CS4450
Computer Networks: Architecture and Protocols

Lecture 7
“Why” Frames
“Why” Switched Ethernet Spanning Tree Protocol

Rachit Agarwal
Announcements

• Prelim on next Monday
  • Open book, open notes
  • If the class is between 8AM and 10:30PM in your local timezone
    • You have to take the prelim during the lecture hours
      • Unless you have gotten permission otherwise
    • If you prefer to take it from home, completely okay
    • If you prefer to take it in the lecture hall, we will be here
  • Otherwise
    • You have 24 hours to take the exam
      • 7:30PM ET on Monday to 7:29PM ET on Tuesday
      • We reserve the right to invite you to an oral exam

• We will send a detailed email about the exam before Monday afternoon
  • Please plan to check your email at least once after noon Monday
Questions?
Goals for Today’s Lecture

• Dive deeper into Link layer design
  • Recap CSMA/CD
  • Why Frames? — Implementing Link Layer on top of Physical Layer
  • Why Switched Ethernet? — Understanding scalability problems

• Introduction to Switched Ethernet: THE Spanning Tree Protocol
Recap from last lecture
Recap: Data Link Layer

• Link Layer: shared *Broadcast* medium
  • **Original design of Link layer protocols**
  • More recent versions have moved to point-to-point
    • We will discuss why so!

• Network Adapters (e.g., NIC — network interface card)
  • The hardware that connects a machine to the network
  • Has a “name” — MAC (Medium access control) address
Recap: Sharing a broadcast channel

• Reservation Based approaches:
  • **Frequency-division multiple access**: reservation in space
  • **Time-division multiple access**: reservation in time
    • Share the same problem: underutilization

• On-demand approaches:
  • **Random access**: uncoordinated sharing
    • Detect collisions, and if needed, recover from collisions
  • **Carrier Sense Multiple Access (CSMA/CD)**
Recap: CSMA (Carrier Sense Multiple Access)

- CSMA: **listen** before transmit
  - If channel sensed idle: transmit entire frame
  - If channel sensed busy: defer transmission

- Does this eliminate all collisions?
  - **No**, because of nonzero propagation delay

- Solution:
  - Include a **Collision Detection (CD) mechanism**
  - If a collision detected
    - Retransmit
    - **When to retransmit?**
Recap: CSMA/CD in one slide!

- **Carrier Sense**: continuously listen to the channel
  - If idle: start transmitting
  - If busy: wait until idle

- **Collision Detection**: listen while transmitting
  - No collision: transmission complete
  - Collision: abort transmission

- **When to retransmit?**: exponential back off
  - After collision, transmit after “waiting time”
  - After k collisions, choose “waiting time” from \( \{0, ..., 2^{k-1}\} \)
  - Exponentially increasing waiting times
  - But also, exponentially larger success probability
Recap: Exponential Back-off: An example

Attempt 1: Suppose a collision happens

Attempt 2: Four possibilities

Success with Probability = 0.5
What is the success probability in attempt 3?

Answer: 0.75
Questions?
Why Frames?

(Layering: Link Layer on top of Physical Layer)
Building Link Layer on top of Physical Layer

• Physical layer sends/receives bits on a link, and forwards to link layer

• View at the destination side physical layer:
  0101011001111110111110111100101000111

• Challenge: how to take the above bits and convert to:
  0101011001111110111110111100101000111

• **Problem**: how does the link layer separate data into correct “chunks”?
  • Chunks belonging to different applications

• Data link layer **interfaces with physical layer using frames**
  • Implemented by the network adaptor
  • **Finally**: What are these frames?
Frames

• Data link layer interfaces with physical layer using frames
  • Sender-side link layer receives data packets from upper layers
  • Converts each packet into one or more ``frames'' such that
    • Destination-side link layer can correctly get back the packets
Identifying start/end of frames: Sentinel Bits

• Delineate frame with special “sentinel” bit pattern
  • e.g., \texttt{01111110} -> start, \texttt{01111111} -> end

\begin{figure}[h]
\centering
\begin{tabular}{|c|c|}
\hline
01111110 & Frame contents & 01111111 \\
\hline
\end{tabular}
\caption{Example frame with sentinel bits.}
\end{figure}

• Problem: what if the sentinel occurs within the frame?

• Solution: bit stuffing
  • Sender always inserts a \texttt{0} after five \texttt{1}s in the frame content
  • Receiver always removes a \texttt{0} appearing after five \texttt{1}s
Example: Sentinel Bits

- Original data, including start/end of frame:
  011111110011111101111101111100101111111

- Sender rule: five 1s -> insert a 0

- After bit stuffing at the sender:
  0111111100111110101111100111111000101111111
When Receiver sees five 1s...

- If next bit is 0, remove it, and begin counting again
  - Because this must be a stuffed bit
  - we can’t be at beginning/end of frame (those had six/seven 1s)

- If next bit is 1 (i.e., we have six 1s) then:
  - If following bit is 0, this is the start of the frame
    - Because the receiver has seen 01111110
  - If following bit is 1, this is the end of the frame
    - Because the receiver has seen 01111111
Example: Sentinel Bits

• Original data, including start/end of frame:
  011111100111111011111101111100101111111

• Sender rule: five 1s -> insert a 0

• After bit stuffing at the sender:
  01111110011111010111110011111000101111111

• Receiver rule: five 1s and next bit 0 -> remove 0
  0111111001111101111101111100101111111
Ethernet “Frames”

- **Preamble:**
  - 7 bytes for clock synchronization
  - 1 byte to indicate start of the frame
- **Names:** 6 + 6 bytes (MAC names/addresses)
- **Protocol type:** 2 bytes, indicating higher layer protocol (e.g., IP)
- **Data payload:** max 1500 bytes, minimum 46 bytes
- **CRC:** 4 bytes for error detection
What about source/destination Addresses?

• Frames are at Layer-2
  • Thus, use Layer-2 addresses (MAC names/addresses)

• MAC name/address
  • Numerical address associated with the network adapter
  • Flat namespace of 6 bytes (e.g., 00-15-C5-49-04-A9 in HEX)
  • Unique, hard coded in the adapter when it is built

• Hierarchical Allocation
  • Blocks: assigned to vendors (e.g., Dell) by IEEE
    • First 24 bits (e.g., 00-15-C5-**-**-**-**)
  • Adapter: assigned by the vendor from its block
    • Last 24 bits
Questions?
Putting it all together
(Traditional Ethernet)
Traditional Ethernet

• (Source) Link layer receives data from the network layer (more later)

• (Source) Link layer divides data into frames
  • How does it know source/destination MAC names?
  • Source name is easy ... destination name is tricky (more later)

• (Source) Link layer passes the frame to physical layer
  • Frames up the frames (using sentinel bits)
  • And broadcasts on the broadcast Ethernet

• (EACH) physical layer regenerates the frame...
  • And sends it up to the (destination) link layer .... If and only if:
    • destination name matches the receiver’s MAC name
    • Or, the destination name is the broadcast address (FF:FF:FF:FF:FF:FF)
Traditional Ethernet

- Ethernet is “plug-n’play”
  - A new host plugs into the Ethernet is good to go
  - No configuration by users or network operators
  - Broadcast as a means of bootstrapping communication
Questions?
WHY Switched Ethernet?
Collision Detection limits Ethernet scalability

- B and D can tell that collision occurred

- However, need restrictions on
  - Minimum frame size
  - Maximum distance
Limits on Traditional Ethernet Scalability

- Latency depends on physical length of link
  - Propagation delay

- Suppose A sends a frame at time 0
  - B sees an idle line at all times before $d$
  - ... so B happily starts transmitting a frame

- B detects a collision at time $d$
  - But A can’t see collision until $2d$
  - A must have a frame size such that transmission time > $2d$
  - Need transmission time > $2 \times$ propagation delay
Limits on Traditional Ethernet Scalability

• Transmission time > 2 * propagation delay

• Requires either very large frames (underutilization) or small scale.
  • Example: consider 100 Mbps Ethernet
  • Suppose minimum frame length: 512 bits (64 bytes)
    • Transmission time = 5.12 μsec
    • Thus, propagation delay < 2.56 μsec
    • Length < 2.56 μsec * speed of light
    • Length < 768m

• Cannot scale beyond ~76.8m for 1Gbps and beyond ~7.68m for 10Gbps
Limits on Traditional Ethernet Scalability

- Transmission time > 2 * propagation delay
- Cannot scale beyond ~76.8m for 1Gbps and beyond ~7.68m for 10Gbps
- This is WHY modern Ethernet networks are “switched”
Evolution

• Ethernet was invented as a broadcast technology
  • Hosts share channel
  • Each “data unit” received by all attached hosts
  • CSMA/CD for access control

• Current Ethernets are “switched”
  • Point-to-point medium between switches;
  • Broadcast (or point-to-point) medium between each host & switch
  • Sharing only when needed (using CSMA/CD)
Questions?
Switched Ethernet
Switched Ethernet

- Enables concurrent communication
  - Host A can talk to C, while B talks to D
  - No collisions -> no need for CSMA, CD
  - No constraints on link lengths or frame size
Routing in Switched Ethernet (Extended LANs)

Bridges relay broadcasts from one LAN to the other
Naïvely Routing in “Extended LANs”: Broadcast storm

Local-Area Network (LAN)

Bridges relay broadcasts from one LAN to the other
How to avoid the Broadcast Storm Problem?

Get rid of the loops!
Lets get back to the graph representation!
Easiest Way to Avoid Loops

• Use a network topology (graph) where loop is impossible!

• Take arbitrary topology (graph)

• Build spanning tree
  • Subgraph that includes all vertices but contains no cycles
  • Links not in the spanning tree are not used in forwarding frames

• Only one path to destinations on spanning trees
  • So don't have to worry about loops!
Consider Graph
Multiple Spanning Trees
Questions?
Following material not included in prelim on 03/08
Spanning Tree Approach

• Take arbitrary topology

• Pick subset of links that form a spanning tree

• Only forward packets on the spanning tree
  • => No loops
  • => No broadcast storm
Spanning Tree Protocol

- Protocol by which bridges construct a spanning tree

- Nice properties
  - Zero configuration (by operators or users)
  - Self healing

- Still used today

- Constraints for backwards compatibility
  - No changes to end-hosts
  - Maintain plug-n-play aspect

- Earlier Ethernet achieved plug-n-play by leveraging a broadcast medium
  - Can we achieve plug-n-play for a switched topology?
Algorithm has Two Aspects...

• Pick a root:
  • Destination to which the shortest paths go
  • Pick the one with the smallest identifier (MAC name/address)

• Compute the shortest paths to the root
  • No shortest path can have a cycle
  • Only keep the links on the shortest path
  • Break ties in some way
    • so we only keep one shortest path from each node

• Ethernet’s spanning tree construction does both with a single algorithm
Breaking Ties

• When there are multiple shortest paths to the root,
  • Choose the path via neighbor switch with the “smallest” identifier

• One could use any tie breaking system
  • This is just an easy one to remember and implement
Constructing a Spanning Tree

• **Messages (Y,d,X)**
  • Proposing Y as the root
  • From node X
  • And advertising a distance d between X and Y

• Switches elect the node with smallest identifier (MAC address) as root
  • Y in messages

• Each switch determines if a link is on its shortest path to the root
  • If not, excludes it from the tree
  • d to Y in the message is used to determine this
Steps in Spanning Tree Protocol

• Messages \((Y,d,X)\)
  - Proposing root \(Y\); from node \(X\); advertising a distance \(d\) to \(Y\)

• Initially each switch proposes itself as the root
  - that is, switch \(X\) announces \((X,0,X)\) to its neighbors

• Each switch \(X\) updates its view
  - Upon receiving message \((Y,d,Z)\) from \(Z\), check \(Y\)'s id
  - If \(Y\)'s id < current root: set root = \(Y\)
  - Set next-hop = \(Z\)

• Switches compute their distance from the root
  - Add 1 to the shortest distance received from a neighbor

• If root changed OR shortest distance to the root changed:
  - send neighbors updated message \((Y,d+1,X)\)
Group Exercise:

Let's run the Spanning Tree Protocol on this example

(assume all links have “distance” 1)
## Round 1

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Graph" /></td>
<td><img src="image" alt="Graph" /></td>
<td><img src="image" alt="Graph" /></td>
<td><img src="image" alt="Graph" /></td>
<td><img src="image" alt="Graph" /></td>
<td><img src="image" alt="Graph" /></td>
<td><img src="image" alt="Graph" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Node</th>
<th>Receive</th>
<th>Send</th>
<th>Next-hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>(1, 0, 1)</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>(2, 0, 2)</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>(3, 0, 3)</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>(4, 0, 4)</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>(5, 0, 5)</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>(6, 0, 6)</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>(7, 0, 7)</td>
<td>7</td>
</tr>
</tbody>
</table>
### Round 2

<table>
<thead>
<tr>
<th>Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (1, 0, 1)</td>
</tr>
<tr>
<td>2 (2, 0, 2)</td>
</tr>
<tr>
<td>3 (3, 0, 3)</td>
</tr>
<tr>
<td>4 (4, 0, 4)</td>
</tr>
<tr>
<td>5 (5, 0, 5)</td>
</tr>
<tr>
<td>6 (6, 0, 6)</td>
</tr>
<tr>
<td>7 (7, 0, 7)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Node</th>
<th>Receive</th>
<th>Send</th>
<th>Next hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(3, 0, 3), (5, 0, 5), (6, 0, 6)</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>(3, 0, 3), (4, 0, 4), (6, 0, 6), (7, 0, 7)</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>(1, 0, 1), (2, 0, 2)</td>
<td>(1, 1, 3)</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>(2, 0, 2), (7, 0, 7)</td>
<td>(2, 1, 4)</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>(1, 0, 1), (6, 0, 6)</td>
<td>(1, 1, 5)</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>(1, 0, 1), (2, 0, 2), (5, 0, 5)</td>
<td>(1, 1, 6)</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>(2, 0, 2), (4, 0, 4)</td>
<td>(2, 1, 7)</td>
<td>2</td>
</tr>
</tbody>
</table>
## Round 3

<table>
<thead>
<tr>
<th>Node</th>
<th>(x, y, z)</th>
<th>Receive</th>
<th>Send</th>
<th>Next hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (1, 0, 1)</td>
<td>(1, 1, 3), (1, 1, 5), (1, 1, 6)</td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1 (2, 0, 2)</td>
<td>(1, 1, 3), (2, 1, 4), (1, 1, 6), (2, 1, 7)</td>
<td>(1, 2, 2)</td>
<td></td>
<td>3 (or 6)</td>
</tr>
<tr>
<td>2 (1, 1, 3)</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>3 (2, 1, 4)</td>
<td>(2, 1, 7)</td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>4 (1, 1, 5)</td>
<td>(1, 1, 6)</td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>5 (1, 1, 6)</td>
<td>(1, 1, 5)</td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>6 (2, 1, 7)</td>
<td>(2,1, 4)</td>
<td></td>
<td></td>
<td>2</td>
</tr>
</tbody>
</table>

The network diagram shows the connections between nodes, with colored dots indicating the nodes and lines representing the connections.
### Round 4

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>Receive</th>
<th>Send</th>
<th>Next hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>(1, 0, 1)</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>(1, 2, 2)</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>(1, 1, 3)</td>
<td>(1, 2, 2)</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>(2, 1, 4)</td>
<td>(1, 2, 2)</td>
<td>(1, 3, 4)</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>(1, 1, 5)</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>(1, 1, 6)</td>
<td>(1, 2, 2)</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>(2, 1, 7)</td>
<td>(1, 2, 2)</td>
<td>(1, 3, 7)</td>
</tr>
</tbody>
</table>
### Round 5

<table>
<thead>
<tr>
<th>Node</th>
<th>Receive</th>
<th>Send</th>
<th>Next hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(1, 0, 1)</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>(1, 2, 2)</td>
<td>(1, 3, 4), (1, 3, 7)</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>(1, 1, 3)</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>(1, 3, 4)</td>
<td>(1, 3, 7)</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>(1, 1, 5)</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>(1, 1, 6)</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>(1, 3, 7)</td>
<td>(1, 3, 4)</td>
<td>2</td>
</tr>
</tbody>
</table>
After Round 5: We have our Spanning Tree

- 3-1
- 5-1
- 6-1
- 2-3
- 4-2
- 7-2
Questions?
Spanning Tree Protocol ++ (incorporating failures)

- Protocol must react to failures
  - Failure of the root node
  - Failure of switches and links

- **Root node sends periodic announcement messages**
  - Few possible implementations, but this is simple to understand
  - Other switches continue forwarding messages

- Detecting failures through timeout (**soft state**)
  - If no word from root, time out and send a \((Y, 0, Y)\) message to all neighbors (in the graph)!

- If multiple messages with a new root received, send message \((Y, d, X)\) to the neighbor sending the message
Suppose link 2-4 fails

- 4 will send (4, 0, 4) to all its neighbors
  - 4 will stop receiving announcement messages from the root
  - Why?
- At some point, 7 will respond with (1, 3, 7)
- 4 will now update to (1, 4, 4) and send update message
- New spanning tree!
Questions?
Where are we?

• You now understand
  • Network sharing (in depth)
  • Delays (in depth)
  • Architectural principles (in depth)
  • End-to-end working of the Internet (at a high-level)

• **Dive deep into Link layer design**
  • Broadcast medium
  • Sharing broadcast medium:
    • Reservations (TDMA, FDMA)
    • On-demand (CSMA/CD)
  • **Why** Frames? — Link Layer on top of Physical Layer
  • **Why** Switched Ethernet? — Understanding scalability problems
  • Spanning Tree Protocol for switched Ethernet
The end of Link Layer ....

And the beginning of network layer :-D

- Built on top of reliable delivery
- Built on top of best-effort forwarding
- Built on top of best-effort routing
- Built on top of physical bit transfer