Chapter 4: Network Layer

Chapter goals:

- understand principles behind network layer services:
  - routing (path selection)
  - dealing with scale
  - how a router works
  - advanced topics: IPv6, mobility
- instantiation and implementation in the Internet

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

Key Network-Layer Functions

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

- **routing**: determine route taken by packets from source to destination

  *analogy*:

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

Network layer

- transport segment from sending host to receiving host
  - on sending side, encapsulate segments into datagrams
  - on receiving side, deliver segments to transport layer
- network layer protocols in every host, router
- Router examines header fields in all IP datagrams passing through it
### Interplay between routing and forwarding

How is the routing table constructed?

#### Network Layer 4-5

![Routing Algorithm Diagram](Image)

- **Routing Algorithm**
  - Value in arriving packet's header:
    - 0100
    - 0101
    - 0111
    - 1001

### Connection setup

- A 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 connection service:
  - Network: between two hosts
  - Transport: between two processes

### Network service model

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

**Example services for individual datagrams:**
- Guaranteed delivery
- Guaranteed delivery with less than 40 msec delay

**Example services for a flow of datagrams:**
- In-order datagram delivery
- Guaranteed minimum bandwidth to flow
- Restrictions on changes in inter-packet spacing

### Network layer service models:

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

“source-to-destination path behaves much like a 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-destination path maintains “state” for each passing connection
- link, router resources (bandwidth, buffers) may be allocated to 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

- Packets 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>2</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... on the other hand**

- **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**
  - Different packets between same source-destination pair may take different paths

**Forwarding table**

<table>
<thead>
<tr>
<th>Destination Address Range (32 bits)</th>
<th>Link Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>11001000 00010111 00010000 00000000 through 0</td>
<td></td>
</tr>
<tr>
<td>11001000 00010111 00010111 11111111 through 1</td>
<td></td>
</tr>
<tr>
<td>11001000 00010111 00011000 00000000 through 11001000 00010111 00011000 11111111 2</td>
<td></td>
</tr>
<tr>
<td>11001000 00010111 00011001 00000000 through 11001000 00010111 00011001 00011111 3</td>
<td></td>
</tr>
<tr>
<td>otherwise</td>
<td></td>
</tr>
</tbody>
</table>

4 billion possible entries

Network Layer 4-13

Network Layer 4-14

Network Layer 4-15

Network Layer 4-16
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 link types
  - different characteristics
  - uniform service difficult

ATM
- evolved from telephony
- human conversation:
  - strict timing, reliability requirements
  - need for guaranteed service
- "dumb" end systems
  - telephones
  - complexity inside network

Chapter 4: Network Layer

4. 1 Introduction
4.2 Virtual circuit and datagram networks
4.3 What's inside a router
4.4 IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP
  - IPv6
4.5 Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing
4.6 Routing in the Internet
  - RIP
  - OSPF
  - BGP
4.7 Broadcast and multicast routing

Router Architecture Overview

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

Input Port Functions

Decentralized switching:
- given datagram dest., lookup output port using forwarding table in input port memory
- goal: complete input port processing at "line speed"
- queuing: if datagrams arrive faster than forwarding rate into switch fabric
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
- 1 Gbps bus, Cisco 1900: sufficient speed for access and enterprise routers (not regional or backbone)

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 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!

**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:

- **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**

**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>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP protocol version</td>
<td>8</td>
<td>Number of protocol version</td>
</tr>
<tr>
<td>header length (bytes)</td>
<td>16</td>
<td>&quot;type&quot; of data</td>
</tr>
<tr>
<td>&quot;type&quot; of data</td>
<td>8</td>
<td>Max number of data remaining hops (decremented at each router)</td>
</tr>
<tr>
<td>upper layer protocol</td>
<td>16</td>
<td>32-bit source IP address</td>
</tr>
<tr>
<td>to deliver payload</td>
<td></td>
<td>32-bit destination IP address</td>
</tr>
<tr>
<td>Options (if any)</td>
<td></td>
<td>E.g. timestamp, record route taken, specify list of routers to visit.</td>
</tr>
<tr>
<td>data</td>
<td>variable length, typically a TCP or UDP segment</td>
<td></td>
</tr>
</tbody>
</table>

**IP Fragmentation & Reassembly**

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

**IP Fragmentation**

- fragmentation: in: one large datagram out: 3 smaller datagrams

---

**Network Layer 4-29**

**Network Layer 4-30**

**Network Layer 4-31**

**Network Layer 4-32**
IP Fragmentation and Reassembly

Example
- 4000 byte datagram
- MTU = 1500 bytes

One large datagram becomes several smaller datagrams

<table>
<thead>
<tr>
<th>Offset</th>
<th>Fragment Number</th>
<th>Data Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1480 bytes</td>
</tr>
<tr>
<td>185</td>
<td>1</td>
<td>1500 bytes</td>
</tr>
<tr>
<td>370</td>
<td>2</td>
<td>1040 bytes</td>
</tr>
</tbody>
</table>

Chapter 4: Network Layer

- 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 may have multiple interfaces
  - An IP address is associated with each interface

Example:
- 223.1.1.1 = 11011111 00000001 00000001 00000001

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.
**Subnets**

How many?

- 223.1.1.1
- 223.1.1.2
- 223.1.1.3
- 223.1.1.4
- 223.1.2.1
- 223.1.2.2
- 223.1.2.3
- 223.1.2.4
- 223.1.3.1
- 223.1.3.2
- 223.1.3.3
- 223.1.3.4
- 223.1.4.1
- 223.1.4.2
- 223.1.4.3
- 223.1.4.4
- 223.1.5.1
- 223.1.5.2
- 223.1.5.3
- 223.1.5.4
- 223.1.6.1
- 223.1.6.2
- 223.1.6.3
- 223.1.6.4
- 223.1.7.1
- 223.1.7.2
- 223.1.7.3
- 223.1.7.4
- 223.1.8.1
- 223.1.8.2
- 223.1.8.3
- 223.1.8.4
- 223.1.9.1
- 223.1.9.2
- 223.1.9.3
- 223.1.9.4

**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

```
subnet part: 11001000 00010111 00010000 00000000
host part: 00010000 00000000
IP address: 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 a server
  - "plug-and-play"
**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 00010100 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>Organization 7</td>
<td>11001000 00010111 00011110 00000000</td>
<td>200.23.30.0/23</td>
</tr>
</tbody>
</table>

**Hierarchical addressing: route aggregation**

Hierarchical addressing allows efficient advertisement of routing information:

- Organization 0
  - 200.23.16.0/23
- Organization 1
  - 200.23.18.0/23
- Organization 2
  - 200.23.20.0/23
- Organization 7
  - 200.23.30.0/23

**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
Motivation: local network uses just one IP address as far as outside world is concerned:
- no need to be allocated range of addresses from ISP: just one IP address is used 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 with (NAT IP address, new port #)
    ... remote clients/servers will respond using (NAT IP address, new port #) as destination addr.
  - **remember (in NAT translation table)**
    - every (source IP address, port #) to (NAT IP address, new port #) translation pair

- **incoming datagrams:**
  - replace (NAT IP address, new port #) in dest fields of every incoming datagram with corresponding (source IP address, port #) stored in NAT table

1: host 10.0.0.1 sends datagram to 128.119.40, 80
2: NAT router changes datagram source addr from 10.0.0.1, 3345 to 138.76.29.7, 5001, updates table
3: Reply arrives dest. address: 138.76.29.7, 5001
4: NAT router changes datagram dest addr from 138.76.29.7, 5001 to 10.0.0.1, 3345
NAT: 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

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>

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

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

- 4.1 Introduction
- 4.2 Virtual circuit and datagram networks
- 4.3 What's inside a router
- 4.4 IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP
  - IPv6
- 4.5 Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing
- 4.6 Routing in the Internet
  - RIP
  - OSPF
  - BGP
- 4.7 Broadcast and multicast routing

IPv6

- Initial motivation: 32-bit address space soon to be completely allocated.
- Additional motivation:
  - header format helps speed processing/forwarding
  - header changes to facilitate QoS

IPv6 datagram format:
- fixed-length 40 byte header
- no fragmentation allowed

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

---

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 and forwarding

- **Tunneling**: IPv6 carried as payload in IPv4 datagram among IPv4 routers

---

Network Layer 4-57

Network Layer 4-58

Network Layer 4-59

Network Layer 4-60
**Network layer functions**

- transport packet from sending to receiving hosts
- network layer protocols in every host, router

Three important functions:

- **path determination**: route taken by packets from source to dest. (Routing algorithms)
- **switching**: move packets from router's input to appropriate router output
- **call setup**: some network architectures require router call setup along path before data flows

**Network service model**

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

- guaranteed bandwidth?
- preservation of inter-packet timing (no jitter)?
- loss-free delivery?
- in-order delivery?
- congestion feedback to sender?

The most important abstraction provided by network layer:

- virtual circuit
- datagram?

**Virtual circuits**

“source-to-destination path behaves much like telephone circuit”

- performance-wise
- network actions along source-to-destination path

- call setup, teardown for each call before data can flow
- each packet carries VC identifier (not destination host OD)
- every router on source-destination path, s, maintains “state” for each passing connection
  - transport-layer connection only involved two end systems
- link, router resources (bandwidth, buffers) may be allocated to VC
  - to get circuit-like performance

**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: the Internet model

- No call setup at network layer
- Routers: no state about end-to-end connections
  - No network-level concept of "connection"
- Packets typically routed using destination host ID
  - Packets between same source-destination pair may take different paths

Network layer service models:

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

- Internet model being extended: Intserv, Diffserv
  - Chapter 6

Routing

Routing protocol

Goal: determine "good" path (sequence of routers) thru network from source to dest.

Graph abstraction for routing algorithms:

- Graph nodes are routers
- Graph edges are physical links
  - Link cost: delay, $, cost, or congestion level

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 link types
  - Different characteristics
  - Uniform service difficult

ATM

- Evolved from telephony
- Human conversation:
  - Strict timing, reliability requirements
  - Need for guaranteed service
- "Dumb" end systems
  - Telephones
  - Complexity inside network
**Routing Issues**

- **Key question:** how are routing table entries determined / updated?
  - **who** determines table entries?
  - **what** info used in determining table entries?
  - **when** do routing table entries change?
  - **where** is routing info stored?
  - **how** can you control table size?
  - **why** are routing tables determined a particular way. What is the theoretical basis?

Answer these and we are done!

---

**Routing Issues...** (more)

- **scalability:** must be able to support large numbers of hosts, routers, networks
- **adapt** to changes in topology or significant changes in traffic, **quickly and efficiently**
  - self-healing: little or no human intervention
- **route selection** may depend on different criteria
- **performance:** "choose route with smallest delay"
- **policy:** "choose a route that doesn’t cross a government network" (equivalently: "let no non-government traffic cross this network")

---

**Graph abstraction**

Graph: \( G = (N,E) \)

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

Remark: Graph abstraction is useful in other network contexts

Example: P2P, where \( N \) is set of peers and \( E \) is set of TCP connections

**Graph abstraction: costs**

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

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

**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

Dynamic:
- routes change more quickly
  - periodic update
  - in response to link cost changes

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. **Loop**
   8. find \( w \) not in \( N’ \) such that \( D(w) \) is a minimum
   9. add \( w \) to \( N’ \)
   10. update \( D(v) \) for all \( v \) adjacent to \( w \) and not in \( N’ \)
   11. \( D(v) = \min(D(v), D(w) + c(w,v)) \)
   12. /* new cost to \( v \) is either old cost to \( v \) or known
   13. shortest path cost to \( w \) plus cost from \( w \) to \( v \) */
   14. 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>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>1</td>
<td>$ux$</td>
<td>2, $u$</td>
<td>4, $x$</td>
<td>2, $x$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>2</td>
<td>$uxy$</td>
<td>2, $u$</td>
<td>3, $y$</td>
<td>4, $y$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>$uxyv$</td>
<td>3, $y$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>$uxyw$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>$uxywz$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Link-State Routing

- **flooding**: router sends out information on all of its ports; its neighbors then send out information on all of their ports, etc.... Eventually every router (no exceptions) receives a copy of the same information

- **Link State**: each router periodically shares its knowledge of its neighborhood with all routers in the network

Link-state routing

**Link State Packet**: When a router floods the network with information about its neighborhood, it is said to be *advertising*. The basis of this advertising is a short packet called a **link state packet (LSP)**, usually containing 4 fields:

- ID of advertiser,
- ID of destination network affected,
- the cost,
- the ID of the neighbor router

Link-State Routing Algorithms

**How get information about neighbors?**

- Periodically send them a short greeting packet
  - responds: assume ok and functioning well
  - no response: assume a change and alerts entire network in its next LSP
  - greeting pkt: small and uses negligible resources (unlike DV routing tables)

**Initialization**

Each router comes up and sends greeting packet to its neighbors; prepares a LSP and floods the network
**Link State Database:**
- Every router receives every LSP and puts the information into a LS database
  - every router builds the same database (as all receive the same info)
- To calculate the routing information, each router applies an algorithm (Dijkstra's Algorithm) to calculate the shortest path between two points on a network of arcs and nodes
  - nodes = networks or routers
  - arcs = connections between router and network with costs applied only to the router --> network arc
  - network --> router are always 0 cost

**Dijkstra's Algorithm**

1. **Initialization:**
   1. \( N = \{ A \} \)
   2. for all nodes \( v \)
   3. if \( v \) adjacent to \( A \)
   4. then \( D(v) = c(A,v) \)
   5. else \( D(v) = \text{infinity} \)

2. **Loop**
   1. find \( w \) not in \( N \) such that \( D(w) \) is a minimum
   2. add \( w \) to \( N \)
   3. update \( D(v) \) for all \( v \) adjacent to \( w \) and not in \( N \):
      \[ D(v) = \min(D(v), D(w) + c(w,v)) \]
   4. /* new cost to \( v \) is either old cost to \( v \) or known shortest path cost to \( w \) plus cost from \( w \) to \( v \) */
   5. until all nodes in \( N \)

**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

- Routing Table
  Each router now uses the shortest path tree to construct its routing table - which are both different for each router
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 (1)

Bellman-Ford Equation (Dynamic Programming)
Define
\[ d_x(y) := \text{cost of least-cost path from } x \text{ to } y \]
Then
\[ d_x(y) = \min \{ c(x,v) + d_v(y) \} \]
where \( \min \) is taken over all neighbors \( v \) of \( x \)

Bellman-Ford example (2)

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

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

Node that achieves minimum is next hop in shortest path forwarding table

Distance Vector Algorithm (3)

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

**Basic idea:**
- Each node periodically sends its own distance vector estimate to neighbors.
- When a node $x$ receives a new DV estimate from a neighbor, it updates its own DV using the Bellman-Ford 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 $D_x(y)$ converges to the actual least cost $d_x(y)$.

**Distance vector algorithm**
- Based on Bellman-Ford algorithm.
- Used in many routing protocols: Internet BGP, ISO IDRP, Novell IPX, original ARPAnet.

**Algorithm (at node X):**

**Initialization:** for all adjacent nodes $v$:
- $D(^*,v) = \infty$
- $D(v,v) = c(X,v)$
- Send shortest path cost to each destination to neighbors.

**Loop:** execute distributed topology update algorithm forever.

*This is the hardest component*

**Distance Vector Routing**
- Each router shares its knowledge about the entire network by sending all of its collected knowledge to its neighbors.
  - At the outset, this may be very sparse but it is all sent.
- Each router periodically sends its information only to those routers to which it has a direct connection - this info. is used by neighbors to update their own information.
- Roughly, every 30 seconds each router sends its information to its neighbors - whether or not network has changed since last exchange.
  - Recall Link State updates ~every 30 minutes.
- **Distance Vector:** each router periodically shares its knowledge of the entire network with each of its neighbors.

**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 of msg from neighbor)
- **recompute** estimates
- **notify** neighbors if DV to any dest has changed.
**Update Algorithm at Node X:**

1. wait (until I see a link cost change to neighbor Y or until I receive an update from neighbor W)

2. if (c(X,Y) changes by delta) {
   /* change my cost to my neighbor Y */
   change all column-Y entries in distance table by delta
   if this changes my least cost path to Z
   send update wrt Z, D^X(Z,*) , to all neighbors
   }

3. if (update received from W wrt Z) {
   /* shortest path from W to some Z has changed */
   D^X(Z,W) = c(X,W) + D^W(Z,*)
   if this changes my least cost path to Z
   send update wrt Z, D^X(Z,*) , to all neighbors
   }

---

### Distance Table: example

```
Distance table:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7</td>
<td>6</td>
<td>4</td>
</tr>
</tbody>
</table>
```

```
Cost to destination via:

<table>
<thead>
<tr>
<th>D^E()</th>
<th>A</th>
<th>B</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>14</td>
<td>5</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>C</td>
<td>4</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>D</td>
<td>4</td>
<td>11</td>
<td>2</td>
</tr>
</tbody>
</table>
```

```
Distance table gives routing table

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>z</th>
<th>x</th>
<th>y</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>cost to destination via</td>
<td>cost to destination via</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>1</td>
<td>14</td>
<td>5</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>8</td>
<td>5</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>C</td>
<td>4</td>
<td>9</td>
<td>4</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>D</td>
<td>4</td>
<td>11</td>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>
```

```
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(y), c(x,z) + D_z(z))
= \min(2+0, 7+1) = 3
```

---

**Network Layer 4-93**

**Network Layer 4-94**

**Network Layer 4-95**

**Network Layer 4-96**
Distance Vector Algorithm (cont.):

8 loop
9 wait (until I see a link cost change to neighbor V
10 or until I receive update from neighbor V)
11
12 if (c(X,V) changes by d)
13 /* change cost to all dest's via neighbor V by d */
14 /* note: d could be positive or negative */
15 for all destinations y: D^X(y,V) = D^X(y,V) + d
16
17 else if (update received from V wrt destination Y)
18 /* shortest path from V to some Y has changed */
19 /* V has sent a new value for its min_{w} D^V(Y,w) */
20 /* call this received new value is "newval" */
21 for the single destination y: D^X(Y,V) = c(X,V) + newval
22
23 if we have a new min_{w} D^X(Y,w) for any destination Y
24 send new value of min_{w} D^X(Y,w) to all neighbors
25
26 forever

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 t₀, y detects the link-cost change, updates its DV, and informs its neighbors.

At time t₀, 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 t₀, 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.

Routing Table
- At startup, router’s knowledge is sparse - knows it is connected to some number of LANs and knows ID of each station.
- In most systems, a station port ID and network ID share the same prefix
  ➔ router can discover to which nets it is connected by examining its own logical addresses (has as many such addresses as it has connected ports)
- Routing Table: at least 3 pieces of info:
  - Network ID, cost, ID of next router
- so for the diagram: initial table at A is
  » 14 1 -
  » 23 1 -
  » 78 1 -
- only dest. at startup are those attached directly ⇒ blank in 3rd column
- no multihop destinations identified yet ⇒ no next routers
- These basic tables are sent out on network
Routing Table

When A receives info. table from B, it sees that B can get to nets 55 and 14; A knows B is its neighbor so if A adds 1 more hop to all costs shown in B table, the sum will be cost to A of getting to these other networks.

<table>
<thead>
<tr>
<th>A old table</th>
<th>combined</th>
<th>A - new table</th>
</tr>
</thead>
<tbody>
<tr>
<td>14 1 -</td>
<td>14 2 B</td>
<td>14 1 -</td>
</tr>
<tr>
<td>23 1 -</td>
<td>23 1 -</td>
<td>23 1 -</td>
</tr>
<tr>
<td>78 1 -</td>
<td>55 2 B</td>
<td>55 2 B</td>
</tr>
<tr>
<td></td>
<td>78 1 -</td>
<td>78 1 -</td>
</tr>
</tbody>
</table>

\[ \begin{pmatrix} 14 & 1 & - \\ 23 & 1 & - \\ 78 & 1 & - \end{pmatrix} + \text{one hop} = \begin{pmatrix} 14 & 2 & B \\ 23 & 1 & - \\ 55 & 2 & B \\ 78 & 1 & - \end{pmatrix} \]

Distance table:
- per-node table recording cost to all other nodes via each of its neighbors.
- $D^E(A,B)$ gives currently computed minimum cost from E to A given that first node on path is B.
  - $D^E(A,B) = c(E,B) + \min DB(A,*)$
  - $\min D^E(A,*)$ gives E’s minimum cost to A
- routing table derived from distance table
- example: $D^E(A,B) = 14$ (note: not 15!)
- example: $D^E(C,D) = 4$, $D^E(C,A) = 6$

Distance Vector Routing

- Packet Cost: Both LS and DV are lowest cost algorithms.
  - LS: cost refers to hop count
  - DV: cost refers a weighted value based on number of factors - security levels, traffic, state of the link,...
  - costs may well be different for the same 2 networks

- In determining the route, cost of a hop is applied to each packet as it leaves a router and enters a network (outbound cost)
  - costs are applied only to routers and not any other station on a network
  - costs are applied as packet leaves NOT as it enters a router
  - most networks are broadcast networks - when a packet enters a network, every station, including the router can pick it up - so don’t assign cost of going from network to router

Comparison of LS and DV algorithms

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

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

Robustness: what happens if router malfunctions?
- LS:
  - node can advertise incorrect link cost
  - each node computes only its own table
- DV:
  - DV node can advertise incorrect path cost
  - each node’s table used by others
    - error propagate thru network

Distance table:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>14</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>7</td>
<td>8</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>6</td>
<td>9</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>4</td>
<td>11</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>
Routing Protocol Requirements

- Minimize routing table space: keep as small as possible - reduces overhead involved with sharing information
- Minimize control messages
- Robust: protect against corruption of tables; periodically run consistency checks, add sequence numbers
- Use optimal paths

% netstat -rn
Routing tables

<table>
<thead>
<tr>
<th>Destination</th>
<th>Gateway</th>
<th>Flags</th>
<th>Refs</th>
<th>Use</th>
<th>Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>default</td>
<td>129.63.8.254</td>
<td>UGS</td>
<td>150</td>
<td>118590671</td>
<td>ee0</td>
</tr>
<tr>
<td>127.0.0.1</td>
<td>127.0.0.1</td>
<td>UHL</td>
<td>146</td>
<td>31991543</td>
<td>lo0</td>
</tr>
<tr>
<td>129.63.8/24</td>
<td>129.63.8.51</td>
<td>U</td>
<td>59</td>
<td>94190467</td>
<td>ee0</td>
</tr>
<tr>
<td>129.63.8.51</td>
<td>129.63.8.51</td>
<td>UHL</td>
<td>5</td>
<td>151087</td>
<td>ee0</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
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

Gateway router
- Direct link to router in another AS

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

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 destination is outside of AS1
  - Router should forward packet towards one of the gateway routers, but which one?

AS1 needs:
1. to learn which destinations 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 AS’s**

- 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 $x$.

- This is also the job on inter-AS routing protocol.

- **Hot potato routing**: send packet towards closest of two routers.

---

**Chapter 4: Network Layer**

- 4.1 Introduction
- 4.2 Virtual circuit and datagram networks
- 4.3 What’s inside a router
- 4.4 IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP
  - IPv6
- 4.5 Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing
- 4.6 Routing in the Internet
  - RIP
  - OSPF
  - BGP
- 4.7 Broadcast and multicast routing

---

**Intra-AS Routing**

- Also known as Interior Gateway Protocols (IGP)
- Most common Intra-AS routing protocols:
  - RIP: Routing Information Protocol
  - OSPF: Open Shortest Path First
  - IGRP: Interior Gateway Routing Protocol (Cisco proprietary)
Chapter 4: Network Layer

- 4.1 Introduction
- 4.2 Virtual circuit and datagram networks
- 4.3 What's inside a router
- 4.4 IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP
  - IPv6
- 4.5 Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing
- 4.6 Routing in the Internet
  - RIP
  - OSPF
  - BGP
- 4.7 Broadcast and multicast routing

RIP (Routing Information Protocol)

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

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

<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

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

Routing table in D

RIP: Link Failure and Recovery

If no advertisement heard after 180 sec --> neighbor/link declared dead
- routes via neighbor invalidated
- new advertisements sent to neighbors
- neighbors in turn send out new advertisements (if tables changed)
- link failure info quickly propagates to entire net
- poison reverse used to prevent ping-pong loops (infinite distance = 16 hops)

RIP Table processing

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

```
NAME
route-d - Manages network routing tables

SYNOPSIS
/usr/sbin/routed [-q | -o] [-g] [-d] [-t] [logfile]

The routed daemon manages the network routing tables.

FLAGS

-d Enables additional debugging information, such as bad packets received, to be logged. The routed daemon remains under control of the host that started it; therefore, an interrupt from the controlling host stops the routed process.

-g Causes the routing daemon to run on a gateway host. This flag is used on internetwork routers to offer a route to the default destination.
```
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 do 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, advertising includes BGP attributes.
  - prefix + attributes = “route”
- Two important attributes:
  - AS-PATH: contains the ASs through which the advertisement 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 advertisement, it 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 customers (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

Network Layer 4-137

**BGP 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!

Network Layer 4-138

---

**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 4-139
Figure 4.39  Source-duplication versus in-network duplication. (a) source duplication, (b) in-network duplication

Figure 4.40: Reverse path forwarding

Figure 4.41: Broadcast along a spanning tree

Figure 4.42: Center-based construction of a 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

**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

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:

```
R1  3  R4
R2
R3  1
R5
R6
R7
```

**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:**
  - fewer 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

![Multicast Diagram](image)
**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!"

**Network Layer: summary**

What we've covered:
- network layer services
- routing principles: link state and distance vector
- hierarchical routing
- IP
- Internet routing protocols RIP, OSPF, BGP
- what's inside a router?
- IPv6

Next stop: the Data Link Layer!