Rochester Institute of Technology

NSSA441

Advanced Routing and Switching

Phase II Report

By

Jeremy Peters

Chase Peterson

Aman Garg

Introduction

VLANs today are a good way to logically partition and isolate in a computer network at Layer 2 level. To preserve these groupings between sites which are geographically separated, It can be expensive and time consuming to set up a connection. We can use Q in Q, which can reduce cost and the hardware requirements needed to communicate between 2 sites. A service provider may provide users with Q in Q tunneling to help them access one or more sites which are geographically separated but are close enough to use the same ISP and to users who might want layer 2 VPN. Schools, universities and companies which are located in metropolitan areas, campuses, or sites which are distributed all across the city may want to use Q in Q to avoid additional operating costs.

To understand the concepts of Q in Q we wanted to demonstrate inter-site connectivity using layer 2 using a scaled down implementation of the technology. To further understand how Q in Q may work within a service provider network, we decided to increase the redundancy inside the topology and see how Q in Q works with loop avoidance protocol like STP. How much time does it take for a connection to come back up if there any topology changes or link failure when the data is being transferred.

In this report we are going to discuss some of the technologies we used to conduct our experiments, these technologies include but are not limited to Q in Q (802.1Q), STP, and Wireshark. We also discuss the experiments we conducted, why we used the topology and the information collected from each of the experiments. We also included the results of the experiments and compared the results we got from each of the experiments. We concluded this report with our observations and recommendations based on our experiments.

Technology Coverage

Q in Q tunneling also known as 802.1Q tunneling is a technique used by Metro Ethernet providers as a layer 2 VPN for their customers. For 802.1Q tunneling, the provider will put an 802.1Q tag on all the frames that it receives from a customer with a unique VLAN tag. By using a different VLAN tag it isolates the traffic of one customer from the other while transparently transferring it throughout the service provider network. When the traffic reaches the other end of the network, the service provider stipps off the tag from the traffic and sends it to the customer VLAN.

STP, also known as Spanning Tree Protocol is a layer 2 network protocol that runs on the switches and bridges to ensure that there are no loops in the topology for Ethernet networks. With STP, all the switches in a network elects a root bridge that becomes the focal point of the network, and all the decisions such as deciding upon the block port to forward port is made from a perspective of the root bridge. Whenever the root bridge is disconnected the network elects a new root port based on the path selected to reach the root port.

Wireshark is a network traffic capture tool using libpcap API (NPCAP on Windows). Unfortunately, VLAN tags are notoriously difficult to capture "It depends on the NIC, the NIC firmware, the driver, and the alignment of the moon and planets" (Wireshark). These issues are compounded by the way in which certain operating systems process & store network traffic. Windows, by default, is set to strip VLAN tags. This makes capturing VLAN traffic difficult, as the only way to disable tag stripping is to modify the registry entry for monitor mode. In comparison, most Linux distros are set to promiscuous mode by default and so the operating system won't strip tags when capturing.

When performing initial testing, our group encountered the issue of tag stripping outlined above. When capturing within our PBN's topology, Windows was stripping the tags introduced by the PBN edge switches. In order to confirm that our outer tag was being stripped. we introduced another PBN on top of the existing one (*See Figure 1.*). This allowed us to see the tag introduced by our original PBN. Upon research of the issue, we decided that the best way to capture traffic was to do so with a device running Linux.

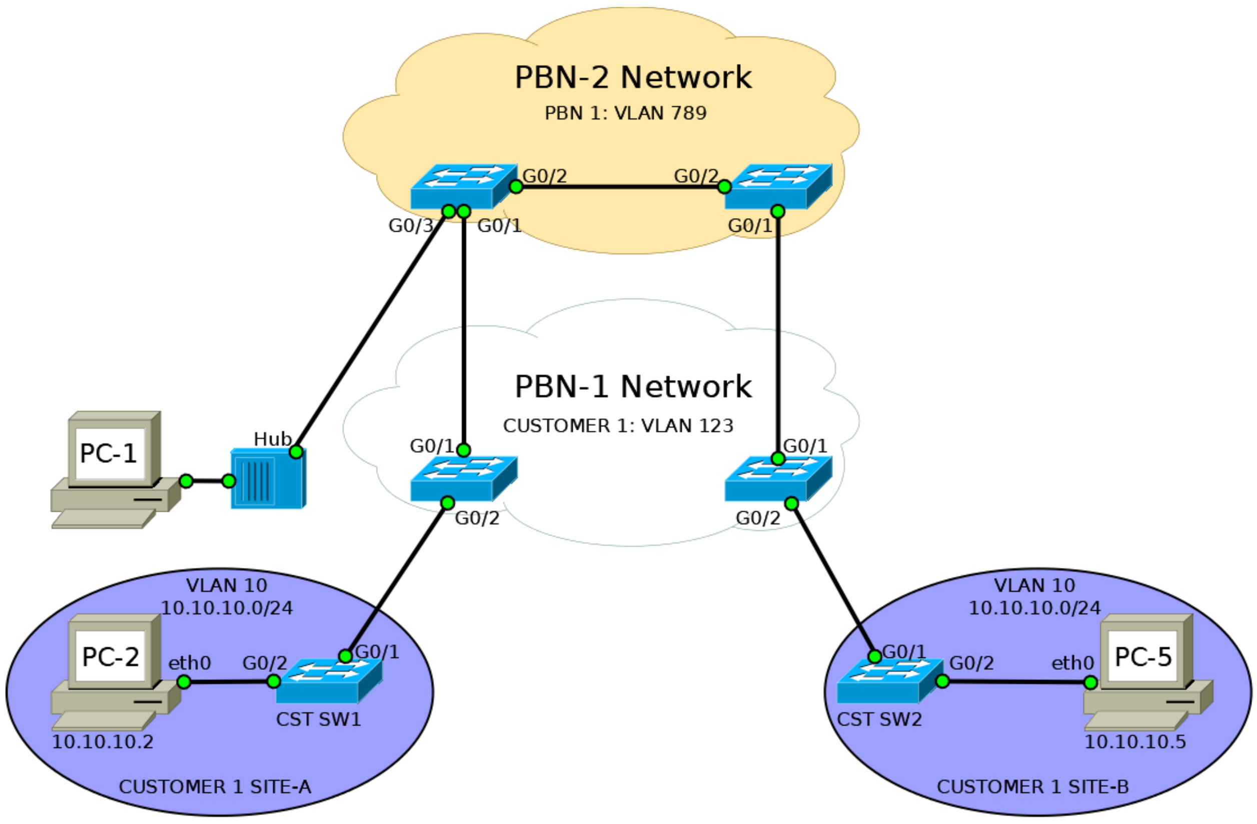


Figure 1. Double Layered PBN

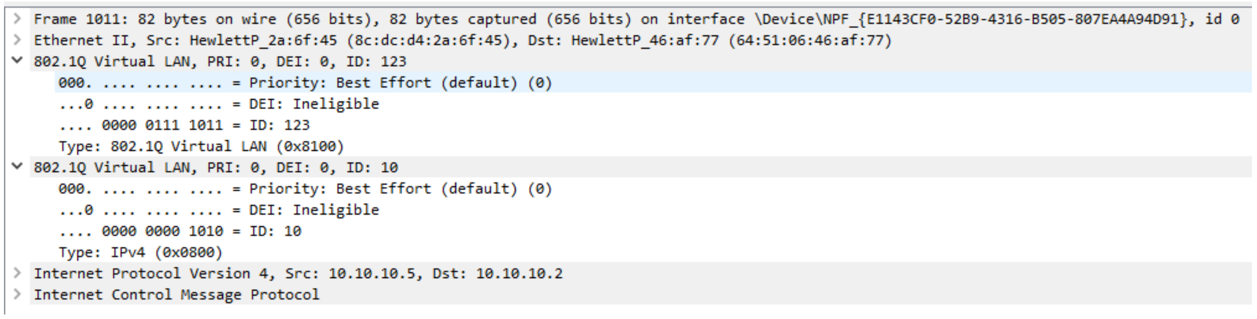
Each of these technologies were implemented as our group increased the complexity of our experiments, we started by implementing 802.1Q tunneling and later added redundancy to our topology in order to understand how Q in Q works with loop avoidance protocols like STP.

Experiments

Experiment 1:

For our first experiment we used one customer network which consisted of two separate sites. Each site included one switch and one client computer. The customer network consists of a single VLAN (VLAN 10) which spanned over either site. We only used one VLAN for the customer network, but adding more VLANs would be possible. The PBN consists of two switches each belonging to a single VLAN. The VLAN had an ID of 123.

When conducting our initial test, we received a destination not found error message. This was due to each computer sending out ARP messages which traversed the PBN. ARP resolution took a few seconds but eventually we started getting successful pings. We were then able to capture both the traffic from the Customer 1 Network. If you dive into any of the messages, you can see what the customer VLAN is as well as the Provider Bridge Network VLAN. For customer 1 you can see that the customer VLAN (at the bottom) shows VLAN 10.



```
> Frame 1011: 82 bytes on wire (656 bits), 82 bytes captured (656 bits) on interface \Device\NPF_{E1143CF0-52B9-4316-B505-807EA4A94D91}, id 0
> Ethernet II, Src: HewlettP_2a:6f:45 (8c:dc:d4:2a:6f:45), Dst: HewlettP_46:af:77 (64:51:06:46:af:77)
v 802.1Q Virtual LAN, PRI: 0, DEI: 0, ID: 123
    000. .... .... .... = Priority: Best Effort (default) (0)
    ...0 .... .... .... = DEI: Ineligible
    .... 0000 0111 1011 = ID: 123
    Type: 802.1Q Virtual LAN (0x8100)
v 802.1Q Virtual LAN, PRI: 0, DEI: 0, ID: 10
    000. .... .... .... = Priority: Best Effort (default) (0)
    ...0 .... .... .... = DEI: Ineligible
    .... 0000 0000 1010 = ID: 10
    Type: IPv4 (0x0800)
> Internet Protocol Version 4, Src: 10.10.10.5, Dst: 10.10.10.2
> Internet Control Message Protocol
```

Figure 2: PBN Tag Capture

The Provider Bridge Network stacked VLAN tag is shown at the top as VLAN 123. After 10 minutes of continuous testing, latency detected was around 1-2ms. The latency involved with traversing a Provider Bridge Network will change depending on the geographical distance between each customer site. The main goal of this experiment was to show a working Provider Bridge Network.

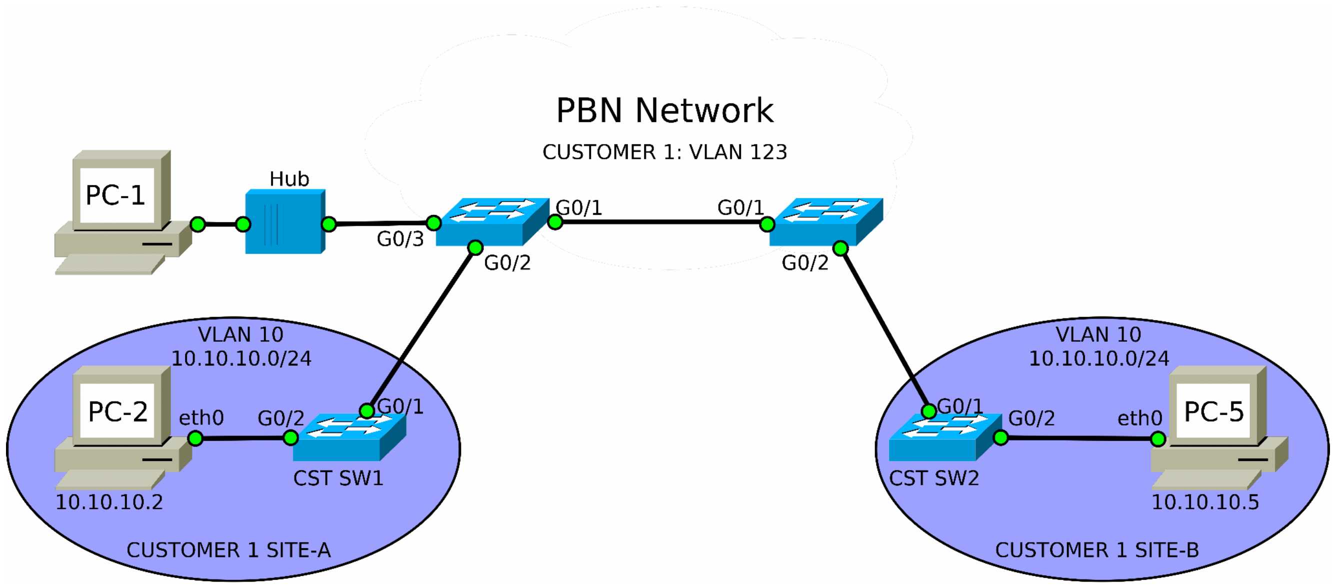


Figure 3: Double Layered PBN Topology

Experiment 2

In the second experiment we introduced a second customer network. We want to show that multiple customers can have the same network addressing scheme and still function properly over the Provider Bridge Network**.** As expected in a real world environment, this particular situation would arise extremely frequently considering a service provider would have many different customers. In this experiment we made it so Customer 1 and Customer 2 each have the same VLAN and IP address scheme. We had the same setup as the last experiment sending ICMP traffic over the Provider Bridge Network. Our results were what we expected. The Provider Bridge Network allowed for traffic from each customer to be isolated and successfully transferred to the corresponding Customer network. The reason this works is because the Provider Bridge Network applies its own VLAN tag on top of the originally customer VLAN tag.

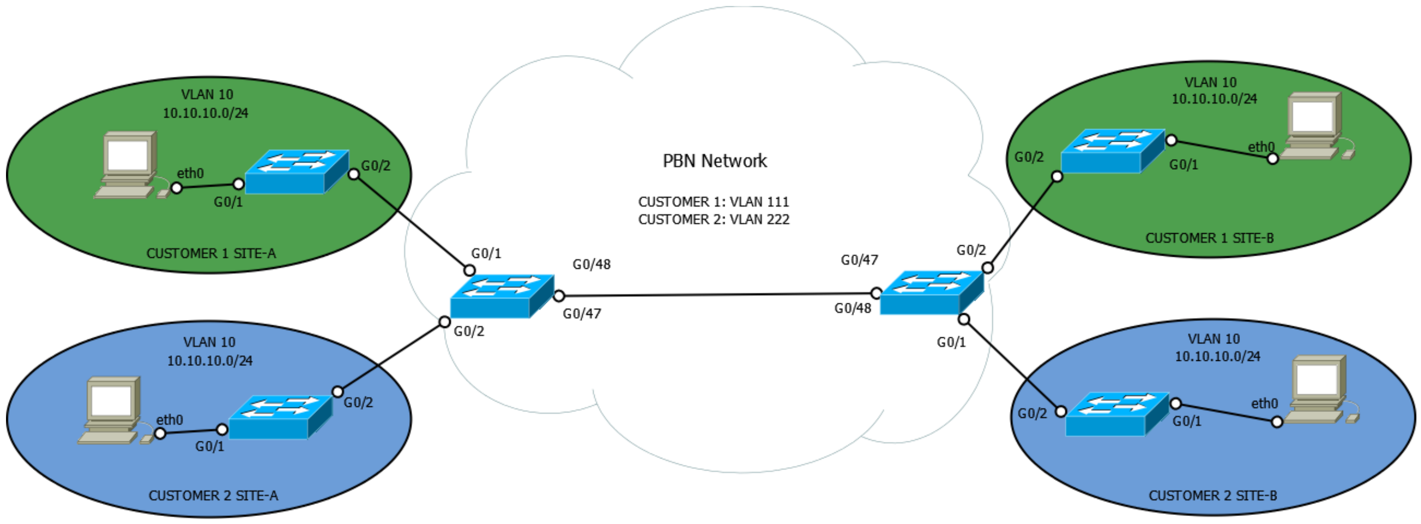


Figure 4: Two Customer PBN Topology

Experiment 3

In our last experiment we wanted to incorporate some form of redundancy within the PBN network. In a real world scenario, it would be an absolute necessity to incorporate redundancy within their network. We added a third PBN switch and formed a full mesh topology. Each PBN switch hosted every single Customer VLAN in case the spanning tree logical topology changed. During our experiment we sent ICMP traffic like before, but increased the network load by sending large multi-gigabyte files. We created Windows network shares which spanned between Customer 1's site A and B, and Customer 2's site A and B. The point of this topology is to show a level of redundancy with our PBN and so while transferring the large files we took out the trunk cables between the switches that connect VLANS (10, 20) and (10, 30).

We also took out the trunk cable between the switches that hosted VLANS (20, 30) and (30, 10).

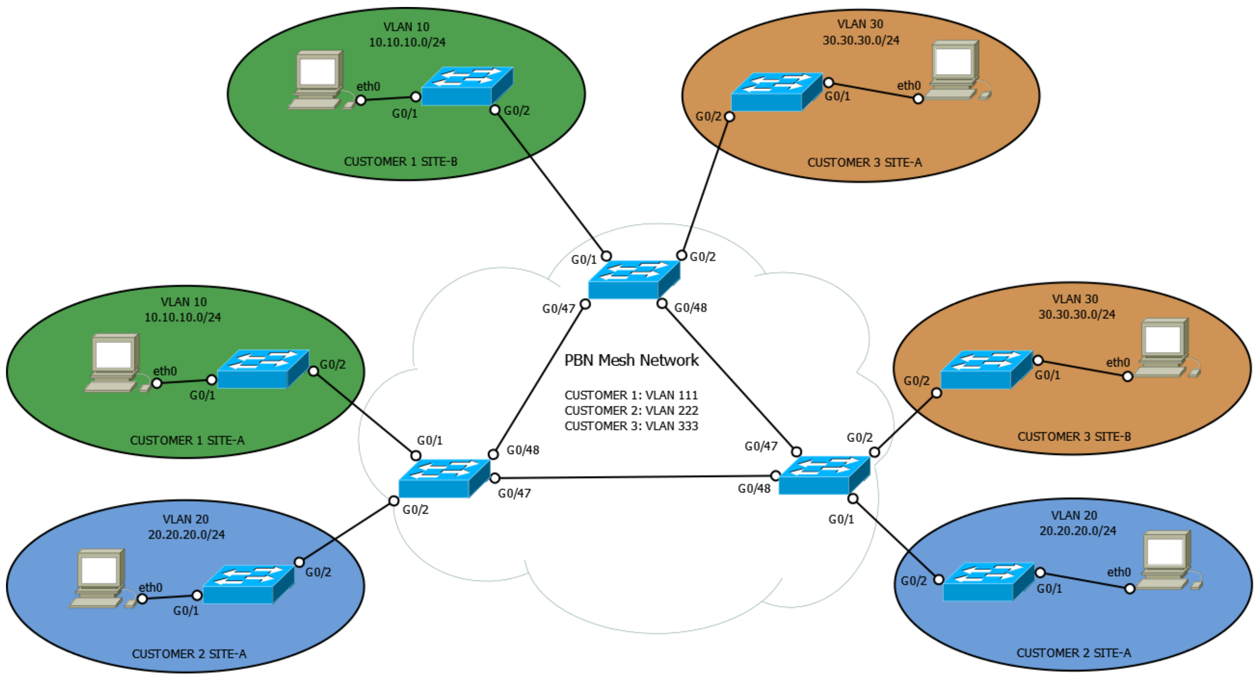


Figure : Redundant PBN Topology

Results:

From all the experiments that we conducted we found out that we need to have a machine running on linux to capture the Q in Q packets as network drivers installed in the windows devices strips off the 802.1Q tag. We observed that the PBN is successful in isolating the traffic from each VLAN and despite having the same network addressing scheme there was no effect on the topology.

From our last experiments we observed that it takes more than 30 seconds for an STP protocol to bring back an active connection in case of a link failure. We also noticed that it took approximately the same amount of time to bring back a connection when the link was disconnected and connected back again with a few seconds.

Discussion of results:

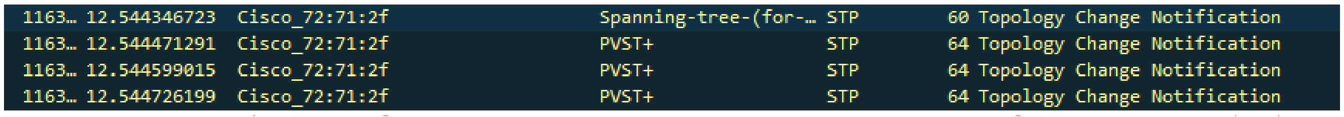
From our experiment one we wanted to test our topology and figure out how Q in Q works. From our initial topology we found out that to capture 802.1Q packets we need to have a device which runs on linux because the network drivers installed in the windows devices strips off the 802.1Q tag. We also observed how Q in Q is used to isolate the traffic from each VLAN, from the second experiment and found that Q in Q is successful in isolating the traffic from each VLAN with the similar network addressing by having each VLAN in the 10.10.10.0/24 network. As per our expectations the network was successful in isolating the traffic and did not interfere with the traffic on the different VLAN i.e we successfully created layer 2 tunneling using Q in Q. From our last test we introduced some redundancy in our network to test out how Q in Q works with loop avoidance protocol like STP. From our test we found out that it took more than 30 seconds for STP to bring back an active connection incase of a link failure. Based upon the time taken by the STP to bring back the connection and noting the fact about the PBN charging a lot of fees to their customer for this service, the use of STP as their loop avoidance protocol is completely unacceptable. To find out the time taken by the STP protocol to bring back the connection incase of a link failure we sent large data packets over the network and broke the connection to the root port, we observed that the transfer stopped for more than 30 seconds before continuing the traffic flow. After this we also isolated a switch from the other switches and had only one connection to another switch for the traffic and while the traffic was flowing we disconnected the link and connected it within a few seconds, we noticed that it took about the same amount of time for the network to bring back the connection.

While conducting our STP tests we observed that the ports were not active and the connection to that switch was not active, We suspected that it could be because the switch was not aware of all the VLANs on the PBN network.when we made all the switches aware about the VLANs on the network, the ports came back up and the connection was stable. From this we came to a conclusion that all the switches should be aware of all the VLANs on the PBN network.

Critical Summary:

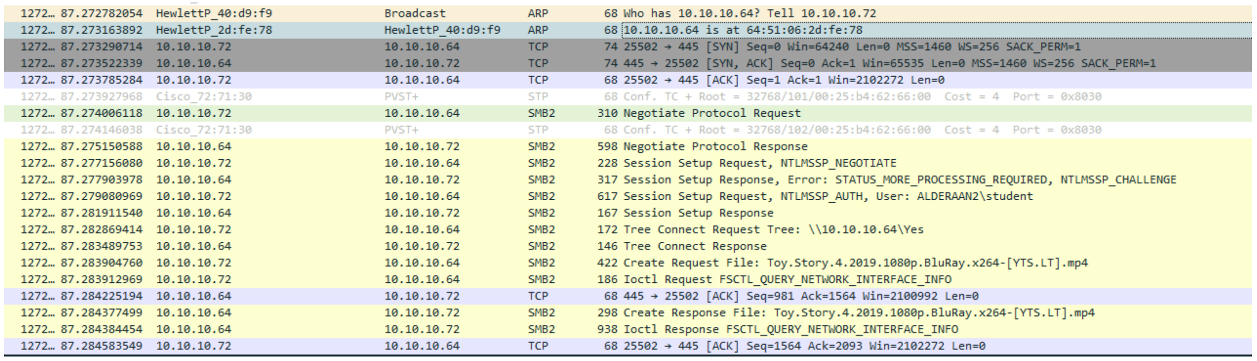
The benefits of using Q in Q lies in its simplicity, but this simplicity also creates problems as the scope of its usage increases i.e. Q in Q has a few scalability issues/limitations. The first limitation faced by Q in Q is that of distance. Q in Q is limited by the geographical distance between the two sites that are being connected. The second limitation, that is even observable in small scale tests, is the need for all switches in PBN to know about all other devices in the topology. As network size increases, so too does the size of the switches' SAT. At a certain point, network traffic will be congested with ARP traffic. There have been some proposed solutions to this issue, but to date none have been implemented into the 802.1q standard. Though not a fault from Q in Q, convergence time for STP will also be an issue for large PBNs. To mitigate this issue, PBN should configure their switches to use RSTP to reduce convergence time and increase network uptime. Despite all of these limitations and scalability concerns, Q in Q is still an extremely useful tool for the preservation of layer 2 logical groupings between sites through the use of tunneling.

Appendix



```
1163… 12.544346723  Cisco_72:71:2f                         Spanning-tree-(for-… STP        60 Topology Change Notification
1163… 12.544471291  Cisco_72:71:2f                         PVST+                STP        64 Topology Change Notification
1163… 12.544599015  Cisco_72:71:2f                         PVST+                STP        64 Topology Change Notification
1163… 12.544726199  Cisco_72:71:2f                         PVST+                STP        64 Topology Change Notification
```

Figure 6: STP Packet capture



```
1272… 87.272782054 HewlettP_40:d9:f9              Broadcast          ARP       68 Who has 10.10.10.64? Tell 10.10.10.72
1272… 87.273163892 HewlettP_2d:fe:78              HewlettP_40:d9:f9  ARP       68 10.10.10.64 is at 64:51:06:2d:fe:78
1272… 87.273290714 10.10.10.72                    10.10.10.64        TCP       74 25502 → 445 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
1272… 87.273522339 10.10.10.64                    10.10.10.72        TCP       74 445 → 25502 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM=1
1272… 87.273785284 10.10.10.72                    10.10.10.64        TCP       68 25502 → 445 [ACK] Seq=1 Ack=1 Win=2102272 Len=0
1272… 87.273927968 Cisco_72:71:30                 PVST+              STP       68 Conf. TC + Root = 32768/101/00:25:b4:62:66:00  Cost = 4  Port = 0x8030
1272… 87.274006118 10.10.10.72                    10.10.10.64        SMB2     310 Negotiate Protocol Request
1272… 87.274146038 Cisco_72:71:30                 PVST+              STP       68 Conf. TC + Root = 32768/102/00:25:b4:62:66:00  Cost = 4  Port = 0x8030
1272… 87.275150588 10.10.10.64                    10.10.10.72        SMB2     598 Negotiate Protocol Response
1272… 87.277156080 10.10.10.72                    10.10.10.64        SMB2     228 Session Setup Request, NTLMSSP_NEGOTIATE
1272… 87.277903978 10.10.10.64                    10.10.10.72        SMB2     317 Session Setup Response, Error: STATUS_MORE_PROCESSING_REQUIRED, NTLMSSP_CHALLENGE
1272… 87.279080969 10.10.10.72                    10.10.10.64        SMB2     617 Session Setup Request, NTLMSSP_AUTH, User: ALDERAAN2\student
1272… 87.281911540 10.10.10.64                    10.10.10.72        SMB2     167 Session Setup Response
1272… 87.282869414 10.10.10.72                    10.10.10.64        SMB2     172 Tree Connect Request Tree: \\10.10.10.64\Yes
1272… 87.283489753 10.10.10.64                    10.10.10.72        SMB2     146 Tree Connect Response
1272… 87.283904760 10.10.10.72                    10.10.10.64        SMB2     422 Create Request File: Toy.Story.4.2019.1080p.BluRay.x264-[YTS.LT].mp4
1272… 87.283912969 10.10.10.72                    10.10.10.64        SMB2     186 Ioctl Request FSCTL_QUERY_NETWORK_INTERFACE_INFO
1272… 87.284225194 10.10.10.64                    10.10.10.72        TCP       68 445 → 25502 [ACK] Seq=981 Ack=1564 Win=2100992 Len=0
1272… 87.284377499 10.10.10.64                    10.10.10.72        SMB2     298 Create Response File: Toy.Story.4.2019.1080p.BluRay.x264-[YTS.LT].mp4
1272… 87.284384454 10.10.10.64                    10.10.10.72        SMB2     938 Ioctl Response FSCTL_QUERY_NETWORK_INTERFACE_INFO
1272… 87.284583549 10.10.10.72                    10.10.10.64        TCP       68 25502 → 445 [ACK] Seq=1564 Ack=2093 Win=2102272 Len=0
```

Figure 7: SMB2 Packet capture

References:

CaptureSetup/VLAN - The Wireshark Wiki. (n.d.). Retrieved April 30, 2021, from https://wiki.wireshark.org/CaptureSetup/VLAN#Linux

My Sniffer Isn't Seeing VLAN, 802.1q, or QoS Tagged Frames. (n.d.). Intel. Retrieved April 30, 2021, from https://www.intel.com/content/www/us/en/support/articles/000005498/network-and-i-o/ethernet-products.html

Seaman, M. (2003). *Large Scale Q-in-Q -- (1) Scalable address learning*. https://www.ieee802.org/1/files/public/docs2003/ScalableQinQLearning.pdf

Understanding and Configuring Spanning Tree Protocol (STP) on Catalyst Switches. (n.d.). Cisco. Retrieved April 30, 2021, from https://www.cisco.com/c/en/us/support/docs/lan-switching/spanning-tree-protocol/5234-5.html