**PacketFest '25**

**Where ntop and Wireshark Communities Meet**

**8 - 9 May, Zürich**

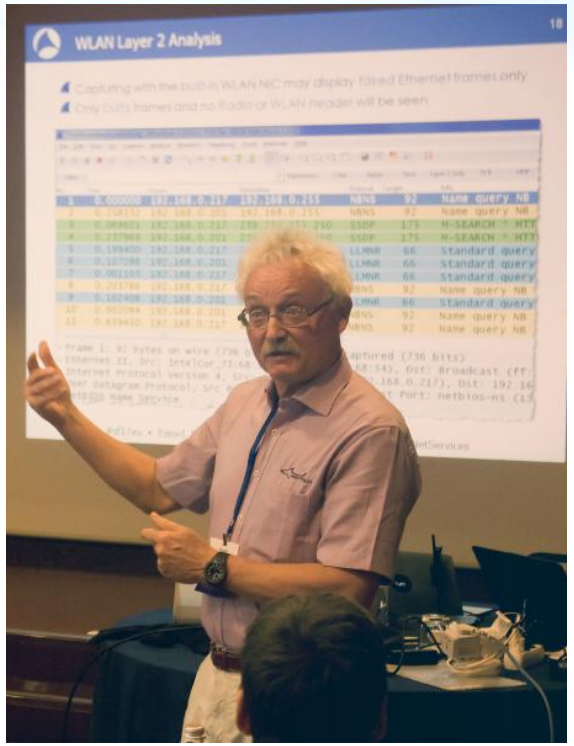
WIRESHARK

*The Open Source Packet Analyzer*

# The 7 Senses Packet Detective

presented by Rolf Leutert

LEUTERT NetServices




Rolf Leutert, Ing. HTL
Network Protocol Expert
Leutert NetServices
Zürich-Airport, Switzerland

- Network Analysis & Troubleshooting
- Trainings TCP/IP, QUIC, WLAN, VoIP, IPv6
- Wireshark® Certified Network Analyst 2010
- Wireshark® Instructor since 2006
- Sniffer® certified Instructor since 1990

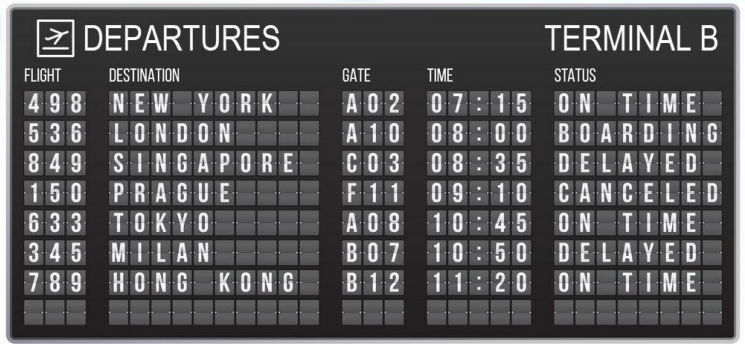
Sniffer Certified MASTER

Wireshark University WIRESHARK® CERTIFIED NETWORK ANALYST

leutert@netsniffing.ch / www.netsniffing.ch

LEUTERT NetServices

# Technology: 40 years ago

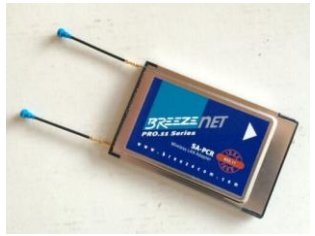



*Split Flap Display Board Zurich Airport 1984*

- 1984 Head of the new LAN group at Swissair
- 1984 Testing Coax-based Broadband LAN technology
- 1985 Roll-out Broadband LAN for Passenger & Staff Information
- > 80 Split Flap Displays, > 150 Public TVs, > 700 Staff devices

*IBM PS/2 & Token-Ring for Check-In & Boarding Gates*

- 1986 Start testing Token Ring and Ethernet components
- 1986 Yellow Coaxial-Cable Ethernet for Flight Simulator Area
- 1989/90 Roll-out Airport Zürich: >100 Token Rings, >4000 PCs
- Largest Rings > 7 km at Boarding Gates





*BreezeNet, first commercial 802.11 WLAN card*

- 1999 First outdoor WLAN 802.11: > 60 Outdoor Access Points
- Airplane Handling: Maintenance, Cleaning, Catering etc.
- 2001 Swissair Grounding, Airport Authority takes over ICT.

+

LEUTERT NetServices

# Network Management: 40 years ago

## Device Management Systems

→ All **proprietary** management systems!
→ No **SNMP** (1988), no **NetFlow** (1996)

- **IBM** SNA Network & Controllers
- **PROTEON** Token Ring MAUs
- **ANDREW** Token Ring Bridges
- **CABLETRON** Ethernet Hubs
- **DEC** Ethernet Bridges
- **SynOptics** Ethernet Hubs & Bridges
- **Wellfleet** Ethernet Routers
- **3COM** Ethernet Hubs & Bridges
- **Bay Networks** Ethernet Hubs & Bridges
- **Cisco** Token Ring & Ethernet Routers
- **BreezeCOM** WLAN Access Points

## Network Protocols

Using **Multiprotocol Routers, Bridges, WAN Gateways**

- **IBM SNA** (not routable)
- **IBM NetBeui** (not routable)
- **DecNet**
- **Novell IPX**
- **AppleTalk**
- **TCP/UDP/IP**
- **Token-Ring, Ethernet**
- **Frame Relay**
- **FDDI**
- **ATM**
- **WAN** Protocols like **ISDN**

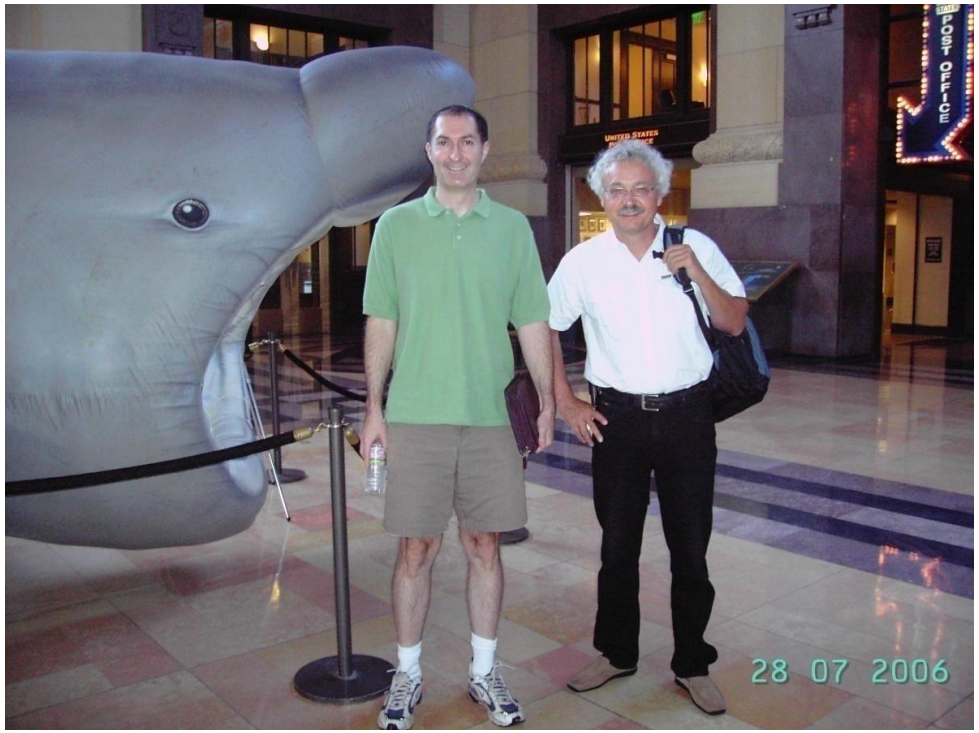
# Network Analysis: 40 years ago

1987 Swissair purchased the first Token Ring / Ethernet Sniffer ® in Europe

- **Compaq Portable** (12 kg)
- **Intel 8088**, 4.7 MHz, 128 KB RAM
- **Price** (with 5 protocol suites)
  **50'000 US$** (todays equivalent)

**Sniffer®** from Network General

**Ethereal ®**




Gerald Combs (left) and Rolf Leutert, July 2006 in Kansas City / USA

LEUTERT NetServices

# Network Management: Today and tomorrow

## What has changed since then?

Network Management Systems

- **SNMP** & **NetFlow** became standards
- No **proprietary** management systems
- Lots of **Open-Source tools** available

Network Protocols

- **Token Ring** and **other** disappeared
- Most **vendor protocol** disappeared
- **Ethernet** became standard
- **Ethernet** replaces most **WAN** protocols
- **POE** Power over Ethernet
- **WLAN** widely implemented
- **VoIP** widely implemented
- **New IPv6** protocol

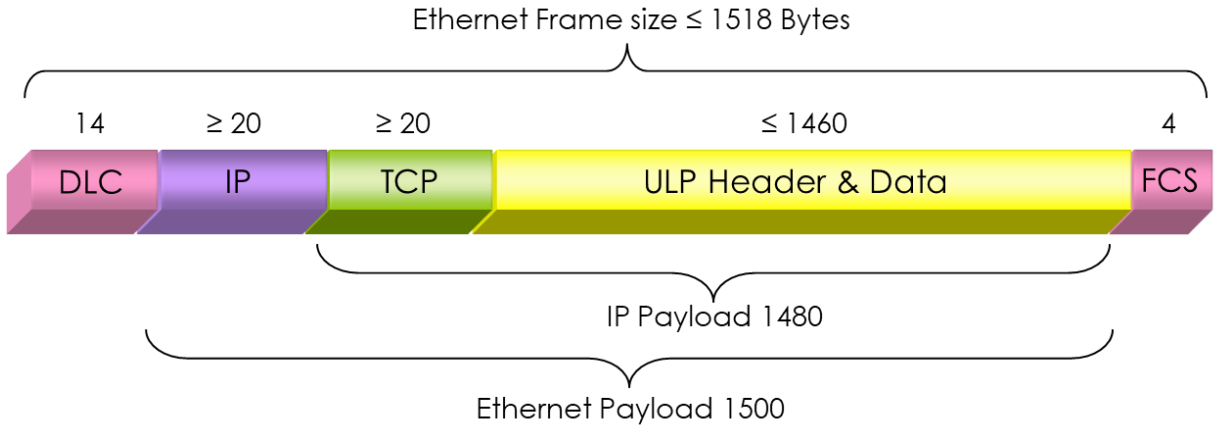
## What are the new challenges?

- **Cyber Threats** became a big issue
- Short product **Live Cycles** and costs
- Increasing **amount** of data
- Complexity of **Network Virtualisation (SDN)**
- New **QUIC** protocol, fully encrypted
- New **AI technology**, curse or blessing?

# What has survived 40 years and is still often used?

- Hardware? **No**
- Operating systems? **No**
- Programming Languages: **Some**
- Proprietary management systems? **No**

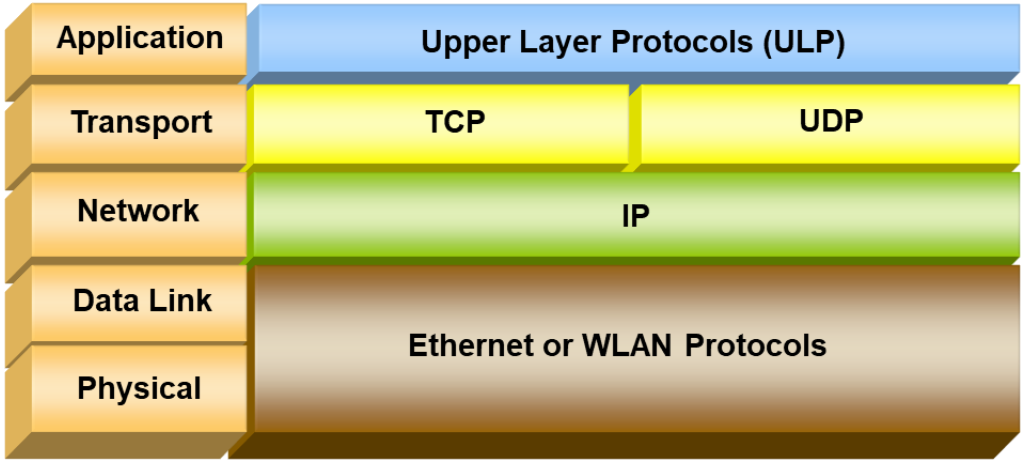
- **Ethernet? Yes**
- Still uses the same original frame format
- Scales from **10Mbps to 1.6 Tbps** and more
- Tagging options added (VLAN, MPLS etc.)




- **IPv4? Yes**
- Still uses the same header format (≥ 20 Bytes)

- **UDP? Yes**
- Still uses the same header format (8 Bytes)

- **TCP? Yes**
- Still uses the same header format (≥ 20 Bytes)
- Some new options (SACK, Window Scaling etc.)



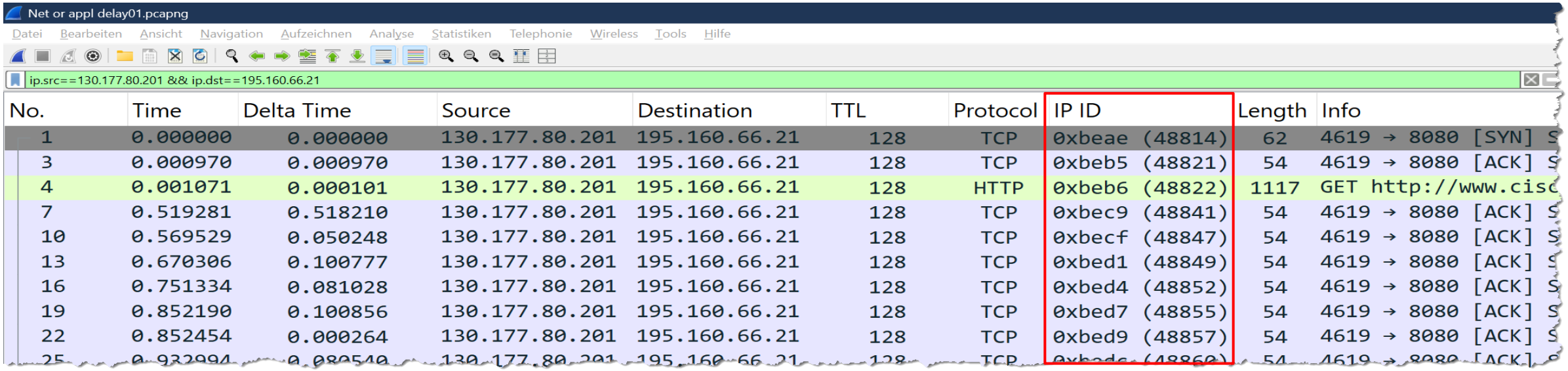
# Troubleshooting using IP and TCP fields

The **UDP** header does not contain many useful fields for network troubleshooting.
However, the **IP and TCP fields** are still the basis for narrowing down network problems

A closer look at the **IP header:**

- In addition to the well-known **IP addresses** and **TTL** fields, there is another helpful field:
- One of the most overlooked field is the **IP Identification** field (IP ID, 2 Bytes)
- Originally defined for **re-assembling fragmented frames**, it can help you in other situations
- The **IP Identification** field is **incremented by "1"** by the sender in each packet sent to **any destination**

→ **Use case 1:** **Large gaps** in the IP ID sequence means the sender is busy serving also **other destinations**



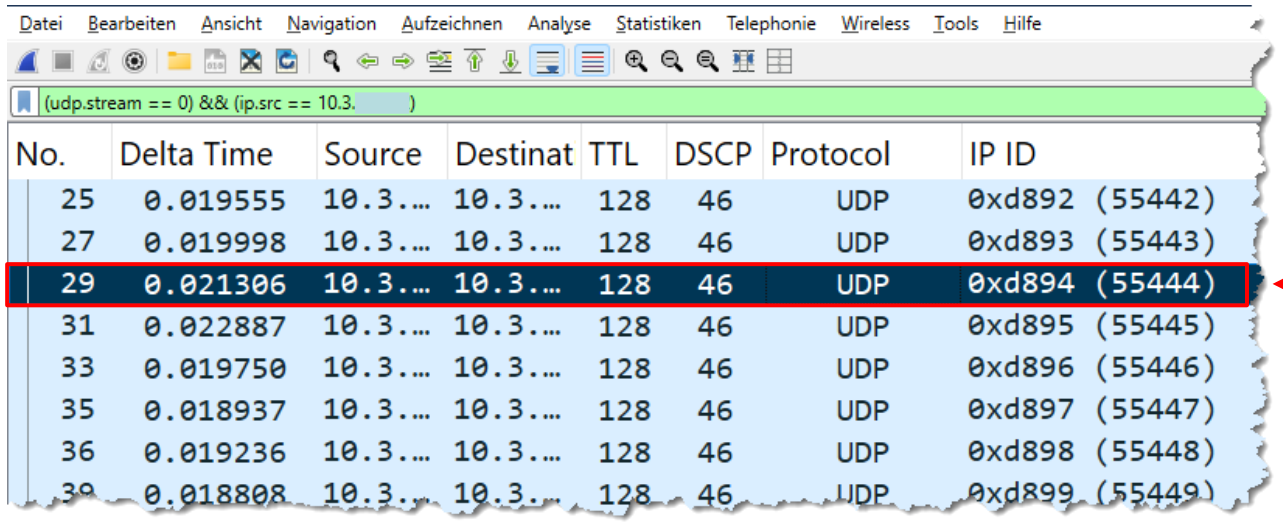

| No. | Time | Delta Time | Source | Destination | TTL | Protocol | IP ID | Length | Info |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.000000 | 0.000000 | 130.177.80.201 | 195.160.66.21 | 128 | TCP | 0xbeae (48814) | 62 | 4619 → 8080 [SYN] S |
| 3 | 0.000970 | 0.000970 | 130.177.80.201 | 195.160.66.21 | 128 | TCP | 0xbeb5 (48821) | 54 | 4619 → 8080 [ACK] S |
| 4 | 0.001071 | 0.000101 | 130.177.80.201 | 195.160.66.21 | 128 | HTTP | 0xbeb6 (48822) | 1117 | GET http://www.cisc |
| 7 | 0.519281 | 0.518210 | 130.177.80.201 | 195.160.66.21 | 128 | TCP | 0xbec9 (48841) | 54 | 4619 → 8080 [ACK] S |
| 10 | 0.569529 | 0.050248 | 130.177.80.201 | 195.160.66.21 | 128 | TCP | 0xbecf (48847) | 54 | 4619 → 8080 [ACK] S |
| 13 | 0.670306 | 0.100777 | 130.177.80.201 | 195.160.66.21 | 128 | TCP | 0xbed1 (48849) | 54 | 4619 → 8080 [ACK] S |
| 16 | 0.751334 | 0.081028 | 130.177.80.201 | 195.160.66.21 | 128 | TCP | 0xbed4 (48852) | 54 | 4619 → 8080 [ACK] S |
| 19 | 0.852190 | 0.100856 | 130.177.80.201 | 195.160.66.21 | 128 | TCP | 0xbed7 (48855) | 54 | 4619 → 8080 [ACK] S |
| 22 | 0.852454 | 0.000264 | 130.177.80.201 | 195.160.66.21 | 128 | TCP | 0xbed9 (48857) | 54 | 4619 → 8080 [ACK] S |
| 25 | 0.932994 | 0.080540 | 130.177.80.201 | 195.160.66.21 | 128 | TCP | 0xbeds (48860) | 54 | 4619 → 8080 [ACK] S |

# Troubleshooting using IP and TCP fields

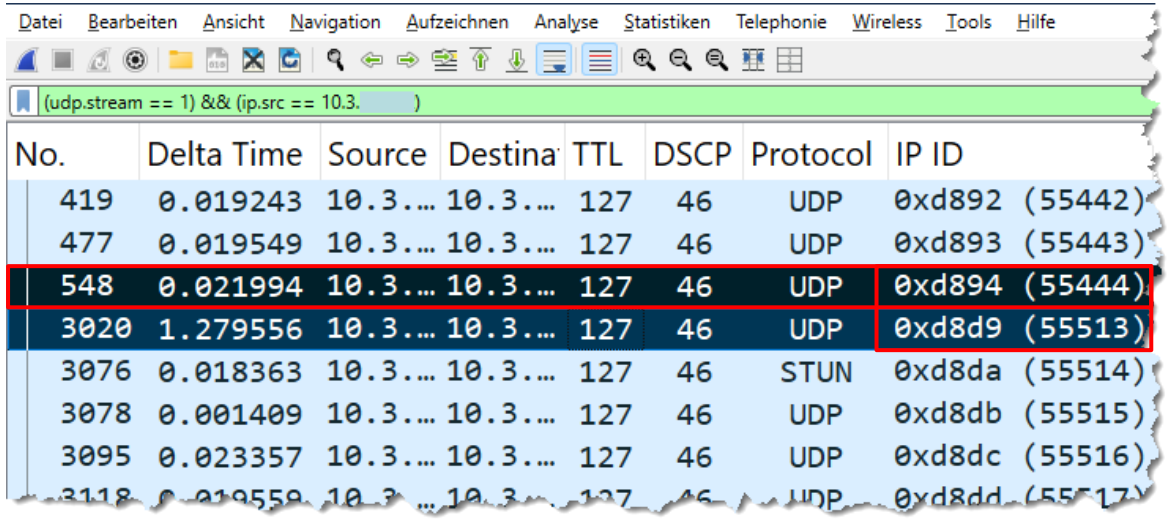
A closer look at the **IP header** (cont):

- With TCP sessions, **lost packet** will be detected by **TCP retransmissions**
- But with **UDP-based** transmissions (e.g. **VoIP**) there are no retransmissions
- Some UDP-based application protocols (e.g. **RTP, SIP**) have a packet number in the header
- But in some application protocols (e.g. **Skype for Business (SfB), Teams**) the UDP payload is encrypted

→ **Use case 2:** A **router is dropping SfP-VoIP packets**, which is visible only by following the **IP ID sequence number**




*Captured in front of the router (all packets visible)*




*Captured after the router (167 packet dropped)*

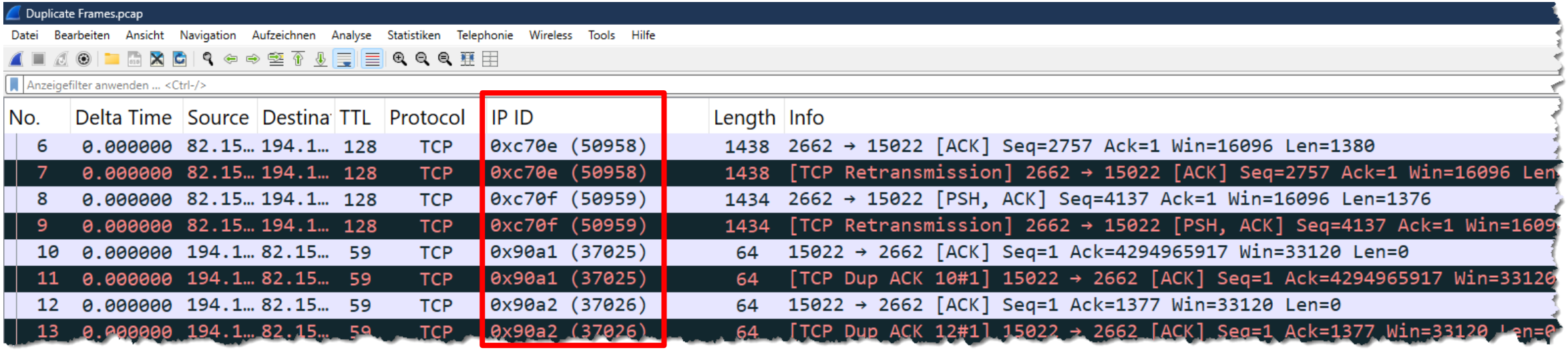
# Troubleshooting using IP and TCP fields



A closer look at the **IP header** (cont):

- It may happen that you see all the **packets twice** in your trace file.
- The **Wireshark Expert** marks these duplicates as **TCP Retransmissions** or **Duplicate Acks**. **Is this correct?**
- **Not always!** In this case: **False positive messages!** The Wireshark looks for repeating TCP Sequence Numbers

→ **Use case 3:** Packets are duplicated by a **network component**. Visible by analyzing the **IP Identification field**




| No. | Delta Time | Source | Destina | TTL | Protocol | IP ID | Length | Info |
|---|---|---|---|---|---|---|---|---|
| 6 | 0.000000 | 82.15… | 194.1… | 128 | TCP | 0xc70e (50958) | 1438 | 2662 → 15022 [ACK] Seq=2757 Ack=1 Win=16096 Len=1380 |
| 7 | 0.000000 | 82.15… | 194.1… | 128 | TCP | 0xc70e (50958) | 1438 | [TCP Retransmission] 2662 → 15022 [ACK] Seq=2757 Ack=1 Win=16096 Len |
| 8 | 0.000000 | 82.15… | 194.1… | 128 | TCP | 0xc70f (50959) | 1434 | 2662 → 15022 [PSH, ACK] Seq=4137 Ack=1 Win=16096 Len=1376 |
| 9 | 0.000000 | 82.15… | 194.1… | 128 | TCP | 0xc70f (50959) | 1434 | [TCP Retransmission] 2662 → 15022 [PSH, ACK] Seq=4137 Ack=1 Win=1609 |
| 10 | 0.000000 | 194.1… | 82.15… | 59 | TCP | 0x90a1 (37025) | 64 | 15022 → 2662 [ACK] Seq=1 Ack=4294965917 Win=33120 Len=0 |
| 11 | 0.000000 | 194.1… | 82.15… | 59 | TCP | 0x90a1 (37025) | 64 | [TCP Dup ACK 10#1] 15022 → 2662 [ACK] Seq=1 Ack=4294965917 Win=33120 |
| 12 | 0.000000 | 194.1… | 82.15… | 59 | TCP | 0x90a2 (37026) | 64 | 15022 → 2662 [ACK] Seq=1 Ack=1377 Win=33120 Len=0 |
| 13 | 0.000000 | 194.1… | 82.15… | 59 | TCP | 0x90a2 (37026) | 64 | [TCP Dup ACK 12#1] 15022 → 2662 [ACK] Seq=1 Ack=1377 Win=33120 Len=0 |

→ Currently, the Wireshark Expert **does not make a relationship between the TCP and IP headers**
→ To remove duplicate frames use: **editcap -d duplicate_frames.pcapng no_duplicate_frames.pcapng**

# Troubleshooting using IP and TCP fields

A closer look at the **TCP header**:

- TCP is still the **most valuable protocol** for network troubleshooting
- **Packet loss or any other irregularities** are detected by TCP
- In addition, the **Wireshark Expert** is highly advanced and a valuable assistant in narrowing down problems

- There are just **four basic parameters** that determine the flow of a TCP session (also called **Stream**):

➢ **Sequence Number:**
- Each frame contains a unique **4-byte number**, starting with a **random value,** generated by the sender
- This number is increased by **number of transmitted payload bytes** (counting transmitted bytes, not packets)

➢ **Acknowledge Number:**
- This **ACK 4-byte number** from the receiver, **points to the next expected byte** (not the last byte received)
- With one ACK, **multiple received packet** can be acknowledged (ACK = 5000 → received all bytes up to 4999)

➢ **Window Size:**
- Is the **size of the receive buffer**. TCP has sophisticated **Flow Control;** flooding of the receiver never occurs!
- In **each packet**, the receiver regularly informs the sender of the size of the **remaining input buffer** (in bytes)

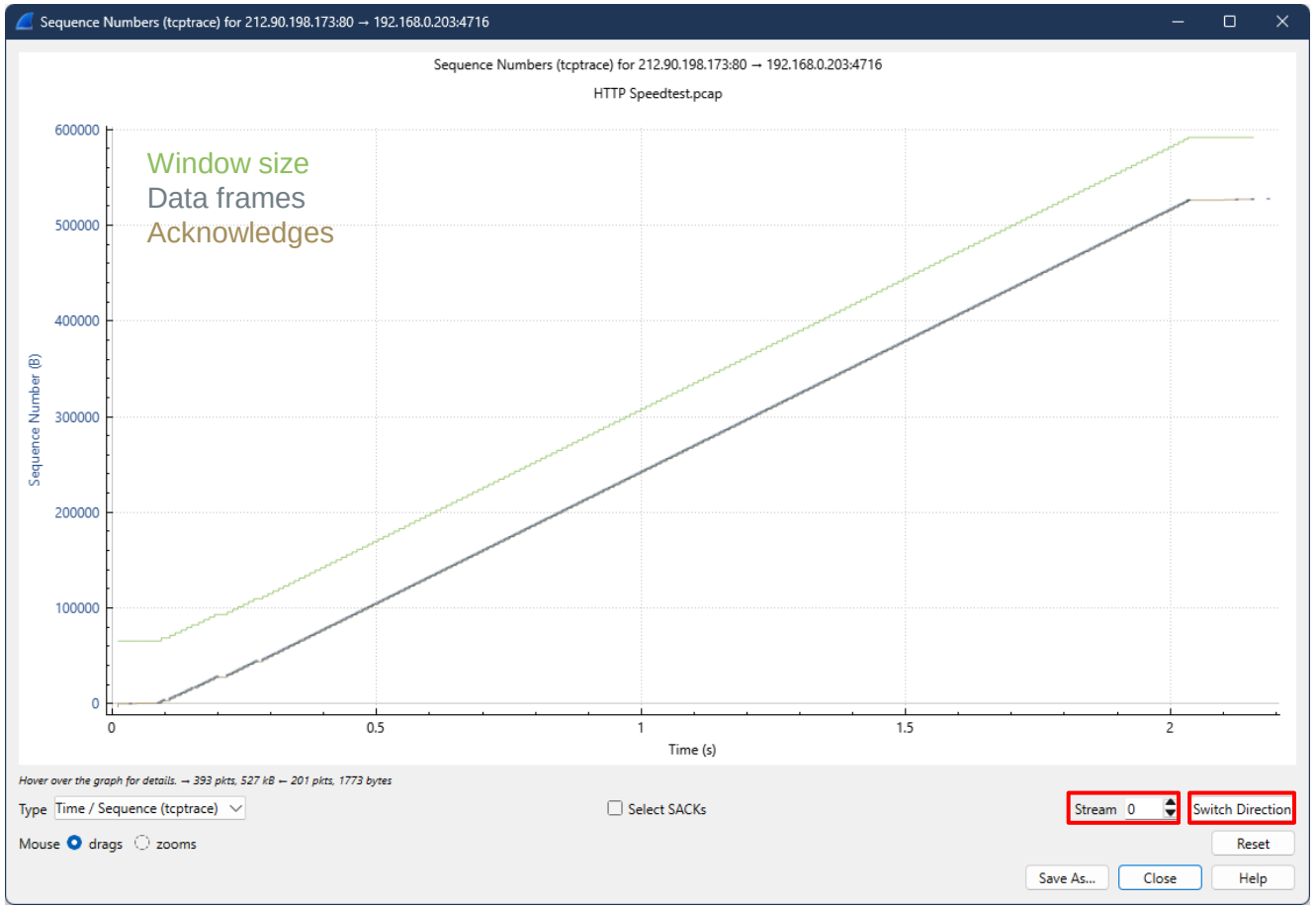
➢ **Timing:**
- For an efficient data flow, **propagation delay is a critical value**, TCP can handle a wide range of delays
- The round-trip time (RTT) depends on **the physical distance and devices** between transmitter and receiver
- With a **large window size**, long RTTs can be compensated so that TCP can **transmit continuously**

→ **Wireshark displays all four parameters and their relationships in the TCP Stream Graph** (my favorite feature)

# Troubleshooting using IP and TCP fields

A closer look at the **TCP header** (cont):

- To create the graph, open a TCP trace file. **Go to → Statistics → TCP Stream Graphs → Time Sequence (tcptrace)**




- A TCP stream is a **full-duplex communication**
- It consists of **two** independent **Half Sessions**
- **One in each** communication direction
- The graph is showing **one Half Session** at a time

- Use **Switch Direction** to change the direction
- Use **Stream** button to show next TCP stream

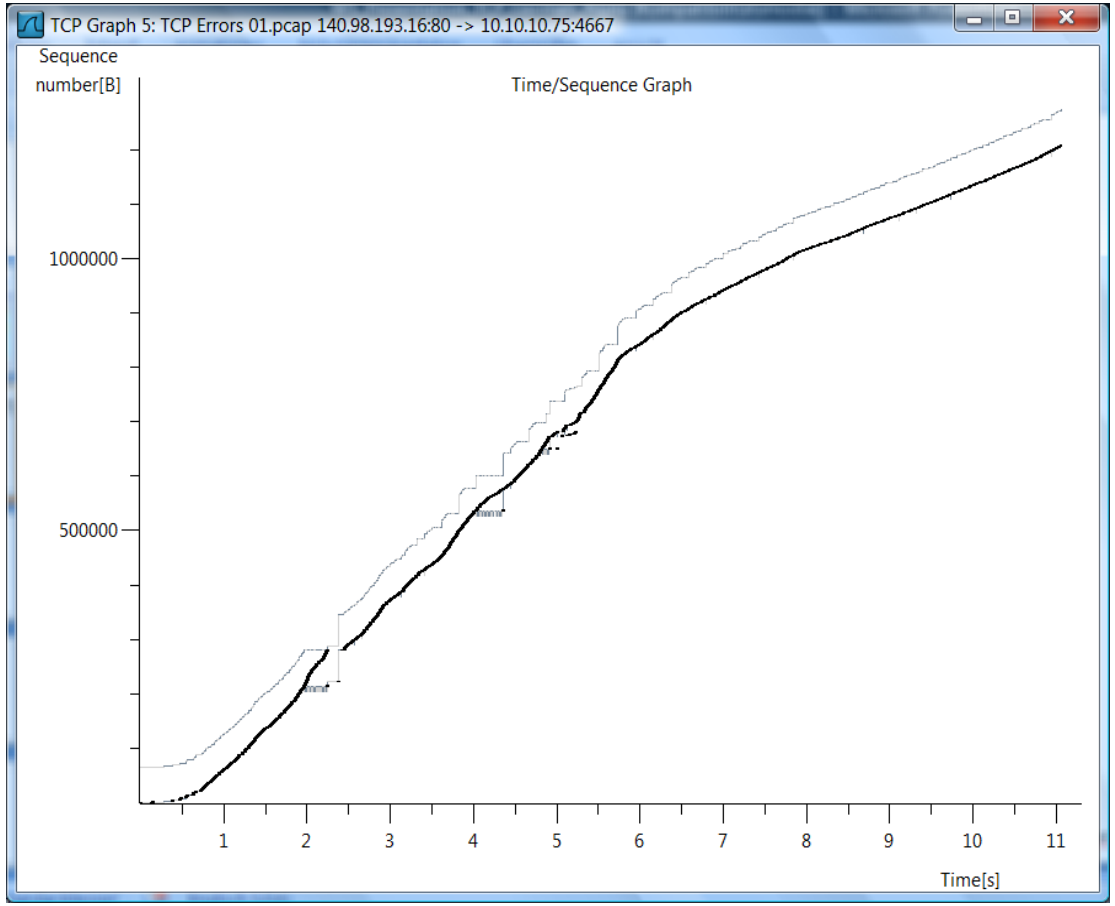
- **Mouse Wheel** to zoom in & out
- **Right mouse** to show navigation details
- **Space Bar** to show crosshair pointer

→ Live demonstration: HTTP Speedtest.pcap

# Troubleshooting using IP and TCP fields

A closer look at the **TCP header** (cont):

→ **Use case 1:** Understanding the diagram will help you identify most of the critical TCP situations




- Lost Frames
- Duplicate Frames
- Out of order Frames
- TCP Sequence number and Segment Sizes
- Acknowledges, Delayed Acknowledges
- Duplicate and Selective Acknowledges
- Retransmissions and Fast Retransmissions
- Windows Sizes, sliding Window
- Frozen Windows Size
- Zero Window and Window Full Situation
- Window Scaling,
- Slow Start, full Flow rate and Flow throttling

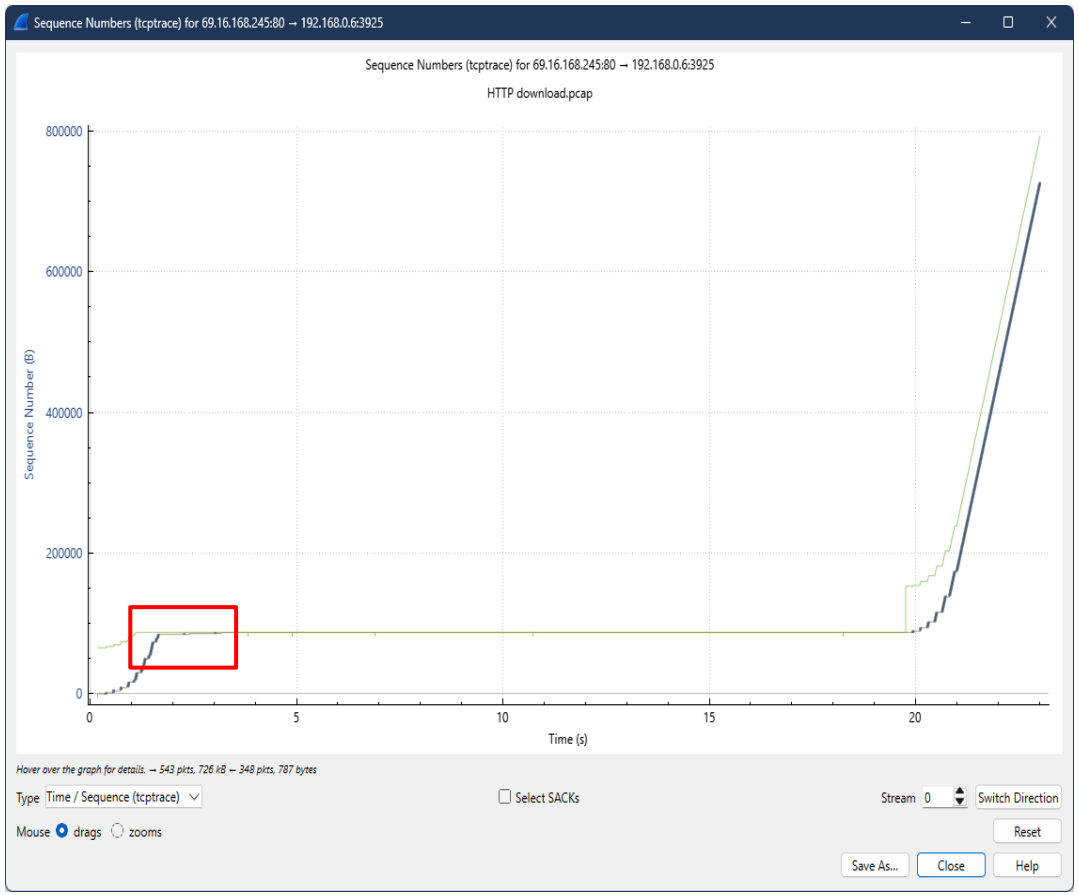
→ Live demonstration: TCP Errors 01.pcap

→ **Conclusion: The network drops packets, but without any significant impact on throughput**

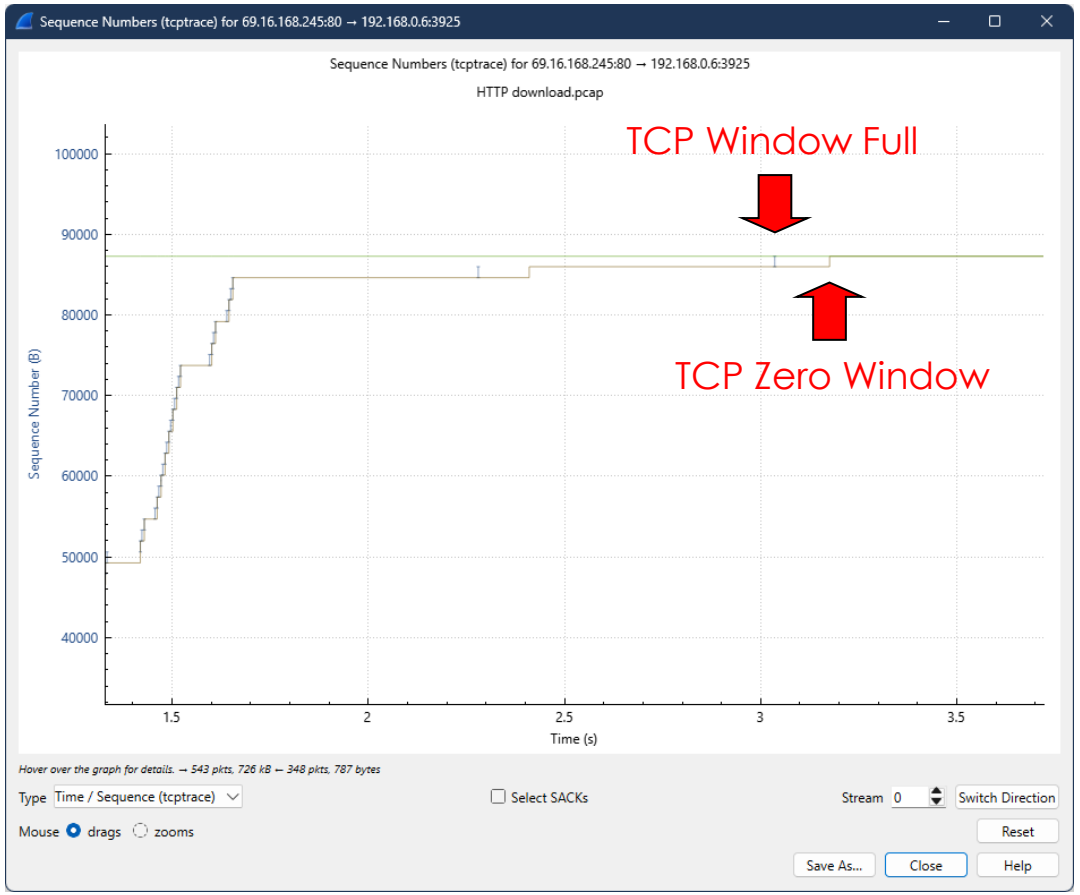
# Troubleshooting using IP and TCP fields

A closer look at the **TCP header** (cont):

→ **Use case 2:** Transmission **blocked by the receiver**. Wireshark messages **TCP Windows Full** and **TCP Zero Window**




**zoom in →**




*Wireshark Expert Messages*

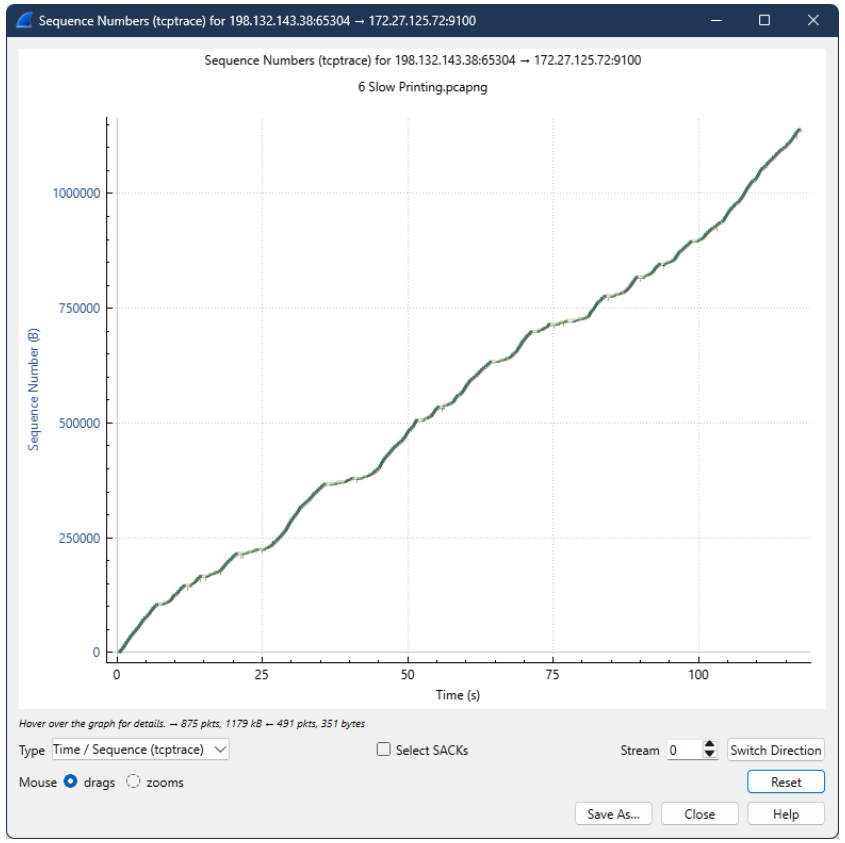
+

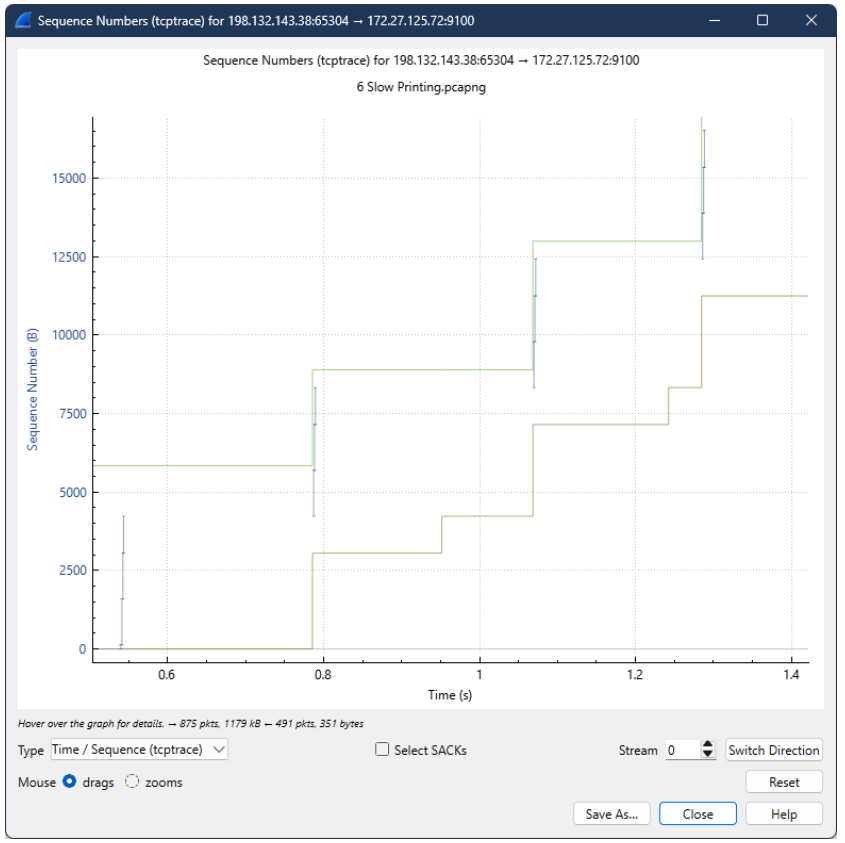
# Troubleshooting using IP and TCP fields

A closer look at the **TCP header** (cont):

- A printer has been installed in a remote warehouse for printing delivery notes, sent from the head office SAP server

→ **Use case 3:** The staff complains that it **takes several minutes** to print a delivery note.




- Printer was tested before at **head office** and was working perfect!
- A bandwidth problem?
- Bandwidth has already been increased from **500 kbps to 1Mbps**
- The Wireshark Expert does not show **any relevant anomalies**
- Only ~ 2% TCP retransmission
- No 'window zero' or 'window full' symptoms

→ Live demo: Slow Printing.pcap

**Conclusion:**

- The **window size** of the printer is too small and limits the speed

- To calculate the minimum required window size, use a BDP calculator: TCP Bandwidth Delay Product calculator

+

# The challenge of Network Troubleshooting

Difficult tasks are often referred to as looking for a **needle in a haystack.**

**That's easy!**

**Use a strong magnet!**

But **Network Troubleshooting** is finding **the needle in a needle stack.**

**That's challenging!**

**Learn how to use**

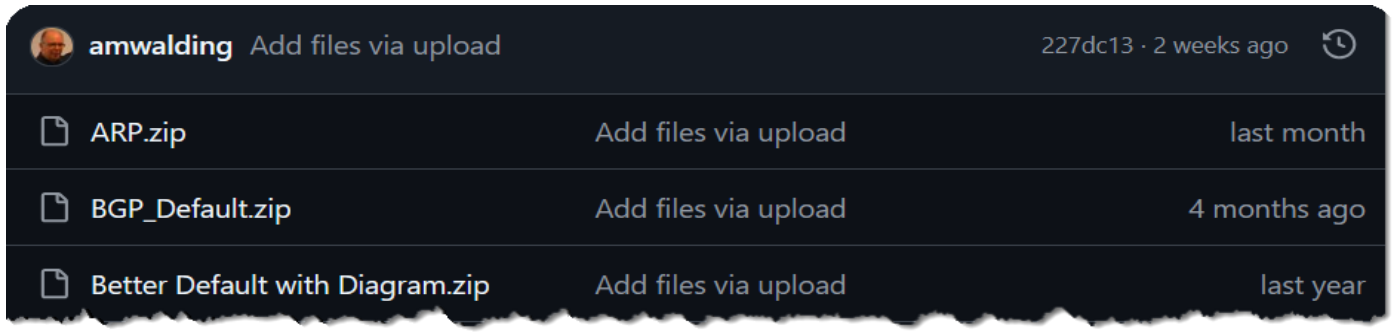
WIRESHARK

# Ingenious but little-known Wireshark functions

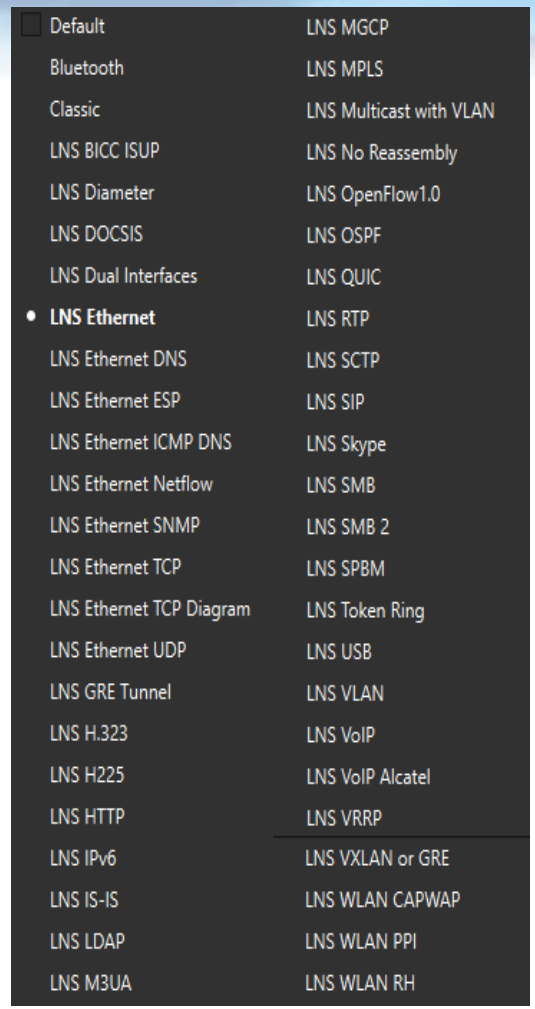
## Wireshark Profiles:

- One of the **most useful capability** of Wireshark to speed-up troubleshooting
- A convenient way to store your **personalized** Wireshark GUI
- In a profile, you can have your **colums, colors, filters and many other settings**
- No need to save, all your changes are **stored automatically** to the active profile
- Profiles can be **exported** to use them on other computers
- Profiles can be **imported** from other PCs or downloaded from the Internet

**Andrew Walding** from CellStream, Inc. has created dozens of Wireshark profiles:
→ https://github.com/amwalding/wireshark_profiles




**Walter Hofstetter** from AnyWeb has explained the most useful **Profiles** in a document.
→ Available after the Packet Fest on https://www.ntop.org/

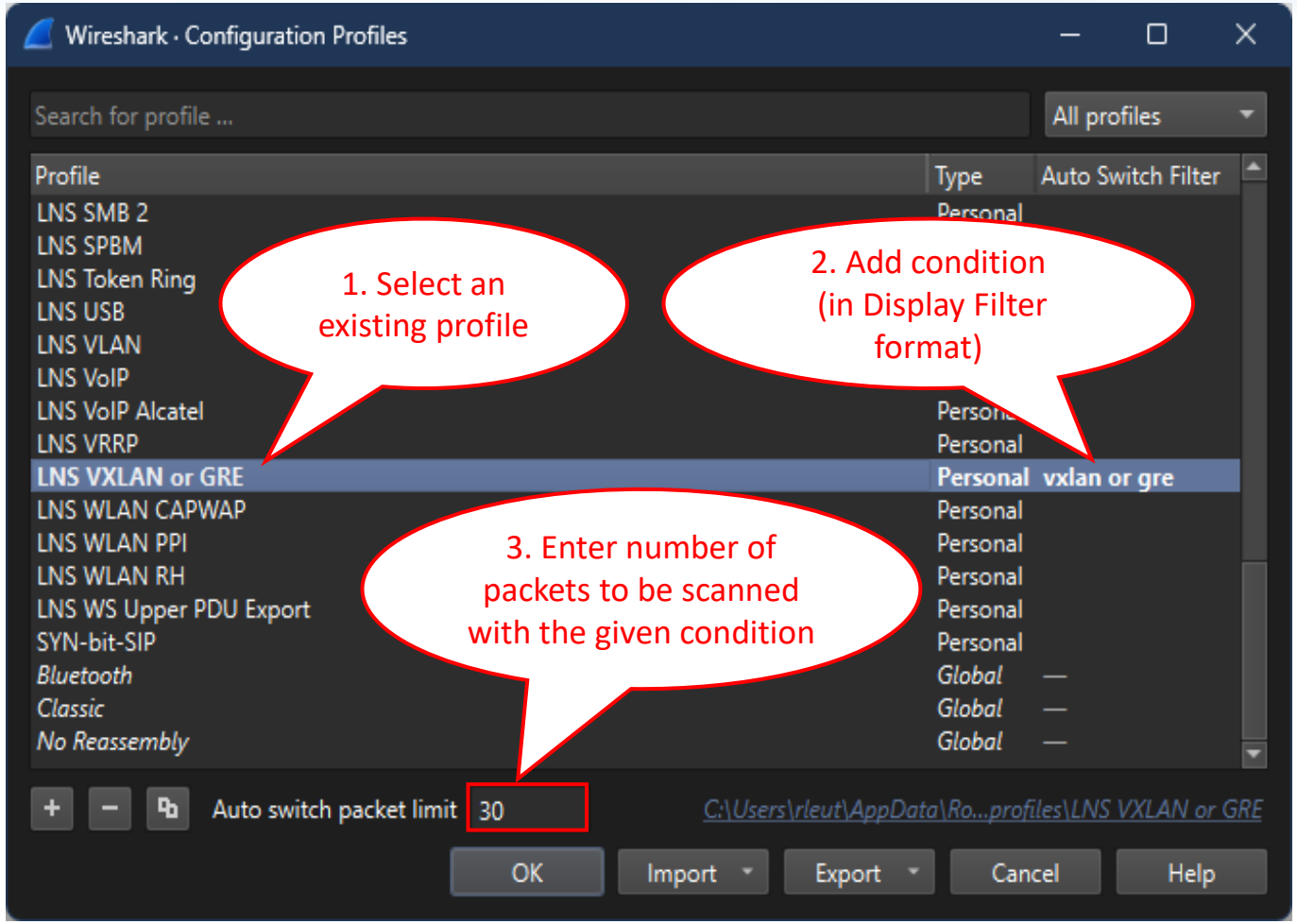



*My Wireshark Profiles*

# Ingenious but little-known Wireshark functions



**Wireshark Profiles** (cont):




*Manage Profiles panel*

- Wireshark offers a new, fancy feature: **Automatic Profile Switching**
- This allows a profile to be **automatically activated** when a trace file is opened

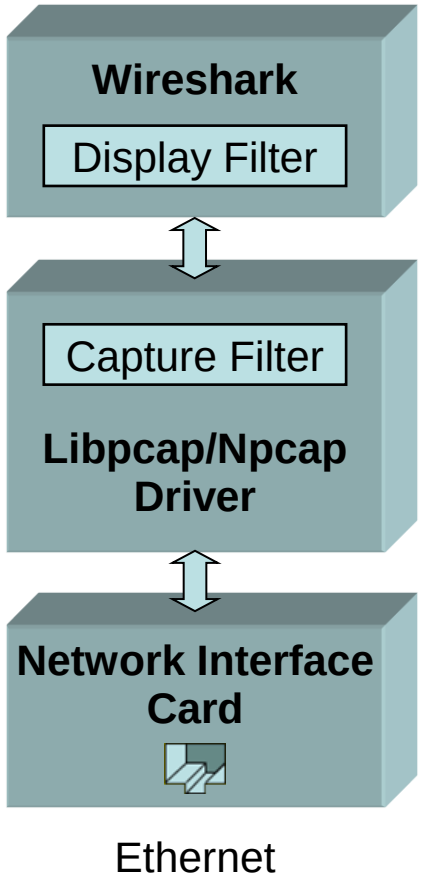
1. Select an **existing profile** or create a **new one**
2. Add **Auto Switch Filter**. Use any **Display Filter** as condition
3. Select **how many packets** in a trace file are **searched** for the condition.

- IIn this example, when a trace file is opened, the **first 30 packets** are scanned for the **vxlan or gre** condition
- If one of **these first packets** matches the condition, the **profile is activated**

→ Live demo: VXLAN Ping Start.pcapng

# Ingenious but little-known Wireshark functions

## Wireshark Filters:




Ethernet

### Capture Filter:

ether host 00:11:95:b7:e0:3e
ether src 00:11:95:b7:e0:3e
ether dst 00:11:95:b7:e0:3e

host 192.168.178.1
src host 192.168.178.1
dst host 192.168.178.1

net 192.168.178.0 mask 255.255.255.0
net 192.168.178/24

arp
not ip
tcp
udp port 138

Operators: not, and, or

http://wiki.wireshark.org/CaptureFilters
or Google: tcpdump filters

### Display Filter:

eth.addr == 00:11:95:b7:e0:3e
eth.src == 00:11:95:b7:e0:3e
eth.dst == 01:00:5e:00:00:09

ip.addr == 192.168.178.1
ip.src == 192.168.178.1
ip.dst == 192.168.178.1

ip.addr == 192.168.178.0/24
ip.host contains "192.168.178"

arp
not ip
tcp
udp.port == 138

Operators: not, and, or

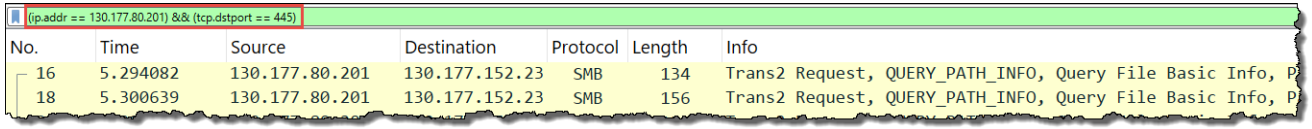
http://wiki.wireshark.org/DisplayFilters

# Ingenious but little-known Wireshark functions

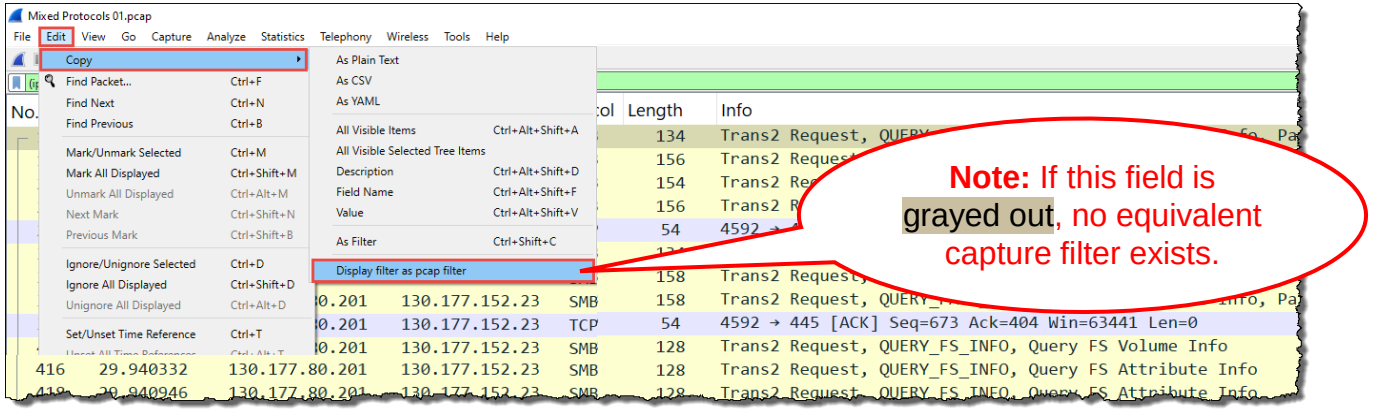


## **Wireshark Filters** (cont):

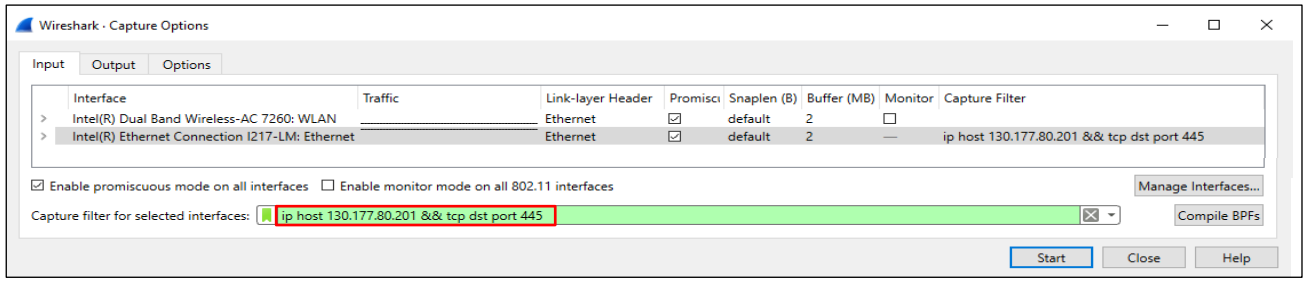
- Example Display Filter: (ip.addr == 130.177.80.201) && (tcp.dstport == 445)




- Go to → Edit → Copy → **Display filter as pcap filter**




- Go to → Capture → Options → select Interface and paste the copied filter




- Wireshark offers a new filter feature: **Display to Capture Filter Translator**
- It translates the **Display Filter syntax** to the **Capture Filter syntax**
- Wireshark at present supports ~3'000 protocols and **~250'000 Display Filters**
- **Capture filters** are based on **TCPdump** and offers **< 1'000 filters**

→ Therefore, **not all display filters** can be translated to capture filter

LEUTERT NetServices

# Hope you learned something useful

Our **public courses** are held at:

STUDERUS
network technologies
Basic Course in Schwerzenbach

**AnyWeb Training**
Cisco Certified Learning Center
Advanced Courses in Zürich Oerlikon

Ask for a **company course** or register for the newsletter:

LEUTERT NetServices
www.netsniffing.ch


## Thank you for your attention