Spanning Tree Protocol Simulation

Amir Razmjou
Fitzroy Nembhard

Table of Contents

Spanning Tree Protocol Simulation

Certain Assumptions were made to not relax the problem but satisfy facts

- Bridges are connected through networks. Each bridge connects at least two networks together.
- Generated Network Structures must be single component so there's a minimum for network edges density.
- Redundant bridges are not accepted.

Description of Application

This application simulates a random bipartite network of LAN Segments (Collision Networks) and Network Bridges.

Pseudocode

Input : Number of Nodes, Density

Optional: seed-network (used to generate graph)
seed-sync (used to desynchronize threads)

1. Create a concept network based on number of nodes and edges
   *The network can be thought of as a matrix of integers, which specify bridge IDs
   *Output the diameter of concept graph since it's important to know if Spanning Tree algorithm gets executed completely to make sure that diverse cost among bridges exists to root.
   *Output message if the current state of randomly generated graph is a tree or not.
   *Output message if the generated graph is a single connected component.
2. Use Bron Kerbosch Clique finder to find cliques in the network.
3. Create a LAN segment for each clique
4. Create a bridge for each node
   * Ensure each bridge is connected to at least two LAN segments.
5. Run each single LAN Segment object and Bridge object in different threads.

For each Bridge thread we have the following algorithm.

1. Each bridge advertises its ID as root ID to all neighbors.
2. Each bridge listens for all root advertisement frames from neighbors. If the root ID of neighbor is less than its own ID, it accepts the neighbor's ID and keep advertising it as new root ID until the specified timeout is reached.
3. Root bridge marks all of its ports as DP.
4. Once root bridge elected. Every bridge advertises its cost to reach root bridge to its neighbors. Each bridge marks the port with lowest cost to root bridge as RP port.

5. Every bridge advertises DP advertisements to neighbor bridges. The bridge with lowest cost to bridge marks its port to LAN Segment as DP port and the other one just block the ports.
    a. If a bridge didn't receive any DP advertisements from neighbor LAN Segments, it means that there is no other LAN Segment on other side of bridge so it marks the port as DP.
    b. If two bridges have the same cost to root, they use their port ID as a tiebreaker. Simply the one with lowest port ID (MAC Address) wins the competition.

For every LAN Segment object, a bridge simply forwards every frame it receives to all other ports.

Once bridges are done, they report status of all of their ports as either "BLOCKED", "RP" or "DP"

Sample Output

Images in examples were obtained using Gephi from resulting CSV output of simulation.

Example 1.

net. spanningtree.Main --show-csv --node 5 --density 0.5 --seed-network 10 --seed-sync 10
graph: ([0, 1, 2, 3, 4], [{0,2}, {0,4}, {1,0}, {1,4}, {2,4}, {3,0}, {3,2}])
degree distribution: {2=2, 3=2, 4=1}
graph diameter: 2.0
networks: [[N2] B2 B4 B0 , [N0] B0 B1 B4 , [N1] B0 B2 B3]
bridges: [[B0] N0 N1 N2 , [B1] N0 N3 , [B2] N1 N2 , [B3] N4 N1 , [B4] N2 N0]
Source,Target,Type
N2,B2,"LAN"
N2,B4,"LAN"
N2,B0,"LAN"
N0,B0,"LAN"
N0,B1,"LAN"
N0,B4,"LAN"
N1,B0,"LAN"
N1,B2,"LAN"
N1,B3,"LAN"
B0,N0,"BRIDGE"
B0,N1,"BRIDGE"
B0,N2,"BRIDGE"
B1,N0,"BRIDGE"
B1,N3,"BRIDGE"

B2,N1,"BRIDGE"
B2,N2,"BRIDGE"
B3,N4,"BRIDGE"
B3,N1,"BRIDGE"
B4,N2,"BRIDGE"
B4,N0,"BRIDGE"
B0 QS:0 Advertising R: 0 to [[N0] B0 B1 B4 , [N1] B0 B2 B3 , [N2] B2 B4 B0]
B1 QS:0 Advertising R: 1 to [[N0] B0 B1 B4 , [N3] B1]
B4 QS:0 Advertising R: 4 to [[N2] B2 B4 B0 , [N0] B0 B1 B4]
B2 QS:3 Advertising R: 2 to [[N1] B0 B2 B3 , [N2] B2 B4 B0]
B3 QS:1 Advertising R: 3 to [[N4] B3 , [N1] B0 B2 B3]
B4 QS:2 Changing root ID to: 0
B2 QS:3 Changing root ID to: 0
B3 QS:1 Changing root ID to: 0
B1 QS:0 Changing root ID to: 0
B4 QS:3 Advertising R: 0 to [[N2] B2 B4 B0 , [N0] B0 B1 B4]
B2 QS:4 Advertising R: 0 to [[N1] B0 B2 B3 , [N2] B2 B4 B0]
B3 QS:1 Advertising R: 0 to [[N4] B3 , [N1] B0 B2 B3]
B1 QS:1 Advertising R: 0 to [[N0] B0 B1 B4 , [N3] B1]
B2: cost to root is 1 from 15 [N2] B2 B4 B0
B3: cost to root is 1 from 11 [N1] B0 B2 B3
B1: cost to root is 1 from 3 [N0] B0 B1 B4
B4: cost to root is 1 from 17 [N2] B2 B4 B0
B1 advertising DP with cost 1 to [N3] B1
B4 advertising DP with cost 1 to [N0] B0 B1 B4
B0 advertising DP with cost 0 to [N0] B0 B1 B4
B0 advertising DP with cost 0 to [N1] B0 B2 B3
B0 advertising DP with cost 0 to [N2] B2 B4 B0
B2 advertising DP with cost 1 to [N1] B0 B2 B3
B3 advertising DP with cost 1 to [N4] B3
[B1] N0 N3 {3=RP, 19=DP}
[B4] N2 N0 {17=RP, 5=DP}
[B0] N0 N1 N2 {1=DP, 7=DP, 13=DP}
[B2] N1 N2 {9=RP, 15=DP}
[B3] N4 N1 {21=DP, 11=RP}

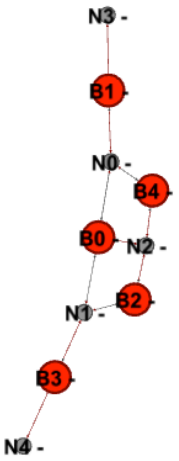

Figure 1: Graphical representation of example 1

Example 2.

B4 QS:0 Advertising R: 4 to [[N0] B4 B5 B7 , [N2] B2 B4 B0 , [N9] B2 B4 B7 , [N7] B2 B4 B6 , [N5] B4 B5 B0]
B5 QS:0 Advertising R: 5 to [[N5] B4 B5 B0 , [N0] B4 B5 B7 , [N1] B0 B1 B5 , [N4] B0 B3 B5 , [N10] B3 B5 B7 , [N8] B5 B7 B1]
B2 QS:1 Advertising R: 2 to [[N9] B2 B4 B7 , [N7] B2 B4 B6 , [N2] B2 B4 B0]
B1 QS:0 Advertising R: 1 to [[N3] B1 B6 , [N1] B0 B1 B5 , [N8] B5 B7 B1]
B6 QS:0 Advertising R: 6 to [[N3] B1 B6 , [N6] B3 B6 , [N7] B2 B4 B6]
B7 QS:5 Advertising R: 7 to [[N8] B5 B7 B1 , [N0] B4 B5 B7 , [N9] B2 B4 B7 , [N10] B3 B5 B7]
B3 QS:3 Advertising R: 3 to [[N6] B3 B6 , [N4] B0 B3 B5 , [N10] B3 B5 B7]
B0 QS:3 Advertising R: 0 to [[N1] B0 B1 B5 , [N4] B0 B3 B5 , [N2] B2 B4 B0 , [N5] B4 B5 B0]
B5 QS:6 Changing root ID to: 4
B6 QS:3 Changing root ID to: 3
B7 QS:7 Changing root ID to: 4
B4 QS:8 Changing root ID to: 2
B5 QS:11 Advertising R: 4 to [[N5] B4 B5 B0 , [N0] B4 B5 B7 , [N1] B0 B1 B5 , [N4] B0 B3 B5 , [N10] B3 B5 B7 , [N8] B5 B7 B1]
B6 QS:3 Advertising R: 3 to [[N3] B1 B6 , [N6] B3 B6 , [N7] B2 B4 B6]
B7 QS:9 Advertising R: 4 to [[N8] B5 B7 B1 , [N0] B4 B5 B7 , [N9] B2 B4 B7 , [N10] B3 B5 B7]
B4 QS:11 Advertising R: 2 to [[N0] B4 B5 B7 , [N2] B2 B4 B0 , [N9] B2 B4 B7 , [N7] B2 B4 B6 , [N5] B4 B5 B0]
B5 QS:11 Changing root ID to: 1
B5 QS:15 Advertising R: 1 to [[N5] B4 B5 B0 , [N0] B4 B5 B7 , [N1] B0 B1 B5 , [N4] B0 B3 B5 , [N10] B3 B5 B7 , [N8] B5 B7 B1]
B6 QS:2 Changing root ID to: 2

B1 QS:5 Changing root ID to: 0
B3 QS:7 Changing root ID to: 0
B2 QS:6 Changing root ID to: 0
B1 QS:7 Advertising R: 0 to [[N3] B1 B6 , [N1] B0 B1 B5 , [N8] B5 B7 B1]
B6 QS:2 Advertising R: 2 to [[N3] B1 B6 , [N6] B3 B6 , [N7] B2 B4 B6]
B3 QS:8 Advertising R: 0 to [[N6] B3 B6 , [N4] B0 B3 B5 , [N10] B3 B5 B7]
B2 QS:7 Advertising R: 0 to [[N9] B2 B4 B7 , [N7] B2 B4 B6 , [N2] B2 B4 B0]
B6 QS:2 Changing root ID to: 1
B4 QS:14 Changing root ID to: 0
B5 QS:16 Changing root ID to: 0
B7 QS:13 Changing root ID to: 1
B6 QS:4 Advertising R: 1 to [[N3] B1 B6 , [N6] B3 B6 , [N7] B2 B4 B6]
B4 QS:16 Advertising R: 0 to [[N0] B4 B5 B7 , [N2] B2 B4 B0 , [N9] B2 B4 B7 , [N7] B2 B4 B6 , [N5] B4 B5 B0]
B5 QS:16 Advertising R: 0 to [[N5] B4 B5 B0 , [N0] B4 B5 B7 , [N1] B0 B1 B5 , [N4] B0 B3 B5 , [N10] B3 B5 B7 , [N8] B5 B7 B1]
B7 QS:18 Advertising R: 1 to [[N8] B5 B7 B1 , [N0] B4 B5 B7 , [N9] B2 B4 B7 , [N10] B3 B5 B7]
B6 QS:3 Changing root ID to: 0
B6 QS:3 Advertising R: 0 to [[N3] B1 B6 , [N6] B3 B6 , [N7] B2 B4 B6]
B7 QS:7 Changing root ID to: 0
B7 QS:7 Advertising R: 0 to [[N8] B5 B7 B1 , [N0] B4 B5 B7 , [N9] B2 B4 B7 , [N10] B3 B5 B7]
B5: cost to root is 1 from 33 [N5] B4 B5 B0
B1: cost to root is 1 from 9 [N1] B0 B1 B5
B7: cost to root is 2 from 5 [N0] B4 B5 B7
B3: cost to root is 1 from 25 [N4] B0 B3 B5
B4: cost to root is 1 from 31 [N5] B4 B5 B0
B2: cost to root is 1 from 15 [N2] B2 B4 B0
B6: cost to root is 2 from 21 [N3] B1 B6
B4 advertising DP with cost 1 to [N0] B4 B5 B7
B4 advertising DP with cost 1 to [N2] B2 B4 B0
B4 advertising DP with cost 1 to [N9] B2 B4 B7
B4 advertising DP with cost 1 to [N7] B2 B4 B6
B5 advertising DP with cost 1 to [N0] B4 B5 B7
B5 advertising DP with cost 1 to [N1] B0 B1 B5
B5 advertising DP with cost 1 to [N4] B0 B3 B5
B5 advertising DP with cost 1 to [N10] B3 B5 B7
B5 advertising DP with cost 1 to [N8] B5 B7 B1
B2 advertising DP with cost 1 to [N9] B2 B4 B7
B2 advertising DP with cost 1 to [N7] B2 B4 B6
B1 advertising DP with cost 1 to [N3] B1 B6
B1 advertising DP with cost 1 to [N8] B5 B7 B1
B6 advertising DP with cost 2 to [N6] B3 B6
B7 advertising DP with cost 2 to [N8] B5 B7 B1
B6 advertising DP with cost 2 to [N7] B2 B4 B6

B7 advertising DP with cost 2 to [N9] B2 B4 B7
B7 advertising DP with cost 2 to [N10] B3 B5 B7
B3 advertising DP with cost 1 to [N6] B3 B6
B3 advertising DP with cost 1 to [N10] B3 B5 B7
B0 advertising DP with cost 0 to [N1] B0 B1 B5
B0 advertising DP with cost 0 to [N4] B0 B3 B5
B0 advertising DP with cost 0 to [N2] B2 B4 B0
B0 advertising DP with cost 0 to [N5] B4 B5 B0
B6 got cost of 1 on port 43 but I already got 2 his portId is 41
B7 got cost of 1 on port 55 but I already got 2 his portId is 53
[B4] N0 N2 N9 N7 N5 {1=DP, 17=DP, 53=DP, 41=DP, 31=RP}
[B5] N5 N0 N1 N4 N10 N8 {33=RP, 3=DP, 11=DP, 27=DP, 59=DP, 47=DP}
[B2] N9 N7 N2 {51=DP, 39=DP, 15=RP}
[B6] N3 N6 N7 {21=RP, 37=DP, 43=BLOCKED}
[B1] N3 N1 N8 {19=DP, 9=RP, 45=DP}
[B7] N8 N0 N9 N10 {49=RP, 5=DP, 55=BLOCKED, 61=DP}
[B3] N6 N4 N10 {35=DP, 25=RP, 57=DP}
[B0] N1 N4 N2 N5 {7=DP, 23=DP, 13=DP, 29=DP}

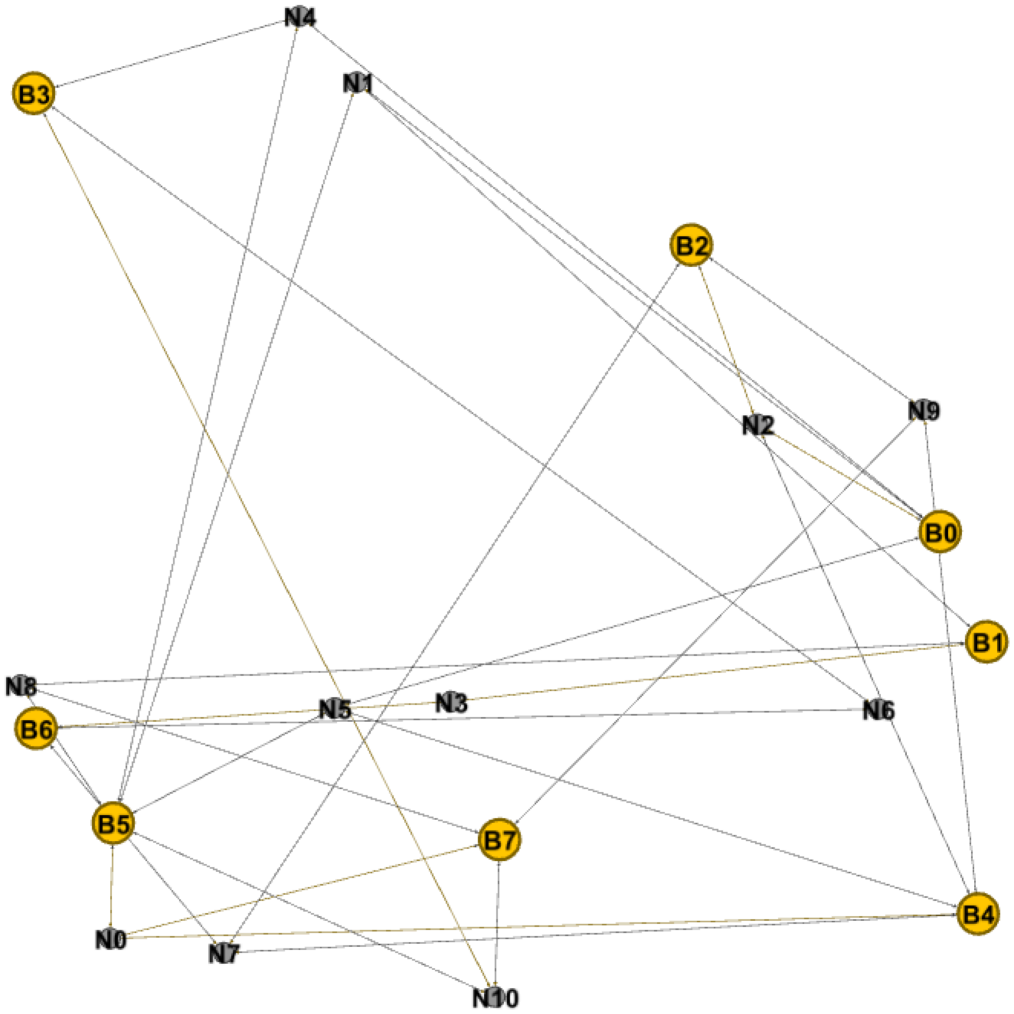

Figure 2: Graphical representation of example 2

Compiling Project

```
ant compile
```

Deploy

```
ant jar
```

Run

```
ant run
```

To Run
java stpsim3.jar --node 20 --density 0.2 --seed-network 10 --seed-sync 10

Design and Method:

- Used density as a probability factor to construct a concept network of bridges.
- Once concept network generated, extracted cliques to make LAN Segments. Just to make LAN Segments look more realistic and having LAN Segments with degree more than two.
- Based on concept network generated created network based on simulation objects.
- Ran all bridges and LAN Segments in distributed fashion. Used random numbers to make sure threads are desynchronized, just like real world.
- On bridge object implemented simple sequential state machine to keep track of the state of bridge.

Appendix A:

Assumptions 1: Network Structure

Bridges can interconnect by shared medium namely networks. Hence more than 2 bridges can participate in Layer-2 by being interconnected through the same network. Therefore network structure forms a bipartite graph of two classes of nodes, bridges and networks where minimum degree for bridges and networks are two and one.

$$min\big(P(b)\big) = 2$$
$$min\big(P(n)\big) = 1$$