Lecture 6: Bridges and Switches

CS/ECE 438: Communication Networks
Prof. Matthew Caesar
February 19, 2010
How can many hosts communicate?

- Naïve approach: full mesh
- Problem: doesn’t scale
How can many hosts communicate?

- Solution: direct-link networks
- But, how to deal with larger networks (more hosts, larger geographic area)?
- How to deal with heterogeneous media? (different physical/MAC technologies?)
1. How can many hosts communicate?

- Solution: Multiplex traffic with switches/bridges/routers:
  - Translate between different link technologies to allow heterogeneity
  - Isolate different physical networks to improve scalability
  - Give illusion of single connection to provide transparency
Challenges in router/switch design

• How to compute paths to destinations?
  – Solution: routing protocols

• How to quickly look up those paths?
  – Solution: lookup algorithms

• How to handle simultaneously-arriving packets?
  – Solution: interconnection networks, queuing
Switches

- Switch provides local star topology
- Build network from stars
Challenges

Which way should they go?

Destination: New York

Destination: Boston

Who goes first?
Challenges

- **Efficient forwarding**
  - Switch with several output ports
  - Decide which output port to use

- **Routing in a dynamic network**
  - Need information for forwarding
  - Construct and maintain the information

- **Handling contention**
  - Multiple packets destined for one output port
  - Decide which packet goes first
  - Decide what to do with others
Router architecture

Update 12.0.0.0/8

Control Plane (routing)

Data Plane (forwarding)

Interconnection network

Protocol daemon

RIB 12.0.0.0/8 → IF 1

FIB 12.0.0.0/8 → IF 1

12.0.0.0/8

12.10.2.3

Data packet
Network Layers and Switches
Network Layers and Switches

Switch between different physical/mac layers

Switch

Network

Data Link A
Physical A

Data Link B
Physical B

interface #2
(wired ethernet)

interface #1
(wireless 802.11)

802.11 Header
IP header
TCP header
data

TCP header
IP header
802.11 Header
data
Switching and Forwarding

- **Switch**
  - A switch forwards packets from input ports to output ports
    - Port selection is based on the destination address in the packet header
  - **Advantages**
    - Can build networks that cover large geographic areas
    - Can build networks that support large numbers of hosts
    - Can add new hosts without affecting the performance of existing hosts
Switching and Forwarding

- **Forwarding**
  - The task of specifying an appropriate output port for a packet
  - Each switch determines the correct output port based on
    - State in FIB
    - State in packet headers
  - Later
    - Building forwarding tables – routing.
Forwarding

• Packet switching
  – Data traffic divided into packets
    • Each packet contains its own header (with address)
    • Packets sent separately through the network
  – Destination reconstructs the message
  – Example: sending a letter through the postal system

• Circuit switching
  – Source first establishes a connection to the destination
    • Each router on the path may reserve bandwidth
  – Source sends data over the connection
    • No destination address, since routers know the path
  – Source tears down connection when done
  – Example: making a phone call on the telephone network
Forwarding with Datagrams

Destination is E, so I should forward out interface #1

Destination is E, so I should forward out interface #2
Each switch maintains a forwarding table that translates a host name to an output port.
Each switch maintains a forwarding table that translates a host name to an output port.
Routing Table

A sends: E DATA
C sends: F DATA
B sends: E DATA
F sends: G DATA
A sends: H DATA

α’s Table
<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>A</td>
<td>3</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>B</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>C</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>D</td>
<td>2</td>
</tr>
<tr>
<td>E</td>
<td>1</td>
<td>E</td>
<td>3</td>
</tr>
<tr>
<td>F</td>
<td>1</td>
<td>F</td>
<td>3</td>
</tr>
<tr>
<td>G</td>
<td>3</td>
<td>G</td>
<td>0</td>
</tr>
</tbody>
</table>

β’s Table
<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>3</td>
<td>A</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>3</td>
<td>B</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>C</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>D</td>
<td>0</td>
</tr>
<tr>
<td>E</td>
<td>1</td>
<td>E</td>
<td>1</td>
</tr>
<tr>
<td>F</td>
<td>3</td>
<td>F</td>
<td>3</td>
</tr>
</tbody>
</table>

γ’s Table
<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>3</td>
<td>A</td>
<td>3</td>
</tr>
<tr>
<td>B</td>
<td>3</td>
<td>B</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>C</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>2</td>
<td>D</td>
<td>0</td>
</tr>
<tr>
<td>E</td>
<td>3</td>
<td>E</td>
<td>1</td>
</tr>
<tr>
<td>F</td>
<td>3</td>
<td>F</td>
<td>3</td>
</tr>
</tbody>
</table>

δ’s Table
<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>A</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>B</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>C</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>D</td>
<td>0</td>
</tr>
<tr>
<td>E</td>
<td>1</td>
<td>E</td>
<td>1</td>
</tr>
<tr>
<td>F</td>
<td>3</td>
<td>F</td>
<td>3</td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td>G</td>
<td>0</td>
</tr>
</tbody>
</table>

DATA
A sends: DATA
C sends: DATA
B sends: DATA
F sends: DATA
A sends: DATA
Forwarding with Datagrams

- Advantages
  - Statistical multiplexing
  - Can route around failures dynamically

- Disadvantages
  - Header requires full unique address
  - No explicit signaling → harder to provide quality guarantees
  - Successive packets may not follow the same route
Data from interface #0 should be sent to interface #1

Data from interface #3 should be sent to interface #2

Circuit establishment
### Circuit switching vs. Datagram switching

<table>
<thead>
<tr>
<th>Circuits</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1. Guaranteed bandwidth</td>
<td>1. Wasted bandwidth</td>
</tr>
<tr>
<td></td>
<td>2. Simple Abstraction</td>
<td>2. Blocked connections</td>
</tr>
<tr>
<td></td>
<td>3. Simple forwarding</td>
<td>3. No communication until channel set up</td>
</tr>
<tr>
<td></td>
<td>4. Low per-packet overhead</td>
<td>4. Routers need per-connection state</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Datagrams</td>
<td>1. Statistical multiplexing</td>
<td>1. Unpredictable performance</td>
</tr>
<tr>
<td></td>
<td>2. Offers “okay” service to everyone</td>
<td>2. Lost, out of order packets</td>
</tr>
<tr>
<td></td>
<td>3. No set-up delay</td>
<td>3. More complex forwarding (prefix matching)</td>
</tr>
<tr>
<td></td>
<td>4. Routers only store aggregated routes</td>
<td>4. Packet header overhead</td>
</tr>
</tbody>
</table>

## Advantages

- Guaranteed bandwidth
- Simple Abstraction
- Simple forwarding
- Low per-packet overhead

## Disadvantages

- Wasted bandwidth
- Blocked connections
- No communication until channel set up
- Routers need per-connection state

- Statistical multiplexing
- Offers “okay” service to everyone
- No set-up delay
- Routers only store aggregated routes

- Unpredictable performance
- Lost, out of order packets
- More complex forwarding (prefix matching)
- Packet header overhead
Virtual Circuits

• Hybrid of packet and circuit switching
  – Logical circuit between source and destination
  – Packets from different VCs multiplex on a link
  – Used in ATM, MPLS, Intserv

• Virtual Circuit Identifier (VCI)
  – Source setup: establish path for each VC
  – Switch: mapping VCI to outgoing link
  – Packet: fixed-length label in packet header
Forwarding with Virtual Circuits

Virtual circuit establishment

12:1

12:2
Label swapping

- Problem: using VCI along the whole path
  - Each virtual circuit consumes a unique ID
  - Starts to use up all of the ID space in the network
- Label swapping (used in MPLS)
  - Map the VCI to a new value at each hop
    - Table has old ID, next link, and new ID
    - Allows reuse of IDs at different links
Forwarding with Virtual Circuits

• Connection oriented
  – Requires explicit setup and teardown
  – Packets follow established route

• Why support connections in a network?
  – Useful for service notions
  – Important for telephony

• Switch
  – Translates virtual circuit ID on incoming link to virtual circuit ID on outgoing link
  – CircuitIds can be per-link or per-switch
Forwarding with Virtual Circuits

- **Set up**
  - A virtual circuit identifier (VCI) is assigned to the circuit for each link it traverses
  - VCI is locally significant
  - \(<\text{incoming port}, \text{incoming VCI}>\) uniquely identifies VC

- **Switch**
  - Maintains a translation table from \(<\text{incoming port}, \text{incoming VCI}>\) to \(<\text{outgoing port}, \text{outgoing VCI}>\)

- **Permanent Virtual Circuits (PVC)**
  - Long-lived

- **Switch Virtual Circuits (SVC)**
  - Uses signaling to establish VC
A simple example setup protocol

- Each host and switch maintains per-link local variable for VCI assignment
- When setup frame leaves host/switch
  - Assign outgoing VCI
- port and circuit id combination is unique
- switches maintain translation table from
  - incoming port/VCI pair to
  - outgoing port/VCI pair
Forwarding with Virtual Circuits

• Assumptions
  – Circuits are simplex
    • On a duplex link, the same VCI can be used for two circuits, one in each direction
  – The same VCI can be used on different ports of the same switch
  – At setup, the lowest available VCI is used
Setting up Virtual Circuit A→E

<table>
<thead>
<tr>
<th>Port IN</th>
<th>VCI IN</th>
<th>Port OUT</th>
<th>VCI OUT</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>no</td>
</tr>
</tbody>
</table>

Setup msg
Dst=E, VCI=0

Port IN | VCI IN | Port OUT | VCI OUT | Valid |
---------|--------|----------|---------|-------|
3        | 0      | 2        | 0       | no    |

Port IN | VCI IN | Port OUT | VCI OUT | Valid |
---------|--------|----------|---------|-------|
0        | 0      | 1        | 0       | no    |

Port IN | VCI IN | Port OUT | VCI OUT | Valid |
---------|--------|----------|---------|-------|
3        | 0      | 2        | 0       | no    |

Host A

Host B

Host C

Host D

Host E

Host F

Host G
Setting up Virtual Circuit A→E

<table>
<thead>
<tr>
<th>Port IN</th>
<th>VCI IN</th>
<th>Port OUT</th>
<th>VCI OUT</th>
<th>Valid ?</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>yes</td>
</tr>
</tbody>
</table>

Host A

<table>
<thead>
<tr>
<th>Port IN</th>
<th>VCI IN</th>
<th>Port OUT</th>
<th>VCI OUT</th>
<th>Valid ?</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>yes</td>
</tr>
</tbody>
</table>

Host B

<table>
<thead>
<tr>
<th>Port IN</th>
<th>VCI IN</th>
<th>Port OUT</th>
<th>VCI OUT</th>
<th>Valid ?</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>yes</td>
</tr>
</tbody>
</table>

Host C

<table>
<thead>
<tr>
<th>Port IN</th>
<th>VCI IN</th>
<th>Port OUT</th>
<th>VCI OUT</th>
<th>Valid ?</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>yes</td>
</tr>
</tbody>
</table>

Host D

<table>
<thead>
<tr>
<th>Port IN</th>
<th>VCI IN</th>
<th>Port OUT</th>
<th>VCI OUT</th>
<th>Valid ?</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>yes</td>
</tr>
</tbody>
</table>

Host F

<table>
<thead>
<tr>
<th>Port IN</th>
<th>VCI IN</th>
<th>Port OUT</th>
<th>VCI OUT</th>
<th>Valid ?</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>yes</td>
</tr>
</tbody>
</table>

Host E

ACK msg VCI=0
Setting up Virtual Circuit G→C

<table>
<thead>
<tr>
<th>Port IN</th>
<th>VCI IN</th>
<th>Port OUT</th>
<th>VCI OUT</th>
<th>Valid ?</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>yes</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>yes</td>
</tr>
</tbody>
</table>

Setup msg
Dst=C, VCI

ACK msg
VCI=0

<table>
<thead>
<tr>
<th>Port IN</th>
<th>VCI IN</th>
<th>Port OUT</th>
<th>VCI OUT</th>
<th>Valid ?</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>yes</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>yes</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>yes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Port IN</th>
<th>VCI IN</th>
<th>Port OUT</th>
<th>VCI OUT</th>
<th>Valid ?</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>no</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>no</td>
</tr>
</tbody>
</table>
Forwarding with Virtual Circuits

Set up circuits:
- A $\rightarrow$ E
- C $\rightarrow$ F
- G $\rightarrow$ E
Forwarding with Virtual Circuits

Table entries after A→E connection is set

<table>
<thead>
<tr>
<th>Port IN</th>
<th>VCI IN</th>
<th>Port OUT</th>
<th>VCI OUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Port IN</th>
<th>VCI IN</th>
<th>Port OUT</th>
<th>VCI OUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>
### Forwarding with Virtual Circuits

Table entries after A→E, C→F, G→E connection is set

<table>
<thead>
<tr>
<th>Port IN</th>
<th>VCI IN</th>
<th>Port OUT</th>
<th>VCI OUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>α</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Port IN</th>
<th>VCI IN</th>
<th>Port OUT</th>
<th>VCI OUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>β</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Port IN</th>
<th>VCI IN</th>
<th>Port OUT</th>
<th>VCI OUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>γ</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>3</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Port IN</th>
<th>VCI IN</th>
<th>Port OUT</th>
<th>VCI OUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>δ</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Forwarding with Virtual Circuits

- Analogous to a game of following a sequence of clues
- **Advantages**
  - Header (for a data packet) requires only virtual circuit ID
    - Connection request contains global address
  - Can reserve resources at setup time
- **Disadvantages**
  - Typically must wait one RTT for setup
  - Cannot dynamically avoid failures, must reestablish connection
  - Global address path information still necessary for connection setup
Similarities between virtual circuits and datagrams

- Data divided into packets
- Store-and-forward transmission
- Packets multiplexed onto links
Differences between virtual circuits and datagrams

- **Forwarding lookup**
  - IP: longest prefix match
  - VC: circuit ID
- **Connection setup**
  - IP: send packets on-demand
  - VC: set up circuit in advance
- **Router state**
  - IP: no per-circuit state, easier failure recovery
  - VC: routers know about connections
- **Quality of service**
  - IP: no reservations, no guarantees
  - VC: reserved bandwidth, soft QoS guarantees
Forwarding with Source Routing

• Packet header specifies directions
  – One direction per switch
    • Absolute
      – Port name
      – Next switch name
    • Relative
      – Turn clockwise 3 ports
  – Switches may delete or rotate directions within packet headers
Forwarding with Source Routing

Ports to use at each hop:

- Host A: 1 2 1
- Host G: 0
- Host B: 3 1
- Host F: 3 2
- Host E: 1
- Host D: 0
- Host C: 1

Port to use at first hop:
- Host G: 3

Port to use at second hop:
- Host F: 0

Port to use at third hop:
- Host E: 1
Forwarding with Source Routing

What happens to the last packet?

A sends:

A → E

C sends:

C → F

B sends:

B → E

F sends:

F → G

A sends:

A → E

C → F

B → E

F → G

A sends:
Forwarding with Source Routing

• Analogous to following directions
• Advantages
  – Simplifies forwarding/lookup
  – End hosts can select paths based on application-specific requirements
• Disadvantages
  – Hosts must know entire up-to-date topology
  – Headers might get large
  – Malicious hosts can DoS-attack by choosing inefficient paths
Forwarding Performance

- General purpose work station
  - Direct memory access (DMA)
  - Supports multiple network interface cards (NICs)
Forwarding Performance of a Software Router

- Switching process
  - Packet arrives on NIC 1
  - NIC 1 DMA’s packet into memory
  - CPU looks at header, decides to send out on NIC 2
  - NIC 2 DMA’s packet into NIC memory
  - Packet leaves via NIC 2

What are the potential bottlenecks?
Forwarding Performance

• Potential Bottlenecks
  – I/O bus bandwidth
  – Memory bus bandwidth
  – Processor computing power

• Example
  – Workstation – switches 100,000 pps
  – Average packet size = 64 bytes
  – Throughput
    • = pps x (BitsPerPacket)
    • = 100 x 10^3 x 64 x 8
    • = 51.2 x 10^6 bits per second

• Solution: do forwarding in hardware
  – Forwarding performed in silicon, fast memory
Bridges and LAN Switches
Bridges: Building Extended LANs

- **Traditional LAN**
  - Shared medium (e.g., Ethernet)
  - Cheap, easy to administer
  - Supports broadcast traffic

- **Problem**
  - Scale LAN concept
  - Larger geographic area (> O(1 km))
  - More hosts (> O(100))
  - But retain LAN-like functionality

- **Solution**
  - bridges
Bridges

- **Problem**
  - LANs have physical limitations
    - Ethernet – 1500m

- **Solution**
  - Connect two or more LANs with a bridge
    - Accept and forward
    - Level 2 connection (no extra packet header)
  - A collection of LANs connected by bridges is called an extended LAN
Bridges vs. Switches

- **Switch**
  - Receive frame on input port
  - Translate address to output port
  - Forward frame

- **Bridge**
  - Connect shared media
  - All ports bidirectional
  - Repeat subset of traffic
    - Receive frame on one port
    - Send on all other ports
Uses and Limitations of Bridges

• Bridges
  – Extend LAN concept
  – Limited scalability
    • to O(1,000) hosts
    • not to global networks
  – Not heterogeneous
    • some use of address, but
    • no translation between frame formats

• Bridge outline
  – Learning bridges
  – Spanning tree algorithm
  – Broadcast and multicast
Learning Bridges

• Problem
  – Which LANs should a frame be forwarded on?

• Trivial algorithm
  – Forward all frames on all (other) LAN’s
  – Potentially heavy traffic and processing overhead

• Optimize by using address information
  – “Learn” which hosts live on which LAN
  – Maintain forwarding table
  – Only forward when necessary
  – Reduces bridge workload
Learning Bridges

- Bridge learns table entries based on source address
  - When receive frame from A on port 1
    add A to list of hosts on port 1
  - Time out entries to allow movement of hosts

- Table is an “optimization”, meaning it helps performance but is not mandatory

- Always forward broadcast frames
Learning Bridges

- Frame for A received on port 1:
- Frame for C received on port 2:
- Frame for S received on port 2: do nothing
  forward to port 1
  forward to port 1
• Problem
  – If there is a topological loop in the extended LAN, a packet could circulate forever

• Solution
  – Select which bridges should actively forward
  – Create a **spanning tree** to eliminate unnecessary edges
    • Not necessarily minimum cost spanning tree
    • Operator can change tree by shifting the root node
  – Prevents loops, but complicates learning/forwarding
Example Extended LAN with Loops
Defining a Spanning Tree

1. Choose a root node
   - Bridge with lowest ID (priority, MAC) is the root bridge
   - Priority value is configurable
2. Determine lowest-cost paths to root bridge
   - Cost of traversing each segment is configurable
   - Each bridge determines cost from itself to the root, selects lowest cost path port as root port
   - Bridges on each segment determine which (designated bridge) has shortest path to root, that bridge’s port is the designated port
3. Break ties
   1. Use lowest bridge ID, then lowest port priority
Spanning Tree Algorithm

- D – designated port
- R – root port
- B – blocked port

Root bridge


- Root bridge

- Designated port

- Root port
Using a Spanning Tree: Forwarding

- **Forwarding**
  - Each bridge forwards frames over each LAN for which it is the designated bridge or connected by a root port

- **Suppressing**
  - A bridge does not forward a frame over a port if it knows that the destination is not on the other side of the port
Using a Spanning Tree: Broadcast and Multicast

- Forward all broadcast/multicast frames to all non-blocked ports
- Learn when there are no group members downstream
  - Have each member of group G send a frame with multicast address G in it to a bridge
Finding the Tree by a Distributed Algorithm

- Bridges run a distributed spanning tree algorithm
  - Select when bridges should actively forward frames

- Developed by Radia Perlman at DEC
- Now IEEE 802.1 specification
Bridges exchange configuration messages
- \((Y,d,X)\)
  - \(Y\) = root node
  - \(d\) = distance to root node
  - \(X\) = originating node

Each bridge records current best configuration message for each port.

Initially, each bridge believes it is the root.

When a bridge discovers it is not the root, stop generating messages.
Distributed Spanning Tree Algorithm

- Bridges forward configuration messages
  - Outward from root bridge
  - i.e., on all designated ports

- Bridge assumes
  - It is designated bridge for a LAN
  - Until it learns otherwise

- Steady State
  - root periodically send configuration messages
  - A timeout is used to restart the algorithm
• Example at bridge B9
  1. B9 receives (B2, 0, B2)
  2. Since 2 < 9, B9 accepts B2 as root
  3. B9 adds 1 to the distance advertised by B2 and sends (B2, 1, B9)
  4. B2 accepts B1 as root and sends (B1, 1, B2)
  5. B5 accepts B1 as root and sends (B1, 1, B5)
  6. B9 accepts B1 as root and stops forwarding
Ethernet vs. IP

• Ethernet has many benefits over IP
  – Simplifies network management, greatly reducing operational expense
  – Simplifies access control lists (ACLs), host mobility

• Why do we still use IP routing inside a single network?
Problem: Ethernet doesn’t scale

- Reasons for poor scalability
  - Network-wide flooding
  - Unbalanced link utilization, low availability and throughput due to tree-based forwarding

- Scalability requirement is growing very fast
  - Large enterprises: 50k end hosts
  - Data centers: 100k servers, 5k switches
  - Metro-area Ethernet: over 1M subscribers
Improving Ethernet’s scalability

• Solution 1: Logically partition topology with Virtual LANs (VLANs)
  – Limit scope of broadcasts within each VLAN

• Solution 2: Avoid using broadcast to construct state
  – E.g., Perlman’s \textit{RBridges} distribute host state in link-state advertisements
Scaling Ethernet with VLANs

- Divide up hosts into logical groups called **VLANs**
- Each VLAN corresponds to IP subnet, single broadcast domain
- Ethernet packet headers have VLAN tag
- Bridges forward packet only on subnets on corresponding VLAN
Virtual LANs

• Downsides of VLANs
  – Are manually configured, complicates network management
  – Hard to seamlessly migrate across VLAN boundaries due to addressing restrictions

• Upsides of VLANs
  – Limits scope of broadcasts
  – Logical separation improves isolation, security
  – Can change virtual topology without changing physical topology
    • E.g., used in data centers for VM migration