

---

# **Praktikum Rechnernetze**

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

Jakob Waibel, Daniel Hiller, Elia Wüstner, Felix Pojtinger

2021-10-26

## 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 . . . . .	14
2.8 MAC . . . . .	18
2.9 STP . . . . .	20
2.10 SNMP . . . . .	21
2.11 Streaming and Downloads . . . . .	22
2.12 Telnet und SSH . . . . .	23
2.13 Wireshark-Filter . . . . .	25

## 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 Koppellement 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

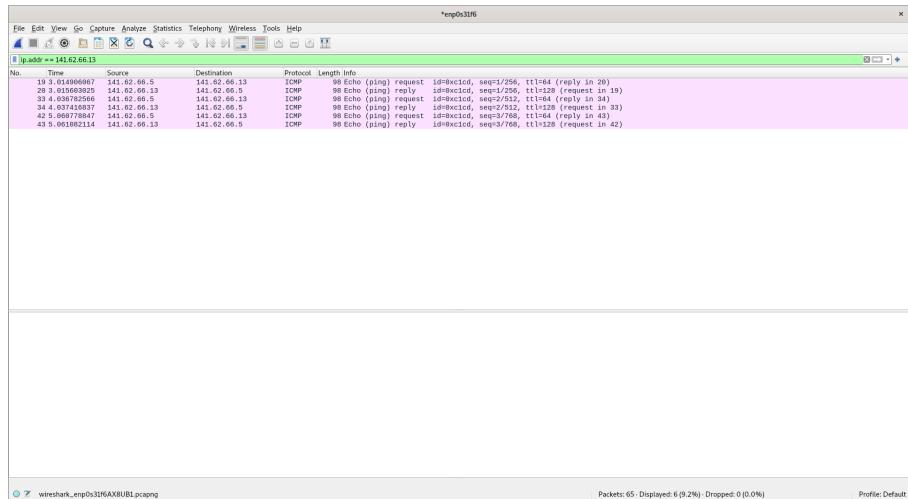


**Abbildung 5:** Ausklammern der eig. IP, Option 2

## 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 initialisieren**

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	19.15.392945861	0.0.0.0	255.255.255.255	DHCP	342	DHCP Discover - Transaction ID 0x70ef81d
2	20.15.393517126	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 initialisieren dann ist der DHCP Ablauf deutlich kürzer, da beim Booten unter der Haube noch deutlich mehr gemacht werden muss (es muss z.B. keine DHCP-Request des BIOS zum Netzwerkboot getätigter 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
```

dns & frame.number < 20						
No.	Time	Source	Destination	Protocol	Length	Info
11	1.357358000	rn05.rnlabo.hdm-st.	opnsense-router.rnl.DNS	DNS	93	Standard query 0xa276 A google.com OPT
12	1.371692978	opnsense-router.rnl.rn05.rnlabo.hdm-st.	DNS	DNS	97	Standard query response 0xa276 A google.com A 142.250.180.174 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	1.205820789	rn05.rnlabor.hdm-st..	opnsense-router.rnl...	DNS	84	Standard query 0xd2b PTR 5.66.62.141.in-addr.arpa
5	1.205849397	rn05.rnlabor.hdm-st..	opnsense-router.rnl...	DNS	88	Standard query 0x8883 PTR 1.1.1.1.in-addr.arpa
6	1.207179251	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	1.207611338	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

**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-router.rnl...	DNS	84	Standard query 0xceef PTR 5.66.62.141.in-addr.arpa
6	1.088436116	rn05.rnlabor.hdm-st..	opnsense.rnlabor.hdm...	DNS	88	Standard query 0x6247 PTR 1.1.1.1.in-addr.arpa
7	1.089061823	opnsense.rnlabor.hdm...	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...	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...	DNS	84	Standard query 0xf6f0 PTR 250.66.62.141.in-addr.arpa
17	2.089268813	opnsense.rnlabor.hdm...	rn05.rnlabor.hdm-st..	DNS	163	Standard query response 0x4fb PTR 250.66.62.141.in-addr.arpa PTR opnsense-router.rnlabor.hdm-stuttgart.de PTR opnsense.rnlabor.hdm...
22	2.089268813	opnsense.rnlabor.hdm...	opnsense.rnlabor.hdm...	DNS	84	Standard query 0x6247 PTR 1.1.1.1.in-addr.arpa
23	3.087954583	rn05.rnlabor.hdm-st..	opnsense.rnlabor.hdm...	DNS	84	Standard query 0x6247 PTR 1.1.1.1.in-addr.arpa
24	3.087959318	rn05.rnlabor.hdm-st..	opnsense.rnlabor.hdm...	DNS	88	Standard query 0x1f24 PTR 255.255.254.169.in-addr.arpa
25	3.088893145	opnsense.rnlabor.hdm...	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...	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...	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	110.515587298	linux-3.local	linux-2.local	ARP	68	141.62.66.6 is at 4c:52:02:0e:54:2b
231	115.675164735	linux-3.local	linux-2.local	ARP	68	Who has 141.62.66.5? Tell 141.62.66.6
262	116.673186793	linux-2.local	linux-3.local	ARP	42	141.62.66.5 is at 4c:52:02:0e:54:0b

**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	118.5156578213	linux-3.local	Broadcast	ARP	42	Who has 141.62.66.6? Tell 141.62.66.5
215	118.5155867288	linux-3.local	linux-2.local	ARP	68	141.62.66.6 is at 4c:52:02:0e: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	115.673186763	linux-3.local	linux-2.local	ARP	42	141.62.66.5 is at 4c:52:02:0e:54:8b

Frame 214: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface enp0s31f6, id 0  
 Ethernet II, Src: Unknown (00:0c:29:02:0e:54:8b), Dst: Broadcast (ff:ff:ff:ff:ff:ff)  
 Type: ARP (0x0806)  
 Version: 2  
 Source: Broadcast (ff:ff:ff:ff:ff:ff)  
 Destination: Broadcast (ff:ff:ff:ff:ff:ff)  
 Type: ARP (0x0806)  
 Subtype: 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	240.109859521	fe80::5e49:79ff:fe6...ff02::c	SSDP	363	NOTIFY * HTTP/1.1	
840	240.109859520	fe80::5e49:79ff:fe6...ff02::c	SSDP	372	NOTIFY * HTTP/1.1	
841	240.119442125	fe80::5e49:79ff:fe6...ff02::c	SSDP	435	NOTIFY * HTTP/1.1	
842	240.113785421	fe80::5e49:79ff:fe6...ff02::c	SSDP	372	NOTIFY * HTTP/1.1	
843	240.114125399	fe80::5e49:79ff:fe6...ff02::c	SSDP	411	NOTIFY * HTTP/1.1	
844	240.117673673	fe80::5e49:79ff:fe6...ff02::c	SSDP	372	NOTIFY * HTTP/1.1	
845	240.112082437	fe80::5e49:79ff:fe6...ff02::c	SSDP	431	NOTIFY * HTTP/1.1	
846	240.112082436	fe80::5e49:79ff:fe6...ff02::c	SSDP	399	NOTIFY * HTTP/1.1	
847	240.122478594	fe80::5e49:79ff:fe6...ff02::c	SSDP	443	NOTIFY * HTTP/1.1	
848	240.124712671	fe80::5e49:79ff:fe6...ff02::c	SSDP	427	NOTIFY * HTTP/1.1	
849	240.126997474	fe80::5e49:79ff:fe6...ff02::c	SSDP	425	NOTIFY * HTTP/1.1	
850	240.129151475	fe80::5e49:79ff:fe6...ff02::c	SSDP	439	NOTIFY * HTTP/1.1	
851	240.129151476	fe80::5e49:79ff:fe6...ff02::c	SSDP	363	NOTIFY * HTTP/1.1	
852	241.110541017	fe80::5e49:79ff:fe6...ff02::c	SSDP	373	NOTIFY * HTTP/1.1	
853	241.110892288	fe80::5e49:79ff:fe6...ff02::c	SSDP	436	NOTIFY * HTTP/1.1	
854	241.114289272	fe80::5e49:79ff:fe6...ff02::c	SSDP	373	NOTIFY * HTTP/1.1	
855	241.114451951	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 0  
 Ethernet II, Src: IP6mcast\_0c (5c:49:79:6a:a9:78), Dst: IPv6mcast\_0c (33:33:00:00:00:00)  
 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





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 S0A 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=1460 WS=64 Tsv=2512581059 Tsecr=0 SACK_PERM=1
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 SACK_PERM=1 Tsv=2045828460 Tsecr=2512581059
720	7.765334	100.64.84.66	209.216.230.240	TCP	66	49314 → 443 [ACK] Seq=1 Ack=1 Win=131712 Len=0 Tsv=2512581211 Tsecr=2045828460
721	7.765826	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: 74 bytes (592 bits)  
Capture Length: 74 bytes (592 bits)  
[Frame is marked: False]  
[Frame is ignored: False]  
[Protocols in frame: eth:ether:ip:tcp]  
[Coloring Rule Name: TCP SYN/FIN]  
[Coloring Rule String: tcp.flags & 0x02 || tcp.flags.fin == 1]  
> Ethernet II, Src: Juniper\_N\_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, EON)  
    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: Tsv=2045828460, Tsecr=2512581059  
    > [SEQ/ACK analysis]

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

No.	Time	Source	Destination	Protocol	Length	Info
716	7.688834	141.70.124.5	100.64.84.66	DNS	158	Standard query response 0x58df AAAA news.ycombinator.com S0A 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=1460 WS=64 Tsv=2512581059 Tsecr=0 SACK_PERM=1
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 SACK_PERM=1 Tsv=2045828460 Tsecr=2512581059
720	7.765334	100.64.84.66	209.216.230.240	TCP	66	49314 → 443 [ACK] Seq=1 Ack=1 Win=131712 Len=0 Tsv=2512581211 Tsecr=2045828460
721	7.765826	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:ether:ip:tcp]  
[Coloring Rule Name: TCP]  
[Coloring Rule String: tcp]  
> Ethernet II, Src: Apple\_44:f3:0e (a4:83: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: 1, Ack: 1, Len: 0  
    Source Port: 49314  
    Destination Port: 443  
    [Stream index: 10]  
    [TCP Segment Len: 0]  
    Sequence Number: 1 (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: Tsv=2512581211, Tsecr=2045828460  
    > [SEQ/ACK analysis]

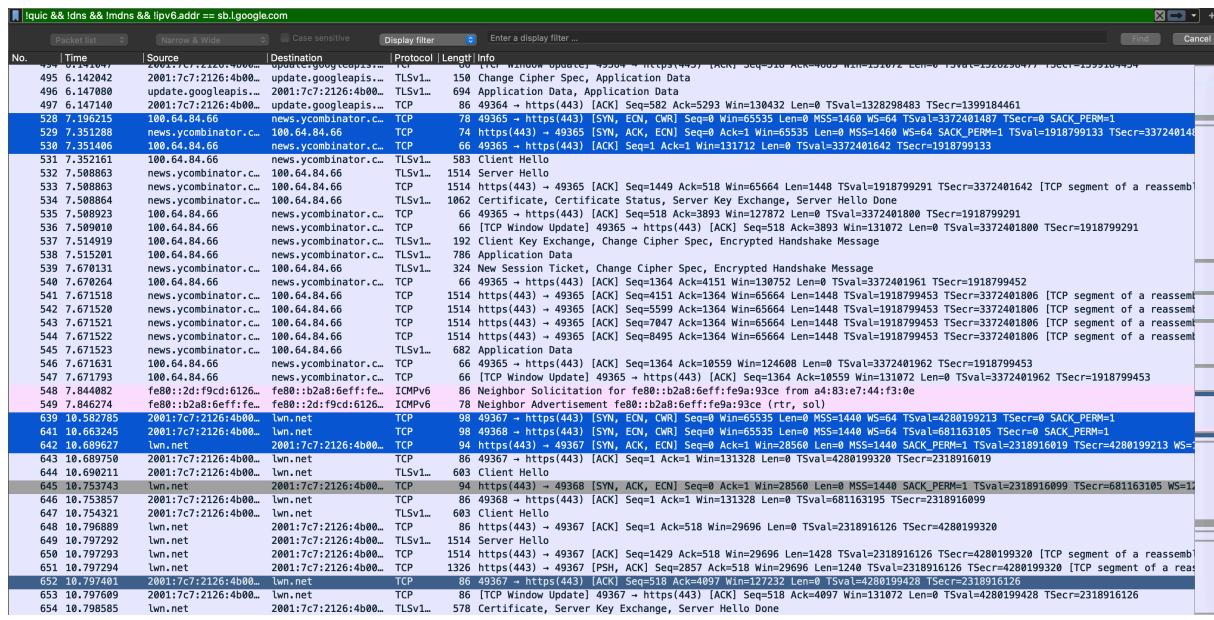
Das ACK Segment hat nur die Timestamps-Option gesetzt.

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.

## 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?



**Abbildung 22:** Es wird eine TCP-Verbindung zur neuen Seite (lwn.net) aufgebaut. Dies sieht man anhand des wiederholten TCP-Handshakes.

## 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

No.	Time	Source	Destination	Protocol	Length	Info
176 63.999716934	HewlettP_aa:8b:be	Spanning-tree-(for- STP	119 MST, Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 228020 Port = 0x8002			
177 65.999832820	HewlettP_aa:8b:be	Spanning-tree-(for- STP	119 MST, Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 228020 Port = 0x8002			
178 66.999834840	HewlettP_aa:8b:be	Spanning-tree-(for- STP	119 MST, Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 228020 Port = 0x8002			
179 67.999837330	HewlettP_aa:8b:be	Spanning-tree-(for- STP	119 MST, Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 228020 Port = 0x8002			
174 71.999585778	HewlettP_aa:8b:be	Spanning-tree-(for- STP	119 MST, Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 228020 Port = 0x8002			
176 73.999729543	HewlettP_aa:8b:be	Spanning-tree-(for- STP	119 MST, Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 228020 Port = 0x8002			
177 75.999566690	HewlettP_aa:8b:be	Spanning-tree-(for- STP	119 MST, Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 228020 Port = 0x8002			
178 76.999834042	HewlettP_aa:8b:be	Spanning-tree-(for- STP	119 MST, Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 228020 Port = 0x8002			
179 77.999888965	HewlettP_aa:8b:be	Spanning-tree-(for- STP	119 MST, Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 228020 Port = 0x8002			
180 81.999802308	HewlettP_aa:8b:be	Spanning-tree-(for- STP	119 MST, Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 228020 Port = 0x8002			
181 83.999531792	HewlettP_aa:8b:be	Spanning-tree-(for- STP	119 MST, Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 228020 Port = 0x8002			
188 85.999529987	HewlettP_aa:8b:be	Spanning-tree-(for- STP	119 MST, Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 228020 Port = 0x8002			
191 87.999791212	HewlettP_aa:8b:be	Spanning-tree-(for- STP	119 MST, Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 228020 Port = 0x8002			
186 90.999834042	HewlettP_aa:8b:be	Spanning-tree-(for- STP	119 MST, Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 228020 Port = 0x8002			
198 93.999871926	HewlettP_aa:8b:be	Spanning-tree-(for- STP	119 MST, Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 228020 Port = 0x8002			
199 95.999796412	HewlettP_aa:8b:be	Spanning-tree-(for- STP	119 MST, Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 228020 Port = 0x8002			
200 97.999834042	HewlettP_aa:8b:be	Spanning-tree-(for- STP	119 MST, Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 228020 Port = 0x8002			
201 98.999834042	HewlettP_aa:8b:be	Spanning-tree-(for- STP	119 MST, Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 228020 Port = 0x8002			
203 101.999558734	HewlettP_aa:8b:be	Spanning-tree-(for- STP	119 MST, Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 228020 Port = 0x8002			
204 103.999773302	HewlettP_aa:8b:be	Spanning-tree-(for- STP	119 MST, Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 228020 Port = 0x8002			
206 105.999642753	HewlettP_aa:8b:be	Spanning-tree-(for- STP	119 MST, Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 228020 Port = 0x8002			
212 108.999246170	HewlettP_aa:8b:be	Spanning-tree-(for- STP	119 MST, Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 228020 Port = 0x8002			
221 111.999584588	HewlettP_aa:8b:be	Spanning-tree-(for- STP	119 MST, Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 228020 Port = 0x8002			
226 113.999732841	HewlettP_aa:8b:be	Spanning-tree-(for- STP	119 MST, Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 228020 Port = 0x8002			
238 115.999658087	HewlettP_aa:8b:be	Spanning-tree-(for- STP	119 MST, Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 228020 Port = 0x8002			
Frame 191: 119 bytes on wire (952 bits), 119 bytes captured (952 bits) on interface enp0s31f6, id 0						
IEEE 802.3 Ethernet						
Destination: Spanning-tree-(for-bridges)_00 (01:00:c2:00:00:00)						
Address: Spanning-tree-(for-bridges)_00 (01:00:c2:00:00:00)						
.... .0. .... .... .... = IG bit: Globally unique address (factory default)						
.... .1. .... .... .... = IG bit: Individual address (unicast)						
Length: 108						
Logical-Link Control						
Spanning Tree Protocol						

**Abbildung 23:** 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`

No.	Time	Source	Destination	Protocol	Length Info
216	110.51588173	linux-3.local	linux-2.local	ICMP	98 Echo (ping) request id=0xc3f seq=1/256 ttl=64 (reply in 216)
221	110.51588173	linux-3.local	linux-2.local	ICMP	98 Echo (ping) reply id=0xc3f seq=2/256 ttl=64 (request in 216)
224	110.547851291	linux-3.local	linux-2.local	ICMP	98 Echo (ping) request id=0xc3f seq=2/512 ttl=64 (reply in 224)
228	110.547828683	linux-3.local	linux-2.local	ICMP	98 Echo (ping) reply id=0xc3f seq=2/512 ttl=64 (request in 219)
222	112.571562475	linux-2.local	linux-3.local	ICMP	98 Echo (ping) request id=0xc3f seq=3/768 ttl=64 (reply in 223)
					Frame 216: 96 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:54:2b) Dst: linux-3.local (4c:52:62:0e:54:2b)
					Address: linux-3.local (4c:52:62:0e:54:2b)
					.....0..... = 16 bit: Globally unique address (factory default)
					.....0..... = 16 bit: Individual address (unicast)
		Source: linux-2.local (4c:52:62:0e:54:2b)			Address: linux-2.local (4c:52:62:0e:54:2b)
					.....0..... = 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
					.0...0.. = Header Length: 20 bytes (5)
					> Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
					Total Length: 84
					Identification: 0xe063 (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 24:** 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`

No.	Time	Source	Destination	Protocol	Length Info
2	0.327088447	rn05.rnlabor.hdm-st. opnsense-router.rnl	ICMP	98 Echo (ping) request id=4/1024 seq=4/1024 ttl=64 (reply in 3)	
4	0.327088447	rn05.rnlabor.hdm-st. opnsense-router.rnl	ICMP	98 Echo (ping) reply id=4/1024 seq=4/1024 ttl=64 (request in 2)	
4	0.935109731	rn05.rnlabor.hdm-st. opnsense-router.rnl	DNS	84 Standard query 0xfdff PTR 5.66.62.141.in-addr.arpa	
5	0.935130905	rn05.rnlabor.hdm-st. opnsense-router.rnl	DNS	86 Standard query 0xb1b9 PTR 250.66.62.141.in-addr.arpa	
6	0.936033635	opnsense-router.rnl	rn05.rnlabor.hdm-st. DNS	127 Standard query response 0xffff PTR 5.66.62.141.in-addr.arpa PTR rn05.rnlabor.hdm-stuttgart.de	
7	0.936034188	opnsense-router.rnl	rn05.rnlabor.hdm-st. DNS	163 Standard query response 0xb1b9 PTR 250.66.62.141.in-addr.arpa PTR opnsense.rnrlabor.hdm-stuttgart.de PTR opnsense-router.rnlabor.hdm-stuttgart.de	
8	1.351378490	opnsense-router.rnl	rn05.rnlabor.hdm-st. ICMP	98 Echo (ping) request id=5/1280 seq=5/1280 ttl=64 (reply in 9)	
11	2.375018675	opnsense-router.rnl	rn05.rnlabor.hdm-st. ICMP	98 Echo (ping) reply id=6/1536 seq=6/1536 ttl=64 (request in 12)	
12	2.375450812	opnsense-router.rnl	rn05.rnlabor.hdm-st. ICMP	98 Echo (ping) reply id=6/1536 seq=6/1536 ttl=64 (request in 11)	
					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 (4c:52:62:0e:54:80) Dst: opnsense-router.rnrlabor.hdm-stuttgart.de (00:0d:b9:4f:b8:14)
					Address: opnsense-router.rnrlabor.hdm-stuttgart.de (00:0d:b9:4f:b8:14)
					.....0..... = 16 bit: Globally unique address (factory default)
		Source: rn05.rnlabor.hdm-stuttgart.de (4c:52:62:0e:54:80)			.....0..... = 16 bit: Individual address (unicast)
					.....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.rnrlabor.hdm-stuttgart.de (141.62.66.250)
					0100 .. = Version: 4
					.0101 = Header Length: 20 bytes (5)
					> Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
					Total Length: 84
					Identification: 0x68d0 (26839)
					> 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.rnrlabor.hdm-stuttgart.de (141.62.66.250)
					> Internet Control Message Protocol

**Abbildung 25:** 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 26:** 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.

No.	Time	Source	Destination	Protocol	Length	Info
393	182.000115690	HewlettP_aa:8b:be	Spanning-tree-(for-. . .)	STP	119	MST. Root = 32768/0:00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
394	184.001659989	HewlettP_aa:8b:be	Spanning-tree-(for-. . .)	STP	119	MST. Root = 32768/0:00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
395	186.000202177	HewlettP_aa:8b:be	Spanning-tree-(for-. . .)	STP	119	MST. Root = 32768/0:00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
397	188.000202636	HewlettP_aa:8b:be	Spanning-tree-(for-. . .)	STP	119	MST. Root = 32768/0:00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
398	190.000136348	HewlettP_aa:8b:be	Spanning-tree-(for-. . .)	STP	119	MST. Root = 32768/0:00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
406	192.000560647	HewlettP_aa:8b:be	Spanning-tree-(for-. . .)	STP	119	MST. Root = 32768/0:00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
407	194.000987110	HewlettP_aa:8b:be	Spanning-tree-(for-. . .)	STP	119	MST. Root = 32768/0:00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
410	196.000987103	HewlettP_aa:8b:be	Spanning-tree-(for-. . .)	STP	119	MST. Root = 32768/0:00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
412	198.0009853659	HewlettP_aa:8b:be	Spanning-tree-(for-. . .)	STP	119	MST. Root = 32768/0:00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
412	200.0009207849	HewlettP_aa:8b:be	Spanning-tree-(for-. . .)	STP	119	MST. Root = 32768/0:00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
413	202.000917163	HewlettP_aa:8b:be	Spanning-tree-(for-. . .)	STP	119	MST. Root = 32768/0:00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
417	209.0009254351	HewlettP_aa:8b:be	Spanning-tree-(for-. . .)	STP	119	MST. Root = 32768/0:00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
420	210.0009015952	HewlettP_aa:8b:be	Spanning-tree-(for-. . .)	STP	119	MST. Root = 32768/0:00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
420	210.0009015953	HewlettP_aa:8b:be	Spanning-tree-(for-. . .)	STP	119	MST. Root = 32768/0:00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
424	210.0009205071	HewlettP_aa:8b:be	Spanning-tree-(for-. . .)	STP	119	MST. Root = 32768/0:00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
425	212.0009277731	HewlettP_aa:8b:be	Spanning-tree-(for-. . .)	STP	119	MST. Root = 32768/0:00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
426	214.001698472	HewlettP_aa:8b:be	Spanning-tree-(for-. . .)	STP	119	MST. Root = 32768/0:00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
427	216.0009676957	HewlettP_aa:8b:be	Spanning-tree-(for-. . .)	STP	119	MST. Root = 32768/0:00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
429	220.0009146054	HewlettP_aa:8b:be	Spanning-tree-(for-. . .)	STP	119	MST. Root = 32768/0:00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
433	222.001177244	HewlettP_aa:8b:be	Spanning-tree-(for-. . .)	STP	119	MST. Root = 32768/0:00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
Frame 426: 119 bytes on wire (952 bits), 119 bytes captured (952 bits) on interface enp0s31f6, id 0						
- IEEE 802.3 Ethernet						
- Destination: Spanning-tree-(for-bridges) 00:01:80:c2:9b:00 (01:80:c2:9b:00:00)						
Address: Spanning-tree-(for-bridges) 00:01:80:c2:9b:00 (factory default)						
..... .1 ..... . . . . = 16 bit: Globally unique address (factory default)						
..... .2 ..... . . . . = 16 bit: Group address (multicast/broadcast)						
- Source: HewlettP_aa:8b:be (04:09:73:aa:8b:be)						
Address: HewlettP_aa:8b:be (04:09:73:aa:8b:be)						
..... .0 ..... . . . . = 16 bit: Globally unique address (factory default)						
..... .1 ..... . . . . = 16 bit: Individual address (unicast)						
Length: 108						
- Logical-Link Control						
- Spanning Tree Protocol						
- Protocol Identifier: Spanning Tree Protocol (0x0000)						
- Port Identifier: Multiple Spanning Tree (3)						
- BPDU type: 0x01 (Multiple Spanning Tree [0x01])						
- BPDU Flags: 0x3e, Forwarding, Learning, Port Role: Designated, Proposal						
- Root Identifier: 32768 / 0 / 0:0:1a:c1:5e:eb:c0						
- Root Path Cost: 220020						
- Port Identifier: 32768 / 0 / 0:4:09:73:aa:8b:be						
- Port Identifier: 0x0002						
- Message Age: 3						
- Max Age: 20						
- Hello Time: 2						
- Forward Delay: 15						
- Version 1 Length: 0						

Abbildung 27: Capture mit Filter für STP

## 2.10 SNMP

### Auf welchen Komponenten im Netzwerk wird das Protokoll SNMP ausgeführt?

Es konnte kein SNMP-Traffic im Netzwerk gefunden werden. SNMP, das Simple Network Management Protocol, wird jedoch meist zur Wartung von verbundenen Geräte im Network verwendet, woraus sich schließen lässt, dass es auf Komponenten wie Switches, Routern oder Servern zum Einsatz kommen würde.

## 2.11 Streaming and Downloads

**Starten Sie einen Download einer größeren Datei aus dem Internet und stoppen Sie ihn während der Übertragung. Dokumentieren Sie, wie der Stop-Befehl innerhalb der Protokolle umgesetzt wird**



**Abbildung 28:** Capture beim Canceln des eines Downloads über HTTPS

Da der Download hier via HTTPS durchgeführt wurde, kann erkannt werden, dass die darunterliegenden TCP-Verbindungen 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 29:** 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.

No.	Time	Source	Destination	Protocol	Length	Info
53	13.371889779	141.62.66.5	141.62.66.207	TELNET	69	Telnet Data ...
55	13.371964177	141.62.66.207	141.62.66.5	TELNET	69	Telnet Data ...
57	13.372108043	141.62.66.5	141.62.66.207	TELNET	69	Telnet Data ...
58	13.372142487	141.62.66.207	141.62.66.5	TELNET	86	Telnet Data ...
65	15.536484821	141.62.66.5	141.62.66.207	TELNET	67	Telnet Data ...
67	15.537258875	141.62.66.207	141.62.66.5	TELNET	67	Telnet Data ...
69	15.712433767	141.62.66.5	141.62.66.207	TELNET	67	Telnet Data ...
71	15.713143086	141.62.66.207	141.62.66.5	TELNET	67	Telnet Data ...
73	15.716452953	141.62.66.5	141.62.66.207	TELNET	67	Telnet Data ...
75	15.718404249	141.62.66.207	141.62.66.5	TELNET	67	Telnet Data ...
76	15.864389554	141.62.66.5	141.62.66.207	TELNET	67	Telnet Data ...
77	15.865998282	141.62.66.207	141.62.66.5	TELNET	67	Telnet Data ...
79	15.991754757	141.62.66.5	141.62.66.207	TELNET	67	Telnet Data ...
80	15.992584487	141.62.66.207	141.62.66.5	TELNET	67	Telnet Data ...
82	15.993360860	141.62.66.5	141.62.66.207	TELNET	67	Telnet Data ...
84	15.994157037	141.62.66.207	141.62.66.5	TELNET	67	Telnet Data ...
86	15.176491685	141.62.66.5	141.62.66.207	TELNET	67	Telnet Data ...
87	16.177306417	141.62.66.207	141.62.66.5	TELNET	67	Telnet Data ...
89	16.344425688	141.62.66.5	141.62.66.207	TELNET	67	Telnet Data ...
90	16.345381998	141.62.66.207	141.62.66.5	TELNET	67	Telnet Data ...

Frame 98: 76 bytes on wire (608 bits), 76 bytes captured (608 bits) on interface enp0s31f6, id 0  
 • Ethernet II, Src: 62:39:f6:b0:b9:87 (62:39:f6:b0:b9:87), Dst: rnlabor.hdm-stuttgart.de (4c:52:62:0e:54:8b)  
 • Internet Protocol Version 4, Src: 141.62.66.207, Dst: 141.62.66.5  
 • Transmission Control Protocol, Src Port: 23, Dst Port: 36234, Seq: 78, Ack: 163, Len: 14  
 - Telnet  
 Data: telnet login:

**Abbildung 30:** Capture des Telnet-Logins

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

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

No.	Time	Source	Destination	Protocol	Length	Info
77	15.865989282	141.62.66.207	141.62.66.5	TELNET	67	Telnet Data ...
79	15.891754757	141.62.66.5	141.62.66.207	TELNET	67	Telnet Data ...
80	15.992584487	141.62.66.207	141.62.66.5	TELNET	67	Telnet Data ...
82	16.056360860	141.62.66.5	141.62.66.207	TELNET	67	Telnet Data ...
83	16.057278317	141.62.66.207	141.62.66.5	TELNET	67	Telnet Data ...
86	16.176491685	141.62.66.5	141.62.66.207	TELNET	67	Telnet Data ...
87	16.177306417	141.62.66.207	141.62.66.5	TELNET	67	Telnet Data ...
89	16.344425688	141.62.66.5	141.62.66.207	TELNET	67	Telnet Data ...
90	16.345381998	141.62.66.207	141.62.66.5	TELNET	67	Telnet Data ...
92	16.528454533	141.62.66.5	141.62.66.207	TELNET	67	Telnet Data ...
93	16.529374168	141.62.66.207	141.62.66.5	TELNET	67	Telnet Data ...
95	17.181471398	141.62.66.5	141.62.66.207	TELNET	68	Telnet Data ...
96	17.183995600	141.62.66.207	141.62.66.5	TELNET	69	Telnet Data ...
98	17.183995601	141.62.66.207	141.62.66.5	TELNET	76	Telnet Data ...
101	19.152499070	141.62.66.5	141.62.66.207	TELNET	67	Telnet Data ...
103	19.344388216	141.62.66.5	141.62.66.207	TELNET	67	Telnet Data ...
105	19.401747844	141.62.66.5	141.62.66.207	TELNET	67	Telnet Data ...
106	19.404738844	141.62.66.5	141.62.66.207	TELNET	67	Telnet Data ...
109	19.688402452	141.62.66.5	141.62.66.207	TELNET	67	Telnet Data ...
111	19.816961612	141.62.66.5	141.62.66.207	TELNET	67	Telnet Data ...
113	19.912438966	141.62.66.5	141.62.66.207	TELNET	67	Telnet Data ...

Frame 98: 76 bytes on wire (608 bits), 76 bytes captured (608 bits) on interface enp0s31f6, id 0  
 • Ethernet II, Src: 62:39:f6:b0:b9:87 (62:39:f6:b0:b9:87), Dst: rnlabor.hdm-stuttgart.de (4c:52:62:0e:54:8b)  
 • Internet Protocol Version 4, Src: 141.62.66.207, Dst: 141.62.66.5  
 • Transmission Control Protocol, Src Port: 23, Dst Port: 36234, Seq: 103, Ack: 174, Len: 10  
 - Telnet  
 Data: Password:

**Abbildung 31:** Capture des Telnet-Passwords

No.	Time	Source	Destination	Protocol	Length	Info
77	15.865988282	141.62.66.207	141.62.66.5	TELNET	67	Telnet Data ...
78	15.891754757	141.62.66.5	141.62.66.207	TELNET	67	Telnet Data ...
80	15.992584487	141.62.66.207	141.62.66.5	TELNET	67	Telnet Data ...
82	16.056360688	141.62.66.5	141.62.66.207	TELNET	67	Telnet Data ...
83	16.057278313	141.62.66.207	141.62.66.5	TELNET	67	Telnet Data ...
87	16.119205505	141.62.66.207	141.62.66.5	TELNET	67	Telnet Data ...
89	16.177386417	141.62.66.207	141.62.66.5	TELNET	67	Telnet Data ...
90	16.345301998	141.62.66.207	141.62.66.5	TELNET	67	Telnet Data ...
92	16.528454531	141.62.66.207	141.62.66.5	TELNET	67	Telnet Data ...
93	16.529374164	141.62.66.207	141.62.66.5	TELNET	67	Telnet Data ...
95	16.530119625	141.62.66.207	141.62.66.5	TELNET	67	Telnet Data ...
97	17.193284469	141.62.66.207	141.62.66.5	TELNET	68	Telnet Data ...
98	17.193599591	141.62.66.207	141.62.66.5	TELNET	76	Telnet Data ...
101	19.152490870	141.62.66.207	141.62.66.207	TELNET	67	Telnet Data ...
103	19.344388219	141.62.66.5	141.62.66.207	TELNET	67	Telnet Data ...
104	19.401116205	141.62.66.5	141.62.66.207	TELNET	67	Telnet Data ...
107	19.410478844	141.62.66.5	141.62.66.207	TELNET	67	Telnet Data ...
109	19.689402452	141.62.66.5	141.62.66.207	TELNET	67	Telnet Data ...
111	19.816961616	141.62.66.5	141.62.66.207	TELNET	67	Telnet Data ...
113	19.912438996	141.62.66.5	141.62.66.207	TELNET	67	Telnet Data ...

Frame 101: 67 bytes on wire (536 bits), 67 bytes captured (536 bits) on interface enp0s31f6, id 8  
 Ethernet II, Src: rn05.rnlabor.hdm-stuttgart.de (4c:52:62:0e:54:8b), Dst: 62:39:f6:7b:b0:87 (62:39:f6:7b:b0:87)  
 Internet Protocol Version 4, Src: 141.62.66.5, Dst: 141.62.66.207  
 Transmission Control Protocol, Src Port: 30234, Dst Port: 23, Seq: 174, Ack: 113, Len: 1  
 Telnet  
 Data: v

**Abbildung 32:** Capture eines Charakters des Telnet-Passwords**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.

No.	Time	Source	Destination	Protocol	Length	Info
202	65.784967321	138.68.70.72	141.62.66.5	SSH	126	Server: Encrypted packet (len=60)
204	65.784229966	141.62.66.5	138.68.70.72	SSH	102	Client: Encrypted packet (len=36)
279	119.032310634	138.68.70.72	141.62.66.5	SSH	126	Server: Encrypted packet (len=60)
319	119.032477959	141.62.66.5	138.68.70.72	SSH	102	Client: Encrypted packet (len=36)
459	177.247707789	141.62.66.5	138.68.70.72	SSH	142	Client: Encrypted packet (len=50)
440	174.282509357	138.68.70.72	141.62.66.5	SSH	108	Server: Encrypted packet (len=132)
448	177.2240986626	141.62.66.5	138.68.70.72	SSH	158	Client: Encrypted packet (len=92)
450	177.230618026	138.68.70.72	141.62.66.5	SSH	182	Server: Encrypted packet (len=116)
452	177.237044982	141.62.66.5	138.68.70.72	SSH	126	Client: Encrypted packet (len=60)
453	177.237128457	141.62.66.5	138.68.70.72	SSH	126	Client: Encrypted packet (len=60)
454	177.237144747	141.62.66.5	138.68.70.72	SSH	126	Client: Encrypted packet (len=60)
458	177.243289805	138.68.70.72	141.62.66.5	SSH	206	Server: Encrypted packet (len=140)
460	177.244314401	141.62.66.5	138.68.70.72	SSH	119	Client: Encrypted packet (len=52)
461	177.259592845	138.68.70.72	141.62.66.5	SSH	1514	Server: Encrypted packet (len=1448)
463	177.259594712	138.68.70.72	141.62.66.5	SSH	662	Server: Encrypted packet (len=796)
464	177.259594712	141.62.66.5	138.68.70.72	SSH	118	Client: Encrypted packet (len=52)
465	177.259594712	141.62.66.5	138.68.70.72	SSH	141	Server: Encrypted packet (len=52)
467	177.258776376	141.62.66.5	138.68.70.72	SSH	119	Client: Encrypted packet (len=52)
468	177.264904430	138.68.70.72	141.62.66.5	SSH	134	Server: Encrypted packet (len=68)
469	177.285330770	141.62.66.5	138.68.70.72	SSH	118	Client: Encrypted packet (len=52)
470	177.285533968	141.62.66.5	138.68.70.72	SSH	118	Client: Encrypted packet (len=52)

Frame 101: 126 bytes on wire (1008 bits), 126 bytes captured (1008 bits) on interface enp0s31f6, id 8  
 Ethernet II, Src: openwrt (00:0d:b9:4f:bb:14), Dst: rn05.rnlabor.hdm-stuttgart.de (4c:52:62:0e:54:8b)  
 Internet Protocol Version 4, Src: 138.68.70.72, Dst: 141.62.66.5  
 Transmission Control Protocol, Src Port: 22, Dst Port: 47840, Seq: 1, Ack: 1, Len: 60  
 SSH Protocol  
 Packet Length (encrypted): 9080f09e4  
 Encrypted Packet: 6bcbb15349d582f55930da2caccb0c73e84abeb992378514580fe2c0b2d9dab4f820ad3e...  
 [Direction: server-to-client]

**Abbildung 33:** Capture eines verschlüsselten SSH-Pakets



**Beim Login-Versuch auf `ftp.bellevue.de` mit von Ihnen wählbaren Account-Daten nur Rahmen herausfiltern, die das gewählte Passwort im Ethernet-Datenfeld enthalten**

Mittels des Filters `ftp.request.command == "PASS"` werden nur Pakete angezeigt, welche das Passwort enthalten.

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:54:8b), 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, Src Port: 51798, Dst Port: 21, Seq: 13, Ack: 57, Len: 23  
 File Transfer Protocol (FTP)  
 [Current working directory:]

**Abbildung 36:** 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 `tcp.port == 80` hätte auch noch UDP-Traffic auf diesem Port dargestellt werden können.

**Abbildung 37:** 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 38:** Capture aller IP-Pakete mit Multicast-Adressen