Wireshark Capture and Port Mirroring

Debugging Walkthrough Guide

Our Support team often finds we are helping customers with unusual problems that are site-based and specific to their control network. In order to help us identify the symptoms and diagnose a solution, we need to have visibility to all network traffic that might be affecting the controllers, nodes or fixtures.

To help us understand what is going on, we will often ask for a "Wireshark trace" - which is extremely useful diagnostically, but can be tricky to set up. This paper covers this process.

Port Mirroring and Wireshark

Port mirroring is the process of setting a port on a switch to output the same data as other ports. This is useful for capturing unicast messages sent between two devices that are not the user's PC, allowing us to see the communication that is happening to a specific device and gives us a deeper understanding to what is being sent on the network.

For the purpose of debugging a project, we would expect the controller that is seeing an issue to be connected to the port that is being mirrored, and the PC running Wireshark should be connected to the mirrored port.

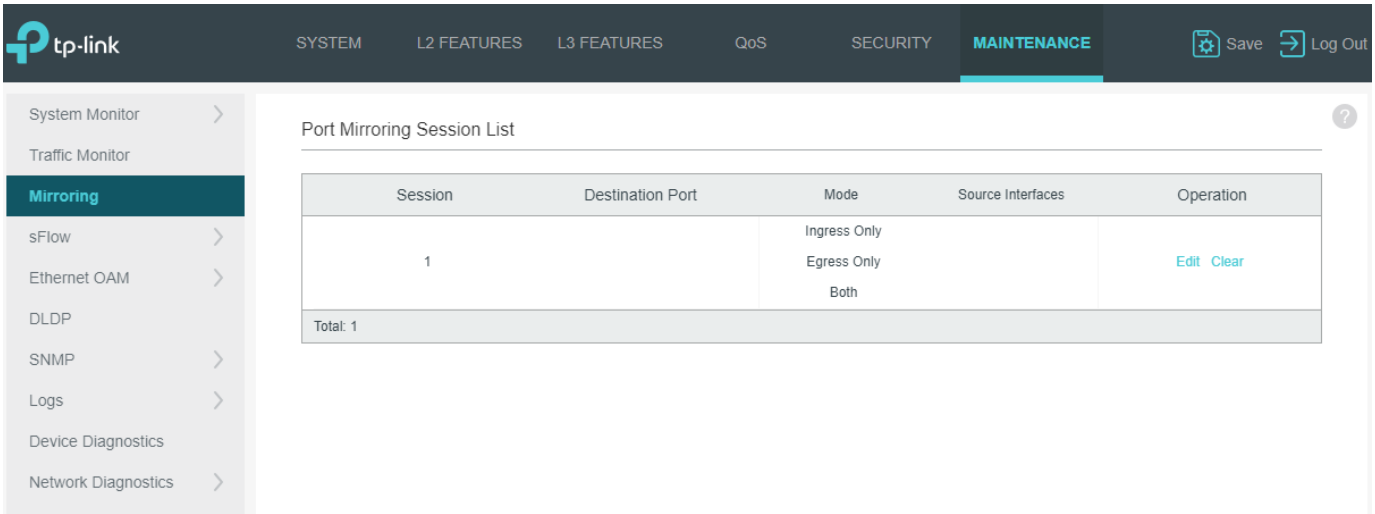
This guide will cover both setting up a mirrored switch and Wireshark, as well as a quick overview of the information that Wireshark provides us.

Port Mirroring

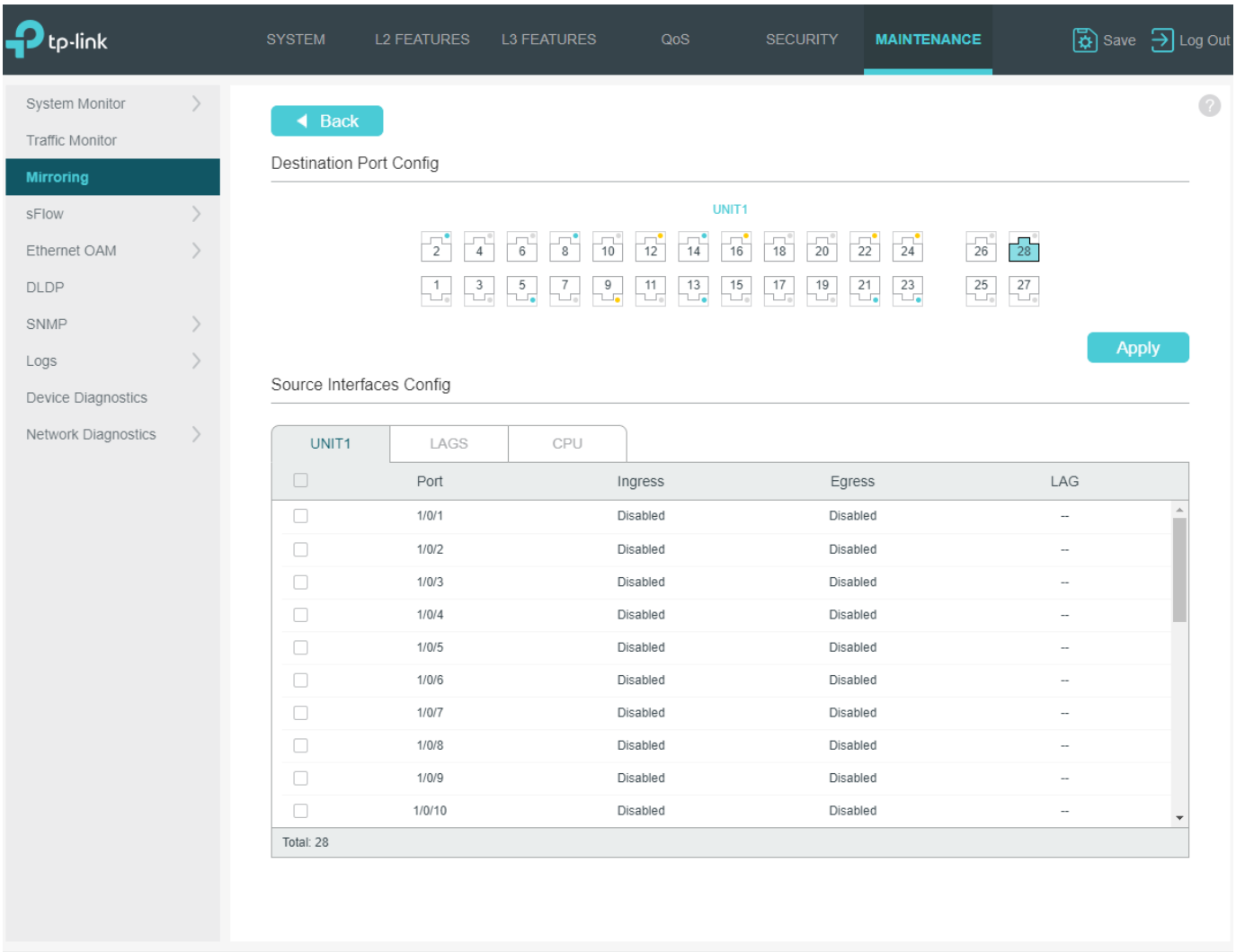
For starters, every brand of switch will have different methods for setting up a mirrored port. This is generally only possible on a managed switch and can be configured via the web interface of most of these switches. If you are very new to this type of configuration, it may be beneficial to have a network engineer to help you to set this up.

However, we will be covering the switch we use in our head office; while others will be set up slightly differently, they should generally follow a similar method.

The location of the mirror setup will be different for every switch, but for our example, this could be found in Maintenance > Mirroring. Please refer to your user manual if the location is not obvious.

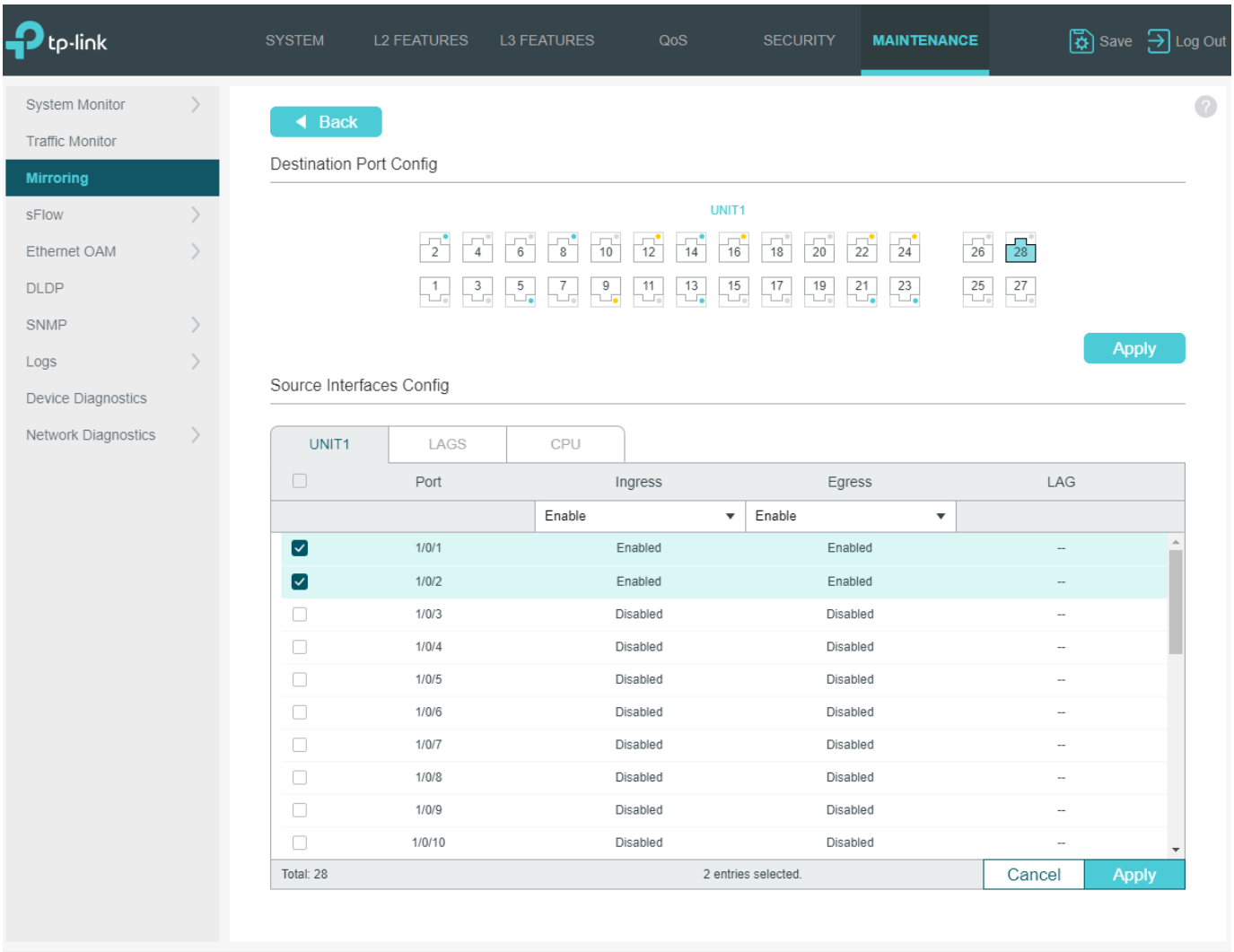


Next, you will need to select the port of your switch to which you will be connecting your monitoring PC. In our example, this is port 28.



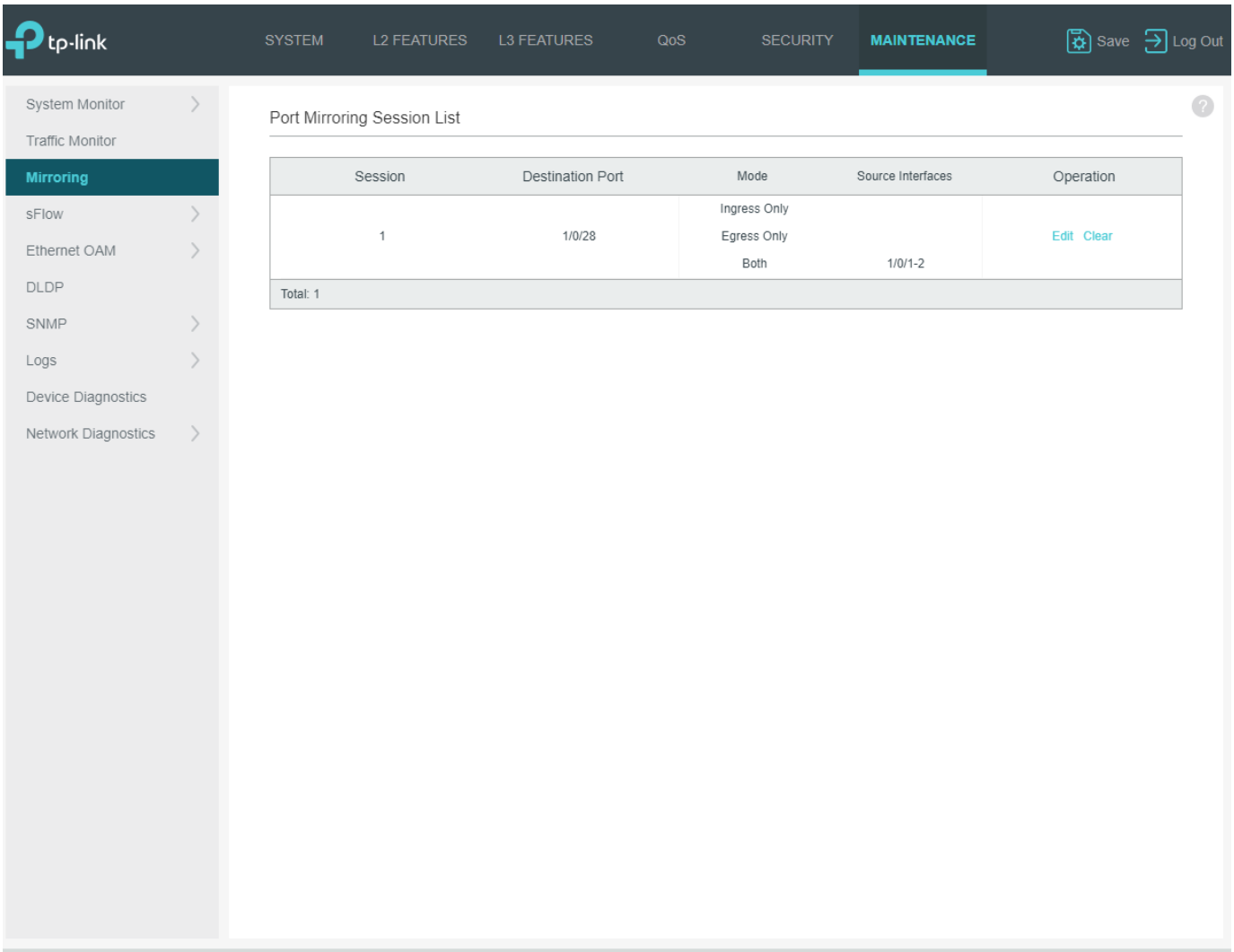
With this, you can either select the current port your PC is connected to or select a port that can be easily marked and left as a mirrored port, as this functionality is often useful for debugging or generalised testing.

Next up, you will need to select the ports that you want to be mirrored. These ports will then send the same data that travels through them to the port you picked in the last step. For our example, we can pick either “ingress” and/or “egress. Ingress is defined as the data being received by the switch, so this would be the data that our controllers are sending out. Egress would be the inverse of that, so the data that the controller is receiving from the network. To aid in debugging issues, we would need a capture of both of these streams of data.

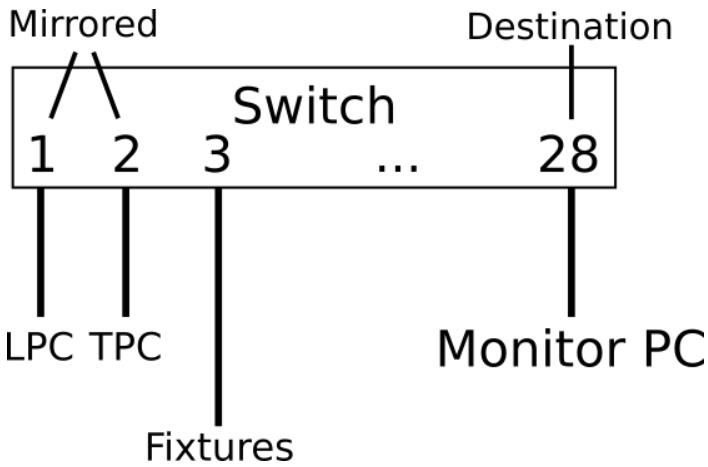


It is important to note here which port is selected, and which port on your controller the switch is connected to. If it is possible, to provide us with the most information possible, both ports of our rack-mounted units being connected to mirrored ports would be preferable. However, if you are debugging an issue to do with the eDMX fixtures not illuminating properly, then we would need to connect the data port to our mirrored switch. If the issue is to do with integration or the controller becoming unresponsive, then connecting the management port of the rack-mounted controllers would be more beneficial.

Once you have set up your mirrored port, ensure you apply your changes, then commit them to the switch. If there is a status or confirmation screen, ensure the data within it is correct, as shown below.



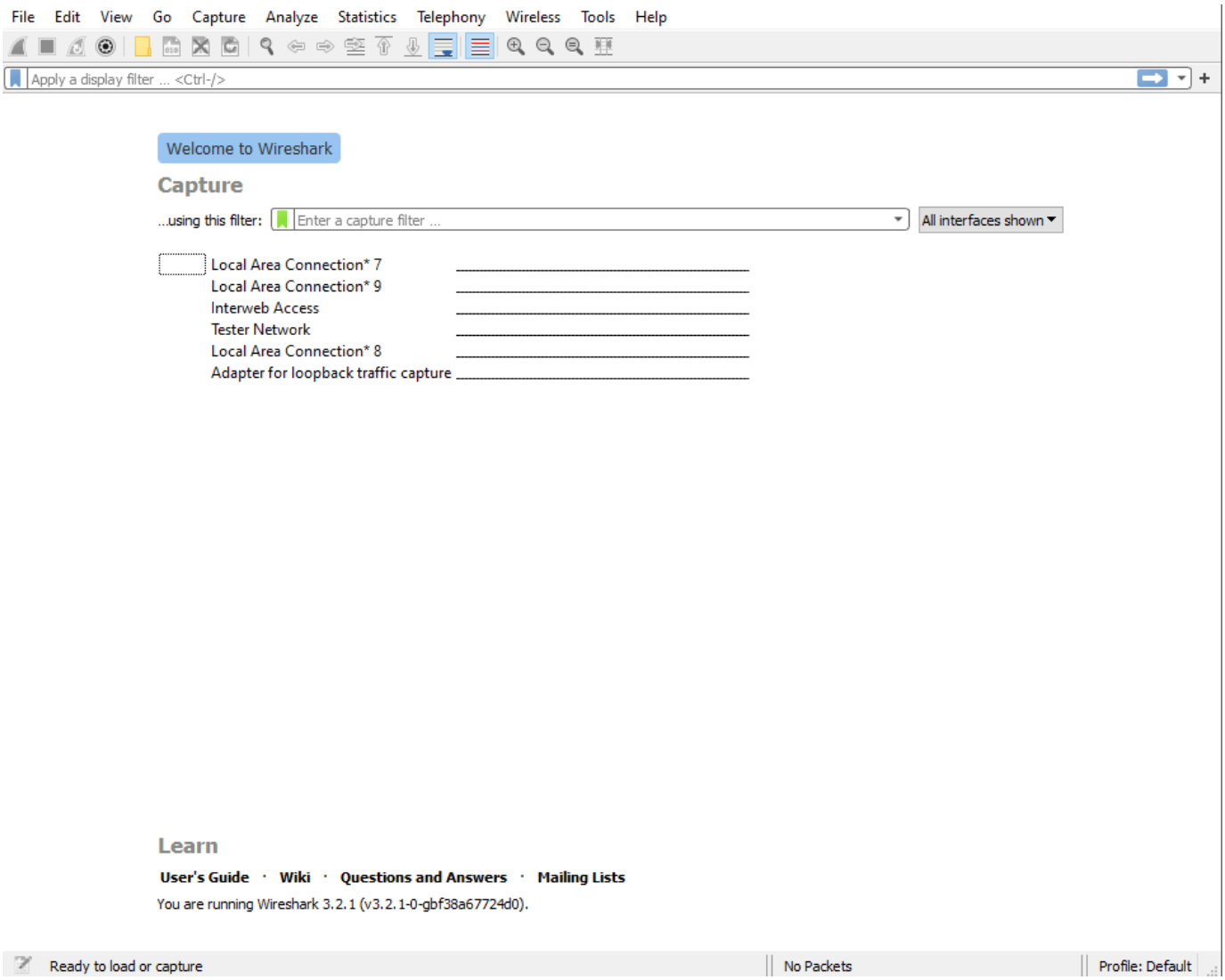
Below is a rough idea of how the above setup would then be set up in the real world. As you can see, two of my controllers are attached to the mirrored ports, and the monitoring PC is connected to the destination port as configured above. The “mirrored” ports will copy all their data to the “destination” port.



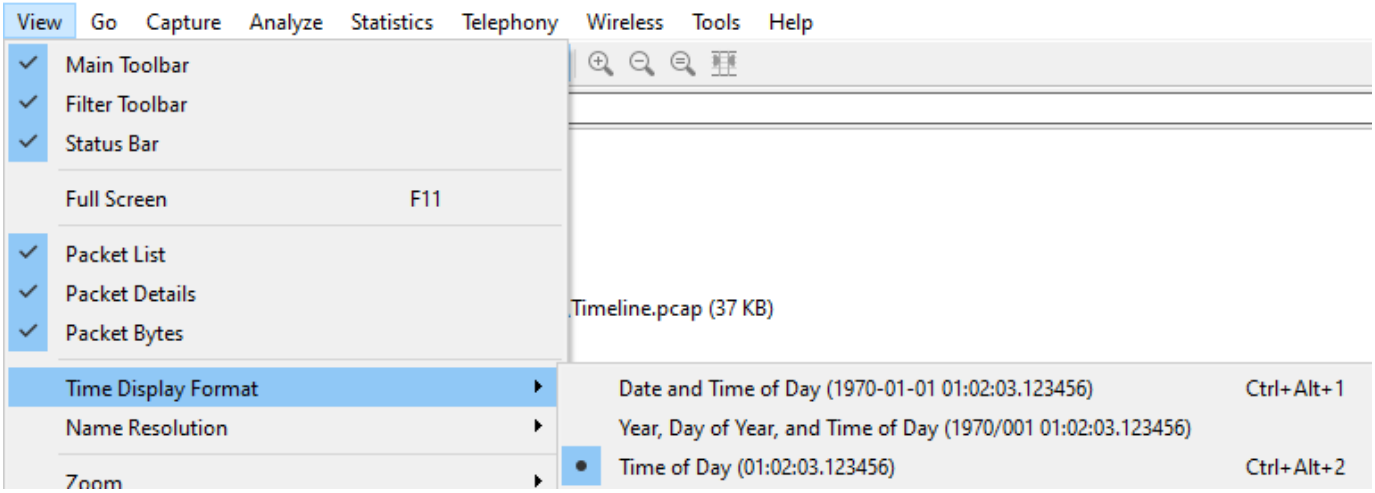
Wireshark Capturing

To start with, please download this software: https://www.wireshark.org/#download. Wireshark is an incredibly useful tool for detecting and decrypting network traffic. It will capture all broadcast, multicast and unicast messages that are received by your PC. To enable this to also detect all network traffic to your controllers, please set up port mirroring as stated above.

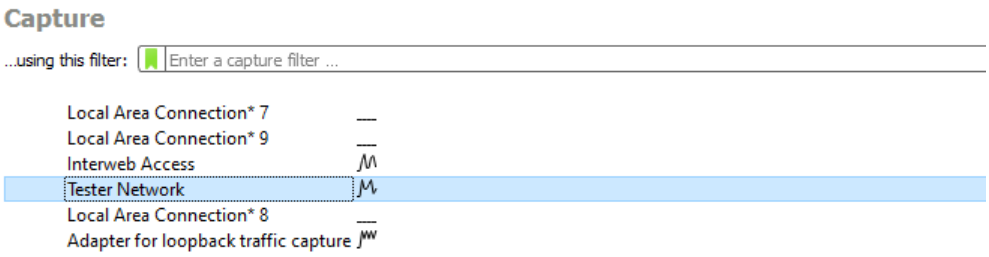
To start with, when opening Wireshark, you will see the following screen:



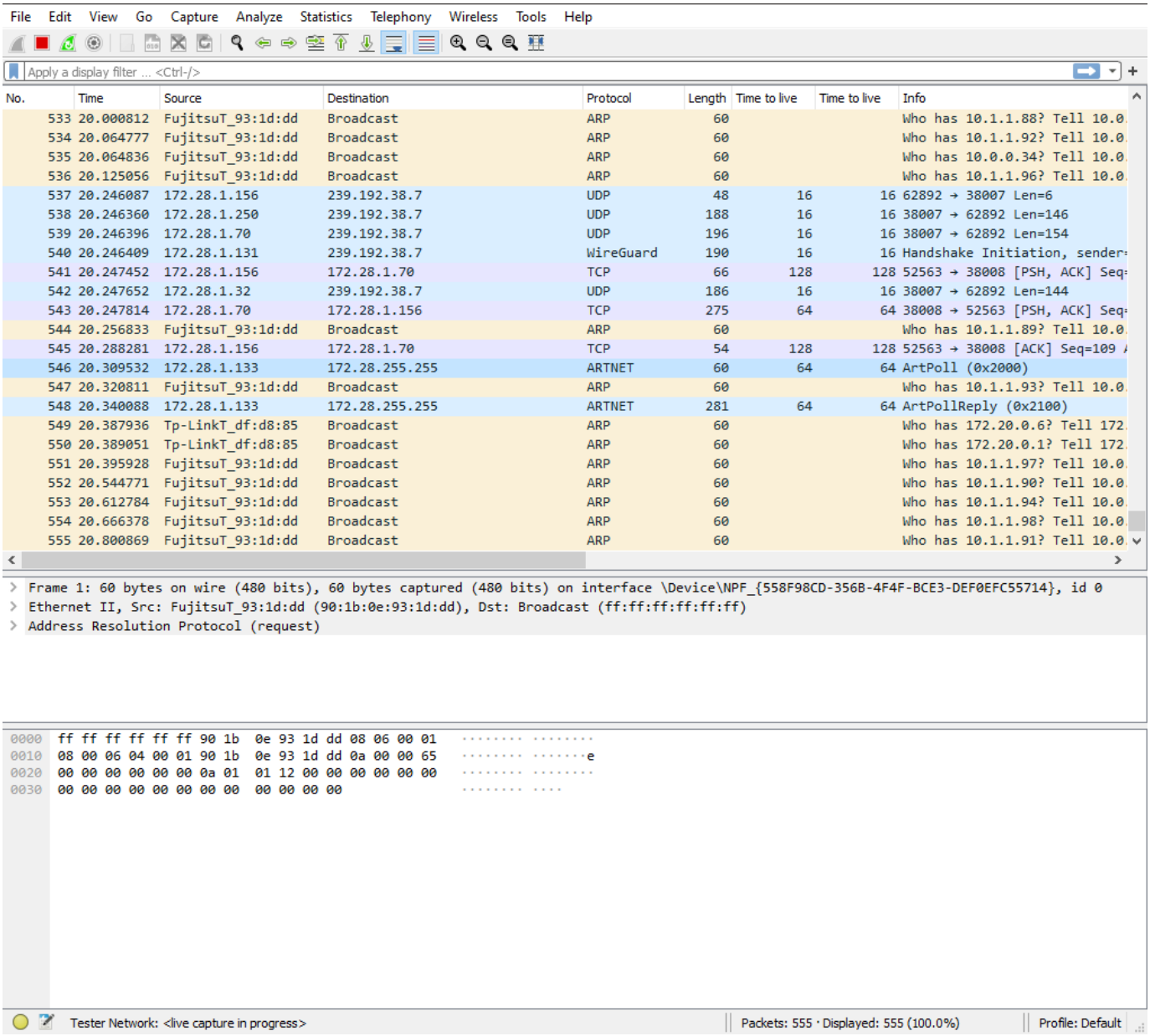
Before we start, a useful setting to change would be the time displayed. By default, this is set to seconds after the Wireshark started. This can be useful in some cases, but for most cases, knowing roughly what time the issue occurred at can be more helpful. To change this, follow the settings as shown in the following screenshot.



Once this has been done, double-click on the network that is the same as the controller. In this example, that would be the “Tester Network”.

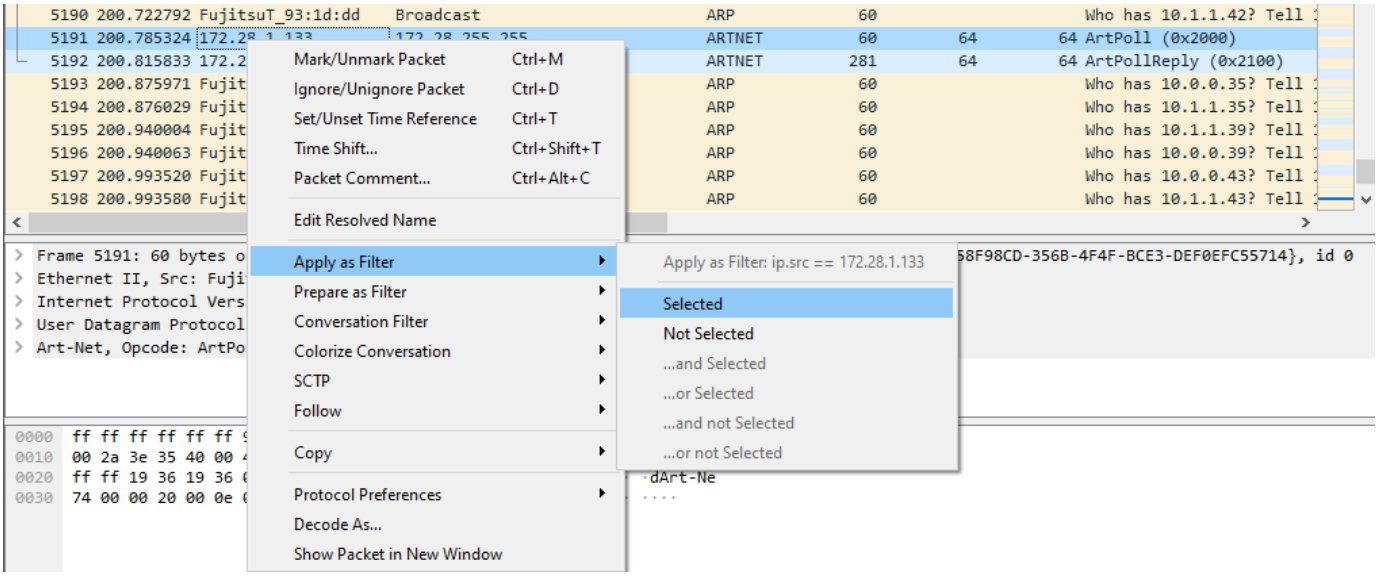


The network table will now appear and show all data that is being sent, as seen below.

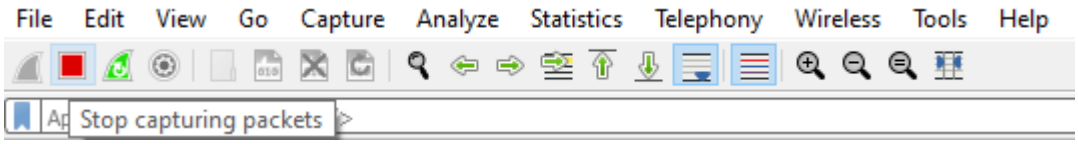


No.	Time	Source	Destination	Protocol	Length	Time to live	Time to live	Info
533	20.000812	FujitsuT_93:1d:dd	Broadcast	ARP	60			Who has 10.1.1.88? Tell 10.0.
534	20.064777	FujitsuT_93:1d:dd	Broadcast	ARP	60			Who has 10.1.1.92? Tell 10.0.
535	20.064836	FujitsuT_93:1d:dd	Broadcast	ARP	60			Who has 10.0.0.34? Tell 10.0.
536	20.125056	FujitsuT_93:1d:dd	Broadcast	ARP	60			Who has 10.1.1.96? Tell 10.0.
537	20.246087	172.28.1.156	239.192.38.7	UDP	48	16	16	62892 → 38007 Len=6
538	20.246360	172.28.1.250	239.192.38.7	UDP	188	16	16	38007 → 62892 Len=146
539	20.246396	172.28.1.70	239.192.38.7	UDP	196	16	16	38007 → 62892 Len=154
540	20.246409	172.28.1.131	239.192.38.7	WireGuard	190	16	16	Handshake Initiation, sender=
541	20.247452	172.28.1.156	172.28.1.70	TCP	66	128	128	52563 → 38008 [PSH, ACK] Seq=
542	20.247652	172.28.1.32	239.192.38.7	UDP	186	16	16	38007 → 62892 Len=144
543	20.247814	172.28.1.70	172.28.1.156	TCP	275	64	64	38008 → 52563 [PSH, ACK] Seq=
544	20.256833	FujitsuT_93:1d:dd	Broadcast	ARP	60			Who has 10.1.1.89? Tell 10.0.
545	20.288281	172.28.1.156	172.28.1.70	TCP	54	128	128	52563 → 38008 [ACK] Seq=109 A
546	20.309532	172.28.1.133	172.28.255.255	ARTNET	60	64	64	ArtPoll (0x2000)
547	20.320811	FujitsuT_93:1d:dd	Broadcast	ARP	60			Who has 10.1.1.93? Tell 10.0.
548	20.340088	172.28.1.133	172.28.255.255	ARTNET	281	64	64	ArtPollReply (0x2100)
549	20.387936	Tp-LinkT_df:d8:85	Broadcast	ARP	60			Who has 172.20.0.6? Tell 172.
550	20.389051	Tp-LinkT_df:d8:85	Broadcast	ARP	60			Who has 172.20.0.1? Tell 172.
551	20.395928	FujitsuT_93:1d:dd	Broadcast	ARP	60			Who has 10.1.1.97? Tell 10.0.
552	20.544771	FujitsuT_93:1d:dd	Broadcast	ARP	60			Who has 10.1.1.90? Tell 10.0.
553	20.612784	FujitsuT_93:1d:dd	Broadcast	ARP	60			Who has 10.1.1.94? Tell 10.0.
554	20.666378	FujitsuT_93:1d:dd	Broadcast	ARP	60			Who has 10.1.1.98? Tell 10.0.
555	20.800869	FujitsuT_93:1d:dd	Broadcast	ARP	60			Who has 10.1.1.91? Tell 10.0.

To filter out data that is not relevant to you, you can use the filter function at the top of the screen. This can either be added through a context menu, as seen below, or manually, via typing in “ip.src == 172.28.1.133 || ip.dst == 172.28.1.133”. This will hide all other data apart from data being sent from that IP address or to that IP address (in this case, an LPC X in our office).



Once you have captured the issue, or for a given amount of time depending on the request from Support, you can stop the capture by clicking the following button in the toolbar:



Once the capture has been stopped, it can be saved and the resulting .pcapng file can be sent directly to Support.

To enable easier debugging, please send any relevant IP addresses, such as the IP address of your controllers, the IP address of any integrated device, your PC IP and ranges of IP for your eDMX nodes.