Part 4: Network Layer

CSE 3461/5461
Reading: Chapter 4, Kurose and Ross
Part 4: Outline

- **Overview and Network Layer Services**
- **Data Plane:** What’s Inside a Router?
- **Data Plane:** Internet Protocol (IP) and Addressing (IPv4, IPv6)
- **Control Plane:** Routing Algorithms: Link-State and Distance-Vector
- **Control Plane:** Internet Routing Protocols:
  - Intra-Domain
  - Inter-Domain
- **Control Plane:** Multicast and Anycast Routing
Network Layer Functions

- Transport packet from sending to receiving hosts
- Network layer protocols in *every* host, router

**Three important functions:**
- **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
- **Path determination:** route taken by packets from source to dest. *routing algorithms*
Two Key Network-Layer Functions

- **Forwarding**: move packets from router’s input to appropriate router output *(data plane)*
- **Routing**: determine route taken by packets from source to destination *(control plane)*
  - *routing algorithms*

**Analogy:**

- **Routing**: process of planning trip from source to destination
- **Forwarding**: process of getting through single interchange
Interplay Between Routing and Forwarding

Routing Algorithm determines end-end-path through network (control plane)

Forwarding table determines local forwarding at this router (data plane)

<table>
<thead>
<tr>
<th>Header Value</th>
<th>Output Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>0100</td>
<td>3</td>
</tr>
<tr>
<td>0101</td>
<td>2</td>
</tr>
<tr>
<td>0111</td>
<td>2</td>
</tr>
<tr>
<td>1001</td>
<td>1</td>
</tr>
</tbody>
</table>

Value in arriving packet’s header
Connection Setup (Control Plane)

• 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 layer: between two hosts (may also involve intervening routers in case of VCs)
  – Transport layer: between two processes on end systems
Network Service Model (1)

**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 ms delay

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

• *Datagram* network provides network-layer *connectionless* service
• *Virtual-circuit* network provides network-layer *connection* service
• Analogous to TCP/UDP connection-oriented, connectionless transport-layer services, but:
  – *Service*: host-to-host
  – *No choice*: network provides one or the other
  – *Implementation*: in network core
Virtual Circuits

- “Source-to-dest path behaves much like telephone circuit”
  - Performance-wise
  - Network actions along source-to-dest path

• Call setup, teardown for each call before data can flow
• Each packet carries VC identifier (not destination host OD)
• *Every* router on source-dest path *s* maintain “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
• Multi-protocol label switching (MPLS) is inspired by VCs to implement more flexible routing (details in “link layer” slides)
Virtual Circuits: Signaling Protocols

- Used to setup, maintain, and teardown VC
- Used in ATM, frame-relay, X.25
- Not used in today’s Internet
Datagram Networks: 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-dest pair may take different paths
Data Plane: Datagram Forwarding Table (1)

Routing Algorithm

### Local Forwarding Table

<table>
<thead>
<tr>
<th>Header Value</th>
<th>Output Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address range 1</td>
<td>3</td>
</tr>
<tr>
<td>Address range 2</td>
<td>2</td>
</tr>
<tr>
<td>Address range 3</td>
<td>2</td>
</tr>
<tr>
<td>Address range 4</td>
<td>1</td>
</tr>
</tbody>
</table>

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

IP address in arriving packet’s header
Data Plane: Datagram Forwarding Table (2)

<table>
<thead>
<tr>
<th>Destination Address Range</th>
<th>Link Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>11001000 00010111 00010000 00000000 11001000 00010111 00010111 11101111</td>
<td>0</td>
</tr>
<tr>
<td>11001000 00010111 00010111 00011000 00000000 11001000 00010111 00011000 11111111</td>
<td>1</td>
</tr>
<tr>
<td>11001000 00010111 00011001 00000000 11001000 00010111 00011111 11111111</td>
<td>2</td>
</tr>
<tr>
<td>otherwise</td>
<td>3</td>
</tr>
</tbody>
</table>

Q: But what happens if ranges don’t divide up so nicely?
Longest Prefix Matching

When looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address.

<table>
<thead>
<tr>
<th>Destination Address Range</th>
<th>Link Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>11001000 00010111 00010*** ***********</td>
<td>0</td>
</tr>
<tr>
<td>11001000 00010111 00011000 ***********</td>
<td>1</td>
</tr>
<tr>
<td>11001000 00010111 00011*** ***********</td>
<td>2</td>
</tr>
<tr>
<td>otherwise</td>
<td>3</td>
</tr>
</tbody>
</table>

**Examples:**
- Dest. Addr.: 11001000 00010111 00010110 10100001
- Dest. Addr.: 11001000 00010111 00011000 10101010
Network Layer Service Models

<table>
<thead>
<tr>
<th>Network Architecture</th>
<th>Service Model</th>
<th>Guarantees?</th>
<th>Congestion Feedback</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Bandwidth</td>
<td>Loss</td>
</tr>
<tr>
<td>Internet</td>
<td>Best effort</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>ATM</td>
<td>CBR*</td>
<td>Constant rate</td>
<td>Yes</td>
</tr>
<tr>
<td>ATM</td>
<td>VBR*</td>
<td>Guaranteed rate</td>
<td>Yes</td>
</tr>
<tr>
<td>ATM</td>
<td>ABR*</td>
<td>Guaranteed minimum</td>
<td>No</td>
</tr>
<tr>
<td>ATM</td>
<td>UBR*</td>
<td>None</td>
<td>No</td>
</tr>
</tbody>
</table>

* Constant bit rate (CBR); variable bit rate (VBR); adaptive bit rate (ABR); uniform bit rate (UBR)

- Internet model being extended with *differentiated services (DiffServ)*
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
Part 4: Outline

- Overview and Network Layer Services
- **Data Plane: What’s Inside a Router?**
- Data Plane: Internet Protocol (IP) and Addressing (IPv4, IPv6)
- Control Plane: Routing Algorithms: Link-State and Distance-Vector
- Control Plane: Internet Routing Protocols:
  - Intra-Domain
  - Inter-Domain
- Control Plane: Multicast and Anycast Routing
Two key router functions:

- Run routing algorithms/protocol (RIP, OSPF, BGP)
- *Switching* datagrams from incoming to outgoing link
Input Port Functions

- Line termination
- Link layer protocol (receive)
- Lookup, forwarding queueing

Switch fabric

Physical layer: bit-level reception
Data link layer: e.g., Ethernet see chapter 5

Decentralized switching:
- Given datagram destination, lookup output port using forwarding table in input port memory
- Goal: complete input port processing at ‘line speed’
- Queueing: if datagrams arrive faster than forwarding rate into switch fabric
Switching Fabrics

• Transfer packet from input buffer to appropriate output buffer

• Switching rate: rate at which packets can be transferred from inputs to outputs
  – Often measured as multiple of input/output line rate
  – $N$ inputs: switching rate $N$ times line rate desirable

• Three types of switching fabrics:

- Memory
- Bus
- Crossbar
Switching via Memory

*First generation routers:*

- Traditional computers with switching under direct control of CPU
- Packet copied to system’s memory
- CPU extracts dest address from packet’s header, looks up output port in forwarding table, copies to output port
- Speed limited by memory bandwidth (2 bus crossings per datagram)
- One packet at a time
Switching via Bus

- Datagram from input port memory to output port memory via a shared bus
- **Bus contention:** switching speed limited by bus bandwidth
- One packet a time
- 32 Gbps bus, Cisco 5600: sufficient speed for access and enterprise routers
Switching via Interconnection Network

- Forwards multiple packets in parallel
- Banyan networks, crossbar, other interconnection nets initially developed to connect processors in multiprocessor
- When packet from port A needs to forwarded to port Y, controller closes cross point at intersection of two buses
- Advanced design: fragmenting datagram into fixed length cells, switch cells through the fabric.

![Diagram of a crossbar network with ports A, B, C, and intersections X, Y, Z.](image)
Output Ports

• **Buffering** required when datagrams arrive from fabric faster than the transmission rate
• **Scheduling discipline** chooses among queued datagrams for transmission
• Suppose $R_{\text{switch}}$ is $N$ times faster than $R_{\text{line}}$
• Still have output buffering when multiple inputs send to same output
• *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 ms) times link capacity $C$
  
  – e.g., $C = 10$ Gpbs link: 2.5 Gbit buffer

• Recent recommendation: with $N$ flows, buffering equal to

$$\frac{RTT \cdot C}{\sqrt{N}}$$
Input Port Queueing

• Fabric slower than input ports combined queuing may occur at input queues
  – Queuing delay and loss due to input buffer overflow!
• Head-of-the-Line (HOL) blocking: queued datagram at front of queue prevents others in queue from moving forward

Output port contention: only one red datagram can be transferred.  
*Lower red packet is blocked*

One packet time later: green packet experiences HOL blocking
Part 4: Outline

• Overview and Network Layer Services
• Data Plane: What’s Inside a Router?
• **Data Plane: Internet Protocol (IP) and Addressing (IPv4, IPv6)**
• Control Plane: Routing Algorithms: Link-State and Distance-Vector
• Control Plane: Internet Routing Protocols:
  – Intra-Domain
  – Inter-Domain
• Control Plane: Multicast and Anycast Routing
The Internet Network Layer

Host, router network layer functions:

Transport layer: TCP, UDP

Routing protocols
- Path selection
- RIP, OSPF, BGP

Routing table

IP protocol
- Addressing conventions
- Datagram format
- Packet handling conventions

ICMP protocol
- Error reporting
- Router “signaling”

Link layer

Physical layer
**IPv4 Datagram Format**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP protocol version</td>
<td>4 bytes</td>
</tr>
<tr>
<td>Header length</td>
<td>4 bytes</td>
</tr>
<tr>
<td>“Type” of data</td>
<td>8 bytes</td>
</tr>
<tr>
<td>Max number remaining</td>
<td>16 bytes</td>
</tr>
<tr>
<td>Max number remaining</td>
<td>16 bytes</td>
</tr>
<tr>
<td>Max number remaining</td>
<td>16 bytes</td>
</tr>
<tr>
<td>Upper layer protocol</td>
<td>4 bytes</td>
</tr>
<tr>
<td>Total datagram length</td>
<td>4 bytes</td>
</tr>
<tr>
<td>Time to live</td>
<td>4 bytes</td>
</tr>
<tr>
<td>Upper layer</td>
<td>4 bytes</td>
</tr>
<tr>
<td>Internet checksum</td>
<td>4 bytes</td>
</tr>
<tr>
<td>Options (if any)</td>
<td>4 bytes</td>
</tr>
<tr>
<td>Data</td>
<td>4 bytes</td>
</tr>
<tr>
<td>How much overhead?</td>
<td>20 bytes of TCP, 20 bytes of IP, 40 bytes + app layer overhead</td>
</tr>
</tbody>
</table>

**How much overhead?**
- 20 bytes of TCP
- 20 bytes of IP
- = 40 bytes + app layer overhead
IP Fragmentation & Reassembly (1)

• 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
IP Addressing: Introduction

- **IP address**: 32-bit identifier for host, router *interface*
- **Interface**: connection between host, router and physical link
  - Routers typically have multiple interfaces
  - Host may have multiple interfaces
  - IP addresses associated with *interface*, not host, router

```
223.1.1.1 = 11011111 00000001 00000001 00000001
223 1 1 1
```
IP Addressing (1)

- **IP address:**
  - Network part (high order bits)
  - Host part (low order bits)
- **What’s a network?** *(from IP address perspective)*
  - Device interfaces with same network part of IP address
  - Can physically reach each other without intervening router

Network consisting of 3 IP networks (for IP addresses starting with 223, first 24 bits are network address)
How to find the networks?

• Detach each interface from router, host

• Create “islands of isolated networks” (subnets)

Interconnected system consisting of six networks
IP Addressing: Classes

Given notion of “network”, let’s re-examine IP addresses:

“Classful” addressing:

<table>
<thead>
<tr>
<th>Class</th>
<th>Network</th>
<th>Host</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td></td>
<td>1.0.0.0 to 127.255.255.255</td>
</tr>
<tr>
<td>B</td>
<td>10</td>
<td></td>
<td>128.0.0.0 to 191.255.255.255</td>
</tr>
<tr>
<td>C</td>
<td>110</td>
<td></td>
<td>192.0.0.0 to 223.255.255.255</td>
</tr>
<tr>
<td>D</td>
<td>1110</td>
<td>multicast address</td>
<td>224.0.0.0 to 239.255.255.255</td>
</tr>
</tbody>
</table>

32 bits
IP Addressing: CIDR

• Classful addressing:
  – Inefficient use of address space, address space exhaustion
  – e.g., class B net allocated enough addresses for 65K hosts, even if only 2K hosts in that network

• **CIDR: Classless InterDomain Routing**
  – Network portion of address of arbitrary length
  – Address format: **a.b.c.d/x**, where *x* is # bits in network portion of address

```
11001000 00010111 00010000 00000000
200.23.16.0/23
```
IP Addresses: How to Get One? (1)

Hosts (host portion):

- **Hard-coded by system admin in a file**
  - Windows: Control Panel → Network → Configuration → TCP/IP → Properties
  - *nix: /etc/rc.config, /etc/network/interfaces

- **DHCP: Dynamic Host Configuration Protocol:** dynamically get address: “plug-and-play”
  - 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 can send system configuration data too
IP Addresses: How to Get One? (2)

Network (network portion):

- Get allocated portion of ISP’s address space:

<table>
<thead>
<tr>
<th>ISP's block</th>
<th>Network</th>
<th>Subnet</th>
<th>Mask</th>
<th>Netmask</th>
</tr>
</thead>
<tbody>
<tr>
<td>11001000 00010111 00010000 00000000</td>
<td>200.23.16.0/20</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11001000 00010111 00010000 00000000</td>
<td>200.23.16.0/23</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11001000 00010111 00010001 00000000</td>
<td>200.23.18.0/23</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11001000 00010111 00010010 00000000</td>
<td>200.23.20.0/23</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11001000 00010111 00010100 00000000</td>
<td>200.23.22.0/23</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11001000 00010111 00010110 00000000</td>
<td>200.23.24.0/23</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11001000 00010111 00011000 00000000</td>
<td>200.23.26.0/23</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11001000 00010111 00011010 00000000</td>
<td>200.23.28.0/23</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11001000 00010111 00011100 00000000</td>
<td>200.23.30.0/23</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
NAT: Network Address Translation (1)

**Motivation:** local network uses just one IP address as far as outside world is concerned:

- Range of addresses not needed from ISP: just one IP address for all devices
- Can change addresses of devices in local network without notifying outside world
- Can change ISP without changing addresses of devices in local network
- Devices inside local net not explicitly addressable, visible by outside world (a security plus)
NAT: Network Address Translation (2)

**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 address

– **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 (3)

1: Host 10.0.0.1 sends datagram to 128.119.40.186, 80

2: NAT router changes datagram source addr from 10.0.0.1, 3345 to 138.76.29.7, 5001, updates table

3: Reply arrives dest. address: 138.76.29.7, 5001

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

NAT Translation Table

<table>
<thead>
<tr>
<th>WAN Side Address</th>
<th>LAN Side Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>138.76.29.7, 5001</td>
<td>10.0.0.1, 3345</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

S: 10.0.0.4, 80
D: 128.119.40.186, 80

S: 10.0.0.1, 3345
D: 128.119.40.186, 80
NAT: Network Address Translation (4)

• 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 (1)

• Client wants to connect to server with address 10.0.0.1
  – Server address 10.0.0.1 local to LAN (client can’t use it as destination address)
  – Only one externally visible NATed address: 138.76.29.7

• **Solution 1:** statically configure NAT to forward incoming connection requests at given port to server
  – e.g., (123.76.29.7, port 25000) always forwarded to 10.0.0.1 port 25000
NAT Traversal Problem (2)

• **Solution 2**: Universal Plug and Play (UPnP) Internet Gateway Device (IGD) Protocol. Allows NATed host to:
  - Learn public IP address (138.76.29.7)
  - Add/remove port mappings (with lease times)

  i.e., automate static NAT port map configuration
NAT Traversal Problem (3)

• **Solution 3:** relaying (used in Skype)
  – NATed client establishes connection to relay
  – External client connects to relay
  – Relay bridges packets between to connections
Hierarchical Addressing: Route Aggregation (1)

Hierarchical addressing allows efficient advertisement of routing information:
Hierarchical Addressing: Route Aggregation (2)

Hierarchical addressing allows efficient advertisement of routing information:

Organization 0
200.23.16.0/23

11001000 00010111 00010000 00000000

Organization 1
200.23.18.0/23

11001000 00010111 00010010 00000000

Organization 2
200.23.20.0/23

11001000 00010111 00010100 00000000

Fly-By-Night-ISP

Organization 7
200.23.30.0/23

11001000 00010111 00011110 00000000

ISP block: 20 bits

“Send me anything with addresses beginning 200.23.16.0/20”
Hierarchical Addressing: More Specific Routes

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

Organization 0
200.23.16.0/23

Organization 2
200.23.20.0/23

Organization 7
200.23.30.0/23

Organization 1
200.23.18.0/23

Fly-By-Night-ISP

“Send me anything with addresses beginning 200.23.16.0/20”

ISPs-R-Us

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

Internet
ICMP: Internet Control Message Protocol

- Used by hosts, routers, gateways to communication 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 header</td>
</tr>
</tbody>
</table>
IPv6

• **Initial motivation:** 32-bit address space completely allocated by 2008.

• **Additional motivation:**
  – Header format helps speed processing/forwarding
  – Header changes to facilitate QoS
  – New “anycast” address: route to “best” of several replicated servers

• **IPv6 datagram format:**
  – fixed-length 40 byte header
  – no fragmentation allowed
IPv6 Header

**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?

• Two proposed approaches:
  – Dual Stack: some routers with dual stack (v6, v4) can “translate” between formats
  – Tunneling: IPv6 carried as payload in IPv4 datagram among IPv4 routers
Dual Stack Approach

The image illustrates a network with nodes labeled A to F, connected by links with IPv6 and IPv4 stacks. The flow of data packets is shown with source and destination nodes.

- **A to B**: IPv6
- **B to C**: IPv4
- **D to E**: IPv4
- **E to F**: IPv6

The data packets are labeled with source and destination nodes, indicating the flow direction and protocol used.
IPv6 Tunneling via IPv4

IPv6 inside IPv4 where needed
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
Getting Datagram from Source to Dest. (1)

**IP datagram:**

- Datagram remains unchanged, as it travels source to destination
- Addr fields of interest here

**Routing table in A**

<table>
<thead>
<tr>
<th>Dest. Net.</th>
<th>Next Router</th>
<th># Hops</th>
</tr>
</thead>
<tbody>
<tr>
<td>223.1.1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>223.1.2</td>
<td>223.1.1.4</td>
<td>2</td>
</tr>
<tr>
<td>223.1.3</td>
<td>223.1.1.4</td>
<td>2</td>
</tr>
</tbody>
</table>

![Routing diagram](image)
Getting Datagram from Source to Dest. (2)

Starting at $A$, given IP datagram addressed to $B$:

- Look up network address of $B$
- Find $B$ is on same network as $A$
- Link layer will send datagram directly to $B$ inside link-layer frame
  - $B$ and $A$ are directly connected
Getting Datagram from Source to Dest. (3)

Starting at A, dest. E:
- Look up network address of E
- E on different network
  - A, E not directly attached
- Routing table: next hop router to E is 223.1.1.4
- Link layer sends datagram to router 223.1.1.4 inside link-layer frame
- datagram arrives at 223.1.1.4
- continued…..
Arriving at 223.1.4, destined for 223.1.2.2

- Look up network address of E
- E on same network as router’s interface 223.1.2.9
  - Router, E directly attached
- Link layer sends datagram to 223.1.2.2 inside link-layer frame via interface 223.1.2.9
- Datagram arrives at 223.1.2.2!!! (hooray!)
Part 4: Outline

• Overview and Network Layer Services
• Data Plane: What’s Inside a Router?
• Data Plane: Internet Protocol (IP) and Addressing (IPv4, IPv6)
• Control Plane: Routing Algorithms: Link-State and Distance-Vector
• Control Plane: Internet Routing Protocols:
  – Intra-Domain
  – Inter-Domain
• Control Plane: Multicast and Anycast Routing
Interplay Between Routing and Forwarding

Routing Algorithm determines end-end-path through network (control plane)

Forwarding table determines local forwarding at this router (data plane)

<table>
<thead>
<tr>
<th>Header Value</th>
<th>Output Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address range 1</td>
<td>3</td>
</tr>
<tr>
<td>Address range 2</td>
<td>2</td>
</tr>
<tr>
<td>Address range 3</td>
<td>2</td>
</tr>
<tr>
<td>Address range 4</td>
<td>1</td>
</tr>
</tbody>
</table>

Routing algorithm determines end-end-path through network (control plane)

Forwarding table determines local forwarding at this router (data plane)

Value in arriving packet’s header
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

- “Good” path:
  - Typically means minimum cost path
  - Other definitions possible
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
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 routing table for that node
• Iterative: after $k$ iterations, know least cost path to $k$ destinations

Notation:

• $c(i, j)$: link cost from node $i$ to $j$. cost infinite 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 $v$, that is, next $v$
• $N$: set of nodes whose least cost path definitively known
Dijkstra’s Algorithm

1: // Initialization:
2: \( N \leftarrow \{A\} \)
3: for all nodes \( v \) do
4: \( \text{if } v \text{ adjacent to } A \text{ then} \)
5: \( D(v) \leftarrow c(A, v) \)
6: else
7: \( D(v) \leftarrow \infty \)
8: end if
9: end for
10: repeat
11: find \( w \notin N \) such that \( D(w) \) is minimized
12: \( N \leftarrow N \cup \{w\} \)
13: for all \( v \) adjacent to \( w \) where \( v \notin N \) do
14: \( D(v) \leftarrow \min\{D(v), D(w) + c(w, v)\} \)
15: // new cost to \( v \) is either old cost to \( v \) or known shortest path cost to \( w \) plus cost from \( w \) to \( v \)
16: end for
17: until \( N \) contains all nodes in graph
### Dijkstra’s Algorithm: Example

<table>
<thead>
<tr>
<th>Step</th>
<th>Start N</th>
<th>$D(B), p(B)$</th>
<th>$D(C), p(C)$</th>
<th>$D(D), p(D)$</th>
<th>$D(E), p(E)$</th>
<th>$D(F), p(F)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$A$</td>
<td>2, $A$</td>
<td>5, $A$</td>
<td>1, $A$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>1</td>
<td>$AD$</td>
<td>2, $A$</td>
<td>4, $D$</td>
<td></td>
<td>2, $D$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>2</td>
<td>$ADE$</td>
<td>2, $A$</td>
<td>3, $E$</td>
<td></td>
<td></td>
<td>4, $E$</td>
</tr>
<tr>
<td>3</td>
<td>$ADEB$</td>
<td></td>
<td>3, $E$</td>
<td></td>
<td></td>
<td>4, $E$</td>
</tr>
<tr>
<td>4</td>
<td>$ADEBC$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4, $E$</td>
</tr>
<tr>
<td>5</td>
<td>$ADEBCF$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

![Graph representation](image.png)
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
Distance Vector Routing Algorithm

**Iterative:**
- Continues until no nodes exchange info.
- *Self-terminating*: no “signal” to stop

**Asynchronous:**
- Nodes need *not* exchange info/iterate in lock step!

**Distributed:**
- Each node communicates *only* with directly-attached neighbors

**Distance Table data structure**
- Each node has its own
- Rows for each possible destination
- Columns for each directly-attached neighbor to node
- Example: at node \( X \), for dest. \( Y \) via neighbor \( Z \):

\[
D^X(Y, Z) \text{ denotes distance from } X \text{ to } Y, \text{ via } Z \text{ as next hop}
\]

\[
D^X(Y, Z) = c(X, Z) + \min_{w \in \{X’s \, neighbors\}} \{D^Z(Y, w)\}
\]

Bellman-Ford equation
Distance Table: Example

\[ D^E(C, D) = c(E, C) + \min_{w \in E's \text{ neighbors}} \{D^D(C, w)\} \]
\[ = 2 + 2 = 4 \]

\[ D^E(A, D) = c(E, D) + \min_{w \in D's \text{ neighbors}} \{D^D(A, w)\} \]
\[ = 2 + 3 = 5 \quad \text{loop!} \]

\[ D^E(A, B) = c(E, B) + \min_{w \in B's \text{ neighbors}} \{D^B(A, w)\} \]
\[ = 8 + 6 = 14 \quad \text{loop!} \]
## Distance Table Yields Routing Table

### Distance Table

<table>
<thead>
<tr>
<th>Destination</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></td>
<td>5</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></td>
<td>4</td>
</tr>
<tr>
<td>D</td>
<td>4</td>
<td>11</td>
<td></td>
<td>2</td>
</tr>
</tbody>
</table>

### Routing Table

<table>
<thead>
<tr>
<th>Destination</th>
<th>Outgoing link to use, cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>A, 1</td>
</tr>
<tr>
<td>B</td>
<td>D, 5</td>
</tr>
<tr>
<td>C</td>
<td>D, 4</td>
</tr>
<tr>
<td>D</td>
<td>D, 2</td>
</tr>
</tbody>
</table>

Distance table ➔ Routing table
Distance Vector Routing: Overview

**Iterative, asynchronous:**
- each local iteration caused by:
  - Local link cost change
  - Message from neighbor: its least cost path change from neighbor

**Distributed:**
- Each node notifies neighbors *only* when its least cost path to any destination changes
  - Neighbors then notify their neighbors if necessary

Each node:

1. **Wait** for (change in local link cost of msg from neighbor)
2. **Recompute** distance table
3. If least cost path to any dest has changed, **notify** neighbors
Distance Vector Algorithm:

1: At all nodes $X$:
2: // Initialization:
3: for all adjacent nodes $V$ do
4: $D^X(\ast, V) \leftarrow \infty$  // * operator means ‘for all’
5: $D^X(V, V) \leftarrow c(X, V)$
6: end for
7: for all destinations $Y$ do
8: Send $\min_{w \in \text{neighbors of } X}\{D^X(Y, w)\}$ to each neighbor
9: end for
10: loop
11: wait until we see a link cost change to neighbor $V$ or until we receive update from neighbor $V$
12: if $c(X, V)$ changes by $d$ then
13: // change cost to all dests via neighbor $V$ by $d$ ($d$ could be positive or negative)
14: for all destinations $Y$ do
15: $D^X(Y, V) \leftarrow D^X(Y, V) + d$
16: end for
17: else if update received from $V$ w.r.t. destination $Y$ then
18: // shortest path from $V$ to some $Y$ changed: $V$ sent a new value
19: $newval \triangleq \min_{w \in \text{neighbors of } V}\{D_w(Y, w)\}$
20: for destination $Y$ do
21: $D^X(Y, V) \leftarrow c(X, V) + newval$
22: end for
23: end if
24: if we have a new $\min_{w \in \text{neighbor of } V}\{D^X(Y, w)\}$ for any destination $Y$ then
25: send new value of $\min_{w \in \text{neighbor of } V}\{D^X(Y, w)\}$ to all neighbors
26: end if
27: // this loop runs forever
28: end loop
Distance Vector Algorithm: Example (1)
Distance Vector Algorithm: Example (2)

\[
\begin{align*}
D^X(Y, Z) &= c(X, Z) + \min_w \{D^Z(Y, w)\} \\
&= 7 + 1 = 8 \\
D^X(Z, Y) &= c(X, Y) + \min_w \{D^Y(Z, w)\} \\
&= 2 + 1 = 3
\end{align*}
\]
Distance Vector: Link Cost Changes (1)

**Link cost changes:**
- Node detects local link cost change
- Updates distance table (line 15)
- If cost change in least cost path, notify neighbors (lines 23, 24)

“Good news travels fast”
Distance Vector: Link Cost Changes (2)

Link cost changes:

- Good news travels fast
- Bad news travels slow - “count to infinity” problem!
Distance Vector: 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?

Algorithm terminates
Comparison of LS and DV Algorithms

Message complexity
- **LS:** with $n$ nodes, $E$ links, $O(nE)$ msgs sent each
- **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 propagates through 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 ASs can run different intra-AS routing protocol

Gateway routers

- Special routers in AS
- Run intra-AS routing protocol with all other routers in AS
- Also responsible for routing to destinations outside AS
  - Run inter-AS routing protocol with other gateway routers
Part 4: Outline

- Overview and Network Layer Services
- Data Plane: What’s Inside a Router?
- Data Plane: Internet Protocol (IP) and Addressing (IPv4, IPv6)
- Control Plane: Routing Algorithms: Link-State and Distance-Vector
- **Control Plane: Internet Routing Protocols:**
  - Intra-Domain
  - Inter-Domain
- Control Plane: Multicast and Anycast Routing
Intra-AS and Inter-AS Routing (1)

Gateways:
- Perform inter-AS routing amongst themselves
- Perform intra-AS routers with other routers in their AS

Inter-AS, intra-AS routing in gateway A.c
We’ll examine specific inter-AS and intra-AS Internet routing protocols shortly.
Routing in the Internet

• The Global Internet consists of Autonomous Systems (AS) interconnected with each other:
  – Stub AS: small corporation
  – Multihomed AS: large corporation (no transit)
  – Transit AS: provider

• Two-level routing:
  – Intra-AS: administrator is responsible for choice
  – Inter-AS: unique standard
Internet AS Hierarchy

Intra-AS border (exterior gateway) routers

Inter-AS interior (gateway) routers
Intra-AS Routing

• Also known as **Interior Gateway Protocols (IGP)**
• Most common IGPs:
  
  – RIP: Routing Information Protocol
  – OSPF: Open Shortest Path First
  – IGRP: Interior Gateway Routing Protocol (Cisco proprietary)
RIP (Routing Information Protocol) (1)

- Distance vector algorithm
- Included in BSD-UNIX Distribution in 1982
- Distance metric: # of hops (max = 15 hops)
  - Can you guess why?
- Distance vectors: exchanged every 30 sec via Response Message (also called advertisement)
- Each advertisement: route to up to 25 destination nets
Routing table in $D$

<table>
<thead>
<tr>
<th>Destination Network</th>
<th>Next Router</th>
<th># Hops to Destination</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
- Poisoned 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
RIP Table Example

Router: giroflee.eurocom.fr

<table>
<thead>
<tr>
<th>Destination</th>
<th>Gateway</th>
<th>Flags</th>
<th>Ref</th>
<th>Use</th>
<th>Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>127.0.0.1</td>
<td>127.0.0.1</td>
<td>UH</td>
<td>0</td>
<td>26492</td>
<td>lo0</td>
</tr>
<tr>
<td>192.168.2.</td>
<td>192.168.2.5</td>
<td>U</td>
<td>2</td>
<td>13</td>
<td>fa0</td>
</tr>
<tr>
<td>193.55.114.</td>
<td>193.55.114.6</td>
<td>U</td>
<td>3</td>
<td>58503</td>
<td>le0</td>
</tr>
<tr>
<td>192.168.3.</td>
<td>192.168.3.5</td>
<td>U</td>
<td>2</td>
<td>25</td>
<td>qaa0</td>
</tr>
<tr>
<td>224.0.0.0</td>
<td>193.55.114.6</td>
<td>U</td>
<td>3</td>
<td>0</td>
<td>le0</td>
</tr>
<tr>
<td>default</td>
<td>193.55.114.129</td>
<td>UG</td>
<td>0</td>
<td>143454</td>
<td></td>
</tr>
</tbody>
</table>

- Three attached class C networks (LANs)
- Router only knows routes to attached LANs
- Default router used to “go up”
- Route multicast address: 224.0.0.0
- Loopback interface (for debugging)
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)
OSPF “Advanced” Features (not in RIP)

• **Security**: all OSPF messages authenticated (to prevent malicious intrusion); TCP connections used
• **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 unicast and **multicast** support:
  – Multicast OSPF (MOSPF) uses same topology data base 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 knows 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 ASs.
IGRP (Interior Gateway Routing Protocol)

- CISCO proprietary; successor of RIP (mid 80s)
- Distance Vector, like RIP
- Several cost metrics (delay, bandwidth, reliability, load etc)
- Uses TCP to exchange routing updates
- Loop-free routing via Distributed Updating Alg. (DUAL) based on diffused computation
Inter-AS routing
Internet inter-AS routing: BGP (1)

- **BGP (Border Gateway Protocol):** *the de facto standard*

- **Path Vector protocol:**
  - Similar to Distance Vector protocol
  - Each Border Gateway broadcast to neighbors (peers) *entire path* (i.e., sequence of ASs) to destination
  - E.g., Gateway $X$ may send its path to dest. $Z$:

$$\text{Path } (X, Z) = X, Y_1, Y_2, Y_3, \ldots, Z$$
Internet inter-AS routing: BGP (2)

*Suppose:* gateway $X$ sends its path to peer gateway $W$

- $W$ may or may not select path offered by $X$
  - Cost, policy (don’t route via competitors AS), loop prevention reasons.
- If $W$ selects path advertised by $X$, then:
  $$\text{Path (}W, Z\text{)} = w, \text{ Path (}X, Z\text{)}$$
- Note: $X$ can control incoming traffic by controlling its route advertisements to peers:
  - e.g., don’t want to route traffic to $Z \Rightarrow$ don’t advertise any routes to $Z$
Internet inter-AS routing: BGP (3)

- 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
Why Different Intra-AS 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
Part 4: Outline

• Overview and Network Layer Services
• Data Plane: What’s Inside a Router?
• Data Plane: Internet Protocol (IP) and Addressing (IPv4, IPv6)
• Control Plane: Routing Algorithms: Link-State and Distance-Vector
• Control Plane: Internet Routing Protocols:
  – Intra-Domain
  – Inter-Domain
• Control Plane: Multicast and Anycast Routing
Broadcast Routing

• Deliver packets from source to all other nodes
• Source duplication is inefficient:

• Source duplication: how does source determine recipient addresses?
In-Network Duplication

• **Flooding**: when node receives broadcast packet, sends copy to all neighbors
  – Problems: cycles & broadcast storm

• **Controlled flooding**: node only broadcasts pkt if it hasn’t broadcast same packet before
  – Node keeps track of packet ids already broadcasted
  – Or reverse path forwarding (RPF): only forward packet if it arrived on shortest path between node and source

• **Spanning tree**:  
  – No redundant packets received by any node
Spanning Tree

- First construct a spanning tree
- Nodes then forward/make copies only along spanning tree

(a) Broadcast initiated at A
(b) Broadcast initiated at D
Spanning Tree: Creation

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

(a) stepwise construction of spanning tree (center: E)
(b) constructed spanning tree
Multicast Routing: Problem Statement

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

- **Tree:** not all paths between routers used
- **Shared-tree:** same tree used by all group members
- **Source-based:** different tree from each sender to rcvrs
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 cost tree (Steiner tree)
  – 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

LEGEND

Router with attached group member
Router with no attached group member
Link used for forwarding, $i$ indicates order link added by 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

Legend:
- Router with attached group member
- Router with no attached group member
- Prune message
- Links with multicast forwarding
Shared-Tree: Steiner Tree

• **Steiner tree**: minimum cost tree connecting all routers with attached group members
• Problem is NP-complete
• Excellent heuristics exists
• Not used in practice:
  – Computational complexity
  – Information about entire network needed
  – Monolithic: rerun whenever a router needs to join/leave
Center-Based Trees

• Single delivery tree shared by all
• One router identified as “center” of tree
• To join:
  – Edge router sends unicast $join$-msg addressed to center router
  – $Join$-msg “processed” by intermediate routers and forwarded towards center
  – $Join$-msg either hits existing tree branch for this center, or arrives at center
  – Path taken by $join$-msg becomes new branch of tree for this router
Center-Based Trees: Example

Suppose $R_6$ chosen as center:

Legend
- Router with attached group member
- Router with no attached group member
- path order in which join messages generated
Internet Multicasting Routing: DVMRP (1)

- **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
Internet Multicasting Routing: DVMRP (2)

- **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
Tunneling

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

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

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

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

**Sparse:**
- # networks with group members small w.r.t. # interconnected networks
- group members “widely dispersed”
- bandwidth not plentiful
Consequences Of Sparse-Dense Dichotomy

**Dense:**
- Group membership by routers *assumed* until routers explicitly prune
- *Data-driven* construction on mcast tree (e.g., RPF)
- Bandwidth and non-group-router processing *profligate*

**Sparse:**
- No membership until routers explicitly join
- *Receiver-driven* construction of mcast tree (e.g., center-based)
- Bandwidth and non-group-router processing *conservative*
PIM: Dense Mode

**Flood-and-prune RPF**: similar to DVMRP but...

- Underlying unicast protocol provides RPF info for incoming datagram
- Less complicated (less efficient) downstream flood than DVMRP reduces reliance on underlying routing algorithm
- Has protocol mechanism for router to detect it is a leaf-node router
PIM: Sparse Mode (1)

- 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

![Diagram showing PIM Sparse Mode](image)
**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!”
Part 4: done!

• Network Layer Services
• What’s inside a router?
• IPv4, IPv6 Addressing
• Inter-AS, Inter-AS routing
• Routing in the Internet
• Multicast Routing

• Understand principles behind network layer services:
  – Network layer service models, forwarding versus routing
    how a router works, routing (path selection), broadcast,
    multicast
• Instantiation, implementation in the Internet