Datalink Classification

- Datalink
  - Switch-based
    - Virtual Circuits: ATM, framerelay
    - Packet Switching: Bridged LANs
  - Multiple Access
    - Scheduled Access: Token ring, FDDI, 802.11
    - Random Access: Ethernet, 802.11, Aloha

Outline

- Bridging and switching
  - Scaling the network
  - Spanning tree protocol
  - Why Ethernet?

- Something different

Scale

- What breaks when we keep adding people to the same wire?
Scale

• What breaks when we keep adding people to the same wire?
• Only solution: split up the people onto multiple wires
• But how can they talk to each other?

Building Larger LANs: Bridges

• Extend reach of a single shared medium
• Connect two or more “segments” by copying data frames between them
  • Only copy data when needed \(\rightarrow\) key difference from repeaters/hubs
  • Reduce collision domain compared with single LAN
  • Separate segments can send at once \(\rightarrow\) much greater bandwidth
• Challenge: learning which packets to copy across links

Transparent Bridges

• Design goals:
  • Self-configuring without hardware or software changes
  • Bridge do not impact the operation of the individual LANs
• Three parts to making bridges transparent:
  1) Forwarding frames
  2) Learning addresses/host locations
  3) Spanning tree algorithm

Frame Forwarding

• A machine with MAC Address lies in the direction of number port of the bridge
• For every packet, the bridge “looks up” the entry for the packets destination MAC address and forwards the packet on that port.
  • Other packets are broadcast – why?
• Timer is used to flush old entries
Learning Bridges

- Manually filling in bridge tables?
  - Time consuming, error-prone
- Keep track of source address of packets arriving on every link, showing what segment hosts are on
  - Fill in the forwarding table based on this information

Spanning Tree Bridges

- More complex topologies can provide redundancy.
  - But can also create loops.
- What is the problem with loops?
- Solution: spanning tree

Spanning Tree Protocol Overview

Embed a tree that provides a single unique path to each destination:
1) Elect a single bridge as a root bridge
2) Each bridge calculates the distance of the shortest path to the root bridge
3) Each LAN identifies a designated bridge, the bridge closest to the root. It will forward packets to the root.
4) Each bridge determines a root port, which will be used to send packets to the root
5) Identify the ports that form the spanning tree

Spanning Tree Algorithm Steps

- Root of the spanning tree is the bridge with the lowest identifier.
  - All ports are part of tree
- Each bridge finds shortest path to the root.
  - Remembers port that is on the shortest path
  - Used to forward packets
- Select for each LAN the designated bridge that has the shortest path to the root.
  - Identifier as tie-breaker
  - Responsible for that LAN
Spanning Tree Algorithm

- Each node sends configuration message to all neighbors:
  - Identifier of the sender
  - Id of the presumed root
  - Distance to the presumed root
  - E.g., B5 sends (B5, B5, 0)
- When B receives a message, it decides whether the solution is better than their local solution:
  - A root with a lower identifier?
  - Same root but lower distance?
  - Same root, distance but sender has lower identifier?
- After convergence, each bridge knows the root, distance to root, root port, and designated bridge for each LAN.

Spanning Tree Algorithm (part 2)

- Each bridge B can now select which of its ports make up the spanning tree:
  - B’s root port
  - All ports for which B is the designated bridge on the LAN
- Bridges can not configure their ports.
  - Forwarding state or blocked state, depending on whether the port is part of the spanning tree
- Root periodically sends configuration messages and bridges forward them over LANs they are responsible for.

Spanning Tree Algorithm Example

- Node B2:
  - Sends (B2, B2, 0)
  - Receives (B1, B1, 0) from B1
  - Sends (B2, B1, 1) “up”
  - Continues the forwarding forever
- Node B1:
  - Will send notifications forever
- Node B7:
  - Sends (B7, B7, 0)
  - Receives (B1, B1, 0) from B1
  - Sends (B7, B1, 1) “up” and “right”
  - Receives (B5, B5, 0) - ignored
  - Receives (B5, B1, 1) - better
  - Continues forwarding the B1 messages forever to the “right”

Ethernet Switches

- Bridges make it possible to increase LAN capacity.
  - Packets are no longer broadcasted - they are only forwarded on selected links
  - Adds a switching flavor to the broadcast LAN
- Ethernet switch is a special case of a bridge: each bridge port is connected to single host.
  - Can make the link full duplex (really simple protocol!)
  - Simplifies the protocol and hardware used (only two stations on the link) – no longer full CSMA/CD
  - Can have different port speeds on the same switch
    - Unlike in a hub, packets can be stored
    - An alternative is to use cut through switching
Ethernet Evolution

**Early Implementations**
- A Local Area Network
- MAC addressing, non-routable
- BUS or Logical Bus topology
- Collision Domain, CSMA/CD
- Bridges and Repeaters for distance/capacity extension
- 1-10 Mbps: coax, twisted pair (10Base-T)

**Current Implementations**
- Switched solution
- Little use for collision domains
- 80% of traffic leaves the LAN
- Servers, routers 10x station speed
- 1/10/100/1000 Mbps, 10Gig coming: Copper, Fiber
- 95% of new LANs are Ethernet

**“Traditional” Topology**
- Hierarchical single tree
- Redundancy for reliability
- Spanning tree (or variant) for loop-free-ness

**Typical Campus Topology**

**Outline**
- Bridging and switching
- Something different
  - Data center networks
  - Software defined networks
Cloud Computing

- Elastic resources
  - Expand and contract resources
  - Pay-per-use
  - Infrastructure on demand
- Multi-tenancy
  - Multiple independent users
  - Security and resource isolation
  - Amortize the cost of the (shared) infrastructure
- Flexibility service management
  - Resiliency: isolate failure of servers and storage
  - Workload movement: move work to other locations

Multi-Tier Applications

- Applications consist of tasks
  - Many separate components
  - Running on different machines
- Commodity computers
  - Many general-purpose computers
  - Not one big mainframe
  - Easier scaling

Some Differences Between DC Networking and Internet/WAN

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Internet/WAN</th>
<th>Commodity Datacenter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latencies</td>
<td>Milliseconds to Seconds</td>
<td>Microseconds</td>
</tr>
<tr>
<td>Bandwidths</td>
<td>Kilobits to Megabits/s</td>
<td>Gigabits to 10's of Gbits/s</td>
</tr>
<tr>
<td>Causes of loss</td>
<td>Congestion, link errors, ...</td>
<td>Congestion</td>
</tr>
<tr>
<td>Administration</td>
<td>Distributed</td>
<td>Central, single domain</td>
</tr>
<tr>
<td>Statistical Multiplexing</td>
<td>Significant</td>
<td>Low - a few flows can dominate link traffic</td>
</tr>
<tr>
<td>Incast: many “fat” flows to same destination</td>
<td>Rare</td>
<td>Frequent, due to synchronized responses</td>
</tr>
</tbody>
</table>

- Historically, DC networks used custom network technologies
  - Low latency, high bandwidth, minimal protocol stack
  - E.g., Myrinet
- Today: leverage commodity ethernet technology - Why?

“Fat Tree” Topology

Capacity?
Reliability?
Fat-Tree Based DC Architecture

- Inter-connect racks (of servers) using a fat-tree topology
- Fat-Tree: a special type of Clos Networks (after C. Clos)
  - K-ary fat tree: three-layer topology (edge, aggregation and core)
  - each pod consists of $(k/2)^2$ servers & 2 layers of k/2 k-port switches
  - each edge switch connects to k/2 servers & k/2 agg. switches
  - each agg. switch connects to k/2 edge & k/2 core switches
  - $(k/2)^2$ core switches: each connects to k pods

Fat-Tree Based Topology ...

- Why Fat-Tree?
  - Fat tree has identical bandwidth at any bisections
  - Each layer has the same aggregated bandwidth
  - Can be built using cheap devices with uniform capacity
  - Each port supports same speed as end host
  - All devices can transmit at line speed if packets are distributed uniform along available paths

Top-of-Rack Architecture

- Rack of servers
  - Commodity servers
  - And top-of-rack switch
- Modular design
  - Preconfigured racks
  - Power, network, and storage cabling
  - Aggregate to the next level

LAN Deployment Comparison

- Traditional LAN deployment
  - Distributed management – see lecture
- Common practice today: VLAN
  - Same but physical LAN is partitioned into “Virtual LANs” – covered later in the course
- Data center networks: requirements are radically different!
  - High traffic load – low latency and loss requirements
  - Need control over bandwidth and strict flow isolation
  - Lots of policy requirements
Traditional Management: Distributed

“Vertically-integrated” Switch / router

Policy (from operator)

Algorithms

Inter-switch protocols
• STP
• VTP
• MPLS
• ...

Fwd. table

Forwarding

New Approach - SDN Software-Defined Networking

SDN Controller

Policy (from operator)

Algorithms

Open interface

Software-defined Switch / router

SW<->SW, SW <--> Controller

Fwd. table

Forwarding

SDN Discussion

• Logically centralized “controller” runs control and management “applications”
  • Uses interface to fill in forwarding table on all switches
  • Can enforce paths and policies for flows (src-dst pair)
• Motivation: easier to manage and centralized algorithms can be “smarter” than distributed ones
  • Customization of decisions per flow, server, ...
• Why now?
  • Need for more sophisticated policies (perf., security, ..)
  • Much better technology, e.g., speed, reliability, ..
  • Currently mostly limited to DC networks

Things to Remember

• Trends from CSMA networks to switched networks
  • Need for more capacity
  • Low cost and higher line rate
• Emphasis on low configuration and management complexity and cost
  • Fully distributed path selection
  • Trend towards centralization, e.g., SDN in DC (and in wireless – later in course)
  • Richer policies – easier to manage centrally
Next Lecture

- Discussion of two “early Internet” papers:
  - “The design philosophy of the DARPA Internet Protocols”, David Clark, ACM SIGCOMM 88
  - “End-to-end arguments in system design”, Saltzer, Reed, and Clark, ACM TOCS, 1984

- Start IP layer lectures