Rick Boshae
SID: 861304798
CS164 W18 Section: 21

STP Code Write Up

In this project I implement the spanning tree protocol using Python. The Spanning Tree protocol is used to build loop free networks while still allowing backup links for fault tolerance. The STP algorithm creates a spanning tree within a set of connected bridges/switches by disabling links that are not part of the spanning tree.

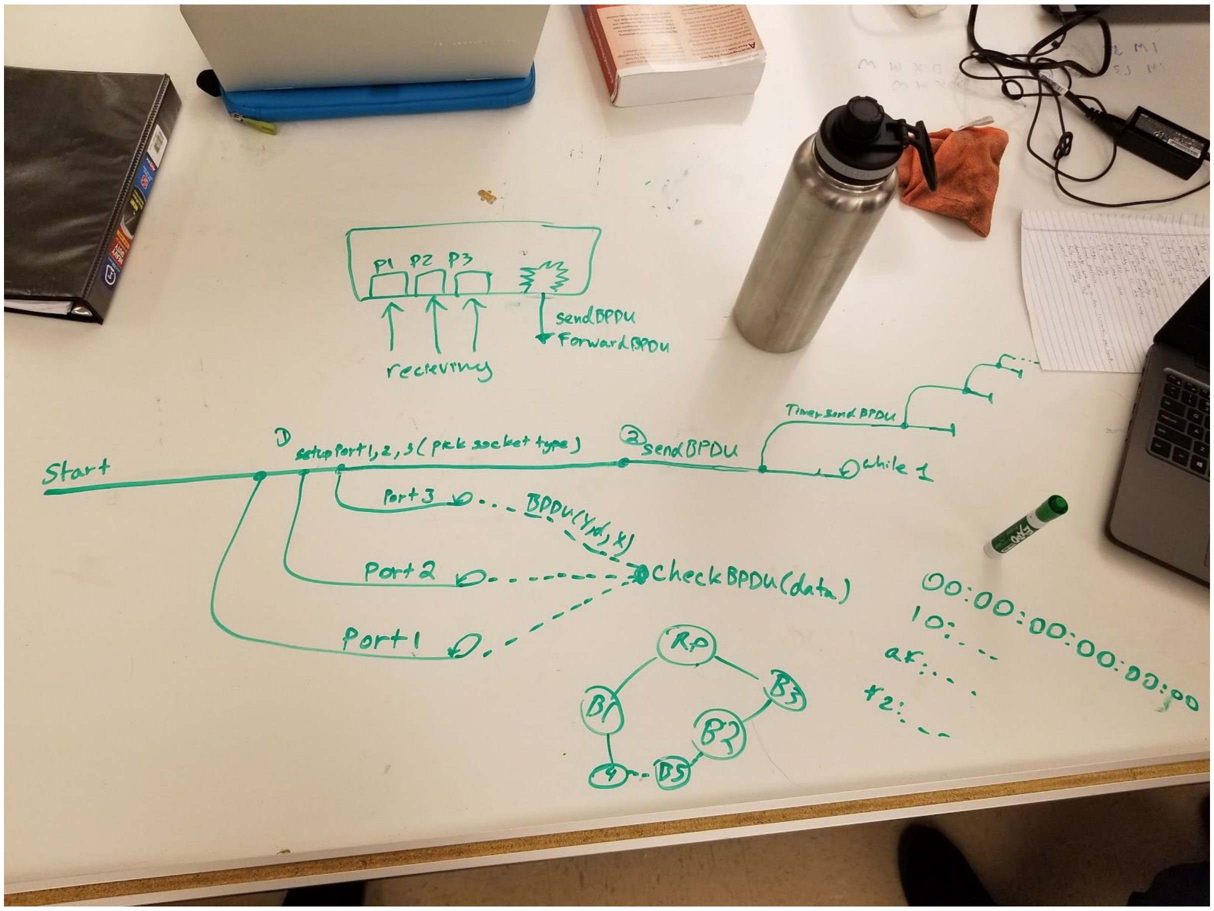


Figure: A graphical representation of Bridge.py

Main Components of the Code

Use of Globals

At the start of the program I retrieve information about the node using the commands.getoutput code provided by Mrs. Liri. I then use global variables to store the local information of each Bridge along with some if statements so I can maintain one program that works for each Bridge. Although I provide a method for retrieving the MAC address of a machine I've opted to hard code the MAC address for each node to produce repeatable results. The MAC addresses have been hard coded in a fashion that matches the project prompt.

printBridgeTable()

Prints the Bridge Table with a version number to identify the point at which the table was created.

sendBPDU(args)

SendBPDU starts by making a call to printBridgeTable to print out the current version of the table. Then each socket s1-s3 are set up to prepare an outgoing message to all DP's listed in the bridge table. Y.d. and X values are all assigned according to the current Root Mac, distance from root, and sender mac Id respectively. If the sending node is still root then the message is sent, else print to console that the node is not root and print a copy of the current bridge table.

Messages are sent based on the Port mapping declared in the project spec.

compareMAC(other_bridge_id, local_bridge)

Compares MAC address of each passed in argument. If other bridge id is less than local bridge return true.

forwardBPDU()

forwardBPDU is triggered when a message is received on one of it's ports from the root node. The method is nearly identical to sendBPDU with the difference being that a forwardBPDU call never sends messages again until it triggered by the root. forwardBPDU will only send messages to it's designated ports.

checkBPDU(data)

checkBPDU is called by the listening port. The data passed in is a BPDU. checkBPDU by printing to the console that a BPDU has been received. It then splits the data into a Y, d, X, and port variable. A series of important checks are made to determine if the sending BPDU is a RP, DP, or BP.

Y Value Equals Local Bridge ID

If the Y value is the same as the local bridge ID the data is thrown away. That is because in this case the root port is itself.

Rick Boshae
SID: 861304798
CS164 W18 Section: 21

Y Value Not Equal to Root Bridge ID

If the Y value is different from the root bridge id then a call to compareMAC is made. Check compareMAC for details. If the Y value is less than the current root port then a new root port is set in the bridge table and the old root port is changed into a DP.

Y Value Equals Root Bridge ID and Root Bridge ID is not the Local BID

A check is made on the distance from the sender to the root. If the distance is less than the current bridges distance then the root port is reassigned to the closer port and all previous BP's listed in the table are reset to DPs.

If the sender's bridge is further away it is then set to BP.

`clientThread(conn)`

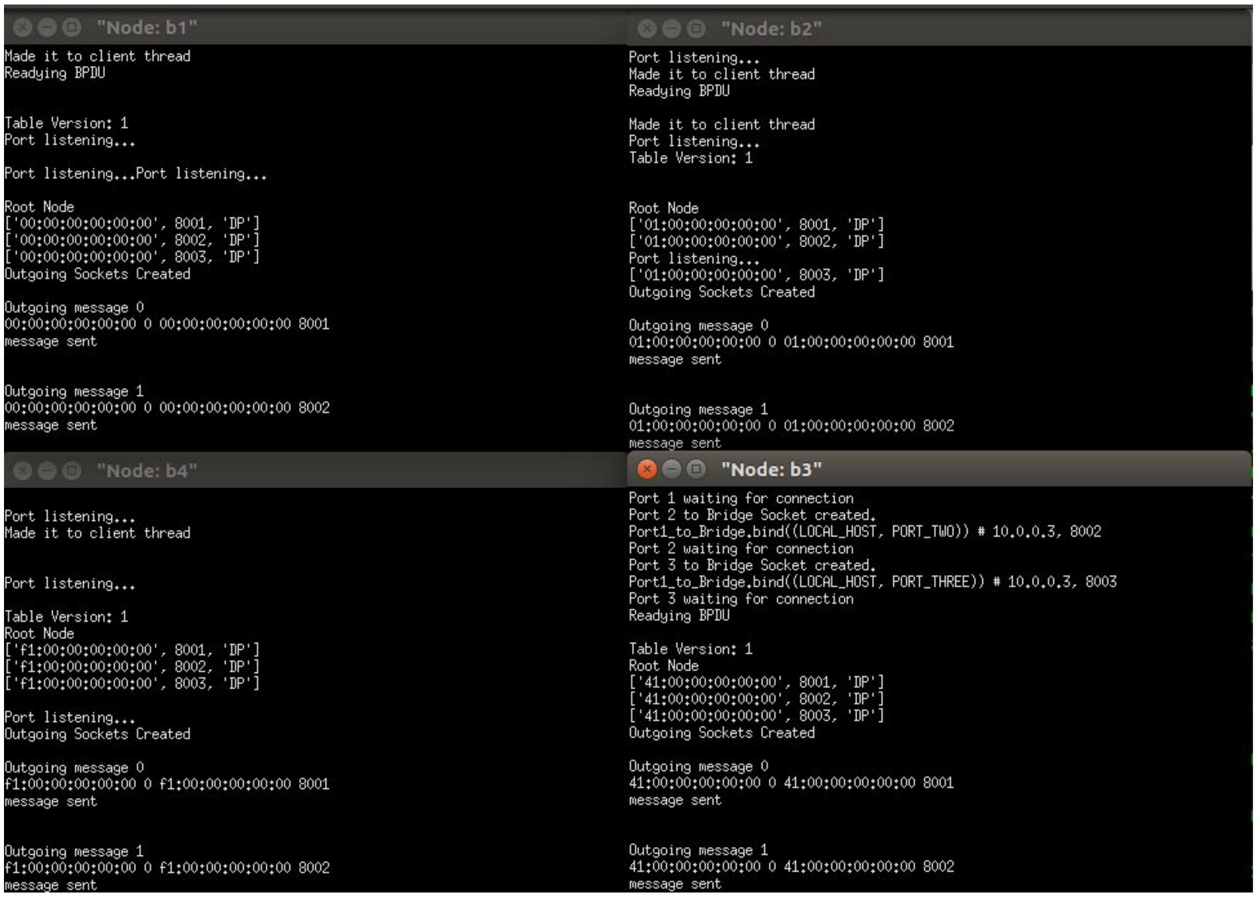
clientThread represents the listening port. When clientThread receives a message a call on checkBPDU is made.

`setup_bridge(Bridge_IP)`

setup_bridge (Bridge_IP) is responsible for setting up the Bridge information. It uses the Bridge IP address to assign the appropriate values. It then defaults all ports in the bridge table to DP. Sockets 1-3 are setup and started on seperate threads so the code can proceed. A new thread is then called to broadcast the first BPDU. This also prevents the program from prematurely exiting. A while 1 loop is then called to keep the program running.

Rick Boshae
SID: 861304798
CS164 W18 Section: 21

Initial State



"Node: b1"

```
Made it to client thread
Readying BPDU


Table Version: 1
Port listening...

Port listening...Port listening...

Root Node
['00:00:00:00:00:00', 8001, 'DP']
['00:00:00:00:00:00', 8002, 'DP']
['00:00:00:00:00:00', 8003, 'DP']
Outgoing Sockets Created

Outgoing message 0
00:00:00:00:00:00 0 00:00:00:00:00:00 8001
message sent


Outgoing message 1
00:00:00:00:00:00 0 00:00:00:00:00:00 8002
message sent
```

"Node: b2"

```
Port listening...
Made it to client thread
Readying BPDU

Made it to client thread
Port listening...
Table Version: 1


Root Node
['01:00:00:00:00:00', 8001, 'DP']
['01:00:00:00:00:00', 8002, 'DP']
Port listening...
['01:00:00:00:00:00', 8003, 'DP']
Outgoing Sockets Created

Outgoing message 0
01:00:00:00:00:00 0 01:00:00:00:00:00 8001
message sent


Outgoing message 1
01:00:00:00:00:00 0 01:00:00:00:00:00 8002
message sent
```

"Node: b4"

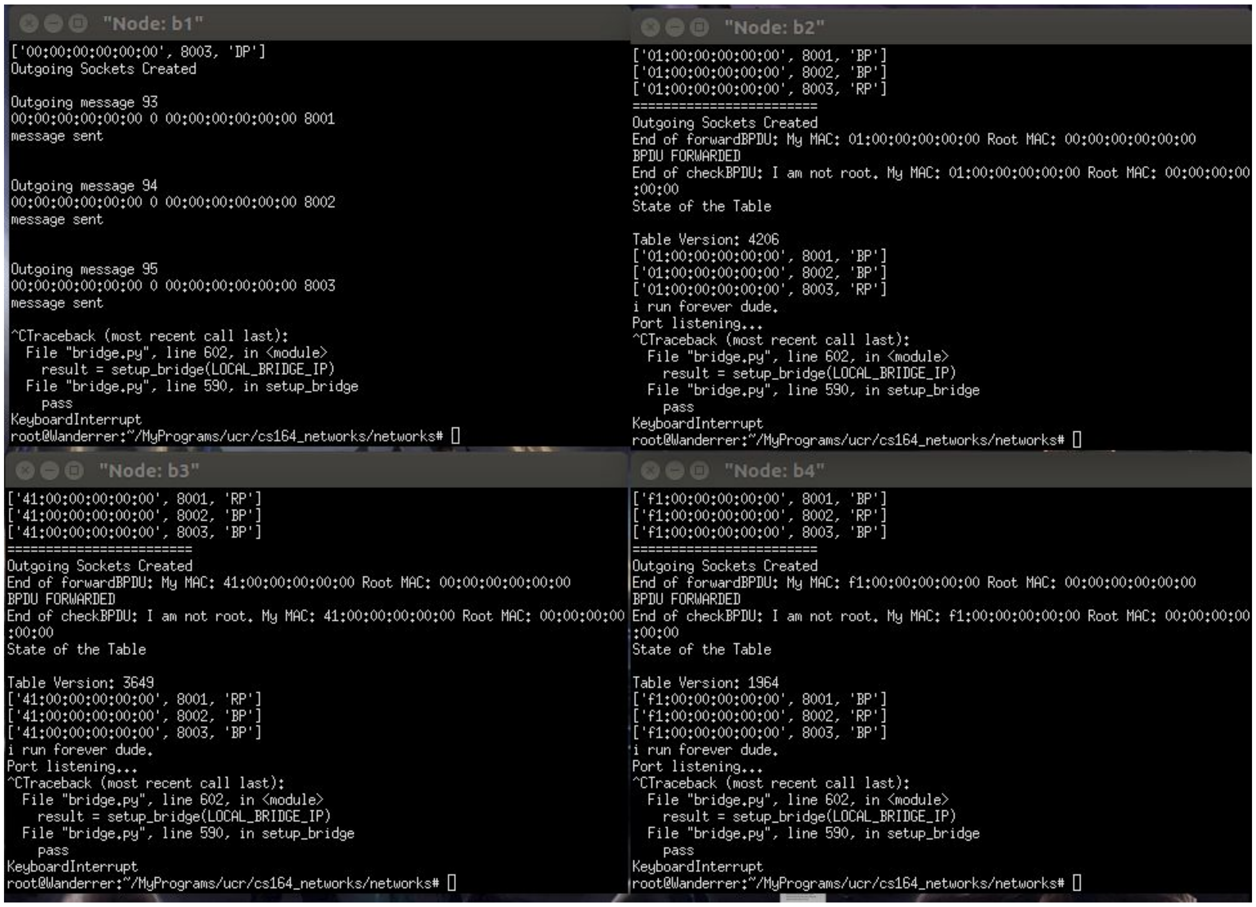
```

Port listening...
Made it to client thread


Port listening...

Table Version: 1
Root Node
['f1:00:00:00:00:00', 8001, 'DP']
['f1:00:00:00:00:00', 8002, 'DP']
['f1:00:00:00:00:00', 8003, 'DP']

Port listening...
Outgoing Sockets Created

Outgoing message 0
f1:00:00:00:00:00 0 f1:00:00:00:00:00 8001
message sent


Outgoing message 1
f1:00:00:00:00:00 0 f1:00:00:00:00:00 8002
message sent
```

"Node: b3"

```
Port 1 waiting for connection
Port 2 to Bridge Socket created.
Port1_to_Bridge.bind((LOCAL_HOST, PORT_TWO)) # 10.0.0.3, 8002
Port 2 waiting for connection
Port 3 to Bridge Socket created.
Port1_to_Bridge.bind((LOCAL_HOST, PORT_THREE)) # 10.0.0.3, 8003
Port 3 waiting for connection
Readying BPDU

Table Version: 1
Root Node
['41:00:00:00:00:00', 8001, 'DP']
['41:00:00:00:00:00', 8002, 'DP']
['41:00:00:00:00:00', 8003, 'DP']
Outgoing Sockets Created

Outgoing message 0
41:00:00:00:00:00 0 41:00:00:00:00:00 8001
message sent


Outgoing message 1
41:00:00:00:00:00 0 41:00:00:00:00:00 8002
message sent
```

Rick Boshae
SID: 861304798
CS164 W18 Section: 21

Final



```
['00:00:00:00:00:00', 8003, 'DP']
Outgoing Sockets Created

Outgoing message 93
00:00:00:00:00:00 0 00:00:00:00:00:00 8001
message sent


Outgoing message 94
00:00:00:00:00:00 0 00:00:00:00:00:00 8002
message sent


Outgoing message 95
00:00:00:00:00:00 0 00:00:00:00:00:00 8003
message sent

^CTraceback (most recent call last):
  File "bridge.py", line 602, in <module>
    result = setup_bridge(LOCAL_BRIDGE_IP)
  File "bridge.py", line 590, in setup_bridge
    pass
KeyboardInterrupt
root@Wanderrer:~/MyPrograms/ucr/cs164_networks/networks# []
```

```
['01:00:00:00:00:00', 8001, 'BP']
['01:00:00:00:00:00', 8002, 'BP']
['01:00:00:00:00:00', 8003, 'RP']
========================
Outgoing Sockets Created
End of forwardBPDU: My MAC: 01:00:00:00:00:00 Root MAC: 00:00:00:00:00:00
BPDU FORWARDED
End of checkBPDU: I am not root. My MAC: 01:00:00:00:00:00 Root MAC: 00:00:00:00
:00:00
State of the Table

Table Version: 4206
['01:00:00:00:00:00', 8001, 'BP']
['01:00:00:00:00:00', 8002, 'BP']
['01:00:00:00:00:00', 8003, 'RP']
i run forever dude.
Port listening...
^CTraceback (most recent call last):
  File "bridge.py", line 602, in <module>
    result = setup_bridge(LOCAL_BRIDGE_IP)
  File "bridge.py", line 590, in setup_bridge
    pass
KeyboardInterrupt
root@Wanderrer:~/MyPrograms/ucr/cs164_networks/networks# []
```

```
['41:00:00:00:00:00', 8001, 'RP']
['41:00:00:00:00:00', 8002, 'BP']
['41:00:00:00:00:00', 8003, 'BP']
========================
Outgoing Sockets Created
End of forwardBPDU: My MAC: 41:00:00:00:00:00 Root MAC: 00:00:00:00:00:00
BPDU FORWARDED
End of checkBPDU: I am not root. My MAC: 41:00:00:00:00:00 Root MAC: 00:00:00:00
:00:00
State of the Table

Table Version: 3649
['41:00:00:00:00:00', 8001, 'RP']
['41:00:00:00:00:00', 8002, 'BP']
['41:00:00:00:00:00', 8003, 'BP']
i run forever dude.
Port listening...
^CTraceback (most recent call last):
  File "bridge.py", line 602, in <module>
    result = setup_bridge(LOCAL_BRIDGE_IP)
  File "bridge.py", line 590, in setup_bridge
    pass
KeyboardInterrupt
root@Wanderrer:~/MyPrograms/ucr/cs164_networks/networks# []
```

```
['f1:00:00:00:00:00', 8001, 'BP']
['f1:00:00:00:00:00', 8002, 'RP']
['f1:00:00:00:00:00', 8003, 'BP']
========================
Outgoing Sockets Created
End of forwardBPDU: My MAC: f1:00:00:00:00:00 Root MAC: 00:00:00:00:00:00
BPDU FORWARDED
End of checkBPDU: I am not root. My MAC: f1:00:00:00:00:00 Root MAC: 00:00:00:00
:00:00
State of the Table

Table Version: 1964
['f1:00:00:00:00:00', 8001, 'BP']
['f1:00:00:00:00:00', 8002, 'RP']
['f1:00:00:00:00:00', 8003, 'BP']
i run forever dude.
Port listening...
^CTraceback (most recent call last):
  File "bridge.py", line 602, in <module>
    result = setup_bridge(LOCAL_BRIDGE_IP)
  File "bridge.py", line 590, in setup_bridge
    pass
KeyboardInterrupt
root@Wanderrer:~/MyPrograms/ucr/cs164_networks/networks# []
```

Conclusion

With a lot of hard work and help from my instructors I was successful in implementing the spanning tree algorithm. The most challenging part of this project was familiarizing myself with Python. This was really the first time I've been asked to code an assignment in Python so doing some advanced things like parsing a string, multithreading, or retrieving information from commands took time to figure out.