CMPE 150 -- Introduction to Computer Networks

- Instructor: Patrick Mantey
  mantey@soe.ucsc.edu
  http://www.soe.ucsc.edu/~mantey/
- Office: Engr. 2 Room 595J
- Office hours: Tues 3-5 PM, Mon 5-6 PM*
- TA: Anselm Kia  akia@soe.ucsc.edu
- Web site:  http://www.soe.ucsc.edu/classes/cmpe150/Winter09/
- Text: Tannenbaum: Computer Networks
  (4th edition – available in bookstore, etc. )
# Syllabus

<table>
<thead>
<tr>
<th>Lecture</th>
<th>Date</th>
<th>Topic</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6-Jan</td>
<td>Intro / Overview</td>
<td>Chapter 1</td>
</tr>
<tr>
<td>2</td>
<td>8-Jan</td>
<td>Internet Architecture</td>
<td>Chapter 1, 2</td>
</tr>
<tr>
<td>3</td>
<td>13-Jan</td>
<td>Physical Layer</td>
<td>Chapter 2</td>
</tr>
<tr>
<td>4</td>
<td>15-Jan</td>
<td>Physical Layer</td>
<td>Chapter 2</td>
</tr>
<tr>
<td>5</td>
<td>20-Jan</td>
<td>Link Layer</td>
<td>Chapter 3</td>
</tr>
<tr>
<td>6</td>
<td>22-Jan</td>
<td>Link Layer / MAC</td>
<td>Chapter 3, 4</td>
</tr>
<tr>
<td>7</td>
<td>27-Jan</td>
<td>MAC Sublayer</td>
<td>Chapter 4</td>
</tr>
<tr>
<td>8</td>
<td>29-Jan</td>
<td>Wireless</td>
<td>Chapter 4</td>
</tr>
<tr>
<td>9</td>
<td>3-Feb</td>
<td>Network Layer</td>
<td>Chapter 5</td>
</tr>
<tr>
<td>10</td>
<td>5-Feb</td>
<td>Midterm Exam 1</td>
<td>Chapter 5</td>
</tr>
<tr>
<td>11</td>
<td>10-Feb</td>
<td>IP</td>
<td>Chapter 5</td>
</tr>
<tr>
<td>12</td>
<td>12-Feb</td>
<td>IP</td>
<td>Chapter 5</td>
</tr>
<tr>
<td>13</td>
<td>17-Feb</td>
<td>IP Routing</td>
<td>Chapter 5</td>
</tr>
<tr>
<td>14</td>
<td>19-Feb</td>
<td>IP QOS, Congestion</td>
<td>Chapter 5</td>
</tr>
<tr>
<td>15</td>
<td>24-Feb</td>
<td>Transport Layer</td>
<td>Chapter 6</td>
</tr>
<tr>
<td>16</td>
<td>26-Feb</td>
<td>Midterm Exam 2</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>3-Mar</td>
<td>Transport Layer / QOS</td>
<td>Chapter 6</td>
</tr>
<tr>
<td>18</td>
<td>6-Mar</td>
<td>Multimedia</td>
<td>Chapter 6</td>
</tr>
<tr>
<td>19</td>
<td>10-Mar</td>
<td>Applications / Summary</td>
<td>Chapter 7</td>
</tr>
<tr>
<td>20</td>
<td>13-Mar</td>
<td>Buffer / Review</td>
<td>Chapter 7</td>
</tr>
<tr>
<td><strong>Final Exam</strong></td>
<td>18-Mar</td>
<td>12-3 PM</td>
<td></td>
</tr>
</tbody>
</table>
Today’s Agenda

- Routing (continued)
  - Distance Vector Protocol
  - Link State Routing
- Network Layer
  - Hierarchical Routing
  - Broadcast Routing
  - Spanning Tree Routing
  - Congestion Control
  - Quality of Service
  - Internetworking
Reading Assignment

- Today: Chapter 5, sections 5.2.4-8 (continued), 5.3, 5.4.1-5.4.3

- Tuesday: Tannenbaum, Chapter 5, section 5.5, 5.6: *Internet Protocol*
<table>
<thead>
<tr>
<th>Level 1</th>
<th>Physical Layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level 2</td>
<td>(Data) Link Layer / MAC sub-layer (a.k.a. Network Interface or Network Access Layer)</td>
</tr>
<tr>
<td>Level 3</td>
<td>Network Layer (a.k.a. Internet) <em>(IP)</em></td>
</tr>
<tr>
<td>Level 4</td>
<td>Transport Layer (a.k.a. Host-to-Host) <em>(TCP, UDP, ARP, ICMP, etc.)</em></td>
</tr>
<tr>
<td>Level 5</td>
<td>Application Layer (rlogin, ftp, SMTP, POP3, IMAP, HTTP..)</td>
</tr>
</tbody>
</table>
Distance-Vector protocol

Initialize routing table with local links
Flood routing table to all routers

Do

Compute local routing table from graph
Wait for update or link cost change or timer
Update network graph

If link cost change
  Flood updated link to all routers

Else if timer expired
  Flood routing table to all routers

Forever
- (a) A subnet. (b) Input from A, I, H, K, and the new routing table for J.
Distance Vector Routing 1

- Each router keeps routing table (or routing vector) giving best known distance to each destination and the corresponding outgoing interface.
- Routing tables are updated by exchanging routing information with neighbors.
- Implements distributed version of Bellman-Ford
  Aka Ford-Fulkerson
- Timer-based refresh vs neighbor tables
Routing table at each router:
- One entry per participating router.
- Each entry contains outgoing interface and distance to corresponding destination.
- Metric: number of hops, delay, queue length.
- Each router knows distance to its neighbors.

Old ARPANET algorithm: DV where cost metric is outgoing link queue length. Also used in RIP
Routing Updates

- Every T interval, routers exchange routing updates.
- Routing update from router X consists of a vector with all destinations and the corresponding distance from X to them.
- When router Y receives an update from X, it can estimate its distance to router Z through X as \( D_{yz} = D_{yx} + D_{xz} \).
- Router Y receives update from all its neighbors; discards its RT and builds a new one.
Distance Vector: Example

Node Distance Next

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>0</th>
<th>-</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

T=T₀

<table>
<thead>
<tr>
<th></th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

T=T₁

T=T₂
Distance Vector: Example

Node Distance Next

<table>
<thead>
<tr>
<th>Node</th>
<th>Distance</th>
<th>Next</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>8</td>
<td>3</td>
</tr>
</tbody>
</table>

T = T₀

<table>
<thead>
<tr>
<th>Node</th>
<th>Distance</th>
<th>Next</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

T = T₁

T = T₂
Problems

1. Routing loops.
2. Slow convergence.
3. Counting to infinity.
Bellman-Ford overview

- Network represented by graph: $G(V,E)$
  - $V$ contains vertices $i, j, ...$
  - $E$ contains edges $(i,j), ...$

- Algorithm data structures
  - “$s$” - source vertex
  - “$d_{ij}$” - cost of the edge $(i,j)$; $d_{ij} = \infty$ if $(i,j) \notin E$
  - “$D^h_i$” - cost of the shortest path with $h$ hops from $s$ to $i$
Bellman-Ford algorithm

\[ D^h_s = 0 \text{ for all } h \]
\[ D^0_i = \infty \text{ for all } \{i \in V \mid i \neq s\} \]
\[ h = 0 \]

**Do**

\[ h = h + 1 \]
\[ D^h_j = \text{Min } \{D^{h+1}_i + d_{ij} \text{ for all } j \in V \mid j \neq s\} \]

**Until** \( (D^h_i = D^{h-1}_i \text{ for all } i \in V) \)
Bellman-Ford illustrated
Count-to-Infinity 1

Initially, A down:
A comes up:

Good news propagates faster.
Count-to-Infinity 2

Initially, all up:

A goes down:

1  2  3  4  
3  2  3  4  (after 1 exchange)
3  4  3  4  (after 2 exchanges)
5  4  5  4  (after 3 exchanges)
5  6  5  6  (after 4 exchanges)
7  6  7  6  (after 5 exchanges)
7  8  7  8  (after 6 exchanges)

....

infinity

But, bad news propagate slower!
Count-to-Infinity 3

- Gradually routers work their way up to infinity.
- Number of exchanges depends on how large is infinity.
- To reduce number of exchanges, if metric is number of hops, \( \text{infinity} = \text{maximum path} + 1 \).
Solution

- Routing loops:
  - Path vector: record actual path used in the DV.
  - Previous hop tracing: records preceding router.

- Count-to-infinity:
  - Split horizon: router doesn’t report true route to next hop of the previous route
  - Poison reverse: router “infinity” to next hop of previous route
  - Split horizon with poison reverse prevent loops involving two routers
  - Triggered updates: if the metric for a route is changed, send an update immediately
  - In the absence of other updates, this solves the problem... in reality, other updates will “keep hope alive”
  - RFC1058 is a good reference for these issues
Split Horizon

- Tries to make bad news spread faster.
- A node reports infinity as distance to node X on link packets to X are sent on.
- Example, in the first exchange, C tells D its distance to A but tells B its distance to A is infinity.
  - So B discovers its link to A is down and C’s distance to A is infinity; so it sets its distance to A to infinity.
Link-State protocol

- Initialize network graph with local links
- Flood local links to all routers

**Do**
- Compute local routing table from graph
- Wait for update or link cost change
- Update network graph

**If** link cost change
- Flood updated link to all routers

**Forever**
Link State Routing

- Link State routing used in the ARPAnet starting in 1979
- Used by the Internet’s OSPF (Open Shortest Path First)
Link State Routing 2

- Link state routing is based on:
  - Discover your neighbors and measure the communication cost to them.
  - Send updates about your neighbors to all other routers.
  - Compute shortest path to every other router.
Finding Neighbors

- When router is booted, its first task is to find who its neighbors are.
- Special single-hop “hello” packets.
- Cost metric:
  - Number of hops: in this case, always 1.
  - Delay: “echo” packets and measure RTT/2.
  - Load?
Link State Routing

- Each router must do the following:
  - Discover its neighbors, learn their network address.
  - Measure the delay or cost to each of its neighbors.
  - Construct a packet telling all it has just learned.
  - Send this packet to all other routers.
  - Compute the shortest path to every other router.
Generating Link State Updates

- Link state packets (LSP).
  - Sender identity.
  - Sequence number.
  - TTL.
  - List of (neighbor, cost).
- When to send updates?
  - Proactive: periodic updates; how often?
  - Reactive: whenever some significant event is detected, e.g., link goes down.
Processing Updates

- When LSP (Link State Packet) is received:
  - Check sequence number.
  - If higher than current sequence number, keep it and flood it; otherwise, discard it.
  - Periodically decrement TTL.
    - When TTL=0, purge LSP.
Computing Routes

- Routers have global view of network.
  - They receive updates from all other routers with their cost to their neighbors.
  - Build network graph.

- Use Dijkstra’s shortest-path algorithm (locally) to compute shortest paths to all other nodes.
Learning about the Neighbors

- (a) Nine routers and a LAN.
- (b) A graph model of (a).
A subnet in which the East and West parts are connected by two lines.
(a) A subnet. (b) The link state packets for this subnet.
Distributing the Link State Packets

<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></td>
<td></td>
<td></td>
<td>A</td>
<td>C</td>
<td>F</td>
</tr>
<tr>
<td>A</td>
<td>21</td>
<td>60</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>F</td>
<td>21</td>
<td>60</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>E</td>
<td>21</td>
<td>59</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>20</td>
<td>60</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>21</td>
<td>59</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

The packet buffer for router B in the previous slide (Fig. 5-13).
DV versus LS

- **DV:**
  - Tell neighbors what you know about everybody.
  - Choose best route given neighbor’s routing tables.
  - Distributed computation.
  - Topology information distributed only where needed

- **LS:**
  - Tell everyone what you know about neighbors.
  - Every node has global view.
  - Compute their own routes.
  - Topology information distributed to everyone
Hierarchical Routing

For scalability:

- As network grows, so does RT size, routing update generation, processing, and propagation overhead, and route computation time and resources.

Divide network into *routing regions*.

- Routers within region know how to route packets to all destinations within region.
- But don’t know how to route within other regions.
- “Border” routers: route within regions.
Hierarchical Routing Example

<table>
<thead>
<tr>
<th>Dest.</th>
<th>Next</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>2B</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>
Hierarchical Routing Example

<table>
<thead>
<tr>
<th>Dest.</th>
<th>Next</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>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>
Hierarchical Routing

(a) Hierarchical routing diagram

(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>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>
Hierarchical Routing

- Optimal paths are not guaranteed.
  - Example: 1A->5C should be via 2 and not 3.

- How many hierarchical levels?
  - Example: 720 routers.
    - 1 level: each router needs 720 RT entries.
    - 2 levels: 24 regions of 30 routers: each router’s RT has 30+23 entries.
    - 3 levels: 8 clusters of 9 regions with 10 routers: each router’s RT 10+8+7.
Many-to-Many Routing

- Support many-to-many communication.
- Example applications: multi-point data distribution, multi-party teleconferencing.
Many-to-Many Routing

- Broadcasting
- Flooding
- Multidestination routing
- Spanning tree
- Reverse path forwarding
Broadcasting

- **Simplistic approach:** send separate packet to each destination.
  - Simple but expensive.
  - Source needs to know about all destinations.

- **Flooding:**
  - May generate too many duplicates (depending on node connectivity).
Multidestination Routing

- Packet contains list of destinations.
- Router checks destinations and determines on which interfaces it will forward packet.
  - Router generates new copy of packet for each output line and includes in packet only the appropriate set of destinations.
  - Eventually, packets will only carry 1 destination.
Spanning Tree Routing

- Use spanning tree (sink tree) rooted at broadcast initiator.
- No need for destination list.
- Each on spanning tree forwards packets on all lines on the spanning tree (except the one the packet arrived on).
- Efficient but needs to generate the spanning tree and routers must have that information.
Reverse Path Forwarding

- Routers don’t have to know spanning tree.
- Router checks whether broadcast packet arrived on interface used to send packets to source of broadcast.
  - If so, it’s likely that it followed best route and thus not a duplicate; router forwards packet on all lines.
  - If not, packet discarded as likely duplicate.
Broadcast Routing

Reverse path forwarding. (a) A subnet. (b) a Sink tree. (c) The tree built by reverse path forwarding.
Multicast Routing

(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.
Multicasting

• Special form of broadcasting:
  – Instead of sending messages to all nodes, send messages to a group of nodes.

• Multicast group management:
  – Creating, deleting, joining, leaving group.
  – Group management protocols communicate group membership to appropriate routers.
Multicast Routing

- Each router computes spanning tree covering all other participating routers.
  - Tree is pruned by removing that do not contain any group members.
Shared Tree Multicasting

• Source-rooted tree approaches don’t scale well!
  – 1 tree per source, per group!
  – Routers must keep state for $m \times n$ trees, where $m$ is number of sources in a group and $n$ is number of groups.

• Core-based trees: single tree per group.
  – Host unicast message to core, where message is multicast along shared tree.
  – Routes may not be optimal for all sources.
  – State/storage savings in routers.
Other Simulations / Animations

AN ANIMATED SIMULATOR FOR PACKET SNIFER

Xiaohong Yuan, Percy Vega, Jinsheng Xu, Huiming Yu, and Stephen Providence

Department of Computer Science, North Carolina A&T State University, 1601 East Market St., Greensboro, NC 27411, Email: {xhyuan, jx, ucsmyu, svp} @ncat.edu; Phone: (336)3347245. *Everett Consulting Corp., 320 McKinley St. #10, Hollywood, FL 33019, Email: percyvega@gmail.com Phone: (954)7325664

Congestion Control

Ideal network behavior:
When too much traffic is offered, congestion sets in and performance degrades sharply.
# Congestion and Policies

<table>
<thead>
<tr>
<th>Layer</th>
<th>Policies</th>
</tr>
</thead>
</table>
| Transport| • Retransmission policy  
• Out-of-order caching policy  
• Acknowledgement policy  
• Flow control policy  
• Timeout determination |
| Network  | • Virtual circuits versus datagram inside the subnet  
• Packet queueing and service policy  
• Packet discard policy  
• Routing algorithm  
• Packet lifetime management |
| Data link| • Retransmission policy  
• Out-of-order caching policy  
• Acknowledgement policy  
• Flow control policy |

• Policies that affect congestion.
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.
Network Congestion

What is network congestion?
- Too many packets in the network.
- Router queues are always full.
  - Routers start dropping packets.
- Congestion can fuel itself.
  - Packet drops lead to retransmissions.
  - More traffic!
- May result in congestion collapse!
  - Close to 0 throughput!
Infinite-Buffer Routers

Intuition says add more memory to routers and that’ll avoid congestion.

- Nagle (1987) showed that infinite buffers actually make congestion worse.
- More packets enqueued for long time; they time out and are retransmitted; but still transmitted by router.
- Therefore, more traffic.
Causes of Congestion

- Mismatch in capacity among different parts of the system.
  - Mismatch in link speeds.
  - Mismatch in router processing capability.
    - Table lookup and update.
    - Queue management.
- Congestion in one point of network tends to propagate backwards toward sender.
Congestion versus Flow Control

- Congestion control tries to ensure the network is able to carry offered traffic.
  - Involves hosts and intermediate routers.

- Flow control ensures that the communication end-points are able to keep up with one another.
  - Involves only the end-points.
Congestion and Flow Control

- Often mixed because tend to use same feedback mechanisms.

- Example: “slow down” message received at host may be caused by receiver not being able to keep up with sender host or by network not being able to handle additional traffic.
Congestion Control Principles

From control theory point of view:
- Open and closed loop solutions.

Open loop solutions:
- Avoidance approach.
  - Tries to make sure problem doesn’t happen.
  - Doesn’t take current network state into account.

Closed loop solutions:
- Feedback loop.
Closed Loop Solutions

3 components:

- Monitoring.
- Feedback generation.
- Operation adjustment.

Monitoring metrics:

- Packet loss.
- Average queue length.
- Number of retransmitted packets.
- Average packet delay.
Feedback

- Send information about the problem once it’s detected.
  - Router that detects problem sends packet to traffic source(s).
  - Special-purpose bit in every packet that router sets when it detects congestion above certain level to warn neighbors.
  - Special probe messages to detect congested areas so they can be avoided.

- Stability: avoid oscillations.
Congestion Control Taxonomy

- Open loop algorithms:
  - Act at source.
  - Act at destination.

- Closed loop algorithms:
  - Explicit feedback.
  - Implicit feedback.
Congestion Control in Virtual-Circuit Subnets

- (a) A congested subnet. (b) A redrawn subnet, eliminates congestion and a virtual circuit from A to B.
Hop-by-Hop Choke Packets (for Datagrams)

- (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.

- **Router sends packets furthest behind schedule**
Quality of Service

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

<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.
Open Loop Approaches

- Traffic Shaping
  - Avoid traffic burstiness by forcing packets to be transmitted at more predictable rate.
  - Used in ATM networks.
  - Regulates average transmission rate.
  - In contrast to sliding window protocols which regulate amount of data in transit.
  - Service agreement between user and carrier.
    - Important to real-time traffic such as audio, video.
1. No matter the rate water enters bucket, the outflow is constant.
2. Once bucket full, water spills and lost.

Unregulated flow

Network interface

Regulated flow

Network
Leaky Bucket 2

- Equivalent to a single-server queuing system with constant service time.
- Same size packets (e.g., ATM cells): use packets as unit.
- Variable-sized packets: use number of bytes per clock tick.
The Token Bucket Algorithm

- (a) Before.
- (b) After.
Token Bucket

- More flexible.
  - Allows packets to go out as fast as they come in provided there are enough tokens.
  - Leaky bucket holds tokens generated every T sec.
  - Allows hosts to save up for later.
    - Hosts can accumulate up to \( n \) tokens, when \( n \) is bucket size.
Leaky and Token Bucket

- Token bucket throws away tokens but never packets.
- Can be used between host and network and between routers.
- Token bucket can still produce bursts.
  - Insert leaky bucket after token bucket.
Leaky and Token Bucket Algorithms

(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.
Flow Specifications

- Way for user/application to specify traffic patterns and desired quality of service.
  - Before connection established or data is sent, source provides flow spec to network.
  - Network can accept, reject, or counter-offer.
  - Traffic spec: maximum packet size, maximum transmission rate.
  - Service desired: maximum acceptable loss rate, maximum delay and delay variation.
Admission Control

<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>

An example of flow specification.
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.
RSVP-The ReSerVation Protocol (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 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

Diagram:
- Mainframe
- SNA network
- ATM network
- FDDI ring
- Router
- Switch
- Ethernet
- Notebook computer
- 802.11
- Connection to Internet
How Networks 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>

Some of the many ways networks can differ.
How Networks Can Be Connected

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

The diagram illustrates the concatenation of virtual circuits in an internetworking context. It shows the connection between X.25, OSI, SNA, and ATM layers through multiprotocol routers. The end-to-end concatenated virtual circuits are highlighted, emphasizing the integrated communication pathways.
Connectionless Internetworking

Packets travel individually and can take different routes

Router

Multiprotocol router

Host
Tunneling

Tunneling a packet from Paris to London.
Tunneling a car from France to England.
Internetwork Routing 1

• 2-level hierarchy:
  – Routing within each network: interior gateway protocol.
  – Routing between networks: exterior gateway protocol.

• Within each network, different routing algorithms can be used.

• Each network is autonomously managed and independent of others: autonomous system (AS).
Internetwork Routing 2

- Typically, packet starts in its LAN. Gateway receives it (broadcast on LAN to “unknown” destination).
- Gateway sends packet to gateway on the destination network using its routing table. If it can use the packet’s native protocol, sends packet directly. Otherwise, tunnels it.
Internetwork Routing

(a) An internetwork.  (b) A graph of the internetwork.
Fragmentation 1

- Network-specific maximum packet size.
  - Width of TDM slot.
  - OS buffer limitations.
  - Protocol (number of bits in packet length field).

- Maximum payloads range from 48 bytes (ATM cells) to 64Kbytes (IP packets).
What happens when large packet wants to travel through network with smaller maximum packet size? *Fragmentation.*

Gateways break packets into *fragments,* each sent as separate packet.

Gateway on the other side have to reassemble fragments into original packet.

2 kinds of fragmentation: transparent and non-transparent.
(a) Transparent fragmentation.

(b) Nontransparent fragmentation.
Fragmentation (2)

- 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.
Transparent Fragmentation

- Small-packet network transparent to other subsequent networks.
- Fragments of a packet addressed to the same exit gateway, where packet is reassembled.
  - OK for concatenated VC internetworking.
- Subsequent networks are not aware fragmentation occurred.
- ATM networks (through special hardware) provide transparent fragmentation: segmentation.
Problems with Transparent Fragmentation

- Exit gateway must know when it received all the pieces.
  - Fragment counter or “end of packet” bit.
- Some performance penalty but requiring all fragments to go through same gateway.
- May have to repeatedly fragment and reassemble through series of small-packet networks.
Non-Transparent Fragmentation

- Only reassemble at destination host.
  - Each fragment becomes a separate packet.
  - Thus routed independently.

- Problems:
  - Hosts must reassemble.
  - Every fragment must carry header until it reaches destination host.
Fragments must be numbered so that original data stream can be reconstructed.

Tree-structured numbering scheme:
- Packet 0 generates fragments 0.0, 0.1, 0.2, ...
- If these fragments need to be fragmented later on, then 0.0.0, 0.0.1, ..., 0.1.0, 0.1.1, ...
- But, too much overhead in terms of number of fields needed
Keeping Track of Fragments 2

- Another way is to define elementary fragment size that can pass through every network.

- When packet fragmented, all pieces equal to elementary fragment size, except last one (may be smaller).

- Packet may contain several fragments.
Keeping Track of Fragments 3

Header contains packet number, number of first fragment in the packet, and last-fragment bit.

(a) Original packet with 10 data bytes.

(b) Fragments after passing through network with maximum packet size = 8 bytes.