CS168: Intro to Internet Architecture

Flipped Lecture #2:

- Project 1 Intro + Tips
- What I Found Confusing About Routing
  - Split Horizon vs Poison Reverse vs Route Poisoning
  - Learning Switches
  - STP

Alex Krentsel, Narek Galstyan, and Sarah McClure
Announcements

- Regular in-person lectures starting from this week - no more pre-recorded videos to watch at home.
- Project 1 is released **today at 12:30pm**, find more info on Ed.
Project 1: Routing

You will implement the Distance Vector protocol!

- Use a network simulator (by Murphy McCauley & others at NetSys) to test, validate, and interact with your router implementation.
- The project is split and scored by 10 stages.
- Goal: guide you through the basics–packet forwarding, route advertisement–to the advanced features, e.g., split horizon, poison reverse, route poisoning, and real-world optimizations.

Due: 11:59pm, Oct. 7th. Logistics & OHs will be announced on Ed.

Project TAs:
- Silvery Fu (lead)
- Tenzin Ukyab
- Ken Lien
Project 1 Tips

- Comment your code for each stage - subsequent stages require editing the past code you’ve written.
- Remember that your latency to a dst is your neighbor’s latency to the dst + your latency to your neighbor.
- Make sure you understand each scenario across the 10 stages.
  - This project is as much a thinking exercise as a coding one.
  - Use `self.s_log()` liberally
    - May be worth logging when you make changes to a routing table. `setter` methods are very good for doing this :)  
- The terminal is a live Python interpreter, use it to inspect state, with `print(node.table)`
What I Found Confusing about Routing

(which also happen to be core concepts for the project)
Another Perspective on Distance-Vector Routing Concepts

Every semester, students get confused between Split Horizon, Route Poisoning, and Poison Reverse.

Commons questions:
- What combinations can I use?
- Which one is the best?
- Route Poisoning vs Poison Reverse?
- Split Horizon vs Poison Reverse?

Let’s clear these up…
Routing Motivation

Core problem behind routing:

- Building the rules used for forwarding packets such that network requirements are satisfied.

Reasonable requirements:

1. Full connectivity
2. Minimized latency (or cost)

Aside: Routing happens in the **control plane**. Forwarding happens in the **data plane**.
We’ve discussed Link State (LS) and Distance Vector (DV) protocols as two approaches.

- LS conceptually easier, and not on this project, so let’s focus on DV.

Distance Vector:
- Each node tells its neighbors how far it is from every destination.
- Each node uses the neighbor closest to each destination as its next hop for that dest.
We’ve discussed Link State (LS) and Distance Vector (DV) protocols as two approaches.

- LS conceptually easier, and not on this project, so let’s focus on DV.

Distance Vector:

- **Speaker**: Each node tells its neighbors how far it is from every destination.
- **Listener**: Each node uses the neighbor closest to each destination as its next hop for that dest.
Problems

Problems in the network are caused by invalid network state…

What problems could we run into leading to invalid network state?

- (1) Missing entries
- (2) Loops
  - R1->R2->R1->R2->...->dropped
- (3) Stale network info (entry in a forwarding table that no longer exists in reality)
(1) Missing Entries

What leads to missing entries?

- Advertisement gets dropped
  
  Retransmit advertisements every X seconds
- Destination is unreachable in the topology
  
  ??? - only fix is to get that destination reconnected
What problems could we run into leading to invalid network state?

- (1) Missing entries  
  - Retransmit Advertisements
- (2) Loops
  - R1->R2->R1->R2->...->dropped
- (3) Stale network info (entry in a forwarding table that no longer exists in reality)
What leads to (direct) loops?

- 2 routers must have have “accepted” routes through each other!
What leads to (direct) loops?

- 2 routers must have have “accepted” routes through each other!

At some point, R2’s route TTL expires…

<table>
<thead>
<tr>
<th>Destination</th>
<th>Next Hop, Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Destination</th>
<th>Next Hop, Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>R2, 3</td>
</tr>
</tbody>
</table>
(2) Loops

What leads to (direct) loops?

- 2 routers must have have “accepted” routes through each other!

R3 advertises its route to D to R2

<table>
<thead>
<tr>
<th>Destination</th>
<th>Next Hop, Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>R3,4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Destination</th>
<th>Next Hop, Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>R2,3</td>
</tr>
</tbody>
</table>

(D,3)
**What leads to (direct) loops?**

- 2 routers must have have “accepted” routes through each other!

R2 advertises its route to D to R3

<table>
<thead>
<tr>
<th>Destination</th>
<th>Next Hop, Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>R3,4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Destination</th>
<th>Next Hop, Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>R2,5</td>
</tr>
</tbody>
</table>

---

---

---

---

---
(2) Loops

What leads to (direct) loops?

- 2 routers must have have “accepted” routes through each other!

This looping continues forever, counting up to infinity. This direct looping is what Split Horizon and Poison Reverse attempt to solve.
Split Horizon says:

- If I route through you, I will *not* tell you about my path.

Can’t have a direct loop form if one side never tells the other about the route, right?
Split Horizon

Split Horizon says:

- If I route through you, I will *not* tell you about my path.

Can’t have a direct loop form if one side never tells the other about the route, right?

- Sort of... There are still some tricky edge cases that could lead to a loop forming - you’ll see one in the project.

If we *do* get into a direct loop, Split Horizon breaks out by letting the Time to Live on the updates expire - loop exists for that long!

<table>
<thead>
<tr>
<th>Destination</th>
<th>Next Hop, Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>R3, 2</td>
</tr>
<tr>
<td>D</td>
<td>R2, 3</td>
</tr>
</tbody>
</table>
Poison Reverse is an improvement on Split Horizon:

- If I route through you, I will lie and tell you *my cost to the dest is* \( \infty \).
  - All routers agree not to use paths that have cost infinity.

Equally good at preventing direct loops, but *guarantees that in case of a direct loop, forwarding entries are removed at next advertisement, rather than waiting for the next timeout.*
A Naming Aside…

Why is it called “split horizon”?

- Guesses?

From the original paper introducing the concept in 1975...

“The forming of [an update] may be looked upon as describing the horizon seen from that node…”

- “Omitting [information to particular neighbor] will be called split horizon” (neighbor sees diff. “split”)
- “If the same [update] will be transmitted to all the neighbors, the procedure will be referred to as whole horizon.”
What problems could we run into leading to invalid network state?

- (1) Missing entries  
  - Retransmit Advertisements
- (2) Loops  
  - Split Horizon or Poison Reverse (for direct loops, indirect loops still possible)
    - R1->R2->R1->R2->...->dropped
- (3) Stale network info (entry in a forwarding table that no longer exists in reality)
What leads to stale network info?

- Any change in the network state, while we wait for it to propagate!

Different classes of information take different amounts of time to propagate:

- **Additive Information**: propagates as quickly as the “advertisement frequency”
  - A new router comes up, new link gets added, new host attaches

- **Subtractive Information**: as quickly as the TTL of entries stored in the table
  - A router crashes, a link goes down, hosts detach

$\text{TTL} \gg \text{“advertisement frequency”}$, so how can we improve propagation time for subtractive information…
Route Poisoning

Key Observation we saw with Poison Reverse - you can override a neighbor’s TTL period by sending an advertisement with infinite cost. Can we do the same thing when something goes down to reconverge more quickly?

**Rule:** When you lose a route, instead of just deleting it from your forwarding table, tell all your neighbors your route now has cost $\infty$.
Route Poisoning

Key Observation we saw with Poison Reverse - you can override a neighbor’s TTL period by sending an advertisement with infinite cost. Can we do the same thing when something goes down to reconverge more quickly?

Rule: When you lose a route, instead of just deleting it from your forwarding table, tell all your neighbors your route now has cost $\infty$. 
What problems could we run into leading to invalid network state?

- (1) Missing entries
  - Retransmit Advertisements
- (2) Loops
  - Split Horizon or Poison Reverse (for direct loops, indirect loops still possible)
    - R1->R2->R1->R2->...->dropped
- (3) Stale network info (entry in a forwarding table that no longer exists in reality)
  - Route Poisoning
Frequently Asked Questions

With all of this in mind, back to the common questions:

● What combinations can I use?

● Which one is the best?

● Route Poisoning vs Poison Reverse?
What Combinations Can I Use?

Split horizon and Poison Reverse can’t be used at the same time - one says to say nothing, when the other says to say cost of infinity…

Pick either 1 or 0 for each category of problem:

- **Loop Prevention**: Split Horizon, Poison Reverse
- **Remove Stale Information**: Route Poisoning

So to enumerate all of the possibilities:

1. None
2. Poison Reverse (for Loop Prevention), Nothing (for Removing Stale Info)
3. Split Horizon (for Loop Prevention), Nothing (for Removing Stale Info)
4. Nothing (for Loop Prevention), Route Poisoning (for Removing Stale Info)
5. Poison Reverse (for Loop Prevention), Route Poisoning (for Removing Stale Info)
6. Split Horizon (for Loop Prevention), Route Poisoning (for Removing Stale Info)
Frequently Asked Questions

● What combinations can I use?

{Split Horizon, Poison Reverse, Nothing} x {Route Poisoning, Nothing}

● Which one is the best?

● Route Poisoning vs Poison Reverse?
Which One Is Best?

Slightly incorrect assumption in this question…you can’t compare all 3.

- Split Horizon and Poison Reverse address one problem
- Route Poisoning address a different, orthogonal problem

Better question: which combination is the best?

- Better to use some strategy for each problem than no strategy
  - Remove Stale Info: Route Poisoning
  - Loop Prevention: Poison Reverse
    - Split Horizon and Poison Reverse are equally good at preventing loops, but Poison Reverse will kill loops faster than Split Horizon when they do arise.

Only caveat: Poison Reverse sends a bit more control traffic than split horizon, but it’s a negligible difference.
Frequently Asked Questions

- What combinations can I use?
  
  \{\text{Split Horizon, Poison Reverse, Nothing}\} \times \{\text{Route Poisoning, Nothing}\}

- Which one is the best?

  Poison Reverse + Route Poisoning is the best combo.

- Route Poisoning vs Poison Reverse?
### Route Poisoning vs Poison Reverse

<table>
<thead>
<tr>
<th>“Trigger”</th>
<th>Poison Reverse</th>
<th>Route Poisoning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Receiving an advertisement that causes you to go through a new next hop to some destination.</td>
<td>Losing a route to a destination</td>
<td></td>
</tr>
<tr>
<td>Problem Being Addressed</td>
<td>Loop prevention, and getting out of loops quickly.</td>
<td>Removing stale routing information.</td>
</tr>
<tr>
<td>Mechanism</td>
<td>Send <em>only the neighbor you will use as next hop</em> an update saying your distance to D is ( \infty ).</td>
<td>Send <em>all neighbors</em> an update saying your distance to D is ( \infty ).</td>
</tr>
</tbody>
</table>

The extent of the *similarity* is that they both use the mechanic of sending an advertisement with a cost of infinity to someone. **But for different purposes, and triggered by different reasons!**
Frequently Asked Questions

● What combinations can I use?

{Split Horizon, Poison Reverse, Nothing} x {Route Poisoning, Nothing}

● Which one is the best?

Poison Reverse + Route Poisoning is the best combo.

● Route Poisoning vs Poison Reverse?

Address different problems (though with a similar mechanism - send $\infty$)
Split Horizon vs Poison Reverse
Split Horizon and Poison Reverse

Let’s take a network and focus on a particular host as the dst.

All edges have cost 1.
Split Horizon and Poison Reverse

Best paths to A.
Split Horizon and Poison Reverse

Advertisements from switch 1 with *no split horizon or poison reverse*

Let’s focus on the perspective of switch 1…
Advertisements from switch 1 with **split horizon**
Split Horizon and Poison Reverse

Advertisements from switch 1 with poison reverse
Split Horizon and Poison Reverse

All advertisements from all switches not sent due to split horizon
All poison reverse advertisements from all switches.

Concept check: What does this remind you of? Why?
L2 vs L3 Routing
Routing in L2 vs L3

One more common confusion: we need routing for both L2 and L3, so which of Distance-Vector, Link-State, or Learning Switches do we use in each?

L2: responsible for *local* packet delivery.
L3: response for *global* packet delivery.

(How) is the routing problem different at each layer?
Routing in L2 vs L3

Layer 3
Goals:
● Short paths
● Resilient to failure
● Paths for all destinations in a (potentially large) network

Properties:
● Must be scalable -> get a new address when you join a new network -> allows routing to group based on CIDR prefixes

Solutions:
● Distance-Vector
● Link-State

Layer 2
Goals:
● Short paths
● Resilient to failure
● Paths for all destinations in a local network
● “Plug and play”

Properties:
● Small network
● Operate with low cost

Solutions:
● Learning switches + STP

Addressing gave us some static configuration, which we can use to “seed” DV and LS

With a small network, we can afford to flood and learn paths to avoid configuration
Addressing
Addressing

<table>
<thead>
<tr>
<th>Destination</th>
<th>Next Hop, Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>R5, 4</td>
</tr>
<tr>
<td>B</td>
<td>R5, 4</td>
</tr>
<tr>
<td>C</td>
<td>R5, 3</td>
</tr>
<tr>
<td>D</td>
<td>R5, 4</td>
</tr>
<tr>
<td>E</td>
<td>R5, 2</td>
</tr>
<tr>
<td>F</td>
<td>R5, 2</td>
</tr>
<tr>
<td>H</td>
<td>R7, 2</td>
</tr>
<tr>
<td>I</td>
<td>R7, 4</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
### Addressing

- **1.1.1.0/16**
  - **A**
  - **B**
  - **C**
  - **D**
  - **R4**

- **2.1.1.0/24**
  - **R1**
  - **R2**
  - **R3**
  - **R5**
  - **R6**
  - **R7**
  - **R8**
  - **I**
  - **H**
  - **G**
  - **E**
  - **F**

- **2.1.2.0/24**

### Table: Destination, Next Hop, Cost

<table>
<thead>
<tr>
<th>Destination</th>
<th>Next Hop, Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>R5, 4</td>
</tr>
<tr>
<td>B</td>
<td>R5, 4</td>
</tr>
<tr>
<td>C</td>
<td>R5, 3</td>
</tr>
<tr>
<td>D</td>
<td>R5, 4</td>
</tr>
<tr>
<td>E</td>
<td>R5, 4</td>
</tr>
<tr>
<td>F</td>
<td>R5, 2</td>
</tr>
<tr>
<td>G</td>
<td>R5, 2</td>
</tr>
<tr>
<td>H</td>
<td>R7, 2</td>
</tr>
<tr>
<td>I</td>
<td>R7, 4</td>
</tr>
<tr>
<td>…</td>
<td></td>
</tr>
</tbody>
</table>
Addressing

<table>
<thead>
<tr>
<th>Destination</th>
<th>Next Hop, Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1.1.0/16</td>
<td>R5, 2</td>
</tr>
<tr>
<td>2.1.1.0/24</td>
<td>R7, 1</td>
</tr>
<tr>
<td>2.1.2.0/30</td>
<td>R5, 2</td>
</tr>
<tr>
<td>2.1.2.4/30</td>
<td>G, 1</td>
</tr>
</tbody>
</table>

1.1.1.0/16

2.1.1.0/24

2.1.2.0/24
Break
Learning Switches and Spanning Tree Protocol

Narek Galstyan
Learning Switches and Spanning Tree Protocol

Goal for these slides:

● Walk through examples of Learning switch and STP algorithm runs
● Consider corner cases
● Check our understanding of them from lectures
Goal for these slides:

- Walk through examples of Learning switch and STP algorithm runs
- Consider corner cases
- Check our understanding of them from lectures

Red boxes are questions for you to think about

- And shout out answers!

I will have answers in the following slide in a green box

- Please do answer them before
Learning Switches
Learning Switches

A: S1

S1: Dest, Next

S2: Dest, Next

S3: Dest, Next

S4: Dest, Next

S5: Dest, Next

B: C
Learning Switches

Table here?
Learning Switches

No, hosts use default route
Learning Switches
Who is A’s message for?
Learning Switches

Who is A’s message for?

A to S5?
Who is A’s message for?

Some end-host (say B)
Who is A’s message for?

Some end-host (say B)

Control plane messaging in different protocols:
- Distance-Vector: Routers to neighbor routers
- Link-state: Routers to all routers
- Learning Switches: None
  - Just data plane messages between end-hosts
Learning Switches

A to B

A

S1

Dest | Next
--- | ---
? | ?

S2

Dest | Next
--- | ---

S3

Dest | Next
--- | ---

S4

Dest | Next
--- | ---

S5

Dest | Next
--- | ---

B

C
Learning Switches

A to B
Learning Switches

A to B

S1

Dest | Next
--- | ---
A | A

S2

S3

S4

S5

Dest | Next
--- | ---

B

C
Learning Switches

A to B

<table>
<thead>
<tr>
<th>Dest</th>
<th>Next</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>A</td>
</tr>
</tbody>
</table>

S1

A to B

<table>
<thead>
<tr>
<th>Dest</th>
<th>Next</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>S1</td>
</tr>
</tbody>
</table>

S2

A to B

<table>
<thead>
<tr>
<th>Dest</th>
<th>Next</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>S1</td>
</tr>
</tbody>
</table>

S3

<table>
<thead>
<tr>
<th>Dest</th>
<th>Next</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

S4

<table>
<thead>
<tr>
<th>Dest</th>
<th>Next</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>S1</td>
</tr>
</tbody>
</table>

S5

<table>
<thead>
<tr>
<th>Dest</th>
<th>Next</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

A to B

B

C
Learning Switches

A to B

A to B

A to B

A to B

A to B

A to B

A to B

A to B

A to B

A to B

A to B
Learning Switches

A to B

A to B

A to B

A to B

A to B

B to A
Learning Switches
Learning Switches

<table>
<thead>
<tr>
<th>Dest</th>
<th>Next</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>B</td>
<td>S2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dest</th>
<th>Next</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>S1</td>
</tr>
<tr>
<td>B</td>
<td>S3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dest</th>
<th>Next</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>S2</td>
</tr>
<tr>
<td>B</td>
<td>B</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dest</th>
<th>Next</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>S1</td>
</tr>
<tr>
<td>A</td>
<td>S4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dest</th>
<th>Next</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>S4</td>
</tr>
<tr>
<td>A</td>
<td>S5</td>
</tr>
</tbody>
</table>
Learning Switches
Learning Switches
Is A to D flooded by S1?
Is A to D flooded by S1?

Yes! S1 has not seen it
Learning Switches

D never sends. When will switches learn about it?
Learning Switches

D never sends. When will switches learn about it?

Never
Learning Switches bad news
Learning Switches bad news: Loops!
Learning Switches bad news: Loops!

Is the problem worse with 2 loops?
Learning Switches bad news: Loops!

Is the problem worse with 2 loops?

Yes! # Packets grow exp-ly with >1 loops
How did Distance-Vector deal with loops?
Learning Switches bad news: Loops!

How did Distance-Vector deal with loops?

No flooding
Learning Switches bad news: Loops!

How did Link-State deal with loops?
Learning Switches bad news: Loops!

How did Link-State deal with loops?

Router Seq numbers to suppress duplicates
Spanning Tree Protocol (STP)

So far:

- Assumed underlying topology is a tree so it is safe to flood
- STP constructs a loop-free forwarding tree on top of loopy topology
  - Every node starts thinking it is root
  - Advertise (my_id, my_root, dist_to_root) to neighbors
Spanning Tree Protocol

Underlying topology, not good for forwarding in Learning Switches because of loops
Spanning Tree Protocol

Legend:
- Solid line: Part of spanning tree
- Red line: Not part of spanning tree
- Dashed line: Unknown

Diagram:
- A connected to S1, S1 connected to S2, S2 connected to S3, S3 connected to B
- D connected to S1, S1 connected to S4, S4 connected to S5, S5 connected to C
- S1, S2, S3, S4, S5 nodes are shaded.
Spanning Tree Protocol

Goal: construct a tree for flooding

Legend
- Part of spanning tree
- Not part of spanning tree
- Unknown
Spanning Tree Protocol

Legend
- Part of spanning tree
- Not part of spanning tree
- Unknown

Goal: construct a tree for flooding

Valid for forwarding?
Spanning Tree Protocol

Goal: construct a tree for flooding

Legend
- Part of spanning tree
- Not part of spanning tree
- Unknown

Valid for forwarding?
Spanning Tree Protocol

Goal: construct a tree for flooding

Valid for forwarding?
Yes! No loops

Legend
- Part of spanning tree
- Not part of spanning tree
- Unknown

A
S1
S2
S3
B
D
S4
S5
C
Spanning Tree Protocol

Goal: construct a tree for flooding

Legend
- Part of spanning tree
- Not part of spanning tree
- Unknown

Would STP ever produce this solution?

No. S1 not root, wrong tie breaking, not shortest paths
Spanning Tree Protocol

Legend
- Part of spanning tree
- Not part of spanning tree
- Unknown

Goal: construct a tree for flooding

Would STP ever produce this solution?
Spanning Tree Protocol

Goal: construct a tree for flooding

Legend
- Part of spanning tree
- Not part of spanning tree
- Unknown

Would STP ever produce this solution?

- Shortest paths from all switches to S1
- S1 (lowest id) can be considered root
Spanning Tree Protocol

Goal: construct a tree for flooding

Legend
- Part of spanning tree
- Not part of spanning tree
- Unknown

Would STP ever produce this solution?

No. S5 would have preferred S5-S2 from S5-S4

✅ Shortest paths from all switches to S1
✅ S1 (lowest id) can be considered root
Spanning Tree Protocol

Goal: construct a tree for flooding

Would STP ever produce this solution?
Spanning Tree Protocol

Legend
- Part of spanning tree
- Not part of spanning tree
- Unknown

Goal: construct a tree for flooding

Would STP ever produce this solution?
Yes!!
Spanning Tree Protocol

Goal: construct a tree for flooding

Legend

- Part of spanning tree
- Not part of spanning tree
- Unknown

Is this a unique equilibrium for STP or can there be others?
Spanning Tree Protocol

Legend
- Part of spanning tree
- Not part of spanning tree
- Unknown

Goal: construct a tree for flooding

Is this a unique equilibrium for STP or can there be others?

It is unique
Spanning Tree Protocol

Goal: construct a tree for flooding

Is this a unique equilibrium for STP or can there be others?

It is unique
Let’s run STP step by step
Spanning Tree Protocol

Legend
- Part of spanning tree
- Not part of spanning tree
- Unknown
Spanning Tree Protocol
Spanning Tree Protocol

<table>
<thead>
<tr>
<th></th>
<th>root</th>
<th>nxt_hop</th>
<th>dist</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>3</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>5</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>4</td>
<td>4</td>
<td>0</td>
</tr>
</tbody>
</table>
Spanning Tree Protocol

<table>
<thead>
<tr>
<th></th>
<th>root</th>
<th>nxt_hop</th>
<th>dist</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>3</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>5</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>4</td>
<td>4</td>
<td>0</td>
</tr>
</tbody>
</table>
Spanning Tree Protocol

```
<table>
<thead>
<tr>
<th></th>
<th>root</th>
<th>nxt_hop</th>
<th>dist</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>S2</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>S3</td>
<td>3</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>S4</td>
<td>4</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>S5</td>
<td>4</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

A -- S1 -- B
A -- S1 -- S4 -- S5 -- C
A -- S1 -- S3 -- S5 -- C
A -- S1 -- S2 -- B
A -- S1 -- S4 -- S5 -- B

Spanning Tree Protocol

<table>
<thead>
<tr>
<th>root</th>
<th>nxt_hop</th>
<th>dist</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>root</th>
<th>nxt_hop</th>
<th>dist</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>root</th>
<th>nxt_hop</th>
<th>dist</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>3</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>root</th>
<th>nxt_hop</th>
<th>dist</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>4</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>root</th>
<th>nxt_hop</th>
<th>dist</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>

Nodes: A, B, C, D, S1, S2, S3, S4, S5
Spanning Tree Protocol
Spanning Tree Protocol

<table>
<thead>
<tr>
<th></th>
<th>root</th>
<th>nxt_hop</th>
<th>dist</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>3</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>4</td>
<td>4</td>
<td>0</td>
</tr>
</tbody>
</table>
Spanning Tree Protocol

<table>
<thead>
<tr>
<th>root</th>
<th>nxt_hop</th>
<th>dist</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>
Spanning Tree Protocol

<table>
<thead>
<tr>
<th>root</th>
<th>nxt_hop</th>
<th>dist</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>root</th>
<th>nxt_hop</th>
<th>dist</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>3</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>root</th>
<th>nxt_hop</th>
<th>dist</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>4</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>root</th>
<th>nxt_hop</th>
<th>dist</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>
Spanning Tree Protocol

- **A**: Root node
- **B**: Next hop node
- **C**: Root node
- **D**: Next hop node
- **S1**: Root node
- **S2**: Next hop node
- **S3**: Root node
- **S4**: Next hop node
- **S5**: Root node

<table>
<thead>
<tr>
<th>root</th>
<th>nxt_hop</th>
<th>dist</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

The diagram shows the relationships and distances between the nodes.
Spanning Tree Protocol

A

S1

B

D

S2

S3

C

S4

S5

<table>
<thead>
<tr>
<th>root</th>
<th>nxt_hop</th>
<th>dist</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
Spanning Tree Protocol

<table>
<thead>
<tr>
<th>root</th>
<th>nxt_hop</th>
<th>dist</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>root</th>
<th>nxt_hop</th>
<th>dist</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>root</th>
<th>nxt_hop</th>
<th>dist</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>root</th>
<th>nxt_hop</th>
<th>dist</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

A —— S1 —— S2 —— S3 —— B
A —— S1 —— S4 —— S5 —— C

D —— S1
Spanning Tree Protocol

We chose some arbitrary order of message exchanges. What would happen if we chose a different order?
We chose some arbitrary order of message exchanges. What would happen if we chose a different order?

We would reach the same equilibrium!
Spanning Tree Protocol

Step 2: Disable all links not on spanning tree

Legend
- Part of spanning tree
- Not part of spanning tree
- Unknown

<table>
<thead>
<tr>
<th></th>
<th>root</th>
<th>nxt_hop</th>
<th>dist</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>S2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>S3</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>S4</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>S5</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Spanning Tree Protocol

Goal: construct a tree for flooding

Legend
- Part of spanning tree
- Not part of spanning tree
- Unknown

<table>
<thead>
<tr>
<th></th>
<th>root</th>
<th>nxt_hop</th>
<th>dist</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>S2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S5</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Legend
- Part of spanning tree
- Not part of spanning tree
- Unknown

<table>
<thead>
<tr>
<th></th>
<th>root</th>
<th>nxt_hop</th>
<th>dist</th>
</tr>
</thead>
<tbody>
<tr>
<td>S2</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>S3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S5</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Spanning Tree Protocol

Goal: construct a tree for flooding

Legend
- Part of spanning tree
- Not part of spanning tree
- Unknown

Which links does S1 enable/disable?
Spanning Tree Protocol

Goal: construct a tree for flooding

Legend
- Part of spanning tree
- Not part of spanning tree
- Unknown

<table>
<thead>
<tr>
<th>root</th>
<th>nxt_hop</th>
<th>dist</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

Which links does S1 enable/disable?
Spanning Tree Protocol

Goal: construct a tree for flooding

Legend
- Part of spanning tree
- Not part of spanning tree
- Unknown

Which links does S2 enable/disable?
Spanning Tree Protocol

Goal: construct a tree for flooding

Legend
- Part of spanning tree
- Not part of spanning tree
- Unknown

Which links does S2 enable/disable?
Spanning Tree Protocol

Goal: construct a tree for flooding

Legend
- Solid line: Part of spanning tree
- Dashed line: Not part of spanning tree
- Dotted line: Unknown

<table>
<thead>
<tr>
<th>root</th>
<th>nxt_hop</th>
<th>dist</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

Which links does S3 enable/disable?
Spanning Tree Protocol

Goal: construct a tree for flooding

Which links does S3 enable/disable?
Spanning Tree Protocol

Goal: construct a tree for flooding

Legend
- Part of spanning tree
- Not part of spanning tree
- Unknown

Which links does S4 enable/disable?
Spanning Tree Protocol

Goal: construct a tree for flooding

Legend
- Part of spanning tree
- Not part of spanning tree
- Unknown

Which links does S4 enable/disable?
Spanning Tree Protocol

Goal: construct a tree for flooding

Which links does S5 enable/disable?
Spanning Tree Protocol

Goal: construct a tree for flooding

Which links does S5 enable/disable?
Spanning Tree Protocol

Goal: construct a tree for flooding

Legend
- Part of spanning tree
- Not part of spanning tree
- Unknown

Which links does S5 enable/disable?
Spanning Tree Protocol

Goal: construct a tree for flooding

Legend
- Part of spanning tree
- Not part of spanning tree
- Unknown

After STP finishes building a spanning tree, will there ever be links still labeled “Unknown”?

No, the endpoint of the link which has a higher ID is guaranteed to change its state into either “Enabled” (part of spanning tree) or “Disabled” (not part of spanning tree)
Spanning Tree Protocol

Goal: construct a tree for flooding

We have a spanning tree now. Learning switches can use this for flooding!
Spanning Tree Protocol

Goal: construct a tree for flooding

We have a spanning tree now. Learning switches can use this for flooding!
Spanning Tree Protocol

Goal: construct a tree for flooding

We have a spanning tree now. Learning switches can use this for flooding!
Spanning Tree Protocol: failures

Legend
- Part of spanning tree
- Not part of spanning tree
- Unknown

Diagram showing the failure of node S1 with a red 'FAIL' label.
Spanning Tree Protocol: failures

Legend
- Solid line: Part of spanning tree
- Dashed line: Not part of spanning tree
- Red: Unknown

<table>
<thead>
<tr>
<th>root</th>
<th>nxt_hop</th>
<th>dist</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>
Spanning Tree Protocol: failures

Legend
- Part of spanning tree
- Not part of spanning tree
- Unknown

Orange highlights table changes
Spanning Tree Protocol: failures

Legend
- Part of spanning tree
- Not part of spanning tree
- Unknown

Orange highlights table changes
Spanning Tree Protocol: failures

Legend
- Part of spanning tree
- Not part of spanning tree
- Unknown

Orange highlights table changes
Spanning Tree Protocol: failures

Legend
- Part of spanning tree
- Not part of spanning tree
- Unknown

Orange highlights table changes
Spanning Tree Protocol: failures

Note:

After both links out of S1 failed, we still had a valid spanning tree and could in theory continue flooding on it

But STP forced us to recompute spanning tree
Spanning Tree Protocol: failures

Note:

After both links out of S1 failed, we still had a valid spanning tree and could in theory continue flooding on it

But STP forced us to recompute spanning tree

Can we do better?
Spanning Tree Protocol: failures

Note:

After both links out of S1 failed, we still had a valid spanning tree and could in theory continue flooding on it

But STP forced us to recompute spanning tree

Can we do better?

Yes. Beyond the course material. See Murphy’s AXE for one example and more references
Spanning Tree Protocol: failures

Note: After both links out of S1 failed, we still had a valid spanning tree and could in theory continue flooding on it. But STP forced us to recompute spanning trees.

Beyond the course material. See Murphy’s AXE for one example and more references.

Can we do better?

Yes. Beyond the course material. See Murphy’s AXE for one example and more references.
Learning Switches and Spanning Tree Protocol (STP)

In short, today we:

● Showed how learning switches learn via flooding in tree-topology
● Introduced loops in topology and realized we need to deal with them
● Introduced STP to “disable” certain links and have a tree for flooding
● Considered different failure scenarios and how STP reacts to them
Attribution

Slides based on Murphy McCauley’s slides the Spring 2020 iteration of CS168.

Written by Narek Galstyan, Sarah McClure, and Alex Krentsel.