Bridges and LAN Switches

- Bridge accepts LAN frames on one port, outputs them on another.

- Optimization: only forward appropriate frames

- Learning bridges
  - watch incoming *source* address S at port number P
  - add entry to forward address S to port P
  - if no entry, broadcast to all ports
Problem: Cycles (Loops)

- Frame gets rebroadcast forever
- Could avoid by construction, BUT:
  - Hard, especially management
  - Often want redundancy
- Solution:
  - Restrict active ports to a Spanning Tree
  - Basic design by Radia Perlman of Digital
  - 802.1 specification of LAN Bridges is based on this algorithm
Spanning Trees (Abstractly)

- Given a connected graph G
- A *spanning tree* is an acyclic, connected subgraph of G that contains all the nodes.
Extended LAN with Loops

A

C

E

G

J

K

B3

B2

B5

B1

B6

B4

B7
Spanning tree (ports disabled)
Spanning Tree Algorithm (Abstractly)

- Pick a root node
- Compute shortest paths to root
- Need to break ties
Distributed Spanning Tree Algorithm

• Messages contain
  – Sender ID
  – Belief about root ID
  – Distance to root (in hops)

• Elect root for family of LAN’s and designated bridge for each LAN.

• Initially: each bridge considers itself root.

• Update and forward protocol.
Spanning Tree Algorithm, ctd

• A configuration message is better if
  – provides root with smaller ID
  – provides root with equal ID but shorter distance to root
  – provides equal root ID and distance to root but sending ID has smaller ID

• Only store & forward better information
  – Add 1 to #hops of forwarded messages

• Only root bridge sends configuration information at the end.

• Time outs to recover from failure.
Limitations of Bridges

• Scaling
  – Connections on order of dozens
  – Spanning tree algorithm scales linearly
  – Transparency incomplete
    • Congestion can be visible to higher protocol layers
    • Latency can be larger and more variable

• Heterogeneity
  – Limited to compatible (similarly addressed) link layers
Internet Protocol (IP)

- Terminology
- Service model
- Addresses
- Forwarding
- ARP
- ICMP
Internet Protocol Interoperability

Overlays (running at hosts)

Virtual Network Infrastructure (runs globally)

Networks (run locally)
Internetworks

- FDDI Token Ring
- Ethernet
- Point-to-Point Link (e.g., ISDN)
Example of protocol layers used to transmit from H1 to H8 in network shown on previous slide.
Service Model

• Choose minimal service model
  – All nets can implement
  – “Tin cans and a string” extremum

• Features:
  – Best-effort datagram delivery
  – Reliability, etc. as overlays
  – Packet format
**IPv4 Packet Format**

<table>
<thead>
<tr>
<th>0</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>19</th>
<th>31</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>Hlen</td>
<td>TOS</td>
<td>Length</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ident</td>
<td>Flags</td>
<td>Offset</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TTL</td>
<td>Protocol</td>
<td>Checksum</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SourceAddr</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DestinationAddr</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Options (variable length)</td>
<td>Pad</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

DATA
Fields of IPv4 Header

• Version
  – Version of IP, example header is IPv4
  – First field so easy to implement case statement

• Hlen
  – Header length, in 32-bit \textit{words}

• TOS
  – Type of Service (rarely used)
  – Priorities, delay, throughput, reliability

• Length
  – Length of datagram, in \textit{bytes}
  – 16 bits, hence max. of 65,536 bytes

• Fields for \textit{fragmentation} and \textit{reassembly}
  – Identifier
  – Flags
  – Offset
Header fields, continued

• TTL
  – Time to live (in reality, hop count)
  – 64 is the current default (128 also used)

• Protocol
  – e.g., TCP (6), UDP(17), etc.

• Checksum
  – Checksum of header (not CRC)
  – If header fails checksum, discard the whole packet

• SourceAddr, DestinationAddr
  – 32 bit IP addresses - global, IP-defined

• Options
  – length can be computed using Hlen
IP Datagram Delivery

• Every IP packet (datagram) contains the destination IP address
• The network part of the address uniquely identifies a single network that is part of the larger Internet.
• All hosts and routers that share the same network part of their address are connected to the same physical network.
• Routers can exchange packets on any network they’re attached to.
IP addresses

- Hierarchical, not flat as in Ethernet

- Written as four decimal numbers separated by dots: 158.130.14.2
## Network Classes

<table>
<thead>
<tr>
<th>Class</th>
<th># of nets</th>
<th># of hosts per net</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>126</td>
<td>~16 million</td>
</tr>
<tr>
<td>B</td>
<td>8192</td>
<td>65534</td>
</tr>
<tr>
<td>C</td>
<td>~2 million</td>
<td>254</td>
</tr>
</tbody>
</table>
IP Forwarding algorithm

- If (Network # dest == Network # interface) then deliver to destination over interface
- else if (Network # dest in forwarding table) deliver packet to NextHop router
- else deliver packet to default router

- Forwarding tables
  - Contain (Network #, NextHop) pairs
  - Additional information
  - Built by routing protocol
Fragmentation and Reassembly

• Networks differ on maximum packet size
  – Fragment packets into pieces
  – Each fragment is itself a complete datagram
  – Receiving host reassembles them

• *Maximum Transmission Unit* (MTU)
  – Path MTU is min MTU for path
  – Sender typically sends at MTU of first hop

• Frag. And Reassembly: bad for performance
  – One missing fragment means entire packet is discarded
  – Fragmentation uses up resources in the routers
MTU = 1500
MTU = 4500
FDDI Token Ring

MTU = 512

Point-to-Point Link (e.g., ISDN)

ETH IP (1400)  FDDI IP (1400)  P2P IP (512)  ETH IP (512)

P2P IP (512)  ETH IP (376)  P2P IP (376)  ETH IP (376)
Packet Fragmentation

Unfragmented packet

- start of header
- Ident = x 0 Offset = 0
- rest of header
- 1400 bytes data

Fragmented packet

- start of header
- Ident = x 1 Offset = 0
- rest of header
- 512 bytes data

- start of header
- Ident = x 1 Offset = 64
- rest of header
- 512 bytes data

- start of header
- Ident = x 0 Offset = 128
- rest of header
- 376 bytes data

Offset x 8 = # bytes

More to come flag