Today

• Types of switching
  – Datagram
  – Virtual circuit
  – Source routing

• Bridges and LAN switches
Packet switching

- Problem: single link networks have limited scale
  - Ethernet < 1024 hosts, 2500 meters
  - Wireless limited by radio ranges
  - Point-to-point links connect only two nodes

- A **packet switch** is a device with several inputs and outputs leading to and from the nodes that the switch interconnects
  - Hosts communicate without being directly connected
A star topology

- A switch has a limited number of input and output ports
- Switches can be connected to each other to build larger networks
- Adding a new host may not reduce the performance for other hosts
  - Not true for shared media networks
  - Why?
Switching technologies

- **Switching / forwarding**: to receive incoming packets on one of its links and to transmit them on some other link.

- **Problem**: how does a switch decide on which output port to place each packet?

- **Solution**: looks at the packet header and makes a decision
  - Connectionless: datagram
  - Connection oriented: virtual circuit
  - Source routing
Challenges

• **Contention**
  – Input rate exceeds output rate
    • Multiple input ports may send to the same output port
  – Switches queue packets until contention disappears

• **Congestion**
  – When a switch runs out of buffer, it discards packets.
  – Too frequent packet loss is said to be congested
Datagram

• Every packet contains the destination address
  – A global unique identifier
  – Ethernet has 48-bit addresses

• A switch maintains a forwarding table that maps a packet to an output port
Switch 2’s forwarding table

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>3</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>3</td>
</tr>
<tr>
<td>D</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td></td>
</tr>
<tr>
<td>G</td>
<td></td>
</tr>
<tr>
<td>H</td>
<td></td>
</tr>
</tbody>
</table>

Q: how does a switch compute the table?
Features of datagram switching

- Connectionless
- Unknown network state
- Independent forwarding
- Robust to failures
  - Switches can re-compute forwarding tables
Virtual circuit switching

• Connection oriented
  – Set up a virtual circuit
  – Data transfer

• Connection setup phase
  – Set up connection state
  – A virtual circuit identifier, an incoming interface, an outgoing interface, and an outgoing virtual circuit identifier
## Virtual circuit table (switch1)

<table>
<thead>
<tr>
<th>Incoming interface</th>
<th>Incoming VCI</th>
<th>Outgoing interface</th>
<th>Outgoing VCI</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>5</td>
<td>1</td>
<td>11</td>
</tr>
</tbody>
</table>
Virtual circuit switching

- Algorithm:
  - If a packet arrives on the matching incoming port with the matching incoming VCI, it will be sent to the corresponding outgoing port with the corresponding VCI

- VCIs are link-local
How to setup connection state

• Administrator configured
  – Permanent virtual circuit (PVC)
  – Admin manually sets up VC tables
  – Does not suit large networks

• Signaling
  – A host sends messages to dynamically setup or tear down a VC
VC setup protocol

• A host A sends a setup message to first hop switch, including the final destination address
  – Similar to a datagram packet

• The switch picks an unused VCI to identify the incoming connection, and fills part of the VC table
  – Why not let the host pick it?

• Every switch repeats the process until the packet reaches the destination B

• The destination B sends an ack to inform its upstream switch the VCI for the connection
- After setup, A sends to B
- A tears down after done
Characteristics of VC switching

- Connection setup wait

+ Data packets contain a small VCI, not the full destination addresses

- One switch failure tears down the entire connection

- Connection sets up require routing algorithms
  – Setup packet is forwarded using a datagram algorithm
VC allows resource reservation

- Buffers can be allocated during the setup phase to avoid congestion
- An example (X.25)
  - Buffers allocated during connection setup
  - Sliding window is run between pairs of nodes (hop-by-hop flow control)
  - Circuit is rejected if no more buffer
Quality of service (QoS)

• Connectionless network is difficult to allocate resources
  – Switches send packets independently
  – How to associate one packet with other packets?

• Virtual circuit can be used to provide different QoS
  – Allocate a fraction of link bandwidth to each circuit
Asynchronous Transfer Mode

• ATM Cells: fixed-size packets
  – 5 bytes header
  – 48 bytes payload
• If payload smaller than 48B, uses padding
• If greater than 48B, breaks it
Why small, fixed-length packets?

- Cons: maximum efficiency $\frac{48}{53}=90.6\%$

- Pros:
  - Suitable for high-speed hardware implementation
  - Many switching elements doing the same thing in parallel
  - Reducing priority packet latency
    - Good for QoS
  - Reducing transmission latency
Switching and Forwarding

- **ATM**
  - **User-Network Interface (UNI)**
    - Host-to-switch format
    - GFC: Generic Flow Control
    - VCI: Virtual Circuit Identifier
    - Type: management, congestion control
    - CLP: Cell Loss Priority
    - HEC: Header Error Check (CRC-8)

  ![ATM Frame Format Diagram]

- **Network-Network Interface (NNI)**
  - Switch-to-switch format
  - GFC becomes part of VPI field
Virtual paths

- 24-bit virtual circuit identifiers (VCIs)
- Two-levels of hierarchy
  - 8-bit virtual path, 16-bit VCI
  - Virtual paths shared by multiple connections
Why 48 bytes

- It’s from the telephone technology
- Thought data would be mostly voice
- A compromise
  - US wanted 64 bytes for efficiency
  - Europe wanted 32 bytes for simplifying echo cancellation
  - $(64 + 32) / 2 = 48$ bytes
- Popular in the late 80s and early 90s due to its high speed
  - Major telecoms supported it
- Popularity faded. IP/Ethernet ruled
  - IP over ATM
  - DSL over ATM: DSL modem takes Ethernet frames and chop them into cells
Switching technologies

• Connectionless: datagram

• Connection oriented: virtual circuit
  – An example of VC switching: ATM

• Source routing
Source routing

- Source host provides all the information for packets to travel across the network
  - Packets carry output port numbers
  - Packets carry switch addresses
  - Variable header length
Handling source routing headers

a. Rotation
b. Stripping
   – No return path!
c. Pointer
Loose or strict source routing

• **Strict**
  – Must visit every node on the path

• **Loose**
  – Waypoints rather than the complete route
Today

• Types of switching
  – Datagram
  – Virtual circuit
  – Source routing

• Bridges and LAN switches
Ethernet Bridges

• An **Ethernet Bridge** is a packet switching device that connects multiple Ethernet segments
  – Bridge is a historic name
  – Newer devices are also called Local Area Network (LAN) switches

• An **extended LAN**
  – Ethernet LANs connected by bridges
Bridges versus Repeaters

• Why not a repeater?
  – No more than four repeaters between two hosts
  – Can’t span longer than 2500m

• An **Ethernet bridge**
  – Buffers of frames prevents collisions.
  – Each port is isolated and builds its own collision domain
Bridges and LAN Switches

- Consider the following figure
  - When a frame from host A that is addressed to host B arrives on port 1, there is no need for the bridge to forward the frame out over port 2.
  - How does a bridge come to learn on which port the various hosts reside?
Bridges and LAN Switches

- Solution
  - Download a table into the bridge
    - Who does the download?
      - Human
        - Too much work for maintenance

<table>
<thead>
<tr>
<th>Host</th>
<th>Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
</tr>
<tr>
<td>X</td>
<td>2</td>
</tr>
<tr>
<td>Y</td>
<td>2</td>
</tr>
<tr>
<td>Z</td>
<td>2</td>
</tr>
</tbody>
</table>
Solution 2: Learning Bridges

• Overall design goal: **complete transparency**
  • “Plug-and-play”

• Three parts to learning bridges:
  • (1) Forwarding of Frames
  • (2) Learning of Addresses
  • (3) Spanning Tree Algorithm
(1) Frame Forwarding

• Each bridge maintains a **forwarding table** with entries

  `< MAC address, port, age>`

  **MAC address**: host or group address

  **port**: outgoing port number of bridge

  **age**: aging time of entry

with interpretation:

• a machine with **MAC address** lies in direction of the **port** number from the bridge. The entry is **age** time units old.
(1) Frame Forwarding

- Assume a MAC frame arrives at port x.

Is MAC address of destination in forwarding table?

Found?

Forward the frame on the appropriate port

Not found?

Flood the frame, i.e., send the frame on all ports except port x.
(2) Address Learning

- When a bridge reboots, its forwarding table is empty
- Forwarding table entries are *learned* automatically with a simple heuristic:

The source field of a frame that arrives on a port tells which hosts are reachable from this port.
(2) Address Learning

Algorithm:

- For each frame received, the source stores the source field in the forwarding table together with the port where the frame was received.

- All entries are deleted after some time (default is 15 seconds).
  - What if the host moved?
Example

• Consider the following packets: (Src=A, Dest=F), (Src=C, Dest=A), (Src=E, Dest=C)

• What have the bridges learned?
So far so good, but

- Strategy works fine if the extended LAN does not have a loop in it

- Why?
  - Frames potentially loop through the extended LAN forever

- Bridges B1, B4, and B6 form a loop
Why loop?

• Consider the two LANs that are connected by two bridges.
  – Two bridges increase fault tolerance to failures.
  – Network is built by more than one administrator

• Solution: the spanning tree algorithm
Spanning Tree Algorithm

- A solution is the spanning tree algorithm that prevents loops in the topology
  - By Radia Perlman at DEC
Algorhyme (the spanning tree poem)

- I think that I shall never see
  A graph more lovely than a tree.
  A tree whose crucial property
  Is loop-free connectivity.
  A tree that must be sure to span
  So packets can reach every LAN.
  First, the root must be selected.
  By ID, it is elected.
  Least-cost paths from root are traced.
  In the tree, these paths are placed.
  A mesh is made by folks like me,
  Then bridges find a spanning tree.

- —Radia Perlman
Graph theory on spanning tree

• For any connected graph consisting of nodes and edges connecting pairs of nodes, a spanning tree of edges maintains the connectivity of the graph but contains no loops
  – N-node’s graph, N-1 edges on a spanning tree
  – No redundancy
The protocol

• IEEE 802.1d has an algorithm that organizes the bridges as spanning tree in a dynamic environment
  – Note: Trees don’t have loops

• Bridges exchange messages to configure the bridge (Configuration Bridge Protocol Data Unit, Configuration BPDUs) to build the tree
  – Select ports they use to forward packets
Configuration BPDUs

- **Destination MAC address**
- **Source MAC address**
- **Configuration Message**

### Protocol Identifier
- **Version**
- **Message Type**
- **Flags**
- **Root ID**
- **Cost**
- **Bridge ID**
- **Port ID**
- **Message Age**
- **Maximum Age**
- **Hello Time**
- **Forward Delay**

- **Set to 0**
- **Set to 0**
- **Set to 0**

- **Lowest bit is "topology change bit (TC bit)"**
- **ID of root**
- **Cost of the path from the bridge sending this message to root bridge**
- **ID of bridge sending this message**
- **ID of port from which message is sent**

- **Time between BPDUs from the root (default: 1 sec)**
- **Time between recalculations of the spanning tree (default: 15 secs)**

- **Time since root sent a message on which this message is based**
What do the BPDUs do?

• Elect a single bridge as the **root bridge**

• Calculate the distance of the shortest path to the root bridge

• Each bridge can determine a **root port**, the port that gives the best path to the root

• Each LAN can determine a **designated bridge**, which is the bridge closest to the root. A LAN's *designated bridge* is the only bridge allowed to forward frames to and from the LAN for which it is the designated bridge.

• A LAN's *designated port* is the port that connects it to the designated bridge

• Select ports to be included in the spanning tree.
Terms

• Each bridge has a unique identifier: **Bridge ID**
  
  Bridge ID = {Priority : 2 bytes; Bridge MAC address: 6 bytes}
  
  • Priority is configured
  • Bridge MAC address is the lowest MAC addresses of all ports

• Each port within a bridge has a unique identifier (port ID)

• **Root Bridge:** The bridge with the lowest identifier is the root of the spanning tree

• **Root Port:** Each bridge has a root port which identifies the next hop from a bridge to the root
Terms

• **Root Path Cost:** For each bridge, the cost of the min-cost path to the root
  – Assume it is measured in #hops to the root

• **Designated Bridge, Designated Port:** Single bridge on a LAN that is closest to the root for this LAN:
  – If two bridges have the same cost, select the one with the highest priority; if they have the same priority, select based on the bridge ID
  – If the min-cost bridge has two or more ports on the LAN, select the port with the lowest identifier
Spanning Tree Algorithm

• Each bridge is sending out BPDUs that contain the following information:
  
  - root ID
  - cost
  - bridge ID
  - port ID

  root bridge (what the sender thinks it is)
  root path cost for sending bridge
  Identifies sending bridge
  Identifies the sending port

• The transmission of BPDUs results in the distributed computation of a spanning tree

• The convergence of the algorithm is very fast
Ordering of Messages

- We define an ordering of BPDU messages (lexicographically)

We say $M_1$ advertises a better path than $M_2$ ("$M_1 \ll M_2$") if

$(R_1 < R_2)$,  
Or $(R_1 = R_2)$ and $(C_1 < C_2)$,  
Or $(R_1 = R_2)$ and $(C_1 = C_2)$ and $(B_1 < B_2)$,  
Or $(R_1 = R_2)$ and $(C_1 = C_2)$ and $(B_1 = B_2)$ and $(P_1 < P_2)$
Initializing the Spanning Tree Protocol

• Initially, all bridges assume they are the root bridge.
• Each bridge B sends BPDUs of this form on its LANs from each port P:

\[
\begin{array}{c}
B \\
0 \\
B \\
P
\end{array}
\]

• Each bridge looks at the BPDUs received on all its ports and its own transmitted BPDUs.
• Root bridge is the smallest received root ID that has been received so far (Whenever a smaller ID arrives, the root is updated)
Spanning Tree Protocol

- Each bridge B looks on all its ports for BPDUs that are better than its own BPDUs
- Suppose a bridge with BPDU:
  
  \[
  \begin{array}{c}
  M1 \\
  \end{array} \\
  \begin{array}{cccc}
  R1 & C1 & B1 & P1 \\
  \end{array}
  \]

receives a “better” BPDU:

\[
\begin{array}{c}
M2 \\
\end{array} \\
\begin{array}{cccc}
R2 & C2 & B2 & P2 \\
\end{array}
\]

Then it will update the BPDU to:

\[
\begin{array}{cccc}
R2 & C2+1 & B1 & P1 \\
\end{array}
\]

- However, the new BPDU is not necessarily sent out
- On each bridge, the port where the “best BPDU” (via relation “<“) was received is the root port of the bridge
  - No need to send out updated BPDUs to root port
When to send a BPDU

- Say, B has generated a BPDU for each port x

<table>
<thead>
<tr>
<th>R</th>
<th>Cost</th>
<th>B</th>
<th>x</th>
</tr>
</thead>
</table>

- B will send this BPDU on port x only if its BPDU is better (via relation “<“) than any BPDU that B received from port x.

- In this case, B also assumes that it is the designated bridge for the LAN to which the port connects

- And port x is the designated port of that LAN
Selecting the Ports for the Spanning Tree

- Each bridge makes a local decision which of its ports are part of the spanning tree
- Now B can decide which ports are in the spanning tree:
  - B’s root port is part of the spanning tree
  - All designated ports are part of the spanning tree
  - All other ports are not part of the spanning tree

- B’s ports that are in the spanning tree will forward packets (=forwarding state)
- B’s ports that are not in the spanning tree will not forward packets (=blocking state)
Building the Spanning Tree

- Consider the network on the right.
- Assume that the bridges have calculated the designated ports (D) and the root ports (R) as indicated.

- What is the spanning tree?
  - On each LAN, connect D ports to the R ports on this LAN
  - Which bridge is the root bridge?

- Suppose a packet is originated in LAN 5. How is the packet flooded?
Example

• Assume that all bridges send out their BPDU’s once per second, and assume that all bridges send their BPDUs at the same time.

• Assume that all bridges are turned on simultaneously at time $T=0$ sec.
Example: BPDUs sent

<table>
<thead>
<tr>
<th></th>
<th>Bridge1</th>
<th>Bridge2</th>
<th>Bridge3</th>
<th>Bridge4</th>
<th>Bridge5</th>
</tr>
</thead>
<tbody>
<tr>
<td>T=1sec</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example: BPDUs sent

<table>
<thead>
<tr>
<th></th>
<th>Bridge1</th>
<th>Bridge2</th>
<th>Bridge3</th>
<th>Bridge4</th>
<th>Bridge5</th>
</tr>
</thead>
<tbody>
<tr>
<td>T=1sec</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example: BPDUs sent

<table>
<thead>
<tr>
<th></th>
<th>Bridge1</th>
<th>Bridge2</th>
<th>Bridge3</th>
<th>Bridge4</th>
<th>Bridge5</th>
</tr>
</thead>
<tbody>
<tr>
<td>T=3sec</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Example: BPDUs sent

<table>
<thead>
<tr>
<th></th>
<th>Bridge1</th>
<th>Bridge2</th>
<th>Bridge3</th>
<th>Bridge4</th>
<th>Bridge5</th>
</tr>
</thead>
<tbody>
<tr>
<td>T=1 sec</td>
<td>Send: A: (B1,0,B1,A) B: (B1,0,B1,B)</td>
<td>Send: A: (B2,0,B2,A) B: (B2,0,B2,B)</td>
<td>Send: A: (B3,0,B3,A) B: (B3,0,B3,B)</td>
<td>Send: A: (B4,0,B4,A) B: (B4,0,B4,B)</td>
<td>Send: A: (B5,0,B5,A) B: (B5,0,B5,B)</td>
</tr>
<tr>
<td></td>
<td>A: (B5,0,B5,A) B: (B2,0,B2,B) B: (B3,0,B3,B)</td>
<td>B: (B1,0,B1,A) (B5,0,B5,A)</td>
<td>A: (B5,0,B5,B) B: (B4,0,B4,B) A: (B3,0,B3,B)</td>
<td>A: (B4,0,B4,A) B: (B1,0,B1,B) B: (B3,0,B3,A)</td>
<td>A: (B2,0,B2,B) B: (B3,0,B3,A)</td>
</tr>
<tr>
<td></td>
<td>Send: A: (B3,0,B3,A) B: (B3,0,B3,B)</td>
<td>Recv: B: (B1,0,B1,B)</td>
<td>Send: A: (B5,0,B5,B) B: (B4,0,B4,B) A: (B3,0,B3,B)</td>
<td>Send: A: (B4,0,B4,A) B: (B1,0,B1,B) B: (B3,0,B3,A)</td>
<td>Send: A: (B2,0,B2,B) B: (B3,0,B3,A)</td>
</tr>
<tr>
<td></td>
<td>B: (B1,0,B1,A) (B5,0,B5,A)</td>
<td>Recv: A: (B5,0,B5,B) B: (B4,0,B4,B) A: (B3,0,B3,B)</td>
<td>Recv: A: (B5,0,B5,B) B: (B4,0,B4,B) A: (B3,0,B3,B)</td>
<td>Recv: A: (B4,0,B4,A) B: (B1,0,B1,B) B: (B3,0,B3,A)</td>
<td>Recv: A: (B2,0,B2,B) B: (B3,0,B3,A)</td>
</tr>
</tbody>
</table>
# Example: BPDU’s sent

<table>
<thead>
<tr>
<th>Bridge1</th>
<th>Bridge2</th>
<th>Bridge3</th>
<th>Bridge4</th>
<th>Bridge5</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>T=2sec</strong></td>
<td><strong>D-port: A,B</strong>&lt;br&gt;Send: A: (B1,0,B1,A) B: (B1,0,B1,B)Recv: A: B: (B1,0,B1,A)</td>
<td><strong>R-port: B</strong>&lt;br&gt;Send: A: (B1,1,B2,A)Recv: A: B: (B1,0,B1,A)</td>
<td><strong>R-port: B</strong>&lt;br&gt;Send: A: (B1,1,B3,A)Recv: A: B: (B1,0,B1,B) B: (B1,0,B1,B)</td>
<td><strong>R-port: A</strong>&lt;br&gt;Send: B: (B1,1,B4,B)Recv: A: (B1,0,B1,B) B: (B1,1,B3,A) (B1,1,B5,B)</td>
</tr>
</tbody>
</table>
Example: BPDU’s sent

<table>
<thead>
<tr>
<th>T=3sec</th>
<th>Bridge 1</th>
<th>Bridge 2</th>
<th>Bridge 3</th>
<th>Bridge 4</th>
<th>Bridge 5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>D-port: A,B</td>
<td>R-port: B</td>
<td>R-port: B</td>
<td>R-port: A</td>
<td>R-port: A</td>
</tr>
<tr>
<td></td>
<td>Send: A: (B1,0,B1,A)</td>
<td>D-port: A</td>
<td>Send: A: (B1,1,B2,A)</td>
<td>Blocked: B</td>
<td>Blocked: B</td>
</tr>
<tr>
<td></td>
<td>B: (B1,0,B1,B)</td>
<td>R-port: B</td>
<td>Send: A: (B1,0,B1,B)</td>
<td>Recv: A:</td>
<td>Recv: A:</td>
</tr>
<tr>
<td></td>
<td>Recv: A:</td>
<td>D-port: A</td>
<td>B: (B1,0,B1,A)</td>
<td>(B1,1,B3,A)</td>
<td>(B1,1,B3,A)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>B: (B1,0,B1,B)</td>
<td></td>
</tr>
</tbody>
</table>

Example: the spanning tree

<table>
<thead>
<tr>
<th></th>
<th>Bridge1</th>
<th>Bridge2</th>
<th>Bridge3</th>
<th>Bridge4</th>
<th>Bridge5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root Port</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Designated bridge</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Designated ports</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

A packet is sent from LAN2
Example: the spanning tree

<table>
<thead>
<tr>
<th></th>
<th>Bridge1</th>
<th>Bridge2</th>
<th>Bridge3</th>
<th>Bridge4</th>
<th>Bridge5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root Port</td>
<td></td>
<td>B</td>
<td>B</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>Designated bridge</td>
<td>LAN2,3</td>
<td>LAN1</td>
<td>LAN4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Designated ports</td>
<td>A,B</td>
<td>A</td>
<td>A</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

A packet is sent from LAN2
Limitations of bridges

• Scalability
  – Broadcast packets reach every host!

• Security
  – Every host can snoop

• Non-heterogeneity
  – Can’t connect ATM networks
Summary

• Switching
  – Datagram
  – Virtual circuit
  – Source routing

• Ethernet learning bridges