
Praktikum Rechnernetze

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

Jakob Waibel, Daniel Hiller, Elia Wüstner, Felicitas
Pojtinger

2021-10-26

Inhaltsverzeichnis

1 Einführung	2
1.1 Mitwirken	2
1.2 Lizenz	2
2 Wireshark	3
2.1 Einführung	3
2.2 Ping	5
2.3 DHCP	6
2.4 DNS	8
2.5 ARP	9
2.6 Layer-2-Protokolle	11
2.7 HTTP und TCP	13
2.8 MAC	17
2.9 STP	19
2.10 SNMP	20
2.11 Streaming and Downloads	21
2.12 Telnet und SSH	22
2.13 Wireshark-Filter	24

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):



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, Felicitas 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 capturen Sie den aktuellen Traffic. Dokumentieren Sie zunächst, was alles auf Wireshark einprasselt.

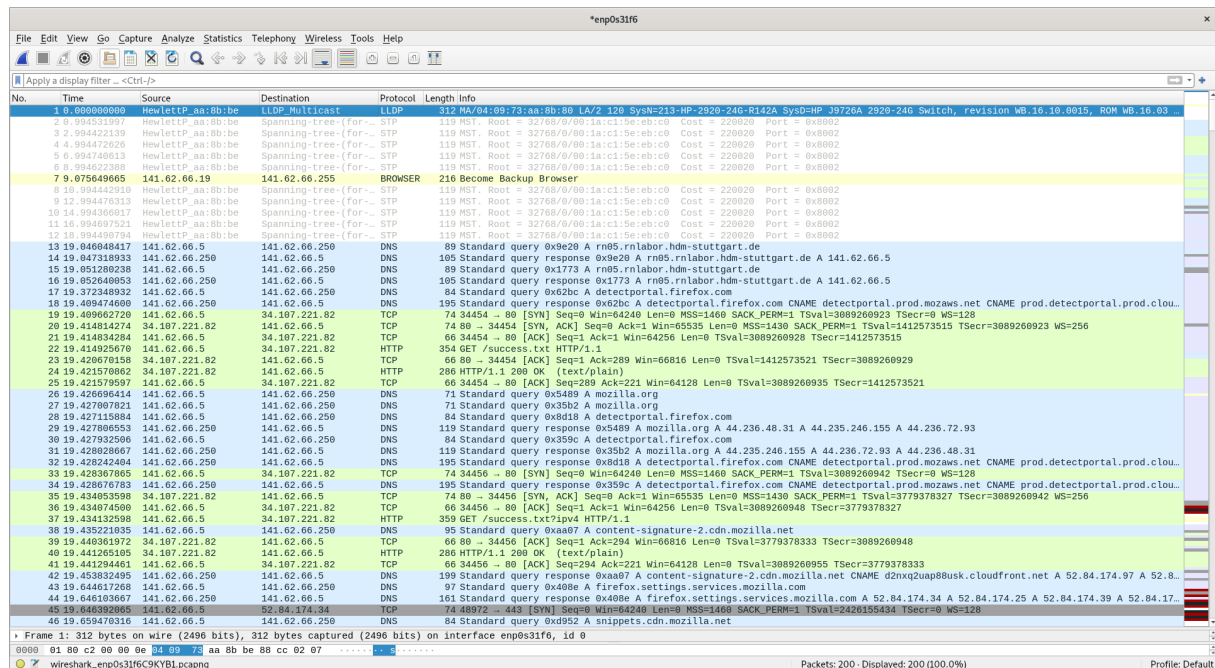


Abbildung 3: Screenshot von Wireshark

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
    
```

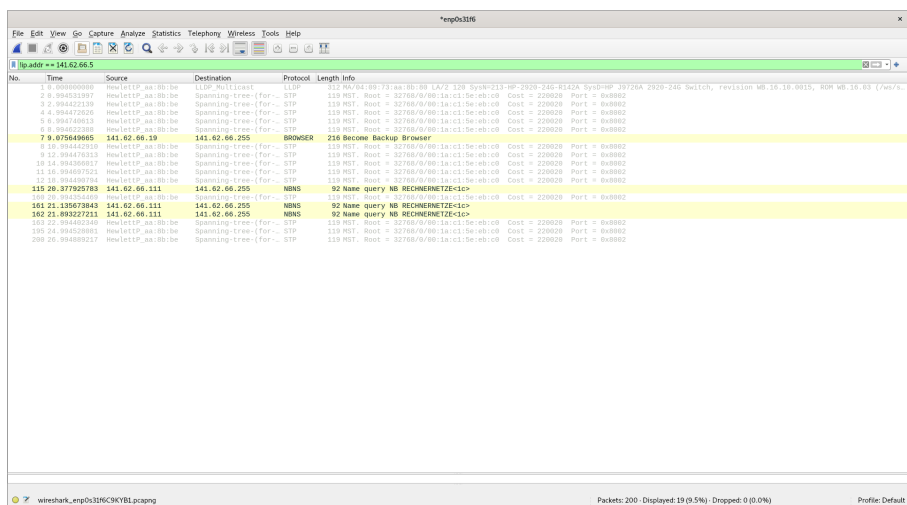


Abbildung 4: Ausklammern der eig. IP, Option 1

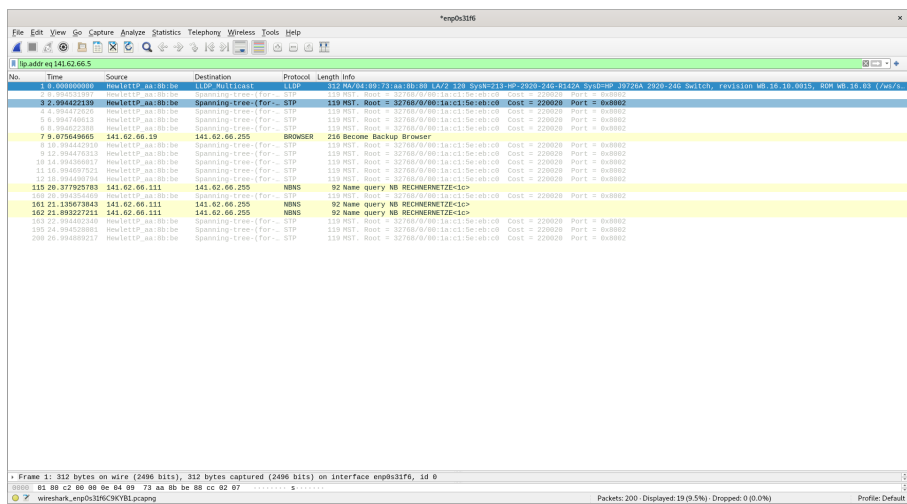
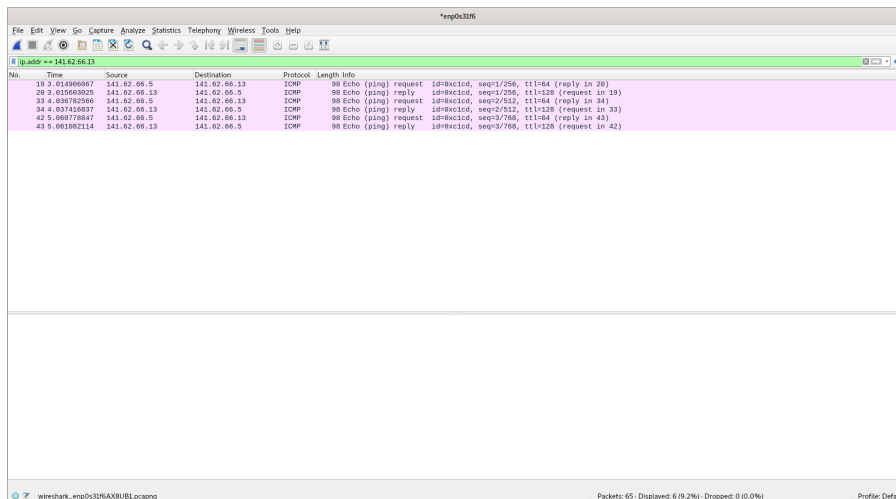


Abbildung 5: Ausklammern der eig. IP, Option 2

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:



The image shows a Wireshark capture window titled '*emp03116'. The filter bar shows 'ip.addr == 141.62.66.13'. The packet list pane displays six ICMP Echo (ping) packets. The first three are requests from 141.62.66.5 to 141.62.66.13, and the last three are replies from 141.62.66.13 to 141.62.66.5. The packet details pane is empty.

No.	Time	Source	Destination	Protocol	Length	Info
19	0.014066987	141.62.66.5	141.62.66.13	ICMP	98	Echo (ping) request id=8xc1cd, seq=1/256, ttl=64 (reply in 26)
20	0.015683920	141.62.66.13	141.62.66.5	ICMP	98	Echo (ping) reply id=8xc1cd, seq=1/256, ttl=328 (request in 19)
33	4.000782066	141.62.66.5	141.62.66.13	ICMP	98	Echo (ping) request id=8xc1cd, seq=2/512, ttl=64 (reply in 34)
34	4.007416837	141.62.66.13	141.62.66.5	ICMP	98	Echo (ping) reply id=8xc1cd, seq=2/512, ttl=328 (request in 33)
42	5.000719847	141.62.66.5	141.62.66.13	ICMP	98	Echo (ping) request id=8xc1cd, seq=3/768, ttl=64 (reply in 43)
43	5.001882114	141.62.66.13	141.62.66.5	ICMP	98	Echo (ping) reply id=8xc1cd, seq=3/768, ttl=328 (request in 42)

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` zugeordnet werden können.

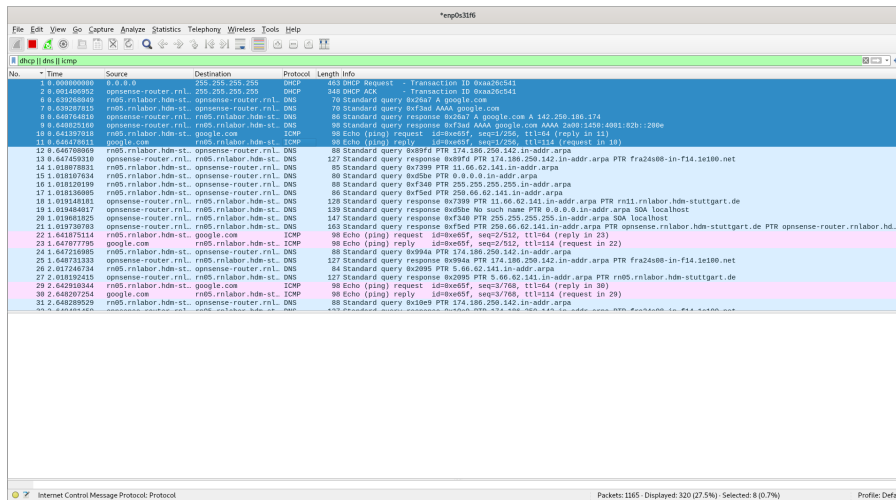


Abbildung 7: Wireshark-Output zu einem Ping nach google.com

Eine beliebige nicht existierenden IP-Adresse

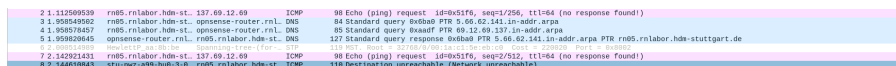


Abbildung 8: Wireshark-Output zu einem Ping nach 137.69.12.69

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 Netzwerk, den der PC bis zum Erhalt der IP-Adresse erzeugt.

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

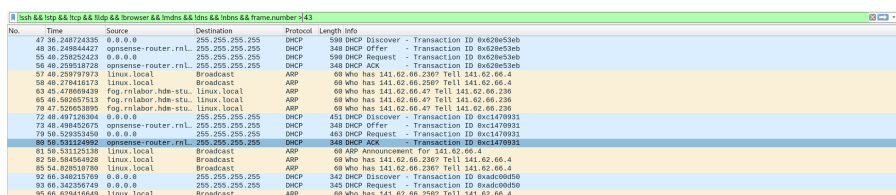


Abbildung 9: Gesamter Bootprozess

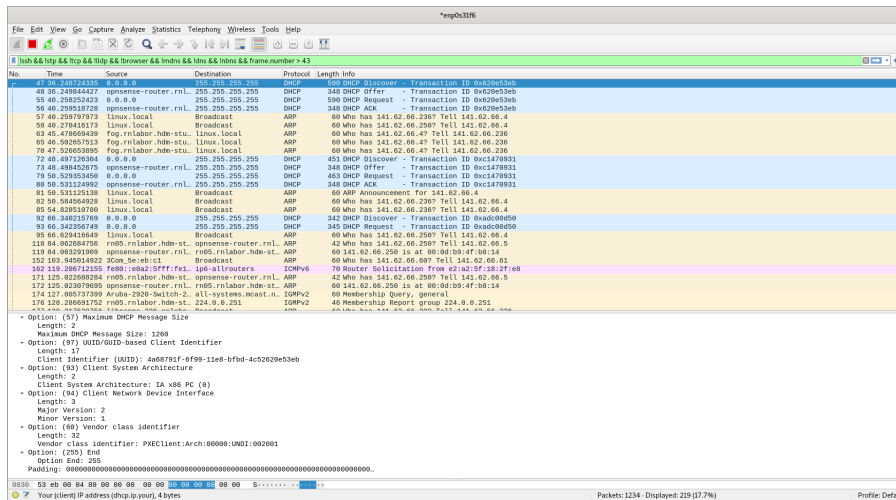


Abbildung 10: Bootprozess: DHCP-Requests des BIOS zum Netzwerkboot, damit der Netzwerkbootloader über i.e. TFTP geladen werden kann

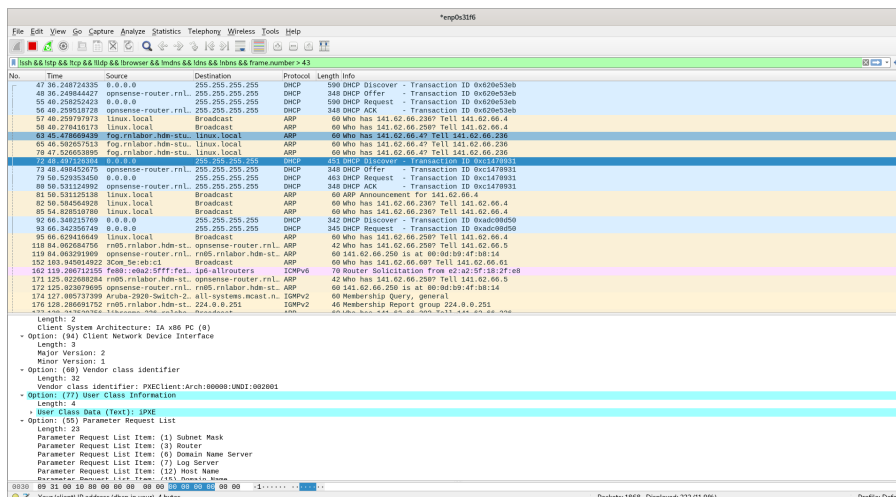


Abbildung 11: Bootprozess: DHCP-Requests des Netzwerkbootloaders iPXE

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.

Vergleichen Sie den Ablauf, wenn Sie den DHCP-Ablauf per `ipconfig /release` und `ipconfig /renew` initiieren

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
19	15.392048961	0.0.0.0	255.255.255.255	DHCP	342	DHCP Discover - Transaction ID 0x78e783d
20	15.393317129	0.0.0.0	255.255.255.255	DHCP	342	DHCP Request - Transaction ID 0x78e783d
21	15.408881886	linux.local	Broadcast	ARP	60	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 getätigt 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	1.357359890	rn05.rnlabor.hdm-st...	opnsense-router.rnl...	DNS	93	Standard query 0xa276 A google.com OPT
12	1.371692978	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`.

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	0.005926935	one.one.one.one	rn05.rnlabor.hdm-st...	DNS	97	Standard query response 0x6247 A google.com A 142.250.185.110 OPT
4	1.205920780	rn05.rnlabor.hdm-st...	opnsense-router.rnl...	DNS	84	Standard query 0xd82b PTR 5.66.62.141.in-addr.arpa
5	1.205949397	rn05.rnlabor.hdm-st...	opnsense-router.rnl...	DNS	88	Standard query 0x8083 PTR 1.1.1.1.in-addr.arpa
6	1.207179251	opnsense-router.rnl...	rn05.rnlabor.hdm-st...	DNS	127	Standard query response 0xda2b PTR 5.66.62.141.in-addr.arpa PTR rn05.rnlabor.hdm-stuttgart.de
7	1.207611330	opnsense-router.rnl...	rn05.rnlabor.hdm-st...	DNS	109	Standard query response 0x8083 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	0.572490372	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...	DNS	84	Standard query 0xc6e8 PTR 5.66.62.141.in-addr.arpa
6	1.088465401	rn05.rnlabor.hdm-st...	opnsense.rnlabor.hdm...	DNS	80	Standard query 0x74b6 PTR 9.8.8.8.in-addr.arpa
7	1.089961823	opnsense.rnlabor.hdm...	rn05.rnlabor.hdm-st...	DNS	127	Standard query response 0xc6e8 PTR 5.66.62.141.in-addr.arpa PTR rn05.rnlabor.hdm-stuttgart.de
8	1.090926625	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	2.087996007	rn05.rnlabor.hdm-st...	opnsense.rnlabor.hdm...	DNS	86	Standard query 0x4f6b PTR 250.66.62.141.in-addr.arpa
17	2.089209013	opnsense.rnlabor.hdm...	rn05.rnlabor.hdm-st...	DNS	103	Standard query response 0x4f6b PTR 250.66.62.141.in-addr.arpa PTR opnsense-router.rnlabor.hdm-stuttgart.de PTR opnsense.rnlabor.hdm...
22	3.087910068	rn05.rnlabor.hdm-st...	opnsense.rnlabor.hdm...	DNS	86	Standard query 0x050b PTR 19.75.254.169.in-addr.arpa
23	3.087945863	rn05.rnlabor.hdm-st...	opnsense.rnlabor.hdm...	DNS	84	Standard query 0xfec6 PTR 251.0.0.224.in-addr.arpa
24	3.087950319	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 0x050b No such name PTR 19.75.254.169.in-addr.arpa SOA localhost
26	3.089611764	opnsense.rnlabor.hdm...	rn05.rnlabor.hdm-st...	DNS	141	Standard query response 0xfec6 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	110.515076213	linux-2.local	Broadcast	ARP	42	Who has 141.62.66.6? Tell 141.62.66.5
215	110.515076208	linux-3.local	linux-2.local	ARP	60	141.62.66.0 is at 4c:52:62:0e:54:2b
231	115.673164735	linux-3.local	linux-2.local	ARP	60	Who has 141.62.66.5? Tell 141.62.66.6
232	115.673186783	linux-2.local	linux-3.local	ARP	42	141.62.66.5 is at 4c:52:62:0e: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	119.515578213	linux-2.local	Broadcast	ARP	42	Who has 141.62.66.6? Tell 141.62.66.5
215	119.51567288	linux-3.local	linux-2.local	ARP	60	141.62.66.6 is at 4c:52:62:0e:54:2b
231	115.673164735	linux-3.local	linux-2.local	ARP	60	Who has 141.62.66.5? Tell 141.62.66.6
232	115.673166793	linux-2.local	linux-3.local	ARP	42	141.62.66.5 is at 4c:52:62:0e:54:8b

* Frame 214: 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
  
```

2.6 Layer-2-Protokolle

Gelegentlich werden vom Analyzer Broadcasts erkannt. Wer sendet sie, warum und in welchen zeitlichen Abständen?

Die Broadcasts sind ARP-Requests. Sie entstehen dadurch, da Geräte versuchen Daten an andere Geräte zu übertragen, für welche sie keinen Eintrag in ihrem ARP-Cache haben, deshalb muss eine ARP-Anfrage in Form eines Broadcasts gesendet werden, da jeder Host potenziell der gesuchte Host sein kann. Dieser besitzt gesuchte IP X und antwortet daraufhin mit seiner Mac.

No.	Time	Source	Destination	Protocol	Length	Info
173	76.898137336	HewlettP_aa:8b:be	Spanning-tree-(for... STP	STP	119	MST, Root = 32768/0/0:1a:c1:5e:b:c0 Cost = 220928 Port = 0x8002
174	71.99955778	HewlettP_aa:8b:be	Spanning-tree-(for... STP	STP	119	MST, Root = 32768/0/0:1a:c1:5e:b:c0 Cost = 220928 Port = 0x8002
175	72.666751597	linux-3.local	224.0.0.251	MDNS	82	Standard query 0x0000 PTR _pgpkey-hkp._tcp.local, "QM" question
176	73.999729543	HewlettP_aa:8b:be	Spanning-tree-(for... STP	STP	119	MST, Root = 32768/0/0:1a:c1:5e:b:c0 Cost = 220928 Port = 0x8002
177	75.999566599	HewlettP_aa:8b:be	Spanning-tree-(for... STP	STP	119	MST, Root = 32768/0/0:1a:c1:5e:b:c0 Cost = 220928 Port = 0x8002
178	77.999639982	HewlettP_aa:8b:be	Spanning-tree-(for... STP	STP	119	MST, Root = 32768/0/0:1a:c1:5e:b:c0 Cost = 220928 Port = 0x8002
179	79.999889965	HewlettP_aa:8b:be	Spanning-tree-(for... STP	STP	119	MST, Root = 32768/0/0:1a:c1:5e:b:c0 Cost = 220928 Port = 0x8002
180	81.999662388	HewlettP_aa:8b:be	Spanning-tree-(for... STP	STP	119	MST, Root = 32768/0/0:1a:c1:5e:b:c0 Cost = 220928 Port = 0x8002
181	83.999531792	HewlettP_aa:8b:be	Spanning-tree-(for... STP	STP	119	MST, Root = 32768/0/0:1a:c1:5e:b:c0 Cost = 220928 Port = 0x8002
182	84.699549741	librenms-226.rnlabo	Broadcast	ARP	60	Who has 141.62.66.227? Tell 141.62.66.226
183	84.731177879	librenms-226.rnlabo	Broadcast	ARP	60	Who has 141.62.66.227? Tell 141.62.66.226
184	85.697465721	librenms-226.rnlabo	Broadcast	ARP	60	Who has 141.62.66.227? Tell 141.62.66.226
185	85.761491538	librenms-226.rnlabo	Broadcast	ARP	60	Who has 141.62.66.227? Tell 141.62.66.226
186	85.954876527	linux-2.local	opnsense.rnlabor.hd	DNS	86	Standard query 0x9e2a PTR 226.66.62.141.in-addr.arpa
187	85.955623699	opnsense.rnlabor.hd	Linux-2.local	DNS	137	Standard query response 0x9e2a PTR 226.66.62.141.in-addr.arpa PTR librenms-226.rnlabor.hdm-stuttgart.de
188	85.99923694	HewlettP_aa:8b:be	Spanning-tree-(for... STP	STP	119	MST, Root = 32768/0/0:1a:c1:5e:b:c0 Cost = 220928 Port = 0x8002
189	86.721454740	librenms-226.rnlabo	Broadcast	ARP	60	Who has 141.62.66.227? Tell 141.62.66.226
190	86.785487391	librenms-226.rnlabo	Broadcast	ARP	60	Who has 141.62.66.227? Tell 141.62.66.226
191	87.999791212	HewlettP_aa:8b:be	Spanning-tree-(for... STP	STP	119	MST, Root = 32768/0/0:1a:c1:5e:b:c0 Cost = 220928 Port = 0x8002
192	88.628764598	linux-3.local	224.0.0.251	MDNS	81	Standard query 0x0000 PTR _nmea-0183._tcp.local, "QM" question
193	89.99989785	HewlettP_aa:8b:be	Spanning-tree-(for... STP	STP	119	MST, Root = 32768/0/0:1a:c1:5e:b:c0 Cost = 220928 Port = 0x8002
194	91.067595494	linux-2.local	opnsense.rnlabor.hd	ARP	42	Who has 141.62.66.250? Tell 141.62.66.5
195	91.069717280	opnsense.rnlabor.hd	Linux-2.local	ARP	60	141.62.66.250 is at 00:0d:b9:4f:b8:14
196	91.999634632	HewlettP_aa:8b:be	Spanning-tree-(for... STP	STP	119	MST, Root = 32768/0/0:1a:c1:5e:b:c0 Cost = 220928 Port = 0x8002
197	93.065371535	HewlettP_aa:8b:be	LLDP_Multicast	LLDP	312	MA/04:09:73:aa:8b:80 LA/2 120 sysn=213-HP-2928-24G-R142A SysDPHP J9726A 2928-24G Switch, revision WB.16.10.0015, ROM WB.16.83
198	93.99981928	HewlettP_aa:8b:be	Spanning-tree-(for... STP	STP	119	MST, Root = 32768/0/0:1a:c1:5e:b:c0 Cost = 220928 Port = 0x8002
199	95.999798412	HewlettP_aa:8b:be	Spanning-tree-(for... STP	STP	119	MST, Root = 32768/0/0:1a:c1:5e:b:c0 Cost = 220928 Port = 0x8002

Abbildung 17: Aufzeichnung der ARP-Requests

Haben Sie noch weitere Protokolle "eingefangen", die offensichtlich im Labor Rechnernetze keinen Sinn machen?

Aus dem Screenshot lässt sich aus der MDNS-Nachricht der `_nmea-0183._tcp.local` Service-String entnehmen. NMEA 0183 ist ein Standard, welcher für die Kommunikation zwischen Navigationsgeräten auf Schiffen definiert wurde. Da es mitunter für die Kommunikation zwischen GPS-Empfänger und PCs verwendet wird, macht es in unserem Netzwerk wenig Sinn.

No.	Time	Source	Destination	Protocol	Length	Info
173	76.898137336	HewlettP_aa:8b:be	Spanning-tree-(for... STP	STP	119	MST, Root = 32768/0/0:1a:c1:5e:b:c0 Cost = 220928 Port = 0x8002
174	71.99955778	HewlettP_aa:8b:be	Spanning-tree-(for... STP	STP	119	MST, Root = 32768/0/0:1a:c1:5e:b:c0 Cost = 220928 Port = 0x8002
175	72.666751597	linux-3.local	224.0.0.251	MDNS	82	Standard query 0x0000 PTR _pgpkey-hkp._tcp.local, "QM" question
176	73.999729543	HewlettP_aa:8b:be	Spanning-tree-(for... STP	STP	119	MST, Root = 32768/0/0:1a:c1:5e:b:c0 Cost = 220928 Port = 0x8002
177	75.999566599	HewlettP_aa:8b:be	Spanning-tree-(for... STP	STP	119	MST, Root = 32768/0/0:1a:c1:5e:b:c0 Cost = 220928 Port = 0x8002
178	77.999639982	HewlettP_aa:8b:be	Spanning-tree-(for... STP	STP	119	MST, Root = 32768/0/0:1a:c1:5e:b:c0 Cost = 220928 Port = 0x8002
179	79.999889965	HewlettP_aa:8b:be	Spanning-tree-(for... STP	STP	119	MST, Root = 32768/0/0:1a:c1:5e:b:c0 Cost = 220928 Port = 0x8002
180	81.999662388	HewlettP_aa:8b:be	Spanning-tree-(for... STP	STP	119	MST, Root = 32768/0/0:1a:c1:5e:b:c0 Cost = 220928 Port = 0x8002
181	83.999531792	HewlettP_aa:8b:be	Spanning-tree-(for... STP	STP	119	MST, Root = 32768/0/0:1a:c1:5e:b:c0 Cost = 220928 Port = 0x8002
182	84.699549741	librenms-226.rnlabo	Broadcast	ARP	60	Who has 141.62.66.227? Tell 141.62.66.226
183	84.731177879	librenms-226.rnlabo	Broadcast	ARP	60	Who has 141.62.66.227? Tell 141.62.66.226
184	85.697465721	librenms-226.rnlabo	Broadcast	ARP	60	Who has 141.62.66.227? Tell 141.62.66.226
185	85.761491538	librenms-226.rnlabo	Broadcast	ARP	60	Who has 141.62.66.227? Tell 141.62.66.226
186	85.954876527	linux-2.local	opnsense.rnlabor.hd	DNS	86	Standard query 0x9e2a PTR 226.66.62.141.in-addr.arpa
187	85.955623699	opnsense.rnlabor.hd	Linux-2.local	DNS	137	Standard query response 0x9e2a PTR 226.66.62.141.in-addr.arpa PTR librenms-226.rnlabor.hdm-stuttgart.de
188	85.99923694	HewlettP_aa:8b:be	Spanning-tree-(for... STP	STP	119	MST, Root = 32768/0/0:1a:c1:5e:b:c0 Cost = 220928 Port = 0x8002
189	86.721454740	librenms-226.rnlabo	Broadcast	ARP	60	Who has 141.62.66.227? Tell 141.62.66.226
190	86.785487391	librenms-226.rnlabo	Broadcast	ARP	60	Who has 141.62.66.227? Tell 141.62.66.226
191	87.999791212	HewlettP_aa:8b:be	Spanning-tree-(for... STP	STP	119	MST, Root = 32768/0/0:1a:c1:5e:b:c0 Cost = 220928 Port = 0x8002
192	88.628764598	linux-3.local	224.0.0.251	MDNS	81	Standard query 0x0000 PTR _nmea-0183._tcp.local, "QM" question
193	89.99989785	HewlettP_aa:8b:be	Spanning-tree-(for... STP	STP	119	MST, Root = 32768/0/0:1a:c1:5e:b:c0 Cost = 220928 Port = 0x8002
194	91.067595494	linux-2.local	opnsense.rnlabor.hd	ARP	42	Who has 141.62.66.250? Tell 141.62.66.5
195	91.069717280	opnsense.rnlabor.hd	Linux-2.local	ARP	60	141.62.66.250 is at 00:0d:b9:4f:b8:14
196	91.999634632	HewlettP_aa:8b:be	Spanning-tree-(for... STP	STP	119	MST, Root = 32768/0/0:1a:c1:5e:b:c0 Cost = 220928 Port = 0x8002
197	93.065371535	HewlettP_aa:8b:be	LLDP_Multicast	LLDP	312	MA/04:09:73:aa:8b:80 LA/2 120 sysn=213-HP-2928-24G-R142A SysDPHP J9726A 2928-24G Switch, revision WB.16.10.0015, ROM WB.16.83
198	93.99981928	HewlettP_aa:8b:be	Spanning-tree-(for... STP	STP	119	MST, Root = 32768/0/0:1a:c1:5e:b:c0 Cost = 220928 Port = 0x8002
199	95.999798412	HewlettP_aa:8b:be	Spanning-tree-(for... STP	STP	119	MST, Root = 32768/0/0:1a:c1:5e:b:c0 Cost = 220928 Port = 0x8002

Abbildung 18: Aufzeichnung der ARP-Requests; hier ist das Protokoll zu sehen

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.

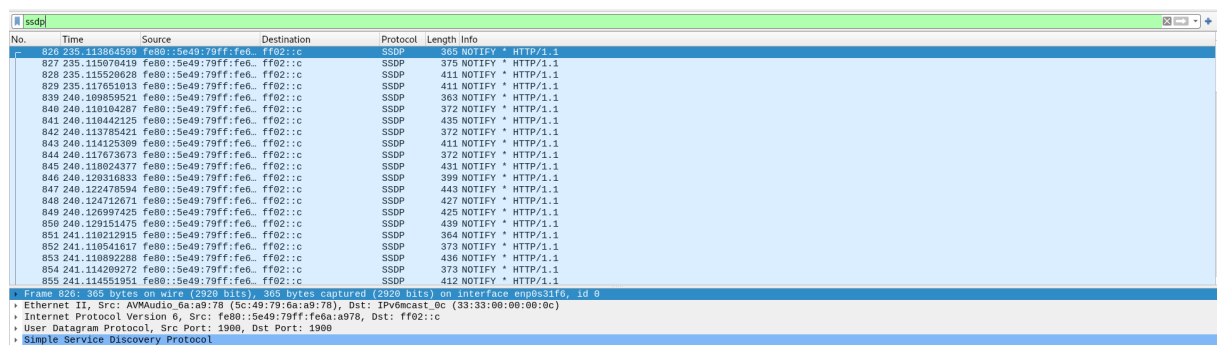


Abbildung 19: Aufzeichnung des SSDP-Protokolls

2.7 HTTP und TCP

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

Zuerst wird ein DNS-Request getätigt. Daraufhin folgt der 3-Way-Handshake. Dieser ist an der charakteristischen Abfolge SYN, SYN-ACK, ACK zu erkennen.

No.	Time	Source	Destination	Protocol	Length	Info
714	7.598625	100.64.84.66	141.70.124.5	DNS	80	Standard query 0x189d A news.ycombinator.com
715	7.598881	100.64.84.66	141.70.124.5	DNS	80	Standard query 0x58df AAAA news.ycombinator.com
716	7.608834	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...
718	7.614306	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 TSval=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 TSval=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 TSval=2512581211 TSecr=2045828460
721	7.765826	100.64.84.66	209.216.230.240	TLSh1..	583	Client Hello
722	7.917493	209.216.230.240	100.64.84.66	TLSh1..	1514	Server Hello
723	7.917494	209.216.230.240	100.64.84.66	TCP	1514	443 → 49314 [ACK] Seq=1449 Ack=518 Win=65664 Len=1448 TSval=2045828612 TSecr=2512581211 [TCP segment of a reassembled PDU]
724	7.917495	209.216.230.240	100.64.84.66	TLSh1..	1062	Certificate, Certificate Status, Server Key Exchange, Server Hello Done
725	7.917501	100.64.84.66	209.216.230.240	TCP	66	49314 → 443 [ACK] Seq=518 Ack=3893 Win=127872 Len=0 TSval=2512581363 TSecr=2045828612
726	7.917726	100.64.84.66	209.216.230.240	TCP	66	[TCP Window Update] 49314 → 443 [ACK] Seq=518 Ack=3893 Win=131872 Len=0 TSval=2512581363 TSecr=2045828612
727	7.937248	100.64.84.66	209.216.230.240	TLSh1..	192	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
728	7.937649	100.64.84.66	209.216.230.240	TCP	786	Application Data
729	8.008705	209.216.230.240	100.64.84.66	TCP	66	443 → 49314 [ACK] Seq=3893 Ack=1364 Win=64832 Len=0 TSval=2045828783 TSecr=2512581383
730	8.093869	209.216.230.240	100.64.84.66	TLSh1..	324	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
731	8.093957	100.64.84.66	209.216.230.240	TCP	66	49314 → 443 [ACK] Seq=1364 Ack=4151 Win=130752 Len=0 TSval=2512581539 TSecr=2045828788
732	8.096295	209.216.230.240	100.64.84.66	TCP	1514	443 → 49314 [ACK] Seq=4151 Ack=1364 Win=65664 Len=1448 TSval=2045828789 TSecr=2512581383 [TCP segment of a reassembled PDU]
733	8.096296	209.216.230.240	100.64.84.66	TCP	1514	443 → 49314 [ACK] Seq=5999 Ack=1364 Win=65664 Len=1448 TSval=2045828789 TSecr=2512581383 [TCP segment of a reassembled PDU]
734	8.096296	209.216.230.240	100.64.84.66	TCP	1514	443 → 49314 [ACK] Seq=7047 Ack=1364 Win=65664 Len=1448 TSval=2045828789 TSecr=2512581383 [TCP segment of a reassembled PDU]
735	8.096297	209.216.230.240	100.64.84.66	TCP	1514	443 → 49314 [ACK] Seq=8495 Ack=1364 Win=65664 Len=1448 TSval=2045828789 TSecr=2512581383 [TCP segment of a reassembled PDU]
736	8.096298	209.216.230.240	100.64.84.66	TLSh1..	681	Application Data
737	8.096371	100.64.84.66	209.216.230.240	TCP	66	49314 → 443 [ACK] Seq=1364 Ack=10558 Win=124608 Len=0 TSval=2512581542 TSecr=2045828789
738	8.096404	100.64.84.66	209.216.230.240	TCP	66	[TCP Window Update] 49314 → 443 [ACK] Seq=1364 Ack=10558 Win=131072 Len=0 TSval=2512581542 TSecr=2045828789
739	8.223532	100.64.84.66	209.216.230.240	TLSh1..	691	Application Data
740	8.252790	100.64.84.66	209.216.230.240	TCP	78	49315 → 443 [SYN, ECN, CWR] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=3827897587 TSecr=0 SACK_PERM=1
741	8.374585	209.216.230.240	100.64.84.66	TCP	1514	443 → 49314 [ACK] Seq=10558 Ack=1989 Win=65664 Len=1448 TSval=2045829070 TSecr=2512581669 [TCP segment of a reassembled PDU]
742	8.374587	209.216.230.240	100.64.84.66	TLSh1..	823	Application Data
743	8.374653	100.64.84.66	209.216.230.240	TCP	66	49314 → 443 [ACK] Seq=1989 Ack=12763 Win=128832 Len=0 TSval=2512581820 TSecr=2045829070
744	8.376081	100.64.84.66	209.216.230.240	TLSh1..	674	Application Data
750	8.419434	209.216.230.240	100.64.84.66	TCP	74	443 → 49315 [SYN, ACK, ECN] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=64 SACK_PERM=1 TSval=1535760379 TSecr=3827897587
751	8.419566	100.64.84.66	209.216.230.240	TCP	66	49315 → 443 [ACK] Seq=1 Ack=1 Win=131712 Len=0 TSval=3827897754 TSecr=1535760379
752	8.424337	100.64.84.66	209.216.230.240	TLSh1..	585	Client Hello
759	8.527067	209.216.230.240	100.64.84.66	TCP	1514	443 → 49314 [ACK] Seq=12763 Ack=2597 Win=65664 Len=1448 TSval=2045829221 TSecr=2512581821 [TCP segment of a reassembled PDU]
760	8.527068	209.216.230.240	100.64.84.66	TLSh1..	793	Application Data
761	8.527151	100.64.84.66	209.216.230.240	TCP	66	49314 → 443 [ACK] Seq=2597 Ack=14938 Win=128896 Len=0 TSval=2512581972 TSecr=2045829221
762	8.591413	209.216.230.240	100.64.84.66	TLSh1..	222	Server Hello, Change Cipher Spec, Encrypted Handshake Message
763	8.591467	100.64.84.66	209.216.230.240	TCP	66	49315 → 443 [ACK] Seq=520 Ack=157 Win=131584 Len=0 TSval=3827897926 TSecr=1535760550
764	8.591689	100.64.84.66	209.216.230.240	TLSh1..	117	Change Cipher Spec, Encrypted Handshake Message
765	8.622083	100.64.84.66	209.216.230.240	TLSh1..	719	Application Data
766	8.772916	209.216.230.240	100.64.84.66	TCP	1514	443 → 49314 [ACK] Seq=14938 Ack=3250 Win=65664 Len=1448 TSval=2045829468 TSecr=2512582067 [TCP segment of a reassembled PDU]

Abbildung 20: Initiierung einer HTTP-TCP-Sitzung

Können Sie den 3-Way-Handshake erkennen? Markieren Sie ihn in der Dokumentation. Welche TCP-Optionen sind beim Handshake aktiviert und welche Bedeutung haben sie?

No.	Time	Source	Destination	Protocol	Length	Info
714	7.598625	100.64.84.66	141.70.124.5	DNS	80	Standard query 0x189d A news.ycombinator.com
715	7.598881	100.64.84.66	141.70.124.5	DNS	80	Standard query 0x58df AAAA news.ycombinator.com
716	7.608834	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 TSval=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 TSval=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 TSval=2512581211 TSecr=2045828460
721	7.765826	100.64.84.66	209.216.230.240	TLShL	583	Client Hello
722	7.917493	209.216.230.240	100.64.84.66	TLShL	1514	Server Hello
723	7.917494	209.216.230.240	100.64.84.66	TCP	1514	443 → 49314 [ACK] Seq=1449 Ack=518 Win=65664 Len=1448 TSval=2045828612 TSecr=2512581211 [TCP segment of a reassembled PDU]
724	7.917495	209.216.230.240	100.64.84.66	TLShL	1062	Certificate, Certificate Status, Server Key Exchange, Server Hello Done
725	7.917581	100.64.84.66	209.216.230.240	TCP	66	49314 → 443 [ACK] Seq=518 Ack=3893 Win=127872 Len=0 TSval=2512581363 TSecr=2045828612
726	7.917726	100.64.84.66	209.216.230.240	TCP	66	[TCP Window Update] 49314 → 443 [ACK] Seq=518 Ack=3893 Win=131072 Len=0 TSval=2512581363 TSecr=2045828612
727	7.937248	100.64.84.66	209.216.230.240	TLShL	192	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
728	7.937649	100.64.84.66	209.216.230.240	TLShL	786	Application Data
729	8.088785	209.216.230.240	100.64.84.66	TCP	66	443 → 49314 [ACK] Seq=3893 Ack=1364 Win=64832 Len=0 TSval=2045828783 TSecr=2512581383
730	8.089369	209.216.230.240	100.64.84.66	TLShL	324	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
731	8.089357	100.64.84.66	209.216.230.240	TCP	66	49314 → 443 [ACK] Seq=1364 Ack=1364 Win=130752 Len=0 TSval=2512581539 TSecr=2045828788
732	8.096295	209.216.230.240	100.64.84.66	TCP	1514	443 → 49314 [ACK] Seq=4151 Ack=1364 Win=65664 Len=1448 TSval=2045828789 TSecr=2512581383 [TCP segment of a reassembled PDU]
733	8.096296	209.216.230.240	100.64.84.66	TCP	1514	443 → 49314 [ACK] Seq=5599 Ack=1364 Win=65664 Len=1448 TSval=2045828789 TSecr=2512581383 [TCP segment of a reassembled PDU]
734	8.096296	209.216.230.240	100.64.84.66	TCP	1514	443 → 49314 [ACK] Seq=7847 Ack=1364 Win=65664 Len=1448 TSval=2045828789 TSecr=2512581383 [TCP segment of a reassembled PDU]
735	8.096297	209.216.230.240	100.64.84.66	TCP	1514	443 → 49314 [ACK] Seq=8495 Ack=1364 Win=65664 Len=1448 TSval=2045828789 TSecr=2512581383 [TCP segment of a reassembled PDU]
736	8.096298	209.216.230.240	100.64.84.66	TLShL	681	Application Data
737	8.096371	100.64.84.66	209.216.230.240	TCP	66	49314 → 443 [ACK] Seq=1364 Ack=10558 Win=124608 Len=0 TSval=2512581542 TSecr=2045828789
738	8.096484	100.64.84.66	209.216.230.240	TCP	66	[TCP Window Update] 49314 → 443 [ACK] Seq=1364 Ack=10558 Win=131072 Len=0 TSval=2512581542 TSecr=2045828789
739	8.223532	100.64.84.66	209.216.230.240	TLShL	691	Application Data
740	8.252790	100.64.84.66	209.216.230.240	TCP	78	49315 → 443 [SYN, ECN, CWR] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=3827897587 TSecr=0 SACK_PERM=1
741	8.374585	209.216.230.240	100.64.84.66	TCP	1514	443 → 49314 [ACK] Seq=18558 Ack=1989 Win=65664 Len=1448 TSval=2045829070 TSecr=2512581669 [TCP segment of a reassembled PDU]
742	8.374587	209.216.230.240	100.64.84.66	TLShL	823	Application Data
743	8.374653	100.64.84.66	209.216.230.240	TCP	66	49314 → 443 [ACK] Seq=1989 Ack=12763 Win=128832 Len=0 TSval=2512581820 TSecr=2045829070
744	8.376081	100.64.84.66	209.216.230.240	TLShL	674	Application Data
750	8.419434	209.216.230.240	100.64.84.66	TCP	74	443 → 49315 [SYN, ACK, ECN] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=64 SACK_PERM=1 TSval=1535760379 TSecr=3827897587
751	8.419506	100.64.84.66	209.216.230.240	TCP	66	49315 → 443 [ACK] Seq=1 Ack=1 Win=131712 Len=0 TSval=3827897754 TSecr=1535760379
752	8.424337	100.64.84.66	209.216.230.240	TLShL	585	Client Hello
759	8.527067	209.216.230.240	100.64.84.66	TCP	1514	443 → 49314 [ACK] Seq=12763 Ack=2597 Win=65664 Len=1448 TSval=2045829221 TSecr=2512581821 [TCP segment of a reassembled PDU]
760	8.527068	209.216.230.240	100.64.84.66	TLShL	793	Application Data
761	8.527151	100.64.84.66	209.216.230.240	TCP	66	49314 → 443 [ACK] Seq=2597 Ack=14938 Win=128896 Len=0 TSval=2512581972 TSecr=2045829221
762	8.591413	209.216.230.240	100.64.84.66	TLShL	222	Server Hello, Change Cipher Spec, Encrypted Handshake Message
763	8.591467	100.64.84.66	209.216.230.240	TCP	66	49315 → 443 [ACK] Seq=520 Ack=157 Win=131584 Len=0 TSval=3827897926 TSecr=1535760550
764	8.591689	100.64.84.66	209.216.230.240	TLShL	117	Change Cipher Spec, Encrypted Handshake Message
765	8.622083	100.64.84.66	209.216.230.240	TLShL	719	Application Data
766	8.772916	209.216.230.240	100.64.84.66	TCP	1514	443 → 49314 [ACK] Seq=14938 Ack=3250 Win=65664 Len=1448 TSval=2045829468 TSecr=2512582067 [TCP segment of a reassembled PDU]

Abbildung 21: 3-Way-Handshake.

```

Frame Number: 718
Frame Length: 78 bytes (624 bits)
Capture Length: 78 bytes (624 bits)
[Frame is marked: False]
[Frame is ignored: False]
(Protocols in frame: eth:ethertype:ip:tcp)
[Coloring Rule Name: TCP SYN/FIN]
[Coloring Rule String: tcp.flags.fin == 1]
> Ethernet II, Src: Apple44:73:0e (84:83:e7:44:73:0e), Dst: JuniperR_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
  [011 ... - Header Length: 44 bytes (11)]
  > Flags: 0x02 (SYN, ECN, CWR)
    Window: 65535
    [Calculated window size: 65535]
    Checksum: 0xf787 [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
  > Options: (4 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 (multiply by 64)
    > 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)
  
```

Das SYN-Segment enthält die Optionen Maximum Segment Size, Window scale, Timestamps und SACK (Selective Acknowledgement).

```

ip.addr == 100.64.84.66 && transmission.number >= 966
No.    Time           Source                Destination           Protocol Length Info
-----
716 7.608034      141.70.124.5         100.64.84.66         DNS                158 Standard query response 0x58df AAAA news.ycombinator.com 50A 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 7.614306      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 TSval=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 TSval=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 TSval=2512581211 TSecr=2045828460
721 7.765826      100.64.84.66         209.216.230.240     TLSv1.2           583 Client Hello

[Time delta from previous displayed frame: 0.150904000 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:ethertype:ip:tcp]
[Coloring Rule Name: TCP SYN/FIN]
[Coloring Rule String: 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
  1000 .... = 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: (20 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: TSval 2045828460, TSecr 2512581059
  > [SEQ/ACK analysis]
    
```

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

```

ip.addr == 100.64.84.66 && transmission.number >= 966
No.    Time           Source                Destination           Protocol Length Info
-----
716 7.608034      141.70.124.5         100.64.84.66         DNS                158 Standard query response 0x58df AAAA news.ycombinator.com 50A 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 7.614306      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 TSval=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 TSval=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 TSval=2512581211 TSecr=2045828460
721 7.765826      100.64.84.66         209.216.230.240     TLSv1.2           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:ethertype: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: 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: 0xf4d4 [unverified]
  [Checksum Status: Unverified]
  Urgent Pointer: 0
  > Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
    > TCP Option - No-Operation (NOP)
    > TCP Option - No-Operation (NOP)
    > TCP Option - Timestamps: TSval 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 retransmitieren 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?

No.	Time	Source	Destination	Protocol	Length	Info
495	6.142042	2001:7c7:2126:4b00::...	update.googleapis.c...	TLSv1	158	Change Cipher Spec, Application Data
496	6.147080	update.googleapis.c...	2001:7c7:2126:4b00::...	TLSv1	694	Application Data, Application Data
497	6.147140	2001:7c7:2126:4b00::...	update.googleapis.c...	TCP	86	49364 → https(443) [ACK] Seq=582 Ack=5293 Win=130432 Len=0 TSval=1328298483 TSecr=1399184461
528	7.195215	100.64.84.66	news.ycombinator.c...	TCP	78	49365 → https(443) [SYN, ECN, CWR] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=3372401487 TSecr=0 SACK_PERM=1
529	7.351288	news.ycombinator.c...	100.64.84.66	TCP	74	https(443) → 49365 [SYN, ACK, ECN] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=64 SACK_PERM=1 TSval=1918799133 TSecr=3372401487
530	7.351406	100.64.84.66	news.ycombinator.c...	TCP	66	49365 → https(443) [ACK] Seq=1 Ack=1 Win=131712 Len=0 TSval=3372401642 TSecr=1918799133
531	7.352161	100.64.84.66	news.ycombinator.c...	TLSv1	583	Client Hello
532	7.588863	news.ycombinator.c...	100.64.84.66	TLSv1	1514	https(443) → 49365 [ACK] Seq=1449 Ack=518 Win=65664 Len=1448 TSval=1918799291 TSecr=3372401642 [TCP segment of a reassemb...
533	7.588863	news.ycombinator.c...	100.64.84.66	TCP	1062	Certificate, Certificate Status, Server Key Exchange, Server Hello Done
534	7.588864	news.ycombinator.c...	100.64.84.66	TCP	66	49365 → https(443) [ACK] Seq=518 Ack=3893 Win=127872 Len=0 TSval=3372401800 TSecr=1918799291
535	7.588923	100.64.84.66	news.ycombinator.c...	TCP	66	49365 → https(443) [ACK] Seq=518 Ack=3893 Win=127872 Len=0 TSval=3372401800 TSecr=1918799291
536	7.589010	100.64.84.66	news.ycombinator.c...	TCP	66	[TCP Window Update] 49365 → https(443) [ACK] Seq=518 Ack=3893 Win=131072 Len=0 TSval=3372401800 TSecr=1918799291
537	7.514919	100.64.84.66	news.ycombinator.c...	TLSv1	192	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
538	7.515281	100.64.84.66	news.ycombinator.c...	TLSv1	786	Application Data
539	7.670131	news.ycombinator.c...	100.64.84.66	TLSv1	324	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
540	7.670264	100.64.84.66	news.ycombinator.c...	TCP	66	49365 → https(443) [ACK] Seq=1364 Ack=4151 Win=130752 Len=0 TSval=3372401961 TSecr=1918799452
541	7.671518	news.ycombinator.c...	100.64.84.66	TCP	1514	https(443) → 49365 [ACK] Seq=4151 Ack=1364 Win=65664 Len=1448 TSval=1918799453 TSecr=3372401806 [TCP segment of a reassemb...
542	7.671520	news.ycombinator.c...	100.64.84.66	TCP	1514	https(443) → 49365 [ACK] Seq=5599 Ack=1364 Win=65664 Len=1448 TSval=1918799453 TSecr=3372401806 [TCP segment of a reassemb...
543	7.671521	news.ycombinator.c...	100.64.84.66	TCP	1514	https(443) → 49365 [ACK] Seq=7847 Ack=1364 Win=65664 Len=1448 TSval=1918799453 TSecr=3372401806 [TCP segment of a reassemb...
544	7.671522	news.ycombinator.c...	100.64.84.66	TCP	1514	https(443) → 49365 [ACK] Seq=8495 Ack=1364 Win=65664 Len=1448 TSval=1918799453 TSecr=3372401806 [TCP segment of a reassemb...
545	7.671523	news.ycombinator.c...	100.64.84.66	TLSv1	682	Application Data
546	7.671631	100.64.84.66	news.ycombinator.c...	TCP	66	49365 → https(443) [ACK] Seq=1364 Ack=10559 Win=124608 Len=0 TSval=3372401962 TSecr=1918799453
547	7.671793	100.64.84.66	news.ycombinator.c...	TCP	66	[TCP Window Update] 49365 → https(443) [ACK] Seq=1364 Ack=10559 Win=131072 Len=0 TSval=3372401962 TSecr=1918799453
548	7.844082	fe80::2d:b2a8:6eff:fe...	fe80::2d:b2a8:6eff:fe...	ICMPv6	86	Neighbor Solicitation for fe80::b2a8:6eff:fe9a:93ce from aa:43:e7:44:f3:0e
549	7.846274	fe80::2d:b2a8:6eff:fe...	fe80::2d:b2a8:6eff:fe...	ICMPv6	78	Neighbor Advertisement fe80::b2a8:6eff:fe9a:93ce (rttr, sol)
639	10.582785	2001:7c7:2126:4b00::...	lwn.net	TCP	98	49367 → https(443) [SYN, ECN, CWR] Seq=0 Win=65535 Len=0 MSS=1440 WS=64 TSval=4280199213 TSecr=0 SACK_PERM=1
641	10.663245	2001:7c7:2126:4b00::...	lwn.net	TCP	98	49368 → https(443) [SYN, ECN, CWR] Seq=0 Win=65535 Len=0 MSS=1440 WS=64 TSval=681163105 TSecr=0 SACK_PERM=1
642	10.689627	lwn.net	2001:7c7:2126:4b00::...	TCP	94	https(443) → 49367 [SYN, ACK, ECN] Seq=0 Ack=1 Win=20560 Len=0 MSS=1440 SACK_PERM=1 TSval=2318916019 TSecr=4280199213 WS=...
643	10.689750	2001:7c7:2126:4b00::...	lwn.net	TCP	86	49367 → https(443) [ACK] Seq=1 Ack=1 Win=131328 Len=0 TSval=4280199320 TSecr=2318916019
644	10.690211	2001:7c7:2126:4b00::...	lwn.net	TLSv1	603	Client Hello
645	10.753743	lwn.net	2001:7c7:2126:4b00::...	TCP	94	https(443) → 49368 [SYN, ACK, ECN] Seq=0 Ack=1 Win=20560 Len=0 MSS=1440 SACK_PERM=1 TSval=2318916099 TSecr=681163105 WS=...
646	10.753857	2001:7c7:2126:4b00::...	lwn.net	TCP	86	49368 → https(443) [ACK] Seq=1 Ack=1 Win=131328 Len=0 TSval=681163195 TSecr=2318916099
647	10.754321	lwn.net	2001:7c7:2126:4b00::...	TLSv1	603	Client Hello
648	10.796889	lwn.net	2001:7c7:2126:4b00::...	TCP	86	https(443) → 49367 [ACK] Seq=1 Ack=518 Win=29696 Len=0 TSval=2318916126 TSecr=4280199320
649	10.797292	lwn.net	2001:7c7:2126:4b00::...	TLSv1	1514	Server Hello
650	10.797293	lwn.net	2001:7c7:2126:4b00::...	TCP	1514	https(443) → 49367 [ACK] Seq=1429 Ack=518 Win=29696 Len=1428 TSval=2318916126 TSecr=4280199320 [TCP segment of a reassemb...
651	10.797294	lwn.net	2001:7c7:2126:4b00::...	TCP	1326	https(443) → 49367 [PSH, ACK] Seq=2857 Ack=518 Win=29696 Len=1240 TSval=2318916126 TSecr=4280199320 [TCP segment of a reassemb...
652	10.797401	2001:7c7:2126:4b00::...	lwn.net	TCP	86	49367 → https(443) [ACK] Seq=518 Ack=4097 Win=127232 Len=0 TSval=4280199428 TSecr=2318916126
653	10.797609	2001:7c7:2126:4b00::...	lwn.net	TCP	86	[TCP Window Update] 49367 → https(443) [ACK] Seq=518 Ack=4097 Win=131072 Len=0 TSval=4280199428 TSecr=2318916126
654	10.798585	lwn.net	2001:7c7:2126:4b00::...	TLSv1	578	Certificate, Server Key Exchange, Server Hello Done

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
170	65.999718934	HewlettP_aa:8b:be	Spanning-tree (for-bridges) 00 (01:00:c2:00:00:00)	STP	119	MST, Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
171	65.999828275	HewlettP_aa:8b:be	Spanning-tree (for-bridges) 00 (01:00:c2:00:00:00)	STP	119	MST, Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
172	67.999494848	HewlettP_aa:8b:be	Spanning-tree (for-bridges) 00 (01:00:c2:00:00:00)	STP	119	MST, Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
173	70.000137336	HewlettP_aa:8b:be	Spanning-tree (for-bridges) 00 (01:00:c2:00:00:00)	STP	119	MST, Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
174	71.999666778	HewlettP_aa:8b:be	Spanning-tree (for-bridges) 00 (01:00:c2:00:00:00)	STP	119	MST, Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
176	73.999729543	HewlettP_aa:8b:be	Spanning-tree (for-bridges) 00 (01:00:c2:00:00:00)	STP	119	MST, Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
177	75.999566689	HewlettP_aa:8b:be	Spanning-tree (for-bridges) 00 (01:00:c2:00:00:00)	STP	119	MST, Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
178	77.999639362	HewlettP_aa:8b:be	Spanning-tree (for-bridges) 00 (01:00:c2:00:00:00)	STP	119	MST, Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
179	79.999888965	HewlettP_aa:8b:be	Spanning-tree (for-bridges) 00 (01:00:c2:00:00:00)	STP	119	MST, Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
180	81.999602398	HewlettP_aa:8b:be	Spanning-tree (for-bridges) 00 (01:00:c2:00:00:00)	STP	119	MST, Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
181	83.999531792	HewlettP_aa:8b:be	Spanning-tree (for-bridges) 00 (01:00:c2:00:00:00)	STP	119	MST, Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
188	85.999230934	HewlettP_aa:8b:be	Spanning-tree (for-bridges) 00 (01:00:c2:00:00:00)	STP	119	MST, Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
191	87.999791212	HewlettP_aa:8b:be	Spanning-tree (for-bridges) 00 (01:00:c2:00:00:00)	STP	119	MST, Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
193	89.99989785	HewlettP_aa:8b:be	Spanning-tree (for-bridges) 00 (01:00:c2:00:00:00)	STP	119	MST, Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
196	91.999634842	HewlettP_aa:8b:be	Spanning-tree (for-bridges) 00 (01:00:c2:00:00:00)	STP	119	MST, Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
198	93.999871926	HewlettP_aa:8b:be	Spanning-tree (for-bridges) 00 (01:00:c2:00:00:00)	STP	119	MST, Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
199	95.999798412	HewlettP_aa:8b:be	Spanning-tree (for-bridges) 00 (01:00:c2:00:00:00)	STP	119	MST, Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
200	97.999598951	HewlettP_aa:8b:be	Spanning-tree (for-bridges) 00 (01:00:c2:00:00:00)	STP	119	MST, Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
201	100.000216073	HewlettP_aa:8b:be	Spanning-tree (for-bridges) 00 (01:00:c2:00:00:00)	STP	119	MST, Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
203	101.999558734	HewlettP_aa:8b:be	Spanning-tree (for-bridges) 00 (01:00:c2:00:00:00)	STP	119	MST, Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
204	103.999773302	HewlettP_aa:8b:be	Spanning-tree (for-bridges) 00 (01:00:c2:00:00:00)	STP	119	MST, Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
206	105.999642753	HewlettP_aa:8b:be	Spanning-tree (for-bridges) 00 (01:00:c2:00:00:00)	STP	119	MST, Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
212	108.000240070	HewlettP_aa:8b:be	Spanning-tree (for-bridges) 00 (01:00:c2:00:00:00)	STP	119	MST, Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
213	109.999891439	HewlettP_aa:8b:be	Spanning-tree (for-bridges) 00 (01:00:c2:00:00:00)	STP	119	MST, Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
221	111.99984588	HewlettP_aa:8b:be	Spanning-tree (for-bridges) 00 (01:00:c2:00:00:00)	STP	119	MST, Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
226	113.999732041	HewlettP_aa:8b:be	Spanning-tree (for-bridges) 00 (01:00:c2:00:00:00)	STP	119	MST, Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x8002
233	115.999658697	HewlettP_aa:8b:be	Spanning-tree (for-bridges) 00 (01:00:c2:00:00:00)	STP	119	MST, Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 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 = LG bit: Globally unique address (factory default)
      ... ..1 = IG bit: Group address (multicast/broadcast)
    Source: HewlettP_aa8b:be (04:09:73:aa:8b:be)
      ... ..0 = LG bit: Globally unique address (factory default)
      ... ..0 = IG bit: Individual address (unicast)
    Length: 105
  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

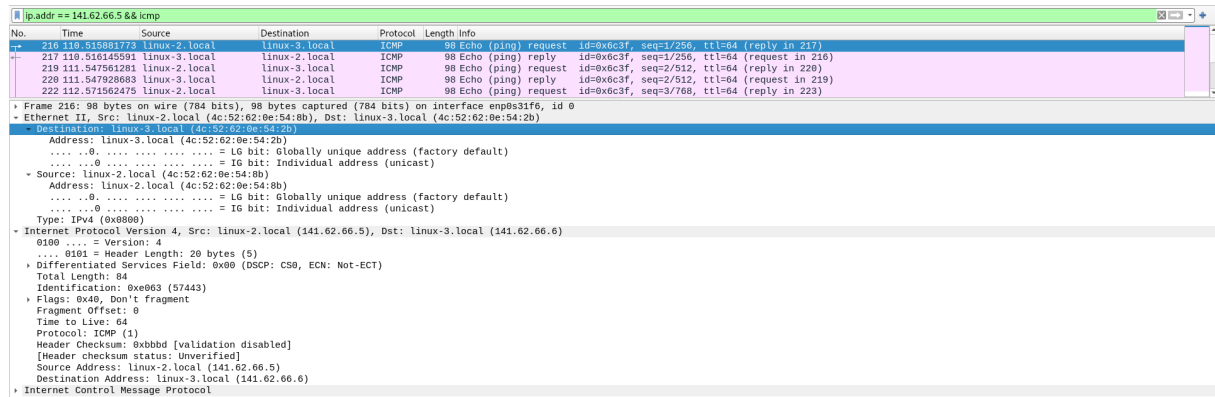


Abbildung 24: MAC-Adresse des Nachbarrechners

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

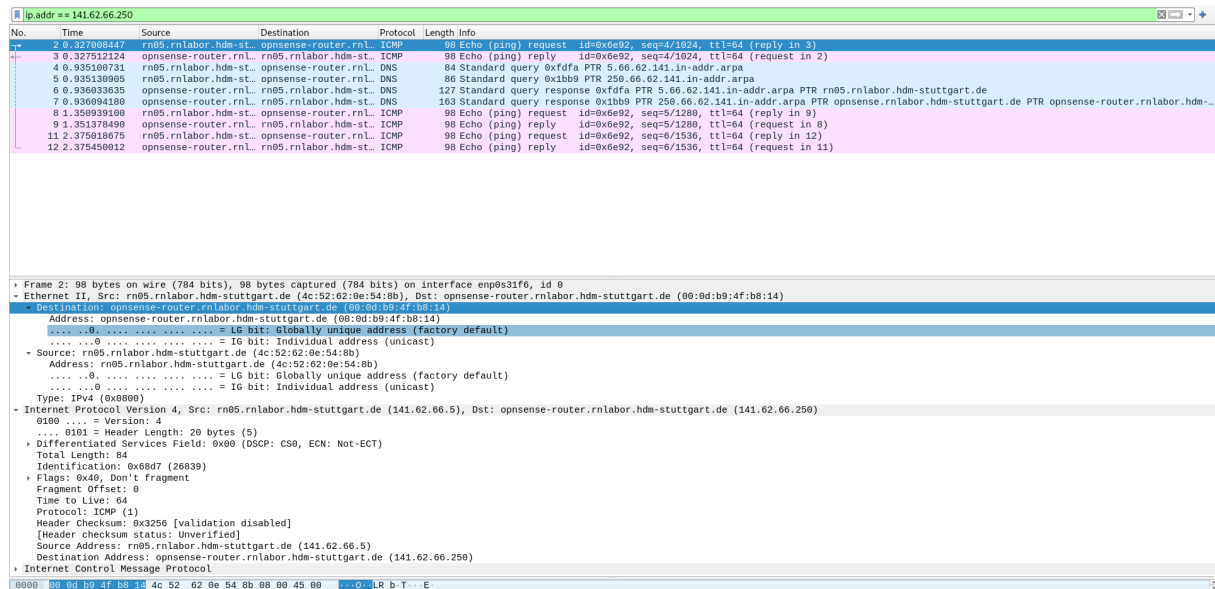


Abbildung 25: MAC-Adresse des Labor-Routers

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.

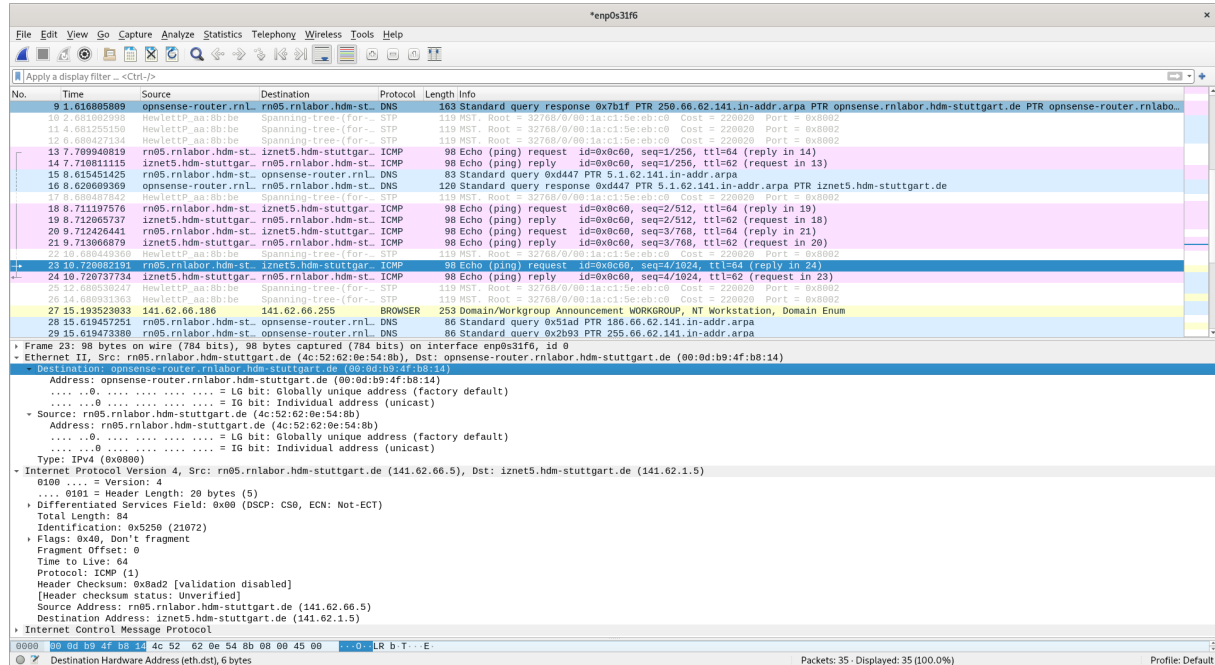


Abbildung 26: MAC-Adresse des externen Rechners

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-bridges).00	STP	119	MST. Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x0002
394	184.001059020	HewlettP_aa:8b:be	Spanning-tree (for-bridges).00	STP	119	MST. Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x0002
395	186.000556517	HewlettP_aa:8b:be	Spanning-tree (for-bridges).00	STP	119	MST. Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x0002
397	188.000292036	HewlettP_aa:8b:be	Spanning-tree (for-bridges).00	STP	119	MST. Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x0002
398	190.000136349	HewlettP_aa:8b:be	Spanning-tree (for-bridges).00	STP	119	MST. Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x0002
406	192.000508647	HewlettP_aa:8b:be	Spanning-tree (for-bridges).00	STP	119	MST. Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x0002
407	194.000871189	HewlettP_aa:8b:be	Spanning-tree (for-bridges).00	STP	119	MST. Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x0002
408	196.000399863	HewlettP_aa:8b:be	Spanning-tree (for-bridges).00	STP	119	MST. Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x0002
411	198.000533659	HewlettP_aa:8b:be	Spanning-tree (for-bridges).00	STP	119	MST. Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x0002
412	200.000207549	HewlettP_aa:8b:be	Spanning-tree (for-bridges).00	STP	119	MST. Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x0002
413	202.000187163	HewlettP_aa:8b:be	Spanning-tree (for-bridges).00	STP	119	MST. Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x0002
417	204.000254351	HewlettP_aa:8b:be	Spanning-tree (for-bridges).00	STP	119	MST. Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x0002
418	206.000015552	HewlettP_aa:8b:be	Spanning-tree (for-bridges).00	STP	119	MST. Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x0002
423	208.000379328	HewlettP_aa:8b:be	Spanning-tree (for-bridges).00	STP	119	MST. Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x0002
424	210.000285071	HewlettP_aa:8b:be	Spanning-tree (for-bridges).00	STP	119	MST. Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x0002
425	212.000277731	HewlettP_aa:8b:be	Spanning-tree (for-bridges).00	STP	119	MST. Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x0002
426	214.001030172	HewlettP_aa:8b:be	Spanning-tree (for-bridges).00	STP	119	MST. Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x0002
427	216.000876807	HewlettP_aa:8b:be	Spanning-tree (for-bridges).00	STP	119	MST. Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x0002
429	218.000280922	HewlettP_aa:8b:be	Spanning-tree (for-bridges).00	STP	119	MST. Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x0002
430	220.000146054	HewlettP_aa:8b:be	Spanning-tree (for-bridges).00	STP	119	MST. Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x0002
433	222.000177744	HewlettP_aa:8b:be	Spanning-tree (for-bridges).00	STP	119	MST. Root = 32768/0/00:1a:c1:5e:eb:c0 Cost = 220020 Port = 0x0002

Frame 426: 119 bytes on wire (952 bits), 119 bytes captured (952 bits) on interface enp8s31f6, id 0

- IEEE 802.3 Ethernet
 - Destination: Spanning-tree (for-bridges).00 (01:80:c2:00:00:00)
 - Address: Spanning-tree (for-bridges).00 (01:80:c2:00:00:00)
 -0..... = LG bit: Globally unique address (factory default)
 -1..... = IG 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..... = LG bit: Globally unique address (factory default)
 -0..... = IG bit: Individual address (unicast)
 - Length: 105
 - Logical-Link Control
 - Spanning Tree Protocol
 - Protocol Identifier: Spanning Tree Protocol (0x0000)
 - Protocol Version Identifier: Multiple Spanning Tree (3)
 - BPDU Type: Rapid/Multiple Spanning Tree (0x82)
 - BPDU flags: 0x3e, Forwarding, Learning, Port Role: Designated, Proposal
 - Root Identifier: 32768 / 0 / 00:1a:c1:5e:eb:c0
 - Root Path Cost: 220020
 - 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

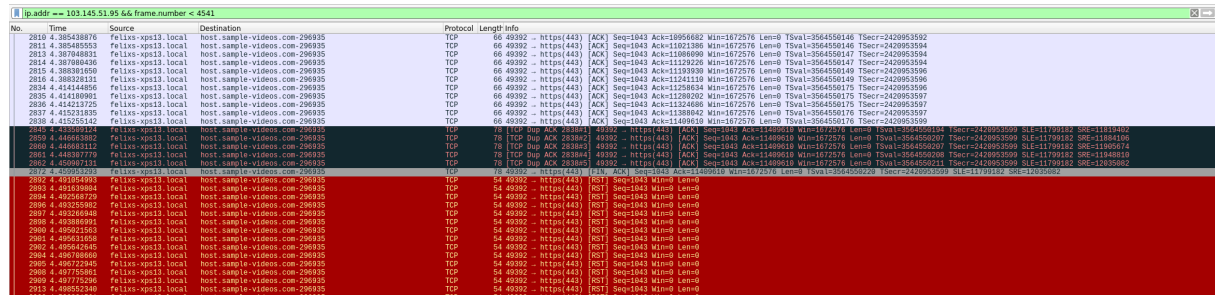


Abbildung 28: Capture beim Canceln des eines Downloads über HTTPS

Da der Download hier via HTTPS durchgeführt wurde, kann erkannt werden, dass die 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

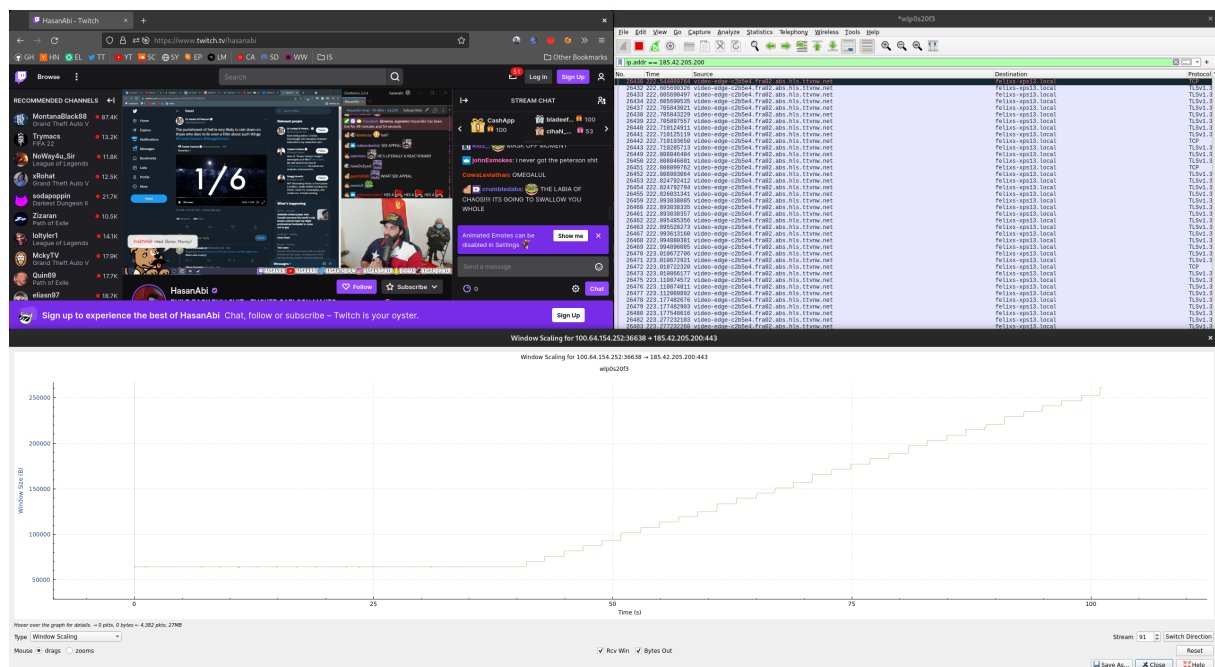


Abbildung 29: Verlauf der TCP-Window-Size beim Streaming von Twitch

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.

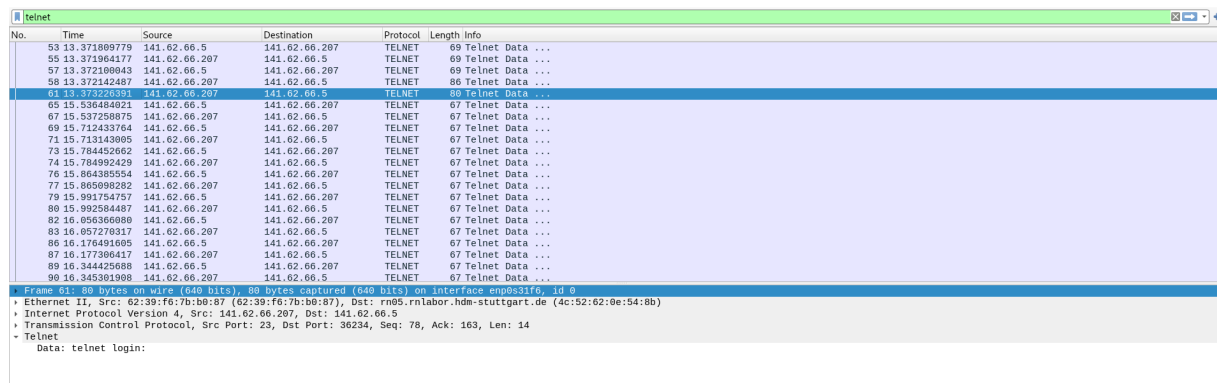


Abbildung 30: Capture des Telnet-Logins

Können Sie Passwörter im Wireshark-Trace identifizieren?

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

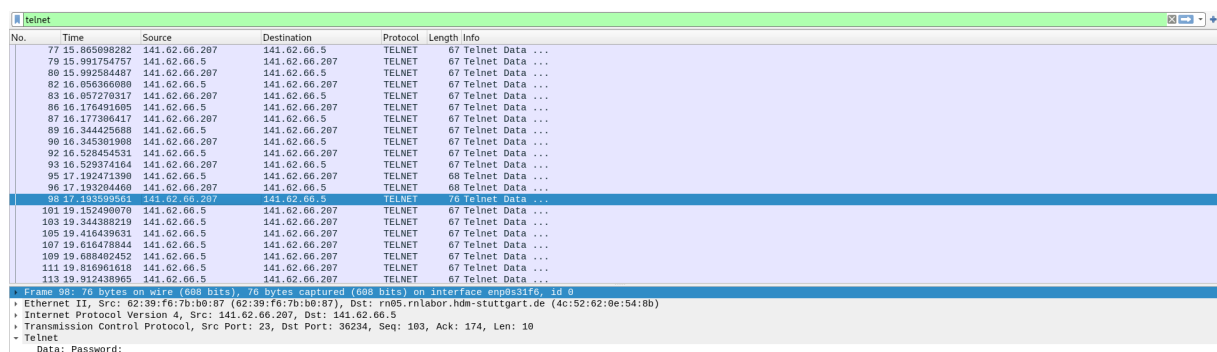


Abbildung 31: Capture des Telnet-Passworts

No.	Time	Source	Destination	Protocol	Length	Info
77	15.865998282	141.62.66.207	141.62.66.5	TELNET	67	Telnet Data ...
78	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	16.056360808	141.62.66.5	141.62.66.207	TELNET	67	Telnet Data ...
83	16.057279317	141.62.66.207	141.62.66.5	TELNET	67	Telnet Data ...
86	16.170491695	141.62.66.5	141.62.66.207	TELNET	67	Telnet Data ...
87	16.177396417	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	16.345391986	141.62.66.207	141.62.66.5	TELNET	67	Telnet Data ...
92	16.528454531	141.62.66.5	141.62.66.207	TELNET	67	Telnet Data ...
93	16.529374164	141.62.66.207	141.62.66.5	TELNET	67	Telnet Data ...
95	17.192471390	141.62.66.5	141.62.66.207	TELNET	68	Telnet Data ...
96	17.193294460	141.62.66.207	141.62.66.5	TELNET	68	Telnet Data ...
98	17.19359561	141.62.66.207	141.62.66.5	TELNET	76	Telnet Data ...
101	19.152488970	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 0
 Ethernet II, Src: rns5.rnlabor.hdm-stuttgart.de (4c:52:62:0e:54:9b), 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: 36234, Dst Port: 23, Seq: 174, Ack: 113, Len: 1
 Telnet
 Data: v

Abbildung 32: 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.

No.	Time	Source	Destination	Protocol	Length	Info
202	65.784067321	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	182	Client: Encrypted packet (len=36)
279	119.032310634	138.68.70.72	141.62.66.5	SSH	126	Server: Encrypted packet (len=60)
280	119.032477995	141.62.66.5	138.68.70.72	SSH	182	Client: Encrypted packet (len=36)
438	174.475555778	141.62.66.5	138.68.70.72	SSH	142	Client: Encrypted packet (len=70)
448	174.482598357	138.68.70.72	141.62.66.5	SSH	198	Server: Encrypted packet (len=132)
448	177.224986826	141.62.66.5	138.68.70.72	SSH	158	Client: Encrypted packet (len=92)
456	177.238561045	138.68.70.72	141.62.66.5	SSH	182	Server: Encrypted packet (len=116)
452	177.237844982	141.62.66.5	138.68.70.72	SSH	126	Client: Encrypted packet (len=60)
453	177.237128451	141.62.66.5	138.68.70.72	SSH	126	Client: Encrypted packet (len=60)
454	177.237283147	141.62.66.5	138.68.70.72	SSH	126	Client: Encrypted packet (len=60)
458	177.243289895	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	118	Client: Encrypted packet (len=52)
461	177.250982945	138.68.70.72	141.62.66.5	SSH	154	Server: Encrypted packet (len=148)
463	177.250574112	138.68.70.72	141.62.66.5	SSH	862	Server: Encrypted packet (len=796)
465	177.252448894	141.62.66.5	138.68.70.72	SSH	118	Client: Encrypted packet (len=52)
466	177.250252894	138.68.70.72	141.62.66.5	SSH	134	Server: Encrypted packet (len=68)
467	177.258776376	141.62.66.5	138.68.70.72	SSH	118	Client: Encrypted packet (len=52)
468	177.264904436	138.68.70.72	141.62.66.5	SSH	134	Server: Encrypted packet (len=68)
469	177.285339776	141.62.66.5	138.68.70.72	SSH	118	Client: Encrypted packet (len=52)
470	177.285833968	141.62.66.5	138.68.70.72	SSH	118	Client: Encrypted packet (len=52)

Frame 202: 126 bytes on wire (1008 bits), 126 bytes captured (1008 bits) on interface enp0s31f6, id 0
 Ethernet II, Src: opsense.rnlabor.hdm-stuttgart.de (08:0d:b9:4f:b8:14), Dst: rns5.rnlabor.hdm-stuttgart.de (4c:52:62:0e:54:9b)
 Internet Protocol Version 4, Src: 138.68.70.72, Dst: 141.62.66.5
 Transmission Control Protocol, Src Port: 22, Dst Port: 47846, Seq: 1, Ack: 1, Len: 60
 SSH Protocol
 Packet Length (encrypted): 90ef09e4
 Encrypted Packet: 60cbb15349f592f55938da2caccb0c73e844beb992378514580fe2c6b2d9dab4f62e0ad3e...
 [Direction: server-to-client]

Abbildung 33: 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

No.	TTL	Time	Source	Destination	Protocol	Length	Info
25	255	1.441955690	100.64.154.254	felixs-xps13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
29	255	1.477088579	100.64.154.254	felixs-xps13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
31	255	1.519793372	100.64.154.254	felixs-xps13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
89	255	3.490491116	100.64.154.254	felixs-xps13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
99	255	1.3.508559800	100.64.154.254	felixs-xps13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
112	255	1.4.554393555	100.64.154.254	felixs-xps13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
113	255	1.4.554393675	100.64.154.254	felixs-xps13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
1527	255	1.21.511898153	100.64.154.254	felixs-xps13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
1567	255	1.21.614196041	100.64.154.254	felixs-xps13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
2031	255	1.25.441398947	100.64.154.254	felixs-xps13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
2044	255	1.25.456619749	100.64.154.254	felixs-xps13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
2045	255	1.25.456619783	100.64.154.254	felixs-xps13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
2049	255	1.25.598822269	100.64.154.254	felixs-xps13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
2050	255	1.25.598822605	100.64.154.254	felixs-xps13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
2051	255	1.25.598822634	100.64.154.254	felixs-xps13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
2052	255	1.25.598822662	100.64.154.254	felixs-xps13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
11826	255	74.573785920	100.64.154.245	224.0.0.251	MDNS	198	Standard query 0x0000 PTR lb._dns-sd._udp.local. "QM" question PTR _companion-link._tcp.local. "QM" quest...
12018	255	75.597569660	100.64.154.245	224.0.0.251	MDNS	198	Standard query 0x0000 PTR lb._dns-sd._udp.local. "QM" question PTR _companion-link._tcp.local. "QM" quest...
12561	255	78.567487619	100.64.154.245	224.0.0.251	MDNS	198	Standard query 0x0000 PTR lb._dns-sd._udp.local. "QM" question PTR _companion-link._tcp.local. "QM" quest...
13269	255	87.681307937	100.64.154.245	224.0.0.251	MDNS	198	Standard query 0x0000 PTR lb._dns-sd._udp.local. "QM" question PTR _companion-link._tcp.local. "QM" quest...
18651	255	1.134.490477999	100.64.154.254	felixs-xps13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
18666	255	1.134.622113475	100.64.154.254	felixs-xps13.local	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
19646	255	140.929118747	100.64.154.245	224.0.0.251	MDNS	198	Standard query 0x0000 PTR lb._dns-sd._udp.local. "QM" question PTR _companion-link._tcp.local. "QM" quest...
19852	255	141.955010901	100.64.154.245	224.0.0.251	MDNS	198	Standard query 0x0000 PTR lb._dns-sd._udp.local. "QM" question PTR _companion-link._tcp.local. "QM" quest...
20394	255	144.924217109	100.64.154.245	224.0.0.251	MDNS	198	Standard query 0x0000 PTR lb._dns-sd._udp.local. "QM" question PTR _companion-link._tcp.local. "QM" quest...
21865	255	154.345592068	100.64.154.245	224.0.0.251	MDNS	198	Standard query 0x0000 PTR lb._dns-sd._udp.local. "QM" question PTR _companion-link._tcp.local. "QM" quest...
21935	255	155.472517304	100.64.154.245	224.0.0.251	MDNS	198	Standard query 0x0000 PTR lb._dns-sd._udp.local. "QM" question PTR _companion-link._tcp.local. "QM" quest...
22140	255	158.443181864	100.64.154.245	224.0.0.251	MDNS	198	Standard query 0x0000 PTR lb._dns-sd._udp.local. "QM" question PTR _companion-link._tcp.local. "QM" quest...
22784	255	167.057466049	100.64.154.245	224.0.0.251	MDNS	198	Standard query 0x0000 PTR lb._dns-sd._udp.local. "QM" question PTR _companion-link._tcp.local. "QM" quest...
22852	255	168.579565631	100.64.154.245	224.0.0.251	MDNS	198	Standard query 0x0000 PTR lb._dns-sd._udp.local. "QM" question PTR _companion-link._tcp.local. "QM" quest...

Abbildung 34: Capture der TTL-Werte ab 200

Der Linux-Kernel stellt standardmäßig die TTL auf 64; hier wurde ab 200 gefiltert, damit ausschließlich "ungewöhnliche" Pakete wie z.B. Type: 11 (Time-to-live exceeded)-ICMP-Pakete angezeigt werden.

Nur IP-Pakete, die fragmentiert sind

Mittels eines Filters auf "Must Fragment" konnten in dieser Aufgabe nur fragmentierte Pakete angezeigt werden.

No.	TTL	Time	Source	Destination	Protocol	Length	Info
16357	64	121.271232400	100.64.154.247	255.255.255.255	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off#0, ID=9a8b) (Reassembled in #16358)
16358	64	121.271232593	100.64.154.247	255.255.255.255	UDP	392	47699 → xmsg(1716) Len=1630

Abbildung 35: Capture von fragmentierten IP-Paketen

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.

No.	Time	Source	Destination	Protocol	Length	Info
3657	651.572178872	212.77.241.212	141.62.66.5	FTP	89	Response: 226 OKnet FTP Daemon
3704	786.895412163	141.62.66.5	212.77.241.212	FTP	78	Request: USER jakob
3706	786.843474962	212.77.241.212	141.62.66.5	FTP	89	Response: 331 Password required for jakob
3713	715.293442858	141.62.66.5	212.77.241.212	FTP	89	Request: PASS passwortlololo
3714	715.313954381	212.77.241.212	141.62.66.5	FTP	88	Response: 530 Login incorrect.
3716	715.313246123	141.62.66.5	212.77.241.212	FTP	72	Request: SYST
3717	715.331546619	212.77.241.212	141.62.66.5	FTP	85	Response: 215 UNIX Type: L8

Frame 3713: 89 bytes on wire (712 bits), 89 bytes captured (712 bits) on interface enp8s31f6, id 0
 • Ethernet II, Src: rne5.rnlabor.hdm-stuttgart.de (4c:52:62:0e:54:8b), Dst: opnsense.rnlabor.hdm-stuttgart.de (08:0d:b9:4f:b8:14)
 • Internet Protocol Version 4, Src: 141.62.66.5, Dst: 212.77.241.212
 • Transmission Control Protocol, Src Port: 51708, Dst Port: 21, Seq: 13, Ack: 57, Len: 23
 • File Transfer Protocol (FTP)
 [Current working directory:]

Abbildung 36: Capture eines FTP-Pakets, welches ein Passwort 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.

No.	TTL	Time	Source	Destination	Protocol	Length	Info
6508	64	11.367453746	felixs-xps13.local	news.ycombinator.com	TCP	74	41206 → http(80) [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=366326180 TSecr=3732855280
6644	64	11.609341374	felixs-xps13.local	news.ycombinator.com	TCP	66	41206 → http(80) [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=366326422 TSecr=3732855280
6645	64	11.609418667	felixs-xps13.local	news.ycombinator.com	HTTP	159	GET / HTTP/1.1
6774	64	11.814668182	felixs-xps13.local	news.ycombinator.com	TCP	66	41206 → http(80) [ACK] Seq=85 Ack=376 Win=64000 Len=0 TSval=366326627 TSecr=373285522
6775	64	11.814783601	felixs-xps13.local	news.ycombinator.com	TCP	66	41206 → http(80) [FIN, ACK] Seq=85 Ack=376 Win=64128 Len=0 TSval=366326628 TSecr=373285522
6888	64	12.019239834	felixs-xps13.local	news.ycombinator.com	TCP	66	41206 → http(80) [ACK] Seq=86 Ack=377 Win=64128 Len=0 TSval=366326832 TSecr=373285528

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.

No.	TTL	Time	Source	Destination	Protocol	Length	Info
2031	1	3.591566536	100.64.154.242	224.0.0.251	MDNS	87	Standard query 0x3a0d PTR 100.64.154.246.in-addr.arpa. "QM" question
2038	1	3.602016273	100.64.154.242	224.0.0.251	MDNS	87	Standard query 0x3a0e PTR 100.64.154.246.in-addr.arpa. "QM" question
2056	1	3.624785943	100.64.154.242	224.0.0.251	MDNS	87	Standard query 0x3a0f PTR 100.64.154.248.in-addr.arpa. "QM" question
2064	1	3.635125087	100.64.154.242	224.0.0.251	MDNS	87	Standard query 0x3a10 PTR 100.64.154.248.in-addr.arpa. "QM" question
2074	1	3.657613100	100.64.154.242	224.0.0.251	MDNS	87	Standard query 0x3a11 PTR 100.64.154.244.in-addr.arpa. "QM" question
2086	1	3.667971318	100.64.154.242	224.0.0.251	MDNS	87	Standard query 0x3a12 PTR 100.64.154.244.in-addr.arpa. "QM" question
2098	1	3.690437532	100.64.154.242	224.0.0.251	MDNS	87	Standard query 0x3a13 PTR 100.64.154.251.in-addr.arpa. "QM" question
2105	1	3.709798914	100.64.154.242	224.0.0.251	MDNS	87	Standard query 0x3a14 PTR 100.64.154.251.in-addr.arpa. "QM" question
2120	1	3.723295332	100.64.154.242	224.0.0.251	MDNS	87	Standard query 0x3a15 PTR 100.64.154.254.in-addr.arpa. "QM" question
2128	1	3.733692895	100.64.154.242	224.0.0.251	MDNS	87	Standard query 0x3a16 PTR 100.64.154.254.in-addr.arpa. "QM" question
2142	1	3.756352360	100.64.154.242	224.0.0.251	MDNS	87	Standard query 0x3a17 PTR 100.64.154.252.in-addr.arpa. "QM" question
2147	1	3.766727292	100.64.154.242	224.0.0.251	MDNS	87	Standard query 0x3a18 PTR 100.64.154.252.in-addr.arpa. "QM" question
3541	1	6.182275800	felixs-xps13.local	239.255.255.250	SSDP	213	M-SEARCH * HTTP/1.1
3543	1	6.184213321	felixs-xps13.local	239.255.255.250	SSDP	213	M-SEARCH * HTTP/1.1
4120	1	7.103316298	felixs-xps13.local	239.255.255.250	SSDP	213	M-SEARCH * HTTP/1.1
4122	1	7.105191345	felixs-xps13.local	239.255.255.250	SSDP	213	M-SEARCH * HTTP/1.1
4703	1	8.184255659	felixs-xps13.local	239.255.255.250	SSDP	213	M-SEARCH * HTTP/1.1
4705	1	8.186126105	felixs-xps13.local	239.255.255.250	SSDP	213	M-SEARCH * HTTP/1.1

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