Chapter 4: network layer

chapter goals:

- understand principles behind network layer services:
  - network layer service models
  - forwarding versus routing
  - how a router works
  - routing (path selection)
  - broadcast, multicast

- instantiation, implementation in the Internet
Chapter 4: outline

4.1 introduction
4.2 virtual circuit and datagram networks
4.3 what’s inside a router
4.4 IP: Internet Protocol
  - datagram format
  - IPv4 addressing
  - ICMP
  - IPv6
4.5 routing algorithms
  - link state
  - distance vector
  - hierarchical routing
4.6 routing in the Internet
  - RIP
  - OSPF
  - BGP
4.7 broadcast and multicast routing
Network layer

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

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

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

  *analogy:*

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

Routing algorithm determines end-end-path through network.

Forwarding table determines local forwarding at this router.

Value in arriving packet’s header:

<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>

Network Layer 4-6
Connection setup

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

- before datagrams flow, two end hosts and intervening routers establish virtual connection
  - routers get involved

- network vs transport layer connection service:
  - network: between two hosts (may also involve intervening routers in case of VCs)
  - transport: between two processes
Network service model

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

example services for individual datagrams:
- guaranteed delivery
- guaranteed delivery with less than 40 msec delay

example services for a flow of datagrams:
- in-order datagram delivery
- guaranteed minimum bandwidth to flow
- restrictions on changes in inter-packet spacing
## Network layer service models:

<table>
<thead>
<tr>
<th>Network Architecture</th>
<th>Service Model</th>
<th>Bandwidth</th>
<th>Loss</th>
<th>Order</th>
<th>Timing</th>
<th>Congestion feedback</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internet</td>
<td>best effort</td>
<td>none</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no (inferred via loss)</td>
</tr>
<tr>
<td>ATM</td>
<td>CBR</td>
<td>constant rate</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no congestion</td>
</tr>
<tr>
<td>ATM</td>
<td>VBR</td>
<td>guaranteed rate</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no congestion</td>
</tr>
<tr>
<td>ATM</td>
<td>ABR</td>
<td>guaranteed minimum</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>ATM</td>
<td>UBR</td>
<td>none</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>

Guarantees?

- Bandwidth: none, constant rate, guaranteed rate, guaranteed minimum
- Loss: no, yes
- Order: no, yes
- Timing: no, yes
- Congestion feedback: no (inferred via loss), no congestion, yes
Chapter 4: outline

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
Connection, connection-less 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 address)
- every router on source-dest path maintains “state” for each passing connection
- link, router resources (bandwidth, buffers) may be allocated to VC (dedicated resources = predictable service)
VC implementation

A VC consists of:

1. **path** from source to destination
2. **VC numbers**, one number for each link along path
3. **entries in forwarding tables** in routers along path

- packet belonging to VC carries VC number (rather than dest address)
- VC number can be changed on each link.
  - new VC number comes from forwarding table
**VC forwarding table**

<table>
<thead>
<tr>
<th>Incoming interface</th>
<th>Incoming VC #</th>
<th>Outgoing interface</th>
<th>Outgoing VC #</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>12</td>
<td>3</td>
<td>22</td>
</tr>
<tr>
<td>2</td>
<td>63</td>
<td>1</td>
<td>18</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>2</td>
<td>17</td>
</tr>
<tr>
<td>1</td>
<td>97</td>
<td>3</td>
<td>87</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

**VC routers maintain connection state information!**
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

- 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
Datagram forwarding table

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

IP destination address in arriving packet’s header

<table>
<thead>
<tr>
<th>dest address</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>
### Datagram forwarding table

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

**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:
- DA: 11001000 00010111 00010110 10100001
- DA: 11001000 00010111 00011000 10101010
Datagram or VC network: why?

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

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

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

Diagram:
- **Routing processor**
- **High-seed switching fabric**
- **Forwarding tables computed, pushed to input ports**
- **Routing, management control plane (software)**
- **Forwarding data plane (hardware)**
- **Router input ports**
- **Router output ports**
Input port functions

- **physical layer**: bit-level reception
- **data link layer**: e.g., Ethernet
  - see chapter 5

**decentralized switching:**
- given datagram dest., lookup output port using forwarding table in input port memory ("match plus action")
- goal: complete input port processing at 'line speed'
- queuing: 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 transfer 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
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)

![Diagram of network switching via memory](network_diagram.png)
Switching via a bus

- datagram from input port memory to output port memory via a shared bus
- **bus contention:** switching speed limited by bus bandwidth
- 32 Gbps bus, Cisco 5600: sufficient speed for access and enterprise routers
Switching via interconnection network

- overcome bus bandwidth limitations
- banyan networks, crossbar, other interconnection nets initially developed to connect processors in multiprocessor
- advanced design: fragmenting datagram into fixed length cells, switch cells through the fabric.
- Cisco 12000: switches 60 Gbps through the interconnection network
Output ports

- **buffering** required when datagrams arrive from fabric faster than the transmission rate
- **scheduling discipline** chooses among queued datagrams for transmission
Output port queueing

- buffering when arrival rate via switch exceeds output line speed
- queueing (delay) and loss due to output port buffer overflow!
How much buffering?

- RFC 3439 rule of thumb: average buffering equal to “typical” RTT (say 250 msec) times link capacity C
  - e.g., C = 10 Gpbs link: 2.5 Gbit buffer
- recent recommendation: with N flows, buffering equal to
  \[
  \frac{\text{RTT} \cdot C}{\sqrt{N}}
  \]
Input port queuing

- fabric slower than input ports combined -> queueing may occur at input queues
  - queueing 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

Diagram:
- Output port contention: only one red datagram can be transferred.
  - lower red packet is blocked
- One packet time later: green packet experiences HOL blocking
Chapter 4: outline

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
### IP datagram format

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP protocol version</td>
<td>version of the IP protocol</td>
</tr>
<tr>
<td>header length (bytes)</td>
<td>number of bytes in the header</td>
</tr>
<tr>
<td>“type” of data</td>
<td>type of data held in the datagram</td>
</tr>
<tr>
<td>16-bit identifier</td>
<td>identifier for the datagram</td>
</tr>
<tr>
<td>time to live</td>
<td>time remaining for the datagram</td>
</tr>
<tr>
<td>upper layer</td>
<td>protocol of the data to be delivered to the upper layer</td>
</tr>
<tr>
<td>source IP address</td>
<td>32-bit address of the source node</td>
</tr>
<tr>
<td>destination IP address</td>
<td>32-bit address of the destination node</td>
</tr>
<tr>
<td>options (if any)</td>
<td>additional options held in the datagram</td>
</tr>
<tr>
<td>data</td>
<td>payload data held in the datagram</td>
</tr>
</tbody>
</table>

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

**for fragmentation/reassembly**
- max number remaining hops (decremented at each router)
- e.g. timestamp, record route taken, specify list of routers to visit.

**total datagram length (bytes)**
- total length of the datagram

**IP datagram format**
- IP protocol version number
- header length (bytes)
- “type” of data
- 16-bit identifier
- time to live
- upper layer
- source IP address
- destination IP address
- options (if any)
- data
- total datagram length (bytes)

**Network Layer 4-34**
IP Packet Header Fields

- Version number (4 bits)
  - Indicates the version of the IP protocol
  - Necessary to know what other fields to expect
  - Typically “4” (for IPv4), and sometimes “6” (for IPv6)

- Header length (4 bits)
  - Number of 32-bit words in the header
  - Typically “5” (for a 20-byte IPv4 header)
  - Can be more when “IP options” are used

- Type-of-Service (8 bits)
  - Allow packets to be treated differently based on needs
  - E.g., low delay for audio, high bandwidth for bulk transfer
IP Packet Header Fields (Continued)

- **Total length (16 bits)**
  - Number of bytes in the packet
  - Maximum size is 63,535 bytes \((2^{16} - 1)\)
  - ... though underlying links may impose harder limits

- **Fragmentation information (32 bits)**
  - Packet identifier, flags, and fragment offset
  - Supports dividing a large IP packet into fragments
  - ... in case a link cannot handle a large IP packet

- **Time-To-Live (8 bits)**
  - Used to identify packets stuck in forwarding loops
  - ... and eventually discard them from the network
Time-to-Live (TTL) Field

- Potential robustness problem
  - Forwarding loops can cause packets to cycle forever
  - Confusing if the packet arrives much later

- Time-to-live field in packet header
  - TTL field decremented by each router on the path
  - Packet is discarded when TTL field reaches 0…
  - …and “time exceeded” message is sent to the source
Application of TTL in Traceroute

- Time-To-Live field in IP packet header
  - Source sends a packet with a TTL of $n$
  - Each router along the path decrements the TTL
  - “TTL exceeded” sent when TTL reaches 0

- Traceroute tool exploits this TTL behavior

Send packets with TTL=1, 2, … and record source of “time exceeded” message
**Example Traceroute: Berkeley to CNN**

<table>
<thead>
<tr>
<th>Hop number</th>
<th>IP address</th>
<th>DNS name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>169.229.62.1</td>
<td>inr-daedalus-0.cs.berkeley.edu</td>
</tr>
<tr>
<td>2</td>
<td>169.229.59.225</td>
<td>soda-cr-1-1-soda-br-6-2</td>
</tr>
<tr>
<td>3</td>
<td>128.32.255.169</td>
<td>vlan242.inr-202-doceev.berkeley.edu</td>
</tr>
<tr>
<td>4</td>
<td>128.32.0.249</td>
<td>gigE6-0-0.inr-666-doceev.berkeley.edu</td>
</tr>
<tr>
<td>5</td>
<td>128.32.0.66</td>
<td>qsv-juniper--ucb-gw.calren2.net</td>
</tr>
<tr>
<td>6</td>
<td>209.247.159.109</td>
<td>POS1-0.hsipaccess1.sanjose1.level3.net</td>
</tr>
<tr>
<td>7</td>
<td>*</td>
<td>?</td>
</tr>
<tr>
<td>8</td>
<td>64.159.1.46</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>209.247.9.170</td>
<td>pos8-0.hsa2.atlanta2.level3.net</td>
</tr>
<tr>
<td>10</td>
<td>66.185.138.33</td>
<td>pop2-atm-P0-2.atdn.net</td>
</tr>
<tr>
<td>11</td>
<td>*</td>
<td>?</td>
</tr>
<tr>
<td>12</td>
<td>66.185.136.17</td>
<td>pop1-atl-P4-0.atdn.net</td>
</tr>
<tr>
<td>13</td>
<td>64.236.16.52</td>
<td>www4.cnn.com</td>
</tr>
</tbody>
</table>

No response from router

No name resolution
Try Running Traceroute Yourself

- On UNIX machine
  - Traceroute
  - E.g., “traceroute www.cnn.com” or “traceroute 12.1.1.1”

- On Windows machine
  - Tracert
  - E.g., “tracert www.cnn.com” or “tracert 12.1.1.1”

- Common uses of traceroute
  - Discover the topology of the Internet
  - Debug performance and reachability problems
IP Packet Header Fields (Continued)

- **Protocol (8 bits)**
  - Identifies the higher-level protocol
    - E.g., “6” for the Transmission Control Protocol (TCP)
    - E.g., “17” for the User Datagram Protocol (UDP)
  - Important for demultiplexing at receiving host
    - Indicates what kind of header to expect next

```
protocol=6
IP header
TCP header
```
```
protocol=17
IP header
UDP header
```
IP Packet Header Fields (Continued)

- Checksum (16 bits)
  - Sum of all 16-bit words in the IP packet header
  - If any bits of the header are corrupted in transit
  - … the checksum won’t match at receiving host
  - Receiving host discards corrupted packets
    - Sending host will retransmit the packet, if needed

\[
\begin{align*}
134 &+ 212 \\
&= 346
\end{align*} \quad \begin{align*}
134 &+ 216 \\
&= 350
\end{align*}
\text{Mismatch!}

Network Layer 42
IP Packet Header (Continued)

- **Two IP addresses**
  - Source IP address (32 bits)
  - Destination IP address (32 bits)

- **Destination address**
  - Unique identifier for the receiving host
  - Allows each node to make forwarding decisions

- **Source address**
  - Unique identifier for the sending host
  - Recipient can decide whether to accept packet
  - Enables recipient to send a reply back to source
The IP Protocol

Some of the IP options.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security</td>
<td>Specifies how secret the datagram is</td>
</tr>
<tr>
<td>Strict source routing</td>
<td>Gives the complete path to be followed</td>
</tr>
<tr>
<td>Loose source routing</td>
<td>Gives a list of routers not to be missed</td>
</tr>
<tr>
<td>Record route</td>
<td>Makes each router append its IP address</td>
</tr>
<tr>
<td>Timestamp</td>
<td>Makes each router append its address and timestamp</td>
</tr>
</tbody>
</table>
What if the Source Lies?

- **Source address should be the sending host**
  - But, who’s checking, anyway?
  - You could send packets with any source you want

- **Why would someone want to do this?**
  - Launch a denial-of-service attack
    - Send excessive packets to the destination
    - ... to overload the node, or the links leading to the node
  - Evade detection by “spoofing”
    - But, the victim could identify you by the source address
    - So, you can put someone else’s source address in the packets
  - Also, an attack against the spoofed host
    - Spoofed host is wrongly blamed
    - Spoofed host may receive return traffic from the receiver
**IP fragmentation, reassembly**

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

*fragmentation:*

*in:* one large datagram

*out:* 3 smaller datagrams

*reassembly*
**IP fragmentation, reassembly**

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

**one large datagram becomes several smaller datagrams**

<table>
<thead>
<tr>
<th>length</th>
<th>ID</th>
<th>fragflag</th>
<th>offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>4000</td>
<td>x</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1500</td>
<td>x</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1500</td>
<td>x</td>
<td>1</td>
<td>185</td>
</tr>
<tr>
<td>1040</td>
<td>x</td>
<td>0</td>
<td>370</td>
</tr>
</tbody>
</table>

1480 bytes in data field

offset = 1480/8
Chapter 4: outline

4.1 introduction
4.2 virtual circuit and datagram networks
4.3 what’s inside a router
4.4 IP: Internet Protocol
   - datagram format
   - IPv4 addressing
   - ICMP
   - IPv6
4.5 routing algorithms
   - link state
   - distance vector
   - hierarchical routing
4.6 routing in the Internet
   - RIP
   - OSPF
   - BGP
4.7 broadcast and multicast routing
IP addressing: introduction

- **IP address**: 32-bit identifier for host, router interface
- **interface**: connection between host/router and physical link
  - router’s typically have multiple interfaces
  - host typically has one or two interfaces (e.g., wired Ethernet, wireless 802.11)
- **IP addresses associated with each interface**

\[
223.1.1.1 = \underbrace{11011111}_{223} \underbrace{00000001}_{1} \underbrace{00000001}_{1} \underbrace{00000001}_{1}
\]

Network Layer 4-49
IP Address (IPv4)

- A unique 32-bit number
- Identifies an interface (on a host, on a router, …)
- Represented in dotted-quad notation

```
  12  34  158  5
```

```
00001100 00100010 10011110 00000101
```
Grouping Related Hosts

The Internet is an “inter-network”
- Used to connect *networks* together, not *hosts*
- Needs a way to address a network (i.e., group of hosts)

LAN = Local Area Network
WAN = Wide Area Network
IP Address Classes

(a) 7 24
    0 Network Host

(b) 14 16
    1 0 Network Host

(c) 21 8
    1 1 1 0 Network Host
### IP Addresses

<table>
<thead>
<tr>
<th>Class</th>
<th>Prefix</th>
<th>Network</th>
<th>Host</th>
<th>Range of Host Addresses</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>Network</td>
<td>Host</td>
<td>1.0.0.0 to 127.255.255.255</td>
</tr>
<tr>
<td>B</td>
<td>10</td>
<td>Network</td>
<td>Host</td>
<td>128.0.0.0 to 191.255.255.255</td>
</tr>
<tr>
<td>C</td>
<td>110</td>
<td>Network</td>
<td>Host</td>
<td>192.0.0.0 to 223.255.255.255</td>
</tr>
<tr>
<td>D</td>
<td>1110</td>
<td></td>
<td>Multicast address</td>
<td>224.0.0.0 to 239.255.255.255</td>
</tr>
<tr>
<td>E</td>
<td>1111</td>
<td></td>
<td>Reserved for future use</td>
<td>240.0.0.0 to 255.255.255.255</td>
</tr>
</tbody>
</table>
### Special IP addresses.

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This host</td>
<td>A host on this network</td>
<td></td>
<td>Broadcast on the local network</td>
<td>Broadcast on a distant network</td>
</tr>
<tr>
<td></td>
<td>0 0</td>
<td>0 0</td>
<td></td>
<td>1 1 1 1 1 1 1 1 1 1 1 1 1 1 1</td>
<td>1 1 1 1 1 1 1 1 1 1 1 1 1 1 1</td>
</tr>
<tr>
<td></td>
<td>0 0</td>
<td></td>
<td>Host</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 1 1 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Network</td>
<td>1 1 1 1</td>
<td>1 1 1 1</td>
<td></td>
<td>1 1 1 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>127</td>
<td></td>
<td></td>
<td></td>
<td>Loopback</td>
</tr>
</tbody>
</table>
Subnets

A campus network consisting of LANs for various departments.
Subnets

**recipe**

- to determine the subnets, detach each interface from its host or router, creating islands of isolated networks
- each isolated network is called a *subnet*

```
subnet mask: /24
```

```
223.1.1.0/24
  223.1.1.1
  223.1.1.2
  223.1.1.3
  223.1.1.4

223.1.2.0/24
  223.1.2.1
  223.1.2.2
  223.1.2.9

223.1.3.0/24
  223.1.3.1
  223.1.3.2
  223.1.3.27
```

Network Layer 4-56
Subnets

how many?
**IP addressing: introduction**

**Q:** how are interfaces actually connected?

**A:** we’ll learn about that in chapter 5, 6.

**A:** wired Ethernet interfaces connected by Ethernet switches

**For now:** don’t need to worry about how one interface is connected to another (with no intervening router)

**A:** wireless WiFi interfaces connected by WiFi base station
Subnets

- **IP address:**
  - subnet part - high order bits
  - host part - low order bits

- **What's a subnet?**
  - device interfaces with same subnet part of IP address
  - can physically reach each other *without intervening router*
A class B network subnetted into 64 subnets.
Subnetted Address

Class B address

Network number 11111111111111111111111111111111
Subnet mask (255.255.255.0)

Network number Subnet ID Host ID

Subnetted address
Scalability Challenge

- Suppose hosts had arbitrary addresses
  - Then every router would need a lot of information
  - ...to know how to direct packets toward the host

```
1.2.3.4  5.6.7.8  2.4.6.8
 host  host  ...  host
LAN 1

1.2.3.5  5.6.7.9  2.4.6.9
 host  host  ...  host
LAN 2
```

```
<table>
<thead>
<tr>
<th></th>
<th>1.2.3.4</th>
<th></th>
<th>1.2.3.5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>←</td>
<td></td>
<td>→</td>
</tr>
<tr>
<td></td>
<td>1.2.3.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

forwarding table
Hierarchical Addressing in U.S. Mail

- **Addressing in the U.S. mail**
  - Zip code: 08540
  - Street: Olden Street
  - Building on street: 35
  - Room in building: 306
  - Name of occupant: Jennifer Rexford

- **Forwarding the U.S. mail**
  - Deliver letter to the post office in the zip code
  - Assign letter to mailman covering the street
  - Drop letter into mailbox for the building/room
  - Give letter to the appropriate person
Hierarchical Addressing: IP Prefixes

- Divided into network & host portions (left and right)
- 12.34.158.0/24 is a 24-bit prefix with $2^8$ addresses
IP Address and a 24-bit Subnet Mask

Address

00001100 00100010 10011110 00000101

12 34 158 5

Mask

11111111 11111111 11111111 00000000

11111111 11111111 11111111 00000000

255 255 255 0

Network Layer 65
Scalability Improved

- Number related hosts from a common subnet
  - 1.2.3.0/24 on the left LAN
  - 5.6.7.0/24 on the right LAN

1.2.3.4 1.2.3.7 1.2.3.156 5.6.7.8 5.6.7.9 5.6.7.212

LAN 1

LAN 2

forwarding table
Easy to Add New Hosts

- No need to update the routers
  - E.g., adding a new host 5.6.7.213 on the right
  - Doesn’t require adding a new forwarding entry

```
<table>
<thead>
<tr>
<th>LAN 1</th>
<th>LAN 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.2.3.0/24</td>
<td>5.6.7.0/24</td>
</tr>
<tr>
<td>1.2.3.4</td>
<td>5.6.7.8</td>
</tr>
<tr>
<td>1.2.3.7</td>
<td>5.6.7.9</td>
</tr>
<tr>
<td>1.2.3.156</td>
<td>5.6.7.212</td>
</tr>
<tr>
<td>host</td>
<td>host</td>
</tr>
<tr>
<td>host</td>
<td>host</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>host</td>
<td>host</td>
</tr>
<tr>
<td>host</td>
<td>host</td>
</tr>
<tr>
<td>route</td>
<td>route</td>
</tr>
<tr>
<td>WAN</td>
<td>WAN</td>
</tr>
<tr>
<td>router</td>
<td>router</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

forwarding table
IP addressing: CIDR

CIDR: Classless InterDomain Routing

- subnet portion of address of arbitrary length
- address format: \texttt{a.b.c.d/x}, where \( x \) is \# bits in subnet portion of address

\begin{align*}
\text{subnet part} & \quad \text{host part} \\
11001000 & \quad 00010111 & \quad 00010000 & \quad 00000000 \\
200.23.16.0/23
\end{align*}
Classful Addressing

- In the olden days, only fixed allocation sizes
  - Class A: 0*
    - Very large /8 blocks (e.g., MIT has 18.0.0.0/8)
  - Class B: 10*
    - Large /16 blocks (e.g., Princeton has 128.112.0.0/16)
  - Class C: 110*
    - Small /24 blocks (e.g., AT&T Labs has 192.20.225.0/24)
  - Class D: 1110*
    - Multicast groups
  - Class E: 11110*
    - Reserved for future use

- This is why folks use dotted-quad notation!
Classless Inter-Domain Routing (CIDR)

Use two 32-bit numbers to represent a network.
Network number = IP address + Mask

IP Address : 12.4.0.0       IP Mask: 255.254.0.0

Address: 00001100 00000100 00000000 00000000
Mask: 11111111 11111110 00000000 00000000

Network Prefix | for hosts

Written as 12.4.0.0/15
CDR – Classless InterDomain Routing

A set of IP address assignments.

<table>
<thead>
<tr>
<th>University</th>
<th>First address</th>
<th>Last address</th>
<th>How many</th>
<th>Written as</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cambridge</td>
<td>194.24.0.0</td>
<td>194.24.7.255</td>
<td>2048</td>
<td>194.24.0.0/21</td>
</tr>
<tr>
<td>Edinburgh</td>
<td>194.24.8.0</td>
<td>194.24.11.255</td>
<td>1024</td>
<td>194.24.8.0/22</td>
</tr>
<tr>
<td>Oxford</td>
<td>194.24.16.0</td>
<td>194.24.31.255</td>
<td>4096</td>
<td>194.24.16.0/20</td>
</tr>
</tbody>
</table>
CIDR: Hierarchical Address Allocation

- Prefixes are key to Internet scalability
  - Address allocated in contiguous chunks (prefixes)
  - Routing protocols and packet forwarding based on prefixes
  - Today, routing tables contain ~150,000-200,000 prefixes
Scalability: Address Aggregation

Provider is given 201.10.0.0/21

Routers in the rest of the Internet just need to know how to reach 201.10.0.0/21. The provider can direct the IP packets to the appropriate customer.
But, Aggregation Not Always Possible

Multi-homed customer with 201.10.6.0/23 has two providers. Other parts of the Internet need to know how to reach these destinations through both providers.
Scalability Through Hierarchy

- Hierarchical addressing
  - Critical for scalable system
  - Don’t require everyone to know everyone else
  - Reduces amount of updating when something changes

- Non-uniform hierarchy
  - Useful for heterogeneous networks of different sizes
  - Initial class-based addressing was far too coarse
  - Classless InterDomain Routing (CIDR) helps

- Next few slides
  - History of the number of globally-visible prefixes
  - Plots are # of prefixes vs. time

Growth faster than improvements in equipment capability
CIDR Deployed (1994-1996): Much Flatter

Efforts to aggregate (even decreases after IETF meetings!)

Good use of aggregation, and peer pressure in CIDR report

Internet boom and increased multi-homing
Long-Term View (1989-2005): Post-Boom
Obtaining a Block of Addresses

- Separation of control
  - Prefix: assigned to an institution
  - Addresses: assigned by the institution to their nodes

- Who assigns prefixes?
  - Internet Corporation for Assigned Names and Numbers
    - Allocates large address blocks to Regional Internet Registries
  - Regional Internet Registries (RIRs)
    - E.g., ARIN (American Registry for Internet Numbers)
    - Allocates address blocks within their regions
    - Allocated to Internet Service Providers and large institutions
  - Internet Service Providers (ISPs)
    - Allocate address blocks to their customers
    - Who may, in turn, allocate to their customers...
Figuring Out Who Owns an Address

- **Address registries**
  - Public record of address allocations
  - Internet Service Providers (ISPs) should update when giving addresses to customers
  - However, records are notoriously out-of-date

- **Ways to query**
  - UNIX: “whois –h whois.arin.net 128.8.130.75”
  - http://www.arin.net/whois/
  - …
Example Output for 128.8.130.75

OrgName: University of Maryland
OrgID: UNIVER-262
Address: Office of Information Technology
Address: Patuxent Building
City: College Park
StateProv: MD PostalCode: 20742
Country: US

NetRange: 128.8.0.0 - 128.8.255.255
CIDR: 128.8.0.0/16
NetName: UMDNET
NetHandle: NET-128-8-0-0-1
Parent: NET-128-0-0-0-0
NetType: Direct Assignment
NameServer: NOC.UMD.EDU
NameServer: NS1.UMD.EDU
NameServer: NS2.UMD.EDU
NameServer: NASANS4.NASA.GOV
Comment: RegDate:
Updated: 2004-04-12

RTechHandle: UM-ORG-ARIN
RTechName: UMD DNS Admin Role Account
RTechPhone: +1-301-405-3003
RTechEmail: dnsadmin@noc.net.umd.edu

OrgAbuseHandle: UARA-ARIN
OrgAbuseName: UMD Abuse Role Account
OrgAbusePhone: +1-301-405-8787
OrgAbuseEmail: abuse@umd.edu

OrgTechHandle: UM-ORG-ARIN
OrgTechName: UMD DNS Admin Role Account
OrgTechPhone: +1-301-405-3003
OrgTechEmail: dnsadmin@noc.net.umd.edu
Are 32-bit Addresses Enough?

- Not all that many unique addresses
  - $2^{32} = 4,294,967,296$ (just over four billion)
  - Plus, some are reserved for special purposes
  - And, addresses are allocated in larger blocks

- And, many devices need IP addresses
  - Computers, PDAs, routers, tanks, toasters, …

- Long-term solution: a larger address space
  - IPv6 has 128-bit addresses ($2^{128} = 3.403 \times 10^{38}$)

- Short-term solutions: limping along with IPv4
  - Private addresses
  - Network address translation (NAT)
  - Dynamically-assigned addresses (DHCP)
Hard Policy Questions

- How much address space per geographic region?
  - Equal amount per country?
  - Proportional to the population?
  - What about addresses already allocated?

- Address space portability?
  - Keep your address block when you change providers?
  - Pro: avoid having to renumber your equipment
  - Con: reduces the effectiveness of address aggregation

- Keeping the address registries up to date?
  - What about mergers and acquisitions?
  - Delegation of address blocks to customers?
  - As a result, the registries are horribly out of date
IP addresses: how to get one?

Q: How does a host get IP address?

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

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

- can renew its lease on address in use
- allows reuse of addresses (only hold address while connected/“on”)
- support for mobile users who want to join network (more shortly)

**DHCP overview:**

- host broadcasts “DHCP discover” msg [optional]
- DHCP server responds with “DHCP offer” msg [optional]
- host requests IP address: “DHCP request” msg
- DHCP server sends address: “DHCP ack” msg
DHCP client-server scenario

DHCP server

arriving DHCP client needs address in this network
DHCP client-server scenario

DHCP server: 223.1.2.5

DHCP discover
src: 0.0.0.0, 68
dest.: 255.255.255.255, 67
yiaddr: 0.0.0.0
transaction ID: 654

arriving client

DHCP offer
src: 223.1.2.5, 67
dest: 255.255.255.255, 68
yiaddr: 223.1.2.4
transaction ID: 654
lifetime: 3600 secs

DHCP request
src: 0.0.0.0, 68
dest: 255.255.255.255, 67
yiaddr: 223.1.2.4
transaction ID: 655
lifetime: 3600 secs

DHCP ACK
src: 223.1.2.5, 67
dest: 255.255.255.255, 68
yiaddr: 223.1.2.4
transaction ID: 655
lifetime: 3600 secs
DHCP: more than IP addresses

DHCP can return more than just allocated IP address on subnet:

- address of first-hop router for client
- name and IP address of DNS sever
- network mask (indicating network versus host portion of address)
connecting laptop needs its IP address, addr of first-hop router, addr of DNS server: use DHCP

- DHCP request encapsulated in UDP, encapsulated in IP, encapsulated in 802.1 Ethernet
- Ethernet frame broadcast (dest: FFFFFFFF) on LAN, received at router running DHCP server
- Ethernet demuxed to IP demuxed, UDP demuxed to DHCP
DCP server formulates DHCP ACK containing client’s IP address, IP address of first-hop router for client, name & IP address of DNS server.

- encapsulation of DHCP server, frame forwarded to client, demuxing up to DHCP at client

- client now knows its IP address, name and IP address of DNS server, IP address of its first-hop router
DHCP: Wireshark output (home LAN)

Message type: Boot Request (1)
Hardware type: Ethernet
Hardware address length: 6
Hops: 0
Transaction ID: 0x6b3a11b7
Seconds elapsed: 0
Bootp flags: 0x0000 (Unicast)
Client IP address: 0.0.0.0 (0.0.0.0)
Your (client) IP address: 0.0.0.0 (0.0.0.0)
Next server IP address: 0.0.0.0 (0.0.0.0)
Relay agent IP address: 0.0.0.0 (0.0.0.0)
Client MAC address: Wistron_23:68:8a (00:16:d3:23:68:8a)
Server host name not given
Boot file name not given
Magic cookie: (OK)
Option: (t=53,l=1) DHCP Message Type = DHCP Request
Option: (61) Client identifier
  Length: 7; Value: 010016D323688A;
  Hardware type: Ethernet
  Client MAC address: Wistron_23:68:8a (00:16:d3:23:68:8a)
Option: (t=50,l=4) Requested IP Address = 192.168.1.101
Option: (t=12,l=5) Host Name = "nomad"
Option: (55) Parameter Request List
  Length: 11; Value: 010F03062C2E2F1F21F92B
  1 = Subnet Mask; 15 = Domain Name
  3 = Router; 6 = Domain Name Server
  44 = NetBIOS over TCP/IP Name Server

Message type: Boot Reply (2)
Hardware type: Ethernet
Hardware address length: 6
Hops: 0
Transaction ID: 0x6b3a11b7
Seconds elapsed: 0
Bootp flags: 0x0000 (Unicast)
Client IP address: 192.168.1.101 (192.168.1.101)
Your (client) IP address: 0.0.0.0 (0.0.0.0)
Next server IP address: 192.168.1.1 (192.168.1.1)
Relay agent IP address: 0.0.0.0 (0.0.0.0)
Client MAC address: Wistron_23:68:8a (00:16:d3:23:68:8a)
Server host name not given
Boot file name not given
Magic cookie: (OK)
Option: (t=53,l=1) DHCP Message Type = DHCP ACK
Option: (t=54,l=4) Server Identifier = 192.168.1.1
Option: (t=1,l=4) Subnet Mask = 255.255.255.0
Option: (t=3,l=4) Router = 192.168.1.1
Option: (6) Domain Name Server
  Length: 12; Value: 445747E2445749F244574092;
  IP Address: 68.87.71.226;
  IP Address: 68.87.73.242;
  IP Address: 68.87.64.146
Option: (t=15,l=20) Domain Name = "hsd1.ma.comcast.net."
**IP addresses: how to get one?**

**Q:** how does network get subnet part of IP addr?

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

<table>
<thead>
<tr>
<th>ISP's block</th>
<th>11001000 00010111 00010000 00000000</th>
<th>200.23.16.0/20</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organization 0</td>
<td>11001000 00010111 00010000 00000000</td>
<td>200.23.16.0/23</td>
</tr>
<tr>
<td>Organization 1</td>
<td>11001000 00010111 00010010 00000000</td>
<td>200.23.18.0/23</td>
</tr>
<tr>
<td>Organization 2</td>
<td>11001000 00010111 00010100 00000000</td>
<td>200.23.20.0/23</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Organization 7</td>
<td>11001000 00010111 00011110 00000000</td>
<td>200.23.30.0/23</td>
</tr>
</tbody>
</table>
Hierarchical addressing allows efficient advertisement of routing information:

<table>
<thead>
<tr>
<th>Organization 0</th>
<th>200.23.16.0/23</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organization 1</td>
<td>200.23.18.0/23</td>
</tr>
<tr>
<td>Organization 2</td>
<td>200.23.20.0/23</td>
</tr>
<tr>
<td>Organization 7</td>
<td>200.23.30.0/23</td>
</tr>
</tbody>
</table>

Fly-By-Night-ISP

Send me anything with addresses beginning 200.23.16.0/20"

ISPs-R-Us

"Send me anything with addresses beginning 199.31.0.0/16"

Internet
Hierarchical addressing: more specific routes

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

Organization 0
200.23.16.0/23

Organization 2
200.23.20.0/23

Organization 7
200.23.30.0/23

Organization 1
200.23.18.0/23

Fly-By-Night-ISP

"Send me anything with addresses beginning 200.23.16.0/20"

ISPs-R-Us

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

Internet
IP addressing: the last word...

Q: how does an ISP get block of addresses?
A: ICANN: Internet Corporation for Assigned Names and Numbers http://www.icann.org/
  ▪ allocates addresses
  ▪ manages DNS
  ▪ assigns domain names, resolves disputes
NAT: network address translation

**all** datagrams *leaving* local network have *same* single source NAT IP address: 138.76.29.7, different source port numbers

datagrams with source or destination in this network have 10.0.0/24 address for source, destination (as usual)
**NAT: network address translation**

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

- range of addresses not needed from ISP: just one IP address for all devices
- can change addresses of devices in local network without notifying outside world
- can change ISP without changing addresses of devices in local network
- devices inside local net not explicitly addressable, visible by outside world (a security plus)
NAT: network address translation

**Implementation:** NAT router must:

- **Outgoing datagrams:** replace (source IP address, port #) of every outgoing datagram to (NAT IP address, new port #).
  
  ... remote clients/servers will respond using (NAT IP address, new port #) as destination addr.

- **Remember (in NAT translation table):** every (source IP address, port #) to (NAT IP address, new port #) translation pair.

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

1: host 10.0.0.1 sends datagram to 128.119.40.186, 80

<table>
<thead>
<tr>
<th>NAT translation table</th>
<th>WAN side addr</th>
<th>LAN side addr</th>
</tr>
</thead>
<tbody>
<tr>
<td>138.76.29.7, 5001</td>
<td>10.0.0.1, 3345</td>
<td></td>
</tr>
<tr>
<td>……</td>
<td>……</td>
<td></td>
</tr>
</tbody>
</table>

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

2: host 10.0.0.1 sends datagram to 128.119.40.186, 80

3: reply arrives dest. address: 138.76.29.7, 5001

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

NAT: network address translation
NAT: network address translation

- 16-bit port-number field:
  - 60,000 simultaneous connections with a single LAN-side address!

- NAT is controversial:
  - routers should only process up to layer 3
  - violates end-to-end argument
    - NAT possibility must be taken into account by app designers, e.g., P2P applications
  - address shortage should instead be solved by IPv6
NAT traversal problem

- client wants to connect to server with address 10.0.0.1
  - server address 10.0.0.1 local to LAN (client can’t use it as destination addr)
  - only one externally visible 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 2500) always forwarded to 10.0.0.1 port 25000
NAT traversal problem

- **solution 2**: Universal Plug and Play (UPnP) Internet Gateway Device (IGD) Protocol. Allows 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

_solution 3:_ relaying (used in Skype)
- NATed client establishes connection to relay
- external client connects to relay
- relay bridges packets between to connections
Chapter 4: outline

4.1 introduction
4.2 virtual circuit and datagram networks
4.3 what’s inside a router
4.4 IP: Internet Protocol
   - datagram format
   - IPv4 addressing
   - ICMP
   - IPv6
4.5 routing algorithms
   - link state
   - distance vector
   - hierarchical routing
4.6 routing in the Internet
   - RIP
   - OSPF
   - BGP
4.7 broadcast and multicast routing
ICMP: internet control message protocol

- used by hosts & routers to communicate network-level information
  - error reporting: unreachable host, network, port, protocol
  - echo request/reply (used by ping)
- network-layer “above” IP:
  - ICMP msgs carried in IP datagrams
- ICMP message: type, code plus first 8 bytes of IP datagram causing error

<table>
<thead>
<tr>
<th>Type</th>
<th>Code</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>echo reply (ping)</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>dest. network unreachable</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>dest host unreachable</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>dest protocol unreachable</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>dest port unreachable</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>dest network unknown</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>dest host unknown</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>source quench (congestion control - not used)</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>echo request (ping)</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>route advertisement</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>router discovery</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>TTL expired</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>bad IP header</td>
</tr>
</tbody>
</table>
Traceroute and ICMP

- source sends series of UDP segments to dest
  - first set has TTL = 1
  - second set has TTL = 2, etc.
  - unlikely port number
- when nth set of datagrams arrives to nth router:
  - router discards datagrams
  - and sends source ICMP messages (type 11, code 0)
  - ICMP messages includes name of router & IP address
- when ICMP messages arrives, source records RTTs

**stopping criteria:**
- UDP segment eventually arrives at destination host
- destination returns ICMP “port unreachable” message (type 3, code 3)
- source stops
IPv6: motivation

- *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 datagram format**

**priority:** identify priority among datagrams in flow

**flow Label:** identify datagrams in same “flow.”

(concept of “flow” not well defined).

**next header:** identify upper layer protocol for data

<table>
<thead>
<tr>
<th>ver</th>
<th>pri</th>
<th>flow label</th>
<th>payload len</th>
<th>next hdr</th>
<th>hop limit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>source address (128 bits)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>destination address (128 bits)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

32 bits
Other changes from IPv4

- **checksum**: removed entirely to reduce processing time at each hop
- **options**: allowed, but outside of header, indicated by “Next Header” field
- **ICMPv6**: new version of ICMP
  - additional message types, e.g. “Packet Too Big”
  - multicast group management functions
Transition from IPv4 to IPv6

- not all routers can be upgraded simultaneously
  - no “flag days”
  - how will network operate with mixed IPv4 and IPv6 routers?
- **tunneling**: IPv6 datagram carried as *payload* in IPv4 datagram among IPv4 routers
Tunneling

**logical view:**

IPv6

**physical view:**

IPv6

IPv4 tunnel connecting IPv6 routers

IPv6

IPv6

IPv6
Tunneling

Physical view:

- A-to-B: IPv6
- B-to-C: IPv6 inside IPv4
- B-to-C: IPv6 inside IPv4
- E-to-F: IPv6

Logical view:

- IPv4 tunnel connecting IPv6 routers
Chapter 4: outline

4.1 introduction
4.2 virtual circuit and datagram networks
4.3 what’s inside a router
4.4 IP: Internet Protocol
   - datagram format
   - IPv4 addressing
   - ICMP
   - IPv6
4.5 routing algorithms
   - link state
   - distance vector
   - hierarchical routing
4.6 routing in the Internet
   - RIP
   - OSPF
   - BGP
4.7 broadcast and multicast routing
Interplay between routing, forwarding

Routing algorithm determines end-end-path through network.

Forwarding table determines local forwarding at this router.

<table>
<thead>
<tr>
<th>dest address</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>

IP destination address in arriving packet’s header.
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)\}

*aside*: graph abstraction is useful in other network contexts, e.g., P2P, where $N$ is set of peers and $E$ is set of TCP connections
Graph abstraction: costs

$c(x,x') = \text{cost of link } (x,x')$

e.g., $c(w,z) = 5$

cost could always be 1, or inversely related to bandwidth, or inversely related to congestion

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

**key question:** what is the least-cost path between $u$ and $z$?

**routing algorithm:** algorithm that finds that least cost path
Routing algorithm classification

Q: 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

Q: static or dynamic?

static:
- routes change slowly over time

dynamic:
- routes change more quickly
  - periodic update
  - in response to link cost changes
Chapter 4: outline

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
Dijkstra’s Algorithm

1 Initialization:
2 \( N' = \{u\} \)
3 for all nodes \( v \)
4 if \( v \) adjacent to \( u \)
5 then \( D(v) = c(u, v) \)
6 else \( D(v) = \infty \)
7
8 Loop
9 find \( w \) not in \( N' \) such that \( D(w) \) is a minimum
10 add \( w \) to \( N' \)
11 update \( D(v) \) for all \( v \) adjacent to \( w \) and not in \( N' \):
12 \( D(v) = \min( D(v), D(w) + c(w,v) ) \)
13 /* new cost to \( v \) is either old cost to \( v \) or known 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)</th>
<th>D(w)</th>
<th>D(x)</th>
<th>D(y)</th>
<th>D(z)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>u</td>
<td>7,u</td>
<td>3,u</td>
<td>5,u</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>1</td>
<td>uw</td>
<td>6,w</td>
<td>5,u</td>
<td>11,w</td>
<td>∞</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>uwx</td>
<td>6,w</td>
<td></td>
<td>11,w</td>
<td>14,x</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>uwxv</td>
<td></td>
<td></td>
<td>10,v</td>
<td>14,x</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>uwxvy</td>
<td></td>
<td></td>
<td></td>
<td>12,y</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>uwxvzy</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

notes:
- construct shortest path tree by tracing predecessor nodes
- ties can exist (can be broken arbitrarily)
Dijkstra’s algorithm: another example

<table>
<thead>
<tr>
<th>Step</th>
<th>N'</th>
<th>D(v),p(v)</th>
<th>D(w),p(w)</th>
<th>D(x),p(x)</th>
<th>D(y),p(y)</th>
<th>D(z),p(z)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>u</td>
<td>2,u</td>
<td>5,u</td>
<td>1,u</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>1</td>
<td>ux</td>
<td>2,u</td>
<td>4,x</td>
<td>2,x</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>2</td>
<td>uxy</td>
<td>2,u</td>
<td>3,y</td>
<td>4,y</td>
<td>4,y</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>uxyv</td>
<td></td>
<td>3,y</td>
<td>4,y</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>uxyvw</td>
<td></td>
<td></td>
<td></td>
<td>4,y</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>uxyvwz</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

![Graph diagram](image-url)
Dijkstra’s algorithm: example (2)

resulting shortest-path tree from u:

resulting forwarding table in u:

<table>
<thead>
<tr>
<th>destination</th>
<th>link</th>
</tr>
</thead>
<tbody>
<tr>
<td>v</td>
<td>(u,v)</td>
</tr>
<tr>
<td>x</td>
<td>(u,x)</td>
</tr>
<tr>
<td>y</td>
<td>(u,x)</td>
</tr>
<tr>
<td>w</td>
<td>(u,x)</td>
</tr>
<tr>
<td>z</td>
<td>(u,x)</td>
</tr>
</tbody>
</table>
Dijkstra’s algorithm, discussion

*algorithm complexity:* n nodes
- each iteration: need to check all nodes, w, not in N
- $n(n+1)/2$ comparisons: $O(n^2)$
- more efficient implementations possible: $O(n \log n)$

*oscillations possible:*
- e.g., support link cost equals amount of carried traffic:

![Diagram showing network with nodes A, B, C, D and evolving costs over iterations.](Image)

Initially:
- Given these costs, find new routing, resulting in new costs.

Resulting in new costs:
- Given these costs, find new routing, resulting in new costs.

Resulting in new costs:
- Given these costs, find new routing, resulting in new costs.
When the Routers Disagree

(during transient periods)
Convergence

- Getting consistent routing information to all nodes
  - E.g., all nodes having the same link-state database

- Consistent forwarding after convergence
  - All nodes have the same link-state database
  - All nodes forward packets on shortest paths
  - The next router on the path forwards to the next hop
Transient Disruptions

- Detection delay
  - A node does not detect a failed link immediately
  - … and forwards data packets into a “blackhole”
  - Depends on timeout for detecting lost hellos
Transient Disruptions

- Inconsistent link-state database
  - Some routers know about failure before others
  - The shortest paths are no longer consistent
  - Can cause transient forwarding loops
Convergence Delay

- **Sources of convergence delay**
  - Detection latency
  - Flooding of link-state information
  - Shortest-path computation
  - Creating the forwarding table

- **Performance during convergence period**
  - Lost packets due to blackholes and TTL expiry
  - Looping packets consuming resources
  - Out-of-order packets reaching the destination

- **Very bad for VoIP, online gaming, and video**
Reducing Convergence Delay

- Faster detection
  - Smaller hello timers
  - Link-layer technologies that can detect failures
- Faster flooding
  - Flooding immediately
  - Sending link-state packets with high-priority
- Faster computation
  - Faster processors on the routers
  - Incremental Dijkstra’s algorithm
- Faster forwarding-table update
  - Data structures supporting incremental updates
Scaling Link-State Routing

- Overhead of link-state routing
  - Flooding link-state packets throughout the network
  - Running Dijkstra’s shortest-path algorithm

- Introducing hierarchy through “areas”
Some Properties

- Routing is a distributed algorithm
  - React to changes in the topology
  - Compute the paths through the network
- Shortest-path link state routing
  - Flood link weights throughout the network
  - Compute shortest paths as a sum of link weights
  - Forward packets on next hop in the shortest path
- Convergence process
  - Changing from one topology to another
  - Transient periods of inconsistency across routers
Chapter 4: outline

4.1 introduction
4.2 virtual circuit and datagram networks
4.3 what’s inside a router
4.4 IP: Internet Protocol
   - datagram format
   - IPv4 addressing
   - ICMP
   - IPv6

4.5 routing algorithms
   - link state
   - distance vector
   - hierarchical routing

4.6 routing in the Internet
   - RIP
   - OSPF
   - BGP

4.7 broadcast and multicast routing
**Distance vector algorithm**

*Bellman-Ford equation* (dynamic programming)

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

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

- cost from neighbor \( v \) to destination \( y \)
- cost to neighbor \( v \)
- \( \min \) taken over all neighbors \( v \) of \( x \)
Bellman-Ford example

clearly, $d_v(z) = 5$, $d_x(z) = 3$, $d_w(z) = 3$

B-F equation says:

$$d_u(z) = \min \{ c(u,v) + d_v(z), c(u,x) + d_x(z), c(u,w) + d_w(z) \}$$

$$= \min \{2 + 5, 1 + 3, 5 + 3\} = 4$$

node achieving minimum is next hop in shortest path, used in forwarding table
Distance vector algorithm

- \( D_x(y) = \) estimate of least cost from \( x \) to \( y \)
  - \( x \) maintains distance vector \( D_x = [D_x(y): y \in N] \)

- node \( x \):
  - knows cost to each neighbor \( v \): \( c(x,v) \)
  - maintains its neighbors’ distance vectors. For each neighbor \( v \), \( x \) maintains
    \( D_v = [D_v(y): y \in N] \)
Distance vector algorithm

**key idea:**

- from time-to-time, each node sends its own distance vector estimate to neighbors
- when \( x \) receives new DV estimate from neighbor, it updates its own DV using B-F equation:

  \[
  D_x(y) \leftarrow \min_v \{c(x,v) + D_v(y)\} \quad \text{for each node } y \in N
  \]

- under minor, natural conditions, the estimate \( D_x(y) \) converge to the actual least cost \( d_x(y) \)
Distance vector algorithm

iterative, asynchronous:
  each local iteration caused by:
  - local link cost change
  - DV update message from neighbor

distributed:
  - each node notifies neighbors only when its DV changes
    - neighbors then notify their neighbors if necessary

each node:

wait for (change in local link cost or msg from neighbor)

recompute estimates

if DV to any dest has changed, notify neighbors
$D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\}$
$= \min\{2+0, 7+1\} = 2$

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

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

Optimum 1-hop paths

<table>
<thead>
<tr>
<th>Table for A</th>
<th>Table for B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dst</td>
<td>Cst</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>4</td>
</tr>
<tr>
<td>C</td>
<td>∞</td>
</tr>
<tr>
<td>D</td>
<td>∞</td>
</tr>
<tr>
<td>E</td>
<td>2</td>
</tr>
<tr>
<td>F</td>
<td>6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table for C</th>
<th>Table for D</th>
<th>Table for E</th>
<th>Table for F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dst</td>
<td>Cst</td>
<td>Hop</td>
<td>Dst</td>
</tr>
<tr>
<td>A</td>
<td>∞</td>
<td>–</td>
<td>A</td>
</tr>
<tr>
<td>B</td>
<td>∞</td>
<td>–</td>
<td>B</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>D</td>
<td>D</td>
</tr>
<tr>
<td>E</td>
<td>∞</td>
<td>–</td>
<td>E</td>
</tr>
<tr>
<td>November 12</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

CMSC417 Set 6 143
### Distance Vector Example: Step 2

#### Optimum 2-hop paths

<table>
<thead>
<tr>
<th>Table for A</th>
<th>Table for B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dst</td>
<td>Cst</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>4</td>
</tr>
<tr>
<td>C</td>
<td>7</td>
</tr>
<tr>
<td>D</td>
<td>7</td>
</tr>
<tr>
<td>E</td>
<td>2</td>
</tr>
<tr>
<td>F</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table for C</th>
<th>Table for D</th>
<th>Table for E</th>
<th>Table for F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dst</td>
<td>Cst</td>
<td>Hop</td>
<td>Dst</td>
</tr>
<tr>
<td>A</td>
<td>7</td>
<td>F</td>
<td>A</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>F</td>
<td>B</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>D</td>
<td>D</td>
</tr>
<tr>
<td>E</td>
<td>4</td>
<td>F</td>
<td>E</td>
</tr>
</tbody>
</table>

November 12

F 2 C
F 3 F
F 0 F
## Distance Vector Example: Step 3

### Optimum 3-hop paths

<table>
<thead>
<tr>
<th>Table for A</th>
<th>Table for B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dst</td>
<td>Cst</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>4</td>
</tr>
<tr>
<td>C</td>
<td>6</td>
</tr>
<tr>
<td>D</td>
<td>7</td>
</tr>
<tr>
<td>E</td>
<td>2</td>
</tr>
<tr>
<td>F</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table for C</th>
<th>Table for D</th>
<th>Table for E</th>
<th>Table for F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dst</td>
<td>Cst</td>
<td>Hop</td>
<td>Dst</td>
</tr>
<tr>
<td>A</td>
<td>6</td>
<td>F</td>
<td>A</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>F</td>
<td>B</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>D</td>
<td>D</td>
</tr>
<tr>
<td>E</td>
<td>4</td>
<td>F</td>
<td>E</td>
</tr>
<tr>
<td>F</td>
<td>4</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

### Network Diagram

- A, B, C, D, E, F
- Connections and costs:
  - A to B: 2
  - A to C: 6
  - A to D: 7
  - A to E: 4
  - A to F: 5
  - B to A: 1
  - B to C: 3
  - B to D: 4
  - B to E: 1
  - B to F: 3
  - C to A: 1
  - C to B: 6
  - C to D: 1
  - C to E: 4
  - C to F: 1
  - D to A: 4
  - D to B: 1
  - D to C: 1
  - D to E: 5
  - D to F: 2
  - E to A: 3
  - E to B: 2
  - E to C: 6
  - E to D: 1
  - E to F: 0
  - F to A: 1
  - F to B: 5
  - F to C: 6
  - F to D: 2
  - F to E: 1

**Note:** The tables represent the cost (Cst) and hop count (Hop) for each node (Dst) in the network.
Distance Vector: Link Cost Changes

Link cost changes:

- Node detects local link cost change
- Updates the distance table
- If cost change in least cost path, notify neighbors

“good news travels fast”

algorithm terminates
Distance Vector: Link Cost Changes

Link cost changes:
- Good news travels fast
- Bad news travels slow - “count to infinity” problem!

algorithm continues on!
Distance Vector Routing

The count-to-infinity problem.
Distance Vector: Poison 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)
- Still, can have problems when more than 2 routers are involved

\[ D \quad X \quad Z \quad Y \]
\[ \begin{array}{c}
\text{Y} \\
\text{X} \\
\text{Z} \\
\text{Y} \\
\end{array} \]

algorithm terminates

\[ t_0 \quad t_1 \quad t_2 \quad t_3 \quad t_4 \]

**time**

November 12

\[ c(X,Y) \]

change
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”

$t_0$: $y$ detects link-cost change, updates its DV, informs its neighbors.

$t_1$: $z$ receives update from $y$, updates its table, computes new least cost to $x$, sends its neighbors its DV.

$t_2$: $y$ receives $z$’s update, updates its distance table. $y$’s least costs do *not* change, so $y$ does *not* send a message to $z$. 
Distance vector: link cost changes

link cost changes:
- node detects local link cost change
- *bad news travels slow* - “count to infinity” problem!
- 44 iterations before algorithm stabilizes: see text

poisoned reverse:
- If Z routes through Y to get to X:
  - Z tells Y its (Z’s) distance to X is infinite (so Y won’t route to X via Z)
- will this completely solve count to infinity problem?
Comparison of LS and DV algorithms

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

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

robustness: what happens if router malfunctions?
- **LS:**
  - node can advertise incorrect link cost
  - each node computes only its own table
- **DV:**
  - DV node can advertise incorrect path cost
  - each node’s table used by others
    - error propagate thru network
Chapter 4: outline

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 600 million destinations:
- can’t store all dest’s in routing tables!
- routing table exchange would swamp links!

administrative autonomy
- internet = network of networks
- each network admin may want to control routing in its own network
Hierarchical routing

- Aggregate routers into regions, "autonomous systems" (AS)
- Routers in same AS run same routing protocol
  - "intra-AS" routing protocol
  - Routers in different AS can run different intra-AS routing protocol

Gateway router:
- At "edge" of its own AS
- Has link to router in another AS
Interconnected ASes

- forwarding table configured by both intra- and inter-AS routing algorithm
  - intra-AS sets entries for internal dests
  - inter-AS & intra-AS sets entries for external dests
Inter-AS tasks

- Suppose router in AS1 receives datagram destined outside of AS1:
  - Router should forward packet to gateway router, but which one?

**AS1 must:**
1. Learn which dests are reachable through AS2, which through AS3
2. Propagate this reachability info to all routers in AS1

*Job of inter-AS routing!*

[Diagram showing inter-AS routing]
Example: setting forwarding table in router 1d

- suppose AS1 learns (via inter-AS protocol) that subnet x reachable via AS3 (gateway 1c), but not via AS2
  - inter-AS protocol propagates reachability info to all internal routers
- router 1d determines from intra-AS routing info that its interface I is on the least cost path to 1c
  - installs forwarding table entry \((x, I)\)
Example: choosing among multiple ASes

- now suppose AS1 learns from inter-AS protocol that subnet x is reachable from AS3 and from AS2.
- to configure forwarding table, router 1d must determine which gateway it should forward packets towards for dest x
  - this is also job of inter-AS routing protocol!
Example: choosing among multiple ASes

- now suppose AS1 learns from inter-AS protocol that subnet $x$ is reachable from AS3 \textit{and} from AS2.
- to configure forwarding table, router 1d must determine towards which gateway it should forward packets for dest $x$:
  - this is also job of inter-AS routing protocol!
- hot potato routing: send packet towards closest of two routers.

<table>
<thead>
<tr>
<th>learn from inter-AS protocol that subnet $x$ is reachable via multiple gateways</th>
<th>use routing info from intra-AS protocol to determine costs of least-cost paths to each of the gateways</th>
<th>hot potato routing: choose the gateway that has the smallest least cost</th>
<th>determine from forwarding table the interface $I$ that leads to least-cost gateway. Enter $(x,I)$ in forwarding table</th>
</tr>
</thead>
</table>
Chapter 4: outline

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)
RIP (Routing Information Protocol)

- included in BSD-UNIX distribution in 1982
- distance vector algorithm
  - distance metric: # hops (max = 15 hops), each link has cost 1
  - DVs exchanged with neighbors every 30 sec in response message (aka advertisement)
  - each advertisement: list of up to 25 destination subnets (in IP addressing sense)

```
from router A to destination subnets:

<table>
<thead>
<tr>
<th>subnet</th>
<th>hops</th>
</tr>
</thead>
<tbody>
<tr>
<td>u</td>
<td>1</td>
</tr>
<tr>
<td>v</td>
<td>2</td>
</tr>
<tr>
<td>w</td>
<td>2</td>
</tr>
<tr>
<td>x</td>
<td>3</td>
</tr>
<tr>
<td>y</td>
<td>3</td>
</tr>
<tr>
<td>z</td>
<td>2</td>
</tr>
</tbody>
</table>
```
RIP: example

![Diagram of a network with routers A, B, C, and D connected by links labeled w, x, y, and z.]

Routing table in router D:

<table>
<thead>
<tr>
<th>destination subnet</th>
<th>next router</th>
<th># 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>x</td>
<td>--</td>
<td>1</td>
</tr>
<tr>
<td>....</td>
<td>....</td>
<td>....</td>
</tr>
</tbody>
</table>
**RIP: example**

A-to-D advertisement

<table>
<thead>
<tr>
<th>dest</th>
<th>next</th>
<th>hops</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>x</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>z</td>
<td>C</td>
<td>4</td>
</tr>
</tbody>
</table>

Routing table in router D

<table>
<thead>
<tr>
<th>destination subnet</th>
<th>next router</th>
<th># hops to dest</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>A</td>
<td>2</td>
</tr>
<tr>
<td>y</td>
<td>B</td>
<td>2</td>
</tr>
<tr>
<td>z</td>
<td>B</td>
<td>7</td>
</tr>
<tr>
<td>x</td>
<td>--</td>
<td>1</td>
</tr>
<tr>
<td>....</td>
<td>....</td>
<td>....</td>
</tr>
</tbody>
</table>
RIP: link failure, 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
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
- advertisements flooded to entire AS
  - carried in OSPF messages directly over IP (rather than TCP or UDP)
- *IS-IS routing* protocol: nearly identical to OSPF
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 ToS; high for real time ToS)
- integrated uni- and **multicast** support:
  - Multicast OSPF (MOSPF) uses same topology database as OSPF
- **hierarchical** OSPF in large domains.
Hierarchical OSPF

- **two-level hierarchy:** local area, backbone.
  - link-state advertisements only in area
  - each nodes 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.
Internet inter-AS routing: BGP

- **BGP (Border Gateway Protocol):** *the de facto inter-domain routing protocol*
  - “glue that holds the Internet together”

- **BGP provides each AS a means to:**
  - **eBGP**: obtain subnet reachability information from neighboring ASs.
  - **iBGP**: propagate reachability information to all AS-internal routers.
  - determine “good” routes to other networks based on reachability information and policy.

- allows subnet to advertise its existence to rest of Internet: “*I am here*”
BGP basics

- **BGP session**: two BGP routers ("peers") exchange BGP messages:
  - advertising *paths* to different destination network prefixes ("path vector" protocol)
  - exchanged over semi-permanent TCP connections

- **when AS3 advertises a prefix to AS1**:
  - AS3 *promises* it will forward datagrams towards that prefix
  - AS3 can aggregate prefixes in its advertisement
BGP basics: distributing path information

- using eBGP session between 3a and 1c, AS3 sends prefix reachability info to AS1.
  - 1c can then use iBGP to distribute new prefix info to all routers in AS1
  - 1b can then re-advertise new reachability info to AS2 over 1b-to-2a eBGP session
- when router learns of new prefix, it creates entry for prefix in its forwarding table.
BGP—Exterior Routing Protocol

Common policy distinction is transit vs. peering:

- Transit carries traffic for pay; peers for mutual benefit
- AS1 carries AS2↔AS4 (Transit) but not AS3 (Peer)
BGP— Exterior Routing Protocol

- BGP propagates messages along policy-compliant routes
  - Message has prefix, AS path (to detect loops) and next-hop IP (to send over the local network)
Path attributes and BGP routes

- advertised prefix includes BGP attributes
  - prefix + attributes = “route”
- two important attributes:
  - **AS-PATH**: contains ASs through which prefix advertisement has passed: e.g., AS 67, AS 17
  - **NEXT-HOP**: indicates specific internal-AS router to next-hop AS. (may be multiple links from current AS to next-hop-AS)
- gateway router receiving route advertisement uses **import policy** to accept/decline
  - e.g., never route through AS x
  - *policy-based* routing
BGP route selection

- router may learn about more than 1 route to destination AS, selects route based on:
  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 between peers over TCP connection
- BGP messages:
  - **OPEN**: opens TCP connection to peer and authenticates sender
  - **UPDATE**: advertises new path (or withdraws old)
  - **KEEPALIVE**: keeps connection alive in absence of UPDATES; also ACKs OPEN request
  - **NOTIFICATION**: reports errors in previous msg; also used to close connection
BGP routing policy

- A, B, C are *provider networks*
- X, W, Y are customer (of provider networks)
- X is *dual-homed*: attached to two networks
  - X does not want to route from B via X to C
  - .. so X will not advertise to B a route to C

Legend:
- Blue circle: provider network
- Blue hexagon: customer network
BGP routing policy (2)

- A advertises path AW to B
- B advertises path BAW to X
- Should B advertise path BAW to C?
  - No way! B gets no “revenue” for routing CBAW since neither W nor C are B’s customers
  - B wants to force C to route to w via A
  - B wants to route *only* to/from its customers!
Why different Intra-, Inter-AS routing?

**Policy:**
- inter-AS: admin wants control over how its traffic routed, who routes through its net.
- intra-AS: single admin, so no policy decisions needed

**Scale:**
- hierarchical routing saves table size, reduced update traffic

**Performance:**
- intra-AS: can focus on performance
- inter-AS: policy may dominate over performance
Chapter 4: outline

4.1 introduction
4.2 virtual circuit and datagram networks
4.3 what’s inside a router
4.4 IP: Internet Protocol
   ▪ datagram format
   ▪ IPv4 addressing
   ▪ ICMP
   ▪ IPv6
4.5 routing algorithms
   ▪ link state
   ▪ distance vector
   ▪ hierarchical routing
4.6 routing in the Internet
   ▪ RIP
   ▪ OSPF
   ▪ BGP
4.7 broadcast and multicast routing
Broadcast and Multicast

- **Broadcast**
  - Send to ALL nodes

- **Multicast**
  - Send to a selected set of nodes
    - Multicast Group
  - Multicast Address
    - Address indirection
  - Internet Group Management Protocol (IGMP)
**Broadcast routing**

- deliver packets from source to all other nodes
- source duplication is inefficient:
  - source duplication: how does source determine recipient addresses?

![Diagram of network routing](image)

- source duplication
- in-network duplication
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
Broadcast Algorithms in Practice

- **Gnutella**
  - Sequence number controlled flooding
    - 16 bit identifier
    - 16 bit payload descriptor
    - TTL field
      - Limited scope flooding

- **OSPF**
  - Link State Advertisements (LSAs)
    - 32 bit sequence number
    - 16 bit age field
Multicast

Applications
- Software distribution
- Streaming media
- Shared data applications (Whiteboard, Teleconferencing)
- Data feeds – stock quotes
- Web cache updating
- Interactive gaming
- …
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

---

**legend**

- **group member**
- **not group member**
- **router with a group member**
- **router without group member**

---

**shared tree**

**source-based trees**
IGMP

- Operates between a host and its directly attached router
- Used to inform the router that an application on the host wants to join a multicast group
- Messages
  - membership_query
  - membership_report
  - leave_group (optional)
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

[Diagram of network with labeled routers and links]
Reverse path forwarding

- rely on router’s knowledge of unicast shortest path from it to sender
- each router has simple forwarding behavior:

\[
\begin{align*}
\text{if} \ (\text{mcast datagram received on incoming link on shortest path back to center}) \\
\text{then} \ \text{flood datagram onto all outgoing links} \\
\text{else} \ \text{ignore datagram}
\end{align*}
\]
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 R6 chosen as center:

LEGEND

- router with attached group member
- router with no attached group member
- path order in which join messages generated
Internet Multicasting Routing: DVMRP

- **DVMRP**: distance vector multicast routing protocol, RFC1075
- **flood and prune**: reverse path forwarding, source-based tree
  - RPF tree based on DVMRP’s own routing tables constructed by communicating DVMRP routers
  - no assumptions about underlying unicast
  - initial datagram to mcast group flooded everywhere via RPF
  - routers not wanting group: send upstream prune msgs
DVMRP: continued…

- **soft state:** DVMRP router periodically (1 min.) “forgets” branches are pruned:
  - mcast data again flows down unpruned branch
  - downstream router: reprune or else continue to receive data
- routers can quickly regraft to tree
  - following IGMP join at leaf
- odds and ends
  - commonly implemented in commercial router
**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 wrt # interconnected networks
- group members “widely dispersed”
- bandwidth not plentiful
Consequences of sparse-dense dichotomy:

**dense**
- group membership by routers *assumed* until routers explicitly prune
- *data-driven* construction on mcast tree (e.g., RPF)
- bandwidth and non-group-router processing *profligate*

**sparse:**
- no membership until routers explicitly join
- *receiver-driven* construction of mcast tree (e.g., center-based)
- bandwidth and non-group-router processing *conservative*
**PIM- dense mode**

**flood-and-prune RPF**: similar to DVMRP but…

- underlying unicast protocol provides RPF info for incoming datagram
- less complicated (less efficient) downstream flood than DVMRP reduces reliance on underlying routing algorithm
- has protocol mechanism for router to detect it is a leaf-node router
PIM - sparse mode

- center-based approach
- router sends *join* msg to rendezvous point (RP)
  - intermediate routers update state and forward *join*
- after joining via RP, router can switch to source-specific tree
  - increased performance: less concentration, shorter paths
**PIM - sparse mode**

**sender(s):**
- unicast data to RP, which distributes down RP-rooted tree
- RP can extend mcast tree upstream to source
- RP can send *stop* msg if no attached receivers  
  - “no one is listening!”

![Diagram of PIM - sparse mode](image)
Chapter 4: done!

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

- 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