Problem: not all networks are directly connected

Limitations of directly connected networks:
- limit on the number of hosts supportable
- limit on the geographic span of the network

Packet Switching

Hongwei Zhang

http://www.cs.wayne.edu/~hzhang

Nature seems ... to reach her ends by long circuitous routes.

--- Rudolph Lotze

Acknowledgement: this lecture is partially based on the slides of Dr. Larry Peterson
Outline

- Switching and Forwarding
- Bridges and Extended LANs
- Cell Switching
- Implementation
- Discussion
Outline

- Switching and Forwarding
- Bridges and Extended LANs
- Cell Switching
- Implementation
- Discussion
Scalable Networks using switches

- **Switch**
  - forwards packets from input port to output port
  - port selected based on address in packet header

- **Advantages**
  - support large numbers of hosts (scalable bandwidth)
  - cover large geographic area (tolerate latency)
Problem statement

- Given a multi-hop network where nodes may not be directly connected, how does a switch decide where (e.g., which output port) to forward each packet?
Source Routing

Host A → Host B
Virtual Circuit Switching

- Explicit connection setup (and tear-down) phase
- Subsequence packets follow same circuit
- Sometimes called *connection-oriented* model
- Analogy: phone call
- Each switch maintains a VC table

Q: how does VC Switching differ from Circuit Switching?
Datagram Switching

- No connection setup phase
- Each packet forwarded independently
- Sometimes called connectionless model
- Analogy: postal system
- Each switch maintains a forwarding (routing) table
Example Tables

- Circuit Table (switch 1, port 2)

<table>
<thead>
<tr>
<th>VC In</th>
<th>VC Out</th>
<th>Port Out</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

- Forwarding Table (switch 1)

<table>
<thead>
<tr>
<th>Address</th>
<th>Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2</td>
</tr>
<tr>
<td>C</td>
<td>3</td>
</tr>
<tr>
<td>F</td>
<td>1</td>
</tr>
<tr>
<td>G</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Virtual Circuit Model

- (+) Connection setup provides an opportunity to reserve resources.

- (+) While the connection request contains the full address for destination, each data packet contains only a small identifier, making the per-packet header overhead small.

- (-) Typically wait full RTT for connection setup before sending first data packet.

- (-) If a switch or a link in a connection fails, the connection is broken and a new one needs to be established.
VC model: systems

- X.25
  - Buffer allocated during connection setup phase
    - Circuit is rejected if a node does not have enough buffers at the time of connection setup
  - *Hop-by-hop* flow control: “sliding window protocol + flow control” between each pair of nodes along a circuit

- Frame Relay:
  - Permanent Virtual Circuit (PVC) => virtual private networks (VPNs)
  - Basic QoS and congestion avoidance, but rather lightweight compared to X.25 and ATM

- Asynchronous Transfer Mode (ATM)

- GMPLS: Generalized Multi-Protocol Label Switching
Datagram Model

- (+) There is no round trip delay waiting for connection setup; a host can send data as soon as it is ready.
- (+) Since packets are treated independently, it is possible to route around link and node failures
- (-) Source host has no way of knowing if the network is capable of delivering a packet or if the destination host is even up.
- (-) Since every packet must carry the full address of the destination, the overhead per packet is higher than for the connection-oriented model.
Outline

- Switching and Forwarding
- Bridges and Extended LANs
- Cell Switching
- Implementation
- Discussion
Bridges and Extended LANs

- LANs have physical limitations (e.g., 4 repeaters, 2500m)

- Connect two or more LANs with a *bridge*
  - accept and forward strategy
  - level 2 connection (does not add packet header)
Learning Bridges

- Do not forward when unnecessary
- Maintain forwarding table

<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>

- Learn table entries based on *source address*
- Table is an optimization; need not be complete
  - Will learn a route to a node only *after* seeing a packet from the node
- Always forward broadcast frames
Spanning Tree Algorithm

- Problem: broadcast storm as a result of loops (because of the initial broadcast before learning the route to a node)
  - Q: how?

- Bridges run a distributed spanning tree algorithm
  - select which bridges actively forward
  - developed by Radia Perlman (DEC)
  - now IEEE 802.1 specification
Algorithm Overview

- Each bridge has unique id (e.g., B1, B2, B3)

- Basic approaches:
  - Bridge with the smallest id becomes the root
  - Each bridge computes the shortest path to the root
    - break ties randomly (e.g., to enable freedom of load-balancing algorithms)
  - On each LAN, the bridge closest to root becomes the designated bridge
    - use id to break ties (Q: why not randomly?)

- Each bridge forwards frames over each LAN for which it is the designated bridge

Q: distributed algorithm?
Algorithm Details

- Bridges exchange configuration messages
  - id for what the sending bridge believes to be root bridge
  - distance (hops) from sending bridge to root bridge
  - id for bridge sending the message

- Initially, each bridge believes it is the root

- Each bridge records current best configuration message for each port
  - Best: smallest $<\text{root id}, \text{hop length}, \text{own node id}>$
Algorithm Detail (contd.)

- When learn not root, stop generating config messages
  - in steady state, only root generates configuration messages

- When learn not designated bridge for a port/LAN, stop forwarding config. messages over that port/LAN
  - in steady state, only designated bridges forward config messages

- Root continues to periodically send config messages

- If any bridge does not receive config message after a period of time, it starts generating config messages claiming to be the root
Bridge $i$

Variables: $\text{root} . i$, $\text{dist} . i$, $p . i$;
$\text{connected} . \text{bridges} . m$: each port $m$;
$\text{designated} . m$: each port $m$;

Actions:

- Initialization: $\text{root} . i$, $\text{dist} . i := i$, 0;
- (Boot up) $p . i := i$;

[ ]

$\text{RCV} < k, d, j, i > \rightarrow$

\begin{align*}
&\text{if } k < \text{root} . i \rightarrow \\
&\quad \text{root} . i := k; \\
&\quad \text{dist} . i := d \cdot k_j + 1; \\
&\quad p . i := j; \\
&\end{align*}

\begin{align*}
&\text{if } k = \text{root} . i \land d \cdot k_j < \text{dist} . i - 1 \rightarrow \\
&\quad \text{dist} . i := d \cdot k_j + 1; \\
&\quad p . i := j; \\
&\end{align*}

- Update $\text{connected} . \text{bridges} . m$, $\text{designated} . m$;

- $\text{if } \text{designated} . j = \text{True}, \forall j : m \rightarrow$

\begin{align*}
&\text{broadcast } < \text{root} . i, \text{dist} . i, i > \text{ via } \\
&\quad \text{Port } j; \\
&\end{align*}

[ ]

- $\text{if } \text{root} . i = i \rightarrow \text{broadcast } < \text{root} . i, \text{dist} . i, i > \text{ to all ports}$

\[\]
Example

- For simplicity, assume synchrony across nodes (which is usually not the case in practice)

- Focus on how B3 behaves

  - Step 0: B1, B2, B5, B3 sets themselves as roots
  - Step 1: B2 and B5 set B1 as roots; B3 sets B2 as root
  - Step 2: B3 sets B1 as its root, and stops forwarding messages on both interfaces since B3 is not the designated bridge on LANs A and C
Example (contd.)

- **Q:** What if B2 fail-stops?
Example (contd.)

- Q: What if B3 has another LAN attached?
Broadcast and Multicast

- Broadcast: simply forwarded to all the output ports specified in the spanning tree

- Multicast: same as broadcast; destination hosts decide whether to accept the received frames
  - current practice

- Can we do better for multicast?
  - Learn when no group members downstream
  - Accomplished by having each member of group G send a frame to bridge multicast address with G in source field
  - Proposed, but not yet implemented as of today
Limitations of Bridges

- Do not scale
  - spanning tree algorithm does not scale: single level rather than hierarchical
  - broadcast does not scale

- Do not accommodate heterogeneity
  - Bridges use “address” field from frame headers, thus they only support networks with the same format for addresses
    - E.g., Ethernet and ATM cannot directly communicate via basic bridges
Outline

- Switching and Forwarding
- Bridges and Extended LANs
- Cell Switching
- Implementation
- Discussion
Cell Switching (ATM)

- 1980s and early 1990s; embraced by telephone industry
- Used in both WAN and LAN settings
- Specified by ATM forum

- Connection-oriented (virtual circuit switching), packet-switched network
  - Signaling (connection setup) Protocol: Q.2931

- Packets are called *cells*: 5-byte header + 48-byte payload

- Commonly transmitted over SONET
  - other physical layers possible
Variable vs. Fixed-Length Packets

- No Optimal Length => variable-length packet
  - if small: high header-to-data overhead
  - if large: low utilization for small messages

- Fixed-Length Easier to Switch in Hardware
  - Simpler to implement
  - Enables parallelism (since length is known and fixed)
Big vs. Small Packets

- Small Improves Queue behavior
  - finer-grained preemption point for scheduling link
    - maximum packet = 4KB
    - link speed = 100Mbps
    - transmission time = 4096 x 8/100 = 327.68us
    - high priority packet may sit in the queue 327.68us

  - in contrast, 53 x 8/100 = 4.24us for ATM

- near cut-through behavior
  - two 4KB packets arrive at same time
  - link idle for 327.68us while both arrive
    - Because the switch must wait to receive the whole first packet before starting transmitting it
  - at end of 327.68us, still have 8KB to transmit

  - in contrast, can transmit first cell after 4.24us
  - at end of 327.68us, just over 4KB left in queue
Big vs. Small (contd.)

- Small Reduces Latency (for voice)
  - voice digitally encoded at 64KBps (8-bit samples at 8KHz)
  - need full cell’s worth of samples before sending cell
  - example: 1000-byte cells implies 125ms per cell (too long)
    - smaller latency implies no need for echo cancellers (since a very small latency feels like “0 latency”)

- ATM Compromise: 48 bytes
  - US: would like it to be 64 bytes (has echo canceller, thus can afford large packets to reduce header-to-payload ratio
  - Europe: advocates 32 bytes (no echo canceller, thus need small packet)
    - Compromise: 48 = (32+64)/2
**Cell Format**

- **User-Network Interface (UNI): host-to-switch format**
  
<table>
<thead>
<tr>
<th>4</th>
<th>8</th>
<th>16</th>
<th>3</th>
<th>1</th>
<th>8</th>
<th>384 (48 bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GFC</td>
<td>VPI</td>
<td>VCI</td>
<td>Type</td>
<td>CLP</td>
<td>HEC (CRC-8)</td>
<td>Payload</td>
</tr>
</tbody>
</table>

  - GFC: Generic Flow Control (still being defined)
  - **VPI**: Virtual Path Identifier
  - **VCI**: Virtual Circuit Identifier
  - **Type**: management, congestion control, AAL5 (later)
  - CLP: Cell Loss Priority
  - HEC: Header Error Check (CRC-8)

- **Network-Network Interface (NNI): switch-to-switch format**
  - GFC becomes part of VPI field
Segmentation and Reassembly

- Accomplished by ATM Adaptation Layer (AAL)

- Two sublayers:
  - Higher layer: Convergence Sublayer (CS)
    - responsible for packaging the higher layer PDU with *additional information required for the adaptation necessary* for specific service types (e.g., bit rate, connection-oriented or connectionless)
  - Lower layer: Segmentation and Reassembly (SAR)
Different types of AALs

- AAL 1 and 2 designed for applications that need guaranteed rate (e.g., voice, video)
- AAL 3/4 designed for data packet
- AAL 5 is an alternative standard for data packet
**AAL 3/4**

- **Convergence Sublayer Protocol Data Unit (CS-PDU)**

- **CPI**: common part indicator (version field); not used yet
- **Btag/Etag**: beginning and ending tag (deal with cell corruption)
- **BASize**: hint on amount of buffer space to allocate
- **Length**: size of whole PDU
**SAR: Cell Format --- payload**

<table>
<thead>
<tr>
<th>ATM header</th>
<th>2</th>
<th>4</th>
<th>10</th>
<th>352 (44 bytes)</th>
<th>6</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>SEQ</td>
<td>MID</td>
<td>Payload</td>
<td>Length</td>
<td>CRC-10</td>
<td></td>
</tr>
</tbody>
</table>

- **Type**
  - BOM: beginning of message
  - COM: continuation of message
  - EOM end of message
- **SEQ**: sequence number
- **MID**: multiplexing identifier
- **Length**: number of bytes of payload (from CS-PDU) in this cell; in bytes
Encapsulation & segmentation for AAL3/4
AAL5

- Simplifies the format of AAL ¾

- CS-PDU Format

  - Pad, so trailer always falls at end of ATM cell
  - Len: size of PDU (data only); in bytes
  - CRC-32 (detects missing or misordered cells)

- SAR: Cell Format --- payload
  - end-of-PDU bit in Type field of ATM header

- Compared with AAL 3/4, AAL 5 does not provide an additional level of multiplexing onto one virtual circuit (which was achieved via MID in AAL ¾)
Encapsulation & segmentation for AAL 5

- User data
- CS-PDU trailer
- ATM header
- Cell payload
- Padding

48 bytes 48 bytes 48 bytes
Virtual path

- Two-level hierarchy of virtual connections
  - Virtual path (VP)
  - Virtual circuit (VC)

- Switches in public network only maintains state about VPs, which is much fewer than the number of VCs
  - Thus, improves systems scalability
Outline

- Switching and Forwarding
- Bridges and Extended LANs
- Cell Switching

- Implementation
- Discussion
A workstation used as a packet switch

- Flexible in trying out different routing/switching strategies
  - Experimental systems

- (-) high switching overhead, and thus potentially low throughput
A $4 \times 4$ switch

- Input ports: switching logic control (e.g., which output ports to forward packets); usually do not buffer packets
- Switch fabric: forward packets from input ports to output ports; may have internal buffer space
- Output ports: buffering
Switch fabrics

- Shared bus
- Shared memory
- Crossbar
Switch fabrics (contd.)

- Self-routing fabric
  - Banyan Network: switch elements in 1\textsuperscript{st} column looks at the most significant bit of output port number: 0 -> route packet to the top; 1 -> route to bottom ...
  - Batcher-Banyan Switch design: Batcher network first sorts packets (for parallel tx.), then the packets are sent to Banyan network
Outline

- Switching and Forwarding
- Bridges and Extended LANs
- Cell Switching
- Implementation
- Discussion
Further reading

- **Spanning tree algorithm**

- **Cell switching**

- **Switch design (e.g., impact of correlated traffic)**
Further reading (contd.)

- Traffic modeling
Summary

- Switching and Forwarding
- Bridges and Extended LANs
- Cell Switching
- Implementation
Assignment - Chapter 3

- Exercise#2
  - Exercise 1
    - Hint: VCI should be unique for each (bidirectional) link
  - Exercises 15 and 17
  - Exercise 21
    - Hint: for 21 (a), you can regard B1 as a simple repeater that would rebroadcast whatever messages it receive but does not generate any new message.

- TinyExam#2