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

- Transports segment from sending to receiving host
  - sending side encapsulates segments into datagrams
  - router examines IP header in rcvd datagrams and forwards
  - rcv'ing side delivers segments to transport layer

Network layer protocols run in every host and router
Key Network-Layer Functions

- **forwarding**: move packets from router’s input to appropriate router output

- **routing**: determine route taken by packets from source to dest.
  - *Routing algorithms*

- **analogy**:

- **routing**: process of planning trip from source to dest

- **forwarding**: process of getting through single interchange
Interplay between routing and forwarding

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: 0111
Connection setup

- 3rd important function in some network architectures:
  - ATM, frame relay, X.25

- Before datagrams flow, two hosts and intervening routers establish virtual connection
  - Routers get involved

- Network and transport layer cnctn service:
  - Network: between two hosts
  - 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 < 40 msec delay

Example services for a flow of datagrams:
- In-order 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>
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
- Virtual circuit (VC) network provides network-layer connection service
- Analogous to 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 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 address)
- every router on source-dest path maintains “state” for each passing VC
- link, router resources (bandwidth, buffers) may be reserved for VC
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 a VC number.
- VC number must 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: Internet

- 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 00010111 00010000 00000000</td>
<td>0</td>
</tr>
<tr>
<td>11001000 00010111 00010111 11111111</td>
<td>1</td>
</tr>
<tr>
<td>11001000 00010111 00011000 00000000 through 00011000 11111111</td>
<td>2</td>
</tr>
<tr>
<td>11001000 00010111 00011001 00000000 through 00011111 11111111</td>
<td>3</td>
</tr>
</tbody>
</table>

Otherwise, 4 billion possible entries.
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 10100001  Which interface?

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

**Internet**
- 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 network types with different characteristics
  - uniform service difficult

**ATM**
- evolved from telephony for human conversation
  - strict timing, reliability, service guarantees
- “dumb” end systems (telephones)
  - reservation, congestion feedback
  - complexity in network
- restricted network types
  - uniform service
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

Physical layer:
bit-level reception

Data link layer:
e.g., Ethernet
see chapter 5

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
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 shared bus
- **bus contention**: switching speed limited by bus bandwidth
- 1 Gbps bus, Cisco 1900: sufficient speed for access and enterprise routers (not regional or backbone)
Switching Via 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 via fabric.
- Cisco 12000: switches many Gbps through the interconnection network
Output Port

- switch
- fabric

- queuing: buffer management
- data link processing (protocol, decapsulation)
- line termination
Output port queueing

- **Buffering** when arrival rate via switch exceeds output line speed
  - causes queueing delay, or loss due to buffer overflow
- **Scheduling** discipline chooses among queued datagrams for transmission
**Input Port Queuing**

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

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

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

- ICMP protocol
  - error reporting
  - router “signaling”

Transport layer: TCP, UDP

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>32 bits</td>
</tr>
<tr>
<td>header length (bytes)</td>
<td>32 bits</td>
</tr>
<tr>
<td>&quot;type&quot; of data</td>
<td>16-bit identifier</td>
</tr>
<tr>
<td>max number remaining hops</td>
<td>time to live</td>
</tr>
<tr>
<td>(decremented at each router)</td>
<td>upper layer</td>
</tr>
<tr>
<td>upper layer protocol</td>
<td>Internet checksum</td>
</tr>
<tr>
<td>to deliver payload to</td>
<td>32 bit source IP address</td>
</tr>
<tr>
<td></td>
<td>32 bit destination IP address</td>
</tr>
<tr>
<td>Options (if any)</td>
<td>Options (if any)</td>
</tr>
<tr>
<td>data</td>
<td>data</td>
</tr>
<tr>
<td>(variable length, typically a</td>
<td>data</td>
</tr>
<tr>
<td>TCP or UDP segment)</td>
<td>data</td>
</tr>
</tbody>
</table>

**how much overhead with TCP?**

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

**E.g. timestamp, record route taken, specify list of routers to visit.**
IP Fragmentation & Reassembly

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

fragmentation: in: one large datagram out: 3 smaller datagrams
reassembly
**IP Fragmentation and Reassembly**

**Example**
- 4000 byte datagram
- **MTU** = 1500 bytes

One large datagram becomes several smaller datagrams.

- **4000 byte datagram**
- **MTU** = 1500 bytes

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>
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 address associated with each interface

<table>
<thead>
<tr>
<th>IP Address</th>
<th>Binary Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>223.1.1.1</td>
<td>11011111 00000001 00000001 00000001</td>
</tr>
<tr>
<td>223.1.1.2</td>
<td>11011111 00000001 00000001 00000001</td>
</tr>
<tr>
<td>223.1.1.3</td>
<td>11011111 00000001 00000001 00000001</td>
</tr>
<tr>
<td>223.1.1.4</td>
<td>11011111 00000001 00000001 00000001</td>
</tr>
<tr>
<td>223.1.2.1</td>
<td>11011111 00000001 00000001 00000001</td>
</tr>
<tr>
<td>223.1.2.2</td>
<td>11011111 00000001 00000001 00000001</td>
</tr>
<tr>
<td>223.1.2.9</td>
<td>11011111 00000001 00000001 00000001</td>
</tr>
<tr>
<td>223.1.3.1</td>
<td>11011111 00000001 00000001 00000001</td>
</tr>
<tr>
<td>223.1.3.2</td>
<td>11011111 00000001 00000001 00000001</td>
</tr>
<tr>
<td>223.1.3.27</td>
<td>11011111 00000001 00000001 00000001</td>
</tr>
</tbody>
</table>
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

![Diagram showing a network consisting of 3 subnets](image)
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 host get IP address?

- hard-coded by system admin in a file
  - Wintel: 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”
  (more in next chapter)
**IP addresses: how to get one?**

**Q:** How does network get subnet part of IP address?

**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:

Organization 0
200.23.16.0/23

Organization 1
200.23.18.0/23

Organization 2
200.23.20.0/23

Organization 7
200.23.30.0/23

Fly-By-Night-ISP
“Send me anything with addresses beginning 200.23.16.0/20”

ISPs-R-Us
“Send me anything with addresses beginning 199.31.0.0/16”

Internet
Hierarchical addressing: more specific routes

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

Organization 0
200.23.16.0/23

Organization 2
200.23.20.0/23

Organization 7
200.23.30.0/23

Organization 1
200.23.18.0/23

"Send me anything with addresses beginning 199.31.0.0/16 or 200.23.18.0/23"

"Send me anything with addresses beginning 200.23.16.0/20"

Fly-By-Night-ISP

ISPs-R-Us

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

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>

1: host 10.0.0.1 sends datagram to 128.119.40.186, 80

3: Reply arrives dest. address: 138.76.29.7, 5001

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

- 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
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 to facilitate quality-of-service (QoS)

**IPv6 datagram format:**
- fixed-length 40 byte header
- no fragmentation allowed
IPv6 Header (Cont)

**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></td>
<td></td>
<td></td>
</tr>
<tr>
<td>source address</td>
<td>hop limit</td>
<td></td>
</tr>
<tr>
<td>(128 bits)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>destination address</td>
<td>hop limit</td>
<td></td>
</tr>
<tr>
<td>(128 bits)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

Data

---

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
IPv6

IPv6

E
IPv6

F
IPv6

Physical view:

A
IPv6

B
IPv6

IPv4

IPv4

E
IPv6

F
IPv6
Tunneling

Logical view:

A
IPv6

B
IPv6

--- tunnel ---

E
IPv6

F
IPv6

Physical view:

A
IPv6

B
IPv6

C
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

B-to-C: IPv4
IPv6 inside IPv4
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

<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>
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, e.g., P2P, where $N$ is set of peers and $E$ is set of TCP connections.
Graph abstraction: costs

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

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 \( D(v) = \infty \)
7
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),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></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>uxyvwz</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The diagram illustrates the network with nodes u, v, w, x, y, and z, and the edges with their respective weights.
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., link cost = amount of carried traffic

![Diagrams showing Dijkstra's algorithm process](image-url)
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 $\rightarrow$ forwarding table
Distance Vector Algorithm

- $D_x(y)$ = estimate of least cost from $x$ to $y$
- Node $x$ maintains distance vector $D_x = [D_x(y): y \in N]$
- Node $x$ knows cost to each neighbor $v$: $c(x,v)$
- 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:

- Each node periodically sends its own distance vector estimate to neighbors.
- 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 due to
- local link cost change
- DV update message from neighbor

Distributed:
- a node notifies neighbors only when its DV changes
  - neighbors then notify their neighbors if needed

Each node:

1. wait for (change in local link cost of msg from neighbor)
2. recompute estimates
3. if DV to any dest has changed, notify neighbors
\[ D_x(y) = \min \{c(x,y) + D_y(y), c(x,z) + D_z(y)\} \]
\[ = \min \{2+0, 7+1\} = 2 \]

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

Poissoned 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 initially. O(n) messages whenever a link cost changes.
- **DV**: exchange between neighbors only
  - convergence time varies

Speed of Convergence
- **LS**: message flooding time plus O(n^2) algorithm
  - 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 through network
Problem

Consider the network fragment shown. x has only two attached neighbors, w and y. w has a minimum-cost path to destination u (not shown) of 5 and y has a minimum-cost path to u of 6. The complete paths from w and y to u (and between w and y) are not shown. All link costs are positive integers.

- Give x’s DV for destinations w, y, and u.
- Give a link-cost change for either c(x,w) or c(x,y) such that x will inform its neighbors of a new minimum-cost path to u as a result of executing the DV algorithm.
- Give a link-cost change for either c(x,w) or c(x,y) such that x will not inform its neighbors of a new minimum-cost path to u as a result of executing the DV algorithm.
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 protocols
- gateway router has direct link to router in another AS
Interconnected ASes

- Forwarding table is 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 for which dest is outside of AS1
  - Router should forward packet towards one of the gateway routers, but which one?

AS1 needs:
1. to learn which dests are reachable through AS2 and which through AS3
2. to propagate this reachability info to all routers in AS1

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

- Suppose AS1 learns from the inter-AS protocol that subnet $x$ is reachable from AS3 (gateway 1c) but not from 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.
- Puts in forwarding table entry $(x, I)$. 
Example: Choosing among multiple ASes

- Now suppose AS1 learns from the 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 \( x \).
  - 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 (DV)
  - OSPF: Open Shortest Path First (LS)
  - IGRP: Interior Gateway Routing Protocol (Cisco proprietary) (DV)
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 subsets:

<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 nets within AS
## RIP: Example

### Routing Table

<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 table in D
### RIP: Example

**Routing table in D**

<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 5</td>
</tr>
<tr>
<td>x</td>
<td>--</td>
<td>1</td>
</tr>
<tr>
<td>....</td>
<td>....</td>
<td>....</td>
</tr>
</tbody>
</table>

**Advertisement from A to D**

<table>
<thead>
<tr>
<th>Dest</th>
<th>Next hops</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>- 1</td>
</tr>
<tr>
<td>x</td>
<td>- 1</td>
</tr>
<tr>
<td>z</td>
<td>C 4</td>
</tr>
<tr>
<td>....</td>
<td>....</td>
</tr>
</tbody>
</table>
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

![Diagram of network layers with routing tables and forwarding information]
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 the reachability information to all routers internal to the AS.
  3. Determine “good” routes to subnets based on reachability information and policy.

- Allows a subnet to advertise its existence to rest of the Internet: “I am here”
BGP basics

- Pairs of routers (BGP peers) exchange routing info over semi-permanent TCP connections: **BGP sessions**
  - Note that BGP sessions may not correspond to physical links.
- When AS2 advertises a prefix to AS1, AS2 is *promising* it will forward any datagrams destined to that prefix towards the prefix.
  - AS2 can aggregate prefixes in its advertisement.
Distributing reachability info

- With eBGP session between 3a and 1c, AS3 sends prefix reachability info to AS1.
- 1c can then use iBGP to distribute this new prefix reach info to all routers in AS1.
- 1b can then re-advertise the new reachability info to AS2 over the 1b-to-2a eBGP session.
- When router learns about a new prefix, it creates an entry for the prefix in its forwarding table.
Path attributes & BGP routes

- When advertising a prefix, advert includes BGP attributes.
  - prefix + attributes = “route”

- Two important attributes:
  - **AS-PATH:** contains the ASs through which the advert for the prefix passed: AS 67 AS 17
  - **NEXT-HOP:** Indicates the specific internal-AS router to next-hop AS. (There may be multiple links from current AS to next-hop-AS.)

- When gateway router receives route advert,
  - uses **import policy** to accept/decline.
  - uses **export policy** to choose to re-advertise
**BGP route selection**

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

- Route selection rules after import policy filter
  1. Local preference attribute: policy decision
     - Customer > peer > provider
  2. Shortest AS-PATH
  3. Closest NEXT-HOP router: hot potato routing
  4. Additional criteria

- Export policy: valley-free routing
  - Do not advertise peer route to peer/provider
  - Do not advertise provider route to peer/provider
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: Valley-free 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: Valley-free routing policy (2)**

- A advertises to B the path AW
- B advertises to X the path BAW
- Should B advertise to C the path BAW?
  - 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 broadcast packet, sends copy to all neighbors
  - Problems: cycles & broadcast storm

- Controlled flooding: node only broadcasts packet if it hasn’t broadcast same packet before
  - Node keeps track of pkt id’s already broadcasted
  - 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

- Some node designated as 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 links between routers used
  - *source-based:* different tree from each sender to rcvrs
  - *shared-tree:* same tree used by all group members
Approaches for building mcast trees

Approaches:
- **source-based tree**: one tree per source
  - shortest path tree, 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
  - e.g., use routes computed by Dijkstra’s algorithm

![Diagram showing a network with routers and links labeled with numbers and symbols representing source, routers with attached or no attached group members, and links used for forwarding added by the algorithm.]

**Legend:**
- S: source
- Router with attached group member
- Router with no attached group member
- Link used for forwarding, i indicates order link added by the 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
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
- R1, R2, R3, R4, R5, R6, R7: routers
- links with multicast forwarding
- P: prune message
- router with attached group member
- router with no attached group member
End of Chapter 4 syllabus
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: an 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 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!”