</td>
<td>2</td>
</tr>
<tr>
<td>address-range 4</td>
<td>1</td>
</tr>
</tbody>
</table>

IP destination address in arriving packet’s header.
Graph abstraction

Graph: $G = (N, E)$

$N$ = set of routers = \{u, v, w, x, y, z\}

$E$ = set of links = \{(u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z)\}

* aside: graph abstraction is useful in other network contexts, e.g., P2P, where $N$ is set of peers and $E$ is set of TCP connections
Graph abstraction: costs

c(x,x’) = cost of link (x,x’)

\[ c(w,z) = 5 \]

cost could always be 1, or inversely related to bandwidth, or inversely related to congestion

cost of path \((x_1, x_2, x_3, \ldots, x_p)\) = \(c(x_1, x_2) + c(x_2, x_3) + \ldots + c(x_{p-1}, x_p)\)

**key question:** what is the least-cost path between \(u\) and \(z\) ?

**routing algorithm:** algorithm that finds that least cost path
Routing algorithm classification

Q: global or decentralized information?

global:
- all routers have complete topology, link cost info
- “link state” algorithms

decentralized:
- router knows physically-connected neighbors, link costs to neighbors
- iterative process of computation, exchange of info with neighbors
- “distance vector” algorithms

Q: static or dynamic?

static:
- routes change slowly over time

dynamic:
- routes change more quickly
  - periodic update
  - in response to link cost changes
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
A Link-State Routing Algorithm

Dijkstra’s algorithm

- net topology, link costs known to all nodes
  - accomplished via “link state broadcast”
  - all nodes have same info
- computes least cost paths from one node (“source”) to all other nodes
  - gives forwarding table for that node
- iterative: after k iterations, know least cost path to k dest.’s

notation:

- \( c(x,y) \): link cost from node x to y; \( = \infty \) if not direct neighbors
- \( D(v) \): current value of cost of path from source to dest. v
- \( p(v) \): predecessor node along path from source to v
- \( N' \): set of nodes whose least cost path definitively known
Dijsktra’s Algorithm

1 *Initialization:*
2 \( N' = \{u\} \)
3 for all nodes \( v \)
4 if \( v \) adjacent to \( u \)
5 then \( D(v) = c(u,v) \)
6 else \( D(v) = \infty \)

8 *Loop*
9 find \( w \) not in \( N' \) such that \( D(w) \) is a minimum
10 add \( w \) to \( N' \)
11 update \( D(v) \) for all \( v \) adjacent to \( w \) and not in \( N' \) :
12 \( D(v) = \min( D(v), D(w) + c(w,v) ) \)
13 /* new cost to \( v \) is either old cost to \( v \) or known
14 shortest path cost to \( w \) plus cost from \( w \) to \( v \ )* /
15 until all nodes in \( N' \)
## Dijkstra’s algorithm: example

<table>
<thead>
<tr>
<th>Step</th>
<th>N'</th>
<th>D(v)</th>
<th>D(w)</th>
<th>D(x)</th>
<th>D(y)</th>
<th>D(z)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>p(v)</td>
<td>p(w)</td>
<td>p(x)</td>
<td>p(y)</td>
<td>p(z)</td>
</tr>
<tr>
<td>0</td>
<td>u</td>
<td>7,u</td>
<td>3,u</td>
<td>5,u</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>1</td>
<td>uw</td>
<td>6,w</td>
<td>5,u</td>
<td>11,w</td>
<td>∞</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>uwx</td>
<td>6,w</td>
<td></td>
<td>11,w</td>
<td>14,x</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>uwxv</td>
<td>6,w</td>
<td></td>
<td>11,w</td>
<td>14,x</td>
<td>10,v</td>
</tr>
<tr>
<td>4</td>
<td>uwxvy</td>
<td></td>
<td></td>
<td>10,v</td>
<td>14,x</td>
<td>12,y</td>
</tr>
<tr>
<td>5</td>
<td>uwxvwy</td>
<td></td>
<td></td>
<td>10,v</td>
<td>14,x</td>
<td></td>
</tr>
</tbody>
</table>

### Notes:
- Construct shortest path tree by tracing predecessor nodes.
- Ties can exist (can be broken arbitrarily).
### Dijkstra’s algorithm: another example

<table>
<thead>
<tr>
<th>Step</th>
<th>N'</th>
<th>D(v),p(v)</th>
<th>D(w),p(w)</th>
<th>D(x),p(x)</th>
<th>D(y),p(y)</th>
<th>D(z),p(z)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>u</td>
<td>2,u</td>
<td>5,u</td>
<td>1,u</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>ux</td>
<td>2,u</td>
<td>4,x</td>
<td>2,x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>uxy</td>
<td>2,u</td>
<td>3,y</td>
<td></td>
<td>4,y</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>uxyv</td>
<td></td>
<td>3,y</td>
<td></td>
<td>4,y</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>uxyvw</td>
<td></td>
<td></td>
<td></td>
<td>4,y</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>uxyvwz</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4,y</td>
</tr>
</tbody>
</table>

![Graph Diagram](Network Layer 4-82)
Dijkstra’s algorithm: example (2)

resulting shortest-path tree from u:

resulting forwarding table in u:

<table>
<thead>
<tr>
<th>destination</th>
<th>link</th>
</tr>
</thead>
<tbody>
<tr>
<td>v</td>
<td>(u,v)</td>
</tr>
<tr>
<td>x</td>
<td>(u,x)</td>
</tr>
<tr>
<td>y</td>
<td>(u,x)</td>
</tr>
<tr>
<td>w</td>
<td>(u,x)</td>
</tr>
<tr>
<td>z</td>
<td>(u,x)</td>
</tr>
</tbody>
</table>
Dijkstra’s algorithm, discussion

**algorithm complexity:** n nodes
- each iteration: need to check all nodes, w, not in N
- \( n(n+1)/2 \) comparisons: \( O(n^2) \)
- more efficient implementations possible: \( O(n\log n) \)

**oscillations possible:**
- e.g., support link cost equals amount of carried traffic:

\[
\begin{array}{c}
A \quad 1+e \\
D \quad 1 \\
C \quad e \\
B \\
\end{array}
\quad \begin{array}{c}
A \quad 2+e \\
D \quad 1+e \\
C \quad 1 \\
B \\
\end{array}
\quad \begin{array}{c}
A \quad 2+e \\
D \quad 1+e \\
C \quad 1 \\
B \\
\end{array}
\]

Initially: given these costs, find new routing….
resulting in new costs

Network Layer 4-84
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
Distance vector algorithm

Bellman-Ford equation (dynamic programming)

let
\[ d_x(y) := \text{cost of least-cost path from } x \text{ to } y \]
then
\[ d_x(y) = \min \left\{ c(x,v) + d_v(y) \right\} \]

- cost from neighbor \( v \) to destination \( y \)
- cost to neighbor \( v \)
- \( \min \) taken over all neighbors \( v \) of \( x \)
clearly, $d_v(z) = 5$, $d_x(z) = 3$, $d_w(z) = 3$

B-F equation says:

$$d_u(z) = \min \{ c(u,v) + d_v(z), \ c(u,x) + d_x(z), \ c(u,w) + d_w(z) \}$$

$$= \min \{ 2 + 5, \ 1 + 3, \ 5 + 3 \} = 4$$

node achieving minimum is next hop in shortest path, used in forwarding table
Distance vector algorithm

- $D_x(y) =$ estimate of least cost from $x$ to $y$
  - $x$ maintains distance vector $D_x = [D_x(y) : y \in N]$
- node $x$:
  - knows cost to each neighbor $v$: $c(x,v)$
  - maintains its neighbors’ distance vectors. For each neighbor $v$, $x$ maintains
  $D_v = [D_v(y) : y \in N]$
Distance vector algorithm

key idea:

- from time-to-time, each node sends its own distance vector estimate to neighbors
- when x receives new DV estimate from neighbor, it updates its own DV using B-F equation:

\[ D_x(y) \leftarrow \min_v \{ c(x,v) + D_v(y) \} \quad \text{for each node } y \in N \]

- under minor, natural conditions, the estimate \( D_x(y) \) converge to the actual least cost \( d_x(y) \)
Distance vector algorithm

iterative, asynchronous:
- each local iteration caused by:
  - local link cost change
  - DV update message from neighbor

distributed:
- each node notifies neighbors only when its DV changes
  - neighbors then notify their neighbors if necessary

each node:

wait for (change in local link cost or msg from neighbor)

recompute estimates

if DV to any dest has changed, notify neighbors
\[ D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\} = \min\{2+0, 7+1\} = 2 \]

\[ D_z(x) = \min\{c(x,y) + D_y(z), c(x,z) + D_z(z)\} = \min\{2+1, 7+0\} = 3 \]
\[ D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\} = \min\{2+0, 7+1\} = 2 \]

\[ D_x(z) = \min\{c(x,y) + D_y(z), c(x,z) + D_z(z)\} = \min\{2+1, 7+0\} = 3 \]
Distance vector: link cost changes

*link cost changes:*

- node detects local link cost change
- updates routing info, recalculates distance vector
- if DV changes, notify neighbors

"good news travels fast"

\[ t_0 \]: y detects link-cost change, updates its DV, informs its neighbors.

\[ t_1 \]: z receives update from y, updates its table, computes new least cost to x, sends its neighbors its DV.

\[ t_2 \]: y receives z’ s update, updates its distance table. y’ s least costs do *not* change, so y does *not* send a message to z.
Distance vector: link cost changes

**link cost changes:**
- node detects local link cost change
- *bad news travels slow* - “count to infinity” problem!
- 44 iterations before algorithm stabilizes: see text

**poisoned reverse:**
- If Z routes through Y to get to X:
  - Z tells Y its (Z’s) distance to X is infinite (so Y won’t route to X via Z)
- will this completely solve count to infinity problem?
Comparison of LS and DV algorithms

**message complexity**
- **LS:** with n nodes, E links, O(nE) msgs sent
- **DV:** exchange between neighbors only
  - convergence time varies

**speed of convergence**
- **LS:** O(n^2) algorithm requires O(nE) msgs
  - may have oscillations
- **DV:** convergence time varies
  - may be routing loops
  - count-to-infinity problem

**robustness:** what happens if router malfunctions?

- **LS:**
  - node can advertise incorrect link cost
  - each node computes only its own table

- **DV:**
  - DV node can advertise incorrect path cost
  - each node’s table used by others
    - error propagate thru network
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
Hierarchical routing

our routing study thus far - idealization
- all routers identical
- network “flat”
... not true in practice

scale: with 600 million destinations:
- can’t store all dest’s in routing tables!
- routing table exchange would swamp links!

administrative autonomy
- internet = network of networks
- each network admin may want to control routing in its own network
Hierarchical routing

- aggregate routers into regions, “autonomous systems” (AS)
- routers in same AS run same routing protocol
  - “intra-AS” routing protocol
  - routers in different AS can run different intra-AS routing protocol

**gateway router:**
- at “edge” of its own AS
- has link to router in another AS
Interconnected ASes

✓ forwarding table configured by both intra- and inter-AS routing algorithm
  ▪ intra-AS sets entries for internal dests
  ▪ inter-AS & intra-AS sets entries for external dests
Inter-AS tasks

- Suppose router in AS1 receives datagram destined outside of AS1:
  - Router should forward packet to gateway router, but which one?

AS1 must:

1. Learn which dests are reachable through AS2, which through AS3
2. Propagate this reachability info to all routers in AS1

Job of inter-AS routing!
Example: setting forwarding table in router 1d

- suppose AS1 learns (via inter-AS protocol) that subnet *x* reachable via AS3 (gateway 1c), but not via AS2
  - inter-AS protocol propagates reachability info to all internal routers
- router 1d determines from intra-AS routing info that its interface 1 is on the least cost path to 1c
  - installs forwarding table entry (x,1)
**Example: choosing among multiple ASes**

- now suppose AS1 learns from inter-AS protocol that subnet \( x \) is reachable from AS3 *and* from AS2.
- to configure forwarding table, router 1d must determine which gateway it should forward packets towards for dest \( x \)
  - this is also job of inter-AS routing protocol!
Example: choosing among multiple ASes

- now suppose AS1 learns from inter-AS protocol that subnet $x$ is reachable from AS3 and from AS2.
- to configure forwarding table, router 1d must determine towards which gateway it should forward packets for dest $x$
  - this is also job of inter-AS routing protocol!
- **hot potato routing**: send packet towards closest of two routers.
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
Intra-AS Routing

- also known as *interior gateway protocols (IGP)*
- most common intra-AS routing protocols:
  - RIP: Routing Information Protocol
  - OSPF: Open Shortest Path First
  - IGRP: Interior Gateway Routing Protocol (Cisco proprietary)
**RIP (Routing Information Protocol)**

- included in BSD-UNIX distribution in 1982
- distance vector algorithm
  - distance metric: # hops (max = 15 hops), each link has a cost of 1
  - DVs exchanged with neighbors every 30 sec in response message (aka advertisement)
  - each advertisement: a list of up to 25 destination subnets (in IP addressing sense)

```
from router A to destination subnets:

<table>
<thead>
<tr>
<th>subnet</th>
<th>hops</th>
</tr>
</thead>
<tbody>
<tr>
<td>u</td>
<td>1</td>
</tr>
<tr>
<td>v</td>
<td>2</td>
</tr>
<tr>
<td>w</td>
<td>2</td>
</tr>
<tr>
<td>x</td>
<td>3</td>
</tr>
<tr>
<td>y</td>
<td>3</td>
</tr>
<tr>
<td>z</td>
<td>2</td>
</tr>
</tbody>
</table>
```

Network Layer 4-106
RIP: example

Routing table in router D

<table>
<thead>
<tr>
<th>destination subnet</th>
<th>next router</th>
<th># hops to dest</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>A</td>
<td>2</td>
</tr>
<tr>
<td>y</td>
<td>B</td>
<td>2</td>
</tr>
<tr>
<td>x</td>
<td>--</td>
<td>1</td>
</tr>
<tr>
<td>....</td>
<td>....</td>
<td>....</td>
</tr>
</tbody>
</table>
RIP: example

A-to-D advertisement

<table>
<thead>
<tr>
<th>dest</th>
<th>next</th>
<th>hops</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>x</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>z</td>
<td>C</td>
<td>4</td>
</tr>
</tbody>
</table>

routing table in router D

<table>
<thead>
<tr>
<th>destination subnet</th>
<th>next router</th>
<th># hops to dest</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>A</td>
<td>2</td>
</tr>
<tr>
<td>y</td>
<td>B</td>
<td>2</td>
</tr>
<tr>
<td>z</td>
<td>B</td>
<td>7</td>
</tr>
<tr>
<td>x</td>
<td>--</td>
<td>1</td>
</tr>
<tr>
<td>....</td>
<td>....</td>
<td>....</td>
</tr>
</tbody>
</table>

Network Layer 4-108
RIP: link failure, recovery

if no advertisement heard after 180 sec -->
neighbor/link declared dead

- routes via neighbor invalidated
- new advertisements sent to neighbors
- neighbors in turn send out new advertisements (if tables changed)
- link failure info quickly (?) propagates to entire net
- *poison reverse* used to prevent ping-pong loops (infinite distance = 16 hops)
**RIP table processing**

- RIP routing tables managed by *application-level* process called route-d (daemon)
- advertisements sent in UDP packets, periodically repeated

![Diagram showing network layer routing tables](image)
OSPF (Open Shortest Path First)

- “open”: publicly available
- uses link state algorithm
  - LS packet dissemination
  - topology map at each node
  - route computation using Dijkstra’s algorithm
- OSPF advertisement carries one entry per neighbor
- advertisements flooded to entire AS
  - carried in OSPF messages directly over IP (rather than TCP or UDP)
- IS-IS routing protocol: nearly identical to OSPF
OSPF “advanced” features (not in RIP)

- **security**: all OSPF messages authenticated (to prevent malicious intrusion)
- **multiple same-cost paths** allowed (only one path in RIP)
- for each link, multiple cost metrics for different TOS (e.g., satellite link cost set “low” for best effort ToS; high for real time ToS)
- integrated uni- and **multicast** support:
  - Multicast OSPF (MOSPF) uses same topology database as OSPF
- **hierarchical** OSPF in large domains.
Hierarchical OSPF

Network Layer 4-113
Hierarchical OSPF

- **two-level hierarchy**: local area, backbone.
  - link-state advertisements only in area
  - each node has detailed area topology; only know direction (shortest path) to nets in other areas.
- **area border routers**: “summarize” distances to nets in own area, advertise to other Area Border routers.
- **backbone routers**: run OSPF routing limited to backbone.
- **boundary routers**: connect to other AS’s.
Internet inter-AS routing: BGP

- BGP (Border Gateway Protocol): the de facto inter-domain routing protocol
  - “glue that holds the Internet together”
- BGP provides each AS a means to:
  - eBGP: obtain subnet reachability information from neighboring ASs.
  - iBGP: propagate reachability information to all AS-internal routers.
  - determine “good” routes to other networks based on reachability information and policy.
- allows subnet to advertise its existence to rest of Internet: “I am here”
BGP basics

- **BGP session**: two BGP routers ("peers") exchange BGP messages:
  - advertising *paths* to different destination network prefixes ("path vector" protocol)
  - exchanged over semi-permanent TCP connections

- when AS3 advertises a prefix to AS1:
  - AS3 *promises* it will forward datagrams towards that prefix
  - AS3 can aggregate prefixes in its advertisement
BGP basics: distributing path information

- using eBGP session between 3a and 1c, AS3 sends prefix reachability info to AS1.
  - 1c can then use iBGP to distribute new prefix info to all routers in AS1
  - 1b can then re-advertise new reachability info to AS2 over 1b-to-2a eBGP session

- when router learns of new prefix, it creates entry for prefix in its forwarding table.
Path attributes and BGP routes

- advertised prefix includes BGP attributes
  - prefix + attributes = “route”
- two important attributes:
  - **AS-PATH**: contains ASs through which prefix advertisement has passed: e.g., AS 67, AS 17
  - **NEXT-HOP**: indicates specific internal-AS router to next-hop AS. (may be multiple links from current AS to next-hop-AS)
- gateway router receiving route advertisement uses **import policy** to accept/decline
  - e.g., never route through AS x
  - *policy-based* routing
BGP route selection

- router may learn about more than 1 route to destination AS, selects route based on:
  1. local preference value attribute: policy decision
  2. shortest AS-PATH
  3. closest NEXT-HOP router: hot potato routing
  4. additional criteria
**BGP messages**

- BGP messages exchanged between peers over TCP connection
- BGP messages:
  - **OPEN**: opens TCP connection to peer and authenticates sender
  - **UPDATE**: advertises new path (or withdraws old)
  - **KEEPALIVE**: keeps connection alive in absence of UPDATES; also ACKs OPEN request
  - **NOTIFICATION**: reports errors in previous msg; also used to close connection
BGP routing policy

- A, B, C are provider networks
- X, W, Y are customer (of provider networks)
- X is dual-homed: attached to two networks
  - X does not want to route from B via X to C
  - .. so X will not advertise to B a route to C
BGP routing policy (2)

- A advertises path AW to B
- B advertises path BAW to X
- Should B advertise path BAW to C?
  - No way! B gets no “revenue” for routing CBAW since neither W nor C are B’s customers
  - B wants to force C to route to w via A
  - B wants to route *only* to/from its customers!
Why different Intra-, Inter-AS routing?

**policy:**
- inter-AS: admin wants control over how its traffic routed, who routes through its net.
- intra-AS: single admin, so no policy decisions needed

**scale:**
- hierarchical routing saves table size, reduced update traffic

**performance:**
- intra-AS: can focus on performance
- inter-AS: policy may dominate over performance
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
Broadcast routing

- deliver packets from source to all other nodes
- source duplication is inefficient:

- source duplication: how does source determine recipient addresses?
**In-network duplication**

- **flooding**: when node receives broadcast packet, sends copy to all neighbors
  - problems: cycles & broadcast storm
- **controlled flooding**: node only broadcasts pkt if it hasn’t broadcast same packet before
  - node keeps track of packet ids already broadcasted
  - or reverse path forwarding (RPF): only forward packet if it arrived on shortest path between node and source
- **spanning tree**:
  - no redundant packets received by any node
Spanning tree

- first construct a spanning tree
- nodes then forward/make copies only along spanning tree

(a) broadcast initiated at A

(b) broadcast initiated at D
Spanning tree: creation

- center node
- each node sends unicast join message to center node
  - message forwarded until it arrives at a node already belonging to spanning tree

(a) stepwise construction of spanning tree (center: E)  
(b) constructed spanning tree
**Multicast routing: problem statement**

**goal:** find a tree (or trees) connecting routers having local mcast group members

- **tree:** not all paths between routers used
- **shared-tree:** same tree used by all group members
- **source-based:** different tree from each sender to rcvrs

**legend**
- group member
- not group member
- router with a group member
- router without group member
Approaches for building mcast trees

approaches:

- **source-based tree:** one tree per source
  - shortest path trees
  - reverse path forwarding

- **group-shared tree:** group uses one tree
  - minimal spanning (Steiner)
  - center-based trees

...we first look at basic approaches, then specific protocols adopting these approaches.
Shortest path tree

- mcast forwarding tree: tree of shortest path routes from source to all receivers
  - Dijkstra’s algorithm
Reverse path forwarding

- rely on router’s knowledge of unicast shortest path from it to sender
- each router has simple forwarding behavior:

\[
\text{if (mcast datagram received on incoming link on shortest path back to center)} \\
\text{then flood datagram onto all outgoing links} \\
\text{else ignore datagram}
\]
Reverse path forwarding: example

- result is a source-specific reverse SPT
  - may be a bad choice with asymmetric links

LEGEND
- router with attached group member
- router with no attached group member
- datagram will be forwarded
- datagram will not be forwarded

s: source
Reverse path forwarding: pruning

- forwarding tree contains subtrees with no mcast group members
  - no need to forward datagrams down subtree
  - “prune” msgs sent upstream by router with no downstream group members

![Diagram of network routers and links]

**LEGEND**
- router with attached group member
- router with no attached group member
- prune message
- links with multicast forwarding

s: source
Shared-tree: steiner tree

- **steiner tree**: minimum cost tree connecting all routers with attached group members
- problem is NP-complete
- excellent heuristics exists
- not used in practice:
  - computational complexity
  - information about entire network needed
  - monolithic: rerun whenever a router needs to join/leave
Center-based trees

- single delivery tree shared by all
- one router identified as “center” of tree
- to join:
  - edge router sends unicast join-msg addressed to center router
  - join-msg “processed” by intermediate routers and forwarded towards center
  - join-msg either hits existing tree branch for this center, or arrives at center
  - path taken by join-msg becomes new branch of tree for this router
Center-based trees: example

suppose R6 chosen as center:

LEGEND

- router with attached group member
- router with no attached group member
- path order in which join messages generated
Internet Multicasting Routing: DVMRP

- **DVMRP**: distance vector multicast routing protocol, RFC1075
- **flood and prune**: reverse path forwarding, source-based tree
  - RPF tree based on DVMRP’s own routing tables constructed by communicating DVMRP routers
  - no assumptions about underlying unicast
  - initial datagram to mcast group flooded everywhere via RPF
  - routers not wanting group: send upstream prune msgs
DVMRP: continued...

- **soft state:** DVMRP router periodically (1 min.) “forgets” branches are pruned:
  - mcast data again flows down unpruned branch
  - downstream router: reprune or else continue to receive data

- routers can quickly regraft to tree
  - following IGMP join at leaf

- odds and ends
  - commonly implemented in commercial router
Tunneling

Q: how to connect “islands” of multicast routers in a “sea” of unicast routers?

- mcast datagram encapsulated inside “normal” (non-multicast-addressed) datagram
- normal IP datagram sent thru “tunnel” via regular IP unicast to receiving mcast router (recall IPv6 inside IPv4 tunneling)
- receiving mcast router unencapsulates to get mcast datagram
PIM: Protocol Independent Multicast

- not dependent on any specific underlying unicast routing algorithm (works with all)
- two different multicast distribution scenarios:

  **dense:**
  - group members densely packed, in “close” proximity.
  - bandwidth more plentiful

  **sparse:**
  - # networks with group members small wrt # interconnected networks
  - group members “widely dispersed”
  - bandwidth not plentiful
Consequences of sparse-dense dichotomy:

**dense**
- group membership by routers *assumed* until routers explicitly prune
- *data-driven* construction on mcast tree (e.g., RPF)
- bandwidth and non-group-router processing *profligate*

**sparse:**
- no membership until routers explicitly join
- *receiver-driven* construction of mcast tree (e.g., center-based)
- bandwidth and non-group-router processing *conservative*
PIM- dense mode

flood-and-prune RPF: similar to DVMRP but…

- underlying unicast protocol provides RPF info for incoming datagram
- less complicated (less efficient) downstream flood than DVMRP reduces reliance on underlying routing algorithm
- has protocol mechanism for router to detect it is a leaf-node router
PIM - sparse mode

- center-based approach
- router sends *join* msg to rendezvous point (RP)
  - intermediate routers update state and forward *join*
- after joining via RP, router can switch to source-specific tree
  - increased performance: less concentration, shorter paths

all data multicast from rendezvous point

rendezvous point

Network Layer 4-144
PIM - sparse mode

sender(s):

- unicast data to RP, which distributes down RP-rooted tree
- RP can extend mcast tree upstream to source
- RP can send stop msg if no attached receivers
  - “no one is listening!”
Chapter 4: done!

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

✓ 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