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 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) in substantially unaltered form, that you mention their source (after all, we’d like people to use our book!)
- If you post any slides in substantially unaltered form 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

All material copyright 1996-2007
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)
  - dealing with scale
  - advanced topics: IPv6, mobility

- instantiation, implementation in the Internet
Chapter 4: Network Layer

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

- **routing**: determine route taken by packets from source to dest.
  - **routing algorithms**
Interplay between routing and forwarding

- Routing algorithm
  - Local forwarding table:
    - Header value | Output link
    - 0100          | 3
    - 0101          | 2
    - 0111          | 2
    - 1001          | 1

Value in arriving packet’s header
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>Bandwidth</th>
<th>Loss</th>
<th>Order</th>
<th>Timing</th>
<th>Congestion feedback</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internet</td>
<td>best effort</td>
<td>none</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no (inferred via loss)</td>
</tr>
<tr>
<td>ATM</td>
<td>CBR</td>
<td>constant  rate</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no congestion</td>
</tr>
<tr>
<td>ATM</td>
<td>VBR</td>
<td>guaranteed rate</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no congestion</td>
</tr>
<tr>
<td>ATM</td>
<td>ABR</td>
<td>guaranteed minimum</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>ATM</td>
<td>UBR</td>
<td>none</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>

Guarantees?
Chapter 4: Network Layer

- 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 connection and connection-less service

- datagram network provides network-layer connectionless service
- VC network provides network-layer connection service
- analogous to the 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
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>

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
  - packets between same source-dest pair may take different paths
# 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>
## Longest prefix matching

<table>
<thead>
<tr>
<th>Prefix Match</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 10101010`  
  Which interface?

- **DA**: `11001000 00010111 00011000 10101010`  
  Which interface?
Datagram or VC network: why?

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

ATM (VC)
- evolved from telephony
- human conversation:
  - strict timing, reliability requirements
  - need for guaranteed service
- "dumb" end systems
  - telephones
  - complexity inside network
Chapter 4: Network Layer

- 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

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

Physical layer: bit-level reception
Data link layer: e.g., Ethernet
see chapter 5
Three types of switching fabrics

Memory

Bus

Crossbar
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 An Interconnection Network

- overcome bus bandwidth limitations
- Banyan networks, 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
- Head-of-the-Line (HOL) blocking: queued datagram at front of queue prevents others in queue from moving forward
- queueing delay and loss due to input buffer overflow!
Chapter 4: Network Layer

- 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:

Transport layer: TCP, UDP

- Routing protocols
  - path selection
  - RIP, OSPF, BGP

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

- ICMP protocol
  - error reporting
  - router “signaling”

Network layer

Link layer

physical layer
Chapter 4: Network Layer

- 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 datagram format**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP protocol version number</td>
<td>8 bits, identifies the IP protocol version (IPv4 = 0, IPv6 = 6)</td>
</tr>
<tr>
<td>header length (bytes)</td>
<td>4 bits, the number of bytes in the header excluding the header length field</td>
</tr>
<tr>
<td>“type” of data</td>
<td>8 bits, identifies the type of data (e.g., TCP, UDP)</td>
</tr>
<tr>
<td>max number remaining hops</td>
<td>16 bits, decremented at each router on the path to the destination</td>
</tr>
<tr>
<td>upper layer protocol</td>
<td>8 bits, identifies the upper layer protocol</td>
</tr>
<tr>
<td>32 bit source IP address</td>
<td>32 bits, the source IP address</td>
</tr>
<tr>
<td>32 bit destination IP address</td>
<td>32 bits, the destination IP address</td>
</tr>
<tr>
<td>Options (if any)</td>
<td>Additional options for specific purposes</td>
</tr>
<tr>
<td>data</td>
<td>Variable length, typically a TCP or UDP segment</td>
</tr>
</tbody>
</table>

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

**How does TCP fragmentation work?**
- Large data packets are broken into smaller fragments.
- Each fragment is encapsulated in an IP datagram.
- A checksum is added to each fragment to ensure integrity.
- The destination IP address is the same for all fragments.

**Checksum (16 bits)**
- Used to verify the integrity of the IP datagram.

**Time to live (8 bits)**
- Used by the network layer to limit the number of hops a datagram can take.
- A hop count is decremented by 1 at each router.
- If the count reaches 0, the datagram is discarded.

**Upper layer protocol**
- The protocol used to deliver the payload to the application layer.

**How to use options?**
- Timestamp: records the time a packet was generated.
- Record route: records the path a packet took.

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

---

**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

Diagram:
- Fragmentation: in: one large datagram out: 3 smaller datagrams
- Reassembly:
IP Fragmentation and Reassembly

Example

- 4000 byte datagram
- MTU = 1500 bytes

1480 bytes in data field

Offset = 1480/8

One large datagram becomes several smaller datagrams
Chapter 4: Network Layer

- 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 interface
  - IP addresses associated with each interface

\[ 223.1.1.1 = 11011111 \ 00000001 \ 00000001 \ 00000001 \]

223 1 1 1 1
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?
**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 its 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 an “on”)
- Support for mobile users who want to join network (more shortly)

**DHCP overview:**

- host broadcasts “DHCP discover” msg
- DHCP server responds with “DHCP offer” msg
- 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
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
## 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:

- Send me anything with addresses beginning 200.23.16.0/20
- Send me anything with addresses beginning 199.31.0.0/16

Diagram:
- 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-R-Us
- Fly-By-Night-ISP
- 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
- 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 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 4-54
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, eg, 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 NATted 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 NATted 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 NATted host
2. connection to relay initiated by client
3. relaying established
Chapter 4: Network Layer

- 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 has TTL = 1
  - Second has TTL = 2, etc.
  - Unlikely port number
- When nth datagram arrives to nth router:
  - Router discards datagram
  - And sends to source an ICMP message (type 11, code 0)
  - Message includes name of router & IP address
- When ICMP message arrives, source calculates RTT
- Traceroute does this 3 times

Stopping criterion
- UDP segment eventually arrives at destination host
- Destination returns ICMP “host unreachable” packet (type 3, code 3)
- When source gets this ICMP, stops.
Chapter 4: Network Layer

- 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
IPv6

- **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
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>
<tr>
<td>payload len</td>
<td>next hdr</td>
<td>hop limit</td>
</tr>
<tr>
<td>source address</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(128 bits)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>destination address</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(128 bits)</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 the network operate with mixed IPv4 and IPv6 routers?
- Tunneling: IPv6 carried as payload in IPv4 datagram among IPv4 routers
Tunneling

Logical view:

A IPv6
B tunnel
E IPv6

Physical view:

A IPv6
B IPv4
E IPv6

IPv4
IPv4
Tunneling

Logical view:

A IPv6

B IPv6

--- tunnel ---

E IPv6

F IPv6

Physical view:

A IPv6

B IPv6

C IPv6 inside IPv4

D IPv4

E IPv6

F IPv6

Flow: X
Src: A
Dest: F
Data

Flow: X
Src: A
Dest: F
Data

Flow: X
Src: A
Dest: F
Data

Flow: X
Src: A
Dest: F
Data

A-to-B: IPv6
B-to-C: IPv6 inside IPv4
E-to-F: IPv6
Chapter 4: Network Layer

- 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

Local forwarding table

<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>

Value in arriving packet's header
Graph abstraction

Graph: $G = (N,E)$

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

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

Remark: Graph abstraction is useful in other network contexts

Example: P2P, where $N$ is set of peers and $E$ is set of TCP connections
Graph abstraction: costs

- $c(x,x') = \text{cost of link } (x,x')$
  - e.g., $c(w,z) = 5$
- cost could always be 1, or inversely related to bandwidth, or inversely related to congestion

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

Question: What’s the least-cost path between $u$ and $z$?

Routing algorithm: algorithm that finds least-cost path
Routing Algorithm classification

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

Static or dynamic?

Static:
- routes change slowly over time

Dynamic:
- routes change more quickly
  - periodic update
  - in response to link cost changes
Chapter 4: Network Layer

- 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 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),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>4,y</td>
<td>4,y</td>
<td>4,y</td>
</tr>
<tr>
<td>3</td>
<td>uxyv</td>
<td>2,u</td>
<td>3,y</td>
<td>4,y</td>
<td>4,y</td>
<td>4,y</td>
</tr>
<tr>
<td>4</td>
<td>uxyvw</td>
<td>3,y</td>
<td>4,y</td>
<td>4,y</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>uxyvwx</td>
<td>4,y</td>
<td>4,y</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Diagram:

![Diagram of the network with nodes u, v, w, x, y, z and edges with weights 1, 2, 3, 4, 5.](diagram.png)
Dijkstra’s algorithm: example (2)

Resulting shortest-path tree from u:

![Network diagram]

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., link cost = amount of carried traffic

![Diagram showing iterations of Dijkstra's algorithm]

\[\text{initially} \quad \begin{array}{cccc}
A & D & B & C \\
\downarrow & 1 & \downarrow & 0 \\
\downarrow & 0 & \downarrow & e \\
\end{array} \]

\[\text{… recompute routing} \quad \begin{array}{cccc}
A & D & B & C \\
\downarrow & 2+e & \downarrow & 1+e \ 1 \\
\downarrow & 1+e \ 0 & \downarrow & 0 \\
\end{array} \]

\[\text{… recompute} \quad \begin{array}{cccc}
A & D & B & C \\
\downarrow & 0 & \downarrow & 1+e \ 1 \\
\downarrow & 1 & \downarrow & 0 \\
\end{array} \]

\[\text{… recompute} \quad \begin{array}{cccc}
A & D & B & C \\
\downarrow & 2+e & \downarrow & e \\
\downarrow & 0 & \downarrow & 0 \\
\end{array} \]
Chapter 4: Network Layer

- 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)

Define
\[ d_x(y) := \text{cost of least-cost path from } x \text{ to } y \]

Then
\[ d_x(y) = \min_v \{ c(x,v) + d_v(y) \} \]

where min is taken over all neighbors \( v \) of \( x \)
Bellman-Ford example

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 that achieves minimum is next hop in shortest path ➔ forwarding table
Distance Vector Algorithm

- $D_x(y) =$ estimate of least cost from $x$ to $y$
- Node $x$ knows cost to each neighbor $v$: $c(x,v)$
- Node $x$ maintains distance vector $D_x = [D_x(y): y \in N ]$
- Node $x$ also maintains its neighbors’ distance vectors
  - For each neighbor $v$, $x$ maintains $D_v = [D_v(y): y \in N ]$
Distance vector algorithm (4)

**Basic idea:**
- From time-to-time, each node sends its own distance vector estimate to neighbors
- Asynchronous
- When a node \( x \) 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 (5)

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

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

At time $t_0$, $y$ detects the link-cost change, updates its DV, and informs its neighbors.

At time $t_1$, $z$ receives the update from $y$ and updates its table. It computes a new least cost to $x$ and sends its neighbors its DV.

At time $t_2$, $y$ receives $z$’s update and updates its distance table. $y$’s least costs do not change and hence $y$ does not send any message to $z$. 
Distance Vector: link cost changes

Link cost changes:
- good news travels fast
- 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: Network Layer

- 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 200 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
- Direct 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 $I$ is on the least cost path to 1c.
  - installs forwarding table entry $(x, I)$
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!
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 the job of inter-AS routing protocol!
- Hot potato routing: send packet towards closest of two routers.

Learn from inter-AS protocol that subnet $x$ is reachable via multiple gateways

Use routing info from intra-AS protocol to determine costs of least-cost paths to each of the gateways

Hot potato routing: Choose the gateway that has the smallest least cost

Determine from forwarding table the interface $I$ that leads to least-cost gateway. Enter $(x,I)$ in forwarding table
Chapter 4: Network Layer

- 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)
Chapter 4: Network Layer

- 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
RIP (Routing Information Protocol)

- distance vector algorithm
- included in BSD-UNIX Distribution in 1982
- distance metric: # of hops (max = 15 hops)

From router A to subnets:

<table>
<thead>
<tr>
<th>destination</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>
RIP advertisements

- **distance vectors**: exchanged among neighbors every 30 sec via Response Message (also called advertisement)
- each advertisement: list of up to 25 destination subnets within AS
## RIP: Example

<table>
<thead>
<tr>
<th>Destination Network</th>
<th>Next Router</th>
<th>Num. of 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>

Routing/Forwarding table in D
RIP: Example

<table>
<thead>
<tr>
<th>Dest</th>
<th>Next Router</th>
<th>Num. of hops to dest.</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>A</td>
<td>2</td>
</tr>
<tr>
<td>x</td>
<td>B</td>
<td>2</td>
</tr>
<tr>
<td>z</td>
<td>A</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>--</td>
<td>1</td>
</tr>
</tbody>
</table>

Advertisement from A to D

Routing/Forwarding table in D
RIP: Link Failure and 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
Chapter 4: Network Layer

- 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
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 router
- advertisements disseminated to entire AS (via flooding)
  - carried in OSPF messages directly over IP (rather than TCP or UDP)
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; high for real time)
- **integrated uni- and multicast** support:
  - Multicast OSPF (MOSPF) uses same topology database as OSPF
- **hierarchical** OSPF in large domains.
Hierarchical OSPF
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.
Chapter 4: Network Layer

- 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
Internet inter-AS routing: BGP

- BGP (Border Gateway Protocol): *the de facto standard*
- BGP provides each AS a means to:
  1. Obtain subnet reachability information from neighboring ASs.
  2. Propagate reachability information to all AS-internal routers.
  3. Determine “good” routes to subnets based on reachability information and policy.
- allows subnet to advertise its existence to rest of Internet: *"I am here"*
BGP basics

- pairs of routers (BGP peers) exchange routing info over semi-permanent TCP connections: BGP sessions
  - BGP sessions need not correspond to physical links.
- when AS2 advertises a prefix to AS1:
  - AS2 *promises* it will forward datagrams towards that prefix.
  - AS2 can aggregate prefixes in its advertisement
Distributing reachability info

- 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 & 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)

- when gateway router receives route advertisement, uses import policy to accept/decline.
BGP route selection

- router may learn about more than 1 route to some prefix. Router must select route.

- elimination rules:
  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 using TCP.
- 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 $C$.

Legend:
- **Provider network**:
- **Customer network**:
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- and 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: Network Layer

- 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 brdcst pkt, sends copy to all neighbors
  - Problems: cycles & broadcast storm

- Controlled flooding: node only brdcsts pkt if it hasn’t brdcst same packet before
  - Node keeps track of pkt ids already brdcsted
  - Or reverse path forwarding (RPF): only forward pkt 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 forward 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
(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
  - *source-based:* different tree from each sender to rcvrs
  - *shared-tree:* same tree used by all group members

[Diagram showing shared tree and source-based trees]
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:

```plaintext
if (mcast datagram received on incoming link on shortest path back to center)
  then flood datagram onto all outgoing links
else ignore datagram
```
Reverse Path Forwarding: example

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

[Diagram showing network with routers labeled R1 to R7, source S, and legend explaining symbols for routers with and without attached group members, and datagram forwarding status.]
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

LEGEND
- S: source
- P: prune message
- Links with multicast forwarding
- Router with attached group member
- Router with no attached group member
**Shared-Tree: Steiner Tree**

- **Steiner Tree**: minimum cost tree connecting all routers with attached group members
- Problem is NP-complete
- Excellent heuristics exist
- 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: an example

Suppose R6 chosen as center:

LEGEND

1. router with attached group member
2. router with no attached group member
3. 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 routers
  - Mbone routing done using DVMRP
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
- 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
**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: summary

- 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