

---

# **Praktikum Rechnernetze**

Protokoll zu Versuch 2 (Protokollanalyse mit Wireshark)  
von Gruppe 1

Jakob Waibel, Daniel Hiller, Elia Wüstner, Felix Pojtinger

2021-10-19

## Inhaltsverzeichnis

<b>1 Einführung</b>	<b>3</b>
1.1 Mitwirken . . . . .	3
1.2 Lizenz . . . . .	3
<b>2 Wireshark</b>	<b>4</b>
2.1 Einführung . . . . .	4
2.2 Ping . . . . .	6
2.3 DHCP . . . . .	7
2.4 DNS . . . . .	9
2.5 ARP . . . . .	10
2.6 Layer-2-Protokolle . . . . .	12
2.7 HTTP und TCP . . . . .	13
2.8 MAC . . . . .	17
2.9 STP . . . . .	19
2.10 SNMP . . . . .	20
2.11 Streaming and Downloads . . . . .	20
2.12 Telnet und SSH . . . . .	22
2.13 Wireshark-Filter . . . . .	23

## 1 Einführung

### 1.1 Mitwirken

Diese Materialien basieren auf Professor Kiefers “Praktikum Rechnernetze”-Vorlesung der HdM Stuttgart.

**Sie haben einen Fehler gefunden oder haben einen Verbesserungsvorschlag?** Bitte eröffnen Sie ein Issue auf GitHub ([github.com/pojntfx/uni-netpractice-notes](https://github.com/pojntfx/uni-netpractice-notes)):



**Abbildung 1:** QR-Code zum Quelltext auf GitHub

Wenn Ihnen die Materialien gefallen, würden wir uns über einen GitHub-Stern sehr freuen.

### 1.2 Lizenz

Dieses Dokument und der enthaltene Quelltext ist freie Kultur bzw. freie Software.



**Abbildung 2:** Badge der AGPL-3.0-Lizenz

Uni Network Practice Notes (c) 2021 Jakob Waibel, Daniel Hiller, Elia Wüstner, Felix Pojtinger

SPDX-License-Identifier: AGPL-3.0

## 2 Wireshark

### 2.1 Einführung

**An welchem Koppelement im Systemschrank sollte der Hardware-Analysator/Netzwerk-Sniffer sinnvollerweise angeschlossen werden und warum? Welche grundsätzlichen Möglichkeiten gibt es noch?**

- Switch, damit Nachrichten auf Layer 2 auch abgefangen werden können
- Grundsätzlich könnte, vor allem auch in Heimnetzwerken, der Router hierzu verwendet werden, da hier oft Router und Switch zu einem Gerät kombiniert sind.

**Starten Sie Wireshark und capturern Sie den aktuellen Traffic. Dokumentieren Sie zunächst, was alles auf Wireshark einprasselt.**



**Abbildung 3:** Screenshot von Wireshark

Zu erkennen sind Pakete von mehreren Protokollen:

- LLDP
- Spanning-Tree-Protokoll (STP)
- DNS
- TCP

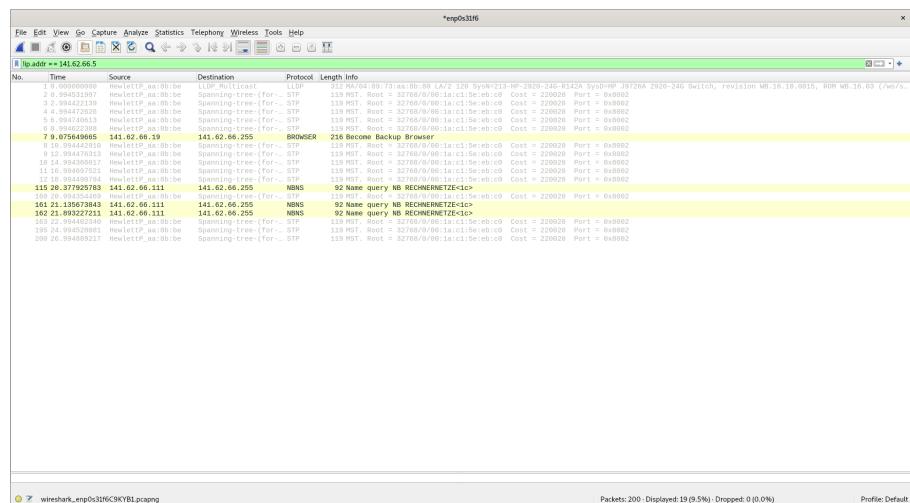
- HTTP

Die letzten beiden Protokolle (TCP, HTTP) lassen sich durch das Öffnen des Browsers erklären.

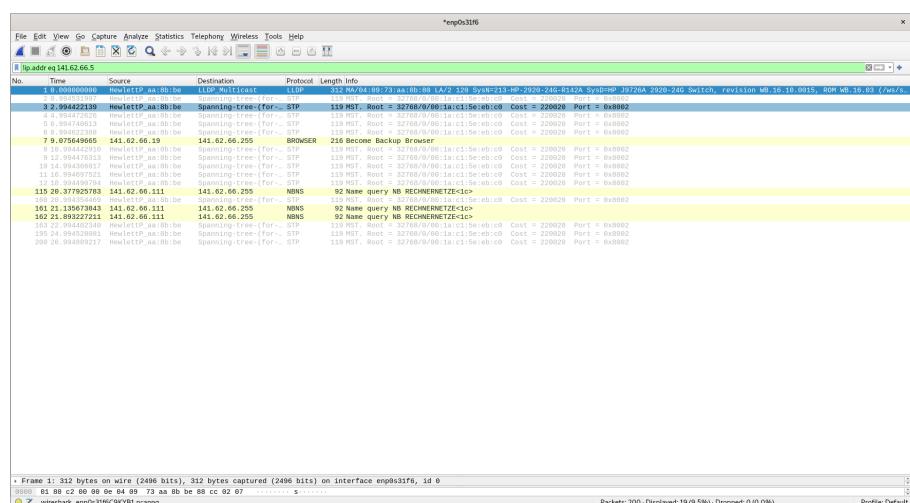
### Wie lautet der Filter, mit dem Sie ihre eigene Verbindung ins Labor ausklammern? Welche Möglichkeiten gibt es?

Hierzu gibt es mehrere Optionen:

```
1 !ip.addr == 141.62.66.5
2 not ip.addr == 141.62.66.5
3 !ip.addr eq 141.62.66.5
```



**Abbildung 4:** Ausklammern der eig. IP, Option 1



**Abbildung 5:** Ausklammern der eig. IP, Option 2

## 2.2 Ping

**Senden Sie einen Ping zu nachfolgenden Empfängern und zeichnen Sie die entsprechenden Protokolle gezielt mit Wireshark auf. Vergleichen Sie die Protokollabläufe: wer sendet welches Protokoll warum an wen? Pingen Sie an ....**

**Einen Rechner Ihrer Wahl im Labornetz:**



**Abbildung 6:** Wireshark-Output zu einem Rechner im Labornetz

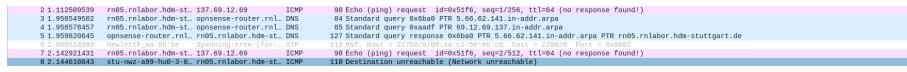
## Einen beliebigen Server im Internet (Google)

Wir haben hierzu die Namensauflösung aktiviert, damit die IPs zur Domain [google.com](http://google.com) zugeordnet werden können.



**Abbildung 7:** Wireshark-Output zu einem Ping nach [google.com](http://google.com)

### Eine beliebige nicht existierenden IP-Adresse



**Abbildung 8:** Wireshark-Output zu einem Ping nach 137.69.12.69

### 2.3 DHCP

**Analysieren Sie die Abläufe bei DHCP (im Labor installiert). Ihre Teilgruppe am Nachbartisch bootet den PC am Arbeitsplatz, protokollieren Sie die DHCP-Abläufe sowie sonstigen Netzverkehr, den der PC bis zum Erhalt der IP-Adresse erzeugt.**

Während des Startens werden drei DHCP-Requests für verschiedene Komponenten abgehandelt.



**Abbildung 9:** Gesamter Bootprozess



**Abbildung 10:** Bootprozess: DHCP-Requests des BIOS zum Netzwerkboot, damit der Netzwerbootloader über i.e. TFTP geladen werden kann



**Abbildung 11:** Bootprozess: DHCP-Requests des Netzwerbootloaders iPXE

## Strukturieren Sie die DHCP-Abläufe und beschreiben Sie, wie DHCP im Detail funktioniert.

Durch Booten des PCs wird dem Rechner mittels DHCP eine IP zugewiesen. Ergänzend kommen noch Standard-Gateway-Adresse und DNS Adresse hinzu. DHCP ermöglicht damit erst, dass verschiedene Rechner in einem Netzwerk kommunizieren können, da dafür jeder Computer eine eigene IP benötigt.

Grundlegend funktioniert DHCP mithilfe von vier Nachrichtentypen. Es gibt den DHCP-Discover, welcher den DHCP-Server in erster Linie benachrichtigen will, dass eine neue IP verlangt wird. Der Server antwortet daraufhin mit einer Offer, welche eine IP reserviert und diese dem Client anbietet. Außerdem

enthält die Offer die IP des DHCP-Servers, die Subnetzmaske und die Lease-Time. Danach kann der Client mit einer DHCP-Request die angebotene IP anfordern. Wenn das in Ordnung ist, antwortet der DHCP-Server mit einem DHCP-Acknowledge.

### Vergleicht Sie den Ablauf, wenn Sie den DHCP-Ablauf per ipconfig /release und ipconfig /renew initiiieren

Mittels der folgenden Commands wurde eine IP-Adresse freigegeben und eine neue angefordert.

```
1 # dhclient -r # Release der IP-Adresse
2 # dhclient # Anfrage einer neuen IP-Adresse
```

No.	Time	Source	Destination	Protocol	Length Info
1	19.15.392845861	0.0.0.0	255.255.255.255	DHCP	342 DHCP Discover - Transaction ID 0x70ef81d
2	20.15.39351726	0.0.0.0	255.255.255.255	DHCP	342 DHCP Request - Transaction ID 0x70ef81d
21	15.408801806	linux.local	Broadcast	ARP	68 Who has 141.62.66.250? Tell 141.62.66.4

Dem bereits hochgefahrenen Rechner wird eine neue IP zugeordnet. Wenn wir die IP Zuweisung auf diese weise neu initiieren dann ist der DHCP Ablauf deutlich kürzer, da beim Booten unter der Haube noch deutlich mehr gemacht werden muss (es muss e.g. keine DHCP-Request des BIOS zum Netzwerkboot getätigigt werden).

## 2.4 DNS

### Dokumentieren Sie den Ablauf bei einer DNS-Abfrage

#### Fall 1: DNS-Server 141.62.66.250:

Mittels folgendem Command wurde eine DNS-Abfrage gemacht:

```
1 $ dig @141.62.66.250 google.com
2 google.com. 163 IN A 142.250.186.174
```

No.	Time	Source	Destination	Protocol	Length Info
11	1.357358000	rn05.rnlabor.hdm-st.	opnsense-router.rnl...	DNS	93 Standard query 0xa276 A google.com OPT
12	1.371692878	opnsense-router.rnl...	rn05.rnlabor.hdm-st.	DNS	97 Standard query response 0xa276 A google.com A 142.250.186.174 OPT

**Abbildung 12:** Ablauf der Anfrage

Hier nutzten wir den internen DNS Server und machen eine Anfrage auf [google.com](http://google.com).

#### Fall 2: DNS-Server 1.1.1.1 (Cloudflare):

Mittels folgendem Command wurde eine DNS-Abfrage gemacht:

```
1 $ dig @1.1.1.1 +noall +answer google.com
2 google.com. 231 IN A 142.250.185.110
```

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	rn05.rnlabor.hdm-st..	one.one.one.one	DNS	93	Standard query 0x6247 A google.com OPT
2	1.285807890	rn05.rnlabor.hdm-st..	opnsense-router.hdm-st..	DNS	84	Standard query 0xd2b PTR 5.66.62.141.in-addr.arpa
5	1.285849397	rn05.rnlabor.hdm-st..	opnsense-router.rnl..	DNS	88	Standard query 0x8883 PTR 1.1.1.1.in-addr.arpa
6	1.297179251	opnsense-router.rnl..	rn05.rnlabor.hdm-st..	DNS	127	Standard query response 0xd2b PTR 5.66.62.141.in-addr.arpa PTR rn05.rnlabor.hdm-stuttgart.de
7	1.297611338	opnsense-router.rnl..	rn05.rnlabor.hdm-st..	DNS	109	Standard query response 0x8883 PTR 1.1.1.1.in-addr.arpa PTR one.one.one.one

**Abbildung 13:** Ablauf der Anfrage

Bei der DNS Anfrage über Cloudflare erscheinen weitere DNS-Requests über DNS Reverse-Zones. Dies wird daran liegen, dass wir über den Router mit dem Internet kommunizieren.

### Fall 3: DNS-Server 8.8.8.9 (DNS-Dienst ist dort nicht installiert):

Mittels folgendem Command wurde eine DNS-Abfrage gemacht:

```
1 $ dig @8.8.8.9 +noall +answer google.com
2 ;; connection timed out; no servers could be reached
```

No.	Time	Source	Destination	Protocol	Length	Info
3	0.572498372	rn05.rnlabor.hdm-st..	8.8.8.9	DNS	93	Standard query 0x73f9 A google.com OPT
5	1.088436116	rn05.rnlabor.hdm-st..	opnsense.rnlabor.hdm-st..	DNS	84	Standard query 0xcedb PTR 5.66.62.141.in-addr.arpa
6	1.088436116	rn05.rnlabor.hdm-st..	opnsense.rnlabor.hdm-st..	DNS	88	Standard query 0x8883 PTR 1.1.1.1.in-addr.arpa
7	1.089061823	opnsense.rnlabor.hdm-st..	rn05.rnlabor.hdm-st..	DNS	127	Standard query response 0xd2b PTR 5.66.62.141.in-addr.arpa PTR rn05.rnlabor.hdm-stuttgart.de
8	1.090026625	opnsense.rnlabor.hdm-st..	rn05.rnlabor.hdm-st..	DNS	148	Standard query response 0x74b6 No such name PTR 9.8.8.8.in-addr.arpa SOA ns1.google.com
13	2.087996807	rn05.rnlabor.hdm-st..	opnsense.rnlabor.hdm-st..	DNS	84	Standard query 0x4fb6 PTR 250.66.62.141.in-addr.arpa
17	2.089268013	opnsense.rnlabor.hdm-st..	rn05.rnlabor.hdm-st..	DNS	163	Standard query response 0x4fb6 PTR 250.66.62.141.in-addr.arpa PTR opnsense-router.rnlabor.hdm-stuttgart.de PTR opnsense.rnlabor.hdm-st..
22	2.089268013	opnsense.rnlabor.hdm-st..	opnsense.rnlabor.hdm-st..	DNS	84	Standard query 0x4fb6 PTR 250.66.62.141.in-addr.arpa
23	3.087954583	rn05.rnlabor.hdm-st..	opnsense.rnlabor.hdm-st..	DNS	84	Standard query 0x4fb6 PTR 250.66.62.141.in-addr.arpa
24	3.087959318	rn05.rnlabor.hdm-st..	opnsense.rnlabor.hdm-st..	DNS	88	Standard query 0x1f24 PTR 255.255.254.169.in-addr.arpa
25	3.088893145	opnsense.rnlabor.hdm-st..	rn05.rnlabor.hdm-st..	DNS	145	Standard query response 0x59b No such name PTR 19.75.254.169.in-addr.arpa SOA localhost
26	3.089011764	opnsense.rnlabor.hdm-st..	rn05.rnlabor.hdm-st..	DNS	141	Standard query response 0xfcfd No such name PTR 251.0.0.224.in-addr.arpa SOA sns.dns.icann.org
27	3.089125772	opnsense.rnlabor.hdm-st..	rn05.rnlabor.hdm-st..	DNS	147	Standard query response 0x1f24 No such name PTR 255.255.254.169.in-addr.arpa SOA localhost

**Abbildung 14:** Ablauf der Anfrage

Wie im Bild zu sehen ist, bekommen wir den Response **No such name PTR 9.8.8.8.**

### Wie erkennen Sie mit Wireshark, dass “versehentlich” ein falscher DNS-Server eingetragen wurde?

Es gibt eine Antwort, welche auf eine nicht gültige IP-Adresse hinweist (Siehe oben).

## 2.5 ARP

### Lösen Sie eine ARP-Anfrage aus und protokollieren Sie die Datenpakete.

Hierzu wurde ein Rechner, welcher zuvor nicht im lokalen ARP-Cache war, neu gestartet.

No.	Time	Source	Destination	Protocol	Length	Info
214	110.515578213	linux-2.local	Broadcast	ARP	42	Who has 141.62.66.6? Tell 141.62.66.5
215	110.5155867298	linux-3.local	linux-2.local	ARP	68	141.62.66.6 is at 4c:52:0e:54:2b
231	115.673164735	linux-3.local	linux-2.local	ARP	68	Who has 141.62.66.5? Tell 141.62.66.6
262	116.673186703	linux-2.local	linux-3.local	ARP	42	141.62.66.5 is at 4c:52:0e:54:8b

**Abbildung 15:** Ablauf der Anfrage

### Wann wird eine ARP-Anfrage gestartet?

Sobald ein Paket an die Zieladresse (in unserem Fall 141.62.66.6) gesendet werden soll, wird eine ARP-Anfrage in Form eines Broadcasts gestartet, um das Zielgerät im Netzwerk zu ermitteln, sofern sich diese nicht bereits im ARP-Cache befindet. Dieser kann mit `ip neigh show` ausgelesen werden. Mit `ip neigh flush all` kann der ARP-Cache geleert werden.

### Welcher Rahmentyp wird für die Anfrage verwendet?

Als Rahmentyp wird Ethernet II verwendet.

No.	Time	Source	Destination	Protocol	Length	Info
214	110.515578213	linux-2.local	Broadcast	ARP	42	Who has 141.62.66.6? Tell 141.62.66.5
215	110.5155807208	linux-3.local	Linux-2.local	ARP	68	141.62.66.6 is at 4c:52:62:0e:54:2b
231	115.673164735	linux-3.local	Linux-2.local	ARP	68	Who has 141.62.66.5? Tell 141.62.66.6
232	115.673186793	linux-2.local	Linux-3.local	ARP	42	141.62.66.5 is at 4c:52:62:0e:54:8b

Frame 234: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface enp0s31f6, id 0  
Ethernet II, Src: Linux-2.local (4c:52:62:0e:54:8b), Dst: Broadcast (ff:ff:ff:ff:ff:ff)  
 » Destination: Broadcast (ff:ff:ff:ff:ff:ff)  
 » Source: Linux-2.local (4c:52:62:0e:54:8b)  
 Type: ARP (0x0806)  
 » Address Resolution Protocol (request)

**Abbildung 16:** Verwendetes Ethernet-Frame

### Beobachten Sie die Veränderung in der ARP-Tabelle Ihres Rechners

Zuvor:

```
1 $ ip neigh show
2 141.62.66.6 dev enp0s31f6 lladdr 4c:52:62:0e:54:2b STALE
3 141.62.66.250 dev enp0s31f6 lladdr 00:0d:b9:4f:b8:14 STALE
4 141.62.66.13 dev enp0s31f6 lladdr 4c:52:62:0e:54:5d STALE
5 141.62.66.236 dev enp0s31f6 lladdr 26:c5:04:8a:fa:eb STALE
```

Danach:

```
1 $ ip neigh show
2 141.62.66.6 dev enp0s31f6 lladdr 4c:52:62:0e:54:2b STALE
3 141.62.66.250 dev enp0s31f6 lladdr 00:0d:b9:4f:b8:14 STALE
4 141.62.66.4 dev enp0s31f6 lladdr 4c:52:62:0e:53:eb STALE
5 141.62.66.13 dev enp0s31f6 lladdr 4c:52:62:0e:54:5d STALE
6 141.62.66.236 dev enp0s31f6 lladdr 26:c5:04:8a:fa:eb STALE
```



**Wie sieht es mit UPnP im Labor aus? Auf welchen Maschinen von welchem Hersteller läuft der Dienst? Mit welchem Wireshark-Filter „fischen“ Sie den Traffic heraus?**

Es existiert ein Gerät von AVMAudio im Netzwerk, welches über UPnP angesteuert wird. Dies wird immer von demselben Gerät angesteuert, welches über eine Link-Lokale Adresse verfügt, was dafür sorgt, dass es nur innerhalb des Netzwerkes erreicht werden kann. Diese Adressen werden nicht geroutet, sprich die Geräte müssen durch einen Switch etc. verbunden sein. Es kann über den Display-Filter „herausgefischt werden“, indem man nach SSDP filtert.

No.	Time	Source	Destination	Protocol	Length	Info
827	235.115878419	fe80::5e49:79ff:fe6...ff02::c	SSDP	375	NOTIFY * HTTP/1.1	
828	235.115520628	fe80::5e49:79ff:fe6...ff02::c	SSDP	411	NOTIFY * HTTP/1.1	
829	235.117651013	fe80::5e49:79ff:fe6...ff02::c	SSDP	411	NOTIFY * HTTP/1.1	
839	249.109859521	fe80::5e49:79ff:fe6...ff02::c	SSDP	363	NOTIFY * HTTP/1.1	
840	249.110985952	fe80::5e49:79ff:fe6...ff02::c	SSDP	372	NOTIFY * HTTP/1.1	
841	249.1109442125	fe80::5e49:79ff:fe6...ff02::c	SSDP	435	NOTIFY * HTTP/1.1	
842	249.113785421	fe80::5e49:79ff:fe6...ff02::c	SSDP	372	NOTIFY * HTTP/1.1	
843	249.114125399	fe80::5e49:79ff:fe6...ff02::c	SSDP	411	NOTIFY * HTTP/1.1	
844	249.117673673	fe80::5e49:79ff:fe6...ff02::c	SSDP	372	NOTIFY * HTTP/1.1	
845	249.12058624373	fe80::5e49:79ff:fe6...ff02::c	SSDP	431	NOTIFY * HTTP/1.1	
846	249.12058624373	fe80::5e49:79ff:fe6...ff02::c	SSDP	399	NOTIFY * HTTP/1.1	
847	249.122478594	fe80::5e49:79ff:fe6...ff02::c	SSDP	443	NOTIFY * HTTP/1.1	
848	249.124712671	fe80::5e49:79ff:fe6...ff02::c	SSDP	427	NOTIFY * HTTP/1.1	
849	249.126699747	fe80::5e49:79ff:fe6...ff02::c	SSDP	425	NOTIFY * HTTP/1.1	
850	249.129151475	fe80::5e49:79ff:fe6...ff02::c	SSDP	439	NOTIFY * HTTP/1.1	
851	249.129151475	fe80::5e49:79ff:fe6...ff02::c	SSDP	363	NOTIFY * HTTP/1.1	
852	249.110541017	fe80::5e49:79ff:fe6...ff02::c	SSDP	373	NOTIFY * HTTP/1.1	
853	249.110892288	fe80::5e49:79ff:fe6...ff02::c	SSDP	436	NOTIFY * HTTP/1.1	
854	249.1142699272	fe80::5e49:79ff:fe6...ff02::c	SSDP	373	NOTIFY * HTTP/1.1	
855	249.1144551951	fe80::5e49:79ff:fe6...ff02::c	SSDP	412	NOTIFY * HTTP/1.1	

Frame 826: 365 bytes on wire (2920 bits), 365 bytes captured (2920 bits) on interface enp0s31f6, id 8  
 Ethernet II, Src: Unknown (00:0c:29:a9:79:6a), Dst: IPv6mcast\_0c (33:33:00:00:00:0c)  
 Internet Protocol Version 6, Src Port: 1900, Dst Port: 1900  
 User Datagram Protocol, Src Port: 1900, Dst Port: 1900  
 Simple Service Discovery Protocol

**Abbildung 19:** Aufzeichnung des SSDP-Protokolls

## 2.7 HTTP und TCP

### Initiiieren Sie eine HTTP-TCP-Sitzung (beliebige Website) und zeichnen Sie die Protokollabläufe auf

Zuerst wird eine DNS-Request getätigter. Daraufhin folgt der 3-Way-Handshake.



No.	Time	Source	Destination	Protocol	Length	Info
716	7.688834	141.70.124.5	100.64.84.66	DNS	158	Standard query response 0x58df AAAA news.ycombinator.com SOA ns-225.awsdns-28.com
717	7.613971	141.70.124.5	100.64.84.66	DNS	233	Standard query response 0x189d A news.ycombinator.com A 209.216.230.240 NS ns-1411.awsdns-48.org NS ns-1914.awsdns-47.co.l
718	7.614386	100.64.84.66	209.216.230.240	TCP	78	49314 → 443 [SYN, ECN, CWR] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSecr=0 SACK_PERM=1 TSecr=2512581059 TSecr=2512581059
719	7.765210	209.216.230.240	100.64.84.66	TCP	74	443 → 49314 [SYN, ACK, ECN] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=64 TSecr=1 TSecr=2045828460 TSecr=2045828460
720	7.765334	100.64.84.66	209.216.230.240	TCP	66	49314 → 443 [ACK] Seq=1 Ack=1 Win=131712 Len=0 TSval=2512581211 TSecr=2045828460
721	7.765826	100.64.84.66	209.216.230.240	TLSv1...	583	Client Hello

Frame Number: 718  
 Frame Length: 78 bytes (624 bits)  
 Capture Length: 78 bytes (624 bits)  
 [Frame is marked: None]  
 [Frame is selected: False]  
 [Protocols in frame: eth:ether:ip:tcp]  
 [Coloring Rule Name: TCP SYN/FIN]  
 [Coloring Rule String: tcp.flags & 0x80 || tcp.flags.fin == 1]  
 > Ethernet II, Src: Apple\_44:f3:0e (ad:83:e7:44:f3:0e), Dst: Juniper\_N\_9a:93:ce (b0:a8:6e:9a:93:ce)  
 > Internet Protocol Version 4, Src: 100.64.84.66, Dst: 209.216.230.240  
 ✓ Transmission Control Protocol, Src Port: 49314, Dst Port: 443, Seq: 0, Len: 0  
 Source Port: 49314  
 Destination Port: 443  
 [Stream index: 10]  
 [TCP Segment Len: 0]  
 Sequence Number: 0 (relative sequence number)  
 Sequence Number (raw): 3747062828  
 [Next Sequence Number: 1 (relative sequence number)]  
 Acknowledgment Number: 0  
 Acknowledgment number (raw): 0  
 1011 .... = Header Length: 44 bytes (11)  
 > Flags: 0x02c (SYN, ECN, CWR)  
 Window: 65535  
 [Calculated window size: 65535]  
 Checksum: 0xf787 [unverified]  
 [Checksum Status: Unverified]  
 Urgent Pointer: 0  
 ✓ Options: (24 bytes), Maximum segment size, No-Operation (NOP), Window scale, No-Operation (NOP), No-Operation (NOP), Timestamps, SACK permitted, End of Option List (EOL)  
 > TCP Option - Maximum segment size: 1460 bytes  
 > TCP Option - No-Operation (NOP)  
 > TCP Option - Window scale: 6 (multiplied by 64)  
 > TCP Option - No-Operation (NOP)  
 > TCP Option - No-Operation (NOP)  
 > TCP Option - No-Operation (NOP)  
 > TCP Option - Timestamps: TSval 2512581059, TSecr 0  
 > TCP Option - SACK permitted  
 > TCP Option - End of Option List (EOL)

**Abbildung 22:** Das SYN-Segment enthält die Optionen Maximum Segment Size, Window scale, Timestamps und SACK (Selective Acknowledgement). Die Maximum Segment Size gibt die maximale Anzahl an Daten in Bytes an, die pro Segment akzeptiert werden. Der Window scale factor ist dazu da, die zuvor gesetzte maximale window-size über 65535 Bytes zu setzen. Der Timestamp misst die derzeitige Roundtrip time. Dadurch kann man den retransmission-timer jederzeit neu evaluieren. Selective Acknowledgement wird benutzt, um bei verlorenen Segmenten wirklich nur die fehlenden retransmitten zu müssen.

No.	Time	Source	Destination	Protocol	Length	Info
716	7.6.688034	141.70.124.5	100.64.84.66	DNS	158	Standard query response 0x58df AAAA news.ycombinator.com SOA ns-225.awsdns-28.com
717	7.6.13971	141.70.124.5	100.64.84.66	DNS	233	Standard query response 0x189d A news.ycombinator.com A 209.216.230.240 NS ns-1411.awsdns-48.org NS ns-1914.awsdns-47.co.l
718	7.6.14386	100.64.84.66	209.216.230.240	TCP	78	49314 → 443 [SYN, ECN, CWR] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 Tsva=2512581059 Tsecr=0 SACK_PERM=1
719	7.7.65210	209.216.230.240	100.64.84.66	TCP	74	443 → 49314 [SYN, ACK, ECN] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=64 SACK_PERM=1 Tsva=2045828460 Tsecr=2512581059
720	7.7.65334	100.64.84.66	209.216.230.240	TCP	66	49314 → 443 [ACK] Seq=1 Ack=1 Win=131712 Len=0 Tsva=2512581211 Tsecr=2045828460
721	7.7.65826	100.64.84.66	209.216.230.240	TLSv1..	583	Client Hello

[Time delta from previous displayed frame: 0.158904000 seconds]  
[Time since reference or first frame: 7.765210000 seconds]  
Frame Number: 719  
Frame Length: 592 bytes (592 bits)  
Capture Length: 74 bytes (592 bits)  
[Frame is marked: False]  
[Frame is ignored: False]  
[Protocols in frame: eth:etherype:ip:tcp]  
[Coloring Rule Name: TCP SYN/FIN]  
[Coloring Rule String: tcp.flags & 0x02 || tcp.flags.fin == 1]  
> Ethernet II, Src: JuniperN\_9a:93:ce (b0:a8:6e:9a:93:ce), Dst: Apple\_44:f3:0e (a4:83:e7:44:f3:0e)  
> Internet Protocol Version 4, Src: 209.216.230.240, Dst: 100.64.84.66  
✓ Transmission Control Protocol, Src Port: 443, Dst Port: 49314, Seq: 0, Ack: 1, Len: 0  
Source Port: 443  
Destination Port: 49314  
[Stream index: 10]  
[TCP Segment Len: 0]  
Sequence Number: 0 (relative sequence number)  
Sequence Number (raw): 2792502608  
[Next Sequence Number: 1 (relative sequence number)]  
Acknowledgment Number: 1 (relative ack number)  
Acknowledgment number (raw): 3747062829  
1010 .... = Header Length: 40 bytes (10)  
> Flags: 0x052 (SYN, ACK, ECN)  
Window: 65535  
[Calculated window size: 65535]  
Checksum: 0xd5db [unverified]  
[Checksum Status: Unverified]  
Urgent Pointer: 0  
Options: (28 bytes), Maximum segment size, No-Operation (NOP), Window scale, SACK permitted, Timestamps  
> TCP Option - Maximum segment size: 1460 bytes  
> TCP Option - No-Operation (NOP)  
> TCP Option - Window scale: 6 (multiply by 64)  
> TCP Option - SACK permitted  
> TCP Option - Timestamps: Tsva 2045828460, Tsecr 2512581059  
> [SEQ/ACK analysis]

**Abbildung 23:** Das SYN-ACK-Segment verwendet wieder die Optionen Maximum Segment Size, Window scale, SACK und Timestamps.

No.	Time	Source	Destination	Protocol	Length	Info
716	7.6.688034	141.70.124.5	100.64.84.66	DNS	158	Standard query response 0x58df AAAA news.ycombinator.com SOA ns-225.awsdns-28.com
717	7.6.13971	141.70.124.5	100.64.84.66	DNS	233	Standard query response 0x189d A news.ycombinator.com A 209.216.230.240 NS ns-1411.awsdns-48.org NS ns-1914.awsdns-47.co.l
718	7.6.14386	100.64.84.66	209.216.230.240	TCP	78	49314 → 443 [SYN, ECN, CWR] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 Tsva=2512581059 Tsecr=0 SACK_PERM=1
719	7.7.65210	209.216.230.240	100.64.84.66	TCP	74	443 → 49314 [SYN, ACK, ECN] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=64 SACK_PERM=1 Tsva=2045828460 Tsecr=2512581059
720	7.7.65334	100.64.84.66	209.216.230.240	TCP	66	49314 → 443 [ACK] Seq=1 Ack=1 Win=131712 Len=0 Tsva=2512581211 Tsecr=2045828460
721	7.7.65826	100.64.84.66	209.216.230.240	TLSv1..	583	Client Hello

[Time delta from previous captured frame: 0.000124000 seconds]  
[Time delta from previous displayed frame: 0.000124000 seconds]  
[Time since reference or first frame: 7.765334000 seconds]  
Frame Number: 720  
Frame Length: 66 bytes (528 bits)  
Capture Length: 66 bytes (528 bits)  
[Frame is marked: False]  
[Frame is ignored: False]  
[Protocols in frame: eth:etherype:ip:tcp]  
[Coloring Rule Name: TCP]  
[Coloring Rule String: tcp]  
> Ethernet II, Src: Apple\_44:f3:0e (a4:83:e7:44:f3:0e), Dst: JuniperN\_9a:93:ce (b0:a8:6e:9a:93:ce)  
> Internet Protocol Version 4, Src: 100.64.84.66, Dst: 209.216.230.240  
✓ Transmission Control Protocol, Src Port: 49314, Dst Port: 443, Seq: 1, Ack: 1, Len: 0  
Source Port: 49314  
Destination Port: 443  
[Stream index: 10]  
[TCP Segment Len: 0]  
Sequence Number: 0 (relative sequence number)  
Sequence Number (raw): 3747062829  
[Next Sequence Number: 1 (relative sequence number)]  
Acknowledgment Number: 1 (relative ack number)  
Acknowledgment number (raw): 2792502609  
1000 .... = Header Length: 32 bytes (8)  
> Flags: 0x010 (ACK)  
Window: 2058  
[Calculated window size: 131712]  
[Window size scaling factor: 64]  
Checksum: 0xfc44 [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: Tsva 2512581211, Tsecr 2045828460  
> [SEQ/ACK analysis]

**Abbildung 24:** Das ACK Segment hat nur die Timestamps-Option gesetzt.

### Dokumentieren und erläutern Sie die Verwendung der Portnummern bei der Dienstanfrage und der Beantwortung des Dienstes durch den Server.

Unser Computer sendet von Port 49314 an Port 443, welcher für HTTPS genutzt wird. Unser Port ist

dabei arbiträr vom System gewählt, der HTTPS Port ist allerdings fest für HTTPS reserviert. Mit einem Port ist ein Dienst eines Rechners gekennzeichnet. Die Kombination aus Port und IP ergibt einen Socket. Wir senden unsere Nachrichten also an den Socket `209.216.230.240:443`.

**Klicken Sie auf der Website ein anderes Bild / Link an. Beobachten und dokumentieren Sie: wie verändert sich der TCP-Ablauf?**

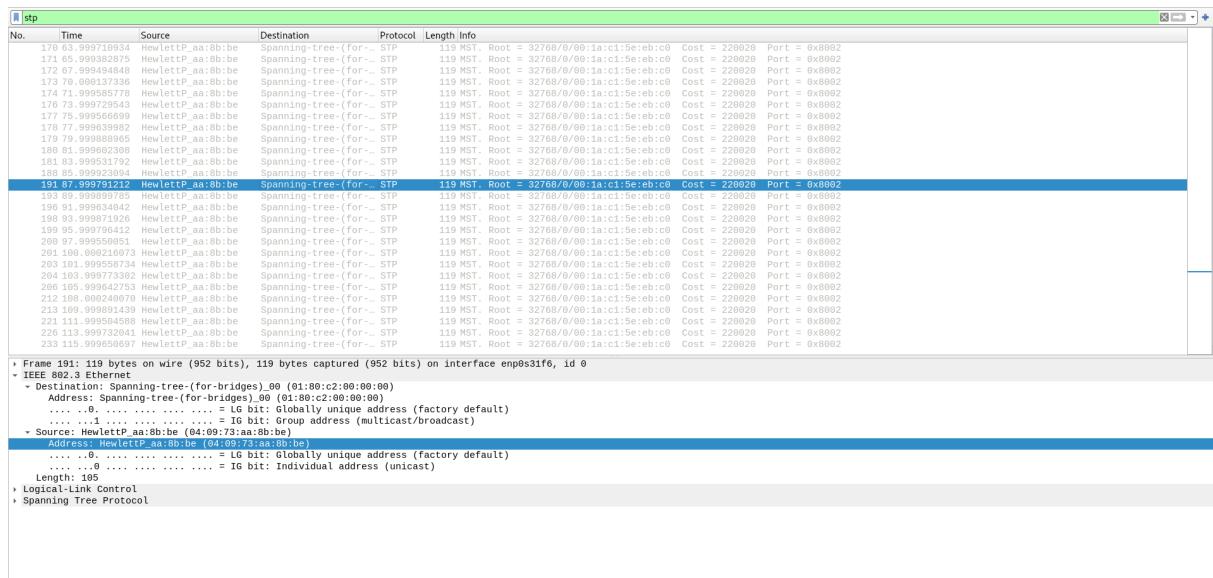
**TODO:** Add valid image

## 2.8 MAC

**Wie lauten die MAC-Adressen der im Labor befindlichen Ethernet-Switches? Wie haben Sie die Switches identifizieren können. Welche Möglichkeiten der Identifizierung gibt es?**

Beim Spanning-Tree-Protocol lässt sich sehen, dass die Quelle der Nachrichten immer ein HP-Gerät ist. Dieses muss ein fähiges Kopplungselement des Netzwerkes sein, welches das Spanning-Tree-Protocol unterstützt. Daher wird dies mit hoher Wahrscheinlichkeit der Ethernet-Switch sein.

**MAC-Adresse:** `04:09:73:aa:8b:be`



**Abbildung 25:** Aufzeichnung des STP-Protokolls

**Welche MAC-Adresse hat ihr Nachbarrechner?**

Durch einen `ping` konnten wir die MAC-Adresse des Switches herausfinden.

**MAC-Adresse:** `4c:52:62:0e:54:2b`

Frame 216: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface enp0s31f6, id 0  
 Ethernet II, Src: linux-2.local (4c:52:62:0e:54:8b), Dst: linux-3.local (4c:52:62:0e:54:2b)  
 Destination: linux-3.local (4c:52:62:0e:54:2b)  
 Address: linux-3.local (4c:52:62:0e:54:2b)  
 .... .B. .... . .... = 16 bit: Globally unique address (factory default)  
 .... .0. .... . .... = 16 bit: Individual address (unicast)  
 Source: linux-2.local (4c:52:62:0e:54:8b)  
 Address: linux-2.local (4c:52:62:0e:54:8b)  
 .... .B. .... . .... = 16 bit: Globally unique address (factory default)  
 .... .0. .... . .... = 16 bit: Individual address (unicast)  
 Type: IPv4 (0x0800)  
 Internet Protocol Version 4, Src: linux-2.local (141.62.66.5), Dst: linux-3.local (141.62.66.6)  
 0100 . . . . . = Version: 4  
 .0101 = Header Length: 20 bytes (5)  
 Differentiated Services Field: 0x00 (DSCH: CS0, ECN: Not-ECT)  
 Total Length: 84  
 Identification: 0xe0d3 (57443)  
 Flags: 0x40, Don't fragment  
 Fragment Offset: 0  
 Time to Live: 64  
 Protocol: ICMP (1)  
 Header Checksum: 0xbbbd [validation disabled]  
 [Header checksum status: Unverified]  
 Source Address: linux-2.local (141.62.66.5)  
 Destination Address: linux-3.local (141.62.66.6)  
 Internet Control Message Protocol

**Abbildung 26:** MAC-Adresse des Nachbarrechners**Welche MAC-Adresse hat der Labor-Router?**

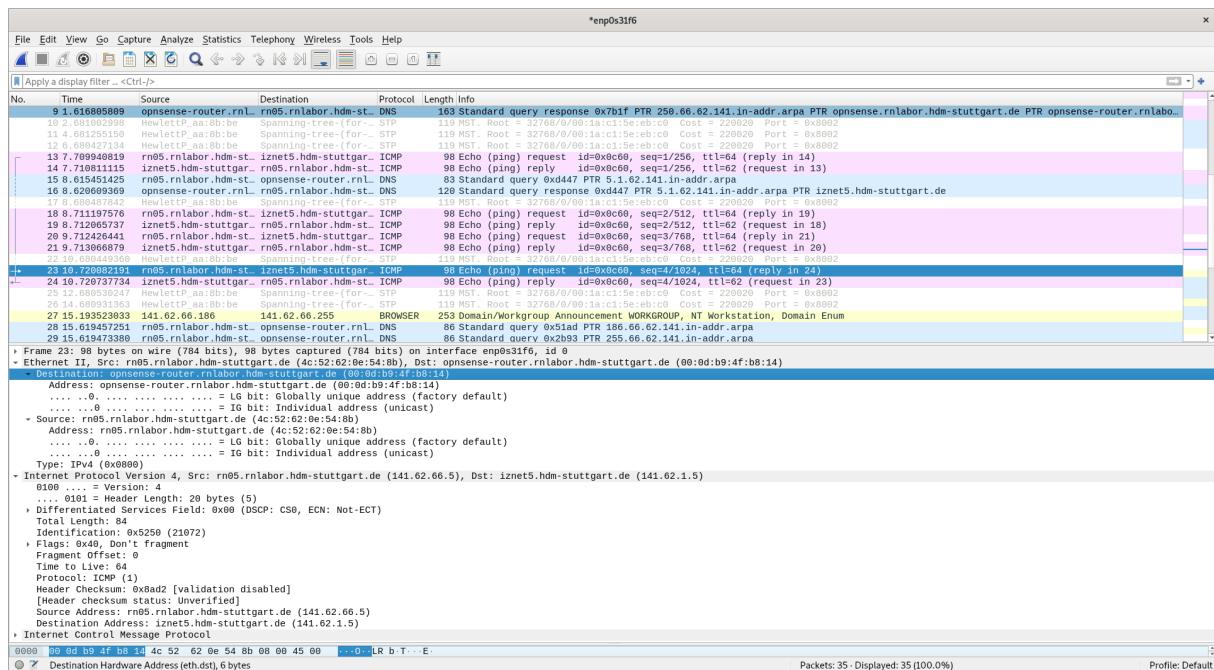
Durch einen [ping](#) konnten wir die MAC-Adresse des Routers herausfinden.

**MAC-Adresse:** 00:0d:b9:4f:b8:14

Frame 2: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface enp0s31f6, id 0  
 Ethernet II, Src: rn05.rnlabor.hdm-stuttgart.de (00:0d:b9:4f:b8:14), Dst: opnsense-router.rnlabor.hdm-stuttgart.de (00:0d:b9:4f:b8:14)  
 Destination: opnsense-router.rnlabor.hdm-stuttgart.de (00:0d:b9:4f:b8:14)  
 Address: opnsense-router.rnlabor.hdm-stuttgart.de (00:0d:b9:4f:b8:14)  
 .... .0. .... . .... = 16 bit: Globally unique address (factory default)  
 .... .0. .... . .... = 16 bit: Individual address (unicast)  
 Source: rn05.rnlabor.hdm-stuttgart.de (00:0d:b9:4f:b8:14)  
 Address: rn05.rnlabor.hdm-stuttgart.de (00:0d:b9:4f:b8:14)  
 .... .0. .... . .... = 16 bit: Globally unique address (factory default)  
 .... .0. .... . .... = 16 bit: Individual address (unicast)  
 Type: IPv4 (0x0800)  
 Internet Protocol Version 4, Src: rn05.rnlabor.hdm-stuttgart.de (141.62.66.5), Dst: opnsense-router.rnlabor.hdm-stuttgart.de (141.62.66.250)  
 0100 . . . . . = Version: 4  
 .0101 = Header Length: 20 bytes (5)  
 Differentiated Services Field: 0x00 (DSCH: CS0, ECN: Not-ECT)  
 Total Length: 84  
 Identification: 0x80d7 (20839)  
 Flags: 0x40, Don't fragment  
 Fragment Offset: 0  
 Time to Live: 64  
 Protocol: ICMP (1)  
 Header Checksum: 0x3266 [validation disabled]  
 [Header checksum status: Unverified]  
 Source Address: rn05.rnlabor.hdm-stuttgart.de (141.62.66.5)  
 Destination Address: opnsense-router.rnlabor.hdm-stuttgart.de (141.62.66.250)  
 Internet Control Message Protocol

**Abbildung 27:** MAC-Adresse des Labor-Routers**Welche MAC-Adresse hat der Server 141.62.1.5 (außerhalb des Labor-Netzes)?**

Da der Rechner außerhalb des Labor-Netzes ist, kann dessen Mac nicht bestimmt werden.



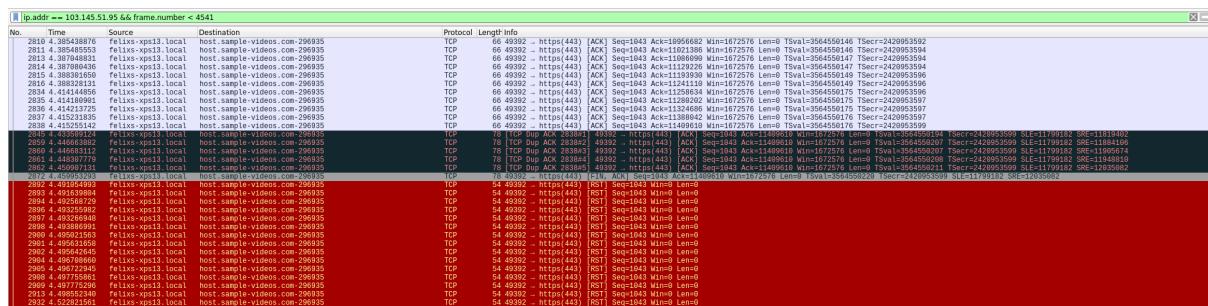
**Abbildung 28:** MAC-Adresse des externen Rechners

## 2.9 STP

### Filtern Sie auf das Protokoll BPDU/STP. Wer sendet es und welchen Sinn hat dieses Protokoll?

Das STP-Protokoll ist das Spanning Tree Protocol. Das STP-Protokoll verhindert Schleifenbildung; dies ist besonders dann von Nutzen, wenn Redundanzen vorhanden sind. Beim STP-Protokoll werden durch alle am Netz beteiligten Switches eine "Root Bridge" gewählt und redundante Links werden deaktiviert. Wie anhand der OUI der MAC-Adresse erkannt werden kann wird dieses hier von einem HP-Switch verwendet.

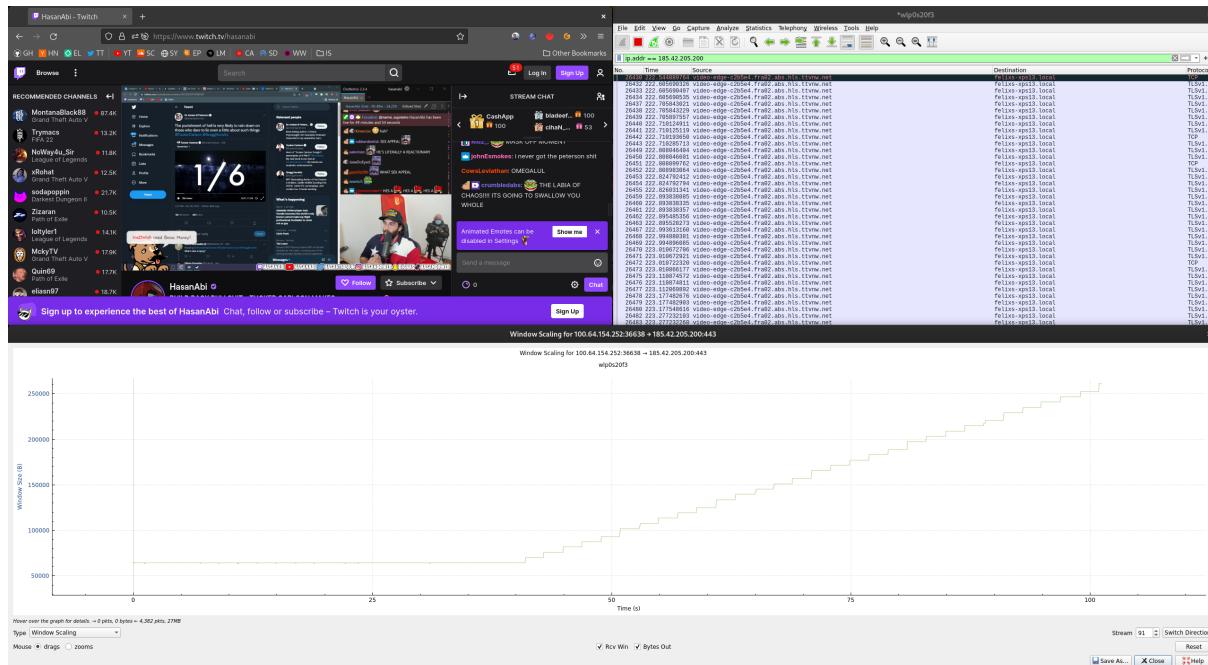




**Abbildung 30:** Capture beim Canceln des eines Downloads über HTTPS

Da der Download hier via HTTPS durchgeführt wurde, kann erkannt werden, dass die darunterliegende TCP-Verbindung unterbrochen wurde, indem die **RST**-Flag gesetzt wurde. Auch ein TCP-Segment, in welchem hier die **FIN**- und **ACK**-Flags gesetzt wurden, ist dementsprechend zu erkennen.

### Protokollieren sie ein Video-Streaming Ihrer Wahl. Welche TCP-Ports werden wozu benutzt? Filtern Sie alle Rahmen, in denen sich das TCP-Window geändert hat



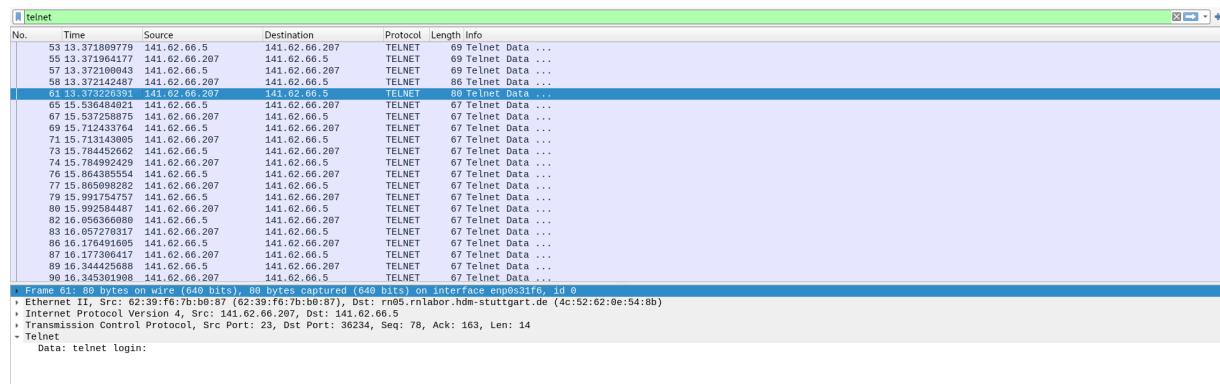
**Abbildung 31:** Verlauf der TCP-Window-Size beim Streaming von Twitch

Hier wurde ein Stream von Twitch konsumiert; wie zu erkennen ist, wird die Window-Size stetig erhöht. Es wird Port 443, der Standard-Port für HTTPS, verwendet. Seitens des Clients wird vom TCP-Stack des Kernels ein temporärer Port zugewiesen.

## 2.12 Telnet und SSH

**Protokollieren Sie den Ablauf einer TELNET-Verbindung zur IP-Adresse 141.62.66.207 (login: praktikum; passwd: versuch). Können Sie Passwörter im Wireshark-Trace identifizieren? Wie verhält sich im Vergleich dazu eine SSH-Verbindung zum gleichen Server?**

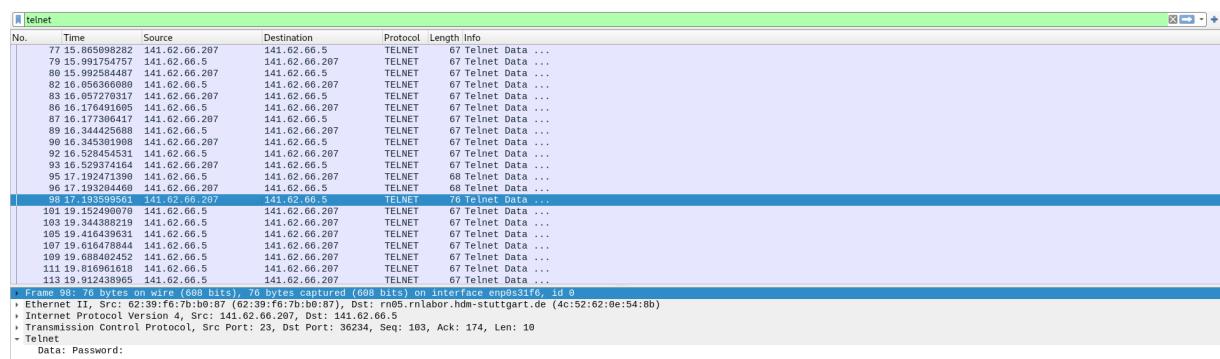
Wie zu erkennen ist, wird für eine Telnet-Verbindung eine TCP-Verbindung aufgebaut. Die Passwörter sind zu erkennen.



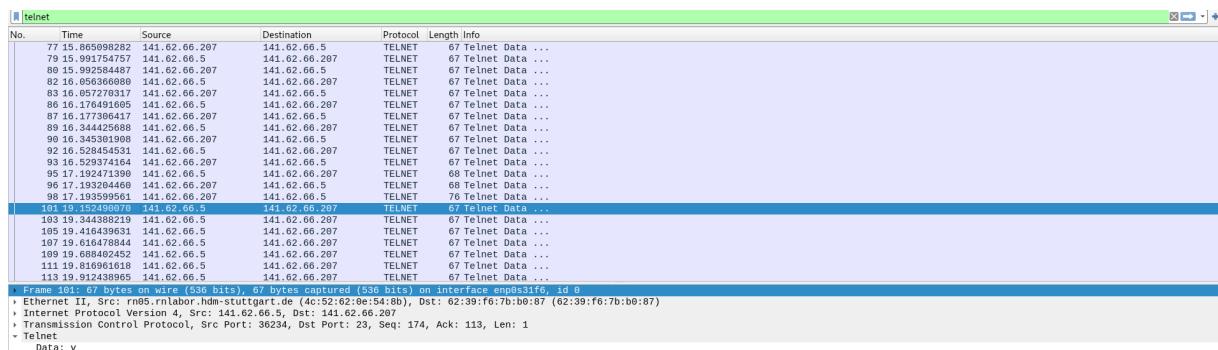
**Abbildung 32:** Capture des Telnet-Logins

### Können Sie Passwörter im Wireshark-Trace identifizieren?

Da Telnet unverschlüsselt ist, können Passwörter identifiziert und ausgelesen werden.

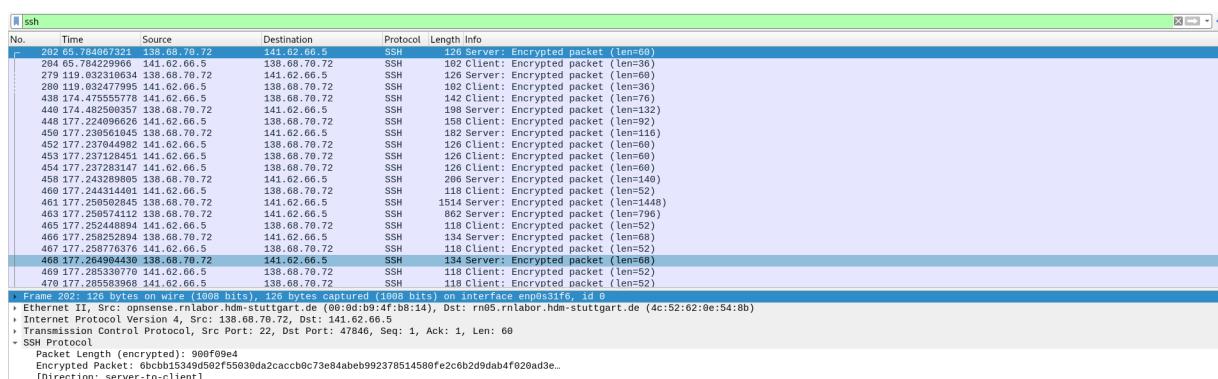


**Abbildung 33:** Capture des Telnet-Passworts

**Abbildung 34:** Capture eines Charakters des Telnet-Passworts

### Wie verhält sich im Vergleich dazu eine SSH-Verbindung zum gleichen Server?

Die SSH-Verbindung ist verschlüsselt; Passwörter, Logins etc. können hier nicht mitgelesen werden.

**Abbildung 35:** Capture eines verschlüsselten SSH-Pakets

## 2.13 Wireshark-Filter

**Entwickeln, testen und dokumentieren Sie Wireshark-Filter zur Lösung folgender Aufgaben:**

**Nur IP-Pakete, deren TTL größer ist als ein von Ihnen sinnvoll gewählter Referenzwert**



Frame 3713: 89 bytes on wire (712 bits), 89 bytes captured (712 bits) on interface enp0s31f6, id 0  
 Ethernet II, Src: rnlabor.hdm-stuttgart.de (4c:52:62:0e:54:b8), Dst: opnsense.rnlabor.hdm-stuttgart.de (00:0d:b9:4f:b8:14)  
 Internet Protocol Version 4, Src: 141.62.66.5, Dst: 212.77.241.212  
 Transmission Control Protocol (TCP), Src Port: 53198, Dst Port: 21, Seq: 13, Ack: 57, Len: 23  
 File Transfer Protocol (FTP)  
 [Current working directory:]

**Abbildung 38:** Capture eines FTP-Pakets, welches ein Password enthält**Nur den Port 80-Verkehr zu Ihrer IP-Adresse (ankommend und abgehend)**

Mittels eines Filters wurde ausschließlich TCP-Traffic auf Port 80 dargestellt. Mittels `|| udp.port == 80` hätte auch noch UDP-Traffic auf diesem Port dargestellt werden können.

**Abbildung 39:** Capture aller TCP-Segmente auf Port 80**Nur Pakete mit einer IP-Multicast-Adresse**

Mittels eines Filters werden nur IPs > 224.0.0.0 dargestellt, was IP-Multicast-Adressen sind.

**Abbildung 40:** Capture aller IP-Pakete mit Multicast-Adressen