First Encounters with the ProfiTap-1G

Contents

Introduction

I recently had a chance to try out a ProfiTap-1G, a new packet capture Tap from the ProfiTap folks. Here is a record of my experience. I document this experience using Windows and Wireshark. In addition, this Tap also ships with Linux drivers and support for a range of commercial analyzers (OmniPeek, OptiView, many others).

Overview

The ProfiTap-1G is a hand-held device with two Ethernet ports and one USB port. As with any Tap, we insert it in-line with the Host-of-Interest (where some problem is occurring), and then the Tap forwards all traffic traversing it to our analyzer.

The Tap appears as just another NIC on your computer.

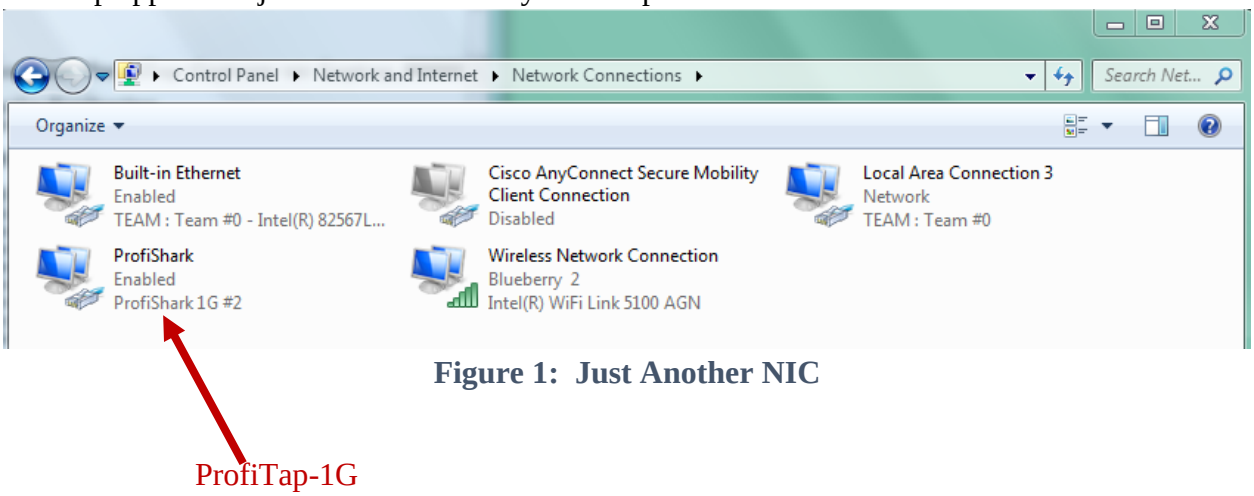


Figure 1: Just Another NIC

Dumpcap sees it as just another NIC.

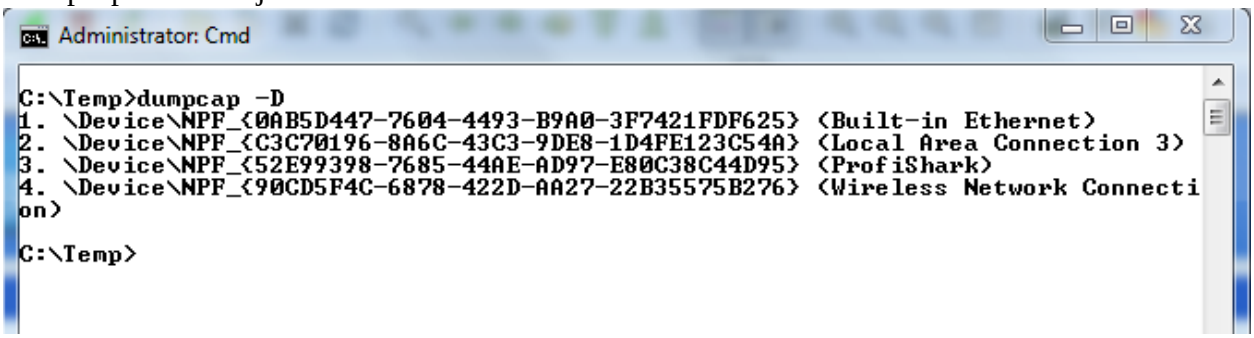


Figure 2: Dumpcap NIC List

Once inside Wireshark, the Tap continues to appear as just another NIC.

Figure 3: Wireshark Start Capture List

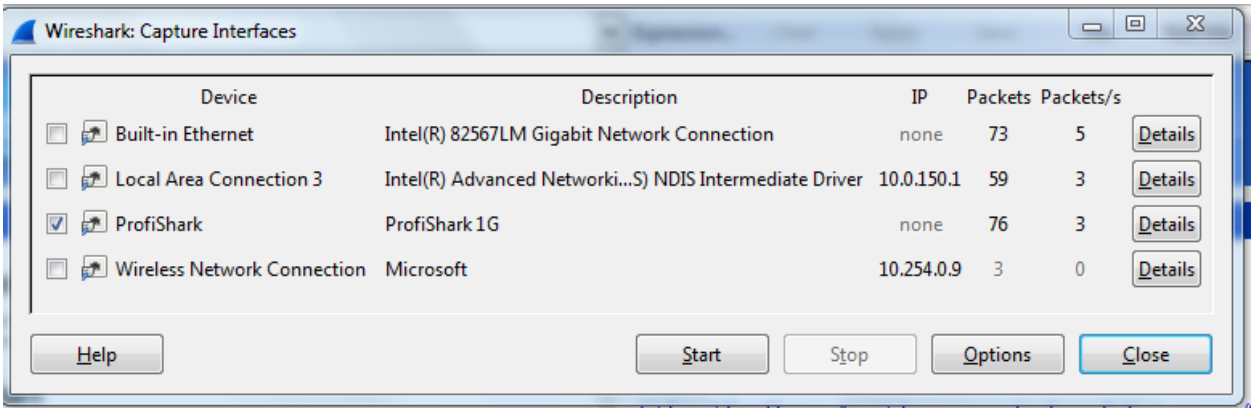


Figure 4: Wireshark Interface List

The Tap ships with a supporting application which allows you to configure its in-line functionality.

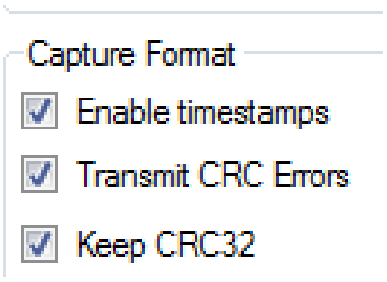


Figure 5: ProfiTap-1G Capture Format options

- *Enable timestamps* invokes the Tap's on-board clock to deliver timestamps with 8ns resolution.
- *Transmit CRC Errors* instructs the Tap to forward Ethernet frames whose CRC trailers do not correctly summarize the frame's contents.
- *Keep CRC32* instructs the Tap to retain the trailing 4 byte CRC on the Ethernet, as the Tap forwards the frame across its USB port and down to our analyzer.

The Tap itself can capture at line-rate, and its USB 3.0 port can easily accommodate the theoretical Gigabit Ethernet maximum of 2Gb/s (1Gb/s transmit plus 1Gb/s receive). Whether your analyzer can actually swallow 2Gb/s of frames arriving across its USB port depends, of course, on the IO capabilities of your analyzer.

Hardware

Here is what the ProfiTap-1G looks like in action. I am using my laptop not only as the Host-of-Interest but also as the analyzer.

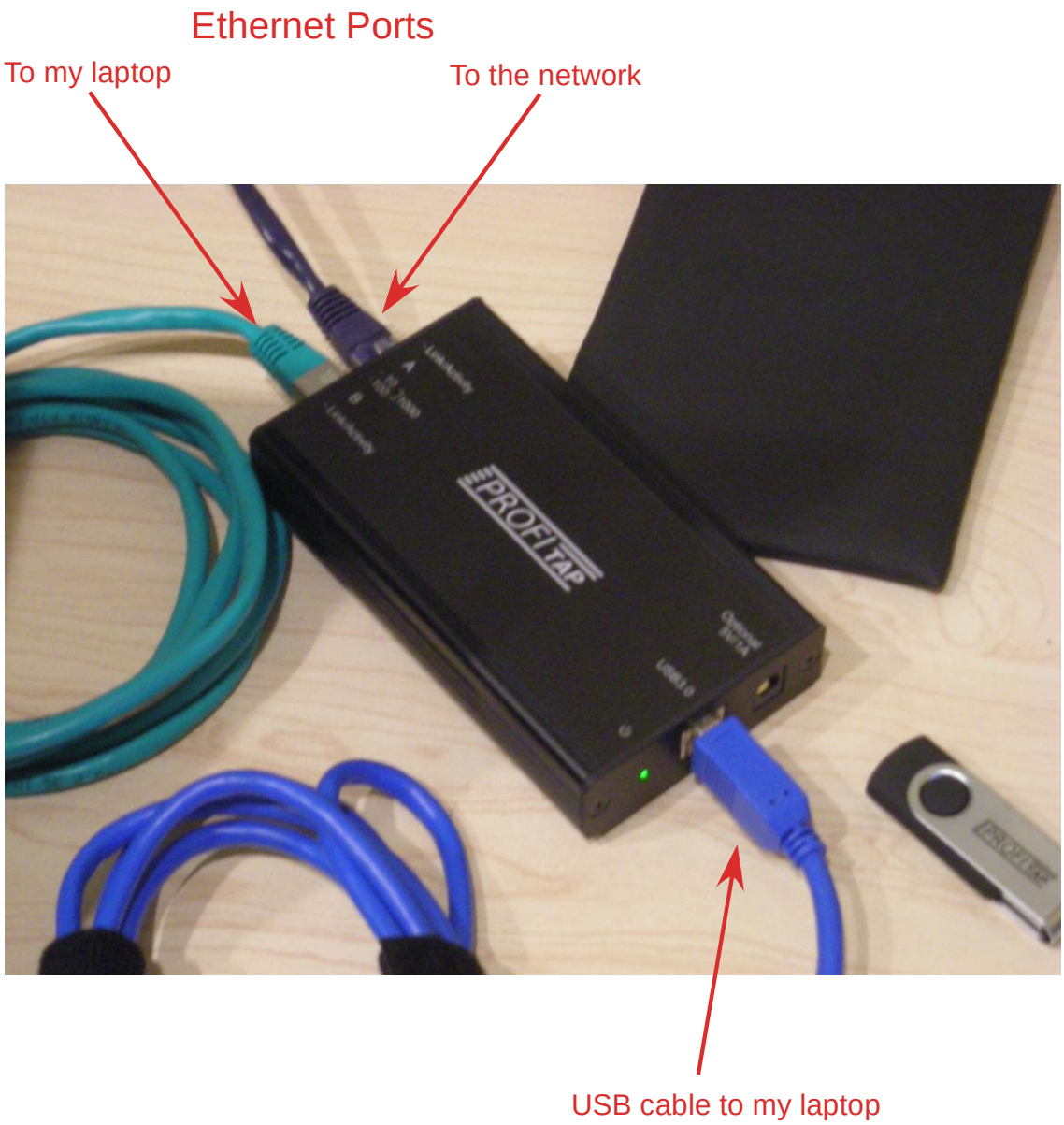


Figure 6: Focus on ProfiTap-1G

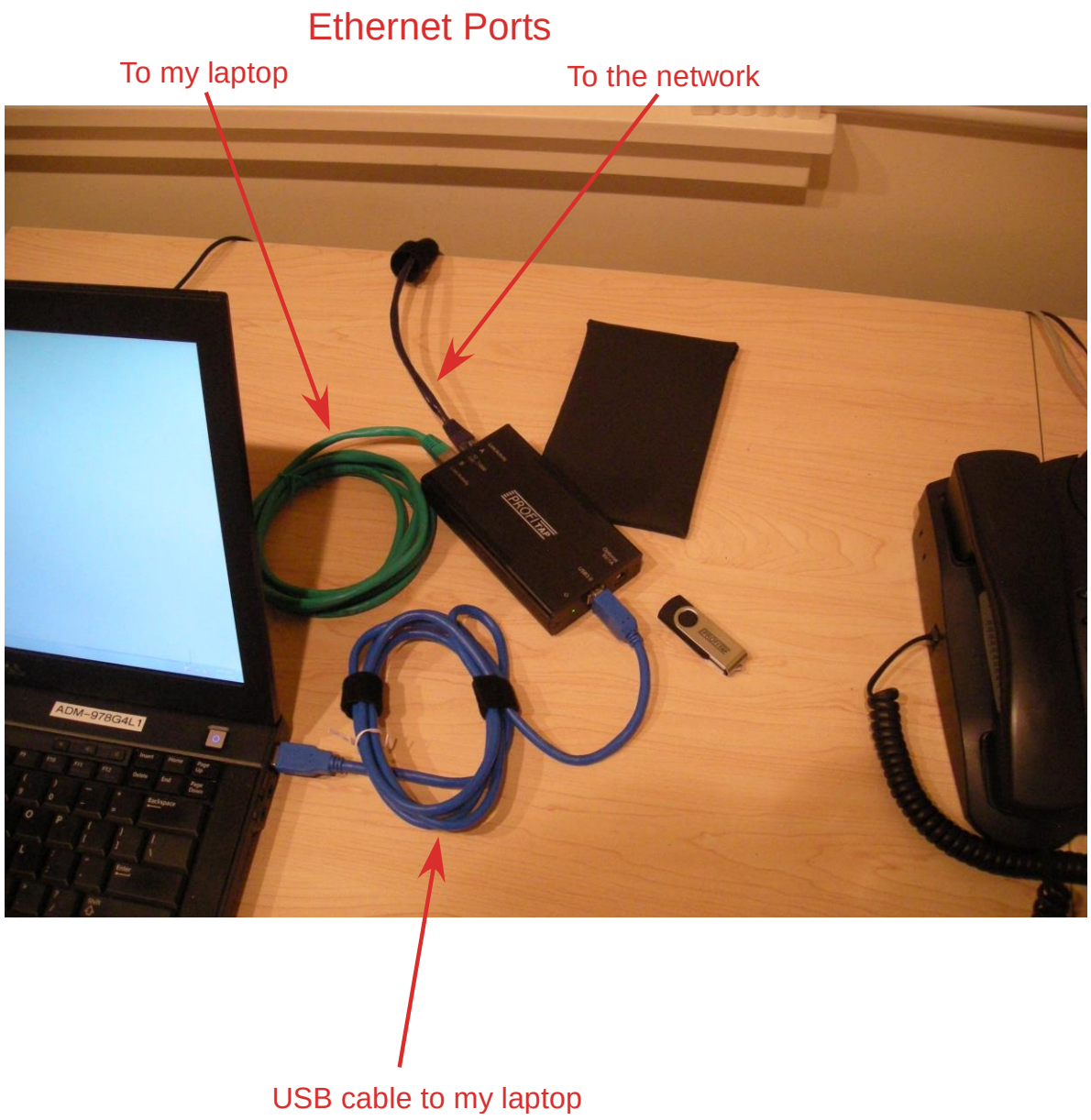


Figure 7: Laptop plus ProfiTap-1G

Installation

Installing the software begins with the usual Installer program.

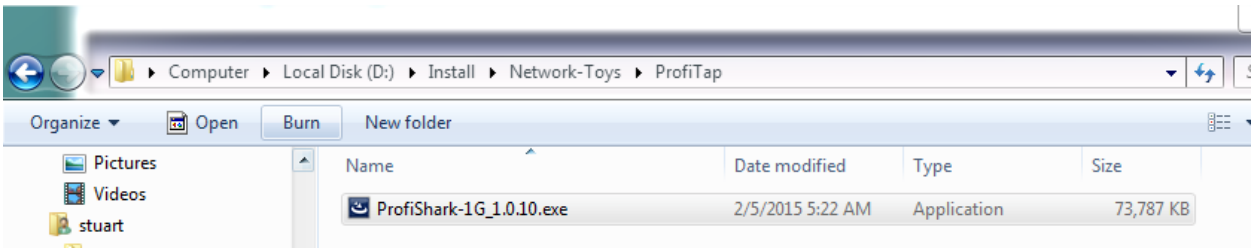


Figure 8: Windows Installer

And the resulting InstallShield Wizard.

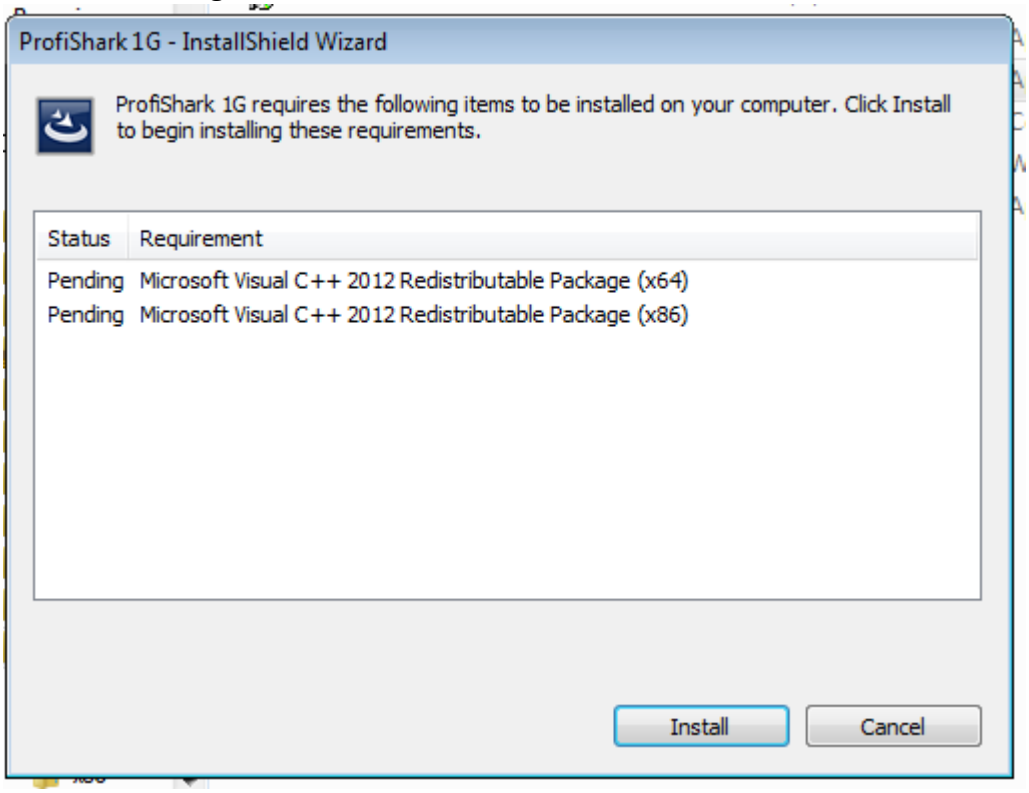


Figure 9: Install Visual C++

Once that finishes, installation progresses as usual.

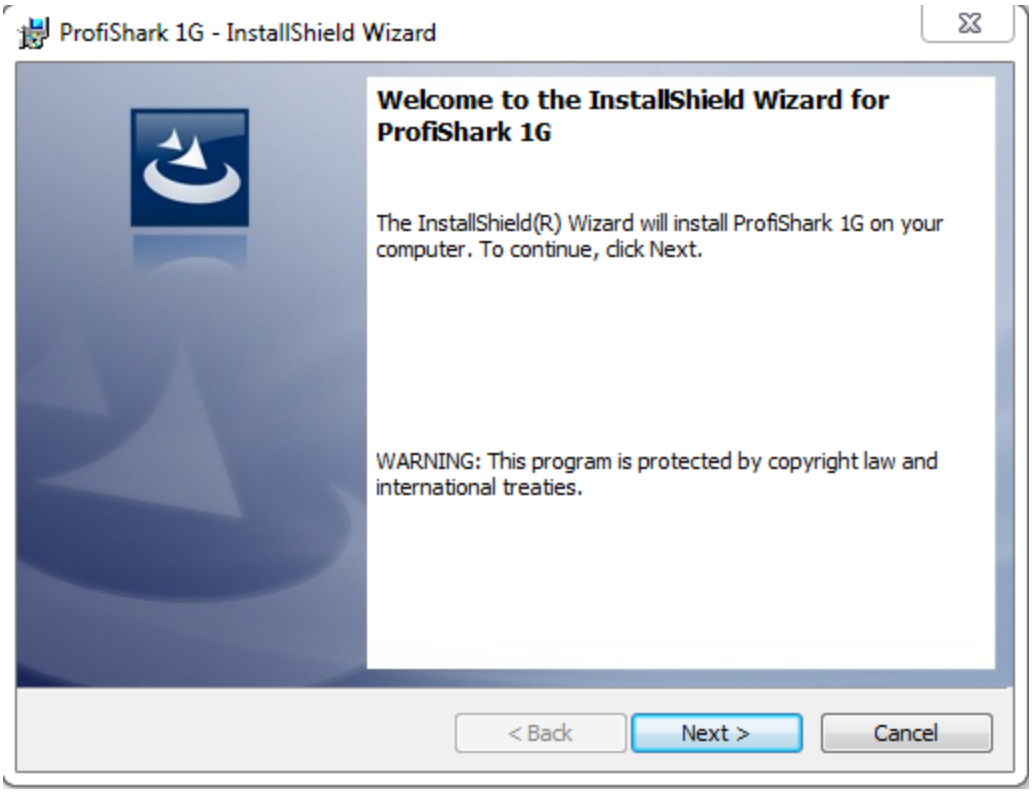
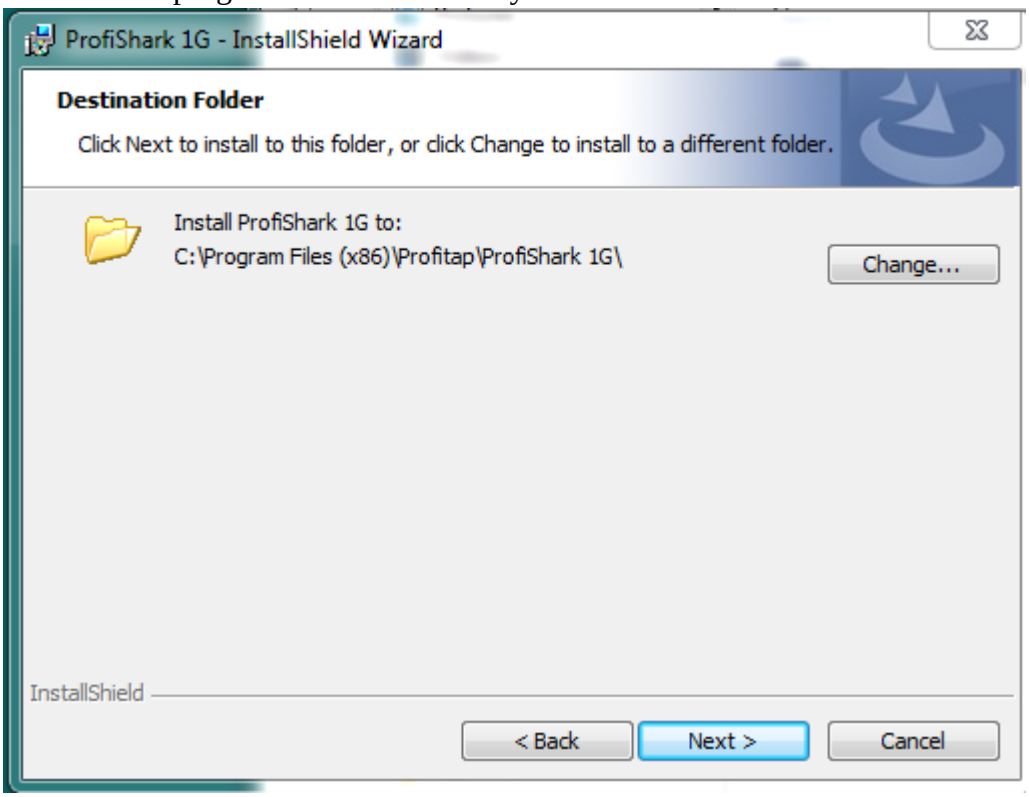


Figure 10: InstallShield Wizard

The Installer progresses in the usual way:



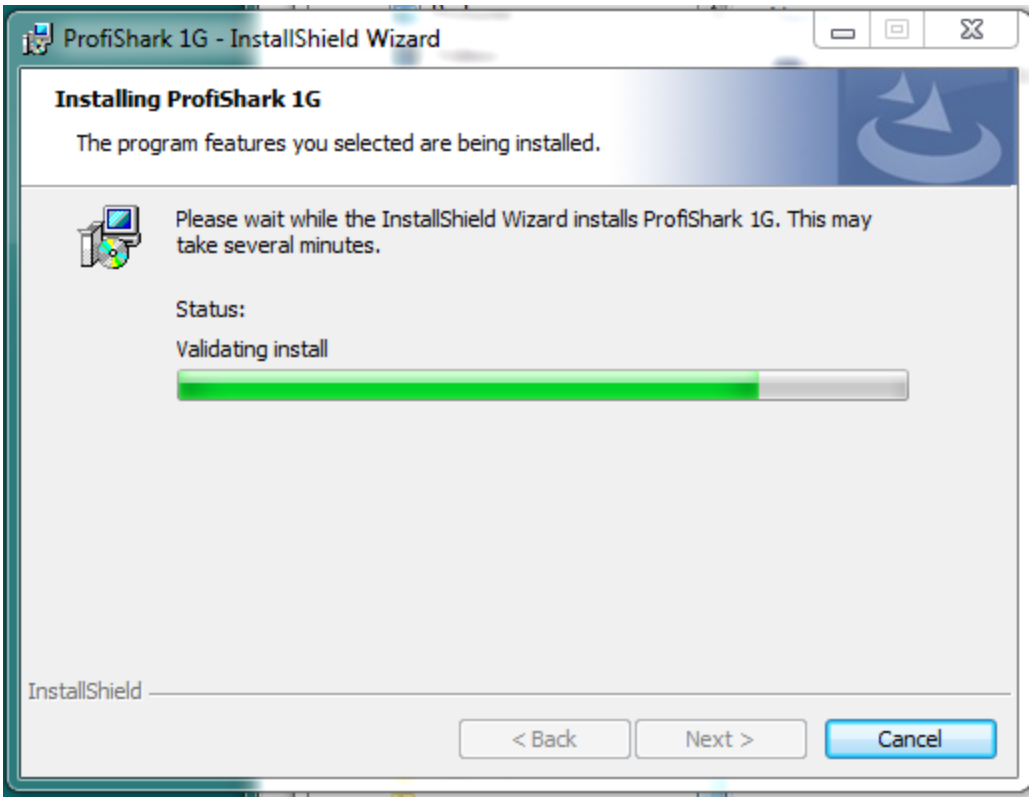


Figure 11: Validating Install

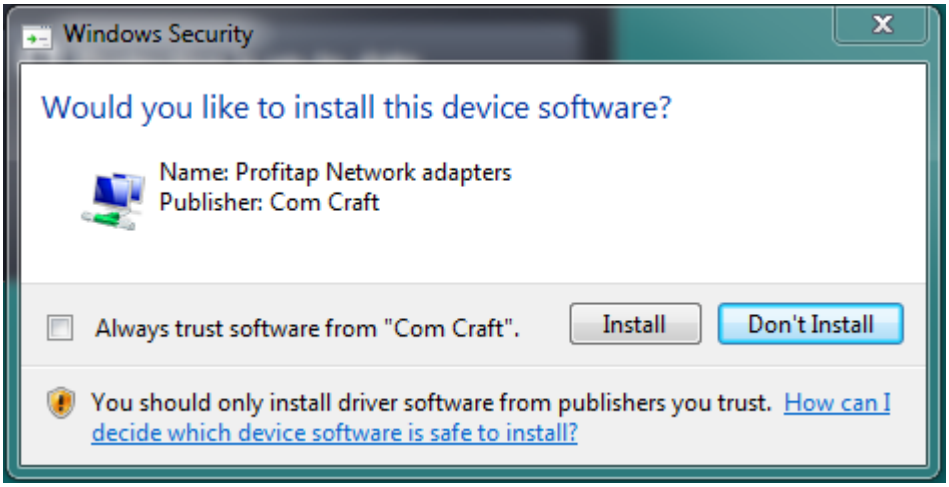


Figure 12: Install Network Driver

If you see one of these File in Use dialogue boxes, I recommend selecting the "Do not close applications. (A reboot will be required.)", as I ran into trouble taking the "Automatically close and attempt to restart applications" approach (I had to re-install the affected applications in order to restore functionality).

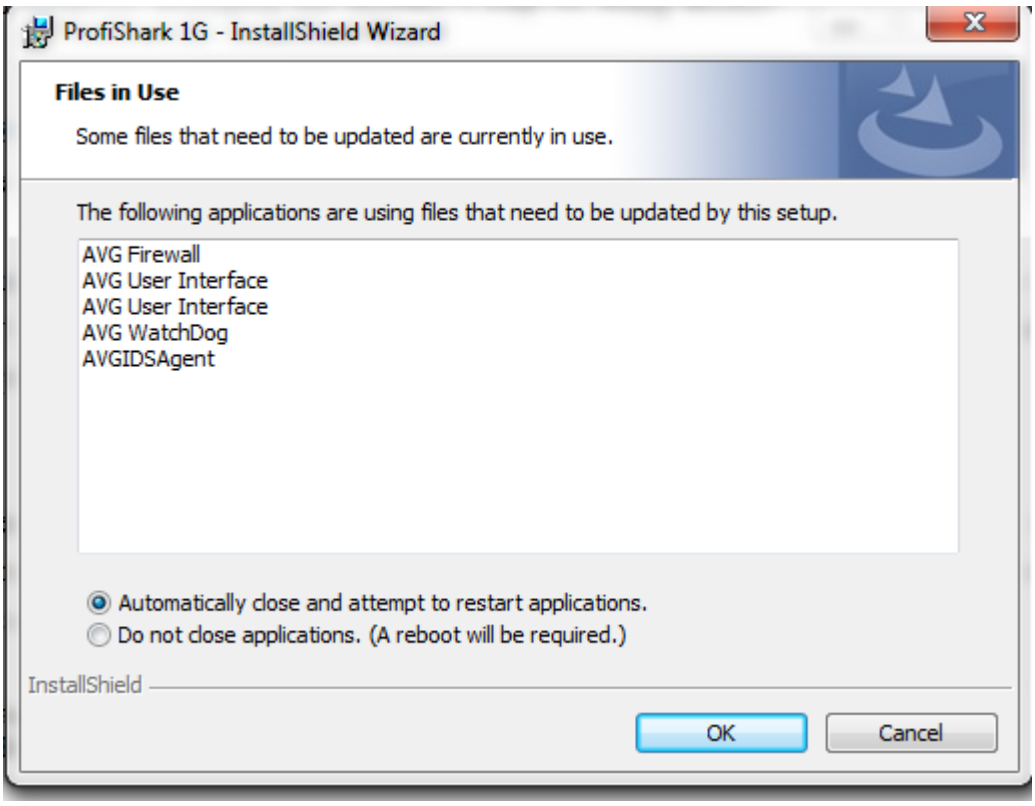


Figure 13: Files in Use

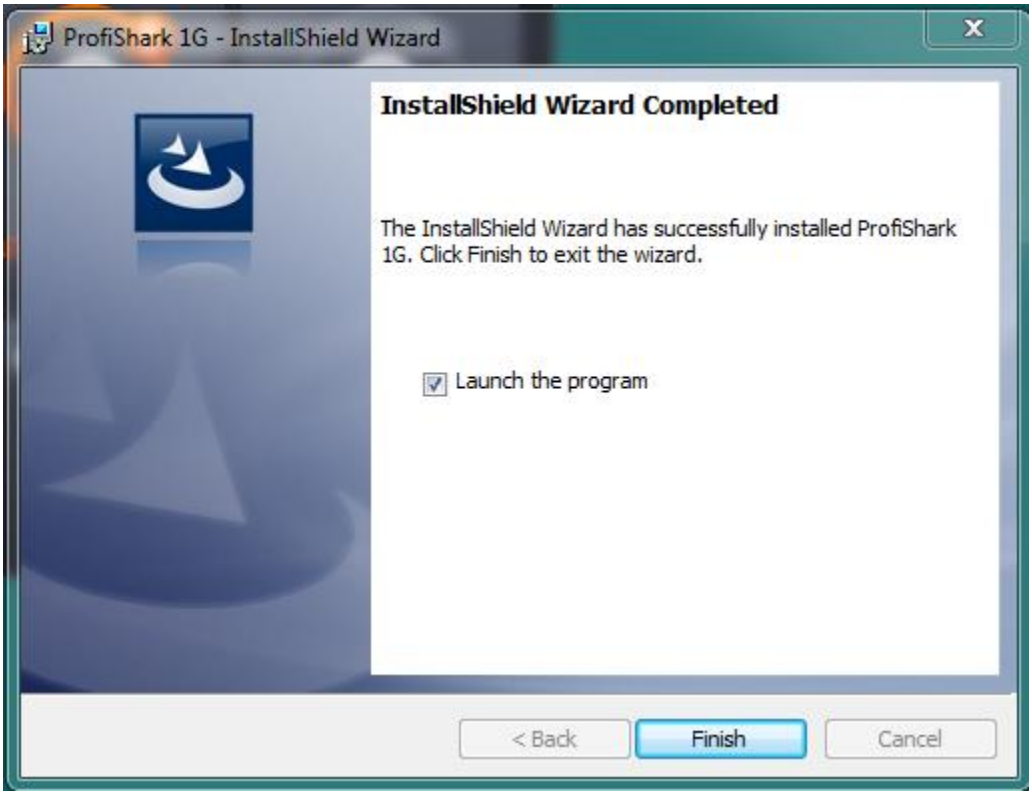


Figure 14: Finished

At this point, reboot, to allow the install to finish.

Finally, manually copy the ProfiShark-1G dissector *profishark_1g.dll* into your Wireshark plugins folder.

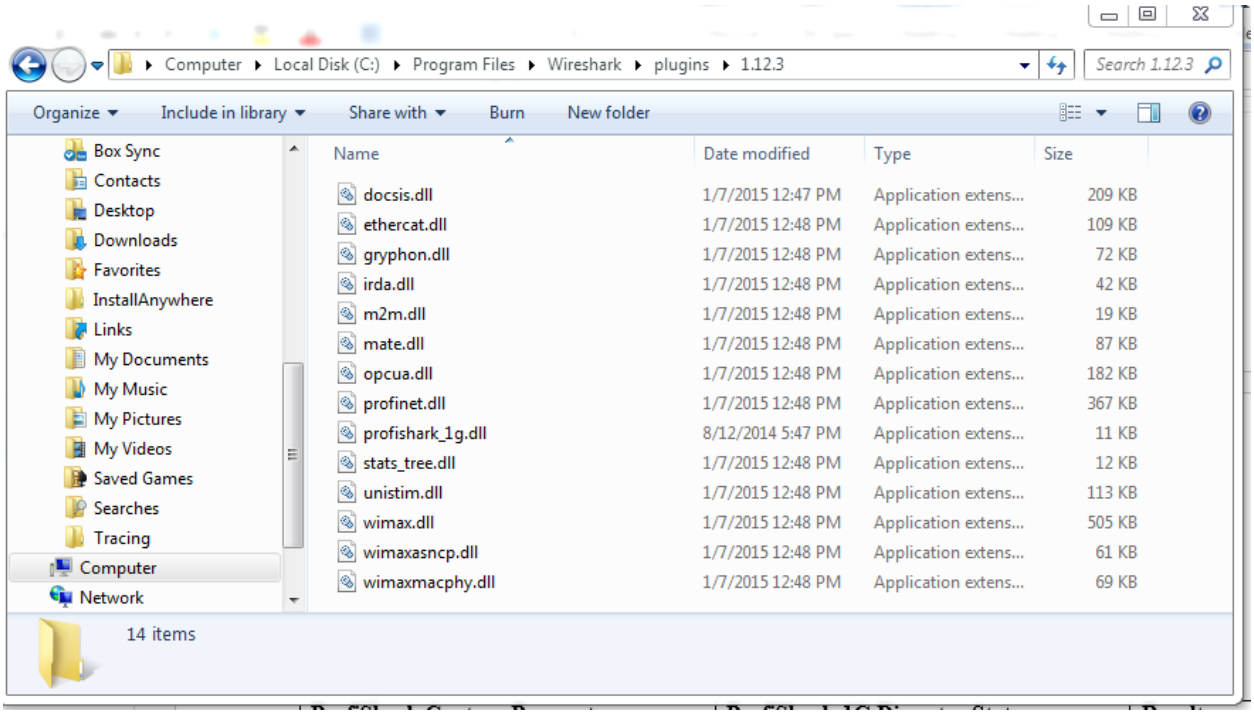
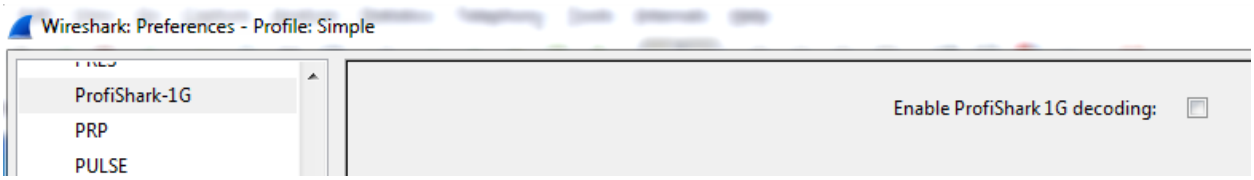


Figure 15: Copy the Dissector into Place

Copying *profishark_1g.dll* into place adds the ProfiShark-1G protocol to the Preferences... Protocols... list



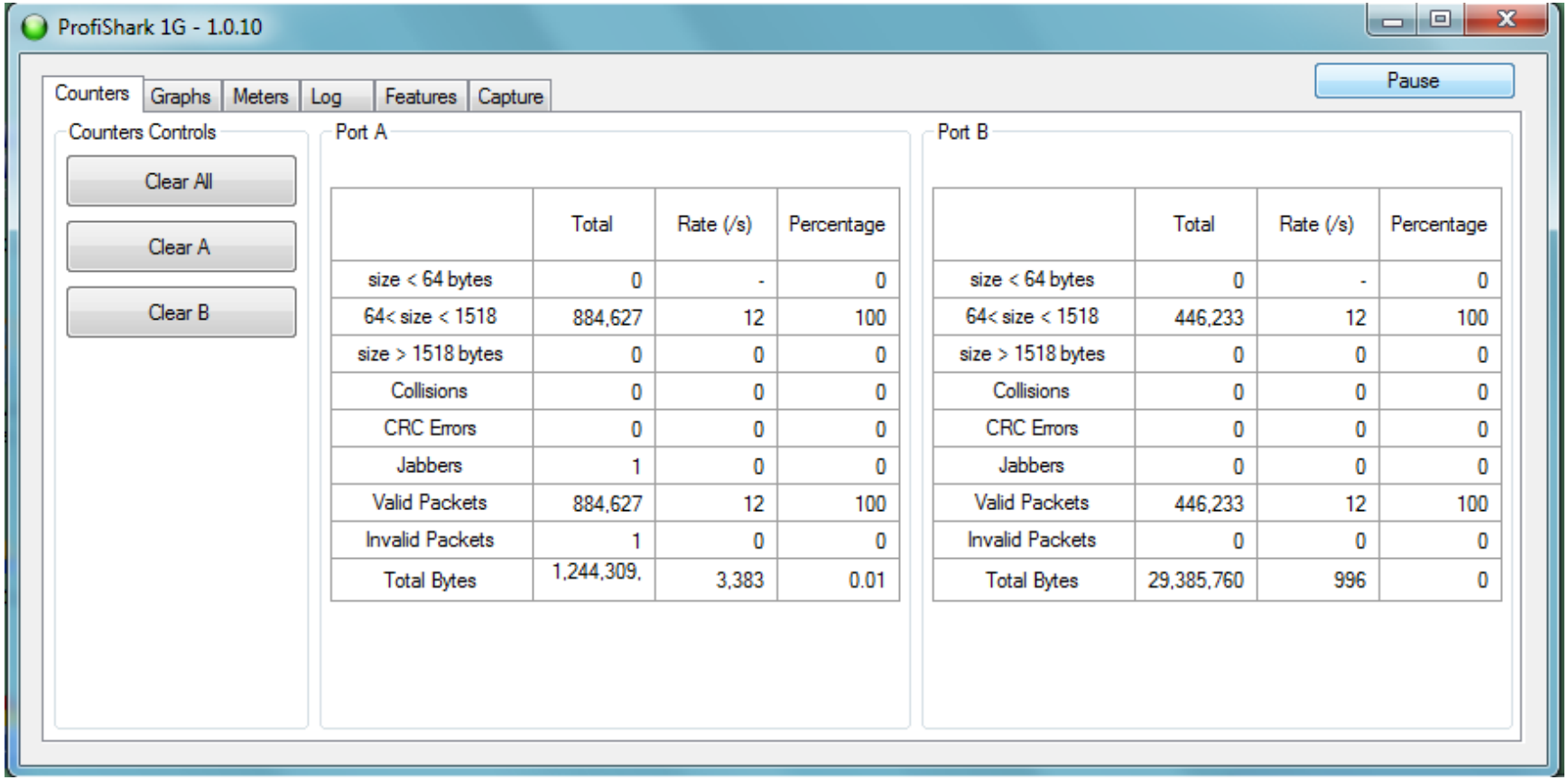
We'll come back to the utility of this dissector later. In the meantime, make sure that this box is unchecked; otherwise, Wireshark will attempt to apply it to your traces and will incorrectly dissect many protocols as a result (unless the trace you are analyzing was captured using the ProfiShark-1G; see below for details).

Talking to the ProfiTap-1G

Opening the newly-installed ProfiShark-1G application allows us to talk directly to the Tap.

Counters

The opening screen tabulates basic traffic statistics.



	Total	Rate (/s)	Percentage
size < 64 bytes	0	-	0
64< size < 1518	884,627	12	100
size > 1518 bytes	0	0	0
Collisions	0	0	0
CRC Errors	0	0	0
Jabbers	1	0	0
Valid Packets	884,627	12	100
Invalid Packets	1	0	0
Total Bytes	1,244,309,	3,383	0.01

	Total	Rate (/s)	Percentage
size < 64 bytes	0	-	0
64< size < 1518	446,233	12	100
size > 1518 bytes	0	0	0
Collisions	0	0	0
CRC Errors	0	0	0
Jabbers	0	0	0
Valid Packets	446,233	12	100
Invalid Packets	0	0	0
Total Bytes	29,385,760	996	0

Figure 16: Counters

Recall that because a Tap sits in-line with traffic, it can capture frames which Ethernet NIC drivers generally filter out, like Collisions, CRC Errors, and Jabbers.

Graphs

The next tab allows for semi-real-time display of the statistics visible in the Counters tab.

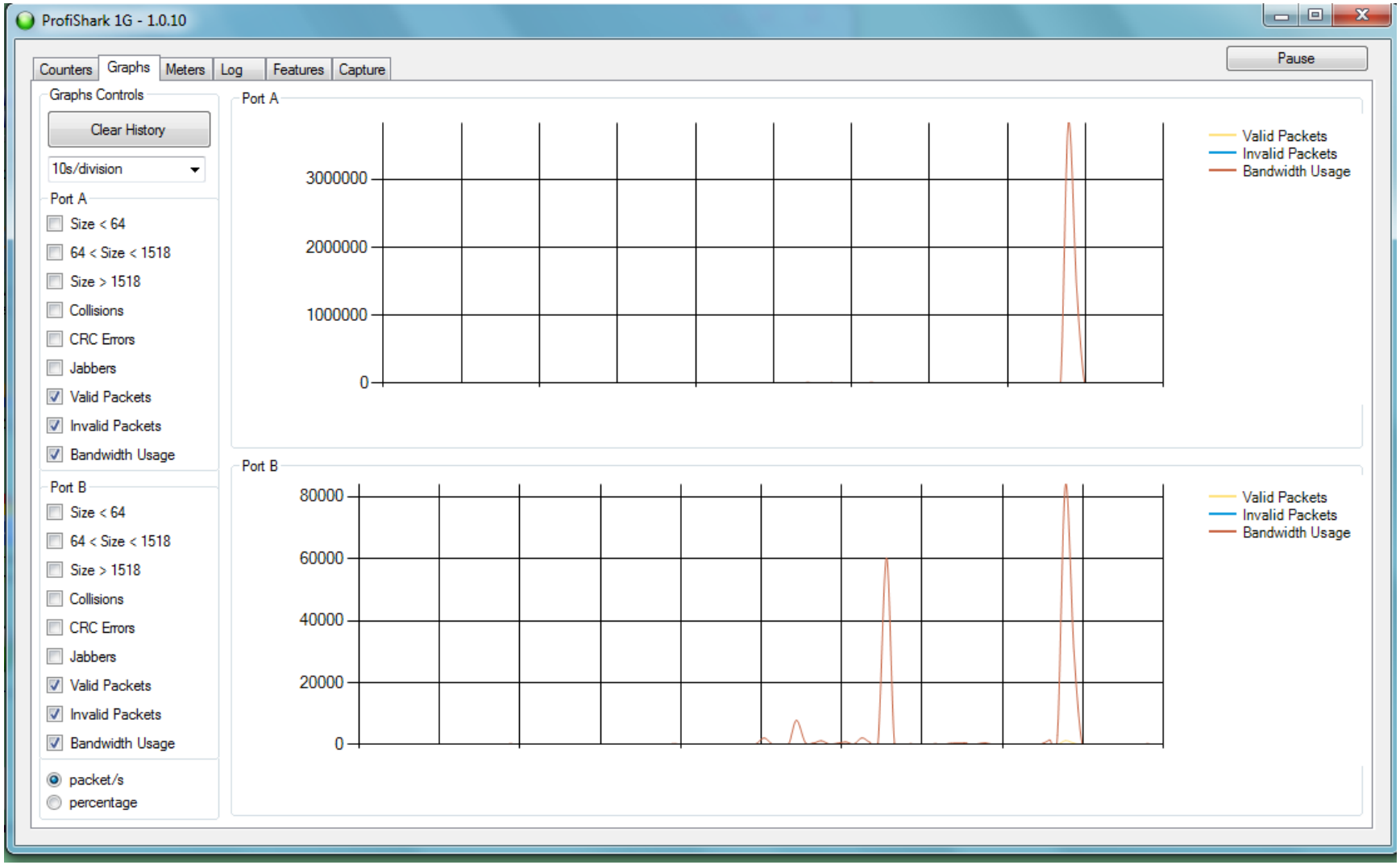


Figure 17: Graphs

Meters

The third tab uses an automobile dashboard metaphor to display utilization.

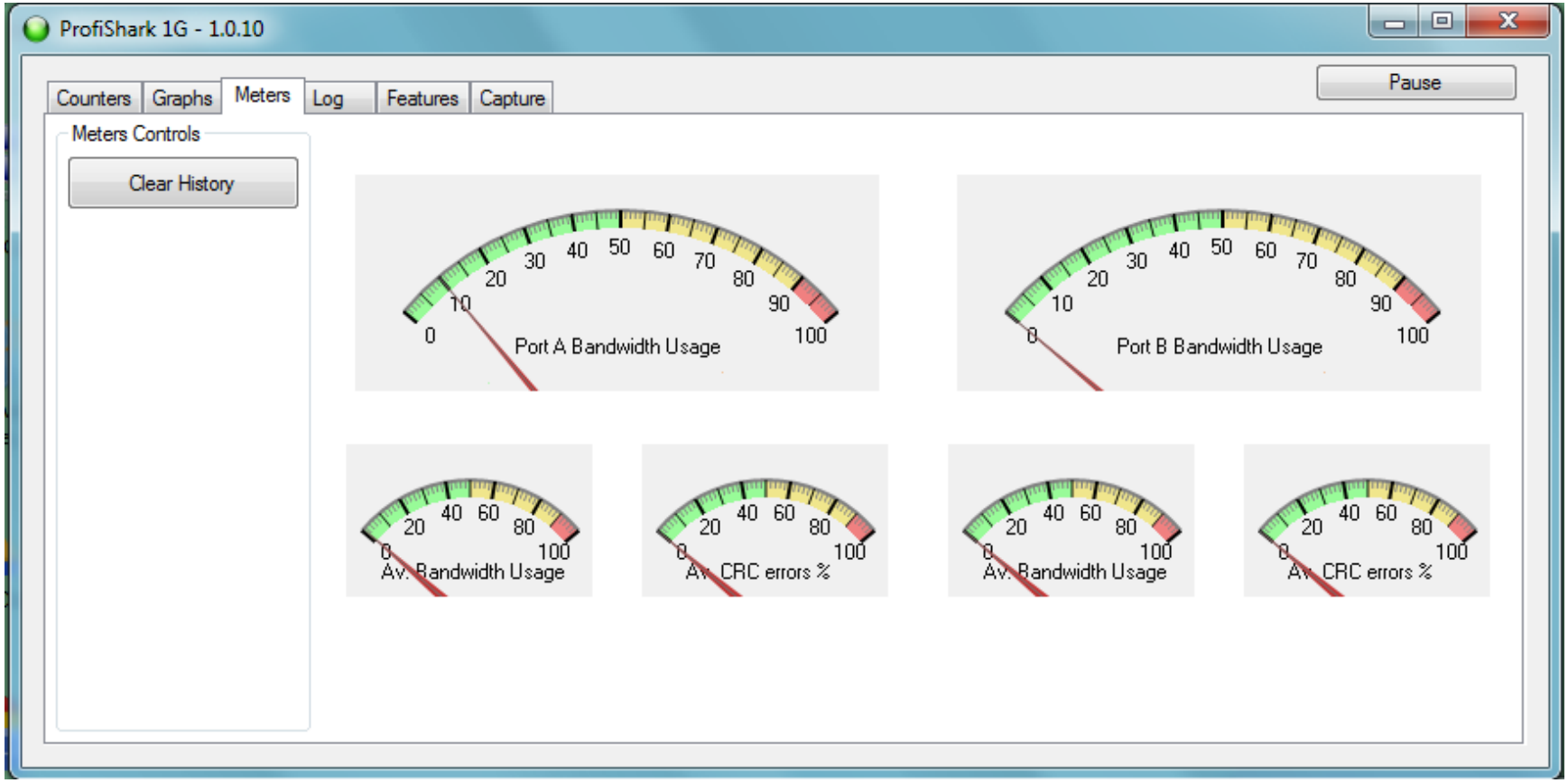


Figure 18: Meters

Log

The next tab allows us to record utilization events.

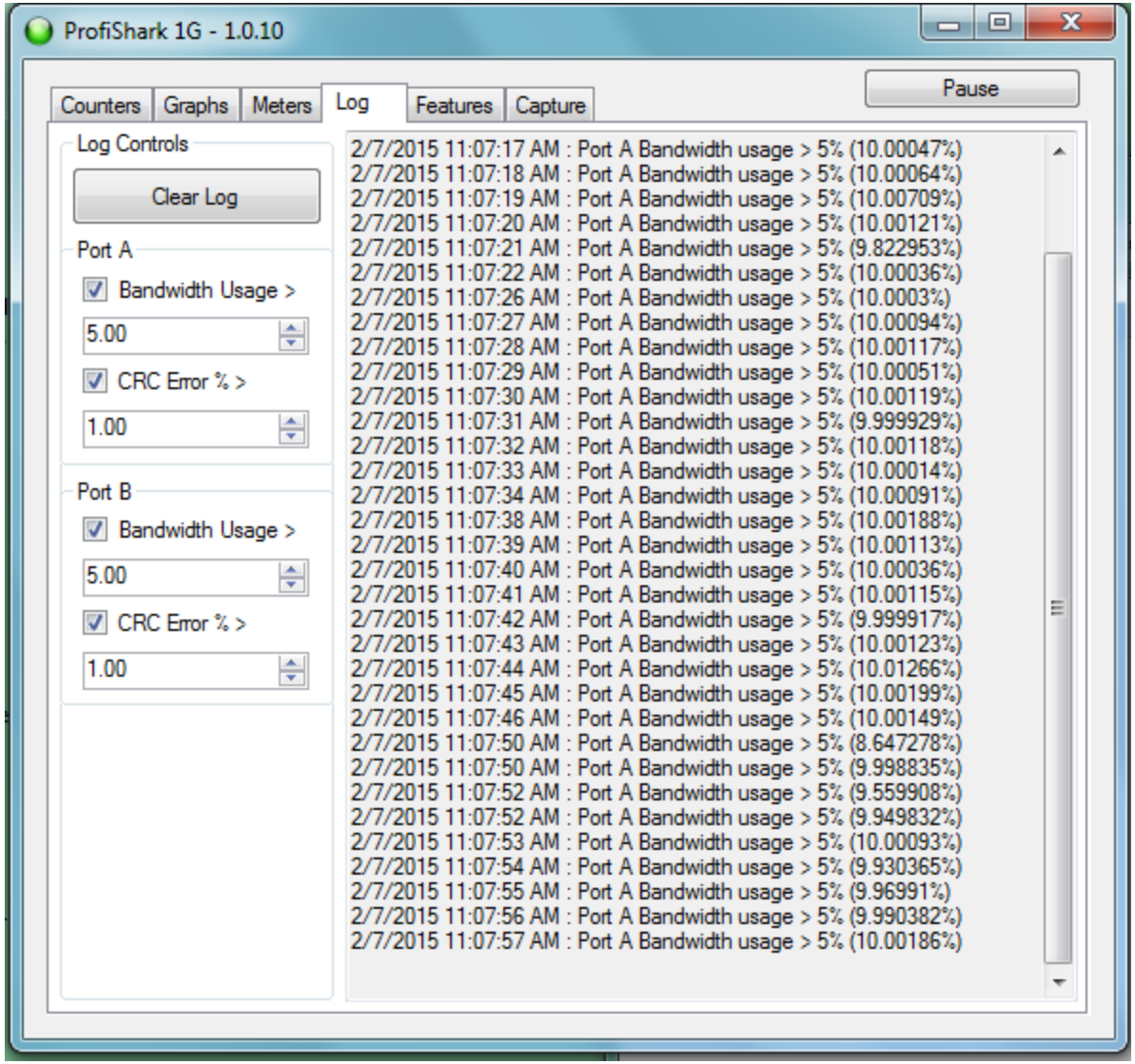


Figure 19: Log

Features

This tab offers a configuration information about the Tap. Notice that here we have access to the Capture Format choices: *Enable timestamps* (8ns resolution), *Transmit CRC Errors*, and retaining the Ethernet CRC trailer (*Keep CRC32*).

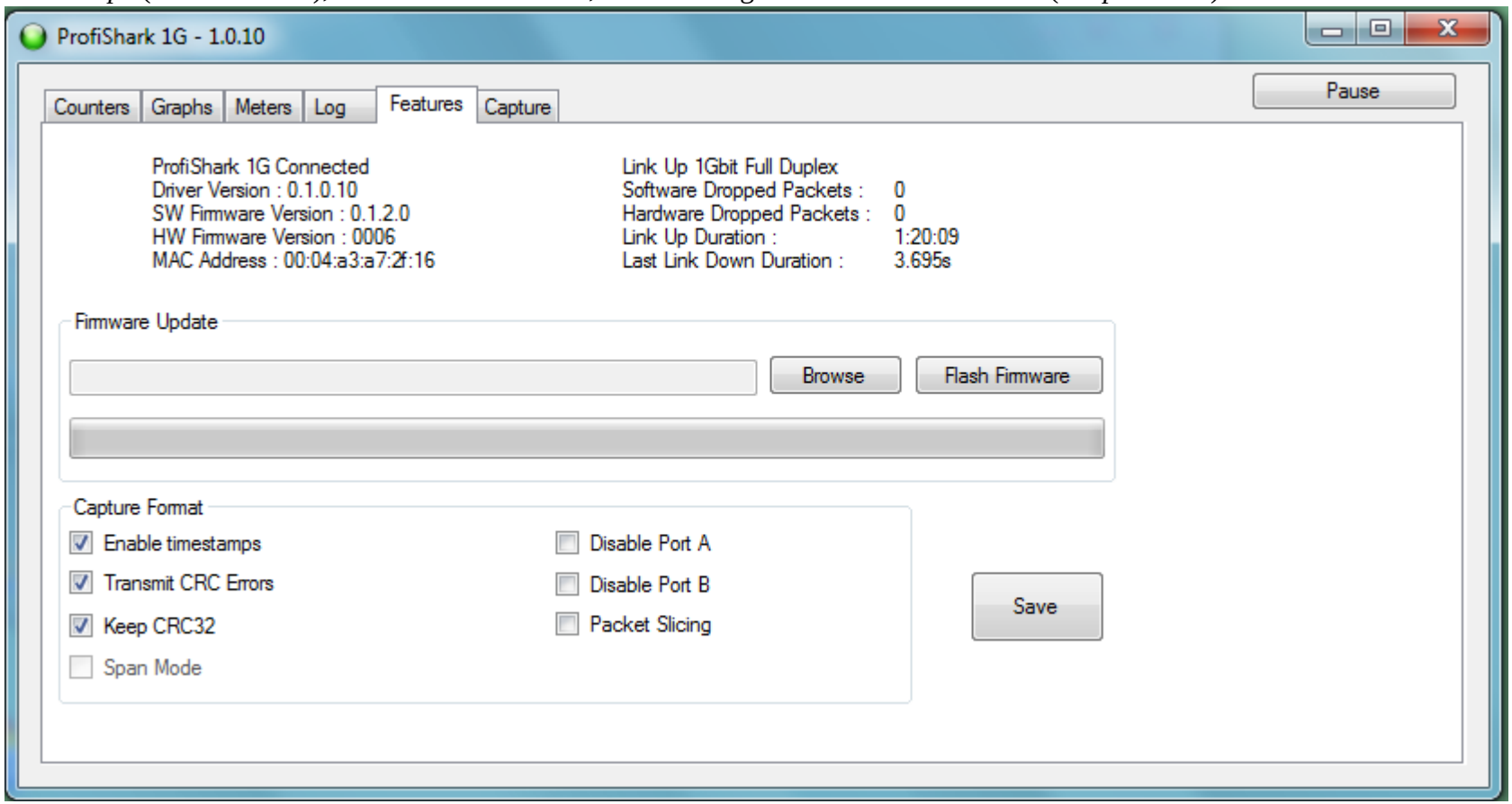


Figure 20: Features

Capture

The last tab allows us to use this application's built-in capture capabilities -- this application bypasses much of the Windows networking stack, capturing frames as they flow across the USB port, dropping them into RAM cache, and from their spooling them

off to disk. The ProfiShark application can achieve line-rate capture using this technique, and if storage IO is sufficiently fast, can save those frames to storage without drops, even at line-rate.

The GUI offers the usual options for a ring-buffer (Loop), size and number of saved files, plus tweaking the RAM cache.

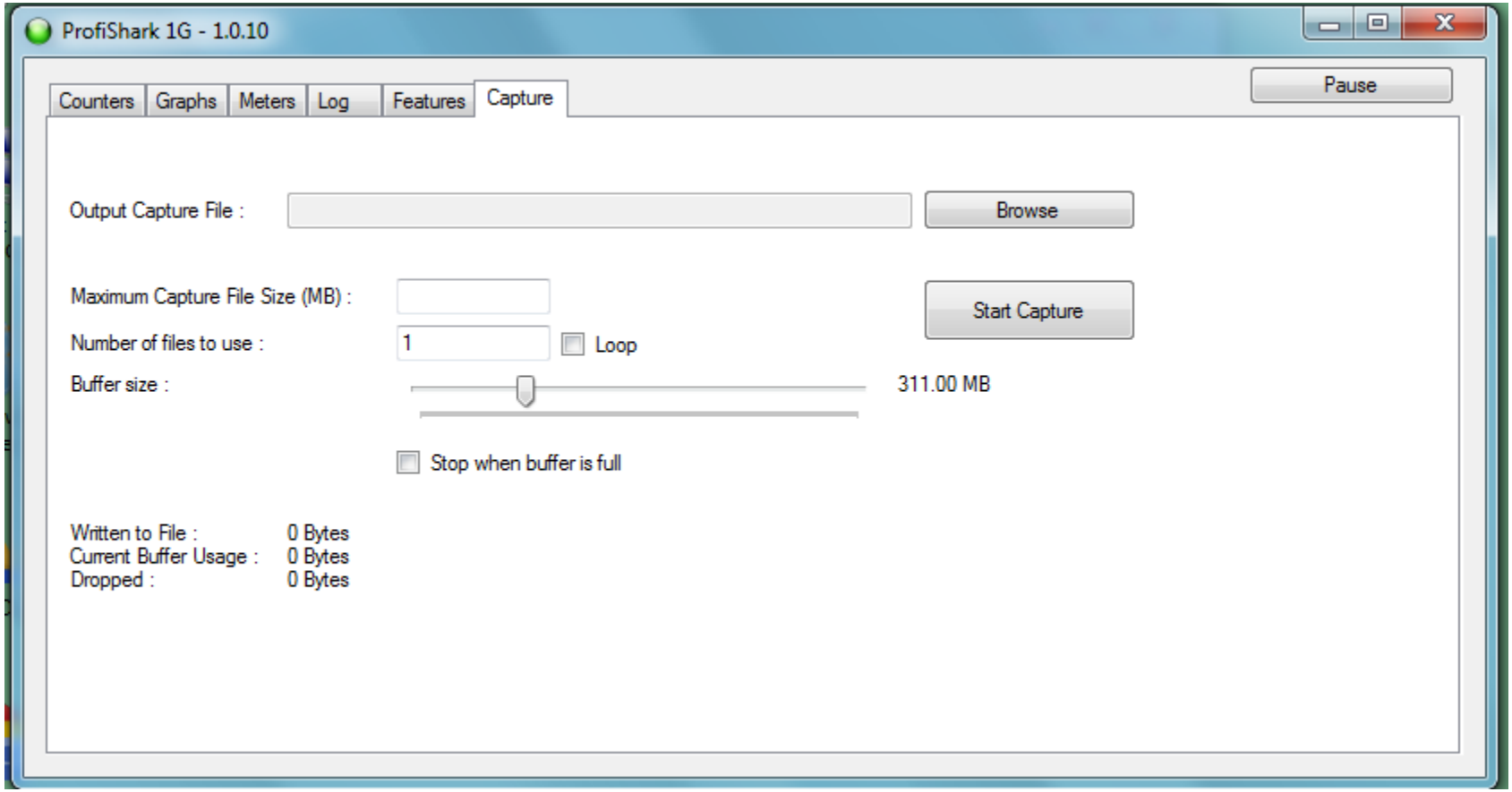


Figure 21: Capture

Live Capture: Capturing and Decoding Using Your Analyzer

We can use the provided ProfiShark application to capture frames, but we can also use any of a number of 3rd party software packages to perform the capture -- the ProfiTap folks call this *Live Capture* mode. To your favorite software analyzer, the Tap looks like any other NIC; I illustrate this using Wireshark:

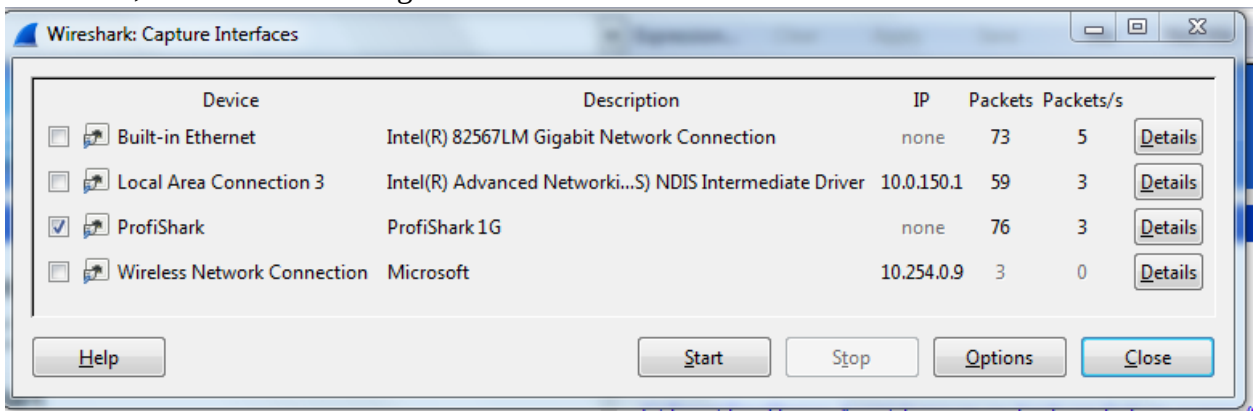


Figure 22: Capture Interfaces

But we must now pay attention to the Capture Format boxes which are checked on the Feature tab

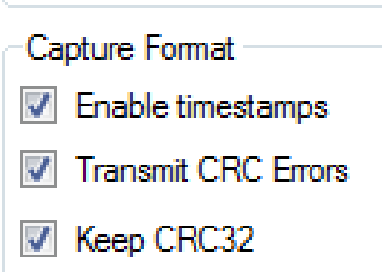


Figure 23: Capture Format

and how they interact with whether or not the ProfiShark-1G dissector is enabled.

Recall that adding the *profishark_1g.dll* dissector adds a ProfiShark-1G item to the list of protocols visible via Preferences... Protocol...

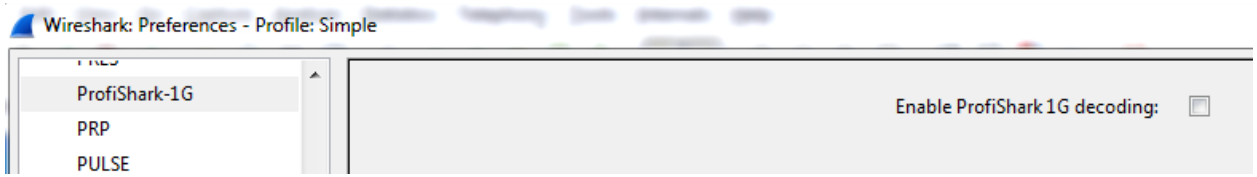


Figure 24: ProfiShark-1G Dissector

The interaction between Capture Format and ProfiShark-1G makes sense once we understand how the ProfiTap transmits hardware timestamp information. It does so by encoding the timestamp into eight (8) bytes and appending these eight bytes to the frame, as it transmits the frame across the USB port to the analyzer.

Enable Capture Formats and Enable ProfiShark-1G Dissector

If the ProfiShark-1G dissector is enabled, then Wireshark correctly interprets those eight bytes, using them to populate any Time columns and also displaying the timestamp as a 'protocol' -- notice the ProfiShark-1G 'frame' located in front of the Ethernet II frame.

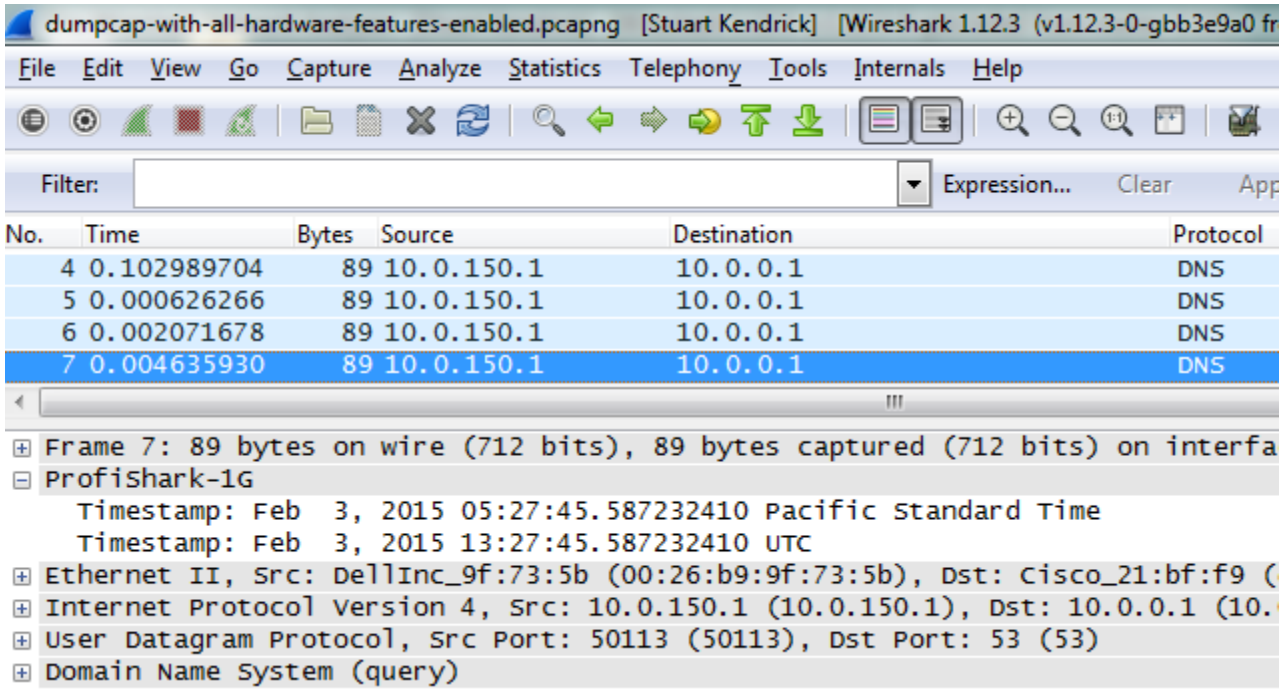


Figure 25: ProfiShark-1G Dissector in Action

Here we can see the eight bytes of timestamp appended to the frame:

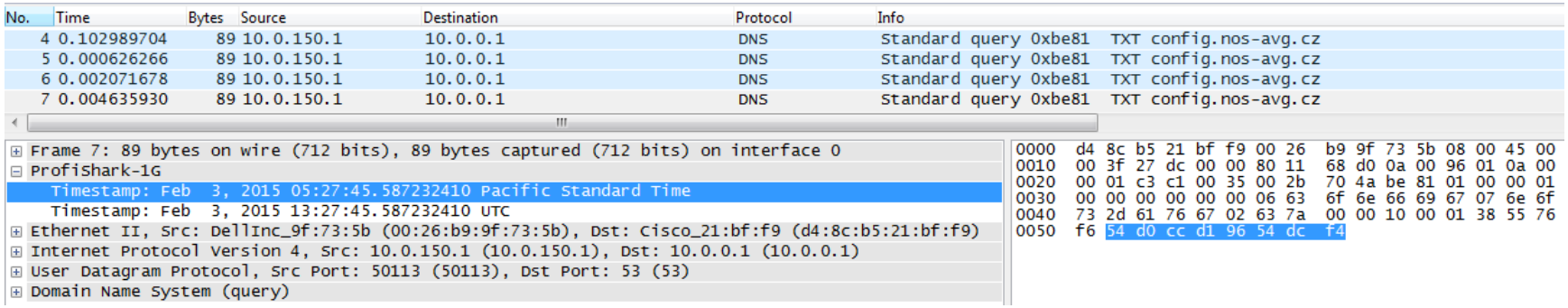


Figure 26: Eight Bytes of Timestamp

Furthermore, with those eight bytes accounted for, Wireshark's heuristics have a better chance of accurately dissecting the Ethernet CRC, which consists of the four (4) bytes which just prior to the newly-added eight byte timestamp.

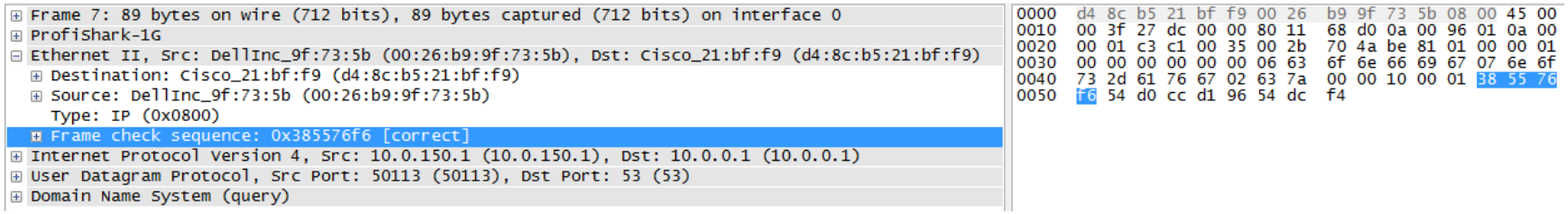


Figure 27: Correctly dissect Ethernet CRC

Enable Capture Formats and Disable ProfiShark-1G Dissector

However, if we disable the ProfiShark-1G dissector, then Wireshark interprets that last four bytes of the frame as an Ethernet CRC and then stumbles from there. Notice here that the ProfiShark-1G frame has vanished and that Wireshark incorrectly believes that the last four bytes consist of the Ethernet CRC.

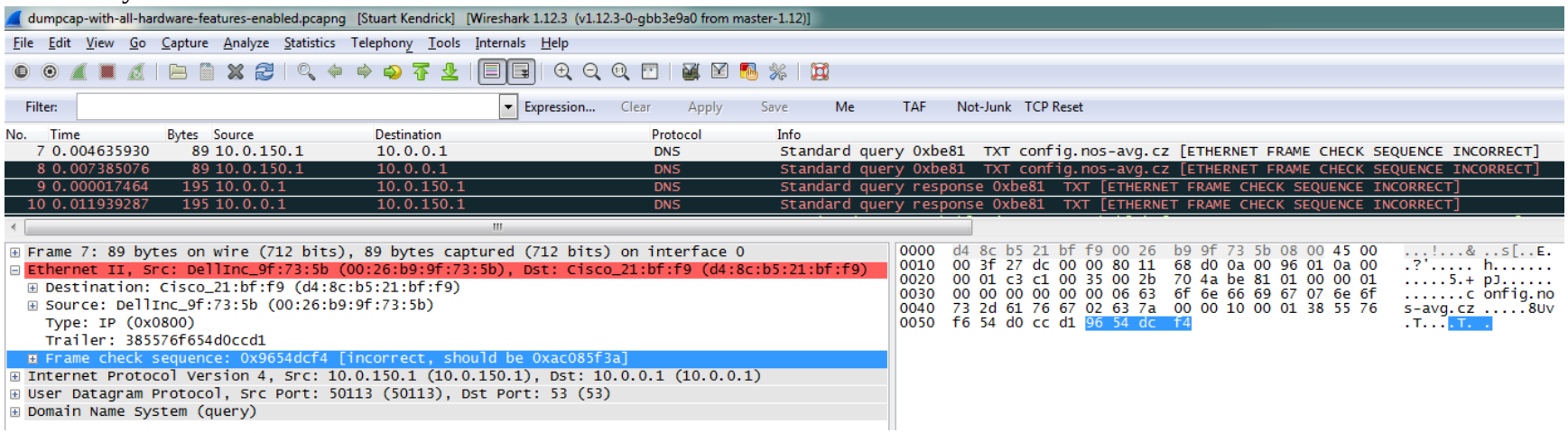


Figure 28: Incorrectly dissect Ethernet CRC

Interestingly enough, if we capture with Hardware Timestamps enabled, Keep CRC32 disabled, but forget to enable the ProfiShark-1G dissector, another dissector, the VSS-Monitoring dissector, will kick in and correctly decode the timestamp, though displaying the result in a different location than the one chosen by the ProfiShark-1G dissector.

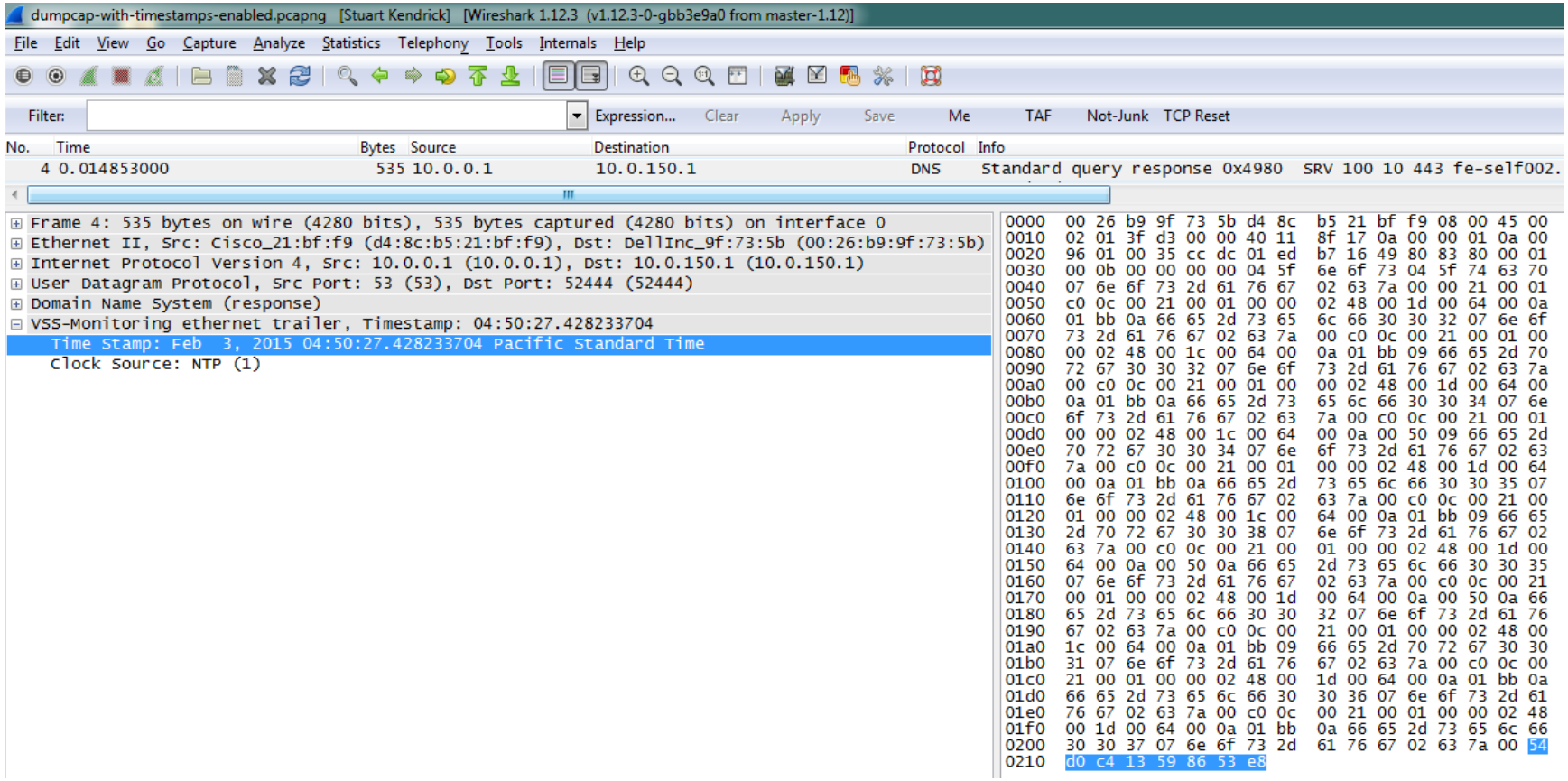


Figure 29: VSS-Monitoring Dissector

Disable Capture Formats and Enable ProfiShark-1G Dissector

Conversely, if we capture without Hardware Timestamps but leave the ProfiShark-1G dissector enabled, then we also confuse Wireshark: we are telling Wireshark "Interpret the last eight bytes as a nanosecond timestamp" ... when in fact, the last eight bytes of the frame do not contain a timestamp; rather they contain, in the example below, the last eight bytes of a TCP frame.

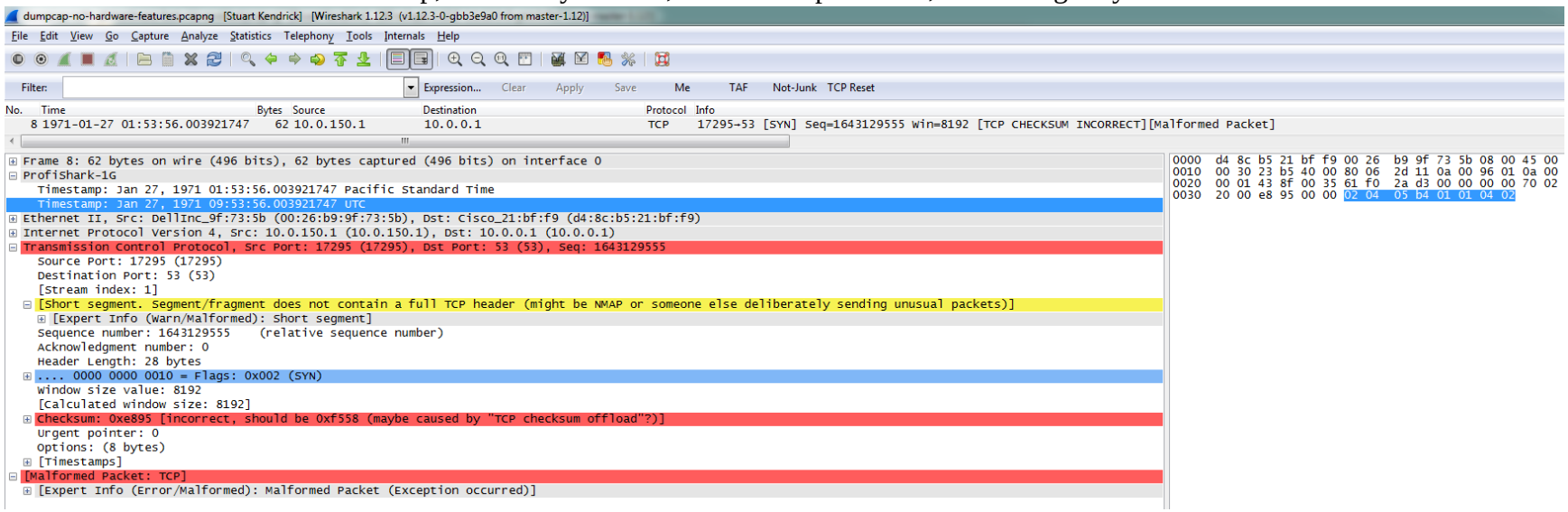


Figure 30: Don't Do This

Summarize Interactions

I summarize what happens based on various combinations in the following table. Basically, if we enable the Hardware Timestamping feature on the ProfiTap, then we want to analyze that trace with the ProfiShark dissector enabled. Otherwise, we want to make sure that the ProfiShark dissector is disabled. The ProfiShark-1G dissector conceptually has nothing to do with Ethernet CRC dissection, but it does get tangled up in the issue, because Wireshark's heuristics around correctly identifying an Ethernet CRC become confused if the ProfiShark dissector does not kick in.

ProfiShark Capture Parameters	ProfiShark-1G Dissector Status	Result
Timestamps & Keep CRC32 Enabled	Enabled	Accurate Wireshark dissection
Timestamps & Keep CRC32 Enabled	Disabled	Mangled Wireshark dissection
Timestamps Enabled	Enabled	Accurate Wireshark dissection
Timestamps Enabled	Disabled	Mostly accurate Wireshark dissection[1]
KeepCRC32 Enabled	Enabled	Accurate Wireshark dissection
KeepCRC32 Enabled	Disabled	Accurate Wireshark dissection
Timestamps & Keep CRC32 Disabled	Enabled	Mangled Wireshark dissection
Timestamps & Keep CRC32 Disabled	Disabled	Accurate Wireshark dissection

Figure 31: Capture Parameters vs Dissector Status

In other words, checking the *Enable timestamps* box in ProfiShark-1G tells ProfiShark-1G to append a timestamp to the frames it forwards across the USB cable to the virtual ProfiShark-1G NIC, where Wireshark is capturing. ProfiShark-1G does it this way because Wireshark, relying on Winpcap / libpcap, does not record the time when it receives the frame with any particular accuracy -- to take advantage of the Tap's 8ns timestamp resolution, the Tap has to communicate the timestamp using some other mechanism, and the mechanism the ProfiTap folks chose was to modify the forwarded frames themselves, by pasting the timestamp onto the end of the frame. To correctly interpret these appended bytes as a timestamp, Wireshark must invoke a dissector which looks for this timestamp.

My Two Bits

I kept forgetting that I had enabled the ProfiShark-1G dissector and then trying to analyze pcaps captured using some other mechanism, which leads to incorrect dissection. At the moment, I've disabled the ProfiShark-1G dissector and always use Direct Capture mode, see next section.

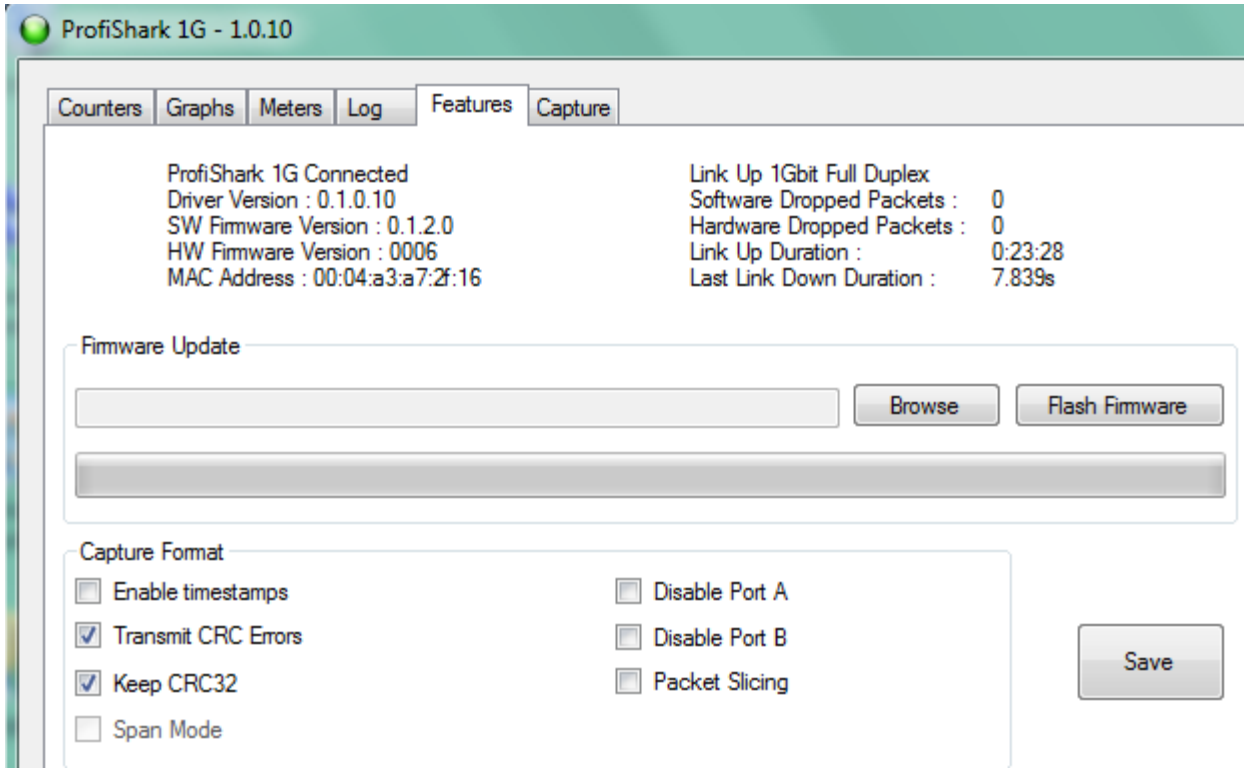
Direct Capture

The subtleties of the ProfiShark-1G dissector evaporate when we use *Direct Capture* mode, i.e. when we use the Capture tab inside ProfiShark Manager. When we do this, ProfiShark-1G owns the creation of the timestamp, which it inserts into the appropriate

[1] Wireshark incorrectly applies the VSS-Monitoring dissector. This dissector correctly decodes the timestamp as a nanosecond timestamp but inaccurately claims that the time source for the stamp comes from NTP.

location in whichever format you have chosen (.erf, .pcap, or .pcapng being the three file formats which ProfiShark-1G supports). No appended timestamp and thus no need for a dedicated dissector.

Here, I've unchecked the *Enable timestamps* box -- now the frames forwarded across the USB port no longer contain the appended hardware-generated timestamp, though the files created by Direct Capture still retain 8ns resolution timestamps.



Link-Local Frames

So what does a Tap do for us, as protocol analysts? Well, it allows us to view *link-local* frames, frames which, for example, the SPAN / Port-Mirroring function of an Ethernet switch would not forward to us.

LACP

Here is an example of LACP Hellos being exchanged between a host and a switch:

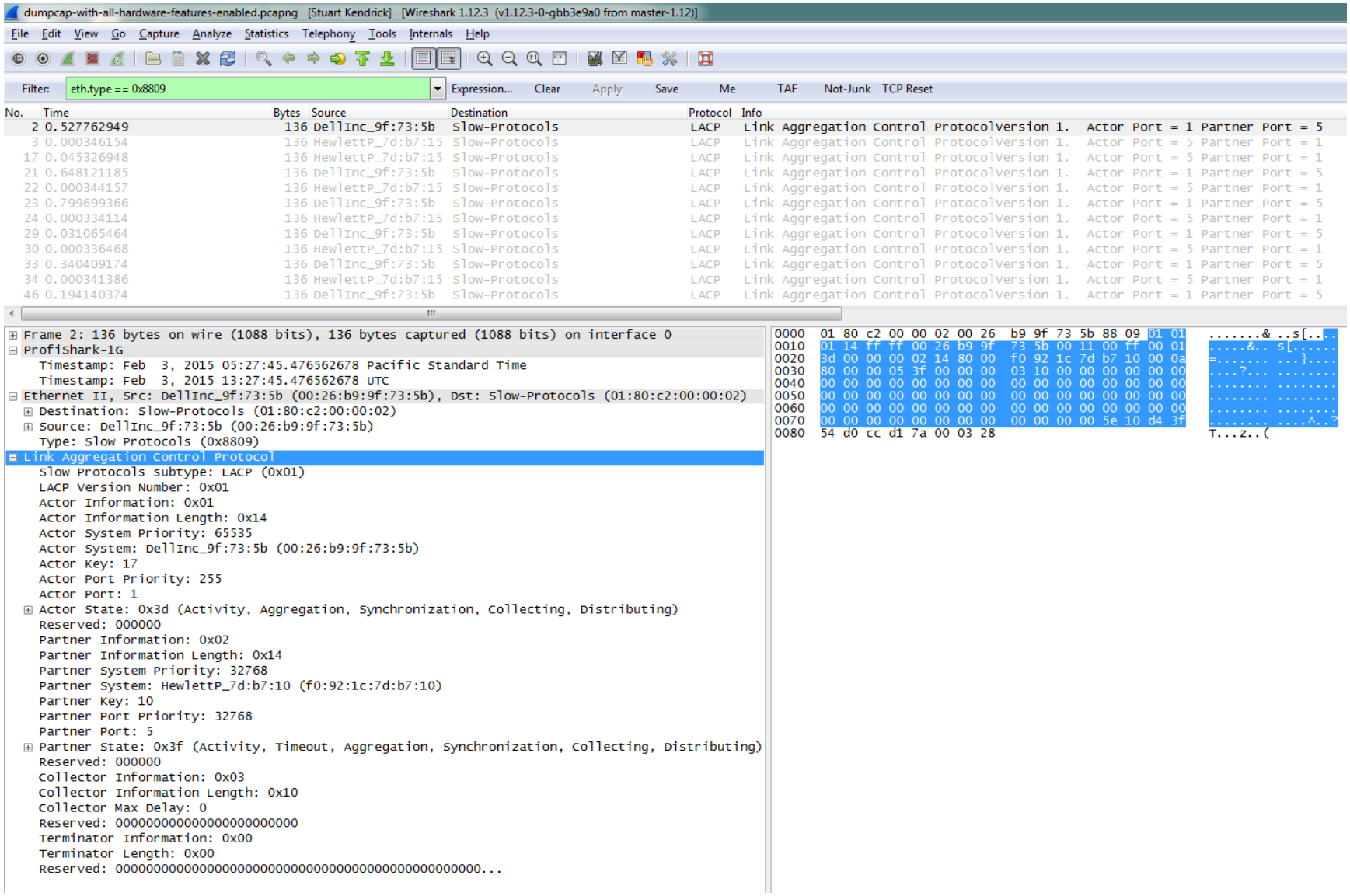


Figure 32: LACP Hellos

Focusing on the LACP frame inside Frame #2, transmitted by the host:

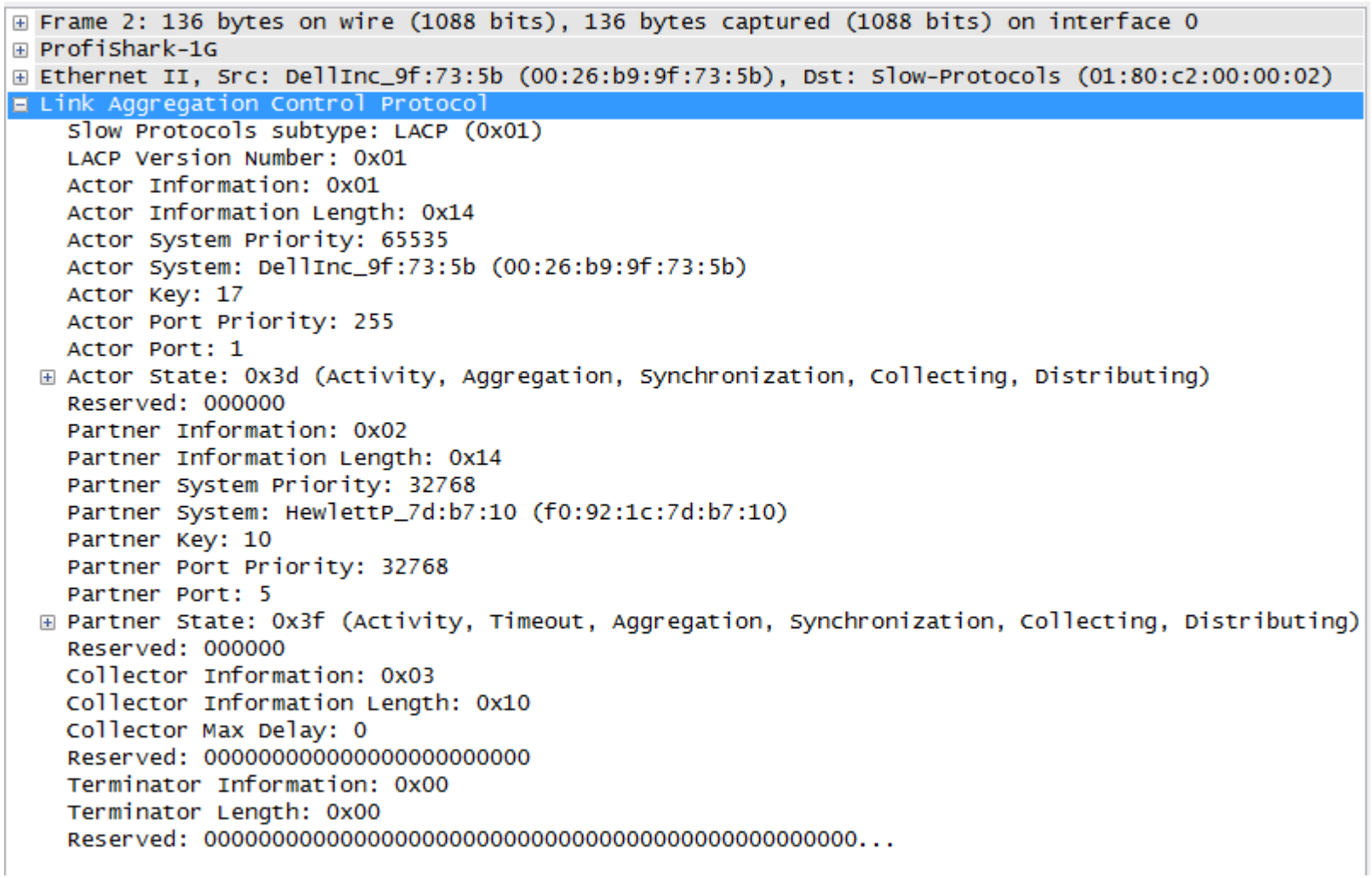


Figure 33: Drill Down on Host LACP Frame

And here's the LACP Hello from the switch:

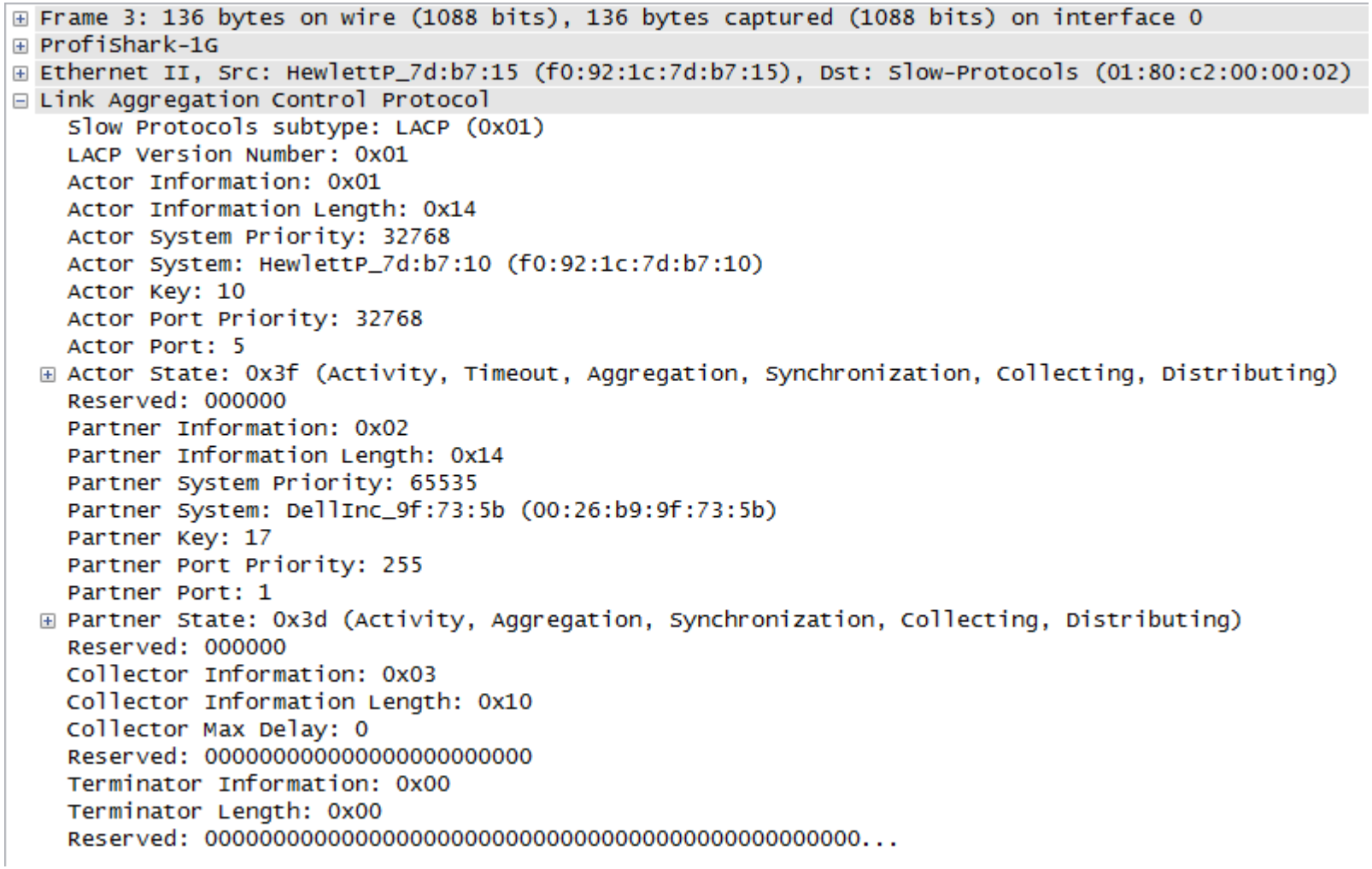


Figure 34: Drill Down on Switch LACP Frame

If we were to Port-Mirror port 1 or port 5 on this Ethernet switch, we would not see these LACP frames.

Bad Ethernet Frame Check Sequence

Similarly, frames in which bits have been flipped, in this case due to a bad cable, are normally discarded by the first Ethernet chipset that sees them (perhaps on your switch, perhaps on your analyzer's NIC)[2], because the originally calculated Ethernet FCS does not match the FCS which the receiving NIC calculates, and modern NICs, by default, discard such frames. The ProfiTap, being specialized hardware, can be instructed to capture them, as seen here. A typical analyzer would not capture Frame 1218 and you the analyst would then be puzzled as to why the new Frame 1218 (1219 in the display below) arrived ~.1487s after the previous one: you would not, a priori, know why the conversation contained such a large delay, but would have to infer what happened from further analysis.

[2] The average NIC driver in a typical PC does not let you override this behavior; even checking 'Promiscuous mode' doesn't help -- your NIC will drop frames which fail the CRC check, long before Winpcap / libpcap see them.

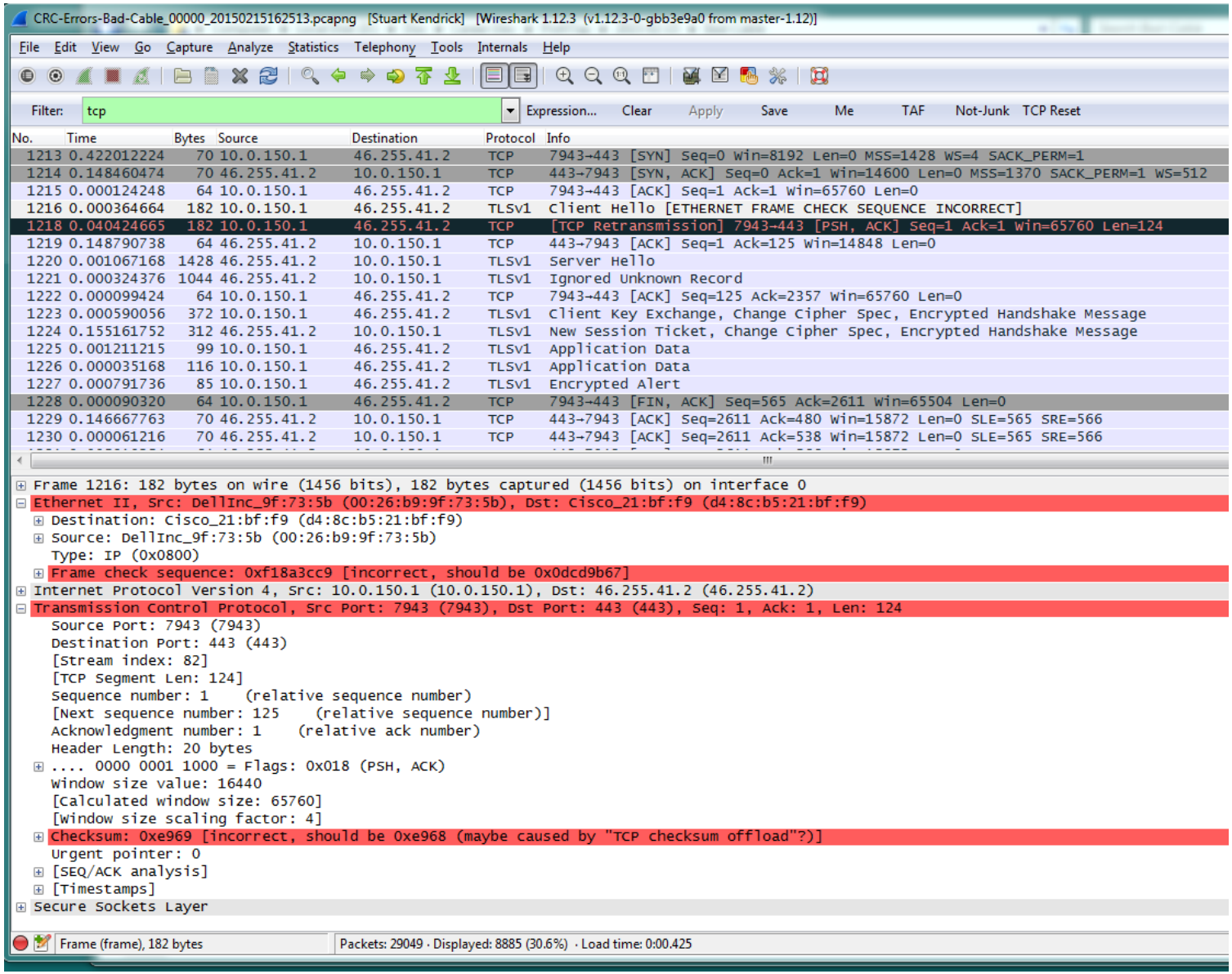


Figure 35: Bad Frame Check Sequence

Summary

As of this writing, the ProfiTap-1G offers the cheapest way I know of to capture in-line -- in one package, it provides an in-line capture engine, leveraging the USB port on your PC, rather than requiring a specialized capture engine to be installed in your PC. As an added bonus, you can choose to continue using your existing analyzer software to capture. Or, you can use the included capture application, which additionally offers various ways to summarize statistics and log events on the capture stream.