The Network Layer
Network Layer Design Issues

- Store-and-Forward Packet Switching
- Services Provided to the Transport Layer
- Implementation of Connectionless Service
- Implementation of Connection-Oriented Service
- Comparison of Virtual-Circuit and Datagram Subnets
Packet Switching (e.g., Internet)

- Data traffic divided into packets
  - Each packet contains a header (with address)
- Packets travel separately through network
  - Packet forwarding based on the header
  - Network nodes may store packets temporarily
- Destination reconstructs the message
Packet Switching: Statistical Multiplexing
IP Service: Best-Effort Packet Delivery

• Packet switching
  – Divide messages into a sequence of packets
  – Headers with source and destination address

• Best-effort delivery
  – Packets may be lost
  – Packets may be corrupted
  – Packets may be delivered out of order
IP Service Model: Why Packets?

- Data traffic is bursty
  - Logging in to remote machines
  - Exchanging e-mail messages
- Don’t want to waste reserved bandwidth
  - No traffic exchanged during idle periods
- Better to allow multiplexing
  - Different transfers share access to same links
- Packets can be delivered by most anything
  - RFC 2549: IP over Avian Carriers (aka birds)
- ... still, packet switching can be inefficient
  - Extra header bits on every packet
IP Service Model: Why Best-Effort?

• IP means never having to say you’re sorry...
  – Don’t need to reserve bandwidth and memory
  – Don’t need to do error detection & correction
  – Don’t need to remember from one packet to next

• Easier to survive failures
  – Transient disruptions are okay during failover

• ... but, applications do want efficient, accurate transfer of data in order, in a timely fashion
IP Service: Best-Effort is Enough

• No error detection or correction
  – Higher-level protocol can provide error checking
• Successive packets may not follow the same path
  – Not a problem as long as packets reach the destination
• Packets can be delivered out-of-order
  – Receiver can put packets back in order (if necessary)
• Packets may be lost or arbitrarily delayed
  – Sender can send the packets again (if desired)
• No network congestion control (beyond “drop”)
  – Sender can slow down in response to loss or delay
Layering in the IP Protocols

- HTTP
- Telnet
- FTP
- DNS
- RTP

Transmission Control Protocol (TCP)

User Datagram Protocol (UDP)

Internet Protocol

- SONET
- Ethernet
- ATM
History: Why IP Packets?

- IP proposed in the early 1970s
  - Defense Advanced Research Project Agency (DARPA)
- Goal: connect existing networks
  - To develop an effective technique for multiplexed utilization of existing interconnected networks
  - E.g., connect packet radio networks to the ARPAnet
- Motivating applications
  - Remote login to server machines
  - Inherently bursty traffic with long silent periods
- Prior ARPAnet experience with packet switching
  - Previous DARPA project
  - Demonstrated store-and-forward packet switching
Other Main Driving Goals (In Order)

• Communication should continue despite failures
  – Survive equipment failure or physical attack
  – Traffic between two hosts continue on another path

• Support multiple types of communication services
  – Differing requirements for speed, latency, & reliability
  – Bidirectional reliable delivery vs. message service

• Accommodate a variety of networks
  – Both military and commercial facilities
  – Minimize assumptions about the underlying network
Other Driving Goals, Somewhat Met

• Permit distributed management of resources
  – Nodes managed by different institutions
  – ... though this is still rather challenging
• Cost-effectiveness
  – Statistical multiplexing through packet switching
  – ... though packet headers and retransmissions wasteful
• Ease of attaching new hosts
  – Standard implementations of end-host protocols
  – ... though still need a fair amount of end-host software
• Accountability for use of resources
  – Monitoring functions in the nodes
  – ... though this is still fairly limited and immature
# IP Packet Structure

<table>
<thead>
<tr>
<th>4-bit Version</th>
<th>4-bit Header Length</th>
<th>8-bit Type of Service (TOS)</th>
<th>16-bit Total Length (Bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>16-bit Identification</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3-bit Flags</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>13-bit Fragment Offset</td>
</tr>
<tr>
<td>8-bit Time to Live (TTL)</td>
<td>8-bit Protocol</td>
<td>16-bit Header Checksum</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>32-bit Source IP Address</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>32-bit Destination IP Address</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Options (if any)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Payload</td>
</tr>
</tbody>
</table>
The IP Protocol

<table>
<thead>
<tr>
<th>Version</th>
<th>IHL</th>
<th>Type of service</th>
<th>Total length</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Identification</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Time to live</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Protocol</td>
<td>Header checksum</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Source address</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Destination address</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Options (0 or more words)</td>
<td></td>
</tr>
</tbody>
</table>

32 Bits
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

Hop number, IP address, DNS name

1  169.229.62.1  inr-daedalus-0.CS.Berkeley.EDU
2  169.229.59.225  soda-cr-1-1-soda-br-6-2
3  128.32.255.169  vlan242.inr-202-doecev.Berkeley.EDU
4  128.32.0.249  gigE6-0-0.inr-666-doecev.Berkeley.EDU
5  128.32.0.66  qsv-juniper--ucb-gw.calren2.net
6  209.247.159.109  POS1-0.hsipaccess1.SanJose1.Level3.net
7  *                  
8  64.159.1.46                  
9  209.247.9.170  pos8-0.hsa2.Atlanta2.Level3.net
10  66.185.138.33  pop2-atm-P0-2.atdn.net
11  *                  
12  66.185.136.17  pop1-atl-P4-0.atdn.net
13  64.236.16.52  www4.cnn.com

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

\[
\begin{array}{c}
134 \\
+ 212 \\
\hline
= 346
\end{array}
\quad \quad
\begin{array}{c}
134 \\
+ 216 \\
\hline
= 350
\end{array}
\]

Mismatch!
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 Addressing and Forwarding
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) 0  Network | Host

(b) 1 0  Network | Host

(c) 1 1 0  Network | Host
IP Addresses

32 Bits

<table>
<thead>
<tr>
<th>Class</th>
<th>Prefix</th>
<th>Description</th>
<th>Range of host addresses</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>Network, 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>128.0.0.0 to 191.255.255.255</td>
</tr>
<tr>
<td>C</td>
<td>110</td>
<td>Network, Host</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>
<tr>
<td>E</td>
<td>1111</td>
<td>Reserved for future use</td>
<td>240.0.0.0 to 255.255.255.255</td>
</tr>
</tbody>
</table>
### IP Addresses (2)

#### Special IP addresses.

<table>
<thead>
<tr>
<th>Network</th>
<th>1111</th>
<th>...</th>
<th>1111</th>
<th>127</th>
<th>(Anything)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This host</td>
<td></td>
<td></td>
<td></td>
<td>Host</td>
</tr>
<tr>
<td></td>
<td>A host on this network</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Broadcast on the local network</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Broadcast on a distant network</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Loopback</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Subnets

A campus network consisting of LANs for various departments.
A class B network subnetted into 64 subnets.
Subnetted Address

Network number | Host number

Class B address

```
11111111111111111111111111111111
00000000
```

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
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

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>34</td>
<td>158</td>
<td>5</td>
</tr>
</tbody>
</table>

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>00001100</td>
<td>00100010</td>
<td>10011110</td>
<td>00000101</td>
</tr>
</tbody>
</table>

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>11111111</td>
<td>11111111</td>
<td>11111111</td>
<td>00000000</td>
</tr>
</tbody>
</table>

Mask

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>255</td>
<td>255</td>
<td>255</td>
<td>0</td>
</tr>
</tbody>
</table>
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

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.2.3.4</td>
<td>1.2.3.7</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5.6.7.8</td>
<td>5.6.7.9</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

forwarding table

October 09
CMSC417 Set 4
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

1.2.3.4  1.2.3.7  1.2.3.156  5.6.7.8  5.6.7.9  5.6.7.212

<table>
<thead>
<tr>
<th>host</th>
<th>host</th>
<th>...</th>
<th>host</th>
<th>host</th>
<th>...</th>
<th>host</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAN 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>host</th>
<th>host</th>
<th>...</th>
<th>host</th>
<th>host</th>
<th>...</th>
<th>host</th>
</tr>
</thead>
<tbody>
<tr>
<td>WAN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1.2.3.0/24</th>
<th>5.6.7.0/24</th>
</tr>
</thead>
<tbody>
<tr>
<td>forward</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>forwarding table</th>
<th>5.6.7.213</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>host</td>
</tr>
</tbody>
</table>
Address Allocation
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

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.
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
Packet Forwarding
Hop-by-Hop Packet Forwarding

• Each router has a forwarding table
  – Maps destination addresses...
  – ... to outgoing interfaces

• Upon receiving a packet
  – Inspect the destination IP address in the header
  – Index into the table
  – Determine the outgoing interface
  – Forward the packet out that interface

• Then, the next router in the path repeats
  – And the packet travels along the path to the destination
Separate Table Entries Per Address

- If a router had a forwarding entry per IP address
  - Match *destination address* of incoming packet
  - ... to the *forwarding-table entry*
  - ... to determine the *outgoing interface*
Separate Entry Per 24-bit Prefix

- If the router had an entry per 24-bit prefix
  - Look only at the top 24 bits of the destination address
  - Index into the table to determine the next-hop interface

```
1.2.3.4  1.2.3.7  1.2.3.156
host     host    host
LAN 1

5.6.7.8  5.6.7.9  5.6.7.212
host     host    host
LAN
```

```
1.2.3.0/24
5.6.7.0/24
forwarding table
```
Separate Entry Classful Address

• If the router had an entry per classful prefix
  – Mixture of Class A, B, and C addresses
  – Depends on the first couple of bits of the destination

• Identify the mask automatically from the address
  – First bit of 0: class A address (/8)
  – First two bits of 10: class B address (/16)
  – First three bits of 110: class C address (/24)

• Then, look in the forwarding table for the match
  – E.g., 1.2.3.4 maps to 1.2.3.0/24
  – Then, look up the entry for 1.2.3.0/24
  – … to identify the outgoing interface
CIDR Makes Packet Forwarding Harder

• There’s no such thing as a free lunch
  – CIDR allows efficient use of the limited address space
  – But, CIDR makes packet forwarding much harder
• Forwarding table may have many matches
  – E.g., table entries for 201.10.0.0/21 and 201.10.6.0/23

![Diagram showing the forwarding table entries and their relationships with different providers.](image)
Longest Prefix Match Forwarding

- Forwarding tables in IP routers
  - Maps each IP prefix to next-hop link(s)
- Destination-based forwarding
  - Packet has a destination address
  - Router identifies longest-matching prefix
  - Cute algorithmic problem: very fast lookups

<table>
<thead>
<tr>
<th>destination</th>
<th>forwarding table</th>
</tr>
</thead>
<tbody>
<tr>
<td>201.10.6.17</td>
<td>4.0.0.0/8</td>
</tr>
<tr>
<td></td>
<td>4.83.128.0/17</td>
</tr>
<tr>
<td></td>
<td>201.10.0.0/21</td>
</tr>
<tr>
<td></td>
<td>201.10.6.0/23</td>
</tr>
<tr>
<td></td>
<td>126.255.103.0/24</td>
</tr>
</tbody>
</table>

outgoing link: Serial0/0.1
Simplest Algorithm is Too Slow

- Scan the forwarding table one entry at a time
  - See if the destination matches the entry
  - If so, check the size of the mask for the prefix
  - Keep track of the entry with longest-matching prefix

- Overhead is linear in size of the forwarding table
  - Today, that means 150,000-200,000 entries!
  - And, the router may have just a few nanoseconds
  - ... before the next packet is arriving

- Need greater efficiency to keep up with \textit{line rate}
  - Better algorithms
  - Hardware implementations
Patricia Tree

• Store the prefixes as a tree
  – One bit for each level of the tree
  – Some nodes correspond to valid prefixes
  – ... which have next-hop interfaces in a table

• When a packet arrives
  – Traverse the tree based on the destination address
  – Stop upon reaching the longest matching prefix
Even Faster Lookups

• Patricia tree is faster than linear scan
  – Proportional to number of bits in the address
• Patricia tree can be made faster
  – Can make a k-ary tree
    • E.g., 4-ary tree with four children (00, 01, 10, and 11)
    – Faster lookup, though requires more space
• Can use special hardware
  – Content Addressable Memories (CAMs)
  – Allows look-ups on a key rather than flat address
• Huge innovations in the mid-to-late 1990s
  – After CIDR was introduced (in 1994)
  – ... and longest-prefix match was a major bottleneck
Where do Forwarding Tables Come From?

- Routers have forwarding tables
  - Map prefix to outgoing link(s)
- Entries can be statically configured
  - E.g., “map 12.34.158.0/24 to Serial0/0.1”
- But, this doesn’t adapt
  - To failures
  - To new equipment
  - To the need to balance load
  - ...
- That is where other technologies come in...
  - Routing protocols, DHCP, and ARP (later in course)
What End Hosts Sending to Others?

- End host with single network interface
  - PC with an Ethernet link
  - Laptop with a wireless link
- Don’t need to run a routing protocol
  - Packets to the host itself (e.g., 1.2.3.4/32)
    - Delivered locally
  - Packets to other hosts on the LAN (e.g., 1.2.3.0/24)
    - Sent out the interface
  - Packets to external hosts (e.g., 0.0.0.0/0)
    - Sent out interface to local gateway
- How this information is learned
  - Static setting of address, subnet mask, and gateway
  - Dynamic Host Configuration Protocol (DHCP)
What About Reaching the End Hosts?

• How does the last router reach the destination?
  - 1.2.3.4
  - 1.2.3.7
  - 1.2.3.156

• Each interface has a permanent, global identifier
  - MAC (Media Access Control) address
  - Burned in to the adaptors Read-Only Memory (ROM)
  - Flat address structure (i.e., no hierarchy)

• Constructing an address resolution table
  - Mapping MAC address to/from IP address
  - Address Resolution Protocol (ARP)
Conclusions

• IP address
  – A 32-bit number
  – Allocated in prefixes
  – Non-uniform hierarchy for scalability and flexibility

• Packet forwarding
  – Based on IP prefixes
  – Longest-prefix-match forwarding

• We’ll cover some topics later
  – Routing protocols, DHCP, and ARP
Routing
Routing Algorithms

• The Optimality Principle
• Shortest Path Routing
• Flooding
• Distance Vector Routing
• Link State Routing
• Hierarchical Routing
• Broadcast Routing
• Multicast Routing
• Routing for Mobile Hosts
• Routing in Ad Hoc Networks
Routing Algorithms (2)

Conflict between fairness and optimality.
What is Routing?

- A famous quotation from RFC 791
  
  “A name indicates what we seek. An address indicates where it is. A route indicates how we get there.”
  -- Jon Postel
Routing vs. Forwarding

• Routing: control plane
  – Computing paths the packets will follow
  – Routers talking amongst themselves
  – Individual router creating a forwarding table

• Forwarding: data plane
  – Directing a data packet to an outgoing link
  – Individual router using a forwarding table
Data and Control Planes

- Processor
- Switching Fabric
- Line card
- Data plane
- Control plane
Router Physical Layout

Juniper T series
Cisco 12000

Switch
Linecards
Line Cards (Interface Cards, Adaptors)

- **Interfacing**
  - Physical link
  - Switching fabric

- **Packet handling**
  - Packet forwarding
  - Decrement time-to-live
  - Buffer management
    - Link scheduling
    - Packet filtering
    - Rate limiting
    - Packet marking
    - Measurement
Switching Fabric

• Deliver packet inside the router
  – From incoming interface to outgoing interface
  – A small network in and of itself

• Must operate very quickly
  – Multiple packets going to same outgoing interface
  – Switch scheduling to match inputs to outputs

• Implementation techniques
  – Bus, crossbar, interconnection network, ...
  – Running at a faster speed (e.g., 2X) than links
  – Dividing variable-length packets into fixed-size cells
Packet Switching

- Link 1, ingress
- Link 1, egress
- Link 2, ingress
- Link 2, egress
- Link 3, ingress
- Link 3, egress
- Link 4, ingress
- Link 4, egress

Choose Egress

Choose Egress

Choose Egress

Choose Egress

Link 1, ingress
Link 2, ingress
Link 3, ingress
Link 4, ingress

Link 1, egress
Link 2, egress
Link 3, egress
Link 4, egress

"4"
Router Processor

• So-called “Loopback” interface
  – IP address of the CPU on the router
• Interface to network administrators
  – Command-line interface for configuration
  – Transmission of measurement statistics
• Handling of special data packets
  – Packets with IP options enabled
  – Packets with expired Time-To-Live field
• Control-plane software
  – Implementation of the routing protocols
  – Creation of forwarding table for the line cards
Where do Forwarding Tables Come From?

• Routers have forwarding tables
  – Map IP prefix to outgoing link(s)
• Entries can be statically configured
  – E.g., “map 12.34.158.0/24 to Serial0/0.1”
• But, this doesn’t adapt
  – To failures
  – To new equipment
  – To the need to balance load
• That is where routing protocols come in
Computing Paths Between Routers

• Routers need to know two things
  – Which router to use to reach a destination prefix
  – Which outgoing interface to use to reach that router

• just how routers reach each other
  – How you knows how to forward packets toward z

Interface along the path to z

Router z that can reach destination

12.34.158.0/24
(a) A subnet. (b) A sink tree for router B.
Computing the Shortest Paths

(assuming you already know the topology)
Shortest-Path Routing

• Path-selection model
  – Destination-based
  – Load-insensitive (e.g., static link weights)
  – Minimum hop count or sum of link weights
Shortest-Path Problem

• Given: network topology with link costs
  – \( c(x,y) \): link cost from node \( x \) to node \( y \)
  – Infinity if \( x \) and \( y \) are not direct neighbors

• Compute: least-cost paths to all nodes
  – From a given source \( u \) to all other nodes
  – \( p(v) \): predecessor node along path from source to \( v \)
Dijkstra’s Shortest-Path Algorithm

• Iterative algorithm
  – After k iterations, know least-cost path to k nodes

• $S$: nodes whose least-cost path definitively known
  – Initially, $S = \{u\}$ where $u$ is the source node
  – Add one node to $S$ in each iteration

• $D(v)$: current cost of path from source to node $v$
  – Initially, $D(v) = c(u,v)$ for all nodes $v$ adjacent to $u$
  – $\ldots$ and $D(v) = \infty$ for all other nodes $v$
  – Continually update $D(v)$ as shorter paths are learned
Dijsktra’s Algorithm

1 Initialization:
2 \( S = \{u\} \)
3 for all nodes \( v \)
4 \( \text{if (v is adjacent to u)} \)
5 \( D(v) = c(u,v) \)
6 \( \text{else } D(v) = \infty \)

8 Loop
9 find \( w \) not in \( S \) with the smallest \( D(w) \)
10 add \( w \) to \( S \)
11 update \( D(v) \) for all \( v \) adjacent to \( w \) and not in \( S \):
12 \( D(v) = \min\{D(v), D(w) + c(w,v)\} \)
13 until all nodes in \( S \)
Dijkstra’s Algorithm Example
Dijkstra’s Algorithm Example
Shortest-Path Tree

- Shortest-path tree from u
- Forwarding table at u

\[
\begin{array}{c|c}
\text{link} & \\
\hline
v & (u, v) \\
w & (u, w) \\
x & (u, w) \\
y & (u, v) \\
z & (u, v) \\
s & (u, w) \\
t & (u, w) \\
\end{array}
\]
Shortest Path Routing

The first 5 steps used in computing the shortest path from A to D.
Dijkstra's algorithm to compute the shortest path through a graph.

#include <iostream>

#define MAX_NODES 1024 /* maximum number of nodes */
#define INFINITY 10000000000 /* a number larger than every maximum path */

int n, dist[MAX_NODES][MAX_NODES]; /* dist[i][j] is the distance from i to j */

void shortest_path(int s, int t, int path[])
{
    struct state {
        int predecessor; /* previous node */
        int length; /* length from source to this node */
        enum {permanent, tentative} label; /* label state */
    } state[MAX_NODES];

    int i, k, min;
    struct state *p;

    for (p = &state[0]; p < &state[n]; p++) /* initialize state */
    {
        p->predecessor = -1;
        p->length = INFINITY;
        p->label = tentative;
    }

    state[t].length = 0; state[t].label = permanent;
    k = t; /* k is the initial working node */
Dijkstra's algorithm to compute the shortest path through a graph.
Learning the Topology

(by the routers talking among themselves)
Link-State Routing

• Each router keeps track of its incident links
  – Whether the link is up or down
  – The cost on the link
• Each router broadcasts the link state
  – To give every router a complete view of the graph
• Each router runs Dijkstra’s algorithm
  – To compute the shortest paths
  – ... and construct the forwarding table
• Example protocols
  – Open Shortest Path First (OSPF)
  – Intermediate System – Intermediate System (IS-IS)
Link State Routing

Each router must do the following:

1. Discover its neighbors, learn their network address.
2. Measure the delay or cost to each of its neighbors.
3. Construct a packet telling all it has just learned.
4. Send this packet to all other routers.
5. Compute the shortest path to every other router.
Learning about the Neighbors

(a) Nine routers and a LAN. (b) A graph model of (a).
Measuring Line Cost

A subnet in which the East and West parts are connected by two lines.
Building Link State Packets

(a) A subnet. (b) The link state packets for this subnet.
Distributing the Link State Packets

The packet buffer for router B in the previous slide

<table>
<thead>
<tr>
<th>Source</th>
<th>Seq.</th>
<th>Age</th>
<th>Send flags</th>
<th>ACK flags</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>21</td>
<td>60</td>
<td>0 1 1</td>
<td>1 0 0</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>21</td>
<td>60</td>
<td>1 1 0</td>
<td>0 0 1</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>21</td>
<td>59</td>
<td>0 1 0</td>
<td>1 0 1</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>20</td>
<td>60</td>
<td>1 0 1</td>
<td>0 1 0</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>21</td>
<td>59</td>
<td>1 0 0</td>
<td>0 1 1</td>
<td></td>
</tr>
</tbody>
</table>
Detecting Topology Changes

• Beaconing
  – Periodic “hello” messages in both directions
  – Detect a failure after a few missed “hellos”

• Performance trade-offs
  – Detection speed
  – Overhead on link bandwidth and CPU
  – Likelihood of false detection
Broadcasting the Link State

• Flooding
  – Node sends link-state information out its links
  – And then the next node sends out all of its links
  – ... except the one where the information arrived
Broadcasting the Link State

• Reliable flooding
  – Ensure all nodes receive link-state information
  – ... and that they use the latest version

• Challenges
  – Packet loss
  – Out-of-order arrival

• Solutions
  – Acknowledgments and retransmissions
  – Sequence numbers
  – Time-to-live for each packet
When to Initiate Flooding

• Topology change
  – Link or node failure
  – Link or node recovery

• Configuration change
  – Link cost change

• Periodically
  – Refresh the link-state information
  – Typically (say) 30 minutes
  – Corrects for possible corruption of the data
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
(a) A subnet. (b) Input from A, I, H, K, and the new routing table for J.
Distance Vector Algorithm

- \( c(x,v) \) = cost for direct link from \( x \) to \( v \)
  - Node \( x \) maintains costs of direct links \( c(x,v) \)
- \( D_x(y) \) = estimate of least cost from \( x \) to \( y \)
  - Node \( x \) maintains distance vector \( D_x = [D_x(y): y \in N] \)
- Node \( x \) maintains its neighbors’ distance vectors
  - For each neighbor \( v \), \( x \) maintains \( D_v = [D_v(y): y \in N] \)
- Each node \( v \) periodically sends \( D_v \) to its neighbors
  - And neighbors update their own distance vectors
    - \( D_x(y) \leftarrow \min_v \{c(x,v) + D_v(y)\} \) for each node \( y \in N \)
- Over time, the distance vector \( D_x \) converges
Distance Vector Algorithm

Iterative, asynchronous: each local iteration caused by:

• Local link cost change
• Distance vector update message from neighbor

Distributed:

• Each node notifies neighbors \textit{only} when its DV changes
• Neighbors then notify their neighbors if necessary

Each node:

\textit{wait} for (change in local link cost or message from neighbor)

\textit{recompute} estimates

if distance to any destination has changed, \textit{notify} neighbors
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><strong>Dst</strong></td>
<td><strong>Cst</strong></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><strong>Dst</strong></td>
<td><strong>Cst</strong></td>
<td><strong>Hop</strong></td>
<td><strong>Dst</strong></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>F</td>
<td>1</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>
## 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><strong>Dst</strong></td>
<td><strong>Cst</strong></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><strong>Dst</strong></td>
<td><strong>Cst</strong></td>
<td><strong>Hop</strong></td>
<td><strong>Dst</strong></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>
<tr>
<td>F</td>
<td>1</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>
# Distance Vector Example: Step 3

## Optimum 3-hop paths

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

<table>
<thead>
<tr>
<th>Dst</th>
<th>Cst</th>
<th>Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4</td>
<td>A</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>B</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
<td>F</td>
</tr>
<tr>
<td>D</td>
<td>3</td>
<td>D</td>
</tr>
<tr>
<td>E</td>
<td>4</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>1</td>
<td>F</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Dst</th>
<th>Cst</th>
<th>Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>7</td>
<td>B</td>
</tr>
<tr>
<td>B</td>
<td>3</td>
<td>B</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>C</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>D</td>
</tr>
<tr>
<td>E</td>
<td>5</td>
<td>C</td>
</tr>
<tr>
<td>F</td>
<td>2</td>
<td>C</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dst</th>
<th>Cst</th>
<th>Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2</td>
<td>A</td>
</tr>
<tr>
<td>B</td>
<td>4</td>
<td>F</td>
</tr>
<tr>
<td>C</td>
<td>4</td>
<td>F</td>
</tr>
<tr>
<td>D</td>
<td>5</td>
<td>F</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
<td>E</td>
</tr>
<tr>
<td>F</td>
<td>3</td>
<td>F</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dst</th>
<th>Cst</th>
<th>Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>5</td>
<td>B</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>B</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>C</td>
</tr>
<tr>
<td>D</td>
<td>2</td>
<td>C</td>
</tr>
<tr>
<td>E</td>
<td>3</td>
<td>E</td>
</tr>
<tr>
<td>F</td>
<td>0</td>
<td>F</td>
</tr>
</tbody>
</table>
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”

c(X,Y) change
Distance Vector: Link Cost Changes

Link cost changes:

- Good news travels fast
- Bad news travels slow - “count to infinity” problem!

![Diagram showing link cost changes over time]
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
Routing Information Protocol (RIP)

• Distance vector protocol
  – Nodes send distance vectors every 30 seconds
  – ... or, when an update causes a change in routing

• Link costs in RIP
  – All links have cost 1
  – Valid distances of 1 through 15
  – ... with 16 representing infinity
  – Small “infinity” → smaller “counting to infinity” problem

• RIP is limited to fairly small networks
  – E.g., used in some campus networks
Comparison of LS and DV Routing

Message complexity
- **LS:** with $n$ nodes, $E$ links, $O(nE)$ messages sent
- **DV:** exchange between neighbors only

Speed of Convergence
- **LS:** relatively fast
- **DV:** convergence time varies
  - May be routing loops
  - Count-to-infinity problem

Robustness: what happens if router malfunctions?

**LS:**
- Node can advertise incorrect \textit{link} cost
- Each node computes only its \textit{own} table

**DV:**
- DV node can advertise incorrect \textit{path} cost
- Each node’s table used by others (error propagates)
Similarities of LS and DV Routing

• Shortest-path routing
  – Metric-based, using link weights
  – Routers share a common view of how good a path is

• As such, commonly used inside an organization
  – RIP and OSPF are mostly used as intradomain protocols
  – E.g., Princeton uses RIP, and AT&T uses OSPF

• But the Internet is a “network of networks”
  – How to stitch the many networks together?
  – When networks may not have common goals
  – ... and may not want to share information
Hierarchical Routing

(a) Hierarchical routing diagram showing regions and nodes. The diagram illustrates the connectivity between different regions and nodes, with lines indicating the routes.

(b) Full table for 1A:

<table>
<thead>
<tr>
<th>Dest.</th>
<th>Line</th>
<th>Hops</th>
</tr>
</thead>
<tbody>
<tr>
<td>1A</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1B</td>
<td>1B</td>
<td>1</td>
</tr>
<tr>
<td>1C</td>
<td>1C</td>
<td>1</td>
</tr>
<tr>
<td>2A</td>
<td>1B</td>
<td>2</td>
</tr>
<tr>
<td>2B</td>
<td>1B</td>
<td>3</td>
</tr>
<tr>
<td>2C</td>
<td>1B</td>
<td>3</td>
</tr>
<tr>
<td>2D</td>
<td>1B</td>
<td>4</td>
</tr>
<tr>
<td>3A</td>
<td>1C</td>
<td>3</td>
</tr>
<tr>
<td>3B</td>
<td>1C</td>
<td>2</td>
</tr>
<tr>
<td>4A</td>
<td>1C</td>
<td>3</td>
</tr>
<tr>
<td>4B</td>
<td>1C</td>
<td>4</td>
</tr>
<tr>
<td>4C</td>
<td>1C</td>
<td>4</td>
</tr>
<tr>
<td>5A</td>
<td>1C</td>
<td>4</td>
</tr>
<tr>
<td>5B</td>
<td>1C</td>
<td>5</td>
</tr>
<tr>
<td>5C</td>
<td>1B</td>
<td>5</td>
</tr>
<tr>
<td>5D</td>
<td>1C</td>
<td>6</td>
</tr>
<tr>
<td>5E</td>
<td>1C</td>
<td>5</td>
</tr>
</tbody>
</table>

(c) Hierarchical table for 1A:

<table>
<thead>
<tr>
<th>Dest.</th>
<th>Line</th>
<th>Hops</th>
</tr>
</thead>
<tbody>
<tr>
<td>1A</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1B</td>
<td>1B</td>
<td>1</td>
</tr>
<tr>
<td>1C</td>
<td>1C</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1B</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1C</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>1C</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>1C</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>1C</td>
<td>4</td>
</tr>
</tbody>
</table>
Reverse path forwarding. (a) A subnet. (b) a Sink tree. (c) The tree built by reverse path forwarding.
(a) A network. (b) A spanning tree for the leftmost router. (c) A multicast tree for group 1. (d) A multicast tree for group 2.
Routing for Mobile Hosts

A WAN to which LANs, MANs, and wireless cells are attached.
Routing for Mobile Hosts (2)

1. Packet is sent to the mobile host's home address
2. Packet is tunneled to the foreign agent
3. Sender is given foreign agent's address
4. Subsequent packets are tunneled to the foreign agent
Routing in Ad Hoc Networks

Possibilities when the routers are mobile:

1. Military vehicles on battlefield.
   - No infrastructure.

2. A fleet of ships at sea.
   - All moving all the time

3. Emergency works at earthquake.
   - The infrastructure destroyed.

4. A gathering of people with notebook computers.
   - In an area lacking 802.11.
Route Discovery

- (a) Range of A's broadcast.
- (b) After B and D have received A's broadcast.
- (c) After C, F, and G have received A's broadcast.
- (d) After E, H, and I have received A's broadcast.

Shaded nodes are new recipients. Arrows show possible reverse routes.
Route Discovery (2)

<table>
<thead>
<tr>
<th>Source address</th>
<th>Request ID</th>
<th>Destination address</th>
<th>Source sequence #</th>
<th>Dest. sequence #</th>
<th>Hop count</th>
</tr>
</thead>
</table>

Format of a ROUTE REQUEST packet.
Route Discovery (3)

<table>
<thead>
<tr>
<th>Source address</th>
<th>Destination address</th>
<th>Destination sequence #</th>
<th>Hop count</th>
<th>Lifetime</th>
</tr>
</thead>
</table>

Format of a ROUTE REPLY packet.
Route Maintenance

(a) D's routing table before G goes down.
(b) The graph after G has gone down.
Node Lookup in Peer-to-Peer Networks

(a) A set of 32 node identifiers arranged in a circle. The shaded ones correspond to actual machines. The arcs show the fingers from nodes 1, 4, and 12. The labels on the arcs are the table indices.

(b) Examples of the finger tables.
Congestion Control Algorithms

- General Principles of Congestion Control
- Congestion Prevention Policies
- Congestion Control in Virtual-Circuit Subnets
- Congestion Control in Datagram Subnets
- Load Shedding
- Jitter Control
When too much traffic is offered, congestion sets in and performance degrades sharply.
General Principles of Congestion Control

1. Monitor the system.
   – detect when and where congestion occurs.
2. Pass information to where action can be taken.
3. Adjust system operation to correct the problem.
# Congestion Prevention Policies

<table>
<thead>
<tr>
<th>Layer</th>
<th>Policies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transport</td>
<td>• Retransmission policy</td>
</tr>
<tr>
<td></td>
<td>• Out-of-order caching policy</td>
</tr>
<tr>
<td></td>
<td>• Acknowledgement policy</td>
</tr>
<tr>
<td></td>
<td>• Flow control policy</td>
</tr>
<tr>
<td></td>
<td>• Timeout determination</td>
</tr>
<tr>
<td>Network</td>
<td>• Virtual circuits versus datagram inside the subnet</td>
</tr>
<tr>
<td></td>
<td>• Packet queueing and service policy</td>
</tr>
<tr>
<td></td>
<td>• Packet discard policy</td>
</tr>
<tr>
<td></td>
<td>• Routing algorithm</td>
</tr>
<tr>
<td></td>
<td>• Packet lifetime management</td>
</tr>
<tr>
<td>Data link</td>
<td>• Retransmission policy</td>
</tr>
<tr>
<td></td>
<td>• Out-of-order caching policy</td>
</tr>
<tr>
<td></td>
<td>• Acknowledgement policy</td>
</tr>
<tr>
<td></td>
<td>• Flow control policy</td>
</tr>
</tbody>
</table>
(a) A congested subnet. (b) A redrawn subnet, eliminates congestion and a virtual circuit from A to B.
Hop-by-Hop Choke Packets

(a) A choke packet that affects only the source.

(b) A choke packet that affects each hop it passes through.
Jitter Control

(a) High jitter.  

(b) Low jitter.
Quality of Service

- Requirements
- Techniques for Achieving Good Quality of Service
- Integrated Services
- Differentiated Services
- Label Switching and MPLS
## Requirements

How stringent the quality-of-service requirements are.

<table>
<thead>
<tr>
<th>Application</th>
<th>Reliability</th>
<th>Delay</th>
<th>Jitter</th>
<th>Bandwidth</th>
</tr>
</thead>
<tbody>
<tr>
<td>E-mail</td>
<td>High</td>
<td>Low</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td>File transfer</td>
<td>High</td>
<td>Low</td>
<td>Low</td>
<td>Medium</td>
</tr>
<tr>
<td>Web access</td>
<td>High</td>
<td>Medium</td>
<td>Low</td>
<td>Medium</td>
</tr>
<tr>
<td>Remote login</td>
<td>High</td>
<td>Medium</td>
<td>Medium</td>
<td>Low</td>
</tr>
<tr>
<td>Audio on demand</td>
<td>Low</td>
<td>Low</td>
<td>High</td>
<td>Medium</td>
</tr>
<tr>
<td>Video on demand</td>
<td>Low</td>
<td>Low</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>Telephony</td>
<td>Low</td>
<td>High</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>Videoconferencing</td>
<td>Low</td>
<td>High</td>
<td>High</td>
<td>High</td>
</tr>
</tbody>
</table>
Buffering

Smoothing the output stream by buffering packets.
The Leaky Bucket Algorithm

(a) A leaky bucket with water. (b) A leaky bucket with packets.
The Leaky Bucket Algorithm

(a) Input to a leaky bucket. (b) Output from a leaky bucket. Output from a token bucket with capacities of (c) 250 KB, (d) 500 KB, (e) 750 KB, (f) Output from a 500KB token bucket feeding a 10-MB/sec leaky bucket.
The Token Bucket Algorithm

(a) Before. (b) After.
Admission Control

An example of flow specification.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Token bucket rate</td>
<td>Bytes/sec</td>
</tr>
<tr>
<td>Token bucket size</td>
<td>Bytes</td>
</tr>
<tr>
<td>Peak data rate</td>
<td>Bytes/sec</td>
</tr>
<tr>
<td>Minimum packet size</td>
<td>Bytes</td>
</tr>
<tr>
<td>Maximum packet size</td>
<td>Bytes</td>
</tr>
</tbody>
</table>
Packet Scheduling

(a) A router with five packets queued for line O.
(b) Finishing times for the five packets.
RSVP-The ReSerVation Protocol

(a) A network,  (b) The multicast spanning tree for host 1.  (c) The multicast spanning tree for host 2.
(a) Host 3 requests a channel to host 1. (b) Host 3 then requests a second channel, to host 2. (c) Host 5 requests a channel to host 1.
Expedited Forwarding

Expedited packets experience a traffic-free network.
Assured Forwarding

A possible implementation of the data flow for assured forwarding.
Label Switching and MPLS

Transmitting a TCP segment using IP, MPLS, and PPP.
Internetworking

• How Networks Differ
• How Networks Can Be Connected
• Concatenated Virtual Circuits
• Connectionless Internetworking
• Tunneling
• Internetwork Routing
• Fragmentation
Connecting Networks

A collection of interconnected networks.
# How Networks Differ

Some of the many ways networks can differ.

<table>
<thead>
<tr>
<th>Item</th>
<th>Some Possibilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service offered</td>
<td>Connection oriented versus connectionless</td>
</tr>
<tr>
<td>Protocols</td>
<td>IP, IPX, SNA, ATM, MPLS, AppleTalk, etc.</td>
</tr>
<tr>
<td>Addressing</td>
<td>Flat (802) versus hierarchical (IP)</td>
</tr>
<tr>
<td>Multicasting</td>
<td>Present or absent (also broadcasting)</td>
</tr>
<tr>
<td>Packet size</td>
<td>Every network has its own maximum</td>
</tr>
<tr>
<td>Quality of service</td>
<td>Present or absent; many different kinds</td>
</tr>
<tr>
<td>Error handling</td>
<td>Reliable, ordered, and unordered delivery</td>
</tr>
<tr>
<td>Flow control</td>
<td>Sliding window, rate control, other, or none</td>
</tr>
<tr>
<td>Congestion control</td>
<td>Leaky bucket, token bucket, RED, choke packets, etc.</td>
</tr>
<tr>
<td>Security</td>
<td>Privacy rules, encryption, etc.</td>
</tr>
<tr>
<td>Parameters</td>
<td>Different timeouts, flow specifications, etc.</td>
</tr>
<tr>
<td>Accounting</td>
<td>By connect time, by packet, by byte, or not at all</td>
</tr>
</tbody>
</table>
How Networks Can Be Connected

(a) Two Ethernets connected by a switch.
(b) Two Ethernets connected by routers.
Concatenated Virtual Circuits

Internetworking using concatenated virtual circuits.
Connectionless Internetworking

Packets travel individually and can take different routes

A connectionless internet.
Tunneling a packet from Paris to London.
Tunneling (2)

Tunneling a car from France to England.
Internetwork Routing

(a) An internetwork. (b) A graph of the internetwork.
Fragmentation

(a) Transparent fragmentation.  (b) Nontransparent fragmentation.
Fragmentation when the elementary data size is 1 byte.

(a) Original packet, containing 10 data bytes.
(b) Fragments after passing through a network with maximum packet size of 8 payload bytes plus header.
(c) Fragments after passing through a size 5 gateway.
The Network Layer in the Internet

- The IP Protocol
- IP Addresses
- Internet Control Protocols
- OSPF – The Interior Gateway Routing Protocol
- BGP – The Exterior Gateway Routing Protocol
- Internet Multicasting
- Mobile IP
- IPv6
Design Principles for Internet

1. Make sure it works.
2. Keep it simple.
3. Make clear choices.
4. Exploit modularity.
5. Expect heterogeneity.
6. Avoid static options and parameters.
7. Look for a good design; it need not be perfect.
8. Be strict when sending and tolerant when receiving.
9. Think about scalability.
The Internet is an interconnected collection of many networks.
NAT – Network Address Translation

Diagram showing the placement and operation of a NAT box.
## Internet Control Message Protocol

<table>
<thead>
<tr>
<th>Message type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Destination unreachable</td>
<td>Packet could not be delivered</td>
</tr>
<tr>
<td>Time exceeded</td>
<td>Time to live field hit 0</td>
</tr>
<tr>
<td>Parameter problem</td>
<td>Invalid header field</td>
</tr>
<tr>
<td>Source quench</td>
<td>Choke packet</td>
</tr>
<tr>
<td>Redirect</td>
<td>Teach a router about geography</td>
</tr>
<tr>
<td>Echo request</td>
<td>Ask a machine if it is alive</td>
</tr>
<tr>
<td>Echo reply</td>
<td>Yes, I am alive</td>
</tr>
<tr>
<td>Timestamp request</td>
<td>Same as Echo request, but with timestamp</td>
</tr>
<tr>
<td>Timestamp reply</td>
<td>Same as Echo reply, but with timestamp</td>
</tr>
</tbody>
</table>
ARP– The Address Resolution Protocol

- Three interconnected /24 networks: two Ethernets and an FDDI ring.
(a) An autonomous system. (b) A graph representation of (a).
The relation between ASes, backbones, and areas in OSPF.
# OSPF (3)

The five types of OSPF messages.

<table>
<thead>
<tr>
<th>Message type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hello</td>
<td>Used to discover who the neighbors are</td>
</tr>
<tr>
<td>Link state update</td>
<td>Provides the sender’s costs to its neighbors</td>
</tr>
<tr>
<td>Link state ack</td>
<td>Acknowledges link state update</td>
</tr>
<tr>
<td>Database description</td>
<td>Announces which updates the sender has</td>
</tr>
<tr>
<td>Link state request</td>
<td>Requests information from the partner</td>
</tr>
</tbody>
</table>
BGP – The Exterior Gateway Routing Protocol

(a) A set of BGP routers.

(b) Information sent to F.

Information F receives from its neighbors about D:
- From B: "I use BCD"
- From G: "I use GCD"
- From I: "I use IFGCD"
- From E: "I use EFGCD"
The Main IPv6 Header

The IPv6 fixed header (required).
# Extension Headers

IPv6 extension headers.

<table>
<thead>
<tr>
<th>Extension header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hop-by-hop options</td>
<td>Miscellaneous information for routers</td>
</tr>
<tr>
<td>Destination options</td>
<td>Additional information for the destination</td>
</tr>
<tr>
<td>Routing</td>
<td>Loose list of routers to visit</td>
</tr>
<tr>
<td>Fragmentation</td>
<td>Management of datagram fragments</td>
</tr>
<tr>
<td>Authentication</td>
<td>Verification of the sender’s identity</td>
</tr>
<tr>
<td>Encrypted security payload</td>
<td>Information about the encrypted contents</td>
</tr>
</tbody>
</table>
Extension Headers (2)

<table>
<thead>
<tr>
<th>Next header</th>
<th>0</th>
<th>194</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jumbo payload length</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The hop-by-hop extension header for large datagrams (jumbograms).
Extension Headers (3)

<table>
<thead>
<tr>
<th>Next header</th>
<th>Header extension length</th>
<th>Routing type</th>
<th>Segments left</th>
</tr>
</thead>
</table>

Type-specific data

The extension header for routing.