Understanding Network Performance

Joe St Sauver, Ph.D.
<jstsauver@gmail.com>
M3AAWG Expert Advisor

M3AAWG 64
June 2nd, 2025
Vancouver, BC Canada

Part 0. Introduction

****ATTENTION****

- ***DO NOT* run Speedtest, iperf3 or any other active bandwidth tester while you're here at M3AAWG.** Bandwidth testing can interfere with other users' ability to use the network.

- ***RUNNING WIRESHARK (or similar network packet capture technology) is also PROHIBITED while at M3AAWG.*** Wait to run Wireshark or similar packet capture technologies until you're on your own network (or another network where you're authorized to do so).

- ***Always ensure you have a CLEAN RECENT BACKUP of your system before you make ANY changes to it.***

- System tuning is an experimental process. Success can depend on many factors, including some factors that may be beyond your direct control (such as the performance of remote systems or networks). ***We do NOT claim that all attempts at improving performance will be successful. In fact, your system may even end up performing WORSE than it did originally if you adjust the default settings.*** Proceed accordingly!

Joe! Is network performance really a 'M3AAWG thing'? *YES*

- M3AAWG's all about "messaging, malware and mobile," e.g., network applications and cyber security topics. That directly relates to network **performance,** particularly now that our online environment is dramatically evolving.

  How are things evolving? Well, **residential connections have gotten REALLY fast & cheap.**

  - **Multi-gigabit+ Fiber To The Home (FTTH) connections are now routinely available in North America.** For example, in the suburban area of Oregon where I live, some sample FTTH offers include (at least as of the time this deck was prepared):

    Xfinity: 2.1 Gbps for USD$85/month
    Hunter Communications: 2.5 Gbps for USD$120/month

    [Consult your local providers for specific offers available in your area]

Imagine if fast connections like these were to fall under the control of bad guys!

Aside: "Are You Going To Talk About 5G Home Wireless, Too?"

- "Mobile" ***IS*** 1/3rd of M3AAWG's portfolio. That said, vendors have indicated that while 5G wireless networks may eventually get up to 20 Gbps, multi-Gbps speeds are not routinely available on 5G wireless connections today.

- For example, in 2023, OpenSignal reported that [users of the fastest 5G US carrier they identified] **"enjoy average 5G download speeds clocking in at 186.3Mbps — more than 100Mbps faster than the speeds" [that users of the 2nd fastest 5G US carrier enjoy].** [https://www.opensignal.com/reports/2023/01/usa/mobile-network-experience-1]

- While 100-200 Mbps speeds are a huge upgrade from 4G wireless speeds, they're still an order of magnitude behind speeds routinely available from fiber optic links. For that reason, we will NOT be focusing on enhancing the performance of 5G wireless network connections today

- **Let's move on to corporate network connectivity.**

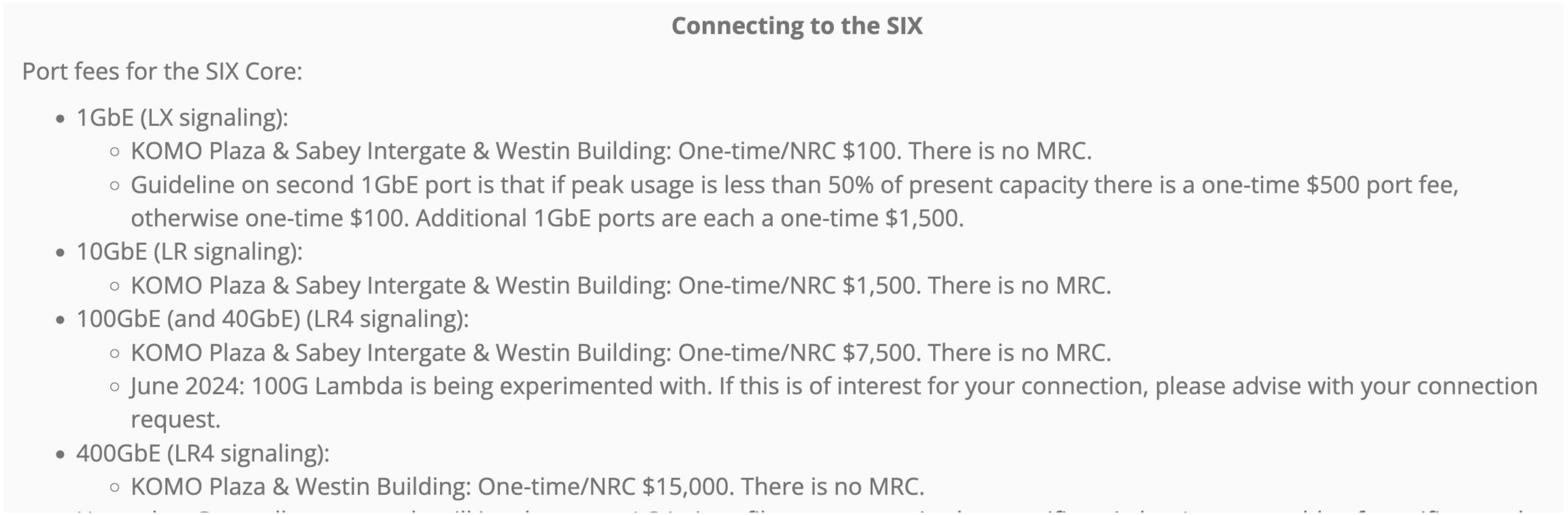
Corporate Connectivity: Internet Transit and Peering

- When talking about corporate Internet connectivity, it is important to be clear on the difference between **peering** and **Internet transit.**
- **Peering** is when two (normally similarly-sized) providers agree to exchange customer traffic (and only customer traffic). This may happen at exchange points or via private interconnections. Peering is how traffic flows from one backbone provider to another. Peering may be **selective and bilateral** (just between two particular parties), or **open and multilateral.** Peering may happen **"settlement free"** but you'll also see **"paid peering"** (depending on the relative role of the peers, their traffic balance, and other factors).
- On the other hand, when you buy **Internet transit,** you pay a network provider for **full Internet connectivity.** You can send traffic to anyone on the Internet, and anyone on the Internet can reach you. This simple ("don't worry about it, we'll handle all the details") sort of access comes at a cost (you'll pay to get transit service from a network provider), but you won't have to arrange for and maintain a bunch of peerings.
- Corporate networks of at least a certain size will often have a mix of some peering (typically to help keep local traffic local, and to skim off some the highest volume traffic sources and sinks), plus Internet transit to handle everything else.

Internet Transit Has Been Getting Substantially Faster and CHEAPER

- At one point, 10 Gbps or OC-192 connections were considered to be very fast, but these days **transit providers** are routinely offering 100 Gbps (and sometimes even 400 Gbps).
- A sample backbone provider, Arelion (formerly Telia), states, "For IP Transit, **10Gb and 100Gb ports are the current standard.** Arelion can offer 400Gb ports in many locations, as many as 50 at the moment." (https://www.arelion.com/products-and-services/internet-and-cloud) Way back in 2020, they went on to claim that "In **2019** alone, Telia Carrier deployed **10,000 global 100-gigabit Ethernet ports,"** see https://blog.arelion.com/2020/08/03/ip-transit-everything-old-is-new-again/
- TeleGeography states that **"In Q2 2024, the lowest 10 GigE prices on offer in the most competitive markets were at the brink of $0.07 per Mbps per month. The lowest for 100 GigE was $0.05 per Mbps per month."** and "Across key cities in the U.S. and Europe, **400 GigE prices range from $0.07 to $0.08 per Mbps.**" See "IP Transit Price Erosion: Significant Regional Differences Remain," https://blog.telegeography.com/ip-transit-price-erosion-significant-regional-differences-remain
- For comparison, our **2.5Gbps FTTH,** $120/2500 Mbps ==> $0.048/Mbps/month.

Peering at The Seattle Internet Exchange, A Sample PNW Peering Point



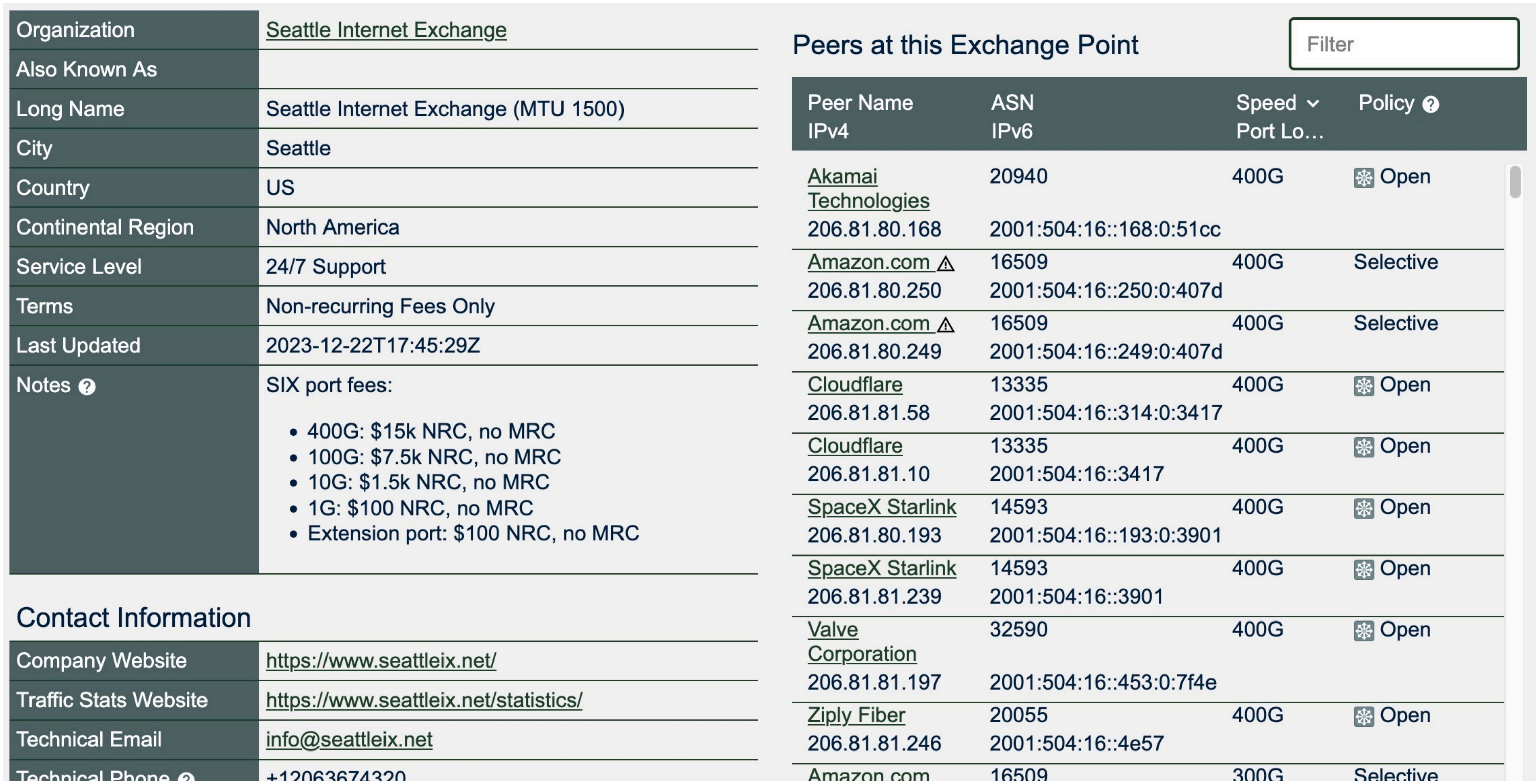
Connecting to the SIX

Port fees for the SIX Core:

- 1GbE (LX signaling):
  - KOMO Plaza & Sabey Intergate & Westin Building: One-time/NRC $100. There is no MRC.
  - Guideline on second 1GbE port is that if peak usage is less than 50% of present capacity there is a one-time $500 port fee, otherwise one-time $100. Additional 1GbE ports are each a one-time $1,500.
- 10GbE (LR signaling):
  - KOMO Plaza & Sabey Intergate & Westin Building: One-time/NRC $1,500. There is no MRC.
- 100GbE (and 40GbE) (LR4 signaling):
  - KOMO Plaza & Sabey Intergate & Westin Building: One-time/NRC $7,500. There is no MRC.
  - June 2024: 100G Lambda is being experimented with. If this is of interest for your connection, please advise with your connection request.
- 400GbE (LR4 signaling):
  - KOMO Plaza & Westin Building: One-time/NRC $15,000. There is no MRC.

Reference: https://www.seattleix.net/join

PeeringDB Details For the SIX (https://www.peeringdb.com/ix/13)



Organization	Seattle Internet Exchange
Also Known As	
Long Name	Seattle Internet Exchange (MTU 1500)
City	Seattle
Country	US
Continental Region	North America
Service Level	24/7 Support
Terms	Non-recurring Fees Only
Last Updated	2023-12-22T17:45:29Z
Notes ?	SIX port fees: • 400G: $15k NRC, no MRC • 100G: $7.5k NRC, no MRC • 10G: $1.5k NRC, no MRC • 1G: $100 NRC, no MRC • Extension port: $100 NRC, no MRC

Contact Information

Company Website	https://www.seattleix.net/
Traffic Stats Website	https://www.seattleix.net/statistics/
Technical Email	info@seattleix.net
Technical Phone ?	+12063674320

Peers at this Exchange Point

Filter

Peer Name IPv4	ASN IPv6	Speed Port Lo…	Policy ?
Akamai Technologies 206.81.80.168	20940 2001:504:16::168:0:51cc	400G	Open
Amazon.com ⚠ 206.81.80.250	16509 2001:504:16::250:0:407d	400G	Selective
Amazon.com ⚠ 206.81.80.249	16509 2001:504:16::249:0:407d	400G	Selective
Cloudflare 206.81.81.58	13335 2001:504:16::314:0:3417	400G	Open
Cloudflare 206.81.81.10	13335 2001:504:16::3417	400G	Open
SpaceX Starlink 206.81.80.193	14593 2001:504:16::193:0:3901	400G	Open
SpaceX Starlink 206.81.81.239	14593 2001:504:16::3901	400G	Open
Valve Corporation 206.81.81.197	32590 2001:504:16::453:0:7f4e	400G	Open
Ziply Fiber 206.81.81.246	20055 2001:504:16::4e57	400G	Open
Amazon.com	16509	300G	Selective

Some Canadian Exchange Points Offering 100Gbps+ Peering Connections

- **TorIX,** Toronto https://www.peeringdb.com/ix/24
- **ONIX,** Toronto https://www.peeringdb.com/ix/4059
- **QIX,** Montreal https://www.peeringdb.com/ix/355
- **YYCIX,** Calgary https://www.peeringdb.com/ix/639
- **VANIX,** Vancouver https://www.peeringdb.com/ix/863

Security Challenges Can Arise From Fast (100Gbps+) Connections

- **At 100 to 400 Gbps, it can be "tricky" (that's pronounced "hard/expensive") to manage, monitor, and secure those connections.** Multiple companies ARE trying to fill that niche... Some high-performance network monitoring companies include (in alphabetical order):

  Cpacket (https://www.cpacket.com/)
  Cubro (https://www.cubro.com/en/),
  Endace (https://www.endace.com/),
  FMADIO (https://www.fmad.io/),
  MantisNet (https://www.mantisnet.com/),
  NEOX (https://neoxnetworks.com/),
  NTOP (https://www.ntop.org/)
  etc.

  But just how fast can these security solutions go? And what do they cost? Ensuring that providers maintain secure networks and adequate internal network visibility feels like something else that should be within M3AAWG's remit.

Are <u>You</u> Fighting Abuse? That Will Drive <u>Your</u> Need For Higher Bandwidth

- If you're going to <u>*effectively*</u> fight online abuse on fast networks, YOU will need to be able to collect empirical evidence at scale.

- Monitoring high performance connections drives a need for YOU as an anti-abuse analyst to HAVE and be able to USE high performance network connections **yourself.**

- Waiting for bulk data to transfer from remote sites represents a delay in the analysis process (and a window during which the bad guys can continue their shenanigans).

  It is critical that your processing of abuse-related data be as **timely as possible.**

The Structure For The Rest of This Training

- **First part of this training:** There are **practical aspects to getting a connection upgraded to multi-gigabit speeds**. This information will be directly helpful for you if you're thinking about upgrading your own connectivity. The network we create in this process will also represent the "lab" environment where we'll work on topics in the 2nd part of this talk.

- In the **second part of this training,** we'll consider some **specific technical topics** you'll bump into as you begin to work with "long fat networks" (aka high latency, high bandwidth networks), including:

  - Latency and its impact on bandwidth delay products
  - Packet loss, and the impact of your choice of TCP congestion control algorithms
  - Bufferbloat

- We've also provided a **brief glossary at the end of this document** given the large and unique vocabulary this talk employs.

Part I. Getting A 2.5 Gbps Network Established

1. Measuring Performance To Get A Starting Baseline

Bottom Line Up Front (BLUF):

You can't tell if things have gotten better if you
don't know where you were in the first place.
Therefore, start by measuring where you're currently at.
Speedtest is a GUI web tool you can use for that process.

Before You Do ANY Testing or Tweaking

1. Before you test, **disable any "power saving" settings** that may be enabled – you want to be able to "put your foot down on the accelerator" and get full power from your gear.

2. Most testing should be done using **wired Ethernet,** but if you do want to use wireless, get as close as possible to your access point. Distance (or walls) can cut your wireless throughput by half or more.

3. Decide if you're testing **IPv4 or IPv6 or both.** Speeds <u>should</u> be roughly comparable.

4. Be sure to also test **download <u>AND</u> upload** speeds – they may be different.

5. Plan to do **multiple runs.** Individual runs may be unusually good (or bad) just due to other network traffic or random factors, so do multiple runs and look at the average.

6. Minimize **network traffic from other applications or users** to the extent possible.

Document Your Starting Network Architecture

1. Our starting home network consisted of gigabit (symmetric) fiber connectivity from a local provider, terminating at a Nokia XS-110G-A Optical Network Terminal (ONT).

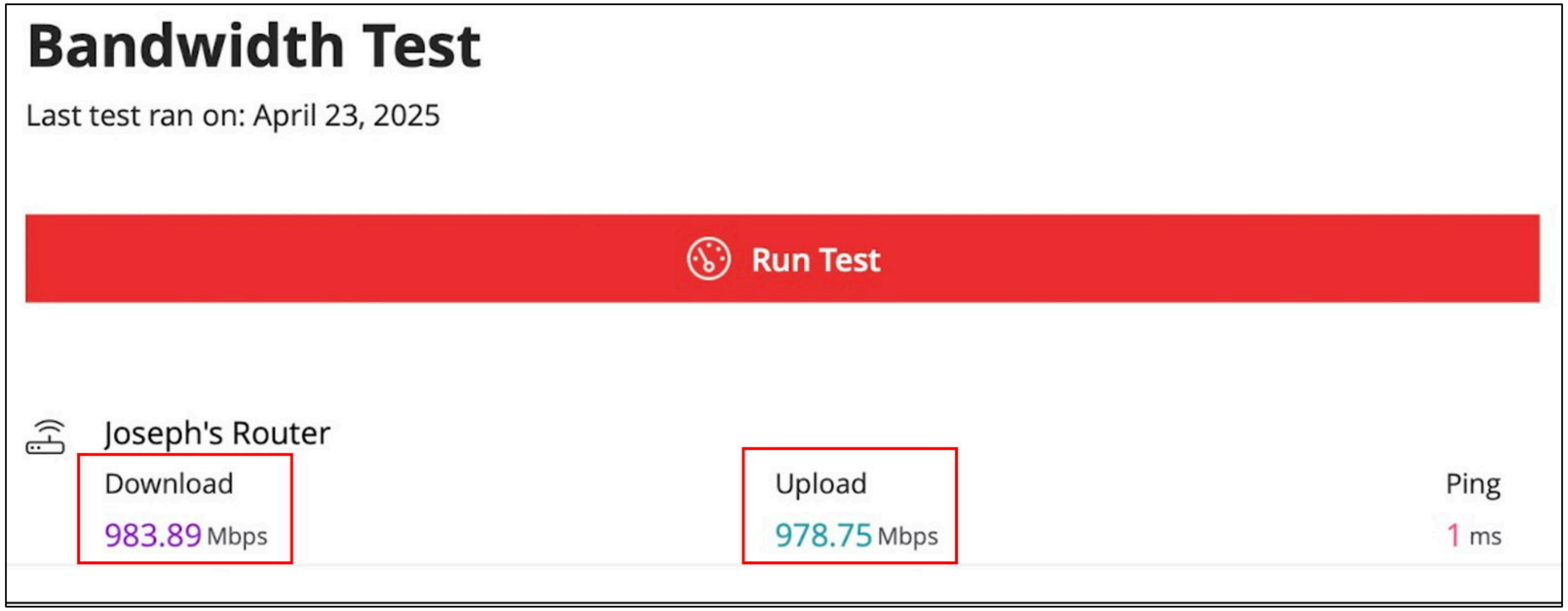
2. The ONT connected to a Calix Gigaspire BLAST u4 Tower. That provided WiFi and gigabit wired Ethernet connections.

3. Our test hosts, chosen to be representative of typical endpoints, connect via wired or wireless Ethernet to the Calix. They are:

   a) A Mac Apple Silicon (M1) laptop, initially connecting wirelessly.
   b) A Windows 11 (AMD Ryzen AI 9) laptop, initially connecting wirelessly.
   c) An Alienware R8 PC server (Intel Core i5) running Debian Linux, connected over gigabit wired Ethernet.

Your network architecture be different. Document it prior to upgrading.

Pre-Upgrade Bandwidth Tested From Our Calix BLAST u4 Tower


Bandwidth Test

Last test ran on: April 23, 2025

Joseph's Router

Download	Upload	Ping
983.89 Mbps	978.75 Mbps	1 ms

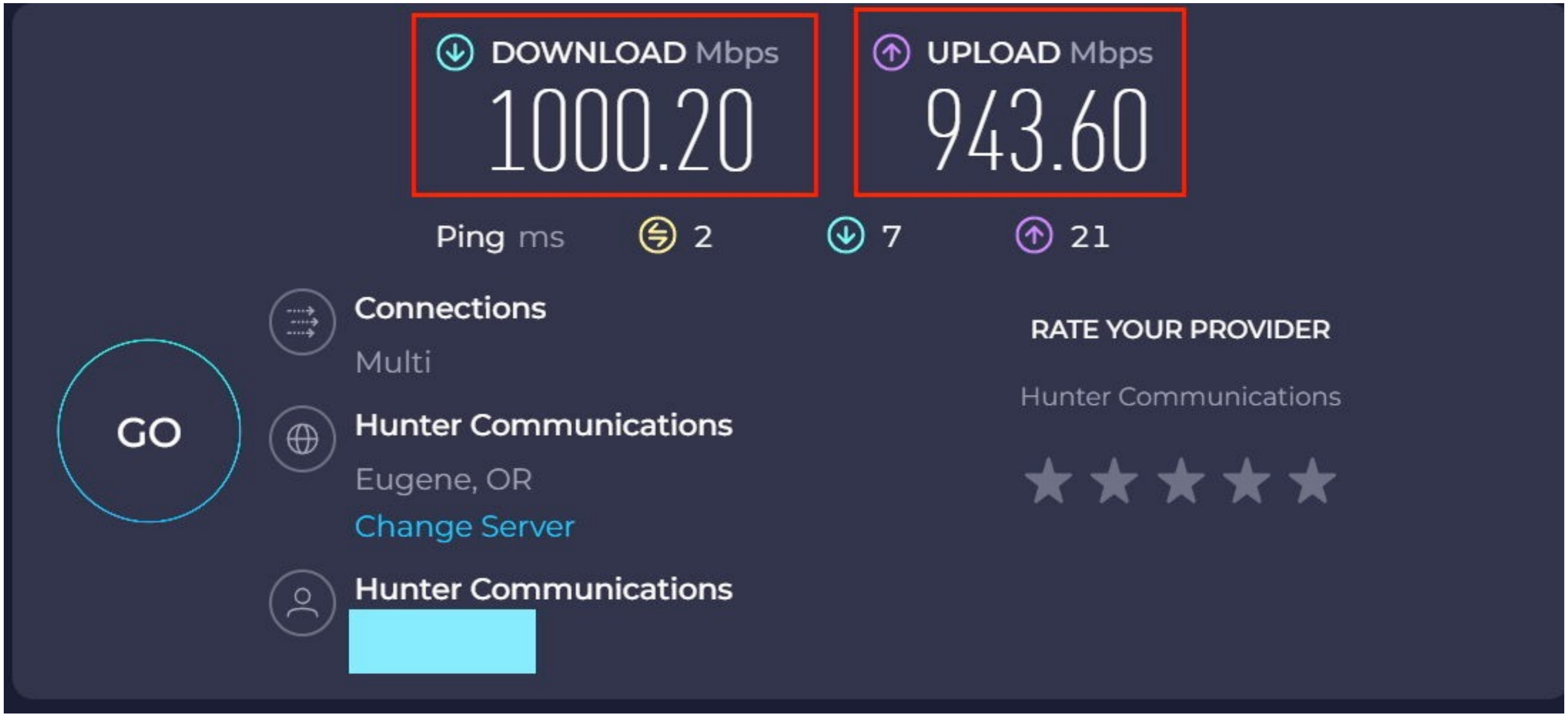
Running this test directly from the Calix (via the ISP-provided app) eliminated any potential issues or impact that might be associated with my test systems or my home network.

Note that this is reporting multi-stream TCP bandwidth run against a VERY close test endpoint (note the ping time of 1 msec).

That Was What We Hoped For/Expected...

- 978.75/1000*100=97.875% of our nominal 1000 Mbps expected rate.
- No, we didn't see *exactly* 1000 Mbps (likely due to Ethernet overhead), and granted, this testing was done by running multiple parallel TCP streams for a fairly short period to a very close bandwidth testing endpoint, but seeing 978 Mbps (basically full line rate) was encouraging.
- What do we see if we test from actual end-user devices?
- Let's test a Windows 11 PC using Speedtest.

WIRELESS Win11 Laptop tested using https://www.speedtest.net/



DOWNLOAD Mbps
1000.20

UPLOAD Mbps
943.60

Ping ms 2 7 21

GO

Connections
Multi

Hunter Communications
Eugene, OR
Change Server

Hunter Communications
[redacted]

RATE YOUR PROVIDER
Hunter Communications

Things To Note About The Preceding Windows Results

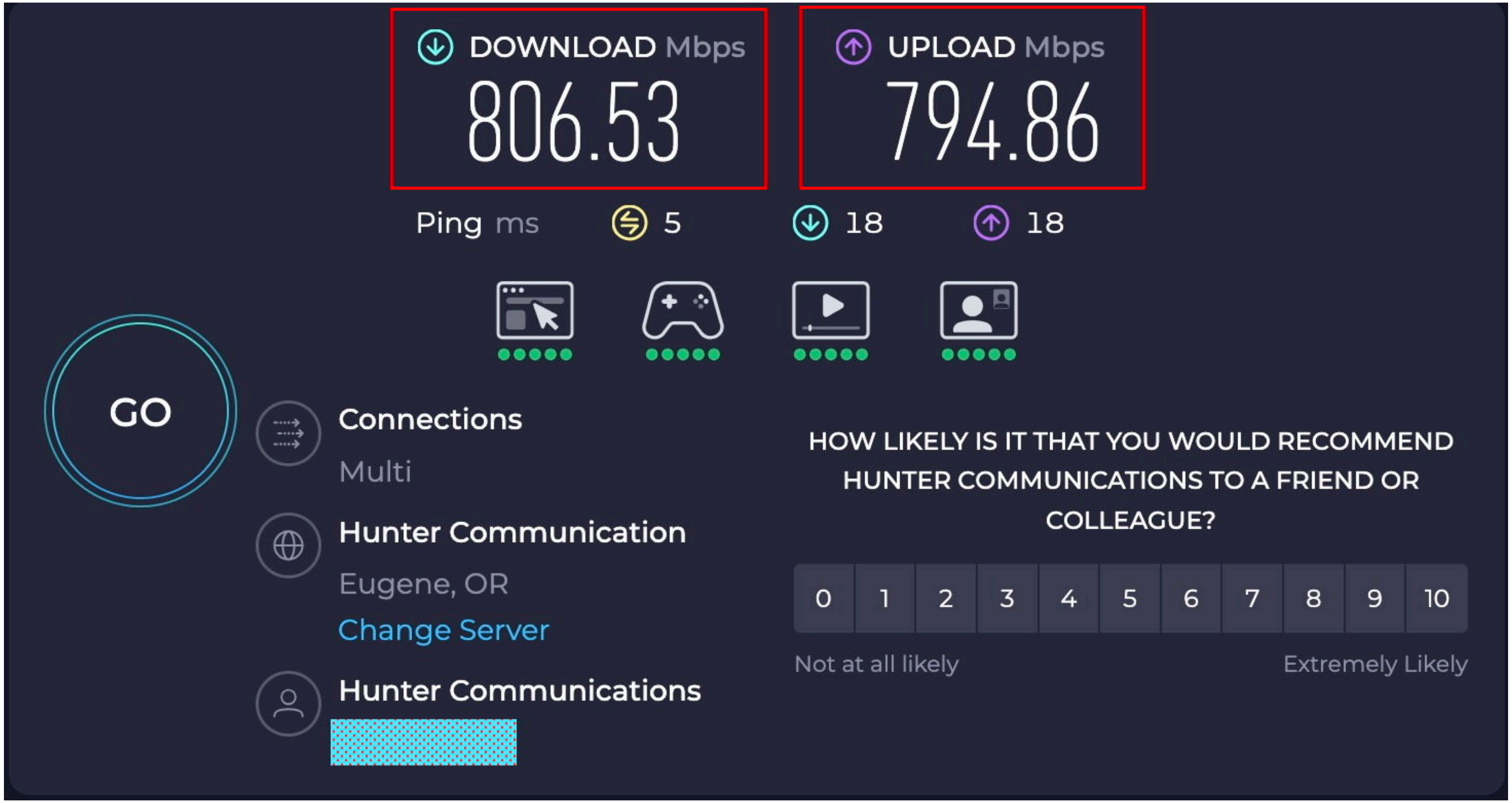
- Baseline performance was excellent from the Windows 11 laptop.

  That system (borrowed from a family member for testing since we don't personally have any Microsoft Windows systems) was a current-generation HP Omnibook Ultra (AMD Ryzen AI 9 HX 375 2.0 GHz with Radeon 890M, 32 GB of RAM, and **802.11ax WiFi 6 running at 160MHz**).

  This is a typical current generation ~$1200-class Windows laptop.

- What did we see for an older Mac PowerBook M1, also connected wirelessly?

WIRELESS Mac M1 Laptop using https://www.speedtest.net/



DOWNLOAD Mbps
806.53

UPLOAD Mbps
794.86

Ping ms 5 18 18

GO

Connections
Multi

Hunter Communication
Eugene, OR
Change Server

Hunter Communications
[redacted]

HOW LIKELY IS IT THAT YOU WOULD RECOMMEND HUNTER COMMUNICATIONS TO A FRIEND OR COLLEAGUE?

0	1	2	3	4	5	6	7	8	9	10

Not at all likely Extremely Likely

Some Notes About The Preceding Mac Results

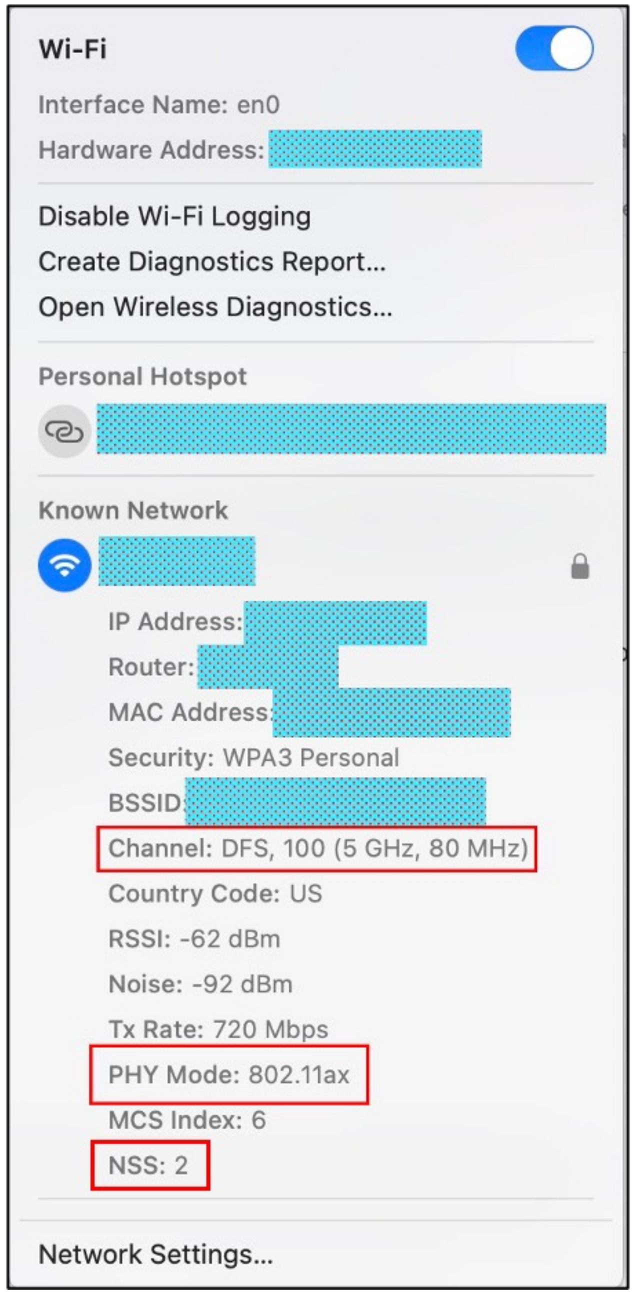
- The Mac results were obviously slower than the Windows 11 results, but 800 Mbps will often be just fine for many users for routine use.

- This was from a 2020 MacBook Pro M1 running MacOS Sequoia 15.4.1, connected over **5GHz 802.11ax wireless with two spatial streams and 80 MHz bandwidth.**

  Aside: How did we find out what our WiFi connection used?

  Under Mac OS/X, hold down the Option key while clicking the WiFi icon in the top menu bar. That will give you a display that should look like the panel that's over to the right ➔

  That summary gives us important information about our WiFi connection, including the protocol (802.11ax), the number of spatial streams (NSS) (2), and the bandwidth (80MHz).



Wi-Fi

Interface Name: en0
Hardware Address: [redacted]

Disable Wi-Fi Logging
Create Diagnostics Report...
Open Wireless Diagnostics...

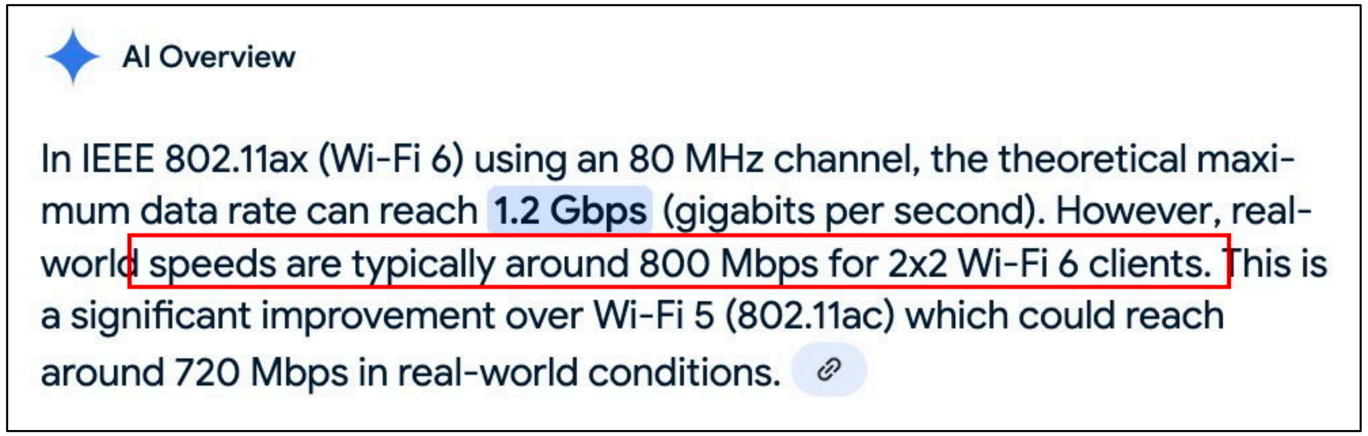
Personal Hotspot
[redacted]

Known Network
[redacted]
IP Address: [redacted]
Router: [redacted]
MAC Address: [redacted]
Security: WPA3 Personal
BSSID: [redacted]
Channel: DFS, 100 (5 GHz, 80 MHz)
Country Code: US
RSSI: -62 dBm
Noise: -92 dBm
Tx Rate: 720 Mbps
PHY Mode: 802.11ax
MCS Index: 6
NSS: 2

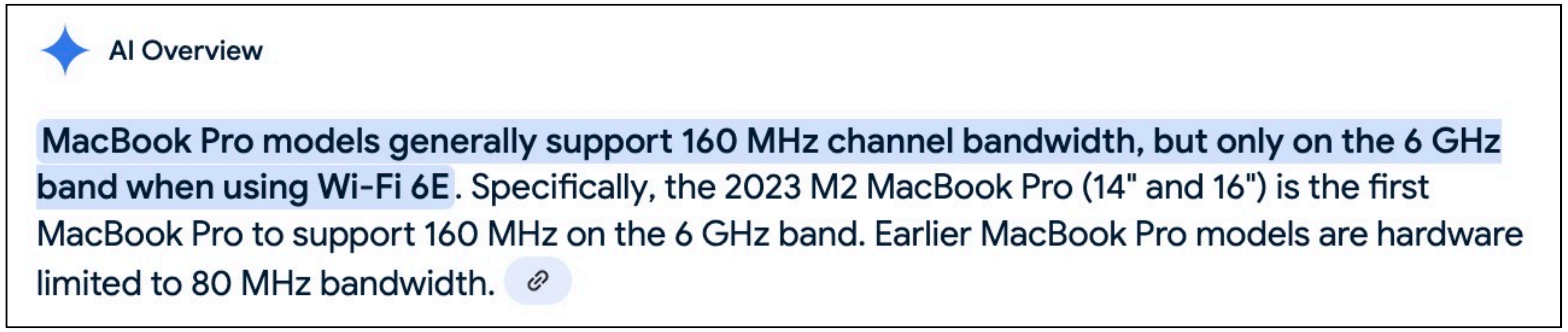
Network Settings...

Obligatory Sample AI-Derived Content: WiFi Throughput by Protocol

- 802.11**b** (real world throughput between 5-7 Mbps)
- 802.11**a** or 802.11**g** (real world throughput around 20 Mbps)
- 802.11**n** (real world throughput around 100 Mbps)
- 802.11**ac** (real world throughput of maybe 400-720 Mbps)
- 802.11**ax:**


AI Overview

In IEEE 802.11ax (Wi-Fi 6) using an 80 MHz channel, the theoretical maximum data rate can reach **1.2 Gbps** (gigabits per second). However, real-world speeds are typically around 800 Mbps for 2x2 Wi-Fi 6 clients. This is a significant improvement over Wi-Fi 5 (802.11ac) which could reach around 720 Mbps in real-world conditions.


AI Overview

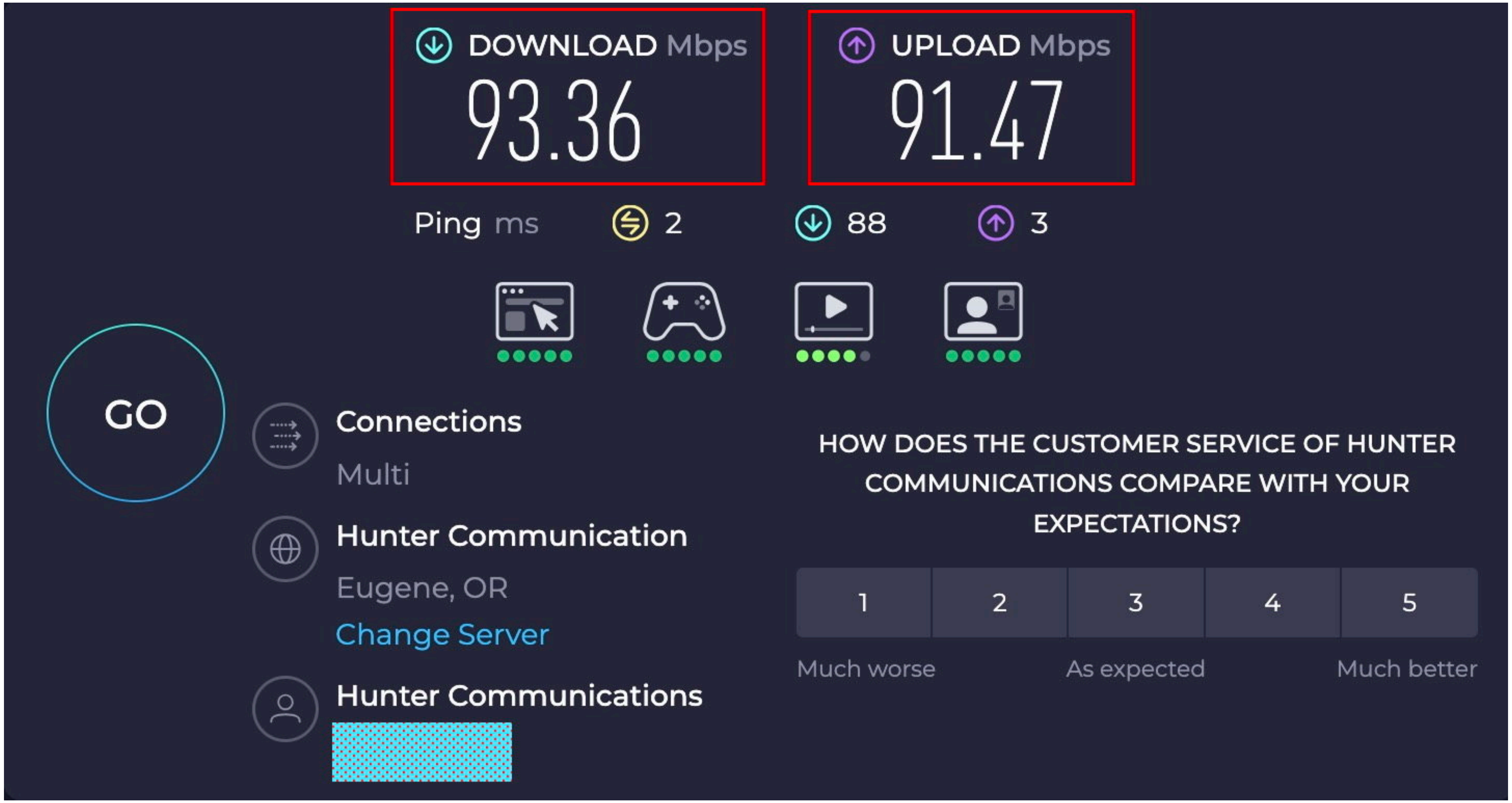
MacBook Pro models generally support 160 MHz channel bandwidth, but only on the 6 GHz band when using Wi-Fi 6E. Specifically, the 2023 M2 MacBook Pro (14" and 16") is the first MacBook Pro to support 160 MHz on the 6 GHz band. Earlier MacBook Pro models are hardware limited to 80 MHz bandwidth.

"Test Using Hardwired Connections, Not Wireless..."

- It is generally well accepted that high bandwidth network throughput testing should be done over a wired Ethernet (rather than WiFi) connection – WiFi just isn't fast enough in many cases, serving as an inherent bottleneck link.

- We had a genuine Apple USB-to-Ethernet adapter that we'd had knocking around for some years, so we decided to try it to see what sort of throughput we'd get from it...

Testing Our Macbook With the Apple Mac Wired Ethernet Adapter



DOWNLOAD Mbps
93.36

UPLOAD Mbps
91.47

Ping ms 2 88 3

GO

Connections
Multi

Hunter Communication
Eugene, OR
Change Server

Hunter Communications
[redacted]

HOW DOES THE CUSTOMER SERVICE OF HUNTER COMMUNICATIONS COMPARE WITH YOUR EXPECTATIONS?

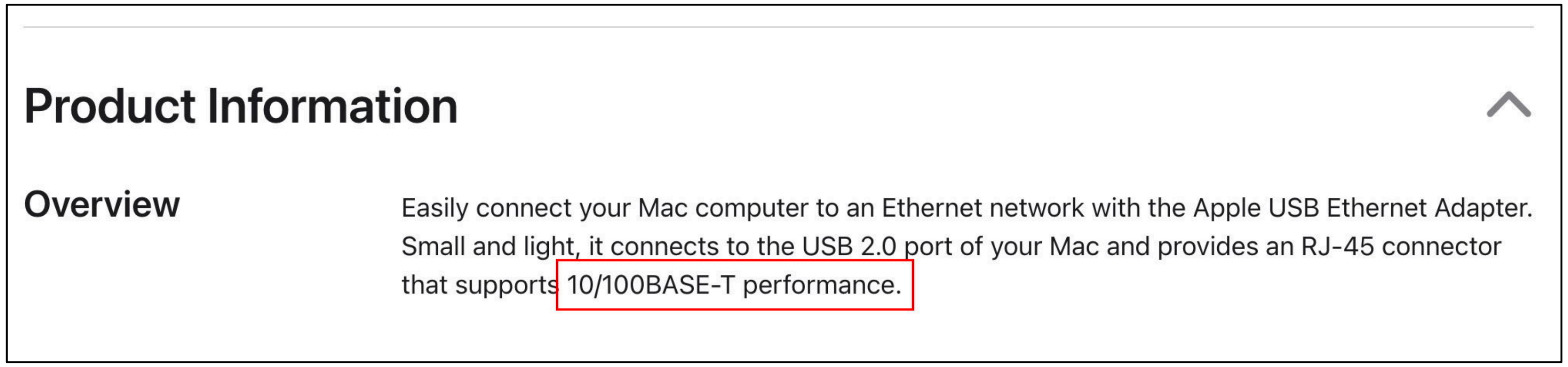
1 2 3 4 5

Much worse As expected Much better

Wired Ethernet Performance *WORSE* Than Wireless? *WHAT?*

- Could it be that our laptop ports are too slow? After all, we used a dongle plugged into a USB port – how fast do those ports go? Answer: even USB **3.0** supports 1 Gbps, but USB **3.1** (what we're using) supports up to 10 Gbps. **The issue here was not one of USB port throughput.**

- The "clue" here was that we're basically seeing Fast Ethernet (100 Mbps) speeds. So, **what's the spec for our genuine Apple Ethernet adapter?** Checking the Apple site, this old adapter is actually a **10/100**BASE-T unit! It's going as fast as it can, its just not very fast.

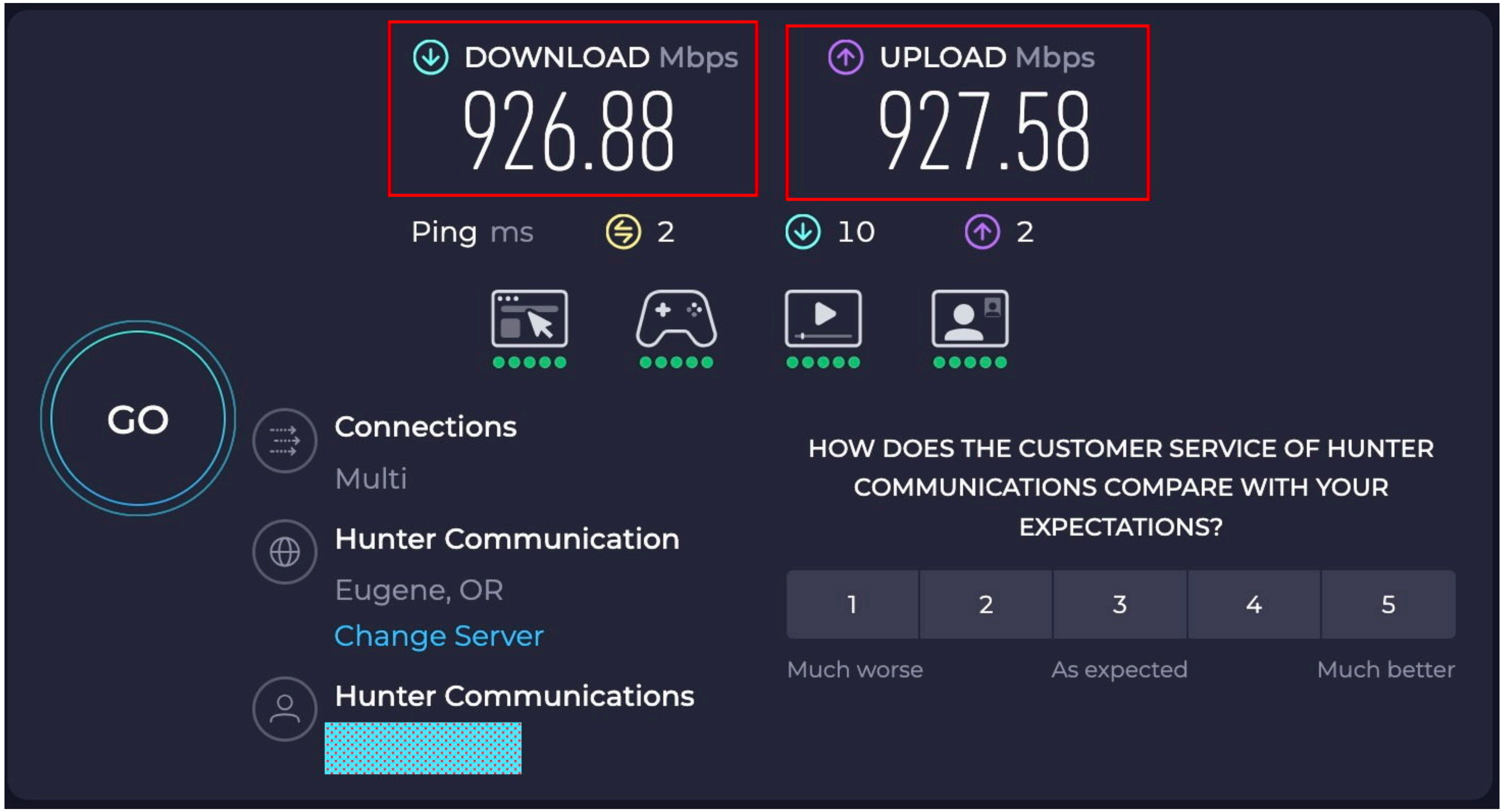

Product Information

Overview

Easily connect your Mac computer to an Ethernet network with the Apple USB Ethernet Adapter. Small and light, it connects to the USB 2.0 port of your Mac and provides an RJ-45 connector that supports 10/100BASE-T performance.

- **ALWAYS TEST! AND ALWAYS CHECK THE SPECIFICATIONS FOR THE GEAR YOU'RE USING!**

What If We Try A ***DIFFERENT*** Wired Ethernet Dongle on the Mac? (Such As An $11 UGREEN Gigabit USB-C to Ethernet Adapter)?



DOWNLOAD Mbps
926.88

UPLOAD Mbps
927.58

Ping ms 2 10 2

Connections
Multi

Hunter Communication
Eugene, OR
Change Server

Hunter Communications
[redacted]

HOW DOES THE CUSTOMER SERVICE OF HUNTER COMMUNICATIONS COMPARE WITH YOUR EXPECTATIONS?

1 2 3 4 5

Much worse As expected Much better

We Now Have A Pretty Good Starting Performance "Baseline"

- We saw near line-rate when we tested from our Calix WiFi router.
- We saw true line-rate (1000 Mbps) throughput from a Windows 11 PC, even when connecting wirelessly
- We saw just under 930 Mbps using a cheap wired Ethernet gigabit adapter on our old test Mac laptop.
- We even saw roughly 800 Mbps to our old test Mac laptop over 802.11ax wireless.
- That's all perfectly okay for most users, but we wanted substantially faster speeds.
- Fortunately, we could get 2.5 Gbps service for our ISP for not much more per month.

2. Upgrading Client-Side Gear for 2.5Gbps Networks

Bottom Line Up Front (BLUF):

You can build out what you need client-side for 2.5Gbps for not a lot of money, but be sure to test any hardware you buy to confirm it can do the job.

Upgrading Client Hardware: What Will We Need For 2.5Gbps?

- If we were going to continue to use laptops, **we needed faster Ethernet USB-to-Ethernet dongles** (neither of our test laptops came with a built-in 2.5Gbps Ethernet adapter).

  We'll use those external Ethernet dongles for both Mac laptops and Windows 11 laptops that need wired Ethernet support.

- **Mac:** If you're aren't a "Mac geek", you should know that there are two types of 2.5 Gbps USB Ethernet dongles for Mac laptops:

  One sort uses "networking control model" drivers **("ncm")**. These run fast.

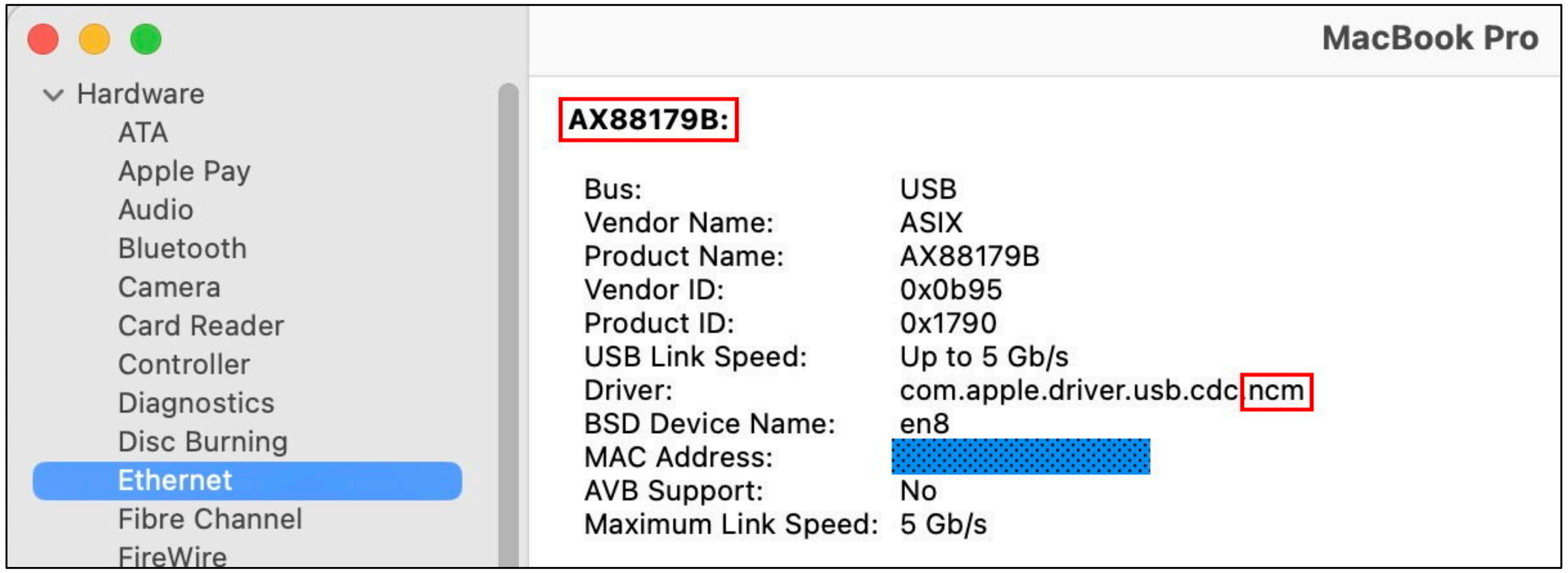
  The other sort uses an older "ethernet control model" driver **("ecm").** These run more slowly.

  It's rare to see details about Mac drivers on Amazon (or other online marketplaces) for USB dongles, so you may need to order devices and check to see what driver they use.

Checking Mac USB Dongle Ethernet Adapter Driver Usage

- Check: `About This Mac → More Info → General → About → System Report → Ethernet` to see what driver's actually being used.

  One example device (this one's our cheapo $11 UGreen 1 Gbps Adapter):



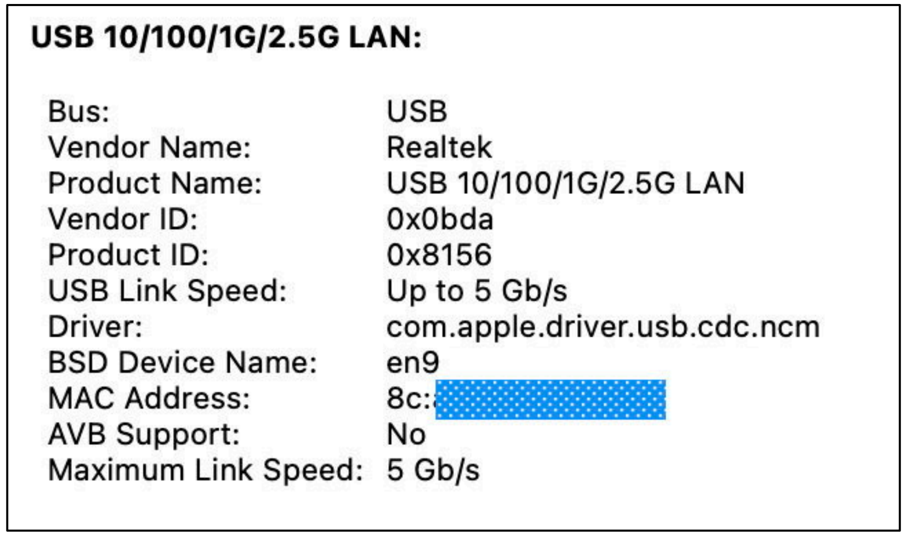
```
AX88179B:

  Bus:                  USB
  Vendor Name:          ASIX
  Product Name:         AX88179B
  Vendor ID:            0x0b95
  Product ID:           0x1790
  USB Link Speed:       Up to 5 Gb/s
  Driver:               com.apple.driver.usb.cdc.ncm
  BSD Device Name:      en8
  MAC Address:          [redacted]
  AVB Support:          No
  Maximum Link Speed:   5 Gb/s
```

- In spite of mentioning "USB Link Speed up to 5 Gb/s", this device was truly JUST a gigabit adapter (as the vendor clearly stated in their ad).

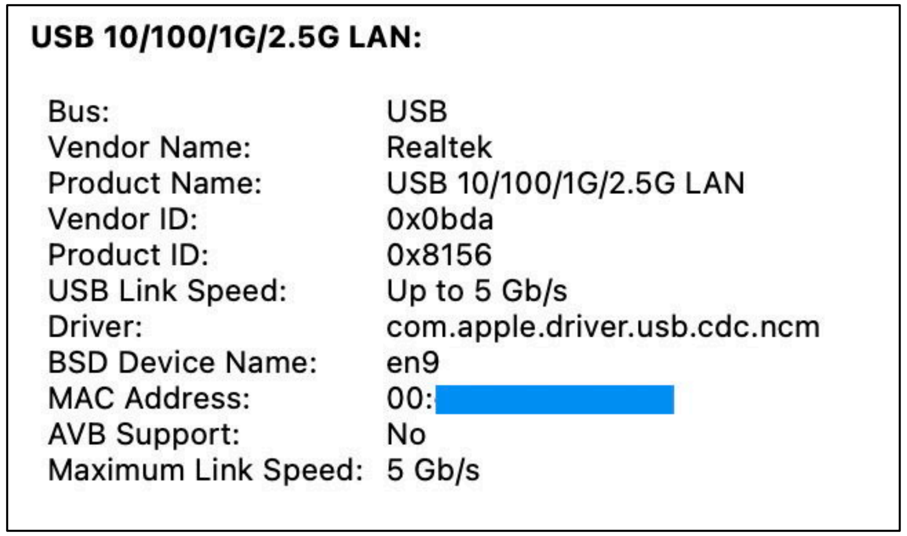
Checking a Couple of Actual 2.5Gbps USB-C to Ethernet "Dongles"...

Pluggable Brand 2.5G USB Dongle (NCM driver), https://www.amazon.com/dp/B084L4JL9K ($28.95)



```
USB 10/100/1G/2.5G LAN:

  Bus:                  USB
  Vendor Name:          Realtek
  Product Name:         USB 10/100/1G/2.5G LAN
  Vendor ID:            0x0bda
  Product ID:           0x8156
  USB Link Speed:       Up to 5 Gb/s
  Driver:               com.apple.driver.usb.cdc.ncm
  BSD Device Name:      en9
  MAC Address:          8c:[redacted]
  AVB Support:          No
  Maximum Link Speed:   5 Gb/s
```



```
USB 10/100/1G/2.5G LAN:

  Bus:                  USB
  Vendor Name:          Realtek
  Product Name:         USB 10/100/1G/2.5G LAN
  Vendor ID:            0x0bda
  Product ID:           0x8156
  USB Link Speed:       Up to 5 Gb/s
  Driver:               com.apple.driver.usb.cdc.ncm
  BSD Device Name:      en9
  MAC Address:          00:[redacted]
  AVB Support:          No
  Maximum Link Speed:   5 Gb/s
```

Anker Brand 2.5G USB Dongle (NCM driver), https://www.amazon.com/dp/B097N5WJY9 ($39.99)

Both used Realtek chips. The only real difference between the two? The MAC address...

What About 2.5 Gbps PCIe NICs for Our Alienware Linux x86 Box?

- We tested three different 2.5 Gbps PCIe network cards:
  - A Leagy 2.5GBase-T PCIe Network Card with RTL8125B Chipset, $12.99, https://www.amazon.com/dp/B0BYDH6HWY
  - A Glotrends LE8105 2.5Gbps PCIe Ethernet Network Card, RTL8125BG Chip, also $12.99, https://www.amazon.com/dp/B0BLN82WQ4
  - A TP-Link 2.5GB PCIe Network Card (TX201) (PCIe to 2.5 Gigabit Ethernet Network Adapter), $26.99, https://www.amazon.com/dp/B0BG685PKM
- Each of the three cards installed easily (no external drivers required) on our test Alienware Aurora R8 desktop minitower with an Intel Core i5 9400 (6 cores), 16 GB of memory, and a 2TB NVMe disk running Debian 12 (kernel: 6.1.0-34-amd64). Each delivered a satisfying 2.3 Gbps up and 2.3 Gbps down when tested using Speedtest.

Wired Ethernet Connections? Don't Forget About Your Cables!

- All Ethernet cables may "look the same," but there is a difference when you begin trying to go fast(er). "Vanilla" Cat5 cables won't reliably deliver multi-Gbps speeds. What will?
  - Cat**5E** cables are rated for **Gigabit** speeds and 100 MHz for a permanent link length (PLL) of up to 90 meters – too slow.
  - Cat**6** cables are rated for **10 Gigabit** speeds and 250 MHz for a PLL of up to 90 meters.
  - Cat**6A** cables are rated for **10 Gigabit** and **500 MHz** for a PLL of up to 90 meters.
  - Cat**8** cables will handle up to **40 Gigabit** speeds with 2000 MHz for a PLL of 24 meters, or they can do 10 Gigabit for a normal PLL of up to 90 meters.
- Rather than worrying about "what cable's what," **my recommendation is to only buy and use Cat6A or Cat8** for everything moving forward (in most cases, pricing differences between the various categories will be negligible, unless you're buying miles of cable).

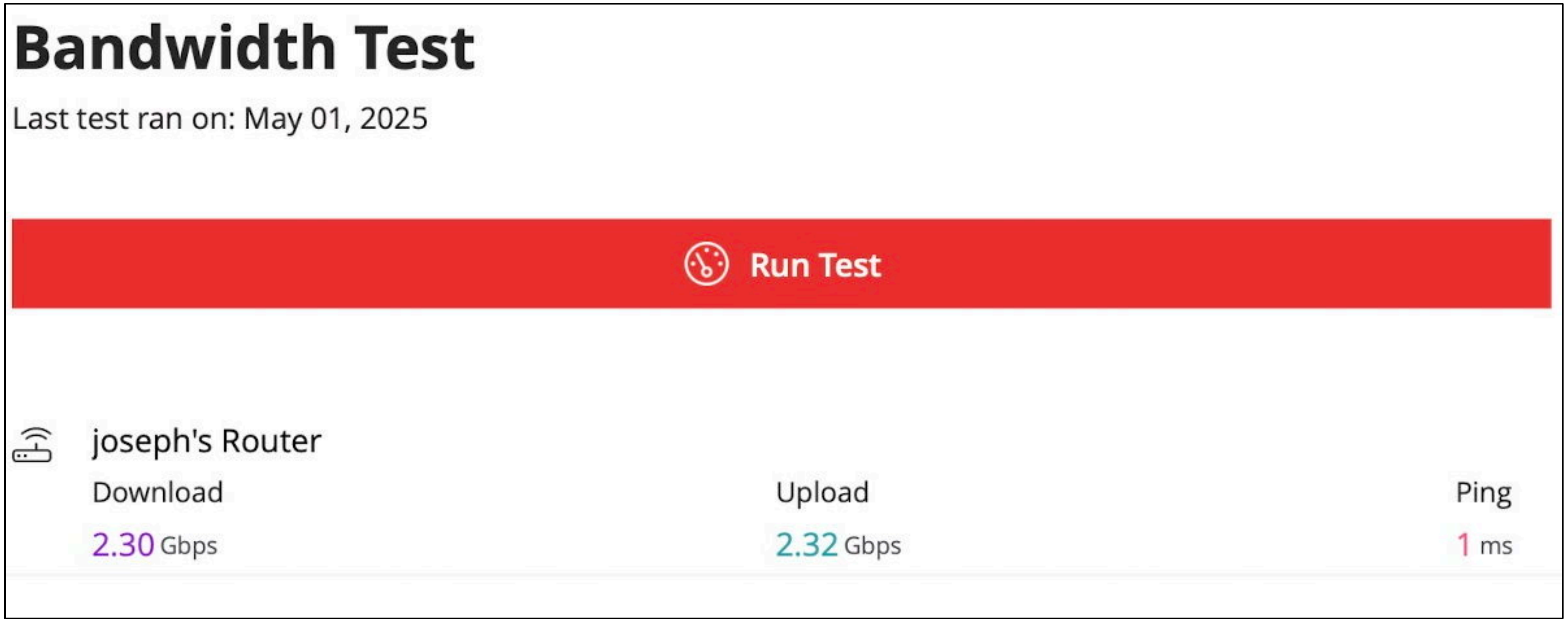
3. Remaining 2.5Gbps Network Infrastructure

Bottom Line Up Front (BLUF):

At least some of your network gear will likely be provided by your ISP. Confirm that it is correctly configured for the speed you bought. Recognize that you may need to add a supplemental switch to get the Ethernet port count you need. Test and CONFIRM your throughput!

Our Provider-Provided Wireless Calix Router Was Upgraded for Free!

- This was done by a provider technician, and it tested OK from the provider's app. This was another Calix Gigaspire, just a different (more powerful) model, the u6.



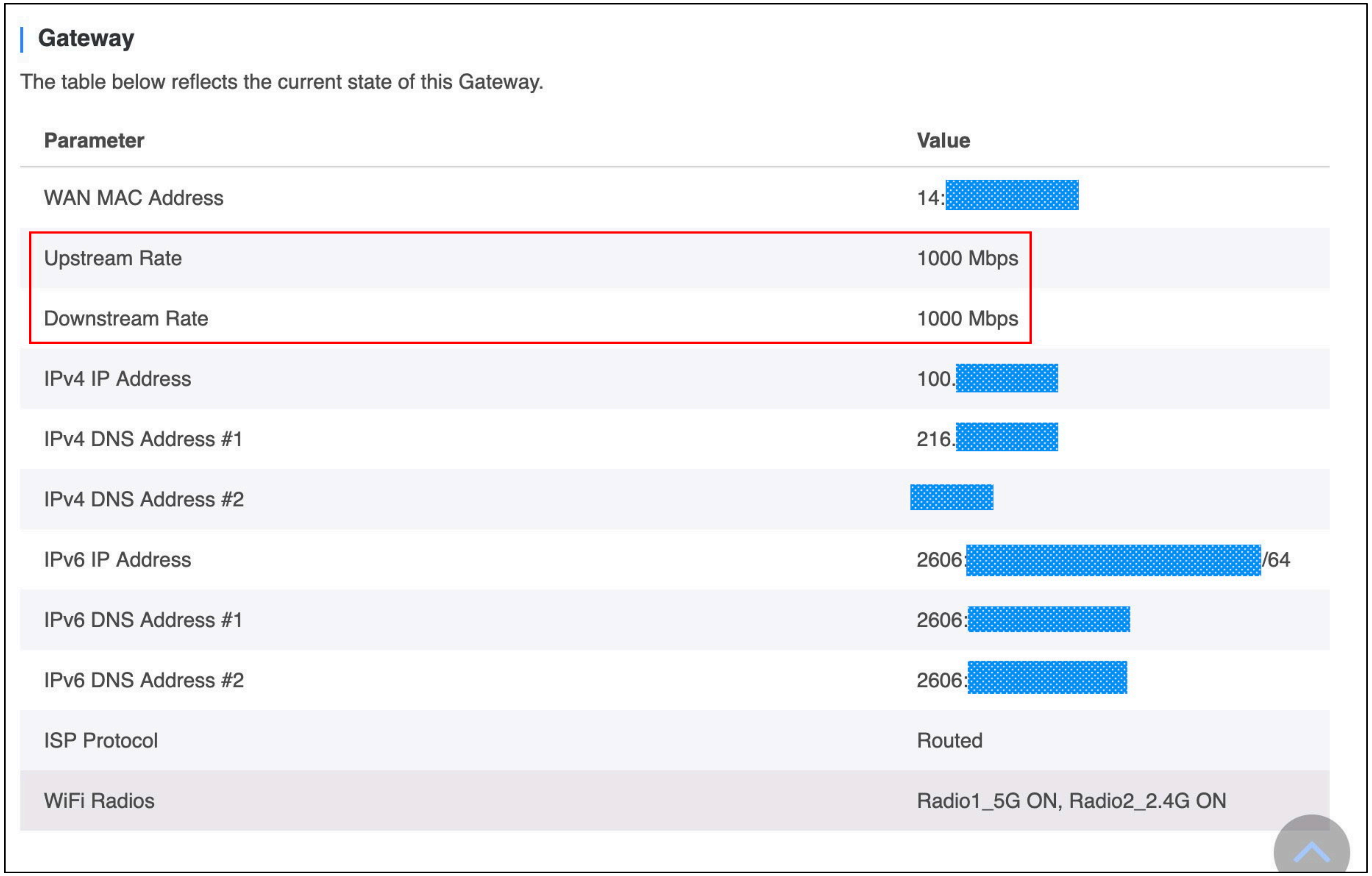
Bandwidth Test

Last test ran on: May 01, 2025

joseph's Router

Download	Upload	Ping
2.30 Gbps	2.32 Gbps	1 ms

The router gateway was initially left configured for just gigabit

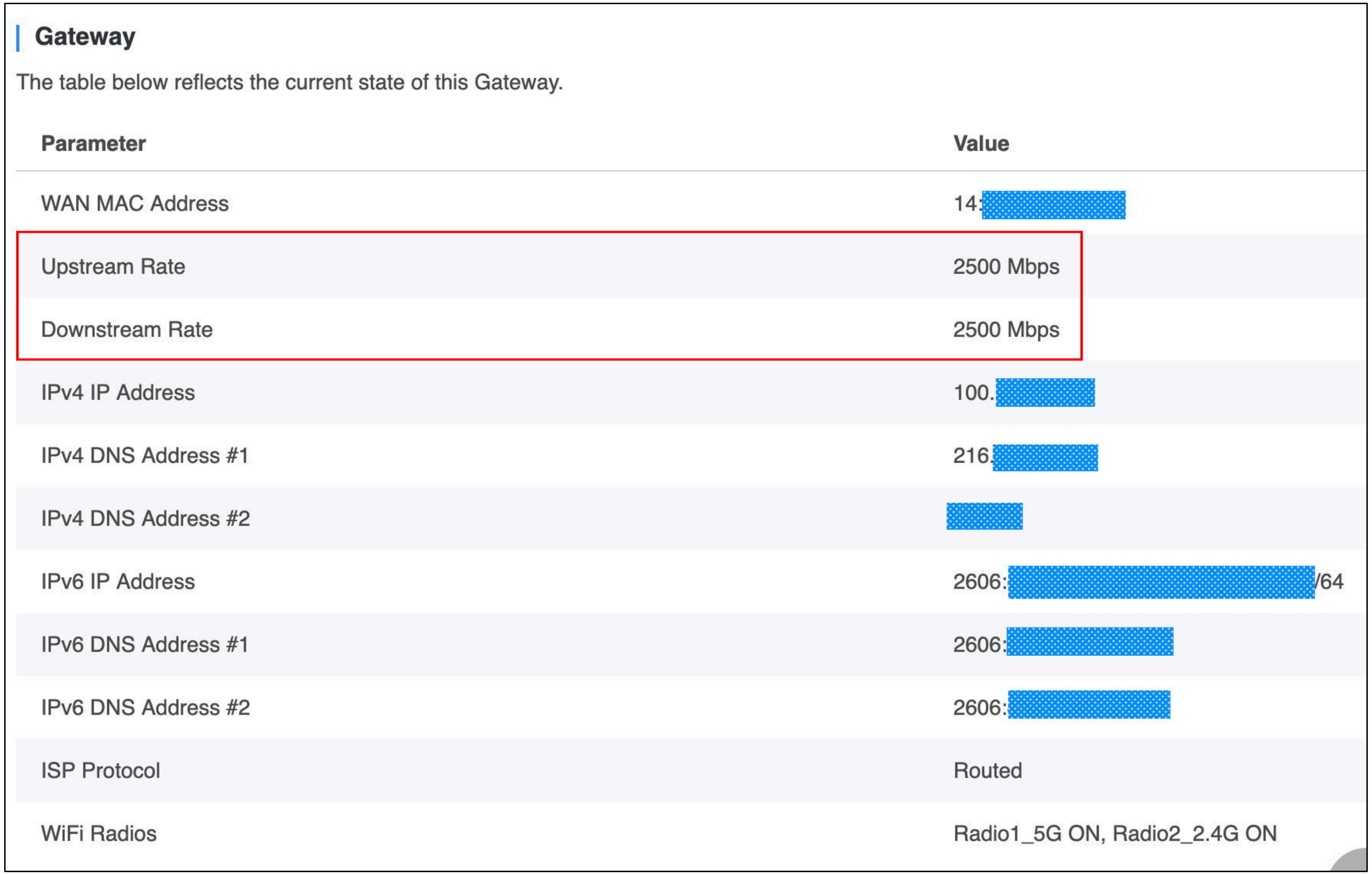


Gateway

The table below reflects the current state of this Gateway.

Parameter	Value
WAN MAC Address	14:[redacted]
Upstream Rate	1000 Mbps
Downstream Rate	1000 Mbps
IPv4 IP Address	100.[redacted]
IPv4 DNS Address #1	216.[redacted]
IPv4 DNS Address #2	[redacted]
IPv6 IP Address	2606:[redacted]/64
IPv6 DNS Address #1	2606:[redacted]
IPv6 DNS Address #2	2606:[redacted]
ISP Protocol	Routed
WiFi Radios	Radio1_5G ON, Radio2_2.4G ON

A support ticket with the provider (and a reboot) got that corrected



Gateway

The table below reflects the current state of this Gateway.

Parameter	Value
WAN MAC Address	14:[redacted]
Upstream Rate	2500 Mbps
Downstream Rate	2500 Mbps
IPv4 IP Address	100.[redacted]
IPv4 DNS Address #1	216.[redacted]
IPv4 DNS Address #2	[redacted]
IPv6 IP Address	2606:[redacted]/64
IPv6 DNS Address #1	2606:[redacted]
IPv6 DNS Address #2	2606:[redacted]
ISP Protocol	Routed
WiFi Radios	Radio1_5G ON, Radio2_2.4G ON

2.5 or 10 Gig Physical RJ45 Port Counts

- CPE devices provided to customers by ISP's may have a **limited number of faster-than-gigabit RJ45 ports** (ours had just ONE upstream and one downstream 10gig port).
- If you need additional 2.5 Gbps/10 Gbps ports, you'll need to **purchase a switch.**
- The cost for a 2.5gbps or 10gbps Ethernet switch will vary depending on multiple factors:
  - **Who makes the switch?** (This is relevant if you're subject to **National Defense Authorization Act (NDAA) restrictions** and/or the **Trade Agreements Act (TAA)).**
  - Is the switch **managed** or **unmanaged**? Put another way, is the switch just "plug-and-play" (effectively functioning as a "one-port-to-multiport adapter"), or can you fully configure & monitor it, etc.? (Can I use VLANS? Jumbo frames? Read SNMP counters?)
  - What's the switch's **port count?** 4 ports? 8 ports? 24 ports?
  - Does the switch support **power-over-Ethernet (POE),** thereby making it easy to deploy infrastructure (such as WiFi access points) even where "wall power" is unavailable?
  - Are ports **copper** (100/1000/2.5G/5G/10Gbase-T) or **fiber optic** (perhaps SFP+)?
  - What's the **architecture of the switch?** Cut-through, or store-and-forward?
  - Is the switch **fully redundant?** Can it lose a power supply or fan w/o going offline?

A Simple Unmanaged 2.5 Gig Switch

- The unmanaged 2.5 Gbps switch we went with was a TP-Link Omada TL-SG108S-M2 8-Port 2.5G Ethernet Switch, $199.99 ($15/port).

- It's small but still rather beefy feeling, only 6.2x4x1" in size. This is a store-and-forward switch, reportedly able to handle 40 Gbps/29.8 Mpps. It's also quiet due to being fan-less. It takes its power from a "wall-wart"-style power supply.

- It's obviously NOT a carrier grade switch, but we hoped it would work for our needs (and it seems to be doing so, at least so far). The box the switch came in states the switch has a 3-year warranty, but the matrix at https://www.tp-link.com/us/support/replacement-warranty/ is somewhat unclear (and there's a long list of exclusions and conditions).

Greater-Than-Gbps Switches CAN Sometimes Involve Fiber Ports

- If you ONLY see **"10Gbase-T" or "2.5Gbase-T" or "RJ45"** mentioned when you're shopping for an Ethernet switch, you're looking at switches that support **copper Ethernet.** That's typically what you're going to want for most smaller networks.

- **Sometimes, however, you may see "SFP+" mentioned for at least some switch ports. If so, BEWARE, because that means FIBER.** Are you sure that's what you want or need?

  Yes, you CAN get SFP+ network interface cards for your Windows or Linux desktop systems, and yes, you can even get 10 Gbps SFP+ to USB dongles for your laptops. But using fiber jumpers instead of just using Cat6E or Cat8 copper Ethernet cords is a bit overkill unless you have distance issues (runs longer than 90-100 meters) or you need to protect against electrical signals from electrical storms or poorly shielded equipment, etc.

  If you DO end up buying an SFP+-based switch and you subsequently want to connect 2.5 or 10Gbps copper Ethernet cables into those SFP+ ports, you CAN do so, you'll just need to ALSO purchase pluggable transceivers, typically $25-$30+ per port.

Testing

- In our earlier baseline testing, we used Speedtest. We could continue to use that, but that's **not the only bandwidth speed testing option** that's available.

- For higher speeds, we rather like **iperf3**, available from https://iperf.fr/

- You'll need a **remote iperf3 endpoint to test against.** One list of available test hosts is **https://iperf3serverlist.net/**

  In picking a host from that list, note the **location of these test hosts** (the importance of latency will be discussed later in this talk).

  Also note the **configured bandwidth.** If you're testing a 2.5 Gbps (or faster!) link, a test endpoint with just 1 Gbps obviously won't be very satisfactory.

Throughput Tested From the Mac Using Wired Ethernet and iperf3

```
$ iperf3 -c speedtest.sea11.us.leaseweb.net -p 5201-5210
Connecting to host speedtest.sea11.us.leaseweb.net, port 5201
[  5] local 2606:[snip] port 59736 connected to 2607:[snip] port 5201
[ ID] Interval           Transfer     Bitrate         Retr  Cwnd
[  5]   0.00-1.00   sec   281 MBytes  2.36 Gbits/sec    0   9.74 MBytes
[  5]   1.00-2.00   sec   276 MBytes  2.32 Gbits/sec    0   9.74 MBytes
[  5]   2.00-3.00   sec   278 MBytes  2.33 Gbits/sec    0   9.74 MBytes
[  5]   3.00-4.00   sec   276 MBytes  2.32 Gbits/sec    0   9.74 MBytes
[  5]   4.00-5.00   sec   276 MBytes  2.32 Gbits/sec    0   9.74 MBytes
[  5]   5.00-6.00   sec   278 MBytes  2.33 Gbits/sec    0   9.74 MBytes
[  5]   6.00-7.00   sec   276 MBytes  2.32 Gbits/sec    0   9.74 MBytes
[  5]   7.00-8.00   sec   276 MBytes  2.32 Gbits/sec    0   9.74 MBytes
[  5]   8.00-9.00   sec   278 MBytes  2.33 Gbits/sec    0   9.74 MBytes
[  5]   9.00-10.00  sec   276 MBytes  2.32 Gbits/sec    0   9.74 MBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Retr
[  5]   0.00-10.00  sec  2.71 GBytes  2.32 Gbits/sec    0             sender
[  5]   0.00-10.02  sec  2.67 GBytes  2.29 Gbits/sec                  receiver
```

What About The Other Direction?

```
$  iperf3 -c speedtest.sea11.us.leaseweb.net -p 5201-5210 -R
Connecting to host speedtest.sea11.us.leaseweb.net, port 5201
Reverse mode, remote host speedtest.sea11.us.leaseweb.net is sending
[  5] local 2606:[snip] port 58860 connected to 2607:[snip] port 5201
[ ID] Interval           Transfer     Bitrate
[  5]   0.00-1.00   sec   244 MBytes  2.05 Gbits/sec
[  5]   1.00-2.00   sec   275 MBytes  2.31 Gbits/sec
[  5]   2.00-3.00   sec   277 MBytes  2.32 Gbits/sec
[  5]   3.00-4.00   sec   277 MBytes  2.32 Gbits/sec
[  5]   4.00-5.00   sec   277 MBytes  2.32 Gbits/sec
[  5]   5.00-6.00   sec   277 MBytes  2.32 Gbits/sec
[  5]   6.00-7.00   sec   277 MBytes  2.32 Gbits/sec
[  5]   7.00-8.00   sec   277 MBytes  2.32 Gbits/sec
[  5]   8.00-9.00   sec   277 MBytes  2.32 Gbits/sec
[  5]   9.00-10.00  sec   277 MBytes  2.32 Gbits/sec
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Retr
[  5]   0.00-10.02  sec  2.69 GBytes  2.31 Gbits/sec  2376             sender
[  5]   0.00-10.00  sec  2.67 GBytes  2.29 Gbits/sec                  receiver

iperf Done.
```

Why Do We "Only" Get ~2.3 Gbps instead of A Full 2.5 Gbps?

The issue is largely one of network overhead...

```
12-byte interframe gap
8-byte preamble
Ethernet header
       DST MAC 6-bytes
       SRC MAC 6-bytes
       Type 2-bytes
1500-byte frame
       IPv4 header 20-bytes
       Standard TCP header 20-bytes
       Timestamps (RFC7323) 10-bytes
       SACK Permitted (RFC2018) 2-bytes
       SACK Blocks (up to 3-4 blocks, 8 bytes each) 24-32 bytes
       Leaving 1500 - (76 to 84 bytes for headers) <= 1416 to 1424 for payload
Ethernet trailer
       FCS 4-bytes
```

1424/(12+8+14+1500+4) * 2500 = **2.314** Gbps (for IPv4).
Add 20 more bytes of headers if using IPv6. Doing that, we get **2.282** Gbps for IPv6.

Note: Our Realized Speed Can Also Be Impacted By OTHER TRAFFIC

- We might fantasize that our path to anywhere on the Internet is **completely unloaded,** just sitting there **totally idle,** awaiting our measurement (or other) traffic.
- That's likely not the case. Other people do use the Internet, so if you don't see the full bandwidth you expect, test again later. Someone else may just have been using part of "your" bandwidth.
- We'll also discuss (later in this session) how your choice of TCP congestion control algorithm can impact your ability to "take your fair share" or "MORE THAN your fair share" of network bandwidth across shared network links.
- But in any event, **we now have a functional 2.5 Gbps testbed.**
- Now let's use it to understand network performance.

Part II. Understanding Network Performance

Our Implicit Goal: Use All or Most of The Bandwidth We've Got

- That seems like it should be self-evident, but **a surprisingly large fraction of people may never actually use the network capacity they've purchased** (sort of like a sports car enthusiast who never really push their car over 65-70 MPH).
- Of course, some users may also want something **OTHER THAN high throughput** (for example, gamers may be primarily interested in low lag). For the purposes of this training, we're going to assume **high utilization/high throughput** is the primary objective.
- Achieving high utilization may mean different things for **different workloads.**
- Are you **sharing your bandwidth with many other users,** as happens with the M3AAWG wireless conference network at our meetings? If so, that sort of common scenario may more-or-less "just work" out of the box.
- Or are you trying to ensure that **ONE user and ONE SINGLE application can take full advantage of all the available network capacity?** Does the application connect to many different sites all over the world, or largely just to one specific remote site (perhaps your office or a cloud site)? Highly specialized consistent workloads may be easier to optimize.

Factors Impacting Your Network Performance

- Multiple factors can impact your network performance, including:

  - **Latency between you and your destination(s)**
  - **Packet loss**
  - **Congestion control algorithms in use**

- **These factors will all interact.**

- We're going to talk about them now, beginning with the latency between you and your destination(s).

4. Latency: The Bandwidth Delay Problem for Long Fat Network (LFN) Connections

Bottom Line Up Front (BLUF):

All network connections have latency. Ping can be used to estimate round-trip latency. The TCP protocol is potentially limited by buffer sizing when TCP is used over long fast networks, but you can tune those. We explain how to do so for Apple Macs, Windows 11, and Debian 12.

Estimating Network Latency With ping

- It takes time for your network traffic to go from one location to another over the Internet.
- That's **"network latency."** Network latency can and will materially impact the performance of TCP network traffic, even if its just 50 or 100 msec worth of delay.
- But do you know **how much** latency exists between YOUR location and destination sites?
- There are many **paid commercial network monitoring tools** that can be used to track latency, BUT M3AAWG is non-commercial so we won't discuss specific products (if interested, see Gartner's report on "Digital Experience Monitoring" for some pointers).
- **We can get a basic estimate of round-trip latency just using free/open-source ping.**

Finding Round Trip Latency With Ping: Oregon to a Random Site

Going from our test location in Eugene, Oregon to the IETF's main web site, doing six pings (-c 6) with the ping command, we see:

```
$ ping -c 6 www.ietf.org
PING www.ietf.org (104.16.45.99): 56 data bytes
64 bytes from 104.16.45.99: icmp_seq=0 ttl=54 time=15.710 ms
64 bytes from 104.16.45.99: icmp_seq=1 ttl=54 time=16.694 ms
64 bytes from 104.16.45.99: icmp_seq=2 ttl=54 time=16.785 ms
64 bytes from 104.16.45.99: icmp_seq=3 ttl=54 time=15.727 ms
64 bytes from 104.16.45.99: icmp_seq=4 ttl=54 time=23.354 ms
64 bytes from 104.16.45.99: icmp_seq=5 ttl=54 time=17.417 ms

--- www.ietf.org ping statistics ---
6 packets transmitted, 6 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 15.710/17.614/23.354/2.636 ms
```

Those are pretty reasonable levels of latency.

Going Coast-to-Coast Involves MORE Latency

St. Joseph's University is located in Philadelphia, Pennsylvania. Again, testing from Eugene, we see:

```
$ ping -c 6 www.sju.edu
PING www.sju.edu (52.202.179.64): 56 data bytes
64 bytes from 52.202.179.64: icmp_seq=0 ttl=40 time=78.068 ms
64 bytes from 52.202.179.64: icmp_seq=1 ttl=40 time=80.726 ms
64 bytes from 52.202.179.64: icmp_seq=2 ttl=40 time=88.071 ms
64 bytes from 52.202.179.64: icmp_seq=3 ttl=40 time=88.647 ms
64 bytes from 52.202.179.64: icmp_seq=4 ttl=40 time=86.132 ms
64 bytes from 52.202.179.64: icmp_seq=5 ttl=40 time=86.484 ms

--- www.sju.edu ping statistics ---
6 packets transmitted, 6 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 78.068/84.688/88.647/3.915 ms
```

Paths (and Hence Network Latency) Won't Always Be "Optimal"

- In a best-case world, **traffic would go directly ("as the crow flies")** from source to destination and back. This would **minimize the latency involved.**

- In reality, traffic flows twist and turn as **fiber runs follow various right-of-way paths.**

- Networks may even send traffic "in the wrong direction," perhaps to a **regional hub,** before that traffic finally heads out "in the right direction" (this is the same sort of back-haul that you may run into when flying commercially out of smaller airports).

- Multiple networks may work together to deliver your traffic. They may only **interconnect at certain locations (**which may not necessarily be directly on the natural direct route).

- A direct link might be down, requiring traffic to take a **"detour"** onto a less optimal route.

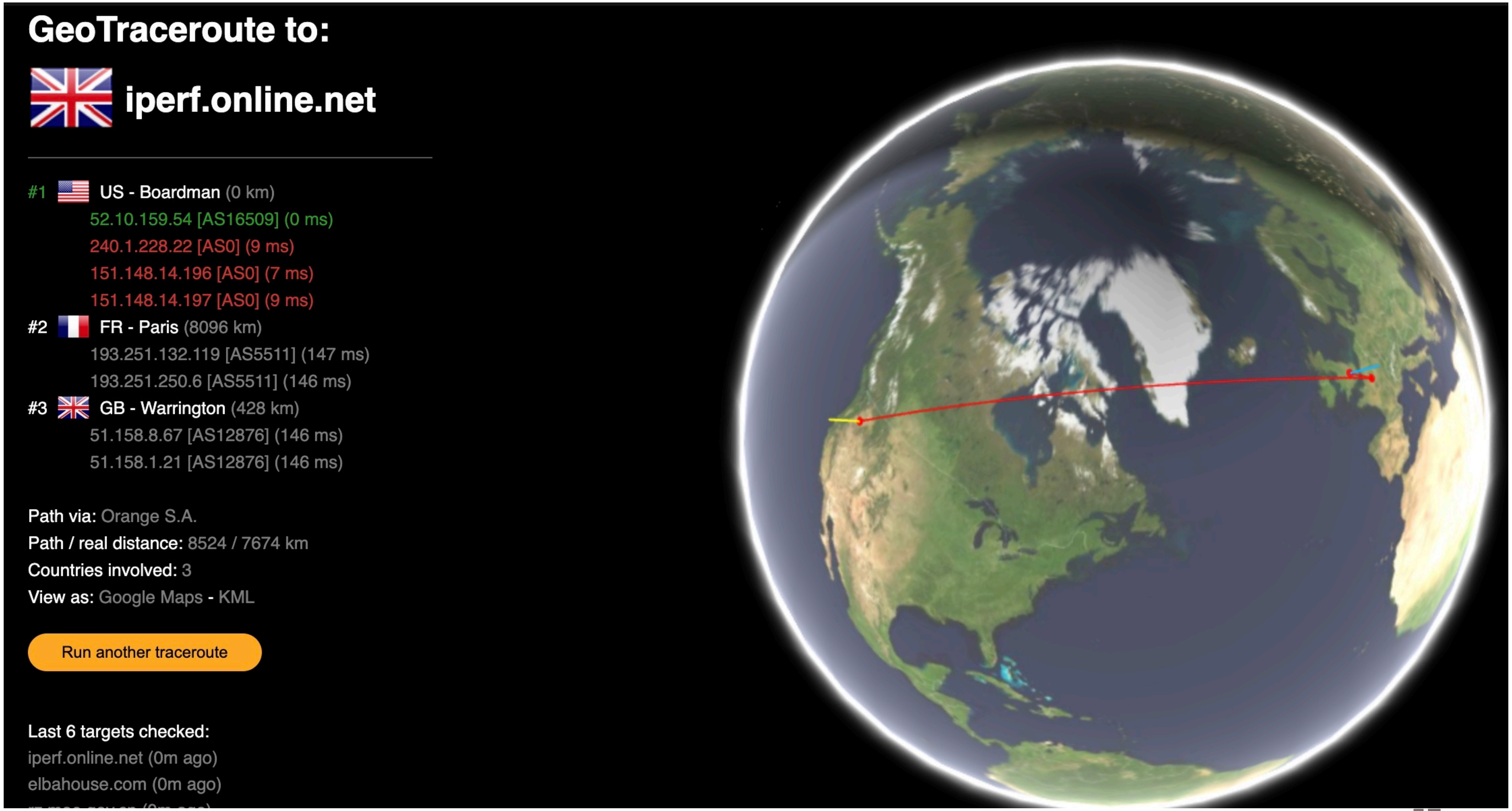
- **VPNs** may intentionally route traffic through remote locations (Iceland, Panama, ...)

Going From the West Coast of the U.S. To A Site In Europe

```
$ ping -c 6 iperf.online.net
PING iperf.online.net (51.158.1.21): 56 data bytes
64 bytes from 51.158.1.21: icmp_seq=0 ttl=43 time=158.980 ms
64 bytes from 51.158.1.21: icmp_seq=1 ttl=43 time=159.222 ms
64 bytes from 51.158.1.21: icmp_seq=2 ttl=43 time=167.050 ms
64 bytes from 51.158.1.21: icmp_seq=3 ttl=43 time=165.766 ms
64 bytes from 51.158.1.21: icmp_seq=4 ttl=43 time=161.161 ms
64 bytes from 51.158.1.21: icmp_seq=5 ttl=43 time=169.655 ms

--- iperf.online.net ping statistics ---
6 packets transmitted, 6 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 158.980/163.639/169.655/4.077 ms
```

Theoretically, 51.158.1.21 is in Paris, France, but note the following (free) GeoTraceroute report showing that IP may somehow actually be in Warrington UK? (Dunno. Geolocation can be a bit of a black art)



GeoTraceroute to:

iperf.online.net

#1 US - Boardman (0 km)
52.10.159.54 [AS16509] (0 ms)
240.1.228.22 [AS0] (9 ms)
151.148.14.196 [AS0] (7 ms)
151.148.14.197 [AS0] (9 ms)
#2 FR - Paris (8096 km)
193.251.132.119 [AS5511] (147 ms)
193.251.250.6 [AS5511] (146 ms)
#3 GB - Warrington (428 km)
51.158.8.67 [AS12876] (146 ms)
51.158.1.21 [AS12876] (146 ms)

Path via: Orange S.A.
Path / real distance: 8524 / 7674 km
Countries involved: 3
View as: Google Maps - KML

Run another traceroute

Last 6 targets checked:
iperf.online.net (0m ago)
elbahouse.com (0m ago)

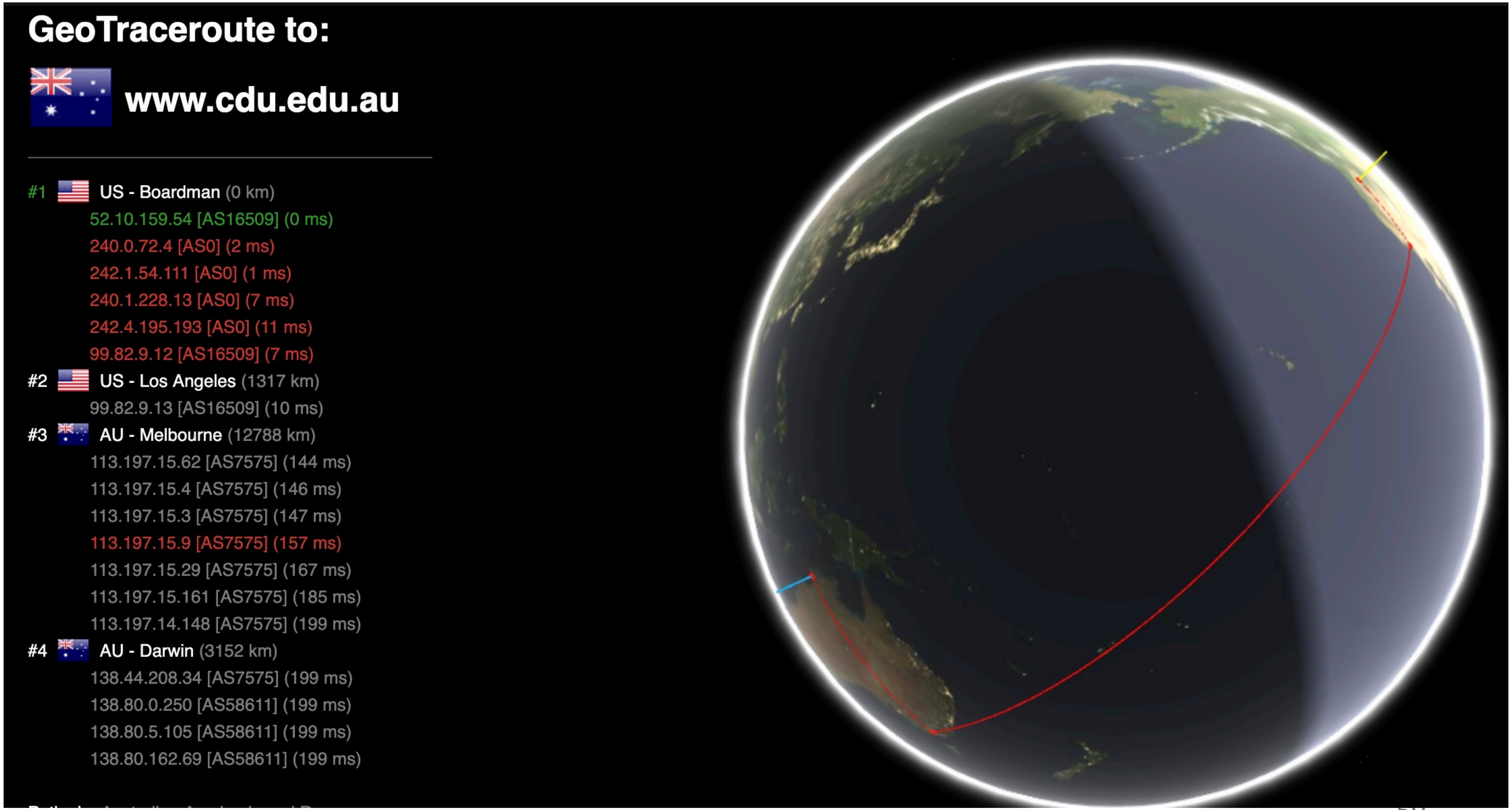
Going Even Further: Darwin University, Northern Territory, AU

Charles Darwin University, located in Australia's Northern Territory ("one-sixth of Australia’s landmass and yet is home to just one per cent of the population"), is still further away:

```
$ ping -c 6 www.cdu.edu.au
PING www.cdu.edu.au (138.80.162.69): 56 data bytes
64 bytes from 138.80.162.69: icmp_seq=0 ttl=233 time=208.307 ms
64 bytes from 138.80.162.69: icmp_seq=1 ttl=233 time=208.869 ms
64 bytes from 138.80.162.69: icmp_seq=2 ttl=233 time=251.510 ms
64 bytes from 138.80.162.69: icmp_seq=3 ttl=233 time=276.133 ms
64 bytes from 138.80.162.69: icmp_seq=4 ttl=233 time=215.322 ms
64 bytes from 138.80.162.69: icmp_seq=5 ttl=233 time=208.593 ms

--- www.cdu.edu.au ping statistics ---
6 packets transmitted, 6 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 208.307/228.122/276.133/26.333 ms
```

It's hard to even visually depict that path on the globe, given how far it's going!



GeoTraceroute to:

www.cdu.edu.au

```
#1  US - Boardman (0 km)
      52.10.159.54 [AS16509] (0 ms)
      240.0.72.4 [AS0] (2 ms)
      242.1.54.111 [AS0] (1 ms)
      240.1.228.13 [AS0] (7 ms)
      242.4.195.193 [AS0] (11 ms)
      99.82.9.12 [AS16509] (7 ms)
#2  US - Los Angeles (1317 km)
      99.82.9.13 [AS16509] (10 ms)
#3  AU - Melbourne (12788 km)
      113.197.15.62 [AS7575] (144 ms)
      113.197.15.4 [AS7575] (146 ms)
      113.197.15.3 [AS7575] (147 ms)
      113.197.15.9 [AS7575] (157 ms)
      113.197.15.29 [AS7575] (167 ms)
      113.197.15.161 [AS7575] (185 ms)
      113.197.14.148 [AS7575] (199 ms)
#4  AU - Darwin (3152 km)
      138.44.208.34 [AS7575] (199 ms)
      138.80.0.250 [AS58611] (199 ms)
      138.80.5.105 [AS58611] (199 ms)
      138.80.162.69 [AS58611] (199 ms)
```

An Aside: Potential Problems With Using Ping To Estimate Latency

- While ping seems like a convenient tool for estimating latency, it can run into some complications, including:
- **Content Distribution Networks (CDNs)** have become very popular as a way to improve web site performance -- that's why I didn't show ping times to Columbia or NYU or U Penn institutional home pages: they all use content distribution networks to accelerate the delivery of their web sites. When CDNs get used, they artificially reduce the latency to specific CDN-ified web pages (such as the institution's main web page), but everything that's actually hosted "on campus" will still have normal latencies.
- **Blocking of ICMP traffic:** Ping uses ICMP traffic to find and report latency. Some sites may block ICMP traffic entirely. When that happens, ICMP-based-ping won't work. M3AAWG's own website is an example of a site that blocks ICMP.
- **ICMP traffic may be passed but given low priority by network devices:** Some network devices will pass ICMP traffic, but at a very low priority relative to all other network traffic. That lower-priority handling may artificially-increase reported latency times or loss.
- Ping's not perfect, but it's the basic tool we've got in our open source toolkit.

We Now Understand How to At Least Roughly ESTIMATE Latency

- **Now let's see how latency IMPACTS some Internet traffic.**

- We'll need to provide a quick **networking refresher, first.**

- To begin with, you all probably already know that Internet traffic is broken up into chunks called "packets." Each "packet" is a chunk of data (normally <= 1500 octets in length).

  Short messages may fit into a single packet; longer messages may be broken up and sent using multiple packets.

  When network packet traffic gets to its destination, the stream of packets get reassembled and consumed by the application.

Most Internet Packet Traffic Will Either Be UDP or TCP

- **UDP** is connectionless, "fire and forget," and MAY have packet loss. UDP is used for some important applications (including most DNS traffic) plus VoIP and video traffic. UDP traffic can be spoofed (since it doesn't involve a three-way handshake the way TCP does).

  UDP normally DOEN'T have wide-area performance problems.

- **TCP** is connection-oriented, has assured delivery (effectively zero packet loss), and includes flow control. TCP is used for HTTP (and HTTPS), SMTP, and many other popular Internet applications. The TCP "three-way handshake" (SYN, SYNACK, ACK) makes TCP flows un-spoofable.

  TCP traffic is what people struggle with when it comes to going fast over long fat pipes.

- Traffic can also include other miscellaneous traffic, such as ICMP, SCTP and QUIC, but we mostly won't worry about those in this talk.

"Reliable Delivery" – The Root of TCP Performance Challenges

- Packets sent via TCP enjoy "reliable delivery" – that is, packets sent by the sender WILL be received by the receiver, and that's certainly terrific. **The way TCP accomplishes that is by having the sender "hold onto" a copy of all the packets that are "in flight" until they've been acknowledged by the receiver.** If a packet doesn't get acknowledged as having been received, TCP can (and will) resend it from the copy it still has in its buffer.

- **But here's the potential high performance stumbling block: TCP transmission buffers need to hold ALL the packets that are "in flight" UNTIL they get acknowledged.**

- This means that the **faster the network** or the **longer the latency** between the source and the destination, the greater the number of packets that must be retained for potential retransmission. **If that buffer fills up, new packets can't be sent until the buffer drains enough to accommodate the new packets. Long fat network links can be demanding!**

- Network buffers can be made larger on the sending system, but **each connection** will have its own buffer, and systems have finite memory available to be dedicated for buffer use.

Other Constraints Also Apply

- **Packets need to "fit" into the available network capacity.** There may be links of different speeds, and with different levels of unutilized network capacity. If the "bottle neck link" has less capacity than you're demanding, traffic may queue or ultimately be dropped. Network capacity can also change dynamically.

- The **receiving system also needs to have buffer space to hold inbound traffic** until it gets taken and processed by the application that's consuming that traffic.

  This can be particularly tricky if there are large mismatches in connectivity. Conceptually, imagine a system that's sending sustained flows at **10 gigabit per second** to a receiving system that's connected at just **100 megabits per second**. The receiving system and the sending system need to work out how fast traffic should be sent if they want to avoid sending traffic faster than the receiving system can ingest it.

- But let's return to our transmission buffers on the sending system.

Exactly How Much Room Do We Need To Hold Traffic In Flight? Let's Compute the Bandwidth Delay Product

- BDP = bandwidth (bits/second) * RTT (seconds)

Gigabit Ethernet:	1,000,000,000 bits/second
10Gig Ethernet:	10,000,000,000 bits/second

Eugene, Oregon to:

iperf.online.net:	163.6 msec or 0.1636 seconds
www.cdu.edu.au:	228.1 msec or 0.2281 seconds

- So, for 2.5 Gbps, Eugene OR to iperf.online.net ➔
  2,500,000,000 * 0.1636 / 8 = 51,125,000 or 51.125 megabytes/buffer.

- And for 2.5 Gbps, Eugene OR to www.cdu.edu.au ➔
  2,500,000,000 * 0.2281 / 8 = 71,281,250 or 71.281 megabytes/buffer.

Sizing Systems for the Required Bandwidth Delay Product

- We now know that **we need buffers big enough to accommodate the bandwidth delay product.** But how do we actually tweak or tune systems to handle "long fat networks?"

- First of all, what's our target sending rate? See https://fasterdata.es.net/host-tuning/linux/

  We could choose to set our buffers to a level that should allow us to fill a 2.5Gbps link with **four parallel flows.** That means that if we're using the iperf3 bandwidth testing tool, we'd specify that we wanted to use four parallel flows with dash capital P 4:

  ```
  $ iperf3 -c iperf.online.net -p 5200-5209 -P 4
  ```

- On the other hand, if you want to realize a full 2.5 Gbps in just a **single stream** to remote hosts, you may need to set your buffers to value that's four times larger.

Macs: Will "Auto-Tune" (But Only Up To A Point)

- Mac OS typically will auto-tune send and receive buffers up to a level that's set in sysctl variables. If you're on a Mac, you can check yours values in a Terminal window with:

```
$ sysctl net.inet.tcp.autorcvbufmax
net.inet.tcp.autorcvbufmax: 4194304

$ sysctl net.inet.tcp.autosndbufmax
net.inet.tcp.autosndbufmax: 4194304
```

- **If you've got a connection that's 2.5 gigabit or faster,** and you potentially want single stream 2.5Gbps throughput, you can try temporarily increasing those limits to 80MB with:

```
$ sudo sysctl -w net.inet.tcp.autorcvbufmax=83886080
$ sudo sysctl -w net.inet.tcp.autosndbufmax=83886080
```

Do NOT tweak these values this high if you're on a slow connection or very busy system.

Macs: Bumping Up the Autotuning Max Values Persistently

You can make those values persistent by adding the following lines to `/etc/sysctl.conf`

```
# Increase maximum receive buffer for TCP autotuning to 80MB
net.inet.tcp.autorcvbufmax=83886080

# Increase maximum send buffer for TCP autotuning to 80MB
net.inet.tcp.autosndbufmax=83886080
```

Note: Theoretically Macs with **Apple's SIP (System Integrity Protection)** enabled (the default) will NOT allow you to modify `/etc/sysctl.conf`, but at least on my test Apple MacBook Pro, changes to `/etc/sysctl.conf` DO appear to be allowed and recognized across reboots. (An easy test: try creating that file and see if the values are as-specified after you reboot)

If SIP does get in the way, see https://unix.stackexchange.com/questions/689295/values-from-sysctl-a-dont-match-etc-sysctl-conf-even-after-restart

Macs: Network Link Conditioner

Sometimes you may want to test how things would perform if latencies were at a particular level, but you don't have time to hunt for a test endpoint with the right latency.

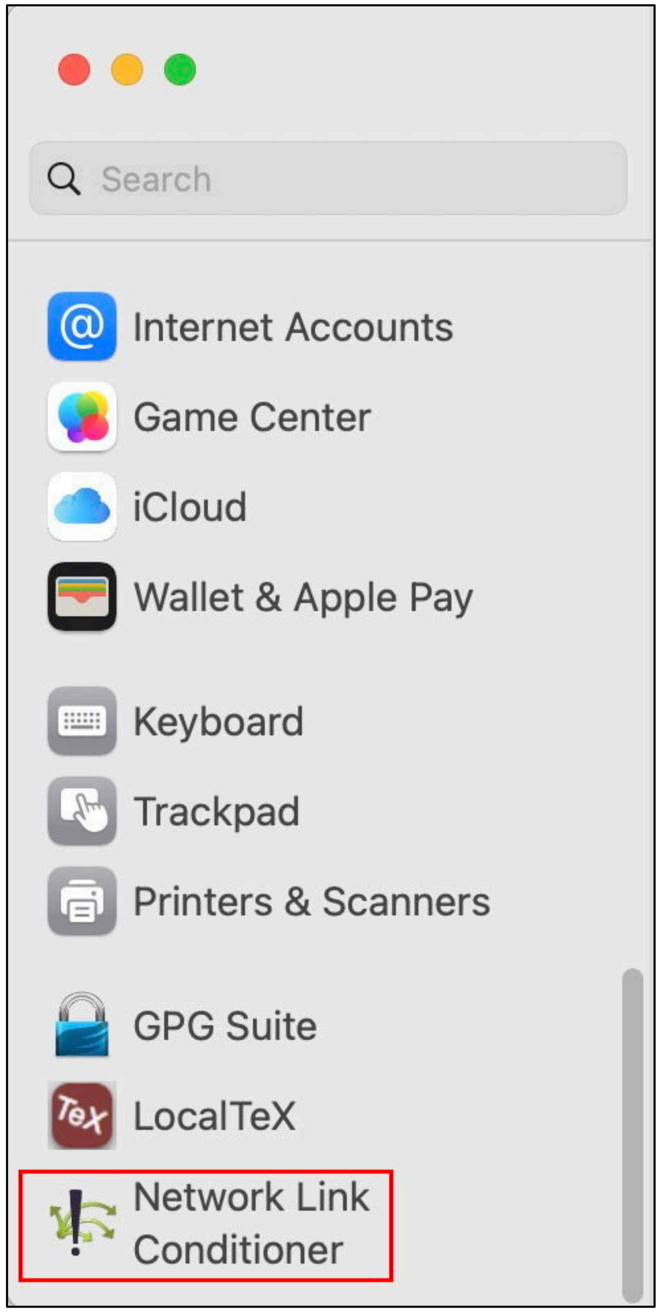
Fortunately, Mac Xcode (the free Apple developer toolkit) offers a supporting utility application called "Network Link Conditioner."

Network Link Conditioner allows you to easily add synthetic network delay (and/or synthetic packet loss) to your actual network connectivity.

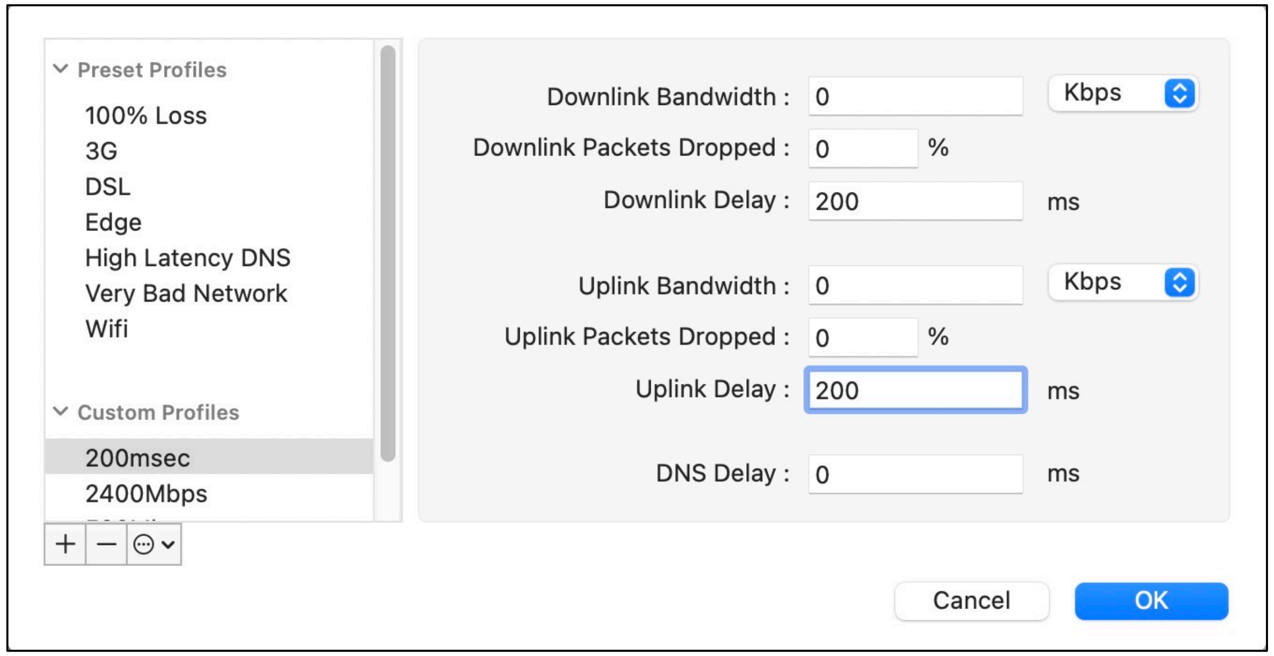
This makes it easy to see the impact of network latency (or packet loss) on your throughput.

See the example on the following slide, showing creation of a profile with 200 msec of extra artificial latency.

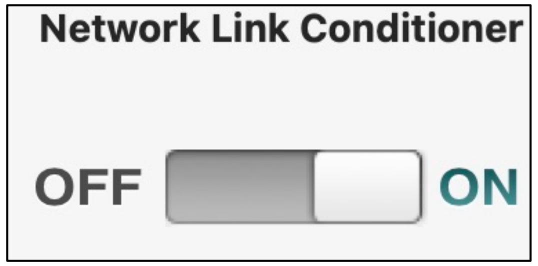
Macs Synthetically Adding Latency with "Network Link Conditioner"



- Install Xcode
- Install Xcode "Additional Tools"
- Run Network Link Condition.prefPane
- Network Link Conditioner will then appear in **System Settings**
- Create a new profile with the desired level of latency
- Activate that profile, and run Speedtest or iperf3 for testing...
- **Don't forget to disable Network Link Conditioner afterwards!**



Downlink Bandwidth :	0	Kbps
Downlink Packets Dropped :	0	%
Downlink Delay :	200	ms
Uplink Bandwidth :	0	Kbps
Uplink Packets Dropped :	0	%
Uplink Delay :	200	ms
DNS Delay :	0	ms



Testing With An Extra 200 msec of Synthetic Latency

```
$ iperf3 -c speedtest.sea11.us.leaseweb.net -p 5201-5210
Connecting to host speedtest.sea11.us.leaseweb.net, port 5201
[  7] local 2606:[snip] port 50393 connected to 2607:[snip] port 5201
[ ID] Interval           Transfer     Bitrate
[  7]   0.00-1.01   sec   128 KBytes  1.04 Mbits/sec
[  7]   1.01-2.00   sec  0.00 Bytes  0.00 bits/sec
[  7]   2.00-3.00   sec   768 KBytes  6.29 Mbits/sec
[  7]   3.00-4.00   sec  1.50 MBytes  12.6 Mbits/sec
[  7]   4.00-5.00   sec  6.12 MBytes  51.3 Mbits/sec
[  7]   5.00-6.00   sec  7.50 MBytes  63.0 Mbits/sec
[  7]   6.00-7.00   sec  5.88 MBytes  49.2 Mbits/sec
[  7]   7.00-8.00   sec  21.5 MBytes   180 Mbits/sec
[  7]   8.00-9.01   sec  16.0 MBytes   134 Mbits/sec
[  7]   9.01-10.01  sec  18.8 MBytes   157 Mbits/sec
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate
[  7]   0.00-10.01  sec  78.1 MBytes  65.5 Mbits/sec                  sender
[  7]   0.00-10.42  sec  78.2 MBytes  62.9 Mbits/sec                  receiver
```

Significantly worse throughput with the extra latency! (Remember to turn it back off!)

Linux: Like Macs, Linux Will Also "Auto-Tune" (Up To A Point)

- Linux hosts will auto-tune send and receive buffers up to a level that's set in sysctl variables. If you're on a Linux host, you can see your current **default settings** with:

  Max TCP buffer size

  ```
  $ sysctl net.core.rmem_max
  net.core.rmem_max = 212992
  $ sysctl net.core.wmem_max
  net.core.wmem_max = 212992
  ```

  Current autotuning TCP buffer limits (the three values are minimum (even under memory pressure), default, and **maximum):**

  ```
  $ sysctl net.ipv4.tcp_rmem
  net.ipv4.tcp_rmem = 4096  131072     6291456
  $ sysctl net.ipv4.tcp_wmem
  net.ipv4.tcp_wmem = 4096  16384      4194304
  ```

Linux: Temporarily Increasing Buffer Limits

- To **temporarily** increase the buffers:

  Adjusting max TCP buffer size

```
$ sudo sysctl net.core.rmem_max=209715200
$ sudo sysctl net.core.wmem_max=209715200
```

  Adjusting autotuning TCP buffer limits

```
$ sudo sysctl net.ipv4.tcp_rmem="4096 87380 83886080"
$ sudo sysctl net.ipv4.tcp_wmem="4096 87380 83886080"
```

Linux: Persistently Increasing the Autotuning Values

Add:

```
net.core.rmem_max=209715200
net.core.wmem_max=209715200
net.ipv4.tcp_rmem=4096 87380 83886080
net.ipv4.tcp_wmem=4096 87380 83886080
```

to `/etc/sysctl.conf`

Linux: tc - Linux's equivalent of the Mac's Network Link Conditioner

Find your interface with ifconfig:

```
# ifconfig -a
```

Then force additional latency to be added to that interface with tc ("traffic control"). Assuming our interface is named ***enp7s0:***

```
# tc qdisc replace dev enp7s0 root netem loss 0% latency 200ms
```

As we did for the Mac Link Conditioner example, you can test with iperf3 to see the impact of the additional latency. To **return to normal minimal latency** (don't forget to do so):

```
# tc qdisc replace dev enp7s0 root netem loss 0% latency 0ms
```

Windows 11: May Be Adequately "Self-Tuned" Out-of-the-box

- Windows 11 does not appear to require additional tuning to self-tune. The stock as shipped configuration appears to be adequate "out of the box"

- For more, see "Performance Tuning Network Adapters," https://learn.microsoft.com/en-us/windows-server/networking/technologies/network-subsystem/net-sub-performance-tuning-nics#bkmk_tcp

  and see

  "MS Windows Tuning"
  https://fasterdata.es.net/host-tuning/ms-windows-2/

  "Set-NetTCPSetting"
  https://learn.microsoft.com/en-us/powershell/module/nettcpip/set-nettcpsetting

Windows 11: Addding Synthetic Network Latency Under Windows

- **Commercial applications** are available to handle adding synthetic network latency for Windows 11. If interested, you should be able to find them pretty easily.

- OR you might also consider trying **clumsy,** see https://jagt.github.io/clumsy/

  Caveats:

  https://jagt.github.io/clumsy/download.html says **"clumsy is alpha quality software** and does have some gotchas. Be sure to read the manual page before using it."

  clumsy builds from source using **zig** (see https://ziglang.org/). I'm not personally acquainted with using zig (nor any considerations that using zig may introduce).

  The documentation at https://jagt.github.io/clumsy/manual.html states that **"Lag is much [...] higher than the parameter set."**

5. Congestion Control Algorithms (CCA): Managing Congestion on LFNs with CUBIC and BBR

Bottom Line Up Front (BLUF):

TCP can't tell in advance how much bandwidth it has available.
TCP uses congestion control algorithms (CCAs) to try to figure
that out and set an approximately optimal sending rate.

Most systems use the CUBIC congestion control algorithm (CCA).
BBR, an alternative CCA, will likely outperform CUBIC, particularly in
the face of network loss. However, as a result, systems using BBR
may end up using more than their "fair share" of network bandwidth.

We explain how to enable BBR on Debian 12 and Windows 11.

The Congestion Window (CWND)

- We've already talked a little about the **TCP transmit buffers** and the **TCP receive buffers.** The transmit buffer holds onto traffic waiting to be acknowledged, while the receive buffer holds onto traffic that's waiting to be consumed by the application. **The sizes of those respective buffers get exchanged by the sender and receiver during the initial three-way handshake and are fixed for the duration of the connection.**

- But now, what about the dynamically-changing available **network capacity** between the sending and receiving host? That gets managed by the **sender** via the **congestion window ("CWND")**. The CWND may (and often will) change during the course of a file transfer or other flow.

- As Wikipedia states, "When a connection is set up, the congestion window, a value maintained independently at each host, is set to a small multiple of the maximum segment size (MSS) allowed on that connection. **Further variance in the congestion window is dictated by an additive increase/multiplicative decrease (AIMD) approach.**" See https://en.wikipedia.org/wiki/Transmission_Control_Protocol [emphasis added]

The Sawtooth Throughput Graph Implied by AIMD

- Whatever specific TCP congestion control algorithm gets deployed, it will usually **additively increase** (and then typically **multiplicatively decrease**) throughput over time.

- Traffic gets sent onto the network in bursts of packets. But how many packets per burst?

- "Additive increase" implies a relatively-gradual "ramp up:" add an extra packet to the burst, then send the burst. Did it get there okay? It did? Okay, add another packet to the next burst. Did that make it okay, too? Add another, etc.

- "Multiplicative decrease" implies a relatively abrupt reaction to detected congestion. If traffic gets dropped, cut the size of the packet bursts dramatically – maybe by 30%, maybe by 50%, maybe by some other amount (the amount of the drop will vary by the congestion control algorithm used).

- One popular congestion control algorithm is called CUBIC.

"CUBIC For Fast and Long-Distance Networks" (RFC 9438)

"CUBIC has been adopted as the default TCP congestion control algorithm in the Linux, Windows, and Apple stacks, and has been used and deployed globally. Extensive, decade-long deployment experience in vastly different Internet scenarios has convincingly demonstrated that CUBIC is **safe for deployment on the global Internet** and delivers substantial benefits over classical Reno congestion control [RFC5681]. It is therefore to be regarded as the currently **most widely deployed standard for TCP congestion control**. [* * *]

"The design of CUBIC was motivated by the well-documented problem classical Reno TCP has with low utilization over fast and long-distance networks [K03] [RFC3649]. This problem arises from a slow increase of the congestion window (cwnd) following a congestion event in a network with a large bandwidth-delay product (BDP). [* * *]

"CUBIC, originally proposed in [HRX08], is a modification to the congestion control algorithm of classical Reno to remedy this problem. Specifically, CUBIC uses a cubic function instead of the linear window increase function of Reno to improve scalability and stability under fast and long-distance networks." [emphasis added]

What Does "Safe for Deployment On the Global Internet" Mean?

- Congestion control algorithms can go catastrophically wrong.

- What could go potentially go wrong? Well, bottom line, you'd REALLY like to AVOID **congestive collapse** (the entire network gets congested, and instead of politely backing off, algorithms might actually try "pushing harder," with the net result being that the network becomes totally unusable for everyone).

  After AVOIDING congestive collapse, "nice to have" congestion control features deliver:

  - Efficiency, with good throughput and minimal delay and loss, and
  - Fairness to other traffic that shares the network

  For more, see "Specifying New Congestion Control Algorithms," https://www.rfc-editor.org/rfc/rfc5033.txt

Aside: Both CUBIC and BBR Leverage ECN (We'll Talk About BBR A Little Later)

- "[Explicit Congestion Notification] allows end-to-end notification of network congestion without dropping packets. ECN is an optional feature that may be used between two ECN-enabled endpoints when the underlying network infrastructure also supports it. Conventionally, TCP/IP networks signal congestion by dropping packets. When ECN is successfully negotiated, an ECN-aware router may set a mark in the IP header instead of dropping a packet in order to signal impending congestion. The receiver of the packet echoes the congestion indication to the sender, which reduces its transmission rate as if it detected a dropped packet."
  https://en.wikipedia.org/wiki/Explicit_Congestion_Notification

- ECN is defined in "The Addition of Explicit Congestion Notification (ECN) to IP," https://www.rfc-editor.org/rfc/rfc3168 and

- "Relaxing Restrictions on Explicit Congestion Notification (ECN) Experimentation," https://www.rfc-editor.org/rfc/rfc8311

If You Want To Ensure ECN is Enabled (ECN will normally default to "On")

- **Windows 11:**

```
netsh interface tcp set global ecncapability=enabled
```

- **Mac:**

```
sudo sysctl -w net.inet.tcp.ecn_negotiate_in=1
sudo sysctl -w net.inet.tcp.ecn_initiate_out=1
```

  To make the settings persistent, put the following lines in `/etc/sysctl.conf`:

```
net.inet.tcp.ecn_negotiate_in=1
net.inet.tcp.ecn_initiate_out=1
```

If You Want To Ensure ECN is Enabled (ECN will normally default to "On") (continued)

- **Debian 12:**

  As root, enable ECN by editing `/etc/sysctl.conf` to include:

  ```
  net.ipv4.tcp_ecn=1
  ```

  Reload that file by running:

  ```
  sysctl -p
  ```

CUBIC's Popular BUT IT Can Have Issues in Loss-Prone Environments

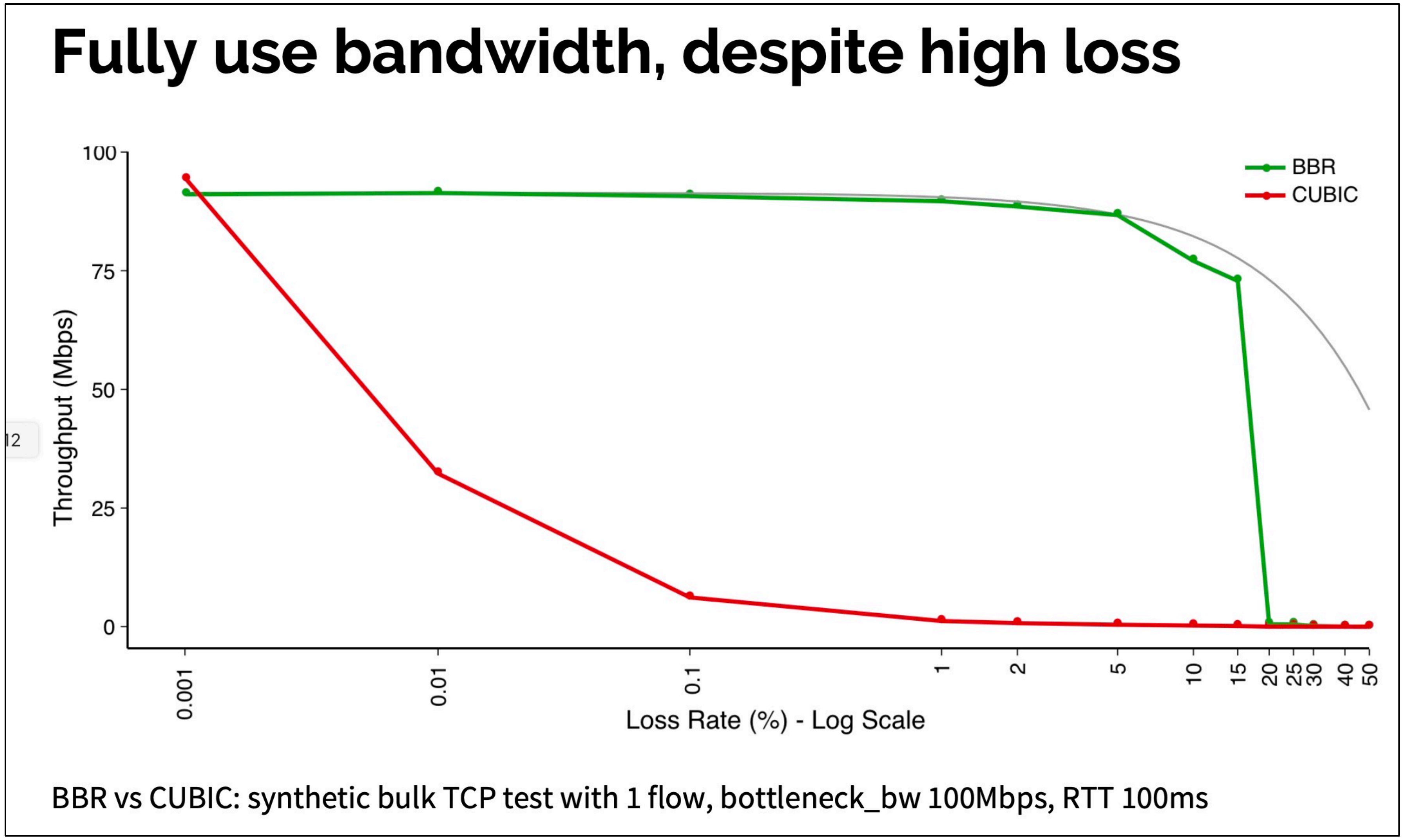
- "In [some common scenarios] it is difficult to achieve full throughput with loss-based congestion control in practice: **for CUBIC, sustaining 10Gbps over 100ms RTT needs a packet loss rate below 0.000003%** (i.e., more than 40 seconds between packet losses), and **over a 100ms RTT path a more feasible loss rate like 1% can only sustain at most 3 Mbps."**

  "BBR Congestion Control,"
  https://www.ietf.org/archive/id/draft-ietf-ccwg-bbr-00.html

- "Our testing shows that BBR can help a lot on certain links. **10X performance improvements are common on some paths. BBR also works much better than CUBIC on network paths where packet loss is due to small switch buffers.** Our testing also shows that for parallel stream tool, BBR works much better with paced streams, so that the streams [are able?] to not step on each other."

  https://fasterdata.es.net/host-tuning/linux/recent-tcp-enhancements/bbr-tcp/

The Case **FOR** BBR In One Graph...

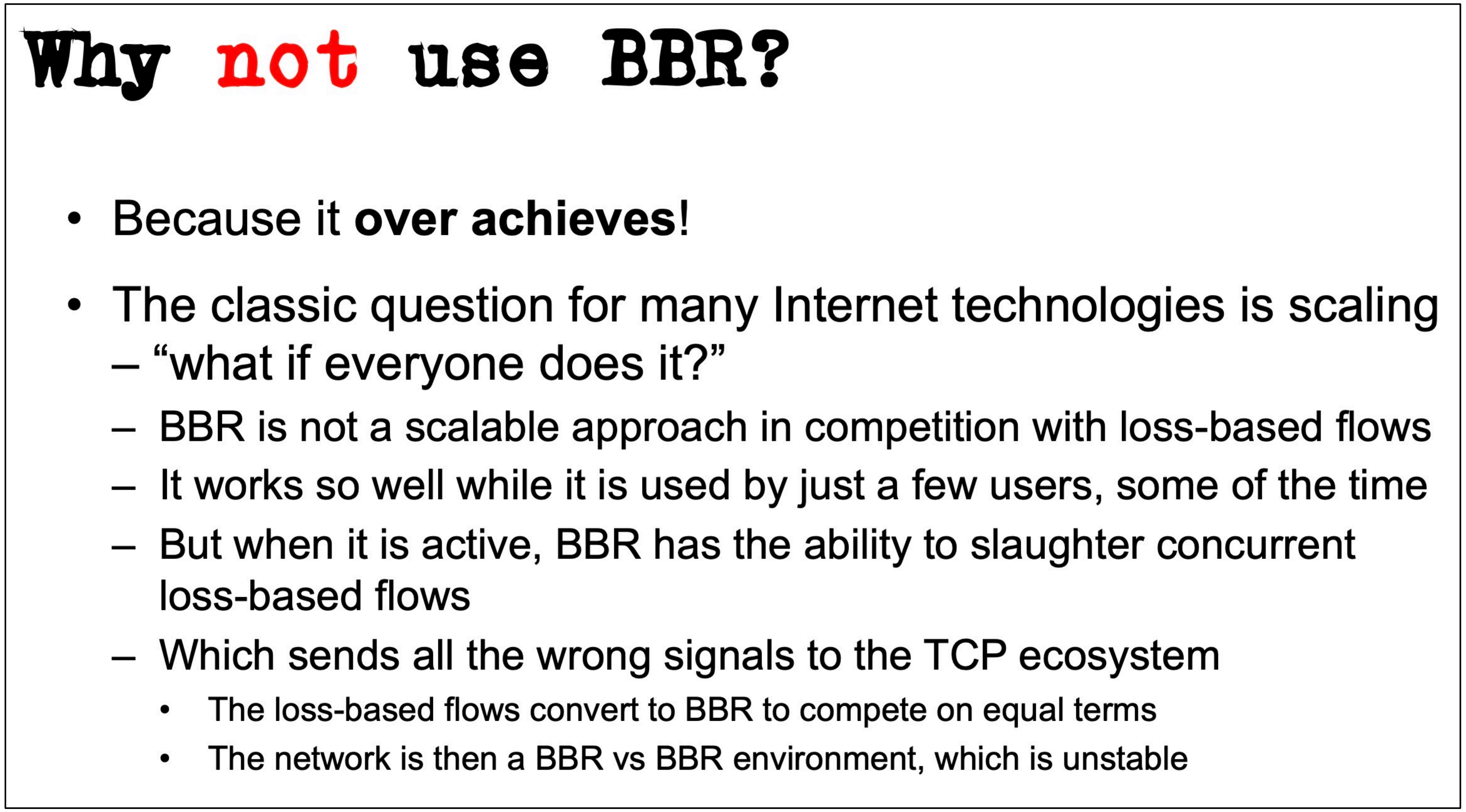


Fully use bandwidth, despite high loss

BBR vs CUBIC: synthetic bulk TCP test with 1 flow, bottleneck_bw 100Mbps, RTT 100ms

"BBR Congestion Control," Neal Cardwell et. al., IETF 97: Seoul, Nov 2016, https://www.ietf.org/proceedings/97/slides/slides-97-iccrg-bbr-congestion-control-02.pdf

The Case <u>AGAINST</u> BBR In One Slide...



Why not use BBR?

- Because it **over achieves**!
- The classic question for many Internet technologies is scaling – "what if everyone does it?"
  - BBR is not a scalable approach in competition with loss-based flows
  - It works so well while it is used by just a few users, some of the time
  - But when it is active, BBR has the ability to slaughter concurrent loss-based flows
  - Which sends all the wrong signals to the TCP ecosystem
    - The loss-based flows convert to BBR to compete on equal terms
    - The network is then a BBR vs BBR environment, which is unstable

"Buffers, Bufferbloat and BBR," Geoff Huston, https://labs.apnic.net/presentations/store/2020-01-31-bbr.pdf

So Why Hasn't BBR Become the New Default (Displacing Cubic)?

- Network engineers are **inherently conservative** and tend to move slowly on stuff that can go really badly. There are an awful lot of different weird environments in which a given congestion control algorithm needs to be able to work, and a new CCA needs to work in ALL (or almost all) of them. It may take a decade to gradually change from one production congestion control algorithm to another. Other factors include...

- BBR may not "play nicely" with Cubic and other congestion management algorithms, taking a disproportionate share of bandwidth on bottleneck connections (in fact, one commentator described BBR as a **"sociopathic protocol"**).

- BBR isn't readily available on all platforms (for example, you can't run **bbr** on Apple Macs)

- But let's assume that you've decided that you DO actually want to try using BBR on Linux or Windows 11. How would you go about doing that?

Running BBR if you're on Debian 12

As root, edit `/etc/sysctl.conf`

Add the following two lines to the bottom of that file:

```
net.core.default_qdisc = fq
net.ipv4.tcp_congestion_control=bbr
```

Re-load `/etc/sysctl.conf` by saying:

```
sysctl -p
```

Finally, confirm that you're running bbr:

```
sysctl net.ipv4.tcp_congestion_control
net.ipv4.tcp_congestion_control = bbr
```

Reverting to the default (CUBIC) congestion control on Debian 12

Comment out the two lines you added to `/etc/sysctl.conf` and reboot.

Confirm that you're running CUBIC once more:

```
sysctl net.ipv4.tcp_congestion_control
net.ipv4.tcp_congestion_control = cubic
```

Using BBR2 on Windows 11 via Powershell

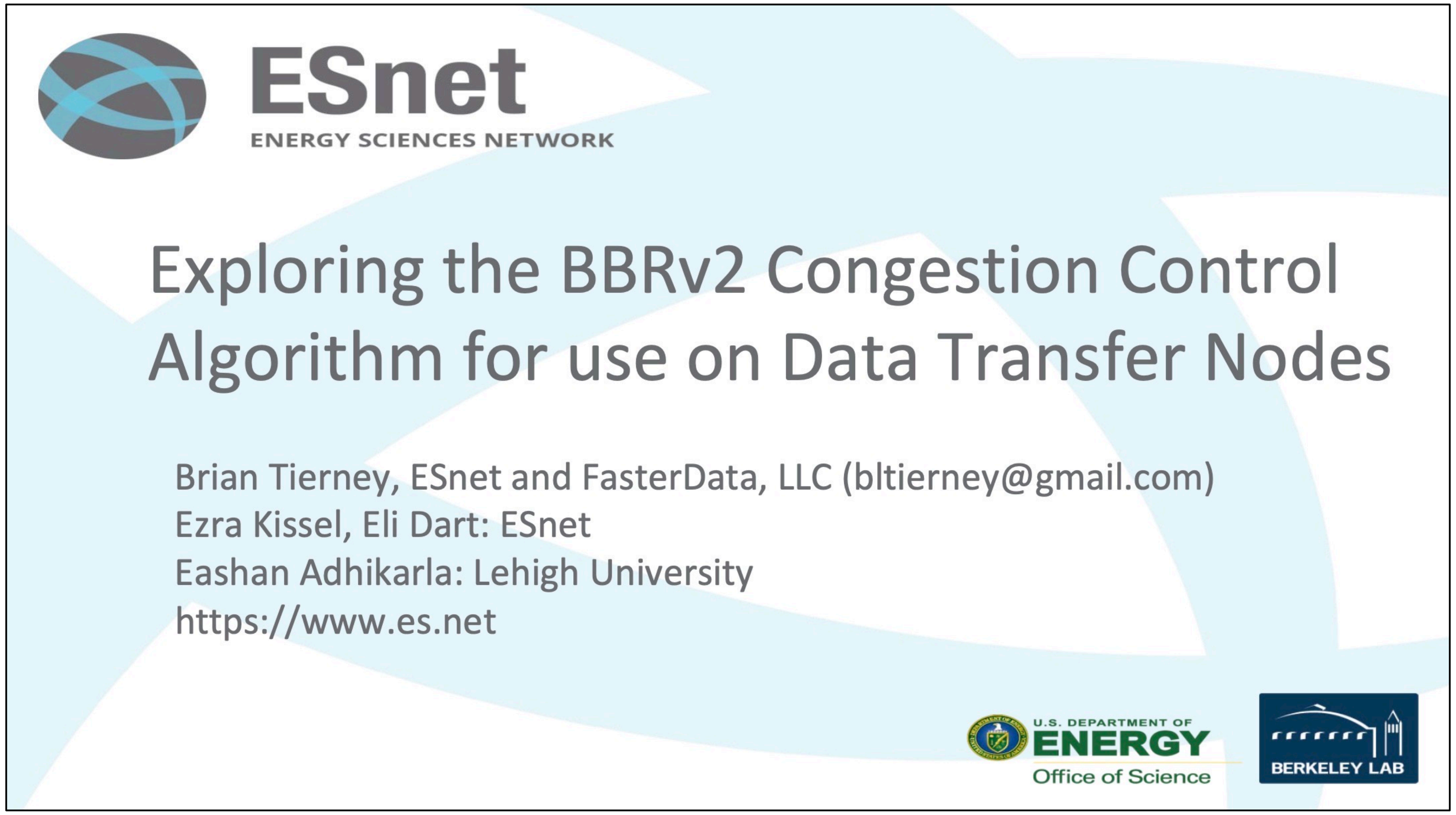
```
netsh int tcp set supplemental Template=Internet CongestionProvider=bbr2
netsh int tcp set supplemental Template=Datacenter CongestionProvider=bbr2
netsh int tcp set supplemental Template=Compat CongestionProvider=bbr2
netsh int tcp set supplemental Template=DatacenterCustom CongestionProvider=bbr2
netsh int tcp set supplemental Template=InternetCustom CongestionProvider=bbr2
```

If you need or want to revert to CUBIC:

```
netsh int tcp set supplemental Template=Internet CongestionProvider=CUBIC
netsh int tcp set supplemental Template=InternetCustom CongestionProvider=CUBIC
netsh int tcp set supplemental Template=Compat CongestionProvider=NewReno
netsh int tcp set supplemental Template=Datacenter CongestionProvider=CUBIC
netsh int tcp set supplemental Template=DatacenterCustom CongestionProvider=CUBIC
```

Nice Analysis of Using BBR on Ultrafast Hosts from ESnet

"Exploring the BBRv2 Congestion Control Algorithm for use on Data Transfer Nodes"
https://internet2.edu/wp-content/uploads/2022/12/techex22-AdvancedNetworking-ExploringtheBBRv2CongestionControlAlgorithm-Tierney.pdf


ESnet
ENERGY SCIENCES NETWORK

Exploring the BBRv2 Congestion Control Algorithm for use on Data Transfer Nodes

Brian Tierney, ESnet and FasterData, LLC (bltierney@gmail.com)
Ezra Kissel, Eli Dart: ESnet
Eashan Adhikarla: Lehigh University
https://www.es.net

U.S. DEPARTMENT OF ENERGY
Office of Science
BERKELEY LAB

Aside: "What About *Satellite* Latencies?"

- We've been implicitly considering latency over terrestrial circuits, but some connections may actually be delivered over **satellite**. Not all satellites are the same – some satellites may be in higher or lower orbits.
- **Geostationary satellites** orbit at a high altitude (35,786 km). As a result, Internet connections delivered over **geostationary satellite tend to have latencies of 600 msec or more** – that's a LOT of latency by traditional terrestrial standards. That can be tricky to accommodate. One comparative review: **"Competing TCP Congestion Control Algorithms over a Satellite Network,"** https://pinhanzhao.com/papers/ccnc22.pdf
- **Low earth orbit satellites,** on the other hand, orbit at just 250-1000 km. They deliver latencies that tend to feel just like terrestrial fiber latencies – 25 to 100 msec (see, e.g., https://www.starlink.com/legal/documents/DOC-1470-99699-90?regionCode=US). They have a different issue, namely **frequent handoffs from one satellite to another (connections to a particular satellite may last from just some seconds to 3+ minutes).** A comparative review of CCAs for the LEO environment is at **"A Comparative Evaluation of TCP Congestion Control Schemes over Low-Earth-Orbit (LEO) Satellite Networks,"** https://dl.acm.org/doi/fullHtml/10.1145/3630590.3630603

6. RFC 1323 Window Scaling

Bottom Line Up Front (BLUF):

High performance networking requires large buffers. RFC 1323 Windows Scaling makes large buffers possible, even when the window size field would normally limit the buffer size to just 64KB.

The Congestion Window (CWND) effectively controls the amount of bandwidth the sender attempts to use during the course of a transfer. You'll see it reported during bandwidth testing with iperf3.

Returning To The TCP Congestion Window (CWND)

To avoid congestive collapse, TCP uses a multi-faceted congestion-control strategy. For each connection, TCP maintains a CWND, limiting the total number of unacknowledged packets that may be in transit end-to-end. [* *]*

The congestion window is maintained by the sender and is a means of preventing a link between the sender and the receiver from becoming overloaded with too much traffic. [* *]*

When a connection is set up, the congestion window, a value maintained independently at each host, is set to a small multiple of the maximum segment size (MSS) allowed on that connection. Further variance in the congestion window is dictated by an additive increase/multiplicative decrease (AIMD) approach. This means that if all segments are received and the acknowledgments reach the sender on time, some constant is added to the window size. It will follow different algorithms. ***A system administrator may adjust the*** *maximum window size limit, or* ***adjust the constant added during additive increase, as part of TCP tuning.***

https://en.wikipedia.org/wiki/TCP_congestion_control [emphasis added]

Maximum CWND and RFC 1323 Windows Scaling

So what's the *MAXIMUM* CWND? The CWND field is part of the TCP headers. That field is only **16 bits long. That means the maximum unscaled value it can store is just 64KB (65,535 bytes).** That's far too small for modern "long fat" networks. That's why the **RFC 1323 Window Scaling** option exists and is important.

> *"Since the [TCP window] size field can't be expanded, a scaling factor is used.* ***TCP window scale is an option used to increase the maximum window size from 65,535 bytes to 1 Gigabyte.***
>
> ***The window scale option is used only during the TCP three-way handshake.*** *The window scale value represents the number of bits to left-shift the 16-bit window size field. The window scale value can be set from 0 (no shift) to 14.*
>
> ***To calculate the true window size, multiply the window size by 2^S where S is the scale value."***
>
> https://learn.microsoft.com/en-us/troubleshoot/windows-server/networking/description-tcp-features

Window Scaling Factor Table (Assuming Initial Window = 64KB)

64KB * 2^**N**

N=0 ➔ 64KB
N=1 ➔ 128KB
N=2 ➔ 256KB
N=3 ➔ 512KB
N=4 ➔ 1MB
[...]
N=8 ➔ 16MB
N=9 ➔ 32MB
N=10 ➔ 64MB
N=11 ➔ 128MB
N=12 ➔ 256MB
N=13 ➔ 512MB
N=14 ➔ 1GB

You'll See CWND Values Reported As Part of The iperf3 output

```
iperf3 -c speedtest.sea11.us.leaseweb.net -p 5201-5210
Connecting to host speedtest.sea11.us.leaseweb.net, port 5201
[  5] local 2606:[snip] port 60366 connected to 2607:[snip] port 5201
[ ID] Interval           Transfer     Bitrate         Retr  Cwnd
[  5]   0.00-1.00   sec   289 MBytes  2.42 Gbits/sec   10   8.24 MBytes
[  5]   1.00-2.00   sec   278 MBytes  2.33 Gbits/sec    0   8.26 MBytes
[  5]   2.00-3.00   sec   276 MBytes  2.32 Gbits/sec    0   8.23 MBytes
[  5]   3.00-4.00   sec   276 MBytes  2.32 Gbits/sec    0   8.24 MBytes
[  5]   4.00-5.00   sec   278 MBytes  2.33 Gbits/sec    0   8.25 MBytes
[  5]   5.00-6.00   sec   276 MBytes  2.32 Gbits/sec    0   8.26 MBytes
[  5]   6.00-7.00   sec   278 MBytes  2.33 Gbits/sec    0   8.26 MBytes
[  5]   7.00-8.00   sec   276 MBytes  2.32 Gbits/sec    0   8.24 MBytes
[  5]   8.00-9.00   sec   276 MBytes  2.32 Gbits/sec    0   8.29 MBytes
[  5]   9.00-10.00  sec   278 MBytes  2.33 Gbits/sec    0   8.26 MBytes
[snip]
```

Note that you will NOT see CWND data if you're using the `-R` flag (so the remote system is doing the sending) because **ONLY** the sender knows the current CWND data value.

Default Window Scaling Factors May Have Too-Low Starting Values

Checking Mac OS Sequoia 15.4.1:

```
sysctl net.inet.tcp.win_scale_factor
net.inet.tcp.win_scale_factor: 3
```

If on a 2.5 Gbps link, you may want to increase the Window Scaling Factor to 11 or 12:

Modify `/etc/sysctl.conf` by appending the line:

```
net.inet.tcp.win_scale_factor=11
```

Reload that file (as root) with:

```
sysctl -f /etc/sysctl.conf
```

Leave ALL RFC 1323 Options Alone ("Enabled" Is Normally The Default)

At this point, all RFC 1323 options (Windows Scaling, Timestamps and SACK) should come pre-enabled on virtually all system.

Leave them alone (e.g., keep them enabled).

Even though Timestamps and SACK may cause some packet header overhead, their benefits FAR outweigh their costs.

7. "What About Bufferbloat?"

Bottom Line Up Front (BLUF):

Large single queue buffers mean that jitter-sensitive packets may end up blocked by other bulk transfers. This is normally referred to as "bufferbloat." Systems that use active queue management, or BBR for congestion control, normally will not be impacted by bufferbloat.

You can empirically test to see if your systems are vulnerable to this issue.

What Is Bufferbloat?

When someone on your network sends a large file, a lot of packets get sent all at once. The router temporarily "buffers those packets", holding them before they're sent. Any new data packets get stuck behind the existing queue of buffered packets. They will arrive at the destination much later than if the router's buffers hadn't been full.

This is "bufferbloat" - undesirable high latency caused by other traffic on your network. It happens when a flow uses more than its fair share of the bottleneck. Bufferbloat is the primary cause of bad performance for real-time Internet applications like VoIP calls, video games, and videoconferencing.

Source: `https://www.waveform.com/tools/bufferbloat`

We've NOT Been Working to "Avoid Jitter and Lag"

OUR tuning objective has been to **use our full available bandwidth.** We've NOT been working to avoid jitter and lag. **Is bufferbloat even really an issue we'll run into?**

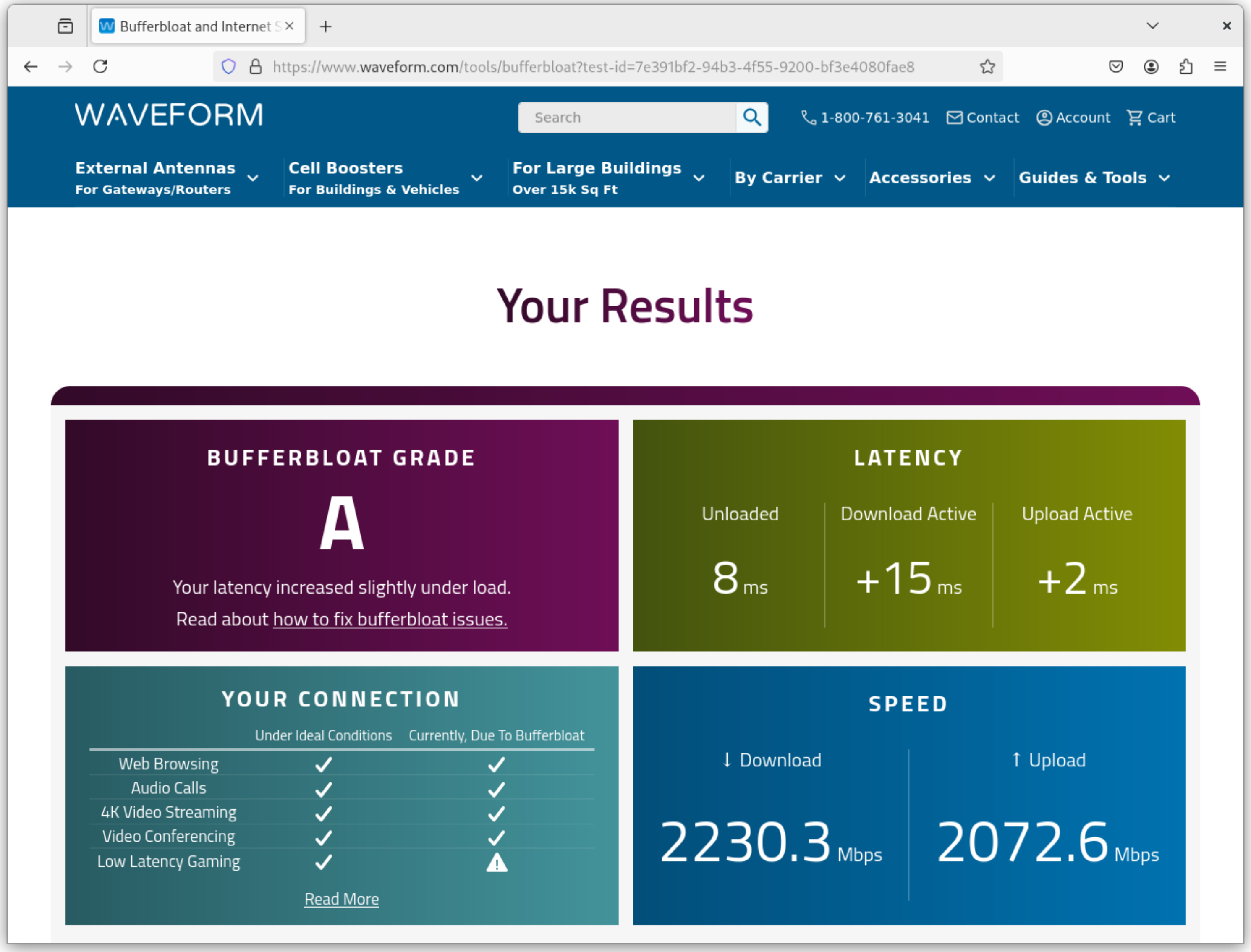
If your system is using `FQ_codel` `("Fair Queuing" Coddle)`, bufferbloat will likely NOT be an issue for you.

If you're using BBR instead of CUBIC for congestion control, bufferbloat may also likely NOT be an issue for you.

But you can test and empirically see! One easy-to-use web page is at `https://www.waveform.com/tools/bufferbloat` (sample output on the next slide)

Another cool tool to use to test for buffer bloat is `flent`, see: `https://flent.org/`

Sample https://www.waveform.com/tools/bufferbloat Results


Bufferbloat and Internet S
https://www.waveform.com/tools/bufferbloat?test-id=7e391bf2-94b3-4f55-9200-bf3e4080fae8
WAVEFORM
Search
1-800-761-3041
Contact
Account
Cart
External Antennas For Gateways/Routers
Cell Boosters For Buildings & Vehicles
For Large Buildings Over 15k Sq Ft
By Carrier
Accessories
Guides & Tools

Your Results

BUFFERBLOAT GRADE

A

Your latency increased slightly under load.
Read about how to fix bufferbloat issues.

LATENCY

Unloaded	Download Active	Upload Active
8 ms	+15 ms	+2 ms

YOUR CONNECTION

	Under Ideal Conditions	Currently, Due To Bufferbloat
Web Browsing	✓	✓
Audio Calls	✓	✓
4K Video Streaming	✓	✓
Video Conferencing	✓	✓
Low Latency Gaming	✓	⚠

Read More

SPEED

↓ Download	↑ Upload
2230.3 Mbps	2072.6 Mbps

Preventing Bufferbloat Symptoms with `FQ_codel`

Modern versions of Linux will use `FQ_codel` by default. That should limit or completely eliminate bufferbloat symptoms (at least on the host side of things).

For more on `FQ_codel`, check out:

"Controlled Delay Active Queue Management"
`https://www.rfc-editor.org/rfc/rfc8289.html`

"The Flow Queue CoDel Packet Scheduler and Active Queue Management Algorithm"
`https://www.rfc-editor.org/rfc/rfc8290.html`

The Codel Wiki:
`https://www.bufferbloat.net/projects/codel/wiki/`

The status of active queue management with FQ_codel or the equivalent is less clear on MacOS and Windows 11. Anyone able to speak authoritatively on this? (Using BBR, as is possible on Window 11, will also fix bufferbloat)

8. Using tcpdump and Wireshark To Begin To Dig Into Network Traffic

Bottom Line Up Fron (BLUF):

If you're going to do performance-related work, you need suitable tools to help you see what's going on. tcpdump and Wireshark are virtually ideal tools for this.

wireshark can seem daunting to get started with, but we'll show you how to bring in a PCAP and get going. You'll be analyzing traffic, producing summary statistics and graphs in no time at all.

Why Go Over tcpdump and Wireshark today?

- Much of what drives network performance happens "under the surface" in network traffic where it can't be "seen by the naked eye."
- Seeing what's going on in network traffic requires **instrumentation,** just as an astronomer needs a telescope or a microbiologist needs a microscope. One basic tool for network performance is a **protocol analyzer.** We'll use **Wireshark,** since it's the most popular option (and it's free/open source). See https://www.wireshark.org/
- Wireshark needs traffic to analyze. While Wireshark has integrated capture capabilities, I prefer to decouple packet capture from the actual analysis of the captured traffic. For that reason, we'll use tcpdump, see https://www.tcpdump.org/
- We won't have time to do a deep dive on either product, but hopefully we can bootstrap you far enough fast enough that you'll at least be able to get rolling. Once we've got you "hooked" you can dig deeper on your own (there are lots of training materials online)

Identifying Your Active Network Interface Under Linux

Wireshark requires a packet capture. When you're **NOT AT M3AAWG** you can easily grab one using tcpdump. **Remember, do NOT do ANY packet captures while at M3AAWG!**

The first step is figuring out your active network interface. Your active network interface has various indicators, including: it will have a "RUNNING" flag, it will show up with your system's IP address(es), and it will show traffic having been received and transmitted in the packet counters. For example:

```
# ifconfig -a
enp7s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.[snip]  netmask 255.255.255.0  broadcast 192.[snip]
        inet6 2606:[snip]  prefixlen 64  scopeid 0x0<global>
        [* * *]
        RX packets 35726607  bytes 48263675457 (44.9 GiB)
        RX errors 0  dropped 225  overruns 0  frame 0
        TX packets 65284248  bytes 85450529828 (79.5 GiB)
        [* * *]
```

Capturing Some Network Traffic Using tcpdump on Linux

If you don't already have tcpdump installed, install it via your package manager:

```
apt install tcpdump
```

After tcpdump is installed, start capturing (as root) with:

```
tcpdump -i enp7s0 -n -w sampletrace.pcap
tcpdump: listening on enp7s0, link-type EN10MB
    (Ethernet), snapshot length 262144 bytes
```

In a second window, run an application to generate some network traffic to capture. We'll limit this to five seconds to keep the capture to semi-reasonable size:

```
iperf3 -c speedtest.sea11.us.leaseweb.net -p 5201-5210 -t 5
```

After our iperf3 run finishes, go to the tcpdump window and stop capturing packets by hitting control-C. That will stop the capture and finish writing out our captured data.

Our sample pcap file

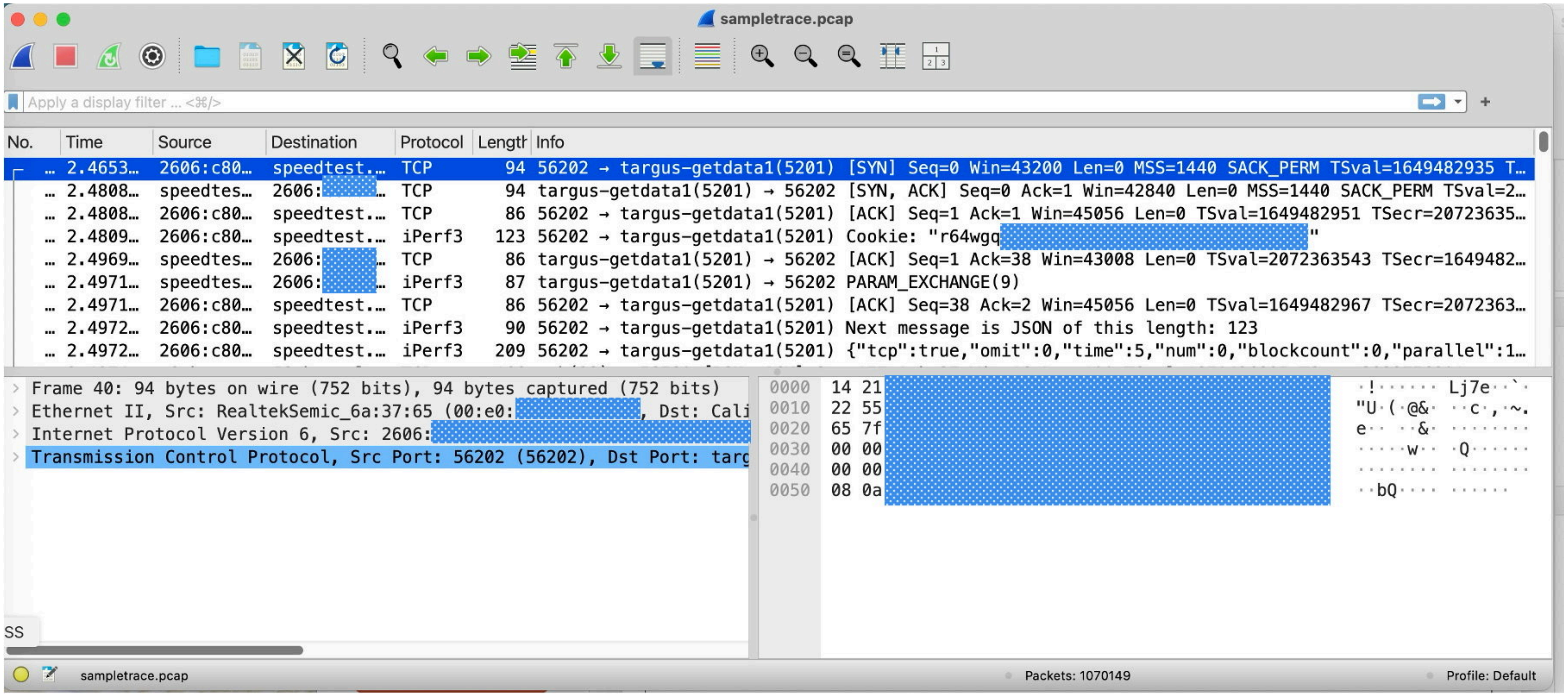
```
# ls -lh sampletrace.pcap
-rw-r--r-- 1 tcpdump tcpdump 1.5G May 14 10:31 sampletrace.pcap
```

When opening or looking at that file in Wireshark on a Mac, be patient – a gig and a half capture file is relatively large, and may take a minute to open in Wireshark (you can watch the progress bar on the bottom of the frame).

You will run into similar "opportunities for patience" while analyzing that file (e.g., some filters require the entire dataset to be re-read in order to apply the specified filter).

Watch that progress bar!

A (Somewhat Daunting) Opening Wireshark Window for a PCAP



```
No.   Time      Source      Destination   Protocol  Length  Info
…  2.4653…  2606:c80…  speedtest.…  TCP      94  56202 → targus-getdata1(5201) [SYN] Seq=0 Win=43200 Len=0 MSS=1440 SACK_PERM TSval=1649482935 T…
…  2.4808…  speedtes…  2606:[redacted]…  TCP      94  targus-getdata1(5201) → 56202 [SYN, ACK] Seq=0 Ack=1 Win=42840 Len=0 MSS=1440 SACK_PERM TSval=2…
…  2.4808…  2606:c80…  speedtest.…  TCP      86  56202 → targus-getdata1(5201) [ACK] Seq=1 Ack=1 Win=45056 Len=0 TSval=1649482951 TSecr=20723635…
…  2.4809…  2606:c80…  speedtest.…  iPerf3  123  56202 → targus-getdata1(5201) Cookie: "r64wgq[redacted]"
…  2.4969…  speedtes…  2606:[redacted]…  TCP      86  targus-getdata1(5201) → 56202 [ACK] Seq=1 Ack=38 Win=43008 Len=0 TSval=2072363543 TSecr=1649482…
…  2.4971…  speedtes…  2606:[redacted]…  iPerf3   87  targus-getdata1(5201) → 56202 PARAM_EXCHANGE(9)
…  2.4971…  2606:c80…  speedtest.…  TCP      86  56202 → targus-getdata1(5201) [ACK] Seq=38 Ack=2 Win=45056 Len=0 TSval=1649482967 TSecr=2072363…
…  2.4972…  2606:c80…  speedtest.…  iPerf3   90  56202 → targus-getdata1(5201) Next message is JSON of this length: 123
…  2.4972…  2606:c80…  speedtest.…  iPerf3  209  56202 → targus-getdata1(5201) {"tcp":true,"omit":0,"time":5,"num":0,"blockcount":0,"parallel":1…
```

```
> Frame 40: 94 bytes on wire (752 bits), 94 bytes captured (752 bits)
> Ethernet II, Src: RealtekSemic_6a:37:65 (00:e0:[redacted], Dst: Cali
> Internet Protocol Version 6, Src: 2606:[redacted]
> Transmission Control Protocol, Src Port: 56202 (56202), Dst Port: targ
```

```
0000  14 21 [redacted]   ·!······ Lj7e··`·
0010  22 55              "U·(·@&· ··c·,·~.
0020  65 7f              e··  ··&· ········
0030  00 00              ·····w·· ·Q······
0040  00 00              ········ ········
0050  08 0a              ··bQ···· ······
```

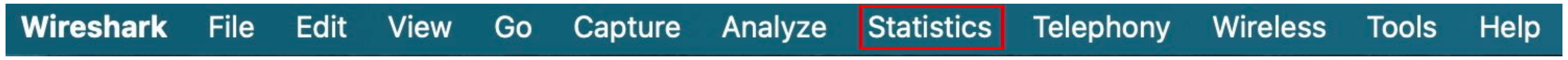
Do NOT let this scary looking opening screen frighten you!

Once you've got a capture opened in Wireshark...

- You can easily work with it. **The biggest impediment to using Wireshark is all the irrelevant stuff that's included in Wireshark (meant to help people working on really weird traffic do so).**

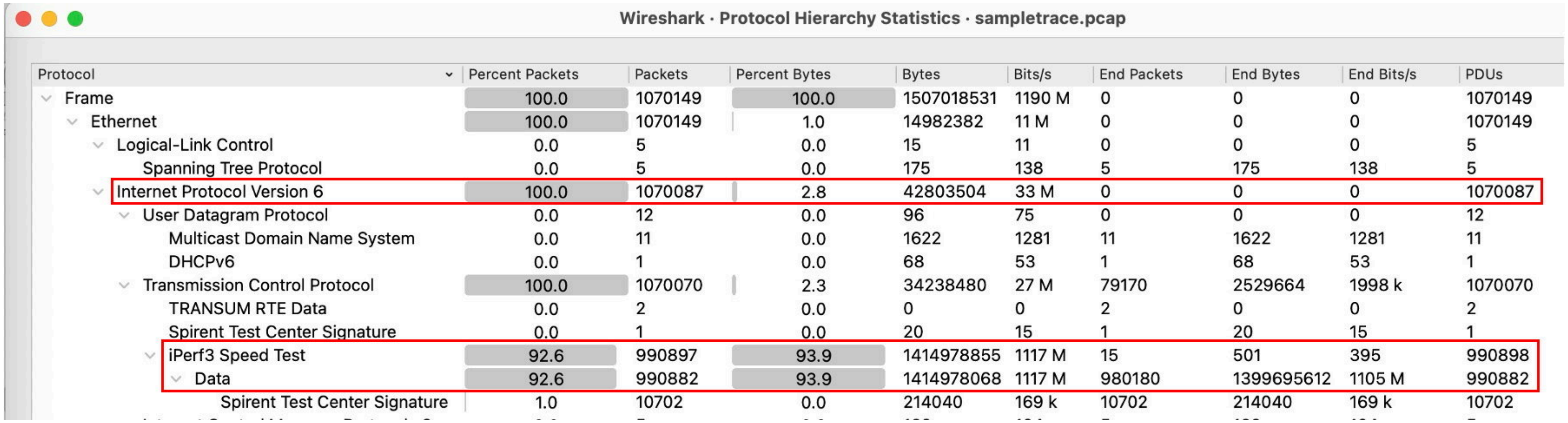
- Much of Wireshark is configurable to your preferences. For example, if you don't find yourself using the packet dump display (shown on the bottom right side of the screen on the preceding slide), you can just drag the middle slider over to the right-hand side and give that screen "real estate" back to the protocol display that's shown on the left-hand side. Similarly, if you want more room to look at particular packet details (bottom of the screen), or more room to pick packets from the capture (top of the screen), you can move the slider between the top and bottom regions to match your preferences.

- Wireshark will also popup **new windows** when appropriate. For example, in the Wireshark menu, try going to **Statistics → Protocol Summary**



Statistics → Protocol Summary (Opens In a New Window)

Hey, it's our iperf3 Speedtest over IPv6! Looks like we captured traffic from it successfully!

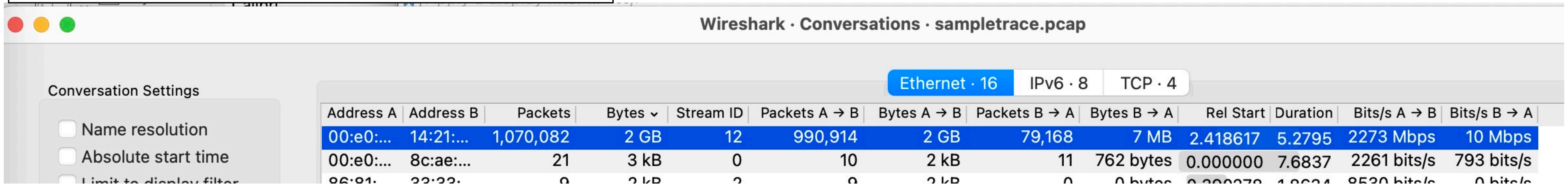


Wireshark · Protocol Hierarchy Statistics · sampletrace.pcap

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s	PDUs
Frame	100.0	1070149	100.0	1507018531	1190 M	0	0	0	1070149
Ethernet	100.0	1070149	1.0	14982382	11 M	0	0	0	1070149
Logical-Link Control	0.0	5	0.0	15	11	0	0	0	5
Spanning Tree Protocol	0.0	5	0.0	175	138	5	175	138	5
Internet Protocol Version 6	100.0	1070087	2.8	42803504	33 M	0	0	0	1070087
User Datagram Protocol	0.0	12	0.0	96	75	0	0	0	12
Multicast Domain Name System	0.0	11	0.0	1622	1281	11	1622	1281	11
DHCPv6	0.0	1	0.0	68	53	1	68	53	1
Transmission Control Protocol	100.0	1070070	2.3	34238480	27 M	79170	2529664	1998 k	1070070
TRANSUM RTE Data	0.0	2	0.0	0	0	2	0	0	2
Spirent Test Center Signature	0.0	1	0.0	20	15	1	20	15	1
iPerf3 Speed Test	92.6	990897	93.9	1414978855	1117 M	15	501	395	990898
Data	92.6	990882	93.9	1414978068	1117 M	980180	1399695612	1105 M	990882
Spirent Test Center Signature	1.0	10702	0.0	214040	169 k	10702	214040	169 k	10702

We can get stats about our flows from Statistics → Conversations

Layer 2 (Note the Ethernet MAC Addresses)


Wireshark · Conversations · sampletrace.pcap

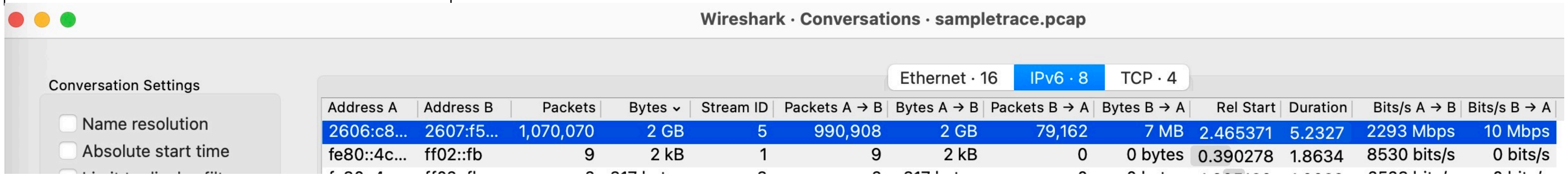
Conversation Settings

- [ ] Name resolution
- [ ] Absolute start time

Ethernet · 16 | IPv6 · 8 | TCP · 4

Address A	Address B	Packets	Bytes ⌄	Stream ID	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
00:e0:...	14:21:...	1,070,082	2 GB	12	990,914	2 GB	79,168	7 MB	2.418617	5.2795	2273 Mbps	10 Mbps
00:e0:...	8c:ae:...	21	3 kB	0	10	2 kB	11	762 bytes	0.000000	7.6837	2261 bits/s	793 bits/s

Layer 3 (Note the IP Addresses)


Wireshark · Conversations · sampletrace.pcap

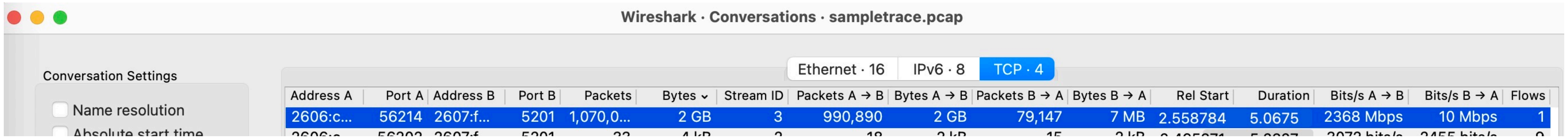
Conversation Settings

- [ ] Name resolution
- [ ] Absolute start time

Ethernet · 16 | IPv6 · 8 | TCP · 4

Address A	Address B	Packets	Bytes ⌄	Stream ID	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
2606:c8...	2607:f5...	1,070,070	2 GB	5	990,908	2 GB	79,162	7 MB	2.465371	5.2327	2293 Mbps	10 Mbps
fe80::4c...	ff02::fb	9	2 kB	1	9	2 kB	0	0 bytes	0.390278	1.8634	8530 bits/s	0 bits/s

TCP (Note the Port Numbers)


Wireshark · Conversations · sampletrace.pcap

Conversation Settings

- [ ] Name resolution

Ethernet · 16 | IPv6 · 8 | TCP · 4

Address A	Port A	Address B	Port B	Packets	Bytes ⌄	Stream ID	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A	Flows
2606:c...	56214	2607:f...	5201	1,070,0...	2 GB	3	990,890	2 GB	79,147	7 MB	2.558784	5.0675	2368 Mbps	10 Mbps	1

Also note the "stream ID" field for each of those. This helps us ID just the data we want.

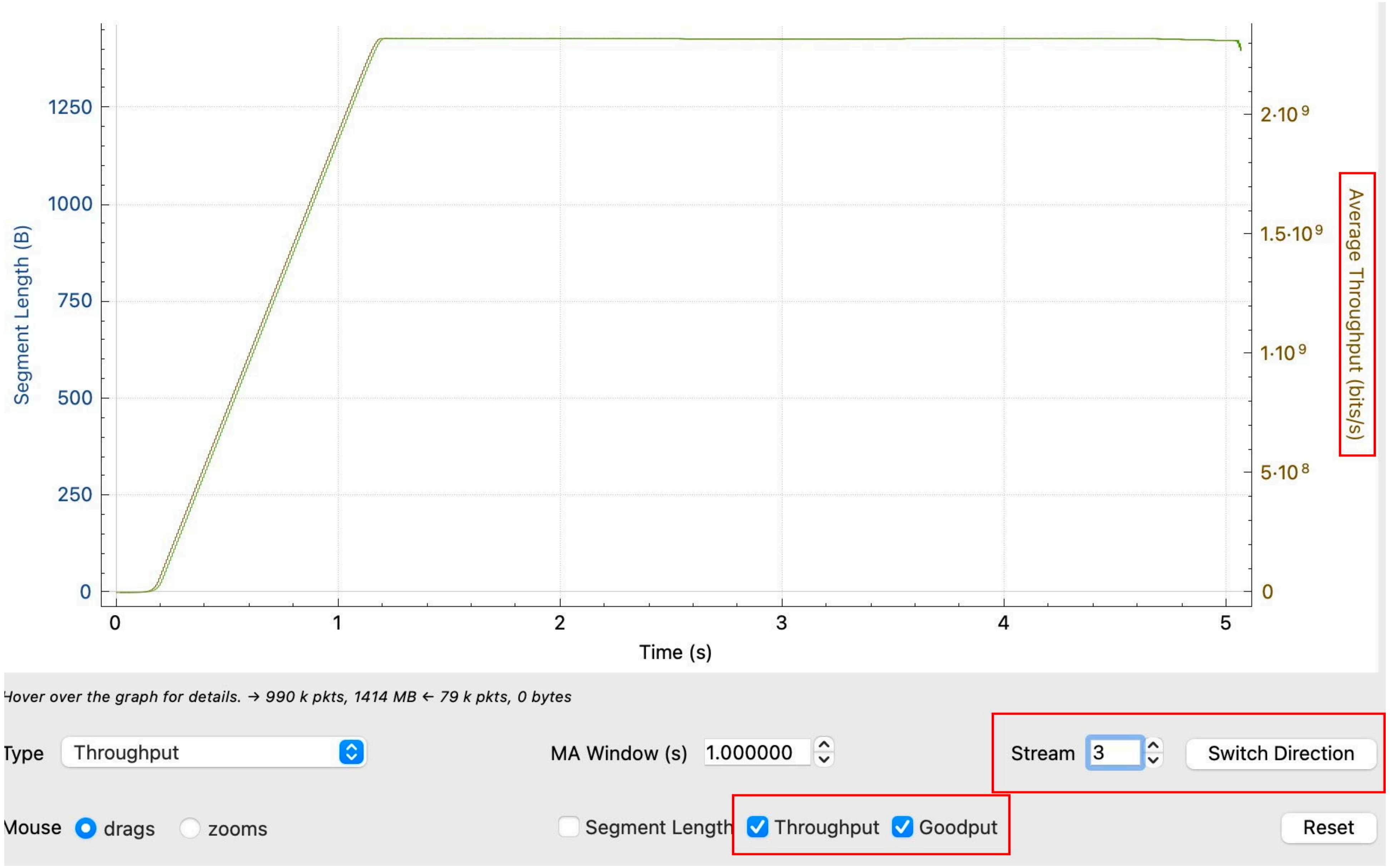
Let's Look at a Traffic-Specific Plot

- For example, since this is a performance session, what does the throughput (and goodput) look like for our iperf3 run over time? We can easily get a "big picture" graph once we know a few basic skills...

- Select **Statistics → TCP Stream Graphs → Time Sequence** from the Wireshark menus
- Select **Throughput** from the left-hand side pull-down menu
- Select **Throughput** (and/or **Goodput)** from the bottom middle what-to-display buttons
- Pick the **right stream** to look at (hint: we showed you how to identify streams in the previous slide -- we're interested in the TCP Stream number with the big packet counts)
- Ensure you're looking "in the right direction" (outbound, not inbound, in this case). You can flip the direction of the flows with the **Switch Direction** button.

The result should be a display something like what you see on the next slide...

Statistics→TCP Stream Graphs→Time Sequence (Throughput, No Zoom)



Hover over the graph for details. → 990 k pkts, 1414 MB ← 79 k pkts, 0 bytes

Type: Throughput

MA Window (s): 1.000000

Stream: 3

Switch Direction

Mouse: [✓ drags] zooms

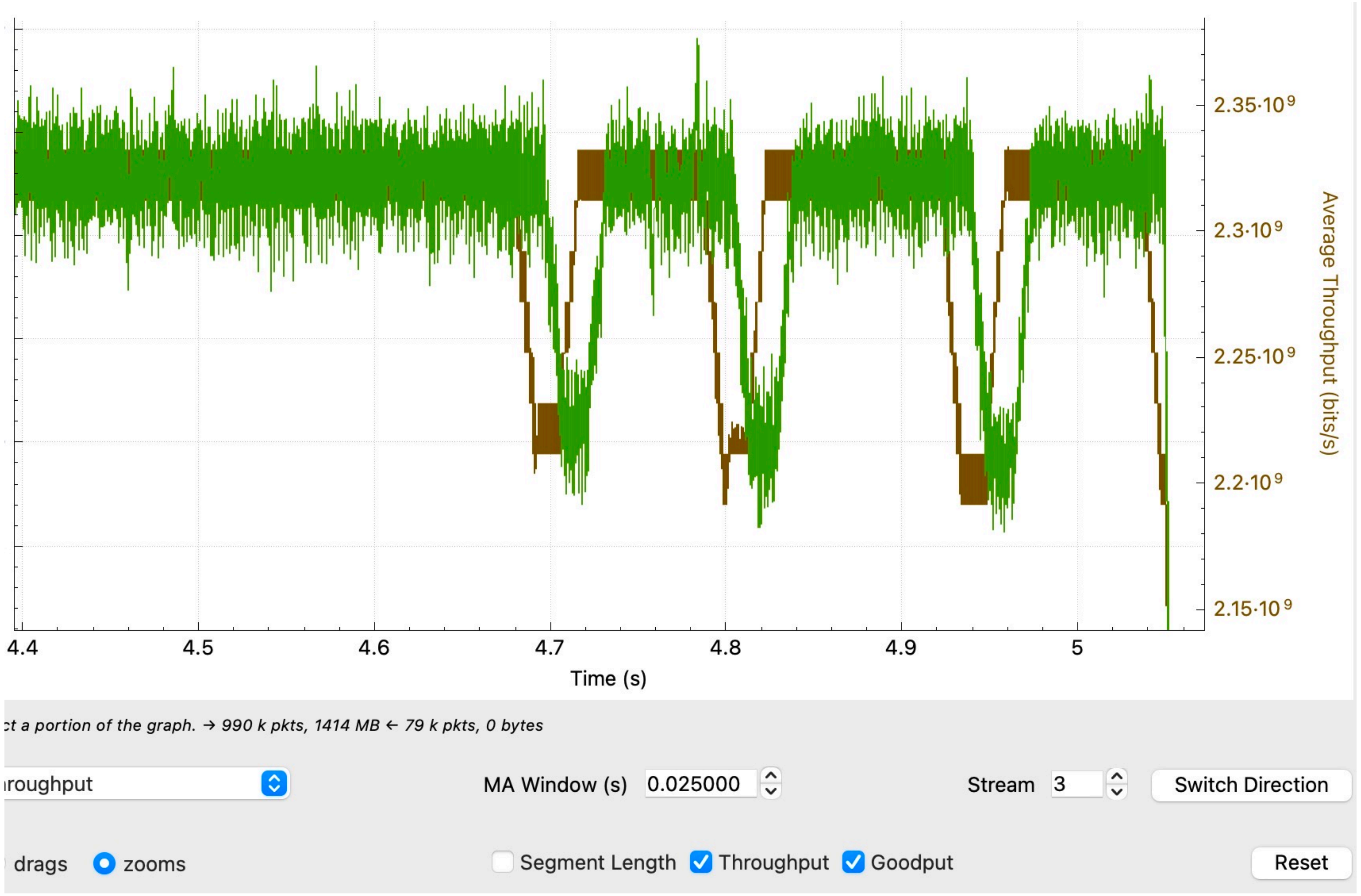
Segment Length [✓ Throughput] [✓ Goodput]

Reset

Mastering The Somewhat Peculiar Wireshark Graph Interface

- Ensure you've got a suitable level of granularity for your graph.
    - Adjust the MA (moving average) setting to a smaller value (the graph on the next page shows the result for a setting of 0.025 seconds). You may need to play with different values until you find something that's works (and looks) right.
- The default display covers the whole capture interval. As a result, it's hard to see details.
- **To zoom in, ensure you've got "Mouse zooms" selected, then drag a box around the region of interest**. Wireshark will zoom in. Need to zoom in still more? Repeat.
- **Zoom in too much (or in the wrong area)?** Use the **Reset** button to "self-rescue."
- Not seeing the region of the graph you're interested in? Select **"Mouse drags"** and drag the display port to the show the portion of the graph you're interested in.
- Move to the wrong region of the graph? Use the Reset button to get back to the "big picture" display you started with.

Sample "Zoomed In" Statistics→TCP Stream Graphs→Time Sequence



ct a portion of the graph. → 990 k pkts, 1414 MB ← 79 k pkts, 0 bytes

roughput

MA Window (s) 0.025000

Stream 3

drags [✓ zooms]

☐ Segment Length ✓ Throughput ✓ Goodput

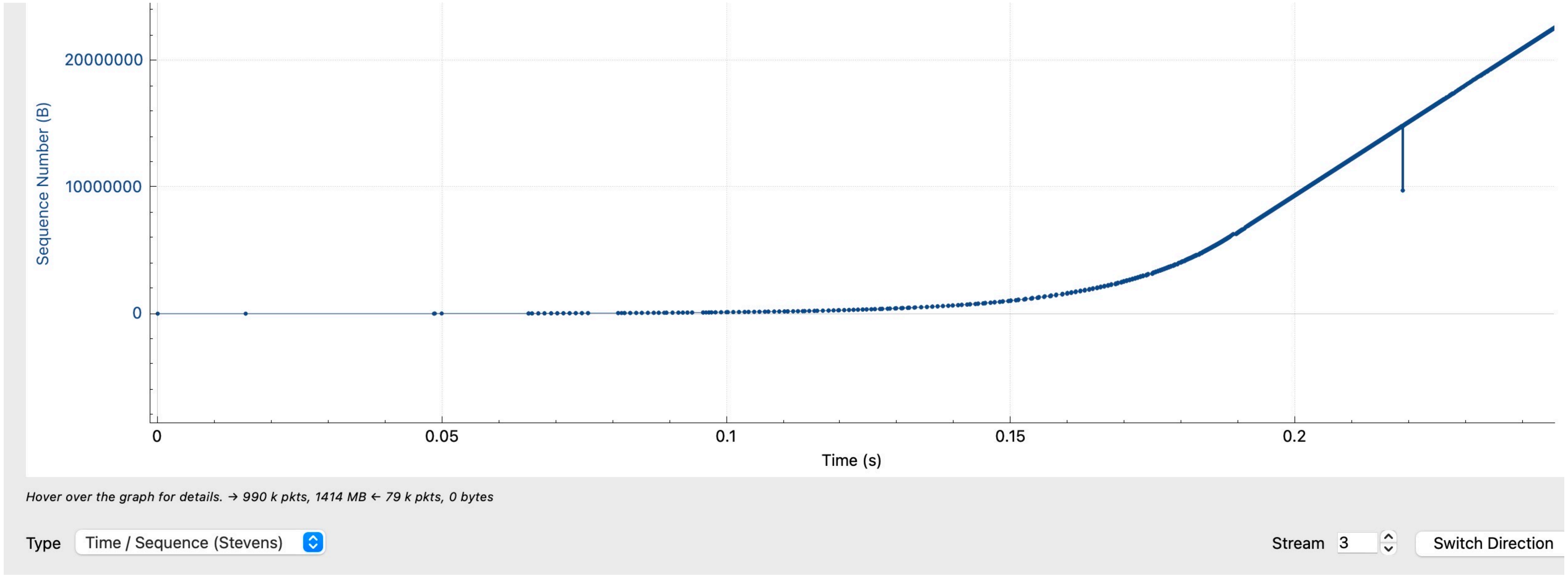
More "Exotic" Graphs Are Also Available

- **The Stevens Time Sequence Graph** (Sequence Number vs. Time) – our example really demonstrates the accelerating packet transfer rate over time.

- **The Window Scaling Graph** lets you see what congestion window was arrived at, and how that window changes over time. In our case, after startuo it's near flat, except for a few brief blips.

- We'll also highlight the **RTT or Round-Trip Time Graph.** This is a graph of observed latencies. Note the vertical scale (and small variance to the RTT values) in this trace.

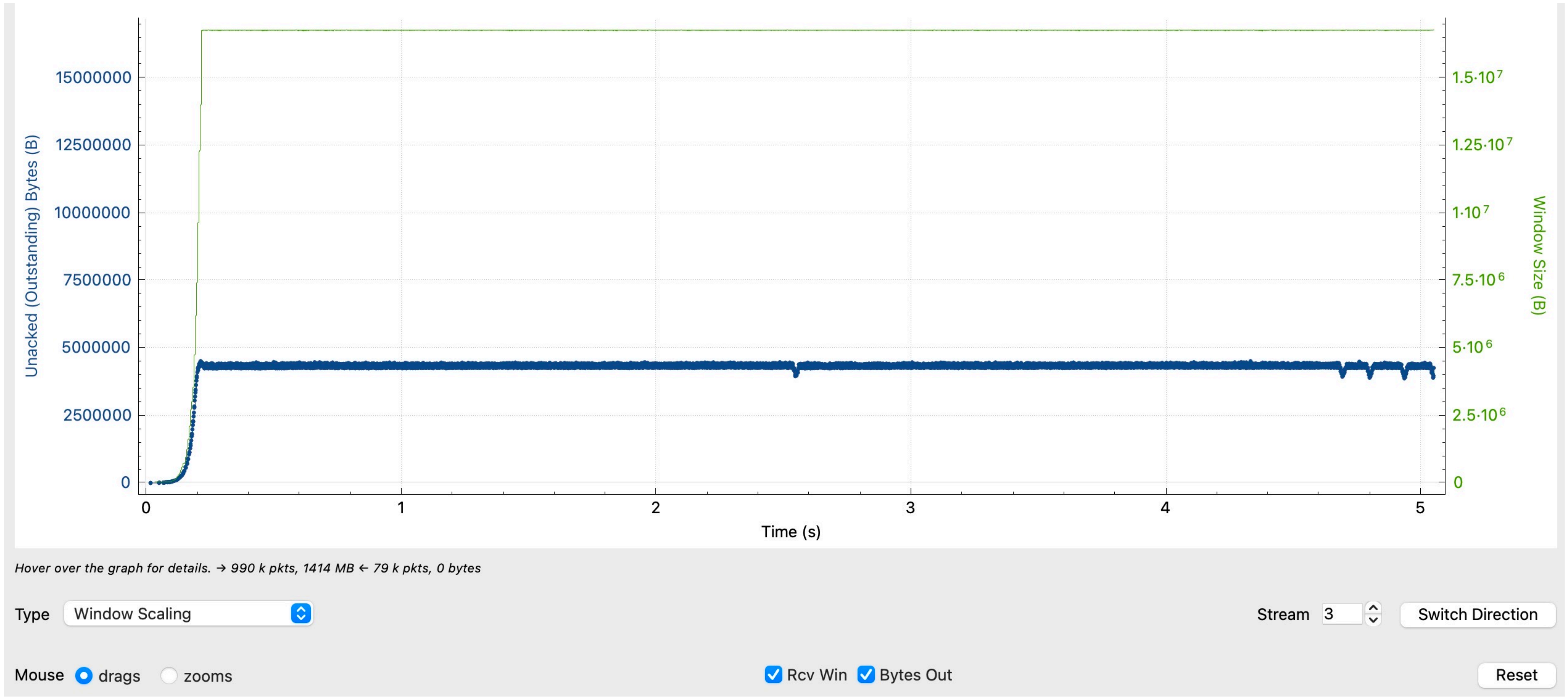
Statistics → TCP Stream Graphs → Time Sequence (Stevens)



Hover over the graph for details. → 990 k pkts, 1414 MB ← 79 k pkts, 0 bytes

Note the initially sparse dots that quickly begin to appear more rapidly and then begin to climb rapidly within just a quarter of a second.

Statistics → TCP Stream Graphs → Time Sequence (Window Scaling)



Hover over the graph for details. → 990 k pkts, 1414 MB ← 79 k pkts, 0 bytes

Type: Window Scaling

Stream: 3

Switch Direction

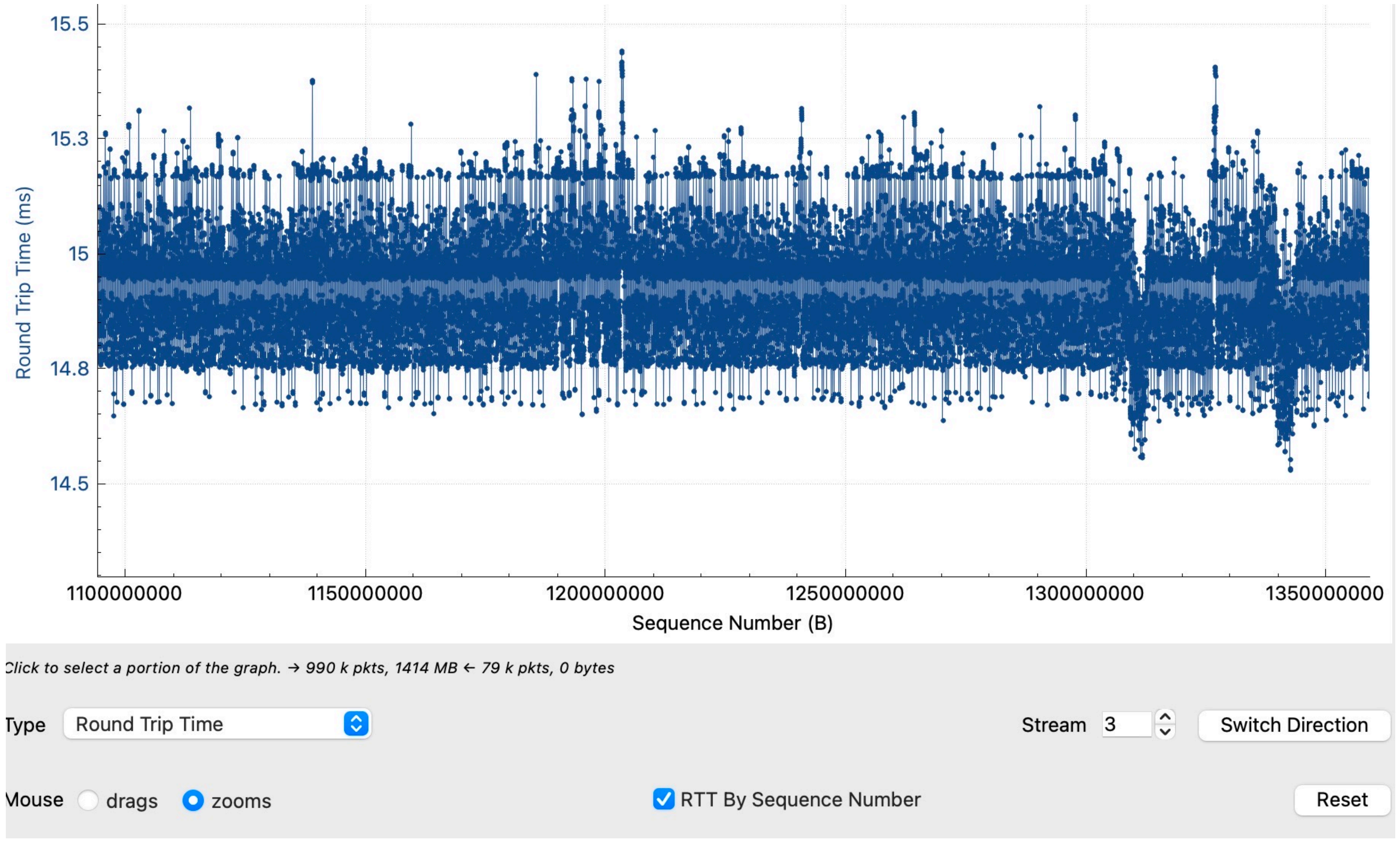
Mouse [✓ drags] zooms

✓ Rcv Win ✓ Bytes Out

Reset

The window scale climbs rapidly, and then basically stays flat…

Statistics → TCP Stream Graphs → Time Sequence (RTT, Zoomed)



Click to select a portion of the graph. → 990 k pkts, 1414 MB ← 79 k pkts, 0 bytes

Type: Round Trip Time

Stream: 3

Mouse: drags [✓ zooms]

[✓] RTT By Sequence Number

Not much variation in round trip times (half a milisecond or so?)

It's Not All "Just About Graphs"

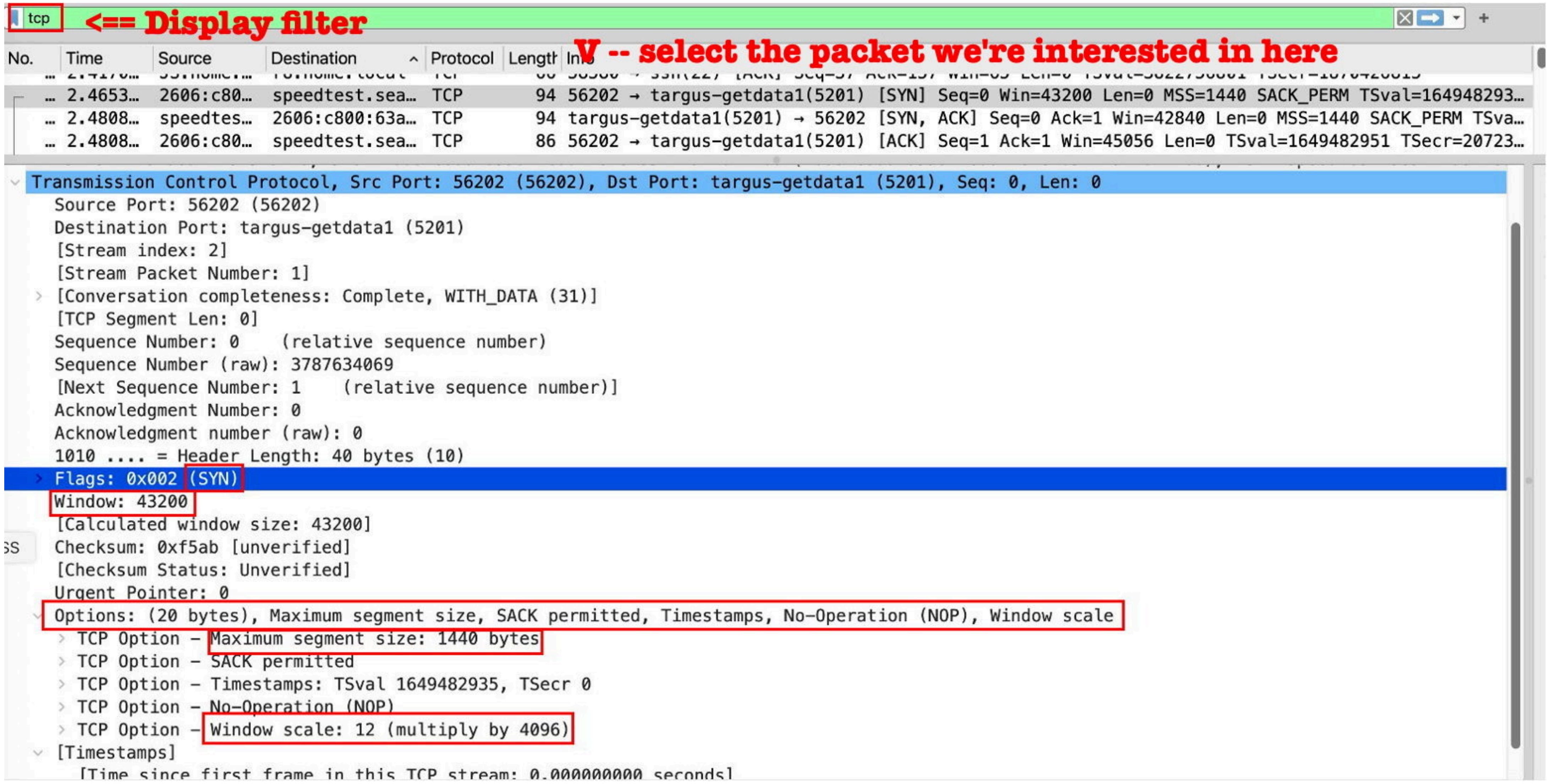
- You can also just dig in on specific packets that may be of interest.

- For example, we talked about various options that get set during the TCP three-way handshake.

- Let's look at a sample SYN, SYNACK, and ACK sequence, and confirm the window scaling factor and TCP options.

Digging In To TCP Packets Details (Set the Display Filter to tcp)

- **SYN packet**, setting maximum segment size, Window Scale; OK'ing SACK and TImestamps

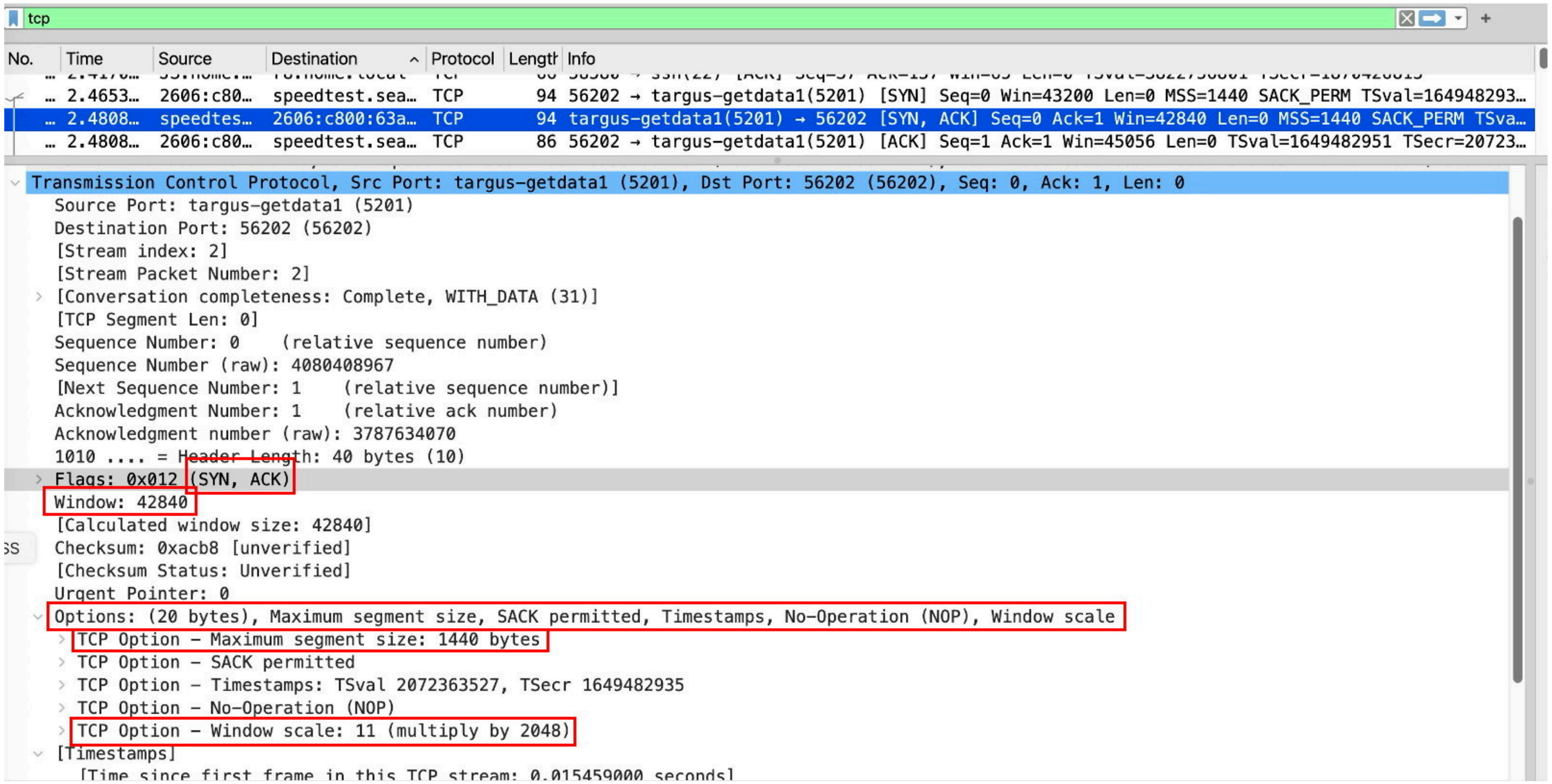


```
No.  Time       Source       Destination       Protocol  Length  Info
...  2.4653...  2606:c80...  speedtest.sea...  TCP       94  56202 → targus-getdata1(5201) [SYN] Seq=0 Win=43200 Len=0 MSS=1440 SACK_PERM TSval=164948293...
...  2.4808...  speedtes...  2606:c800:63a...  TCP       94  targus-getdata1(5201) → 56202 [SYN, ACK] Seq=0 Ack=1 Win=42840 Len=0 MSS=1440 SACK_PERM TSva...
...  2.4808...  2606:c80...  speedtest.sea...  TCP       86  56202 → targus-getdata1(5201) [ACK] Seq=1 Ack=1 Win=45056 Len=0 TSval=1649482951 TSecr=20723...

Transmission Control Protocol, Src Port: 56202 (56202), Dst Port: targus-getdata1 (5201), Seq: 0, Len: 0
    Source Port: 56202 (56202)
    Destination Port: targus-getdata1 (5201)
    [Stream index: 2]
    [Stream Packet Number: 1]
  > [Conversation completeness: Complete, WITH_DATA (31)]
    [TCP Segment Len: 0]
    Sequence Number: 0    (relative sequence number)
    Sequence Number (raw): 3787634069
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 0
    Acknowledgment number (raw): 0
    1010 .... = Header Length: 40 bytes (10)
  > Flags: 0x002 (SYN)
    Window: 43200
    [Calculated window size: 43200]
    Checksum: 0xf5ab [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
    Options: (20 bytes), Maximum segment size, SACK permitted, Timestamps, No-Operation (NOP), Window scale
      > TCP Option - Maximum segment size: 1440 bytes
      > TCP Option - SACK permitted
      > TCP Option - Timestamps: TSval 1649482935, TSecr 0
      > TCP Option - No-Operation (NOP)
      > TCP Option - Window scale: 12 (multiply by 4096)
  v [Timestamps]
      [Time since first frame in this TCP stream: 0.000000000 seconds]
```

We saw a SYN, next we see a SYN, ACK

- **SYNACK,** setting maximum segment size, Window Scale, OK'ing SACK and TImestamps



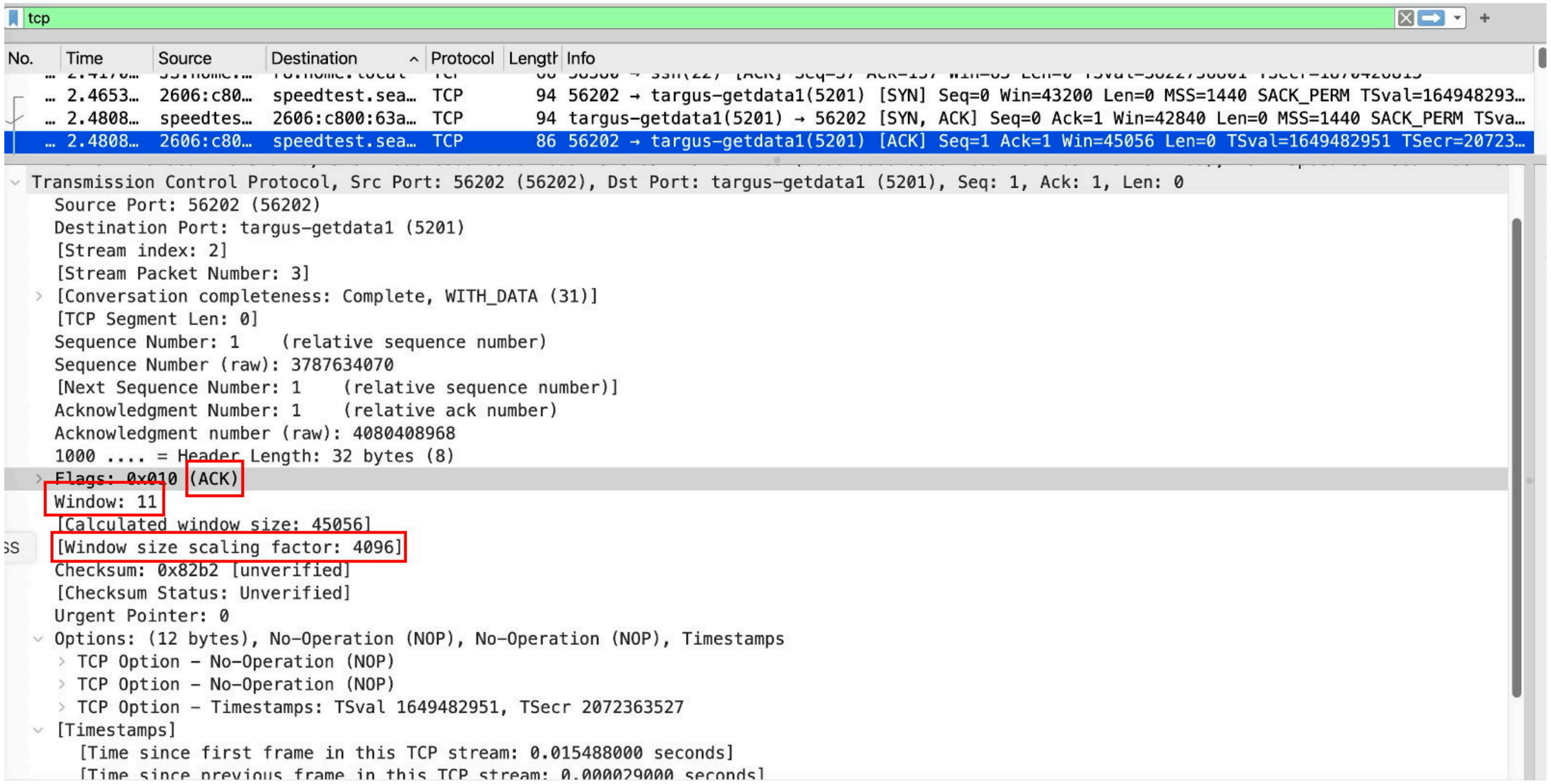
```
tcp

No.  Time      Source     Destination       Protocol  Length Info
…    2.4653…   2606:c80…  speedtest.sea…    TCP       94  56202 → targus-getdata1(5201) [SYN] Seq=0 Win=43200 Len=0 MSS=1440 SACK_PERM TSval=164948293…
…    2.4808…   speedtes…  2606:c800:63a…    TCP       94  targus-getdata1(5201) → 56202 [SYN, ACK] Seq=0 Ack=1 Win=42840 Len=0 MSS=1440 SACK_PERM TSva…
…    2.4808…   2606:c80…  speedtest.sea…    TCP       86  56202 → targus-getdata1(5201) [ACK] Seq=1 Ack=1 Win=45056 Len=0 TSval=1649482951 TSecr=20723…

Transmission Control Protocol, Src Port: targus-getdata1 (5201), Dst Port: 56202 (56202), Seq: 0, Ack: 1, Len: 0
    Source Port: targus-getdata1 (5201)
    Destination Port: 56202 (56202)
    [Stream index: 2]
    [Stream Packet Number: 2]
  > [Conversation completeness: Complete, WITH_DATA (31)]
    [TCP Segment Len: 0]
    Sequence Number: 0    (relative sequence number)
    Sequence Number (raw): 4080408967
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 1    (relative ack number)
    Acknowledgment number (raw): 3787634070
    1010 .... = Header Length: 40 bytes (10)
  > Flags: 0x012 (SYN, ACK)
    Window: 42840
    [Calculated window size: 42840]
    Checksum: 0xacb8 [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
  v Options: (20 bytes), Maximum segment size, SACK permitted, Timestamps, No-Operation (NOP), Window scale
    > TCP Option - Maximum segment size: 1440 bytes
    > TCP Option - SACK permitted
    > TCP Option - Timestamps: TSval 2072363527, TSecr 1649482935
    > TCP Option - No-Operation (NOP)
    > TCP Option - Window scale: 11 (multiply by 2048)
  v [Timestamps]
      [Time since first frame in this TCP stream: 0.015459000 seconds]
```

We saw a SYN and a SYNACK, finally we see an ACK

- **Our three-way handshake is now complete...**



```
tcp

No.  Time      Source      Destination       Protocol Length Info
… 2.4653…  2606:c80…  speedtest.sea…  TCP   94  56202 → targus-getdata1(5201) [SYN] Seq=0 Win=43200 Len=0 MSS=1440 SACK_PERM TSval=164948293…
… 2.4808…  speedtes…  2606:c800:63a…  TCP   94  targus-getdata1(5201) → 56202 [SYN, ACK] Seq=0 Ack=1 Win=42840 Len=0 MSS=1440 SACK_PERM TSva…
… 2.4808…  2606:c80…  speedtest.sea…  TCP   86  56202 → targus-getdata1(5201) [ACK] Seq=1 Ack=1 Win=45056 Len=0 TSval=1649482951 TSecr=20723…

Transmission Control Protocol, Src Port: 56202 (56202), Dst Port: targus-getdata1 (5201), Seq: 1, Ack: 1, Len: 0
  Source Port: 56202 (56202)
  Destination Port: targus-getdata1 (5201)
  [Stream index: 2]
  [Stream Packet Number: 3]
  [Conversation completeness: Complete, WITH_DATA (31)]
  [TCP Segment Len: 0]
  Sequence Number: 1    (relative sequence number)
  Sequence Number (raw): 3787634070
  [Next Sequence Number: 1    (relative sequence number)]
  Acknowledgment Number: 1    (relative ack number)
  Acknowledgment number (raw): 4080408968
  1000 .... = Header Length: 32 bytes (8)
  Flags: 0x010 (ACK)
  Window: 11
  [Calculated window size: 45056]
  [Window size scaling factor: 4096]
  Checksum: 0x82b2 [unverified]
  [Checksum Status: Unverified]
  Urgent Pointer: 0
  Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
    TCP Option - No-Operation (NOP)
    TCP Option - No-Operation (NOP)
    TCP Option - Timestamps: TSval 1649482951, TSecr 2072363527
  [Timestamps]
    [Time since first frame in this TCP stream: 0.015488000 seconds]
    [Time since previous frame in this TCP stream: 0.000029000 seconds]
```

There's A Ton More You Can Do With Wireshark, But...

- We've got limited time today!

- The trick is taking it a step at a time. Spend some time playing with Wireshark, digging into using your own sample captures.

  There are also some books you may want to check out, such as:

  Troubleshooting with Wireshark: Locate the Source of Performance Problems,

  https://www.amazon.com/Troubleshooting-Wireshark-Locate-Performance-Problems/dp/1893939979

Aside: Need Something To Investigate? Run `$ netstat -st`

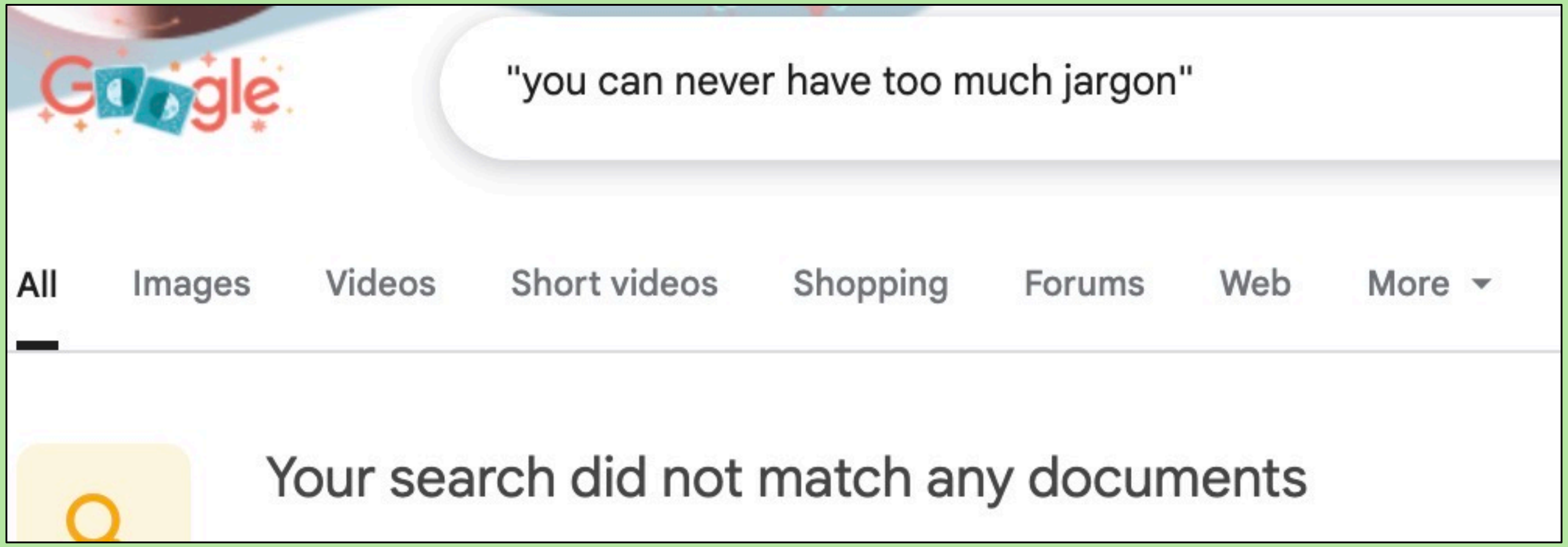
```
$ netstat -st
tcp:
[...]
           500 data packets sent after flow control
           79 challenge ACKs sent due to unexpected SYN
           41 challenge ACKs sent due to unexpected RST
[...]
           13504 completely duplicate packets (35068032 bytes)
           43 old duplicate packets
           [...]
           1986 packets with some dup. data (6469008 bytes duped)
           3570597 out-of-order packets (4290634244 byte)
[...]
           236971 packets recovered after loss
           7498 packets received after close
           41 bad resets
[...]
```

Part III. Summary

We've Covered A Lot of Material Today – TAKE AWAYS

- We explained that multigigabit residential networks have become affordable and easy to deploy, and we showed you how (if you use MacOS, Windows 11, or Debian 12 Linux).
- Corporate networks are similarly growing in speed and dropping in price – 100 Gbps is routine & 400 Gbps isn't rare any longer (but YOUR network may not be keeping up!)
- These new powerful networks are (or should be) on M3AAWG's "radar" as a potential source of abusive traffic.
- We showed you how to use SpeedTest & iperf3 to actively measure network throughput.
- We talked about some of the factors that can technically impact network performance, such as latency. We made recommendations for coping with this by suitably-sizing buffers and adopting an alternative to the CUBIC congestion control algorithm known as BBR.
- We talked briefly about bufferbloat and its potential impact on jitter and latency sensitive traffic, and how BBR can largely eliminate it as an issue.
- We finished up by showing you how to get started using tcpdump and Wireshark to look more closely at network traffic.
- We hope you learned at least something new today! **Are there any questions?**

Part IV. Glossary



"you can never have too much jargon"

Your search did not match any documents

Glossary

AIMD	Additive Increase, Multiplicative Decrease – the general rule for congestion management
amd64	Modern Intel-compatible 64-bit CPU family
Apple SIP	System Integrity Protection – protective measure meant to avoid O/S compromises
Apt	Linux package manager, used to manage add-on software packages (like brew for the Mac)
Baseline	Starting profile
Bandwidth	Network capacity (measured in bits per second)
Bandwidth Delay Product (BDP)	bandwidth * latency == size of required buffers
BBR	Bottleneck Bandwidth and Round-Trip propagation time, a congestion control scheme
Bit	One binary digit (0 or 1); eight bits make up a byte (also known as an octet)
Bottleneck	In a network with multiple sequential links, the link with the lowest available capacity
Buffer	Area for temporary storage of incoming or outgoing packets
Bufferbloat	Undesirable condition where large bulk flows delay delivery of small (jitter sensitive) flows
Byte	One octet (eight bits)
Cat<N> cable	Copper Ethernet cable rated to handle a particular speed of traffic. You want Cat6a or Cat8.
CDN	Content distribution network: service that replicates content at diverse locations to ensure content is close to where its demanded, and there's sufficient redundancy
CLI	Command Line Interface (type-commands-in/non-graphical interface)
Congestion	Network that's facing more demand than it has capacity

Glossary (cont. 1)

Congestion Control Algorithm	Algorithm that specifies what happens when sending traffic through limited-capacity network links – how fast does traffic grow? how does traffic decrease?
Congestion Control Window	CWND: dynamically manages the speed at which traffic gets transferred
Congestive Collapse	A disaster where a network fills up, and then, instead of backing off, multiple senders just keep blasting away
CUBIC	Most common congestion control algorithm
Cut-through	Network switch that begins forwarding a packet as soon as it has partially received it
Debian 12	The latest version of Debian Linux as of the time this was written (known as "Bookworm")
Dongle	Adapter that dangles from the side or back of a system
Download	Transfer data down from the Internet to the user
Driver	Software that is supplied with a peripheral that allows a system to see and use the device
ECN	Explicit Control Notification: way of explicitly telling a sender when the receiver's full
Ethernet	Fundamental data link protocol (running at layer 2 of the OSI model)
Ethernet switch	Layer 2 device allowing Ethernet devices to interconnect
Fairness	Scenario where multiple users get an appropriate share of network capacity
FTTH	Fiber To The Home
Gbps	1,000,000,000 bits per second (125,000,000 bytes per second)
Fairness	Scenario where multiple users get an appropriate share of network capacity

Glossary (cont. 2)

FTTH	Fiber To The Home
Gbps	1,000,000,000 bits per second (125,000,000 bytes per second)
Geolocation	Mapping an IP to a physical location on the globe
Geostationary satellite	Satellite orbiting at 22,236 miles above the equator, matching the earth's rotation
GUI	Graphical (Point-and-Click) User Interface
ICMP	Internet Control Message Protocol – used for error reporting, flow control and other tasks
Interframe gap	Space between two Ethernet frames (12 bytes long)
Iperf3	Command-line interface bandwidth testing tool
IPv4	Classic dotted quad format addresses (www.m3aawg.org is 34.214.179.220)
IPv6	New longer-form addresses (www.ietf.org uses 2606:4700::6810:2d63)
Jitter	Variability in packet pacing
Jumbo frames	Most networks use 1500-byte frames, but some may be configured to use 9,000 byte "jumbos"
Lag	Undesirable delay when playing an interactive online game
Lambda	Optical wavelength used for dedicated point-to-point high bandwidth connections
Latency	Network delay between two locations, normally measured in milliseconds
Layer 2	Switch Ethernet traffic based just on hardware MAC addresses
Layer 3	Route Ethernet traffic based on IP addresses
LFN	Long Fat Network (a long distance, high-capacity network)

Glossary (cont. 3)

Line rate	Traffic running at the full capacity of the connection (e.g., 1000 Mbps over a gigabit link)
LEO satellite	Satellite in Low Earth Orbit (under 2,000 kilometers), relying on a constellation of multiple satellites that "hand off" traffic from one satellite to another as they pass overhead
MAC address	A (normally globally unique) hardware address associated with an Ethernet adapter
Managed switch	"Smart" switch that can be manually configured or monitored by a network administrator
Mbps	1,000,000 bits per second (125,000 bytes per second)
MHz	Megahertz (amount of wireless spectrum used)
Mpps	Million packets per second, a measure of a switch's packet handling capacity
MRC	Monthly recurring charge
Msec	1 millisecond (1/1,000th of a second)
Multistream	Application that uses multiple parallel streams instead of relying on just a single stream
M1/M2/M3/M4	Family of Apple Silicon CPUs used for MacBooks and similar systems
NDAA	U.S. National Defense Authorization Act, limits purchase of some networking equipment
Network interface	Ethernet connection or wireless link used to send/receive traffic on a system
Nominal	Satisfactory, normal
NRC	Non-recurring charge ("one time fee")
Optical Terminal	Fiber optic "modem"
O/S	Operating system (Windows 11, MacOS, Linux, etc.)

Glossary (cont. 4)

OSI 7 Layer Model	Way of stratifying the network connection process: physical layer, data link layer, network layer, transport layer, session layer, presentation layer, application layer.
Paced traffic	Network traffic released to the network at a chosen bitrate
Packet capture	Also known as a PCAP file. Often captured using tcpdump or directly within Wireshark.
Packets	Traffic on the Internet is broken up into 1500 byte or smaller chunks known as packets
Packet loss	What happens to traffic on a congested network after the buffers fill up and data overflows
PCAP	Packet capture file (e.g., as created with tcpdump or Wireshark)
PCIe	PCI Express, a high-speed interface to the computer's bus used for some network cards
Peering	Exchange customer traffic only (may be "settlement (fee) free," or paid)
Peering point	Location where multiple networks meet to exchange customer traffic
Ping	Un*x command line tool for estimating network latency.
PLL	Permanent link length (length of permanently installed network cabling)
Pluggable transceivers	Modules that can be used on SFP+ switches to connect different types of cable
POE	Power over Ethernet. Allows you to connect remote wireless access points w/o wall power.
Port	Place on a network device where you can plug in a network cable. Alternatively, the number of a network service (port 443=https, 25=SMTP, 53=DNS, etc.)
Port fee	Amount paid by an ISP to connect a circuit to the network service provider's network device
QUIC	Quick UDP Internet Connections (see https://quicwg.org/ for more)

Glossary (cont. 5)

Redundant	A switch with "redundant power supplies" can lose one power supply and keep running.
Reliable delivery	Traffic that's sent are guaranteed to be received, and will be resent if necessary
RFC 1323 Window Scaling	Extension that allows transmission windows to scale up to sizes required for Long Fat Networks
Right of way	Right to run cable along a highway, railroad, powerline, pipeline, etc.
RJ45	Copper Ethernet cable connection format
Round trip time	The time it takes to go from source to destination and back, usually measured in msec
SACK	RFC2018 Selective Acknowledgements (way of efficiently acknowledging packet reception)
SCTP	Stream Control Transmission Protocol (RFC9260)
Sequoia 15.4.1	Latest operating system from Apple for laptops and servers (as of the time this was written)
SFP+	Pluggable fiber optic connection standard
SNMP counters	Simple counters on some devices that track and report things like packets sent in/out
Spatial streams	Number of streams simultaneously sent over individual WiFi antennas
Speedtest	Consumer GUI/web-based bandwidth testing tool
Store-and-forward	Switch that receives an entire packet before it subsequently forwards it back out.
Symmetric	Same in both directions (as in "same bandwidth whether uploading or downloading")
TAA	U.S. Trade Agreements Act, requires some to purchase America-made products
TCP	Transmission Control Protocol

Glossary (cont. 6)

Tcpdump	Command line tool used to capture packet traffic for subsequent analysis
Three-Way Handshake	The SYN, SYNACK, ACK process used to establish a TCP connection
Timestamps	RFC1323 timestamps put into packets to enable packet pacing & delay measurement
Transit	Obtain access to the full Internet (involves a charge for service)
Tuning	Tweaking (adjusting) network parameters to try to achieve desired network performance
UDP	User Datagram Protocol, a fire-and-forget packet delivery protocol, alternative to TCP
Unmanaged switch	"Dumb" switch that auto-negotiates all connections; non-manually configurable.
Upload	Transfer data up from the user to the Internet
USB	Universal Serial Bus (socket for plugging in mice, keyboards, dongles, etc. on a system)
VLANs	Virtual LANs -- way of logically separating traffic on a switch's physical interfaces
VPN	Virtual Private Network (temporary encrypted tunnel from a workstation to a concentrator)
WiFi	Wireless Ethernet network
Windows 11	Latest operating system from Microsoft for laptops and servers (as of the time this written)
Wireshark	Packet capture analysis tool
X86	Family of Intel (and Intel-compatible) CPUs
10/100/1000/2.5Gbase-T	10/100/1000/2500 Mbps Ethernet over copper cable
5G	Fifth-generation advanced cellular wireless service
802.11ax	Example of one specific WiFi protocol

"I can't get no satisfaction
I can't get no satisfaction
'Cause I try and I try and I try and I try"

Rolling Stones, "Satisfaction" (1981)
https://www.youtube.com/watch?v=qAzqSYQ9X9U
3.44 million views