The Problem

- We know how to build (broadcast) LANs
- Want to connect several LANs together to overcome scaling limits
  - Recall: speed of light limits on efficiency of MAC protocols
- Goal: make a set of interconnected LANs "look like" a single, large LAN
  - Each station uses the same protocol it normally would to transmit to any station on the extended LAN
    - But the LANs may be different technologies (e.g. Ethernet, WiFi)
  - Use standard IEEE 802 addresses to identify destinations
Solution Approach

• We get to design special boxes (ISs) to interconnect LANs
  – Call these *bridges*

• Each *bridge* acts like a station on each LAN to which it is connected
Solution Approach

• We get to design special boxes (ISs) to interconnect LANs
  – Call these *bridges*

• Each *bridge* acts like a station on each LAN to which it is connected
Learning Bridge

• Bridge keeps a table mapping addresses to interfaces
• Bridge looks at source addresses of received packets to populate table
  – Packet with source B received on interface 0 indicates that B is in that direction

<table>
<thead>
<tr>
<th>Addr</th>
<th>i/f</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>0</td>
</tr>
</tbody>
</table>

Problem: Where is A?
Learning Bridge

• If the destination address is not in the table
  – Forward out all interfaces other than the incoming one
• Reply (if any) reveals location of destination
Learning Bridge

• If the destination address is not in the table
  – Forward out all interfaces other than the incoming one
• Reply (if any) reveals location of destination

<table>
<thead>
<tr>
<th>Addr</th>
<th>i/f</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>0</td>
</tr>
<tr>
<td>A</td>
<td>2</td>
</tr>
</tbody>
</table>

Diagram:
- Nodes labeled B and A connected to interfaces labeled 0, 1, and 2.
Bridge Processing

For each received frame:

1. Remember the interface it arrived on
2. Look up the source address in the table
   • If not present, add (source addr, incoming i/f) to table
3. Look up the destination address in the table
   • If found: transmit frame on the indicated interface, if different from the incoming i/f
   • If not found: flood, i.e. transmit on all interfaces except incoming

   Steps 2 and 3 can be done in parallel

• Works when paths include multiple bridges
• Content-addressable memory (CAM) is ideal for implementing this processing
Advantages of Bridging

- Communication on different LANs can occur in parallel
- Any station can reach any other station as if they were connected to the same LAN
- Heterogenous LANs can be accommodated
- Minimal administrative overhead
  - Bridges populate tables automatically
  - No additional address assignment required
Issues

• The learning bridge algorithm works provided there are no cycles in the network
  – The interconnected network topology is constrained to be a tree
  – Only one path between any pair of nodes
  – If this constraint is violated, broadcast frames circulate forever!

• Drawbacks of the tree topology constraint:
  – Any bridge failure partitions the network
  – Capacity limited by minimum-rate LAN
  – Heavy traffic on startup (when no addresses are known)
Broadcast Loops
Robustness Solution: Spanning Tree Protocol

- Allow arbitrary topologies
  - Including cycles of bridges and LANs
- Bridges collaborate to construct a spanning tree (overlay) on the network
  - Before any packets can be flooded
  - Each bridge determines which of its interfaces are part of the tree
  - Interfaces not in the tree are not used in the forwarding algorithm
  - All interfaces participate in the spanning tree algorithm
- When a partition occurs, protocol adjusts the spanning tree
Robustness Solution: Spanning Tree Protocol
Robustness Solution: Spanning Tree Protocol

No traffic forwarded over these interfaces
Spanning Tree Construction

1. Elect a **root bridge**
   - Bridge with lowest ID

2. Each bridge determines its shortest path to root
   - Interface closest to the root is always part of the tree

3. Elect a "designated bridge" for each LAN
   - Bridge with shortest path to root
   - Break ties using lowest ID

4. Any interface connecting a bridge to a LAN for which it is designated bridge is in the tree; any others are not
Spanning Tree Protocol

- Each bridge maintains two global state variables
  - Root ID
  - Distance to Root

and one per-interface variable:
  - designated bridge on this LAN

- Each bridge broadcasts messages containing triples:
  (Who I think is root, distance to that root, my ID)

- Triples are lexicographically ordered
  - \((2,7,0) < (3,4,5)\)
  - \((2,0,5) < (2,1,5)\)
  - \((0,0,0) < (0,0,2)\)
Spanning Tree Protocol

• Each bridge initially proclaims itself as root
• Each bridge updates its state variables based on the smallest triples it sees
  
  Root ID := Root ID from smallest triple yet seen
  Distance to Root := Distance from smallest triple + 1  
  (unless smallest triple is that bridge's)
  Designated Bridge on LAN := sender of smallest triple seen on that LAN
• Bridges turn off interfaces not in the tree
Spanning Tree Protocol Example

- Root: 0
  - Distance: 0
  - Des. a: 0
  - Des. b: 0

- Root: 1
  - Distance: 0
  - Des. a: 1
  - Des. b: 1

- Root: 2
  - Distance: 0
  - Des. a: 2
  - Des. b: 2
  - Des. c: 2

- Root: 3
  - Distance: 0
  - Des. a: 3
  - Des. b: 3

- Root: 4
  - Distance: 0
  - Des. a: 4
  - Des. b: 4

- Root: 5
  - Distance: 0
  - Des. a: 5
  - Des. b: 5
Spanning Tree Protocol Example

Root: 0
Distance: 0
Des. a: 0
Des. b: 0

Root: 1
Distance: 0
Des. a: 1
Des. b: 1

Root: 2
Distance: 0
Des. a: 2
Des. b: 2
Des. c: 2

Root: 3
Distance: 0
Des. a: 3
Des. b: 3

Root: 4
Distance: 0
Des. a: 4
Des. b: 4

Root: 5
Distance: 0
Des. a: 5
Des. b: 5
Spanning Tree Protocol Example

Root: 0
Distance: 0
Des. a: 0
Des. b: 0

Root: 1
Distance: 1
Des. a: 1
Des. b: 0

Root: 2
Distance: 1
Des. a: 0
Des. b: 2

Root: 2
Distance: 0
Des. a: 2
Des. b: 2
Des. c: 2

(1,0,1)
(0,0,0)

(0,0,0)
(1,0,1)

(3,0,3)
(2,0,2)

(2,0,2)
(4,0,4)

(2,0,2)
Spanning Tree Protocol Example

Root: 0
Distance: 0
Des. a: 0
Des. b: 0

Root: 1
Distance: 1
Des. a: 1
Des. b: 2

Root: 2
Distance: 1
Des. a: 2
Des. b: 2
Des. c: 2

(0,0,0) — (0,1,1) — (0,1,5) — (2,0,2) — (2,0,2)

(1,1,3) — (2,1,4) — (2,0,2)

(0,0,0) — (0,1,1) — (1,1,3) — (1,1,3)

(0,0,0) — (0,1,1) — (1,1,3) — (1,1,3)
Spanning Tree Protocol Example
Spanning Tree Protocol Example

Root: 0
Distance: 0
Des. a: 0
Des. b: 0

Root: 0
Distance: 1
Des. a: 0
Des. b: 1

Root: 0
Distance: 2
Des. a: 0
Des. b: 5

Root: 0
Distance: 2
Des. a: 2
Des. b: 5
Des. c: 2

Root: 0
Distance: 3
Des. a: 2
Des. b: 2

(0,0,0)
(0,1,1)
(0,2,2)
(0,1,1)
(0,2,3)
(0,2,4)
(0,2,2)
Spanning Tree Protocol Example

Root: 0
Distance: 0
Des. a: 0
Des. b: 0

Root: 0
Distance: 1
Des. a: 0
Des. b: 1

Root: 0
Distance: 2
Des. a: 1
Des. b: 0

Root: 0
Distance: 3
Des. a: 2
Des. b: 2

Root: 0
Distance: 2
Des. a: 2
Des. b: 5
Des. c: 2

(0,0,0)

(0,1,5)

(0,2,2)

(0,2,3)

(0,3,4)
Tree Maintenance

• Packet forwarding begins only after tree construction has converged

• Bridges periodically broadcast their triples after convergence
  – Detect bridge failure
Spanning Tree Protocol: Repair

Bridge 2 fails
Spanning Tree Protocol: Repair

Root: 0
Distance: 1
Des. a: 0
Des. b: 1

Root: 0
Distance: 0
Des. a: 0
Des. b: 0

Root: 0
Distance: 1
Des. a: 0
Des. b: 1

Root: 0
Distance: 2
Des. a: 1
Des. b: 2

Root: 0
Distance: 3
Des. a: 2
Des. b: 2

Root: 0
Distance: 3
Des. a: 2
Des. b: 2
Spanning Tree Protocol: Repair

Root: 0
Distance: 0
Des. a: 0
Des. b: 0

Root: 0
Distance: 1
Des. a: 0
Des. b: 1

Root: 0
Distance: 2
Des. a: 1
Des. b: 3

Root: 0
Distance: 3
Des. a: 2
Des. b: 4

(0,0,0)
(0,1,5)
(0,2,3)
(0,3,4)
(0,3,4)
Spanning Tree Protocol: Repair

Root: 0
Distance: 0
Des. a: 0
Des. b: 0

Root: 1
Distance: 0
Des. a: 1
Des. b: 1

Root: 2
Distance: 0
Des. a: 2
Des. b: 2
Des. c: 2

Root: 3
Distance: 0
Des. a: 3
Des. b: 3

Root: 4
Distance: 0
Des. a: 4
Des. b: 4

Root: 5
Distance: 0
Des. a: 5
Des. b: 5
Transparent Bridging: Summary

• End Systems: no change
  – Extended LAN looks exactly like regular LAN
• IS's: build forwarding table via local computation
• Topology constrained to be (logically) a tree
  – IS's communicate to construct spanning tree overlay
• Factors limiting scalability
  – Net-wide broadcast when address is not known
  – Tree topology limits capacity between LANs
  – IS tables eventually contain every destination address
Bridging with Source Routes

• Used with IBM's Token Ring Protocol (also in 802.5)
• No constraint on forwarding topology
• Assumptions:
  – Each LAN has a unique 12-bit ID (max 4096 LANS)
  – Each Bridge on a LAN has a 4-bit ID (unique between two LANs)
• Not transparent: Sources (ESs) must place a route description in each packet
  = sequence of alternating LAN and bridge IDs
  – Bridges (ISs) forward packets from LAN to LAN based on route description in packet
**Source Routing Frame Format**

<table>
<thead>
<tr>
<th>Source Addr</th>
<th>RII bit</th>
<th>Dest Addr</th>
<th>RIF</th>
<th>Payload</th>
<th>FCS</th>
</tr>
</thead>
</table>

**Routing Control**
- **LAN ID**
- **Bridge ID**
- **LAN ID**
- **Bridge ID**
- **LAN ID**
- **0**

**Bridge Algorithm for Directed Frames:**
1. Find ID of LAN received on in Source Route
2. If next Bridge ID = my ID:
   - forward onto "next" LAN

Direction bit determines if "next" = left or right

**Note:** Assumptions imply that LAN-Bridge-LAN ID sequences are unique!
How Sources Find Paths

- **Source** broadcasts "All Paths Explorer" frame
- Each **bridge** broadcasts explorer frame to every LAN except the one it was received on
- As frame is forwarded, bridge appends:
  - Its own ID
  - ID of LAN forwarded on to the source route
- **Destination** returns explorer frames to **Source** as directed frames (reversing path via Direction bit)
- Source chooses the "best" path
- Looping prevention: no bridge forwards an explorer frame more than once
Path Discovery via Explorer Frames
Path Discovery via Explorer Frames

A

B

C

D

E

F

Y D/2/E X

Y D/2/F X

Y D/5/C X

Y D/2/E X

1

0

3

4

X
Path Discovery via Explorer Frames
Path Discovery via Explorer Frames

A

0

C

D

5

X

B

Y

D/5/C/0/A/1/B

X

Y

D/2/E/3/B/1/A

X

Y

D/2/F/4/E/3/B

X

Y

D/2/E/3/B

X

E

Y

D/2/F/4/E/3/B

X

F

4

X

2

2

2
Path Discovery via Explorer Frames

Diagram showing path discovery via explorer frames, with nodes and edges illustrating the process.
Path Discovery via Explorer Frames

A → D/2/F/4/E/3/B/1/A/0/C

B

C

D

E

F
Path Discovery via Explorer Frames
Path Discovery via Explorer Frames
Route Selection & Caching

• Source is free to choose any returned route
  – Typically choose the one arriving first!
  – Other possibilities:
    • Each explorer has an MTU field that collects the minimum MTU along the path; choose the path with largest MTU
    • Avoid some LANs known to be heavily loaded
    • Any other policy

• Sources should cache routes for efficiency
  – Re-explore periodically to catch changes
Source Routing Bridging Summary

• End Systems bear the burden:
  – Route discovery
  – Route selection, caching
  – But unused routes consume no overhead!

• Intermediate systems (Bridges): life is simple
  – Completely stateless
  – No "background" protocol between bridges
  – Works with any topology

• Factors limiting scalability
  – Explorer packets create heavy load, waste capacity