Packet Switching

Switching and Forwarding
Bridges and LAN Switches
Cell Switching
Switching Hardware
Where are We?

- Directly connected networks
  - Point-to-point connectivity

- Limitations
  - Number of hosts
  - Geographic area

- Next Goal
  - Communication between hosts that are not directly connected
Next Topic: Switching

- **Motivation**
  - Why not just one direct link network?

- **Basic approach**
  - How can we extend the direct link abstraction?

- **Challenges**
  - What problems must we address?

- **Examples**
  - Where are these issues addressed in real networks?

- **Details of the switch**
  - What are the goals in design?
    - How are these goals typically addressed?
After Switching

- **Heterogeneity**
  - Switching allows multiple physical networks
  - But assumes one switching strategy

- **Scale**
  - Direct link networks: $O(100)$ hosts
  - Packet-switched networks: $O(100,000)$ hosts

- **Beyond the basics**
  - Quality of service
  - Congestion and performance analysis
  - Network trends and their importance
How can many hosts communicate?

- Best: use one direct link network
- Naïve approach: full mesh
- Problem: doesn't scale
How can many hosts communicate?

- Slightly better solution: shared media
- But, how to deal with larger networks (more hosts, larger geographic area)?
- How to deal with heterogeneous media? (different physical/MAC technologies?)
Real Solution: Switching

- Most basic function: allow communication between arbitrary endpoints
  - Provide illusion of one (reliable) physical network
  - On demand rather than connecting all pairs in advance
  - Share infrastructure across users (“statistical multiplexing”)
Real Solution: Switching

Telephone exchange in 1893

Telephone operator at Aberdeen proving grounds, ~1940
Real Solution: Switching

- Computer networks: more critical benefits
  - Translates between different link technologies, allowing heterogeneity
  - Isolates different physical networks to improve scalability and modularity
Switches

- Build network from stars
Challenges

Which way should they go?

first challenge: forwarding

New York

Boston
Challenges

second challenge: routing

How do we maintain forwarding information?
Challenges

Who goes first?

New York

Boston

third challenge: contention
Challenges

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

- Routing in 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
Network Layers and Switches

Host

Application
Presentation
Session
Transport
Network
Data Link
Physical

Switch

Application
Presentation
Session
Transport
Network
Data Link
Physical
Network Layers and Switches

switch between different physical layers

Switch

- Network
- Data Link A
- Physical A

Switch

- Network
- Data Link B
- Physical B
Switching task: Translate betw. lower-layer technologies

Router

Network

Data Link A

Physical A

Data Link B

Physical B

interface #1
(wireless 802.11)

interface #2
(wired ethernet)
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
Router/switch Architecture

Protocol daemon

RIB 12.0.0.0/8 → IF 1

FIB 12.0.0.0/8 → IF 1

Update 12.0.0.0/8

Interconnection network

12.10.2.3 Data packet
Switching and Forwarding

Forwarding

- The task of specifying an appropriate output port for a packet
  - Datagram
  - Virtual Circuit Switching
  - Source Routing
- Each packet contains enough information for a switch to determine the correct output port
- Later
  - Building forwarding tables – routing.

Packet Header

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

Destination is E, so I should forward out interface #2

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

- Connectionless
  - Each packet travels independently
- Switch
  - Translates global address to output port
  - Maintains table of translations
- Used in traditional data networks
  - i.e., Internet
Forwarding Table

Each switch maintains a forwarding table that translates a host name to an output port.

<table>
<thead>
<tr>
<th></th>
<th>α’ s Table</th>
<th>β’ s Table</th>
<th>γ’ s Table</th>
<th>δ’ s Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>E</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>F</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>G</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>0</td>
</tr>
</tbody>
</table>
Each switch maintains a forwarding table that translates a host name to an output port.
Forwarding with Datagrams

What happens to the last packet?
Forwarding with Datagrams

- Analogous to following signs

- Advantages
  - Can send packets immediately
  - All necessary information in packet header
  - Can route around failures dynamically

- Disadvantages
  - Header requires full unique address
  - Might not be possible to deliver packet
  - Successive packets may not follow the same route
  - Global address to path translations requires significant storage
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 Circuit Switching

- Data from interface #0 should be sent to interface #1
- Data from interface #3 should be sent to interface #2

Circuit establishment
Only for A \(\rightarrow E\)
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
Virtual Circuits: The Circuit

- Logical circuit between source and destination
  - Packets from different VCs multiplex on a link
  - Used in ATM, MPLS, Intserv
Virtual Circuits: The Circuit

- Identified by 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

Host A
12:1

Host B
12:2

Host C
12:1

Host D
12:1

Host E
12:1

Host F
12:1

Host G
12:1

DATA
Virtual Circuits: 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
Virtual Circuits: Label Swapping

- Solution: Label swapping / label switching (used, e.g., 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

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
- Set up is the job of a signaling protocol
  - Switched Virtual Circuits (SVC)
  - Or static “permanent” configuration (less common): Permanent Virtual Circuits (PVC)

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

- 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
    - Increment assignment counter
  - 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

Host A:
- Setup msg
  - Dst=E, VCI=0

Host B:
- Port IN: 3
- VCI IN: 0
- Port OUT: 2
- VCI OUT: 0
- Valid?: no

Host C:
- Port IN: 3
- VCI IN: 0
- Port OUT: 2
- VCI OUT: 0
- Valid?: no

Host D:
- Port IN: 3
- VCI IN: 0
- Port OUT: 2
- VCI OUT: 0
- Valid?: no

Host E:
- Port IN: 3
- VCI IN: 0
- Port OUT: 2
- VCI OUT: 0
- Valid?: no

Host F:
- Port IN: 3
- VCI IN: 0
- Port OUT: 2
- VCI OUT: 0
- Valid?: no

Host G:
- Port IN: 3
- VCI IN: 0
- Port OUT: 2
- VCI OUT: 0
- Valid?: no

<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>

<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>
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><strong>yes</strong></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>2</td>
<td>0</td>
<td><strong>yes</strong></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>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td><strong>yes</strong></td>
</tr>
</tbody>
</table>

ACK msg VCI=0
Virtual Circuit G→C

Setup msg Dst=C, VCI

Port IN | VCI IN | Port OUT | VCI OUT | Valid ?
-------|--------|----------|---------|--------
 0      |   0    |     1    |   0     |  yes   
 3      |   0    |     1    |   1     |  no    

Port IN | VCI IN | Port OUT | VCI OUT | Valid ?
-------|--------|----------|---------|--------
 3      |   0    |     2    |   0     |  yes   
 3      |   1    |     1    |   0     |  no    

ACK msg VCI=0

Port IN | VCI IN | Port OUT | VCI OUT | Valid ?
-------|--------|----------|---------|--------
 3      |   0    |     0    |   0     |  yes   
 3      |   0    |     1    |   0     |  no    

© CS 438 Staff, University of Illinois
Forwarding with Virtual Circuits

Set up circuits:
A → E
C → F
G → 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

\[\begin{array}{|c|c|c|c|}
\hline
\text{Port IN} & \text{VCI IN} & \text{Port OUT} & \text{VCI OUT} \\
\hline
\alpha & & & \\
0 & 0 & 1 & 0 \\
3 & 0 & 1 & 1 \\
\beta & & & \\
1 & 0 & 2 & 1 \\
3 & 0 & 2 & 0 \\
3 & 1 & 2 & 2 \\
\hline
\end{array}\]

\[\begin{array}{|c|c|c|c|}
\hline
\text{Port IN} & \text{VCI IN} & \text{Port OUT} & \text{VCI OUT} \\
\hline
\gamma & & & \\
1 & 0 & 3 & 0 \\
\end{array}\]
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
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>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Datagrams</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<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>
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

What happens to the last packet?

A sends: 1 2 1 DATA

C sends: 3 2 3 DATA

B sends: 2 1 DATA

F sends: 0 2 3 DATA

A sends: 1 1 2 1 DATA
Forwarding with Source Routing

- Analogous to following directions

**Advantages**
- Simple switches
- Fast and cheap

**Disadvantages**
- Hosts must know entire topology
- Changes must propagate to all hosts
- Headers might get large
Forwarding Performance

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

- 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
    - $= \text{pps} \times (\text{BitsPerPacket})$
    - $= 100 \times 10^3 \times 64 \times 8$
    - $= 51.2 \times 10^6$
Bridges and LAN Switches
Bridges: Building Extended LAN’s

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

- Problem
  - Scale
    - Larger geographic area (> O(1 km))
    - More hosts (> O(100))
  - Retain LAN-like functionality
Bridges: Building Extended LAN’s

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

- Problem
  - Scale
    - Larger geographic area (> O(1 km))
    - More hosts (> O(100))
  - Retain LAN-like functionality

Solution: bridges

Spring 2017 © CS 438 Staff, University of Illinois
Bridges (LAN Switches)

- Connect two or more LANs at the link layer (traditionally)
  - Extract destination address from the frame
  - Look up the destination in a table
  - Forward the frame to the appropriate LAN segment
    - Or point-to-point link, for higher-speed Ethernet
    - Each segment is its own collision domain

- 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: Misses

- What about frames with unfamiliar destinations?
  - Forward the frame on all interfaces (“flooding”)
  - … except for source interface
- Hopefully, this case won’t happen very often
- When destination replies,
  - Switch learns that node, too
Learning Bridges

- Examples
  - Frame for A received on port 1: do nothing
  - Frame for C received on port 2: forward to port 1
  - Frame for S received on port 2: forward to port 1
Learning Bridges: Forwarding

- Simple algorithm
  - `<receive frame>`
  - index switch table using MAC dest address
  - if entry found for destination
    - if dest on segment from which frame arrived
      - drop frame
    - else forward frame on interface indicated
  - else flood

Problems?

- forward on all but the interface on which the frame arrived
Problem: Loops

- If there is a loop in the extended LAN, a packet could circulate forever

  Side question: Are loops good or bad?
Learning Bridges

Solution: Spanning Tree

- Select which bridges should actively forward
- Create a spanning tree to eliminate unnecessary edges
- Adds robustness
- Complicates learning/forwarding
Learning Bridges

- Spanning tree
  - Sub-graph that covers all vertices but has no cycles
  - Links not in the spanning tree do not forward frames
Example Extended LAN with LOOPS
Spanning Tree Algorithm

- View extended LAN as bipartite graph
  - LAN’s are graph nodes
  - Bridges are also graph nodes
  - Ports are edges connecting LAN’s to bridges

- Spanning tree required
  - Connect all LAN’s
  - Can leave out bridges
Defining a Spanning Tree

Basic Rules

- Bridge with the lowest ID is the root
- For a given bridge
  - A port in the direction of the root bridge is the root port
- For a given LAN
  - The bridge closest to the root (or the bridge with the lowest ID to break ties) is the designated bridge for a LAN
  - The corresponding port is the designated port
- Bridges with no designated ports and ports that are neither a root port nor a designated port are not part of the tree
Spanning Tree Algorithm

- **Root**
- **D** – designated port
- **R** – 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
Using a Spanning Tree: Broadcast and Multicast

- Forward all broadcast/multicast frames
- 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
Algo-rhyme

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 which 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
Distributed Spanning Tree Algorithm

- 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 sends configuration messages
  - A timeout is used to restart the algorithm
Example at bridge B3

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
Robust Spanning Tree Algorithm

- Algorithm must react to failures
  - Failure of the root node
    - Must elect new root, with the next lowest identifier
  - Failure of other switches and links
    - Need to recompute some part of the spanning tree

- Root switch continues sending messages
  - Periodically re-announcing itself as the root (1, 0, 1)
  - Other switches continue forwarding messages

- Detecting failures through timeout (soft state)
  - Switch waits to hear from others
  - Eventually times out and claims to be the root
Bridges: Limitations

- Does not scale
  - Spanning tree algorithm scales linearly
  - Broadcast does not scale

- Virtual LANs (VLAN)
  - An extended LAN that is partitioned into several networks
  - Each network appears separate
  - Limits effect of broadcast
  - Simple to change virtual topology
Bridges: Limitations

- Does not accommodate heterogeneity
  - Networks must have the same address format
    - e.g. Ethernet-to-Ethernet

- Caution
  - Beware of transparency
    - May break assumptions of the point-to-point protocols
      - Frames may get dropped
      - Variable latency
      - Reordering
    - Bridges happen!
Bridging: how good is it?

Advantages of bridges over routers
- Plug-and-play
  - Routers could be too in principle, but typically are not
- Fast filtering and forwarding of frames

Disadvantages of switches over routers
- Topology restricted to a spanning tree
  - Long paths, congestion
- Large networks require large tables
- Broadcast storms can cause the network to collapse
- Can’t accommodate non-Ethernet segments (why?)