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

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

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

Interplay between routing and forwarding
- routing algorithms
- router forwarding table
- header value output trip
  - 0100 3
  - 0101 2
  - 1000 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>Guarantees?</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</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no congestion</td>
</tr>
<tr>
<td>ATM</td>
<td>VBR</td>
<td>guaranteed</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no congestion</td>
</tr>
<tr>
<td>ATM</td>
<td>ABR</td>
<td>guaranteed</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
  - 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**

<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 00011011 00011011 11111111</td>
<td>0</td>
</tr>
<tr>
<td>11001000 00010111 00011100 00000000 through 11001000 00011011 00011111 11111111</td>
<td>1</td>
</tr>
<tr>
<td>11001000 00010111 00011001 00000000 through 11001000 00011011 00011111 11111111</td>
<td>2</td>
</tr>
<tr>
<td>11001000 00010111 00011111 11111111 through 11001000 00011011 00011011 11111111</td>
<td>3</td>
</tr>
</tbody>
</table>

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 00010100 00000000</td>
<td>0</td>
</tr>
<tr>
<td>11001000 00010111 00011000 00000000</td>
<td>1</td>
</tr>
<tr>
<td>11001000 00010111 00011011 00000000</td>
<td>2</td>
</tr>
<tr>
<td>11001000 00010111 00011111 11111111</td>
<td>3</td>
</tr>
</tbody>
</table>

Examples:

- DA: 11001000 00010111 00011011 01010000
  - Which interface?

- DA: 11001000 00010111 00011010 01010101
  - 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

Physical layer: bit-level reception
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

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 $\frac{\text{RTT} \cdot C}{N}$

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
  - RIP, OSPF, BGP
- IP protocol
  - Addressing conventions
  - Datagram format
  - Packet handling conventions
- ICMP protocol
  - Error reporting
  - Router signaling

IP datagram format

- IP protocol version number
- Header length (bytes)
- Type of service
- Total length (bytes)
- Identification
- Flags
- Fragment offset
- Time to live
- Protocol
- Header checksum
- Source IP address
- Destination IP address
- Options (if any)

IP Fragmentation & Reassembly

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

Example

- 4000 byte datagram
- MTU = 1500 bytes

- One large datagram becomes several smaller datagrams

- 1480 bytes in data field

- Offset = 1480/8
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

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

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.

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 00101000 00000000
200.23.16.0/23
IP addresses: how to get one?

Q: How does a host get IP address?

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

DHCP: Dynamic Host Configuration Protocol

Goal: allow host to dynamically obtain its IP address from network server when it joins network

Can renew its lease on address in use

 Allows reuse of addresses (only hold address while connected on "off")

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 client-server scenario

IP addresses: how to get one?

Q: How does network get subnet part of IP addr?

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

ISP's block          11001000  00010111  0001  00000000    200.23.16.0/20
Organization 0    11001000  00010111  00010000  00000000    200.23.16.0/23
Organization 1    11001000  00010111  00010010  00000000    200.23.18.0/23
Organization 2    11001000  00010111  00010100  00000000    200.23.20.0/23
...                                          0elipsis..                                   0elipsis.                0elipsis.
Organization 7    11001000  00010111  00011110  00000000    200.23.30.0/23

Hierarchical addressing: route aggregation

Hierarchical addressing allows efficient advertisement of routing information:

Send me anything with addresses beginning

200.23.16.0/20

200.23.18.0/23

200.23.20.0/23

200.23.30.0/23

Fly-By-Night-ISP

ISP-R-U-S

Internet
Hierarchical addressing: more specific routes

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

Send me anything with addresses

beginning 200.23.16.0/23 or 200.23.18.0/23

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

IP addressing: the last word...

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).

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

- 16-bit port-number field:
  - 60,000 simultaneous connections with a single LAN-side address!
- NAT is controversial:
  - routers should only process up to layer 3
  - violates end-to-end argument
    - NAT possibility must be taken into account by app designers, e.g., P2P applications
  - address shortage should instead be solved by IPv6

NAT traversal problem

- 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 2500

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

Chapter 4: Network Layer

- 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

<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
  - 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

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
- 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

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

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:
IPv6 IPv6 IPv6 IPv6

Physical view:
IPv6 IPv6 IPv6 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
4.5 Routing algorithms
4.6 Routing in the Internet
4.7 Broadcast and multicast routing

Graph abstraction

Graph: $G = (N,E)$

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

Interplay between routing, forwarding

Routing algorithm: algorithm that finds least-cost path

Graph abstraction: costs

Cost of path $(x_1, x_2, \ldots, x_p) = c(x_1,x_2) + c(x_2,x_3) + \ldots + c(x_{p-1},x_p)$
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
4.5 Routing algorithms
4.6 Routing in the Internet
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; c = ∞ 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) = ∞
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) ) /* new cost to v is either old cost to v or known shortest path cost to w plus cost from w to v */
13 until all nodes in N'

Dijsktra'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></td>
<td>u: 2,u: 5,u: 1,u: ∞</td>
<td>w: ∞</td>
<td>v: ∞</td>
<td>x: 2</td>
<td>y: 2</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>u: 2,u: 4</td>
<td>x: 2</td>
<td>y: 2</td>
<td>z: 2</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>w: 2</td>
<td>x: 3</td>
<td>y: 3</td>
<td>z: 3</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>x: 3</td>
<td>y: 4</td>
<td>z: 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>y: 4</td>
<td>z: 4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>z: 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Dijsktra's algorithm: example (2)

Resulting shortest-path tree from u:

Resulting forwarding table in u:

destination link
v (u,v)
x (u,x)
y (u,y)
w (u,x)
z (u,z)
Dijkstra's Algorithm, Discussion

Algorithms Complexity:
- 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

Distance Vector Algorithm

Bellman-Ford Equation (Dynamic Programming)
Define:
- \( d_x(y) \): cost of least-cost path from \( x \) to \( y \)

Then:
- \( d_x(y) = \min \{ c(x, v) + d_v(y) \} \)

where \( \min \) is taken over all neighbors \( v \) of \( x \)

Bellman-Ford Example

Clearly, \( d_z(z) = 5 \), \( d_x(z) = 3 \), \( d_w(z) = 3 \)

B-F equation says:
- \( d_z(z) = \min \{ c(u, v) + d_v(z), c(u, x) + d_x(z), c(u, w) + d_w(z) \} \)
- \( = \min \{ 2 + 5, 1 + 3, 5 + 3 \} = 4 \)

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 \( x \) 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 \( D_x = [D_x(y) : y \in N] \)

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:
  - \( D_x(y) \leftarrow \min \{ c(x, v) + D_v(y) \} \) 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)
- recomputes estimates
- if DV to any dest has changed, notify neighbors

---

**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

- 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.
- 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, $O(nE)$ msgs sent
- **DV:** exchange between neighbors only
  - convergence time varies

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

**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

**Network Layer:** 4-89
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!

Intra-AS Routing

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

RIP (Routing Information Protocol)

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

**Routing/Forwarding 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</td>
</tr>
<tr>
<td>x</td>
<td>--</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<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

### 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 data base as OSPF
  - hierarchical OSPF in large domains.

**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
**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:
  - flooding: when node receives brdcst pckt, 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 pckt ids already brdcsted
    - Or reverse path forwarding (RPF): only forward pckt if it arrived on shortest path between node and source
  - spanning tree
    - No redundant packets received by any node

**In-network duplication**

- 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

Network Layer 4)

A
B
G
D
E
c
F
1
2
3
4
5
(a) Stepwise construction of spanning tree  (b) Constructed spanning tree

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:

  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
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

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

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

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: an example

Suppose R6 chosen as center:

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 if 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

**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!"

PIM - Sparse Mode

- Computes leaf-node router
- All data multicast from rendezvous point
- "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