

# Praktikum Rechnernetze

Versuch 2  
vom 05.05.2020  
Protokoll-Analyse

Protokoll Gruppe 1

Yannick Möller (ym018)

Bernd Maier (bm075)

Michael Vanhee (mv068)

Rebecca Mombrei (rm048)

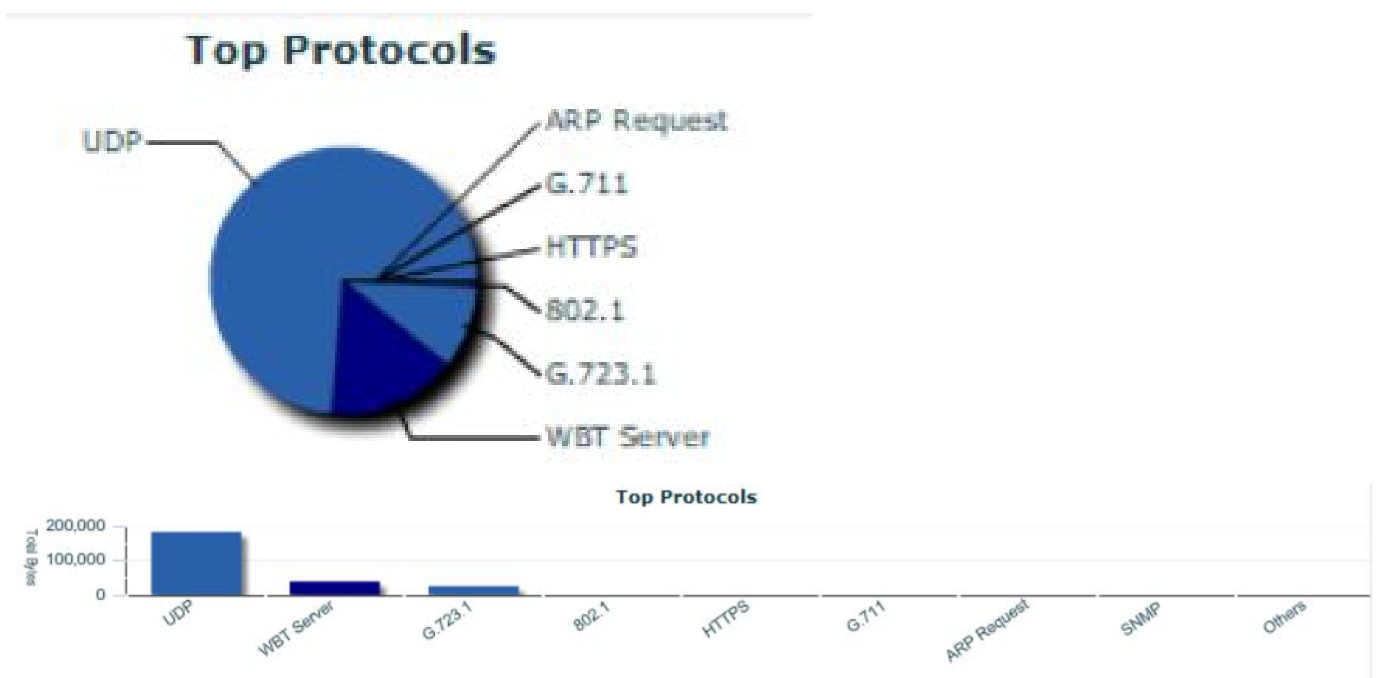
# Aufgabe 1

## Korrektter Anschluss des Netzwerk-Sniffers

Damit unser Netzwerk-Sniffer sämtlichen Datenverkehr aufzeichnen kann, ist es am sinnvollsten ihn am Router (HP 1920-24G 141.62.66.216) anzuschließen. Möchte man jedoch z.B. nur den Datenverkehr des Rechnernetze Labors mitlesen, muss der Sniffer an den beiden Switches (141.62.66.214, 141.62.66.213) angeschlossen werden.

## Protokoll Verteilung (mit WildPackets Compass)

Nach dem Speichern des aktuellen Traffics durch Wireshark zeigt WildPackets Compass folgende Protokoll Verteilung:

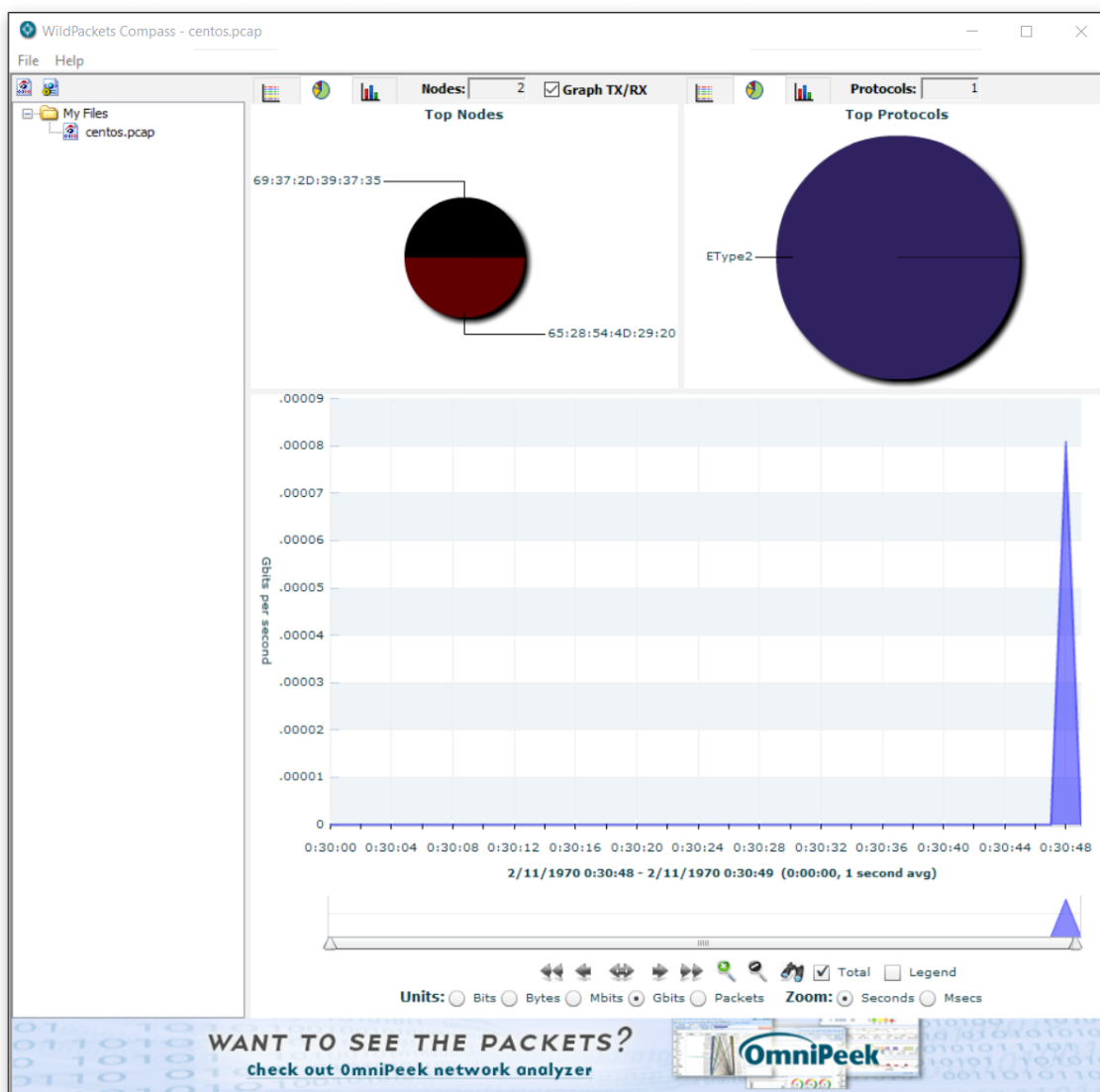


UDP ist dabei am häufigsten vertreten. Die meisten Pakete werden vom eigenen Laborrechner mit der IP 141.62.66.2 gesendet. WildPackets Compass bietet als Funktionalität nur die grafische Darstellung von Protokollen und Nodes. Wireshark hingegen bietet unter anderem die Möglichkeit, auch Statistiken zu Protokoll-Hierarchien, Adressauflösungen und Paketlängen abzurufen, was mithilfe von WildPackets Compass nicht möglich ist. Außerdem können Pakete in Wireshark detailliert betrachtet und gefiltert werden.

## Aufgabe 2

### Download einer größeren Datei und Auswertung mithilfe von WildPackets Compass

Wir haben den Download kurz vor dem Start von Wireshark auf einer VM gestartet. Dabei wurde der Download über EType2 (Ethernet Frame Type 2) übertragen. Durch die VM ist sehr viel anderer Traffic nicht mit aufgezeichnet, wie er auf einem anderen PC stattfinden würde, da die VM nur für Rechnernetze genutzt wird. Der Download erfolgte mit einer 1 GBit / Sekunde Verbindung, weshalb der Download ziemlich schnell erfolgte. Auch mehrfache Downloads haben kein anderes Ergebnis geliefert.



## Aufgabe 3

### Pinging around

Ein Ping vom Heimrechner an ein Gerät im selben Netzwerk liefert die folgende Paket-Kommunikation. Per **ICMP** wird eine Request an die angegebene Adresse (192.168.178.36) gesendet und diese (in diesem Fall mein Handy) antwortet entsprechend nach wenigen Millisekunden.

8628	10.451045	192.168.178.51	192.168.178.36	ICMP	74	Echo (ping) request	id=0x0001, seq=1478/50693, ttl=128 (reply in 8664)
8664	10.704279	192.168.178.36	192.168.178.51	ICMP	74	Echo (ping) reply	id=0x0001, seq=1478/50693, ttl=64 (request in 8628)
8782	11.470306	192.168.178.51	192.168.178.36	ICMP	74	Echo (ping) request	id=0x0001, seq=1479/50949, ttl=128 (reply in 8789)
8789	11.508198	192.168.178.36	192.168.178.51	ICMP	74	Echo (ping) reply	id=0x0001, seq=1479/50949, ttl=64 (request in 8782)
8946	12.491605	192.168.178.51	192.168.178.36	ICMP	74	Echo (ping) request	id=0x0001, seq=1480/51205, ttl=128 (reply in 8989)
8989	12.753701	192.168.178.36	192.168.178.51	ICMP	74	Echo (ping) reply	id=0x0001, seq=1480/51205, ttl=64 (request in 8946)
9102	13.506248	192.168.178.51	192.168.178.36	ICMP	74	Echo (ping) request	id=0x0001, seq=1481/51461, ttl=128 (reply in 9147)
9147	13.770574	192.168.178.36	192.168.178.51	ICMP	74	Echo (ping) reply	id=0x0001, seq=1481/51461, ttl=64 (request in 9102)

Bei einem Ping an [www.hdm-stuttgart.de](http://www.hdm-stuttgart.de) aus dem eigenen Heimnetzwerk ohne VPN-Verbindung kommt eine ähnliche Paket-Kommunikation zustande. Unterschiede sind in der **ttl** der Antwort zu erkennen.

No.	Time	Source	Destination	Protocol	Length	Info
67382	341.476380	192.168.178.51	141.62.1.59	ICMP	74	Echo (ping) request id=0x0001, seq=1482/51717, ttl=128 (reply in 67385)
67385	341.494141	141.62.1.59	192.168.178.51	ICMP	74	Echo (ping) reply id=0x0001, seq=1482/51717, ttl=55 (request in 67382)
67657	342.481339	192.168.178.51	141.62.1.59	ICMP	74	Echo (ping) request id=0x0001, seq=1483/51973, ttl=128 (reply in 67660)
67660	342.498329	141.62.1.59	192.168.178.51	ICMP	74	Echo (ping) reply id=0x0001, seq=1483/51973, ttl=55 (request in 67657)
68021	343.491676	192.168.178.51	141.62.1.59	ICMP	74	Echo (ping) request id=0x0001, seq=1484/52229, ttl=128 (reply in 68024)
68024	343.508491	141.62.1.59	192.168.178.51	ICMP	74	Echo (ping) reply id=0x0001, seq=1484/52229, ttl=55 (request in 68021)
68280	344.498374	192.168.178.51	141.62.1.59	ICMP	74	Echo (ping) request id=0x0001, seq=1485/52485, ttl=128 (reply in 68286)
68286	344.514558	141.62.1.59	192.168.178.51	ICMP	74	Echo (ping) reply id=0x0001, seq=1485/52485, ttl=55 (request in 68280)

Bei einem Ping an eine nicht existierende IP-Adresse (hier 1.2.3.4) lassen sich mit Hilfe von Wireshark (logischerweise) nur ausgehende Requests "einfangen". In diesem Fall wurde der Filter "icmp" verwendet. Responses gab es logischerweise keine.

600.514519	192.168.178.51	1.2.3.4	ICMP	74	Echo (ping) request	id=0x0001, seq=1584/12294, ttl=128 (no response found!)
605.384736	192.168.178.51	1.2.3.4	ICMP	74	Echo (ping) request	id=0x0001, seq=1585/12550, ttl=128 (no response found!)
610.384510	192.168.178.51	1.2.3.4	ICMP	74	Echo (ping) request	id=0x0001, seq=1586/12806, ttl=128 (no response found!)
615.393732	192.168.178.51	1.2.3.4	ICMP	74	Echo (ping) request	id=0x0001, seq=1587/13062, ttl=128 (no response found!)

# Aufgabe 4

## DHCP - Dynamic Host Configuration Protocol

DHCP ist ein Protokoll welches IPs im TCP/IP Netzwerk verwaltet. Dabei werden durch einen DHCP Server die Hosts verteilt. Mit DHCP können sich die Clients selbst konfigurieren.

### Wozu benötigt man DHCP?

Um innerhalb eines Netzwerks teilnehmen zu können, müssen einige Punkte beachtet werden. Dazu ist folgende IP-Konfiguration notwendig:

- IP-Adresse
- Subnetzmaske
- Gateways
- DNS-Server

DHCP funktioniert nach dem Client-Server Modell. Der Client fragt beim Server nach einer IP-Konfiguration. Dabei verfügt der Server über einen Pool von IP-Adressen, die er ausgeben kann. Je nach Netzkonfiguration ist es notwendig, dass der Server über die Subnetze und die Gateways Bescheid weiß. In den meisten Fällen ist ein DHCP-Server auch ein Router.

### Wie bekommt denn ein Client nun eine IP Adresse?

Wenn ein Host startet, wird ein funktionales Setup des TCP/IP Stacks gestartet. In diesem Modus besitzt der Host keine gültige IP Adresse, keine Subnetzmaske und kein Gateway. Das einzige was der Host zu diesem Zeitpunkt machen kann, ist IP Broadcasts zu verschicken. Dabei wird nach folgendem Schema gearbeitet.

### DHCP Discover

Der Client verschickt ein UDP Paket mit der Ziel-Adresse 255.255.255.255 und der Quell-Adresse 0.0.0.0. Das ist das Zeichen einer Adressanforderung an den DHCP Server im Netzwerk. Es sollte nur einen DHCP Server im Netzwerk geben, da es bei mehrere DHCP Servern im Netzwerk zu unterschiedlichen Konfigurationen kommen kann. Der Client bekommt die IP, welche von dem DHCP Server am schnellsten als Antwort geschickt worden ist. Um solche Konflikte im Idealfall zu vermeiden, ist es die optimale Lösung, nur einen DHCP Server zu haben.

### DHCP Offer

Der DHCP Server reagiert auf den vorher verschickten DHCP Discover des Clients mit einer freien IP Adresse und einigen weiteren Parametern. Dabei werden folgende weiteren Informationen mit einem UDP Paket zurück geschickt:

- MAC Adresse des Clients
- freie IP Adresse
- Lease-Time
- Subnetzmaske
- IP des DHCP Servers

**DHCP Request**

Aus den DHCP Offer Paketen (falls es mehrere DHCP Server im Netzwerk gibt) sucht sich der Client eine heraus und verschickt eine Meldung, dass er diese IP Adresse nimmt. Auch an die anderen Server wird diese Information geschickt, die diese Information berücksichtigen

**DHCP Acknowledgement**

Die IP Adresse muss vom DHCP Server noch abschließend bestätigt werden. Falls der DHCP Client noch weitere Parameter auswerten kann, übermittelt der DHCP Server noch weitere Optionen zur Vervollständigung:

- NTP Time Server
- DNS Domain Name Server
- Domain Name (beispielsweise .lan)
- die Standard IP TTL
- Broadcast Adresse
- E-Mail Konfigurationen (SMTP, POP3)
- WINS Server

Nachdem der Client das DHCP Acknowledgement Paket erhalten hat, speichert der Client sich die Daten lokal ab und verlässt das Setup. Daraufhin wird der TCP/IP Stack ohne das Setup gestartet.

**Gibt es noch weitere Pakete auf die wir eingehen können?**

Was passiert eigentlich, wenn der DHCP Server keine freien IPs mehr in seinem Pool hat?

**DHCP Not Acknowledged**

Der DHCP Server sendet daraufhin dieses Paket. Dies ist auch der Fall, falls der Client eine IP von einem anderen DHCP Server erhält.

Bei jeder IP Adressvergabe wird eine Lease-Time mit angegeben. Was passiert eigentlich, wenn die sie erreicht ist?

**DHCP Refresh**

Im DHCP Acknowledgement Paket wird eine Lease-Dauer mit angegeben. Sie gibt dem Client die Auskunft darüber, wie lange er die IP Adresse verwenden darf. Dabei agiert der Client nicht auf den letzten Drücker um die IP zu behalten, sondern fragt bereits nach der Hälfte der Lease Zeit mit einem erneuten DHCP Request. Meistens antwortet dabei der DHCP Server mit einem DHCP Acknowledgment mit identischen Informationen und einem aktualisierten Lease. So kann die Nutzung der IP Adresse verlängert werden.

Was passiert dann aber, falls der DHCP Server gerade in diesem Moment nicht verfügbar ist?

Auch dafür gibt es eine Lösung. Der Client nutzt weiterhin die IP, bis die Lease endgültig erreicht ist. Jetzt, auf den letzten Drücker, versucht er nochmals eine Verlängerung zu erhalten. Sollte auch das Fehlschlagen, so nimmt er mit DHCP Discover erneut Kontakt zu allen DHCP Servern im Netzwerk auf. Schlägt das auch Fehl, wird der Client wieder in den besonderen Modus des TCP/IP Stacks zurück fallen.

**Was passiert, falls der Client keine IPv4 Konfiguration erhält?**

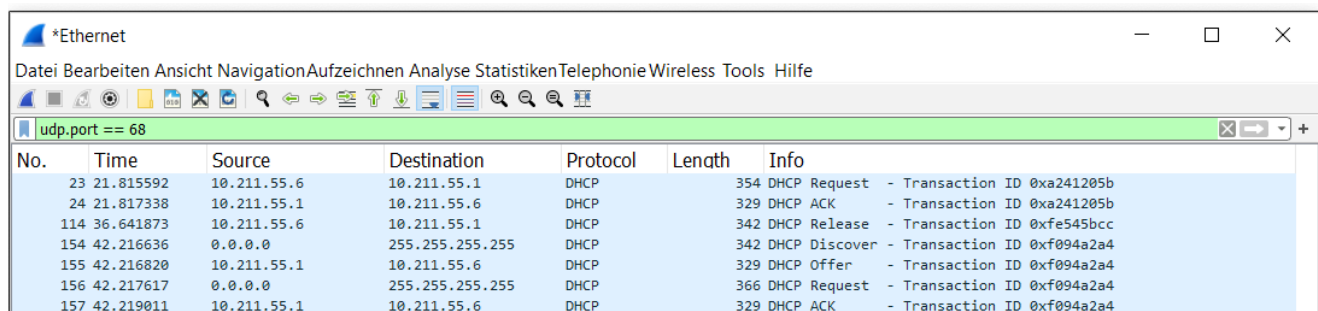
Dabei gibt es mehrere verschiedene Gründe, die wir berücksichtigen müssen:

- Steckt überhaupt ein Kabel am Ethernet Port? Dann ist der Client mit keinem Netzwerk verbunden.
- Der Stecker steckt, aber es existiert kein DHCP Server im Netzwerk
- Der DHCP Server ist ausgeschaltet, deaktiviert oder sein Kabel steckt nicht
- Der DHCP Server hat keine freien Adressen mehr in seinem IP Pool
- Der DHCP Server hat eine fehlerhafte Konfiguration

In allen dieser Fälle holt der Client sich eine IPv4 Adresse aus dem link-local Adressbereich (169.253.0.0/16). So ist immerhin eine Kommunikation innerhalb des link-local Netzwerks möglich.

**Wir haben folgenden Ablauf befolgt um DHCP Traffic aufzuzeichnen:**

1. Wireshark Capture starten
2. Prompt geöffnet
3. ipconfig /renew
4. ipconfig /release
5. ipconfig /renew
6. Prompt geschlossen
7. Wireshark Capture gestoppt
8. Nach udp.port == 68 suchen



No.	Time	Source	Destination	Protocol	Length	Info
23	21.815592	10.211.55.6	10.211.55.1	DHCP	354	DHCP Request - Transaction ID 0xa241205b
24	21.817338	10.211.55.1	10.211.55.6	DHCP	329	DHCP ACK - Transaction ID 0xa241205b
114	36.641873	10.211.55.6	10.211.55.1	DHCP	342	DHCP Release - Transaction ID 0xfe545bcc
154	42.216636	0.0.0.0	255.255.255.255	DHCP	342	DHCP Discover - Transaction ID 0xf094a2a4
155	42.216820	10.211.55.1	10.211.55.6	DHCP	329	DHCP Offer - Transaction ID 0xf094a2a4
156	42.217617	0.0.0.0	255.255.255.255	DHCP	366	DHCP Request - Transaction ID 0xf094a2a4
157	42.219011	10.211.55.1	10.211.55.6	DHCP	329	DHCP ACK - Transaction ID 0xf094a2a4



# DNS

Wir haben folgenden Ablauf befolgt um DNS Traffic aufzuzeichnen:

1. Wireshark Capture starten
2. Prompt geöffnet
3. ipconfig /flushdns
4. ipconfig /displaydns
5. Das Ergebnis daraufhin sollte leer sein. Die einzigen Einträge kommen von der hosts Datei, die in jedem Betriebssystem vorhanden ist.
6. nslookup en.wikiversity.org
7. Jetzt werden Einträge angelegt
8. Prompt schließen
9. Wireshark Capture stoppen
10. Nach udp.port == 53 suchen

Die folgenden 4 Screenshots zeigen den Ablauf.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	fe80::a0b5:71b1:687...	fe80::21c:42ff:fe00...	DNS	152	Standard query 0x2d26 PTR 8.1.0.0.0.e.f.f.f.2.4.c.1.2.0.0.0...
2	0.003465	fe80::21c:42ff:fe00...	fe80::a0b5:71b1:687...	DNS	152	Standard query response 0x2d26 No such name PTR 8.1.0.0.0.0.e...
8	0.846511	fe80::a0b5:71b1:687...	fe80::21c:42ff:fe00...	DNS	104	Standard query 0x2f41 PTR 252.0.0.224.in-addr.arpa
9	0.866796	fe80::21c:42ff:fe00...	fe80::a0b5:71b1:687...	DNS	161	Standard query response 0x2f41 No such name PTR 252.0.0.224.i...
38	18.711238	fe80::a0b5:71b1:687...	fe80::21c:42ff:fe00...	DNS	95	Standard query 0xf211 A cdn.onenote.net
39	18.711450	fe80::a0b5:71b1:687...	fe80::21c:42ff:fe00...	DNS	95	Standard query 0x87cf AAAA cdn.onenote.net
40	18.711843	fe80::a0b5:71b1:687...	fe80::21c:42ff:fe00...	DNS	114	Standard query 0x826a A tile-service.weather.microsoft.com
41	18.712058	fe80::a0b5:71b1:687...	fe80::21c:42ff:fe00...	DNS	114	Standard query 0x70d4 AAAA tile-service.weather.microsoft.com
42	18.732151	fe80::21c:42ff:fe00...	fe80::a0b5:71b1:687...	DNS	185	Standard query response 0xf211 A cdn.onenote.net CNAME cdn.on...
43	18.736914	fe80::21c:42ff:fe00...	fe80::a0b5:71b1:687...	DNS	220	Standard query response 0x826a A tile-service.weather.microso...
44	18.739813	fe80::21c:42ff:fe00...	fe80::a0b5:71b1:687...	DNS	225	Standard query response 0x87cf AAAA cdn.onenote.net CNAME cdn...
64	19.048060	fe80::a0b5:71b1:687...	fe80::21c:42ff:fe00...	DNS	152	Standard query 0x487c PTR 8.1.0.0.0.e.f.f.f.2.4.c.1.2.0.0.0...
65	19.050379	fe80::21c:42ff:fe00...	fe80::a0b5:71b1:687...	DNS	152	Standard query response 0x487c No such name PTR 8.1.0.0.0.0.e...
71	19.704692	10.211.55.6	10.211.55.1	DNS	94	Standard query 0x70d4 AAAA tile-service.weather.microsoft.com
72	19.726361	10.211.55.1	10.211.55.6	DNS	242	Standard query response 0x70d4 AAAA tile-service.weather.micr...
83	19.893307	fe80::a0b5:71b1:687...	fe80::21c:42ff:fe00...	DNS	107	Standard query 0x8629 PTR 52.181.227.172.in-addr.arpa
84	19.912052	fe80::21c:42ff:fe00...	fe80::a0b5:71b1:687...	DNS	173	Standard query response 0x8629 PTR 52.181.227.172.in-addr.arp...
90	20.985488	fe80::a0b5:71b1:687...	fe80::21c:42ff:fe00...	DNS	152	Standard query 0xd6fe PTR 3.0.0.0.1.0.0.0.0.0.0.0.0.0.0.0.0...
91	21.014029	fe80::21c:42ff:fe00...	fe80::a0b5:71b1:687...	DNS	216	Standard query response 0xd6fe No such name PTR 3.0.0.0.1.0.0...
97	21.861765	fe80::a0b5:71b1:687...	fe80::21c:42ff:fe00...	DNS	104	Standard query 0xcdfc PTR 252.0.0.224.in-addr.arpa
98	21.882246	fe80::21c:42ff:fe00...	fe80::a0b5:71b1:687...	DNS	161	Standard query response 0xcdfc No such name PTR 252.0.0.224.i...
104	22.721208	fe80::a0b5:71b1:687...	fe80::21c:42ff:fe00...	DNS	104	Standard query 0xb7d9 PTR 1.55.211.10.in-addr.arpa
105	22.723540	fe80::21c:42ff:fe00...	fe80::a0b5:71b1:687...	DNS	104	Standard query response 0xb7d9 No such name PTR 1.55.211.10.i...
118	27.237588	fe80::a0b5:71b1:687...	fe80::21c:42ff:fe00...	DNS	152	Standard query 0x61be PTR 8.1.0.0.0.e.f.f.f.2.4.c.1.2.0.0.0...
119	27.239604	fe80::21c:42ff:fe00...	fe80::a0b5:71b1:687...	DNS	152	Standard query response 0x61be No such name PTR 8.1.0.0.0.0.e...
124	28.078722	fe80::a0b5:71b1:687...	fe80::21c:42ff:fe00...	DNS	152	Standard query 0xd0b0 PTR 3.0.0.0.1.0.0.0.0.0.0.0.0.0.0.0.0...
125	28.108529	fe80::21c:42ff:fe00...	fe80::a0b5:71b1:687...	DNS	216	Standard query response 0xd0b0 No such name PTR 3.0.0.0.1.0.0...
131	28.978527	fe80::a0b5:71b1:687...	fe80::21c:42ff:fe00...	DNS	104	Standard query 0x64a2 PTR 252.0.0.224.in-addr.arpa
132	28.990492	fe80::21c:42ff:fe00...	fe80::a0b5:71b1:687...	DNS	161	Standard query response 0x64a2 No such name PTR 252.0.0.224.i...
143	30.649109	fe80::a0b5:71b1:687...	fe80::21c:42ff:fe00...	DNS	109	Standard query 0x9f44 A v10.events.data.microsoft.com
144	30.649598	fe80::a0b5:71b1:687...	fe80::21c:42ff:fe00...	DNS	109	Standard query 0xb3e4 AAAA v10.events.data.microsoft.com
145	30.676792	fe80::21c:42ff:fe00...	fe80::a0b5:71b1:687...	DNS	227	Standard query response 0x9f44 A v10.events.data.microsoft.co...
146	30.679183	fe80::21c:42ff:fe00...	fe80::a0b5:71b1:687...	DNS	274	Standard query response 0xb3e4 AAAA v10.events.data.microsoft...
163	31.261513	fe80::a0b5:71b1:687...	fe80::21c:42ff:fe00...	DNS	152	Standard query 0x0001 PTR 8.1.0.0.0.e.f.f.f.2.4.c.1.2.0.0.0...
164	31.263568	fe80::21c:42ff:fe00...	fe80::a0b5:71b1:687...	DNS	152	Standard query response 0x0001 No such name PTR 8.1.0.0.0.0.e...
165	31.269926	fe80::a0b5:71b1:687...	fe80::21c:42ff:fe00...	DNS	110	Standard query 0x0002 A en.wikiversity.org.localdomain
166	31.270080	fe80::21c:42ff:fe00...	fe80::a0b5:71b1:687...	DNS	110	Standard query response 0x0002 No such name A en.wikiversity...
167	31.270272	fe80::a0b5:71b1:687...	fe80::21c:42ff:fe00...	DNS	110	Standard query 0x0003 AAAA en.wikiversity.org.localdomain
168	31.270383	fe80::21c:42ff:fe00...	fe80::a0b5:71b1:687...	DNS	110	Standard query response 0x0003 No such name AAAA en.wikiversi...
169	31.270583	fe80::a0b5:71b1:687...	fe80::21c:42ff:fe00...	DNS	98	Standard query 0x0004 A en.wikiversity.org
170	31.288656	fe80::21c:42ff:fe00...	fe80::a0b5:71b1:687...	DNS	143	Standard query response 0x0004 A en.wikiversity.org CNAME dyn...
171	31.292269	fe80::a0b5:71b1:687...	fe80::21c:42ff:fe00...	DNS	98	Standard query 0x0005 AAAA en.wikiversity.org
172	31.308647	fe80::21c:42ff:fe00...	fe80::a0b5:71b1:687...	DNS	155	Standard query response 0x0005 AAAA en.wikiversity.org CNAME ...
182	31.533274	fe80::a0b5:71b1:687...	fe80::21c:42ff:fe00...	DNS	104	Standard query 0x954a PTR 2.36.114.52.in-addr.arpa
183	31.556918	fe80::21c:42ff:fe00...	fe80::a0b5:71b1:687...	DNS	183	Standard query response 0x954a No such name PTR 2.36.114.52.i...
204	23.403104	fe80::a0b5:71b1:687...	fe80::21c:42ff:fe00...	DNS	152	Standard query 0x0027 PTR 8.1.0.0.0.e.f.f.f.2.4.c.1.2.0.0.0...

> Frame 1: 152 bytes on wire (1216 bits), 152 bytes captured (1216 bits) on interface \Device\NPF\_{A90CE091-D694-449E-939A-5123B0C10761}, id 0

> Ethernet II, Src: Parallel\_a4:60:aa (00:1c:42:a4:60:aa), Dst: Parallel\_00:00:18 (00:1c:42:00:00:18)

> Internet Protocol Version 6, Src: fe80::a0b5:71b1:687f:859c, Dst: fe80::21c:42ff:fe00:18

> User Datagram Protocol, Src Port: 64181 (64181), Dst Port: domain (53)

0000 00 1c 42 00 00 18 00 1c 42 a4 60 aa 86 dd 60 08 ..B....B.....

0010 50 21 00 62 11 80 fe 80 00 00 00 00 00 a0 b5 P!b.....

0020 71 b1 68 7f 85 9c fe 80 00 00 00 00 00 02 1c qh.....

wireshark\_Ethernet\_20200505165549\_a078 | Pakete: 259 · Angezeigt: 56 (21.6%) · Verworfen | Profil: Defa



\*Ethernet

Datei Bearbeiten Ansicht NavigationAufzeichnen Analyse StatistikenTelephonieWireless Tools Hilfe

tcp

No.	Time	Source	Destination	Protocol	Length	Info
64	19.048060	fe80::a0b5:71b1:687...	fe80::21c:42ff:fe00...	DNS	152	Standard query 0x487c PTR 8.1.0.0.0.0.e.f.f.f.2.4.c.1.2.0.0.0...
65	19.050379	fe80::21c:42ff:fe00...	fe80::a0b5:71b1:687...	DNS	152	Standard query response 0x487c No such name PTR 8.1.0.0.0.0.e...
71	19.704692	10.211.55.6	10.211.55.1	DNS	94	Standard query 0x70d4 AAAA tile-service.weather.microsoft.com
72	19.726361	10.211.55.1	10.211.55.6	DNS	242	Standard query response 0x70d4 AAAA tile-service.weather.micr...
83	19.893307	fe80::a0b5:71b1:687...	fe80::21c:42ff:fe00...	DNS	107	Standard query 0x8629 PTR 52.181.227.172.in-addr.arpa
84	19.912052	fe80::21c:42ff:fe00...	fe80::a0b5:71b1:687...	DNS	173	Standard query response 0x8629 PTR 52.181.227.172.in-addr.arp...
90	20.985488	fe80::a0b5:71b1:687...	fe80::21c:42ff:fe00...	DNS	152	Standard query 0xd6fe PTR 3.0.0.0.1.0.0.0.0.0.0.0.0.0.0.0.0...
91	21.014029	fe80::21c:42ff:fe00...	fe80::a0b5:71b1:687...	DNS	216	Standard query response 0xd6fe No such name PTR 3.0.0.0.1.0.0...
97	21.861765	fe80::a0b5:71b1:687...	fe80::21c:42ff:fe00...	DNS	104	Standard query 0xcfd4 PTR 252.0.0.224.in-addr.arpa
98	21.882246	fe80::21c:42ff:fe00...	fe80::a0b5:71b1:687...	DNS	161	Standard query response 0xcfd4 No such name PTR 252.0.0.224.i...
104	22.721208	fe80::a0b5:71b1:687...	fe80::21c:42ff:fe00...	DNS	104	Standard query 0xb7d9 PTR 1.55.211.10.in-addr.arpa
105	22.723540	fe80::21c:42ff:fe00...	fe80::a0b5:71b1:687...	DNS	104	Standard query response 0xb7d9 No such name PTR 1.55.211.10.i...
118	27.237588	fe80::a0b5:71b1:687...	fe80::21c:42ff:fe00...	DNS	152	Standard query 0x61be PTR 8.1.0.0.0.0.e.f.f.f.2.4.c.1.2.0.0.0...
119	27.239604	fe80::21c:42ff:fe00...	fe80::a0b5:71b1:687...	DNS	152	Standard query response 0x61be No such name PTR 8.1.0.0.0.0.e...
124	28.078722	fe80::a0b5:71b1:687...	fe80::21c:42ff:fe00...	DNS	152	Standard query 0xd0b0 PTR 3.0.0.0.1.0.0.0.0.0.0.0.0.0.0.0.0...
125	28.108529	fe80::21c:42ff:fe00...	fe80::a0b5:71b1:687...	DNS	216	Standard query response 0xd0b0 No such name PTR 3.0.0.0.1.0.0...
131	28.970527	fe80::a0b5:71b1:687...	fe80::21c:42ff:fe00...	DNS	104	Standard query 0x64a2 PTR 252.0.0.224.in-addr.arpa
132	28.990492	fe80::21c:42ff:fe00...	fe80::a0b5:71b1:687...	DNS	161	Standard query response 0x64a2 No such name PTR 252.0.0.224.i...
143	30.649109	fe80::a0b5:71b1:687...	fe80::21c:42ff:fe00...	DNS	109	Standard query 0x9f44 A v10.events.data.microsoft.com
144	30.649598	fe80::a0b5:71b1:687...	fe80::21c:42ff:fe00...	DNS	109	Standard query 0xb3e4 AAAA v10.events.data.microsoft.com
145	30.676792	fe80::21c:42ff:fe00...	fe80::a0b5:71b1:687...	DNS	227	Standard query response 0x9f44 A v10.events.data.microsoft.co...
146	30.679183	fe80::21c:42ff:fe00...	fe80::a0b5:71b1:687...	DNS	274	Standard query response 0xb3e4 AAAA v10.events.data.microsoft...
163	31.261513	fe80::a0b5:71b1:687...	fe80::21c:42ff:fe00...	DNS	152	Standard query 0x0001 PTR 8.1.0.0.0.0.e.f.f.f.2.4.c.1.2.0.0.0...
164	31.263568	fe80::21c:42ff:fe00...	fe80::a0b5:71b1:687...	DNS	152	Standard query response 0x0001 No such name PTR 8.1.0.0.0.0.e...
165	31.269926	fe80::21c:42ff:fe00...	fe80::21c:42ff:fe00...	DNS	110	Standard query 0x0002 A en.wikiversity.org.localdomain
166	31.270080	fe80::21c:42ff:fe00...	fe80::a0b5:71b1:687...	DNS	110	Standard query response 0x0002 No such name A en.wikiversity...
167	31.270272	fe80::a0b5:71b1:687...	fe80::21c:42ff:fe00...	DNS	110	Standard query 0x0003 AAAA en.wikiversity.org.localdomain
168	31.270383	fe80::21c:42ff:fe00...	fe80::a0b5:71b1:687...	DNS	110	Standard query response 0x0003 No such name AAAA en.wikiversi...
169	31.270583	fe80::a0b5:71b1:687...	fe80::21c:42ff:fe00...	DNS	98	Standard query 0x0004 A en.wikiversity.org
170	31.288656	fe80::21c:42ff:fe00...	fe80::a0b5:71b1:687...	DNS	143	Standard query response 0x0004 A en.wikiversity.org CNAME dyn...
171	31.292269	fe80::a0b5:71b1:687...	fe80::21c:42ff:fe00...	DNS	98	Standard query 0x0005 AAAA en.wikiversity.org
172	31.308647	fe80::21c:42ff:fe00...	fe80::a0b5:71b1:687...	DNS	155	Standard query response 0x0005 AAAA en.wikiversity.org CNAME ...
182	31.533274	fe80::a0b5:71b1:687...	fe80::21c:42ff:fe00...	DNS	104	Standard query 0x954a PTR 2.36.114.52.in-addr.arpa
183	31.556918	fe80::21c:42ff:fe00...	fe80::a0b5:71b1:687...	DNS	183	Standard query response 0x954a No such name PTR 2.36.114.52.i...
204	32.407104	fe80::a0b5:71b1:687...	fe80::21c:42ff:fe00...	DNS	152	Standard query 0x4927 PTR 8.1.0.0.0.0.e.f.f.f.2.4.c.1.2.0.0.0...
205	32.409859	fe80::21c:42ff:fe00...	fe80::a0b5:71b1:687...	DNS	152	Standard query response 0x4927 No such name PTR 8.1.0.0.0.0.e...
211	33.251792	fe80::a0b5:71b1:687...	fe80::21c:42ff:fe00...	DNS	152	Standard query 0x0899 PTR 3.0.0.0.1.0.0.0.0.0.0.0.0.0.0.0.0...
212	33.273294	fe80::21c:42ff:fe00...	fe80::a0b5:71b1:687...	DNS	216	Standard query response 0x0899 No such name PTR 3.0.0.0.1.0.0...
222	34.970669	fe80::a0b5:71b1:687...	fe80::21c:42ff:fe00...	DNS	104	Standard query 0x19e1 PTR 252.0.0.224.in-addr.arpa
223	34.990429	fe80::21c:42ff:fe00...	fe80::a0b5:71b1:687...	DNS	161	Standard query response 0x19e1 No such name PTR 252.0.0.224.i...
246	38.394525	fe80::a0b5:71b1:687...	fe80::21c:42ff:fe00...	DNS	152	Standard query 0x9195 PTR 3.0.0.0.1.0.0.0.0.0.0.0.0.0.0.0.0...
247	38.417614	fe80::21c:42ff:fe00...	fe80::a0b5:71b1:687...	DNS	216	Standard query response 0x9195 No such name PTR 3.0.0.0.1.0.0...
253	39.267657	fe80::a0b5:71b1:687...	fe80::21c:42ff:fe00...	DNS	152	Standard query 0xf959 PTR 8.1.0.0.0.0.e.f.f.f.2.4.c.1.2.0.0.0...
254	39.270428	fe80::21c:42ff:fe00...	fe80::a0b5:71b1:687...	DNS	152	Standard query response 0xf959 No such name PTR 8.1.0.0.0.0.e...
259	40.105274	fe80::a0b5:71b1:687...	fe80::21c:42ff:fe00...	DNS	104	Standard query 0x06e2 PTR 252.0.0.224.in-addr.arpa

> Frame 1: 152 bytes on wire (1216 bits), 152 bytes captured (1216 bits) on interface \Device\NPF\_{A90CE091-D694-449E-939A-5123B0C10761}, id 0  
> Ethernet II, Src: Parallel\_a4:60:aa (00:1c:42:a4:60:aa), Dst: Parallel\_00:00:18 (00:1c:42:00:00:18)  
> Internet Protocol Version 6, Src: fe80::a0b5:71b1:687f:859c, Dst: fe80::21c:42ff:fe00:18  
> User Datagram Protocol, Src Port: 64181 (64181), Dst Port: domain (53)

0000 00 1c 42 00 00 18 00 1c 42 a4 60 aa 86 dd 60 08 ..B.... B`....  
0010 50 21 00 62 11 80 fe 80 00 00 00 00 00 a0 b5 P!b....  
0020 71 b1 68 7f 85 9c fe 80 00 00 00 00 00 02 1c q-h....

wireshark\_Ethernet\_20200505165549\_a078 Pakete: 259 · Anzeigt: 56 (21.6%) · Verworfen: Profil: Defa

```
Auswählen C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.18363.778]
(c) 2019 Microsoft Corporation. Alle Rechte vorbehalten.

C:\Users\blauwiggle>ipconfig /flushdns

Windows-IP-Konfiguration

Der DNS-Auflösungscache wurde geleert.

C:\Users\blauwiggle>ipconfig /displaydns

Windows-IP-Konfiguration

psf
-----
Keine Einträge vom Typ AAAA

psf
-----
Eintragsname . . . . . : psf
Eintragstyp . . . . . : 1
Gültigkeitsdauer . . . : 1223
Datenlänge . . . . . : 4
Abschnitt. . . . . : Antwort
(Host-)A-Eintrag . . : 0.0.0.0

.psfc
-----
```

```
Auswählen C:\Windows\system32\cmd.exe

Abschnitt. . . . . : Antwort
PTR-Eintrag . . . . : psf

mac
-----
Keine Einträge vom Typ AAAA

mac
-----
Eintragsname . . . . . : Mac
Eintragstyp . . . . . : 1
Gültigkeitsdauer . . . : 1223
Datenlänge . . . . . : 4
Abschnitt. . . . . : Antwort
(Host-)A-Eintrag . . : 0.0.0.0

C:\Users\blauwiggle>nslookup en.wikiversity.org
Server: UnKnown
Address: fe80::21c:42ff:fe00:18

Nicht autorisierende Antwort:
Name: dyna.wikimedia.org
Addresses: 2620:0:862:ed1a::1
          91.198.174.192
Aliases: en.wikiversity.org
```

## Aufgabe 5

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

Intel(R) Ethernet Connection (2) I219-LM: Ethernet 2

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

arp

No.	Time	Source	Destination	Protocol	Length	Info
54	0.499497	da:40:c6:be:ab:b8	4c:52:62:0e:e0:e6	ARP	60	Who has 141.62.66.1? Tell 141.62.66.239
55	0.499506	4c:52:62:0e:e0:e6	da:40:c6:be:ab:b8	ARP	42	141.62.66.1 is at 4c:52:62:0e:e0:e6
124	3.342520	4c:52:62:0e:53:eb	ff:ff:ff:ff:ff:ff	ARP	60	Who has 141.62.66.104? Tell 141.62.66.4
700	13.828627	da:40:c6:be:ab:b8	ff:ff:ff:ff:ff:ff	ARP	60	Who has 141.62.66.17? Tell 141.62.66.239
764	14.499700	da:40:c6:be:ab:b8	ff:ff:ff:ff:ff:ff	ARP	60	Who has 141.62.66.17? Tell 141.62.66.239
868	15.499583	da:40:c6:be:ab:b8	ff:ff:ff:ff:ff:ff	ARP	60	Who has 141.62.66.17? Tell 141.62.66.239
1107	18.844054	da:40:c6:be:ab:b8	ff:ff:ff:ff:ff:ff	ARP	60	Who has 141.62.66.17? Tell 141.62.66.239
1118	19.499578	da:40:c6:be:ab:b8	ff:ff:ff:ff:ff:ff	ARP	60	Who has 141.62.66.17? Tell 141.62.66.239
1167	20.499551	da:40:c6:be:ab:b8	ff:ff:ff:ff:ff:ff	ARP	60	Who has 141.62.66.17? Tell 141.62.66.239
1230	23.859514	da:40:c6:be:ab:b8	ff:ff:ff:ff:ff:ff	ARP	60	Who has 141.62.66.17? Tell 141.62.66.239
1242	24.499589	da:40:c6:be:ab:b8	ff:ff:ff:ff:ff:ff	ARP	60	Who has 141.62.66.17? Tell 141.62.66.239
1265	25.499717	da:40:c6:be:ab:b8	ff:ff:ff:ff:ff:ff	ARP	60	Who has 141.62.66.17? Tell 141.62.66.239
1299	28.875076	da:40:c6:be:ab:b8	ff:ff:ff:ff:ff:ff	ARP	60	Who has 141.62.66.17? Tell 141.62.66.239
1314	29.499645	da:40:c6:be:ab:b8	ff:ff:ff:ff:ff:ff	ARP	60	Who has 141.62.66.17? Tell 141.62.66.239
1335	30.499370	da:40:c6:be:ab:b8	ff:ff:ff:ff:ff:ff	ARP	60	Who has 141.62.66.17? Tell 141.62.66.239
1412	33.890570	da:40:c6:be:ab:b8	ff:ff:ff:ff:ff:ff	ARP	60	Who has 141.62.66.17? Tell 141.62.66.239
1418	34.499510	da:40:c6:be:ab:b8	ff:ff:ff:ff:ff:ff	ARP	60	Who has 141.62.66.17? Tell 141.62.66.239
1425	35.499532	da:40:c6:be:ab:b8	ff:ff:ff:ff:ff:ff	ARP	60	Who has 141.62.66.17? Tell 141.62.66.239
1426	35.622800	26:c5:04:8a:fa:eb	ff:ff:ff:ff:ff:ff	ARP	60	Who has 141.62.66.113? Tell 141.62.66.236
1434	36.646545	26:c5:04:8a:fa:eb	ff:ff:ff:ff:ff:ff	ARP	60	Who has 141.62.66.113? Tell 141.62.66.236
1443	37.642198	26:c5:04:8a:fa:eb	ff:ff:ff:ff:ff:ff	ARP	60	Who has 141.62.66.112? Tell 141.62.66.236
1444	37.670405	26:c5:04:8a:fa:eb	ff:ff:ff:ff:ff:ff	ARP	60	Who has 141.62.66.113? Tell 141.62.66.236
1463	38.666410	26:c5:04:8a:fa:eb	ff:ff:ff:ff:ff:ff	ARP	60	Who has 141.62.66.112? Tell 141.62.66.236
1497	38.986463	da:40:c6:be:ab:b8	ff:ff:ff:ff:ff:ff	ARP	60	Who has 141.62.66.17? Tell 141.62.66.239
1501	39.499497	da:40:c6:be:ab:b8	ff:ff:ff:ff:ff:ff	ARP	60	Who has 141.62.66.17? Tell 141.62.66.239
1510	39.670491	26:c5:04:8a:fa:eb	ff:ff:ff:ff:ff:ff	ARP	60	Who has 141.62.66.17? Tell 141.62.66.236
1511	39.686352	26:c5:04:8a:fa:eb	ff:ff:ff:ff:ff:ff	ARP	60	Who has 141.62.66.112? Tell 141.62.66.236
1516	40.499477	da:40:c6:be:ab:b8	ff:ff:ff:ff:ff:ff	ARP	60	Who has 141.62.66.17? Tell 141.62.66.239
1517	40.678540	26:c5:04:8a:fa:eb	ff:ff:ff:ff:ff:ff	ARP	60	Who has 141.62.66.17? Tell 141.62.66.236
1518	40.773136	00:0d:b9:4f:b8:14	ff:ff:ff:ff:ff:ff	ARP	60	Who has 141.62.66.230? Tell 141.62.66.250

▼ Frame 4097: 60 bytes on wire (480 bits), 60 bytes captured (480 bits)

Encapsulation type: Ethernet (1)

Arrival Time: May 5, 2020 15:46:46.193278000 Mitteleuropäische Sommerzeit

[Time shift for this packet: 0.000000000 seconds]

Epoch Time: 1588686406.193278000 seconds

[Time delta from previous captured frame: 0.006556000 seconds]

[Time delta from previous displayed frame: 0.999876000 seconds]

[Time since reference or first frame: 111.498874000 seconds]

Frame Number: 4097

Frame Length: 60 bytes (480 bits)

Capture Length: 60 bytes (480 bits)

[Frame is marked: False]

[Frame is ignored: False]

[Protocols in frame: eth:ethertype:arp]

[Coloring Rule Name: ARP]

[Coloring Rule String: arp]

▼ Ethernet II, Src: da:40:c6:be:ab:b8, Dst: ff:ff:ff:ff:ff:ff

```

0000  ff ff ff ff ff ff da 40 c6 be ab b8 00 00 00 01  .....@ .....
0010  00 00 00 04 00 01 da 40 c6 be ab b8 8d 3e 42 ef  .....@ .....>B
0020  00 00 00 00 00 00 8d 3e 42 11 00 00 00 00 00 00  .....>B .....
0030  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....

```

## Wann wird eine ARP-Anfrage gestartet?

Bevor Datenpakete an einen anderen Rechner im Netzwerk gesendet werden können, muss durch das Address Resolution Protocol eine Adressauflösung stattfinden, bei der ein MAC-Broadcast an alle Systeme im Netzwerk gesendet wird. Die ARP-Auflösung unterscheidet zwischen lokalen IP-Adressen und IP-Adressen in einem anderen Subnetz. Anhand der Subnetzmaske wird geprüft, ob sich die IP-Adresse im gleichen Subnetz befindet. Ist das der Fall, wird im ARP-Cache geprüft, ob eine MAC-Adresse für die IP-Adresse bereits hinterlegt ist. Nur wenn keine MAC-Adresse vorhanden ist wird ein ARP-Request gesetzt. Nachdem alle Systeme im Netzwerk diese Anfrage entgegengenommen und ausgewertet haben schickt das gesuchte System ein Antwortpaket unter Angabe seiner MAC-Adresse. In unserem Versuch (s. folgende Abbildung) erfragt der Router die MAC-Adresse eines Rechners im Netzwerk, da er diese nicht im Cache gefunden hat.

Bernd Maier (bm075), Yannick Möller (ym018), Rebecca Mombrei (rm048), Michael Vanhee (mv068)

```

1821 258.052410  AVMAudio_54:f9:01  Micro-St_f6:72:a7  ARP  60 Who has 192.168.178.34? Tell 192.168.178.1
1822 258.052434  Micro-St_f6:72:a7  AVMAudio_54:f9:01  ARP  42 192.168.178.34 is at [REDACTED]

```

## Welcher Rahmentyp wird für die Anfrage verwendet?

Das ARP-Paket ist am Typfeld des Ethernet-Frames zu erkennen. Nummer 0x0806(2054) ist hierbei für das ARP-Protokoll reserviert und man kann dadurch ARP-Pakete von Paketen anderer Protokolle (z.B. IP) unterscheiden. Da ARP-Pakete recht kurz sind werden sie, wenn sie die minimale Framelänge nicht erreichen aufgefüllt (Padding).

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

Nachdem wir mit dem Befehl **“arp -d \*”** unseren ARP-Cache gelöscht haben füllt sich dieser nach einiger Zeit wieder mit Einträgen. Zuerst taucht der HdM-Router (141.62.66.250) wieder im Cache auf, danach kommen kontinuierlich die unten angezeigten Systeme hinzu.

```

C:\Windows\System32>arp /a

Schnittstelle: 141.62.66.1 --- 0x7
Internetadresse   Physische Adresse   Typ
141.62.66.2       4c-52-62-0c-fd-c9   dynamisch
141.62.66.3       4c-52-62-0e-54-2c   dynamisch
141.62.66.12      4c-52-62-0e-e0-e9   dynamisch
141.62.66.214     04-09-73-aa-8a-c0   dynamisch
141.62.66.216     44-31-92-50-6c-61   dynamisch
141.62.66.236     26-c5-04-8a-fa-eb   dynamisch
141.62.66.239     da-40-c6-be-ab-b8   dynamisch
141.62.66.250     00-0d-b9-4f-b8-14   dynamisch
141.62.66.255     ff-ff-ff-ff-ff-ff   statisch
224.0.0.2         01-00-5e-00-00-02   statisch
224.0.0.22        01-00-5e-00-00-16   statisch
224.0.0.251       01-00-5e-00-00-fb   statisch
224.0.0.252       01-00-5e-00-00-fc   statisch
239.255.255.250   01-00-5e-7f-ff-fa   statisch
255.255.255.255   ff-ff-ff-ff-ff-ff   statisch

```

Wir haben den Versuch auch in einem unserer Heimnetze durchgeführt. Dort meldet sich zuerst der Rechner auf dem der Cache gelöscht wurde (192.168.178.34) und erfragt über die Broadcast Adresse die MAC-Adresse des Routers (192.168.178.1). Im Anschluss daran will der Router natürlich auch wissen wer sich hinter der anfragenden IP versteckt und wer sich sonst noch im Netzwerk aufhält. Deswegen schickt er im weiteren Verlauf unserer Aufzeichnung mehrere Request und wartet auf Antworten. In dem Versuchs-Netzwerk sind außerdem ein Smartphone (192.168.178.24) und ein Drucker (192.168.178.40) mittels WLAN verbunden. Diese tauchen am Versuchs-Rechner erst nach einem Ping an die entsprechende Adresse im ARP-Cache auf, da sie wohl zuvor noch keine Daten ausgetauscht haben.



8	5.726836	Micro-St_f6:72:a7	Broadcast	ARP	42 Who has 192.168.178.1? Tell 192.168.178.34
9	5.731611	AVMAudio_54:f9:01	Micro-St_	ARP	60 192.168.178.1 is at
78	26.775463	AVMAudio_54:f9:01	Micro-St_	ARP	60 Who has 192.168.178.34? Tell 192.168.178.1
79	26.775483	Micro-St_f6:72:a7	AVMAudio_	ARP	42 192.168.178.34 is at
183	58.741552	AVMAudio_54:f9:01	Micro-St_	ARP	60 Who has 192.168.178.34? Tell 192.168.178.1
184	58.741576	Micro-St_f6:72:a7	AVMAudio_	ARP	42 192.168.178.34 is at
302	91.439057	AVMAudio_54:f9:01	Broadcast	ARP	60 Who has 192.168.178.2? Tell 192.168.178.1
304	92.432449	AVMAudio_54:f9:01	Broadcast	ARP	60 Who has 192.168.178.2? Tell 192.168.178.1
306	93.432487	AVMAudio_54:f9:01	Broadcast	ARP	60 Who has 192.168.178.2? Tell 192.168.178.1
310	94.432864	AVMAudio_54:f9:01	Broadcast	ARP	60 Who has 192.168.178.2? Tell 192.168.178.1
312	95.432453	AVMAudio_54:f9:01	Broadcast	ARP	60 Who has 192.168.178.2? Tell 192.168.178.1
314	96.048734	AVMAudio_54:f9:01	Broadcast	ARP	60 Who has 192.168.178.26? Tell 192.168.178.1
315	96.432466	AVMAudio_54:f9:01	Broadcast	ARP	60 Who has 192.168.178.2? Tell 192.168.178.1
335	103.505444	AVMAudio_54:f9:01	Broadcast	ARP	60 Who has 192.168.178.24? Tell 192.168.178.1
341	104.502552	AVMAudio_54:f9:01	Broadcast	ARP	60 Who has 192.168.178.24? Tell 192.168.178.1

## Aufgabe 6

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

Im Heimnetzwerk tauchen in Wireshark immer wieder Broadcast-Abfragen per **ARP** auf.

Im nachfolgenden Screenshot ein Ausschnitt der Aufzeichnung. Hier ist zu erkennen, dass

192.168.178.24 (ein Samsung-Fernseher) im 2-Sekunden-Takt die 192.168.178.1 (Fritz!Box) abfragt.

Auch 192.168.178.42 (ein ausgeschalteter Netzwerkdrucker) wird immer wieder durch 192.168.178.52 (ein eingeschalteter Windows-Rechner) angefragt.

2289	16.736182	SamsungE_d9:f6:7b	Broadcast	ARP	64 Who has 192.168.178.1? Tell 192.168.178.24
2313	16.948788	ee:df:70:14:95:58	Broadcast	0x8912	64 Ethernet II
2331	17.092425	f6:b0:14:83:5b:0b	Broadcast	HomePlug AV	60 Qualcomm Atheros, GET_SW.REQ (Get Device/SW Version Request)
2332	17.092579	f6:b0:14:83:5b:0b	Broadcast	0x8912	60 Ethernet II
2408	17.755416	AVM_54:a2:6b	Broadcast	0x8912	60 Ethernet II
2466	18.152855	AVM_54:a2:6b	Broadcast	0x8912	60 Ethernet II
2536	18.765814	SamsungE_d9:f6:7b	Broadcast	ARP	64 Who has 192.168.178.1? Tell 192.168.178.24
2558	18.947244	ee:df:70:14:95:58	Broadcast	0x8912	64 Ethernet II
2579	19.093319	f6:b0:14:83:5b:0b	Broadcast	HomePlug AV	60 Qualcomm Atheros, GET_SW.REQ (Get Device/SW Version Request)
2580	19.093495	f6:b0:14:83:5b:0b	Broadcast	0x8912	60 Ethernet II
2650	19.751319	AVM_54:a2:6b	Broadcast	0x8912	60 Ethernet II
2693	20.153702	AVM_54:a2:6b	Broadcast	0x8912	60 Ethernet II
2774	20.818394	SamsungE_d9:f6:7b	Broadcast	ARP	64 Who has 192.168.178.1? Tell 192.168.178.24
2813	21.093827	f6:b0:14:83:5b:0b	Broadcast	HomePlug AV	60 Qualcomm Atheros, GET_SW.REQ (Get Device/SW Version Request)
2814	21.093827	f6:b0:14:83:5b:0b	Broadcast	0x8912	60 Ethernet II
2894	21.756757	AVM_54:a2:6b	Broadcast	0x8912	60 Ethernet II
2908	21.889441	ASUSTekC_67:e3:21	Broadcast	ARP	60 Who has 192.168.178.42? Tell 192.168.178.52
2941	22.166033	AVM_54:a2:6b	Broadcast	0x8912	60 Ethernet II
3003	22.714248	ASUSTekC_67:e3:21	Broadcast	ARP	60 Who has 192.168.178.42? Tell 192.168.178.52
3022	22.867372	SamsungE_d9:f6:7b	Broadcast	ARP	64 Who has 192.168.178.1? Tell 192.168.178.24
3036	22.955496	ee:df:70:14:95:58	Broadcast	0x8912	64 Ethernet II
3057	23.094153	f6:b0:14:83:5b:0b	Broadcast	HomePlug AV	60 Qualcomm Atheros, GET_SW.REQ (Get Device/SW Version Request)
3058	23.094595	f6:b0:14:83:5b:0b	Broadcast	0x8912	60 Ethernet II
3157	23.714261	ASUSTekC_67:e3:21	Broadcast	ARP	60 Who has 192.168.178.42? Tell 192.168.178.52
3162	23.752652	AVM_54:a2:6b	Broadcast	0x8912	60 Ethernet II
3206	24.165259	AVM_54:a2:6b	Broadcast	0x8912	60 Ethernet II
3233	24.316928	192.168.178.24	192.168.178.255	UDP	77 47367 → 15600 Len=35
3306	24.897903	SamsungE_d9:f6:7b	Broadcast	ARP	64 Who has 192.168.178.1? Tell 192.168.178.24
3313	24.947096	ee:df:70:14:95:58	Broadcast	0x8912	64 Ethernet II

Haben Sie noch weitere Protokolle "eingefangen", die für Sie keinen Sinn machen? Welche sind das?

835.901311	192.168.178.66	224.0.0.251	MDNS	154 Standard query 0x0000 PTR
835.901551	fe80::106c:864c:860...	ff02::fb	MDNS	174 Standard query 0x0000 PTR

Im Heimnetzwerk ließ sich **MDNS** "einfangen". Nach kurzer Recherche per **nslookup** stellte sich heraus, dass es sich um ein iPhone handelt, welches ein Multicast-DNS verwendet (vgl. <https://www.ionos.de/digitalguide/server/knowhow/multicast-dns/> )

835.973422	141.62.75.42	192.168.178.51	STUN	198 Binding Request user: CVM5:lak78xh5hX7kCKFX
835.973776	192.168.178.51	141.62.75.42	STUN	106 Binding Success Response XOR-MAPPED-ADDRESS: 141.62.75.42:23654

Auch **STUN** ließ sich "einfangen" - hier in Verbindung mit dem zeitgleich laufenden BBB-Raum auf dem Server der HdM (vgl. [https://de.wikipedia.org/wiki/Session Traversal Utilities for NAT](https://de.wikipedia.org/wiki/Session_Traversal_Uilities_for_NAT))

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

Das Universal Plug and Play (UPnP) dient zur herstellerübergreifenden Ansteuerung von Geräten wie z.B. Audio-Geräte oder Drucker. In unseren Aufzeichnungen läuft der Dienst auf einem AVM Fritz!Box Router und auf dem dazugehörigen WLAN Repeater. Dabei greift es unter anderem auf das standardisierte Netzwerkprotokoll SSDP (Simple Service Discovery Protocol) zurück, welches die UPnP-Geräte im Netzwerk ausfindig macht. Sobald ein UPnP-Gerät im Netzwerk bekannt wird, muss es seine Existenz mittels UDP über die Multicast-Adresse 239.255.255.250 auf Port 1900 an die Kontrollpunkte melden. Gleichzeitig können diese Kontrollpunkte auch nach Geräten suchen. In den gesendeten "discovery messages" werden nur die wichtigsten Angaben über das Gerät gemacht (Gerätenamen, Typ). Möchte man nun mit Wireshark den UPnP/SSDP Traffic "fischen" erreicht man dies mittels eines UDP-Port Filters auf Port 1900 (s. Abb.)



udp.port==1900						
No.	Time	Source	Destination	Protocol	Length	Info
1304	138.179137	192.168.178.24	239.255.255.250	SSDP	136	M-SEARCH * HTTP/1.1
1305	138.181612	192.168.178.24	239.255.255.250	SSDP	136	M-SEARCH * HTTP/1.1
1412	167.962942	192.168.178.1	239.255.255.250	SSDP	338	NOTIFY * HTTP/1.1
1413	167.964008	192.168.178.1	239.255.255.250	SSDP	347	NOTIFY * HTTP/1.1
1414	167.979012	192.168.178.1	239.255.255.250	SSDP	390	NOTIFY * HTTP/1.1
Checksum: 0x1e47 [unverified] [Checksum Status: Unverified] [Stream index: 26] [Timestamps] [Time since first frame: 0.000000000 seconds] [Time since previous frame: 0.000000000 seconds]						
Simple Service Discovery Protocol						
> NOTIFY * HTTP/1.1\r\n						
HOST: 239.255.255.250:1900\r\n						
LOCATION: http://192.168.178.1:49000/MediaServerDevDesc.xml\r\n						
SERVER: router UPnP/1.0 AVM FRITZ!Box 7430 146.07.12\r\n						
CACHE-CONTROL: max-age=1800\r\n						
NT: upnp:rootdevice\r\n						
NTS: ssdp:alive\r\n						
USN: uuid:fa095ecc-e13e-40e7-8e6c-e8df7054f901::upnp:rootdevice\r\n\r\n						
0000	01 00 5e 7f ff fa e8 df	70 54 f9 01 08 00 45 00	..^..... pT....E.			
0010	01 44 18 14 00 00 04 11	3a f1 c0 a8 b2 01 ef ff	.D..... :.....			
0020	ff fa 07 6c 07 6c 01 30	1e 47 4e 4f 54 49 46 59	...1.1.0 .GNOTIFY			
0030	20 2a 20 48 54 54 50 2f	31 2e 31 0d 0a 48 4f 53	* HTTP/ 1.1..HOS			
0040	54 3a 20 32 33 39 2e 32	35 35 2e 32 35 35 2e 32	T: 239.2 55.255.2			
0050	35 30 3a 31 39 30 30 0d	0a 4c 4f 43 41 54 49 4f	50:1900. .LOCATIO			
0060	4e 3a 20 68 74 74 70 3a	2f 2f 31 39 32 2e 31 36	N: http: //192.16			
0070	38 2e 31 37 38 2e 31 3a	34 39 30 30 30 2f 4d 65	8.178.1: 49000/Me			
0080	64 69 61 53 65 72 76 65	72 44 65 76 44 65 73 63	diaServe rDevDesc			
0090	2e 78 6d 6c 0d 0a 53 45	52 56 45 52 3a 20 72 6f	.xml..SE RVER: ro			
00a0	75 74 65 72 20 55 50 6e	50 2f 31 2e 30 20 41 56	uter UPn P/1.0 AV			
00b0	4d 20 46 52 49 54 5a 21	42 6f 78 20 37 34 33 30	M FRITZ! Box 7430			
00c0	20 31 34 36 2e 30 37 2e	31 32 0d 0a 43 41 43 48	146.07. 12..CACH			
00d0	45 2d 43 4f 4e 54 52 4f	4c 3a 20 6d 61 78 2d 61	E-CONTRO L: max-a			
00e0	67 65 3d 31 38 30 30 0d	0a 4e 54 3a 20 75 70 6e	ge=1800. .NT: upn			
00f0	70 3a 72 6f 6f 74 64 65	76 69 63 65 0d 0a 4e 54	p:rootde vice..NT			
0100	53 3a 20 73 73 64 70 3a	61 6c 69 76 65 0d 0a 55	S: ssdp: alive..U			



# Aufgabe 7

## TCP 3-Way-Handshake

Der Three-Way Handshake von TCP lässt sich dadurch erkennen, dass das SYN Flag gesetzt wird:

369	5.874336	141.62.66.2	141.62.1.53	TCP	66 50856 → http(80) [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
370	5.875378	141.62.1.53	141.62.66.2	TCP	66 http(80) → 50856 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 WS=128
371	5.875408	141.62.66.2	141.62.1.53	TCP	54 50856 → http(80) [ACK] Seq=1 Ack=1 Win=262656 Len=0
372	5.875507	141.62.66.2	141.62.1.53	HTTP	432 GET / HTTP/1.1
373	5.877227	141.62.1.53	141.62.66.2	TCP	60 http(80) → 50856 [ACK] Seq=1 Ack=379 Win=30336 Len=0
374	5.878004	141.62.1.53	141.62.66.2	HTTP	505 HTTP/1.1 302 Found (text/html)
375	5.881985	141.62.66.2	141.62.1.53	TCP	66 50857 → https(443) [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
377	5.883052	141.62.1.53	141.62.66.2	TCP	66 https(443) → 50857 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 WS=128
378	5.883081	141.62.66.2	141.62.1.53	TCP	54 50857 → https(443) [ACK] Seq=1 Ack=1 Win=262656 Len=0
379	5.883801	141.62.66.2	141.62.1.53	TLSv...	571 Client Hello
380	5.884853	141.62.1.53	141.62.66.2	TCP	60 https(443) → 50857 [ACK] Seq=1 Ack=518 Win=30336 Len=0
381	5.885664	141.62.1.53	141.62.66.2	TLSv...	1514 Server Hello
382	5.885949	141.62.1.53	141.62.66.2	TCP	1514 https(443) → 50857 [ACK] Seq=1461 Ack=518 Win=30336 Len=1460 [TCP segment of a reassembled PDU]
383	5.885958	141.62.66.2	141.62.1.53	TCP	54 50857 → https(443) [ACK] Seq=518 Ack=2921 Win=262656 Len=0
384	5.886172	141.62.1.53	141.62.66.2	TCP	1230 https(443) → 50857 [PSH, ACK] Seq=2921 Ack=518 Win=30336 Len=1176 [TCP segment of a reassembled PD...
385	5.896373	141.62.1.53	141.62.66.2	TCP	1514 https(443) → 50857 [ACK] Seq=4097 Ack=518 Win=30336 Len=1460 [TCP segment of a reassembled PDU]
386	5.896373	141.62.1.53	141.62.66.2	TLSv...	1514 Certificate [TCP segment of a reassembled PDU]

Im ersten Teil des 3-Way Handshakes (Zeile 1 im obigen Bild) wird vom anfragenden Host nur das SYN-Flag gesetzt. Der empfangende Host (hier der Server, an welchen die Anfrage gesendet wird) erkennt dadurch, dass es sich um eine Anfrage für einen Verbindungsaufbau handelt. Der anfragende Rechner sendet außerdem eine ISN (Initial Sequence Number), welche zufällig generiert wurde. Ist der Server bereit, die Verbindung aufzubauen, so sendet er eine Antwort (Zeile 2 im obigen Bild), in welcher die Flags SYN und ACK gesetzt werden. Dies bedeutet, dass sowohl die Felder "Sequence Number", in welchem er die eigene ISN sendet, und "Acknowledgement Number", in welchem die um 1 erhöhte ISN des Partners steht, belegt sind. Im letzten Teil des 3-Way Handshake bestätigt der anfragende Rechner, dass er die ISN des Servers erhalten hat, indem er das ACK-Flag setzt (Zeile 3) und im Feld "Acknowledge Number" die um 1 erhöhte ISN des Servers zurücksendet. Er übernimmt außerdem als "Sequence Number" den erhöhten Wert. Im Feld "Window Size" senden beide Teilnehmer ihre aktuelle Puffergröße, sie teilen dem Partner also jeweils mit wie viele Bytes unaufgefordert gesendet werden dürfen.

Mithilfe von Wireshark lassen sich außerdem alle weiteren Optionen anzeigen, welche aktiviert sind:

MSS (Maximum Segment Size)	Maximale Größe des Datenfeldes, um Fragmentierung zu vermeiden
WS (Window Scale Factor)	Faktor, mit welchem die Window Size multipliziert werden kann, um den Wert weiter zu erhöhen. Funktioniert nur, wenn beide Teilnehmer dies unterstützen.
SACK_PERM	Selective Acknowledgement SACK: Möglichkeit, bei fehlenden Segmenten nur die fehlenden Pakete erneut zu senden. Ohne die Fehlererkennung mit SACK werden alle Pakete ab dem fehlenden Paket erneut gesendet. SACK_PERM gibt an, ob die Funktion unterstützt wird.

## Verwendung der Portnummern

Für die ausgehende Verbindung nutzt der Rechner einen dynamisch allokierten Port (Portnummer zwischen 49.152 - 65.535). Die TCP-Verbindungsanfrage wird über HTTP gesendet, also an den Port 80. Der Server antwortet auf den Port, von welchem aus die TCP-Anfrage gesendet wurde. Dass TCP mehrere Verbindungen gleichzeitig managen kann (Multiplexing), sieht man hier daran, dass von einem anderen Port aus ein HTTPS-Request gesendet wird. Bei HTTPS lässt sich außerdem beobachten, dass es auf mehreren Ports gleichzeitig Daten sendet. Natürlich wird für jeden Port ein extra Handshake ausgeführt. Im Bild unten lässt sich erkennen, dass gleichzeitig mehrere Ports an den HTTPS-Port 443 senden und Daten von dort empfangen.

```
50856 → http(80) [S
http(80) → 50856 [S
50856 → http(80) [A
GET / HTTP/1.1
http(80) → 50856 [A
HTTP/1.1 302 Found
50857 → https(443)
https(443) → 50857
```

534	6.151169	141.62.1.53	141.62.66.2	TCP	1514 https(443) → 50858 [ACK] Seq=21498 Ack=1076 Win=31360 Len=1460 [TCP segment of a reassembled PDU]
535	6.151178	141.62.66.2	141.62.1.53	TCP	54 50858 → https(443) [ACK] Seq=1076 Ack=22958 Win=262656 Len=0
536	6.151187	141.62.1.53	141.62.66.2	TLSv1.2	1514 Application Data [TCP segment of a reassembled PDU]
537	6.151197	141.62.1.53	141.62.66.2	TCP	1514 https(443) → 50858 [ACK] Seq=24418 Ack=1076 Win=31360 Len=1460 [TCP segment of a reassembled PDU]
538	6.151202	141.62.66.2	141.62.1.53	TCP	54 50858 → https(443) [ACK] Seq=1076 Ack=25878 Win=262656 Len=0
539	6.151209	141.62.1.53	141.62.66.2	TLSv1.2	1399 Application Data, Application Data
540	6.151590	141.62.66.2	141.62.1.53	TLSv1.2	474 Application Data
541	6.151882	141.62.66.2	141.62.1.53	TLSv1.2	458 Application Data
542	6.152039	141.62.66.2	141.62.1.53	TLSv1.2	463 Application Data
543	6.152652	141.62.66.2	141.62.1.53	TCP	66 50860 → https(443) [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_P
544	6.153168	141.62.66.2	141.62.1.53	TCP	66 50861 → https(443) [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_P
545	6.153593	141.62.1.53	141.62.66.2	TCP	66 https(443) → 50860 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SA
546	6.153633	141.62.66.2	141.62.1.53	TCP	54 50860 → https(443) [ACK] Seq=1 Ack=1 Win=262656 Len=0
547	6.153886	141.62.1.53	141.62.66.2	TCP	66 https(443) → 50861 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SA
548	6.153923	141.62.66.2	141.62.1.53	TCP	54 50861 → https(443) [ACK] Seq=1 Ack=1 Win=2102272 Len=0
549	6.154593	141.62.66.2	141.62.1.53	TLSv1.2	571 Client Hello
550	6.155312	141.62.66.2	141.62.1.53	TLSv1.2	571 Client Hello
551	6.155701	141.62.1.53	141.62.66.2	TCP	60 https(443) → 50860 [ACK] Seq=1 Ack=518 Win=30336 Len=0
552	6.156516	141.62.1.53	141.62.66.2	TLSv1.2	1514 Server Hello
553	6.156852	141.62.1.53	141.62.66.2	TCP	1514 https(443) → 50860 [ACK] Seq=1461 Ack=518 Win=30336 Len=1460 [TCP segment of a reassembled PDU]
554	6.156852	141.62.1.53	141.62.66.2	TCP	1230 https(443) → 50860 [PSH, ACK] Seq=2921 Ack=518 Win=30336 Len=1176 [TCP segment of a reassembled PDU]
555	6.156853	141.62.1.53	141.62.66.2	TCP	60 https(443) → 50861 [ACK] Seq=1 Ack=518 Win=30336 Len=0
556	6.156878	141.62.66.2	141.62.1.53	TCP	54 50860 → https(443) [ACK] Seq=518 Ack=4097 Win=262656 Len=0
557	6.157450	141.62.1.53	141.62.66.2	TLSv1.2	1514 Server Hello
558	6.157451	141.62.1.53	141.62.66.2	TCP	1514 https(443) → 50861 [ACK] Seq=1461 Ack=518 Win=30336 Len=1460 [TCP segment of a reassembled PDU]
559	6.157477	141.62.66.2	141.62.1.53	TCP	54 50861 → https(443) [ACK] Seq=518 Ack=2921 Win=2102272 Len=0
560	6.157495	141.62.1.53	141.62.66.2	TCP	1230 https(443) → 50861 [PSH, ACK] Seq=2921 Ack=518 Win=30336 Len=1176 [TCP segment of a reassembled PDU]
561	6.160444	141.62.1.53	141.62.66.2	TCP	1514 https(443) → 50859 [ACK] Seq=7451 Ack=1481 Win=32512 Len=1460 [TCP segment of a reassembled PDU]
562	6.160445	141.62.1.53	141.62.66.2	TLSv1.2	142 Application Data
563	6.160480	141.62.66.2	141.62.1.53	TCP	54 50859 → https(443) [ACK] Seq=1481 Ack=8999 Win=262656 Len=0
565	6.165152	141.62.1.53	141.62.66.2	TLSv1.2	516 Application Data
566	6.165560	141.62.1.53	141.62.66.2	TCP	1514 https(443) → 50857 [ACK] Seq=61855 Ack=1938 Win=33536 Len=1460 [TCP segment of a reassembled PDU]
567	6.165606	141.62.1.53	141.62.66.2	TLSv1.2	521 Application Data
568	6.165615	141.62.66.2	141.62.1.53	TCP	54 50857 → https(443) [ACK] Seq=1938 Ack=63782 Win=262656 Len=0
569	6.168565	141.62.1.53	141.62.66.2	TCP	1514 https(443) → 50860 [ACK] Seq=4097 Ack=518 Win=30336 Len=1460 [TCP segment of a reassembled PDU]
570	6.168592	141.62.66.2	141.62.1.53	TCP	54 50860 → https(443) [ACK] Seq=518 Ack=5557 Win=262656 Len=0
571	6.168813	141.62.1.53	141.62.66.2	TLSv1.2	1514 Certificate [TCP segment of a reassembled PDU]
572	6.168814	141.62.1.53	141.62.66.2	TLSv1.2	194 Server Key Exchange, Server Hello Done
573	6.168833	141.62.66.2	141.62.1.53	TCP	54 50860 → https(443) [ACK] Seq=518 Ack=7157 Win=262656 Len=0
574	6.169717	141.62.1.53	141.62.66.2	TCP	1514 https(443) → 50861 [ACK] Seq=4097 Ack=518 Win=30336 Len=1460 [TCP segment of a reassembled PDU]
575	6.169769	141.62.66.2	141.62.1.53	TCP	54 50861 → https(443) [ACK] Seq=518 Ack=5557 Win=2102272 Len=0
576	6.169907	141.62.1.53	141.62.66.2	TLSv1.2	1514 Certificate [TCP segment of a reassembled PDU]
577	6.169908	141.62.1.53	141.62.66.2	TLSv1.2	194 Server Key Exchange, Server Hello Done
578	6.169928	141.62.66.2	141.62.1.53	TCP	54 50861 → https(443) [ACK] Seq=518 Ack=7157 Win=2102272 Len=0
579	6.171013	141.62.66.2	141.62.1.53	TLSv1.2	180 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message

## Veränderung des TCP-Ablaufs durch Navigieren auf der Website

Wird auf der Webseite ein Link oder ein Bild angeklickt, so wird ein neuer http-Request an Port 80 mithilfe von TCP gesendet. Hierbei wird vom eigenen Rechner aus ein anderer Port verwendet, als für die vorherige Seite. Allerdings wird kein erneuter Handshake ausgeführt. Dies sieht man am folgenden Ausschnitt aus Wireshark (gefiltert nach TCP-Anfragen auf Port 80):

Bernd Maier (bm075), Yannick Möller (ym018), Rebecca Mombrei (rm048), Michael Vanhee (mv068)

170	2.583356	141.62.66.2	93.184.220.29	TCP	55 50851 → http(80) [ACK] Seq=1 Ack=1 Win=1023 Len=1
171	2.587989	93.184.220.29	141.62.66.2	TCP	66 http(80) → 50851 [ACK] Seq=1 Ack=2 Win=135 Len=0 SLE=1 SRE=2
369	5.874336	141.62.66.2	141.62.1.53	TCP	66 50856 → http(80) [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
370	5.875378	141.62.1.53	141.62.66.2	TCP	66 http(80) → 50856 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 WS=
371	5.875408	141.62.66.2	141.62.1.53	TCP	54 50856 → http(80) [ACK] Seq=1 Ack=1 Win=262656 Len=0
372	5.875507	141.62.66.2	141.62.1.53	HTTP	432 GET / HTTP/1.1
373	5.877227	141.62.1.53	141.62.66.2	TCP	60 http(80) → 50856 [ACK] Seq=1 Ack=379 Win=30336 Len=0
374	5.878004	141.62.1.53	141.62.66.2	HTTP	505 HTTP/1.1 302 Found (text/html)
393	5.926979	141.62.66.2	141.62.1.53	TCP	54 50856 → http(80) [ACK] Seq=379 Ack=452 Win=262144 Len=0
3189	9.395965	141.62.66.2	216.58.207.67	TCP	55 50841 → http(80) [ACK] Seq=1 Ack=1 Win=1021 Len=1
3190	9.400823	216.58.207.67	141.62.66.2	TCP	66 http(80) → 50841 [ACK] Seq=1 Ack=2 Win=242 Len=0 SLE=1 SRE=2
3570	10.880762	141.62.66.2	141.62.1.53	TCP	54 50856 → http(80) [FIN, ACK] Seq=379 Ack=452 Win=262144 Len=0
3571	10.881874	141.62.1.53	141.62.66.2	TCP	60 http(80) → 50856 [FIN, ACK] Seq=452 Ack=380 Win=30336 Len=0
3572	10.881959	141.62.66.2	141.62.1.53	TCP	54 50856 → http(80) [ACK] Seq=380 Ack=453 Win=262144 Len=0
3751	12.598939	141.62.66.2	93.184.220.29	TCP	55 [TCP Keep-Alive] 50851 → http(80) [ACK] Seq=1 Ack=1 Win=1023 Len=1
3752	12.604279	93.184.220.29	141.62.66.2	TCP	66 [TCP Keep-Alive ACK] http(80) → 50851 [ACK] Seq=1 Ack=2 Win=135 Len=0 SLE=1 SRE=

Die Zeile mit der Nummer 3189 bildet den Klick auf einen Link auf der Seite ab. Von dort an wird der Port 50841 genutzt, während zuvor der Port 50856 genutzt wurde. Daraufhin wird die Verbindung auf dem Port 50856 mit dem FIN Flag beendet (Zeilennummer 3570).

## Aufgabe 8

Welche MAC-Adresse sind sonst noch in Ihrem Netzwerk zu finden?

Man kann unter Mac und Linux das Tool arp-scan benutzen. arp-scan sendet ARP Pakete an Hosts im lokalen Netzwerk und zeigt dann die Antworten der Hosts an.

```

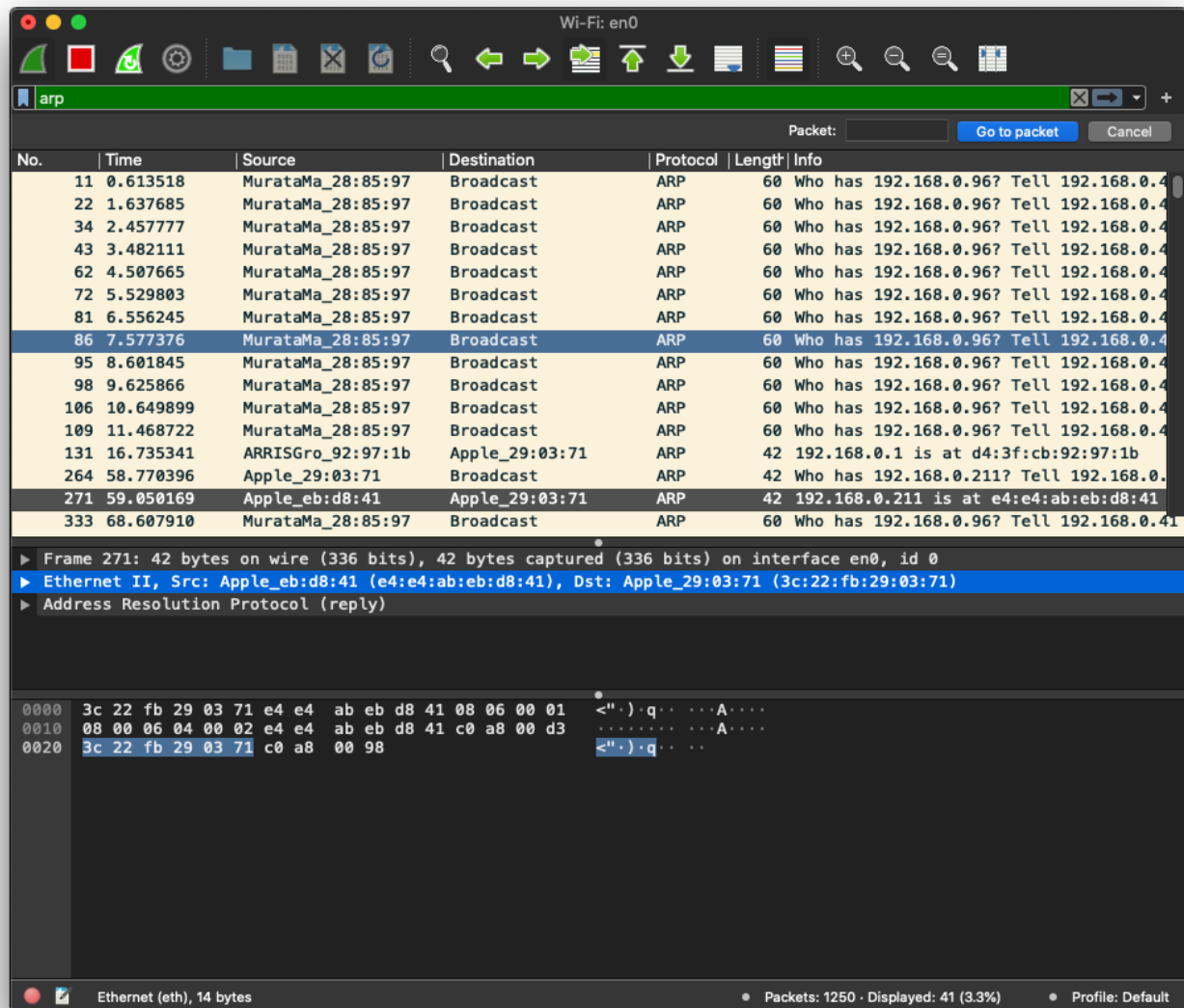
blauwiggle@CXMP: ~/HdM/5/Praktikum-Rechnernetze
> arp-scan -l -s 192.168.0.0
Interface: en0, type: EN10MB, MAC: 3c:22:fb:29:03:71, IPv4: 192.168.0.152
Starting arp-scan 1.9.7 with 256 hosts (https://github.com/royhills/arp-scan)
192.168.0.41    44:91:60:28:85:97    Murata Manufacturing Co., Ltd.
192.168.0.177  50:de:06:a7:ae:2a    Apple, Inc.
192.168.0.251  4c:e1:73:49:c7:55    IEEE Registration Authority
192.168.0.135  ec:e5:12:10:92:18    tado GmbH

514 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.9.7: 256 hosts scanned in 1.866 seconds (137.19 hosts/sec). 4
responded

```

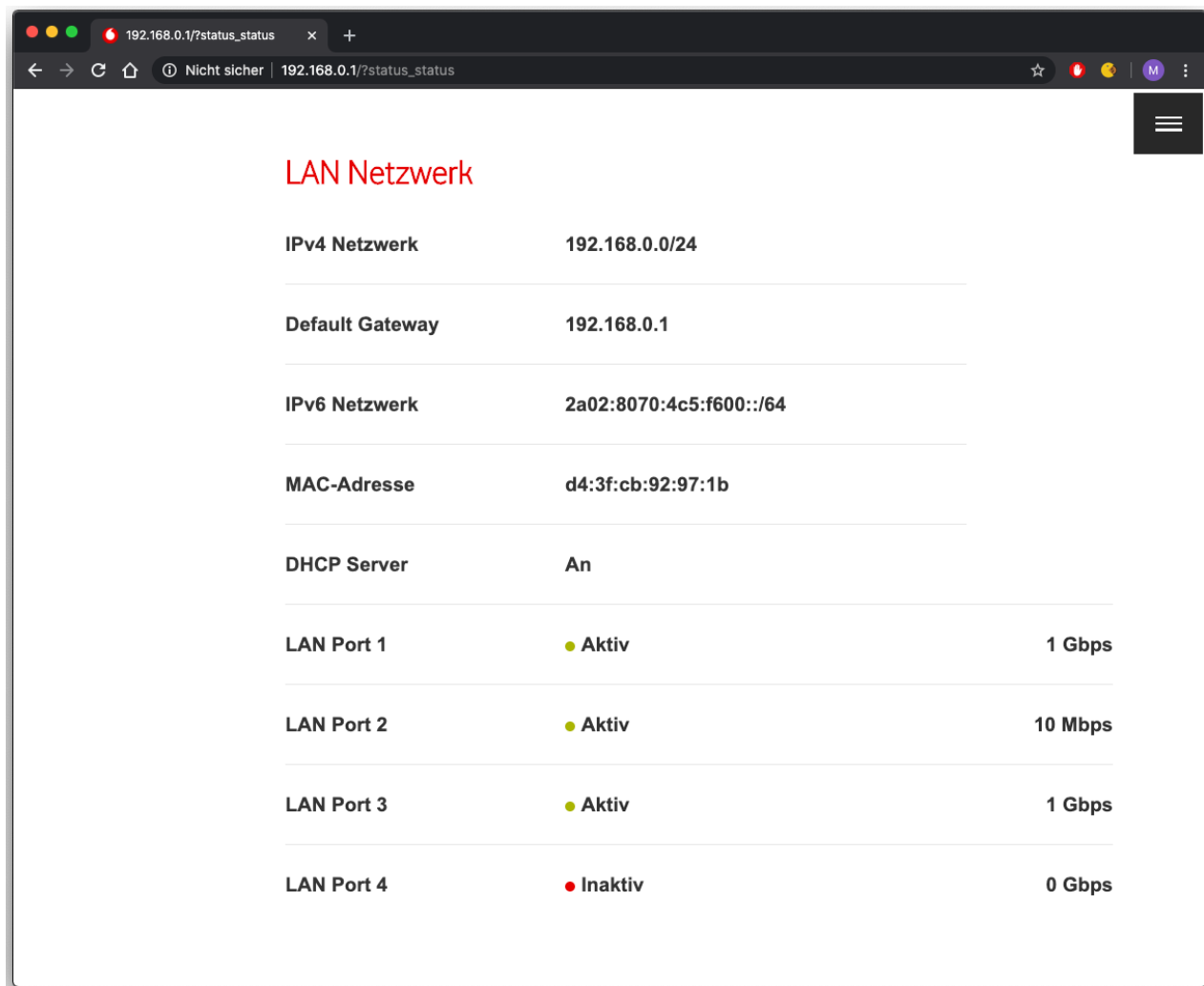


Man kann in Wireshark, nachdem man nach arp gefiltert hat, die selben Geräte erkennen. Auffällig oft versendet das Murata Gerät einen Broadcast. Dabei handelt es sich um das IKEA TRADFRI Gateway, welches als IoT Gateway dient.

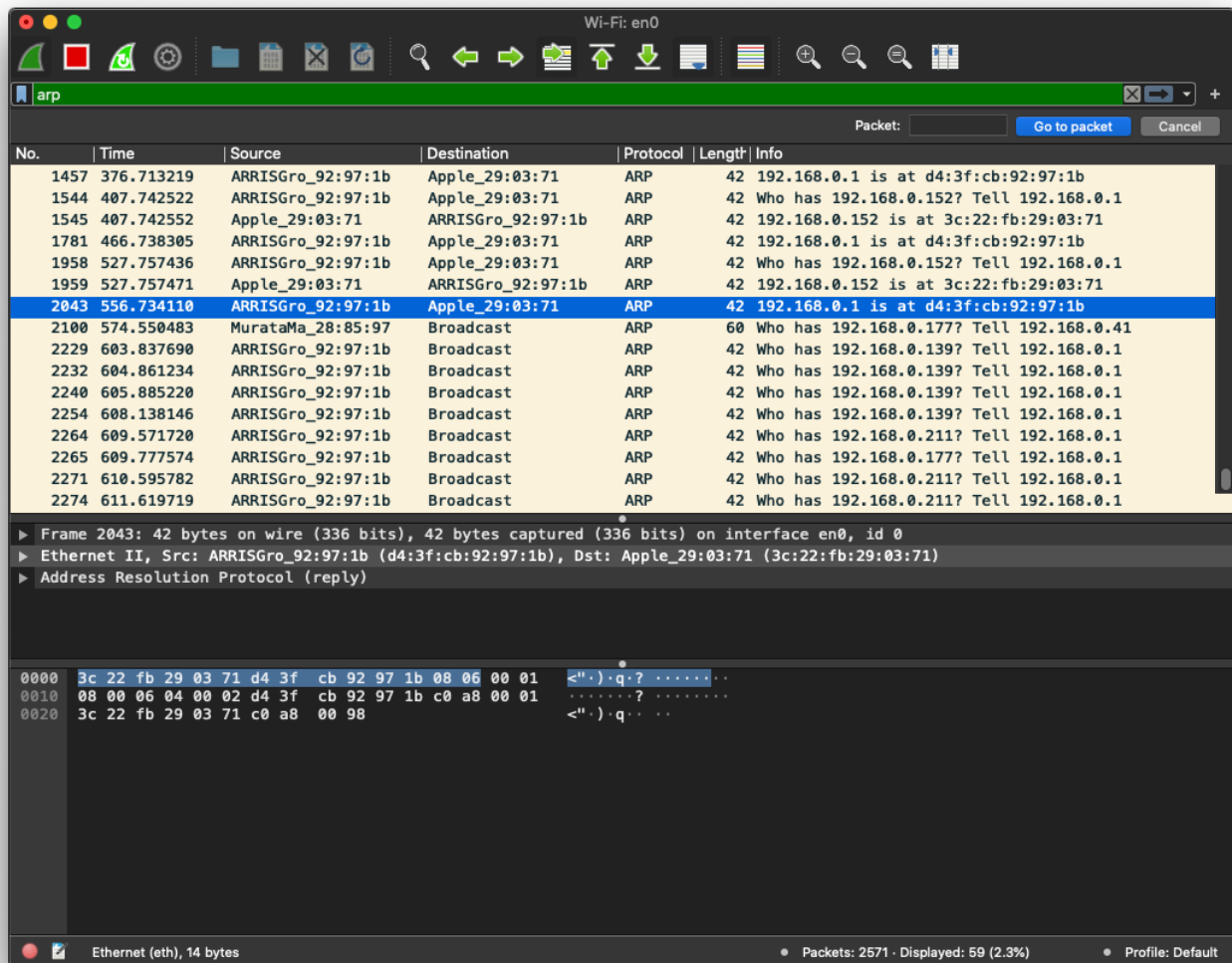


## Welche MAC-Adresse hat der Router?

Die MAC Adresse meines Routers lautet D4:3F:CB:92:97:1B, man kann das im Interface des Routers nachsehen. Ein Blick in Wireshark bestätigt es.



<b>LAN Netzwerk</b>		
IPv4 Netzwerk	192.168.0.0/24	
Default Gateway	192.168.0.1	
IPv6 Netzwerk	2a02:8070:4c5:f600::/64	
MAC-Adresse	d4:3f:cb:92:97:1b	
DHCP Server	An	
LAN Port 1	● Aktiv	1 Gbps
LAN Port 2	● Aktiv	10 Mbps
LAN Port 3	● Aktiv	1 Gbps
LAN Port 4	● Inaktiv	0 Gbps



Welche MAC-Adresse hat der Server 141.62.1.5 ?

Der Server hat die MAC-Adresse 00:0d:b9:4f:b8:14, welche von der IEEE Standards Association an den Hersteller "PC Engines GmbH" in der Flughafenstrasse 58, 8152 Glattbrugg (Schweiz) vergeben wurde.





# Aufgabe 10

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

Bei Livestreaming des aktuellen ARD-Programms (<https://live.daserste.de/>) werden in diesem Fall die TCP-Ports 62761 und 62762 genutzt.

2955...	790.621602	8.241.88.124	192.168.178.51	TCP	1506	https(443) → 62762	[ACK]	Seq=83629082	Ack=64947	Win=63640	Len=1452	[TCP segment of a reassembled PDU]
2955...	790.621626	8.241.88.124	192.168.178.51	TCP	1506	https(443) → 62762	[ACK]	Seq=836390534	Ack=64947	Win=63640	Len=1452	[TCP segment of a reassembled PDU]
2955...	790.621639	192.168.178.51	8.241.88.124	TCP	54	62762 → https(443)	[ACK]	Seq=64947	Ack=83631986	Win=65340	Len=0	
2955...	790.623921	8.241.88.124	192.168.178.51	TLv1.3	1506	Application Data	[TCP segment of a reassembled PDU]					
2955...	790.624064	8.241.88.124	192.168.178.51	TCP	1506	https(443) → 62762	[ACK]	Seq=83633438	Ack=64947	Win=63640	Len=1452	[TCP segment of a reassembled PDU]
2955...	790.624086	192.168.178.51	8.241.88.124	TCP	54	62762 → https(443)	[ACK]	Seq=64947	Ack=83634890	Win=65340	Len=0	
2955...	790.624310	8.241.88.124	192.168.178.51	TCP	1506	https(443) → 62762	[ACK]	Seq=83634890	Ack=64947	Win=63640	Len=1452	[TCP segment of a reassembled PDU]
2955...	790.624510	8.241.88.124	192.168.178.51	TCP	1506	https(443) → 62762	[ACK]	Seq=83636342	Ack=64947	Win=63640	Len=1452	[TCP segment of a reassembled PDU]
2955...	790.624531	192.168.178.51	8.241.88.124	TCP	54	62762 → https(443)	[ACK]	Seq=64947	Ack=83637794	Win=65340	Len=0	
2955...	790.624680	8.241.88.124	192.168.178.51	TCP	1506	https(443) → 62762	[ACK]	Seq=83637794	Ack=64947	Win=63640	Len=1452	[TCP segment of a reassembled PDU]
2955...	790.624831	8.241.88.124	192.168.178.51	TCP	1506	https(443) → 62762	[ACK]	Seq=83639246	Ack=64947	Win=63640	Len=1452	[TCP segment of a reassembled PDU]
2955...	790.624856	192.168.178.51	8.241.88.124	TCP	54	62762 → https(443)	[ACK]	Seq=64947	Ack=83640698	Win=65340	Len=0	
2955...	790.625003	8.241.88.124	192.168.178.51	TCP	1506	https(443) → 62762	[ACK]	Seq=83640698	Ack=64947	Win=63640	Len=1452	[TCP segment of a reassembled PDU]
2955...	790.625414	8.241.88.124	192.168.178.51	TCP	1506	https(443) → 62762	[ACK]	Seq=83642150	Ack=64947	Win=63640	Len=1452	[TCP segment of a reassembled PDU]
2955...	790.625436	192.168.178.51	8.241.88.124	TCP	54	62762 → https(443)	[ACK]	Seq=64947	Ack=83643602	Win=65340	Len=0	
2955...	790.626703	8.241.88.124	192.168.178.51	TCP	1506	https(443) → 62762	[ACK]	Seq=83643602	Ack=64947	Win=63640	Len=1452	[TCP segment of a reassembled PDU]
2955...	790.626704	8.241.88.124	192.168.178.51	TCP	1506	https(443) → 62762	[ACK]	Seq=83645054	Ack=64947	Win=63640	Len=1452	[TCP segment of a reassembled PDU]
2955...	790.626730	192.168.178.51	8.241.88.124	TCP	54	62762 → https(443)	[ACK]	Seq=64947	Ack=83646506	Win=65340	Len=0	
2955...	790.641448	8.241.88.124	192.168.178.51	TLv1.3	1506	https(443) → 62762	[ACK]	Seq=83646506	Ack=64947	Win=63640	Len=1452	[TCP segment of a reassembled PDU]
2955...	790.641484	192.168.178.51	8.241.88.124	TCP	54	62762 → https(443)	[ACK]	Seq=64947	Ack=83648458	Win=65340	Len=0	
2958...	793.842412	192.168.178.51	8.241.88.124	TLv1.3	630	Application Data						
2958...	793.861018	8.241.88.124	192.168.178.51	TCP	1506	https(443) → 62762	[ACK]	Seq=83648458	Ack=65523	Win=65088	Len=1452	[TCP segment of a reassembled PDU]
2958...	793.861074	8.241.88.124	192.168.178.51	TCP	1506	https(443) → 62762	[ACK]	Seq=83649910	Ack=65523	Win=65088	Len=1452	[TCP segment of a reassembled PDU]
2958...	793.861092	192.168.178.51	8.241.88.124	TCP	54	62762 → https(443)	[ACK]	Seq=65523	Ack=83651362	Win=65340	Len=0	
2958...	793.861304	8.241.88.124	192.168.178.51	TCP	1506	https(443) → 62762	[ACK]	Seq=83651362	Ack=65523	Win=65088	Len=1452	[TCP segment of a reassembled PDU]
2958...	793.861516	8.241.88.124	192.168.178.51	TCP	1506	https(443) → 62762	[ACK]	Seq=83652814	Ack=65523	Win=65088	Len=1452	[TCP segment of a reassembled PDU]
2958...	793.861549	192.168.178.51	8.241.88.124	TCP	54	62762 → https(443)	[ACK]	Seq=65523	Ack=83654266	Win=65340	Len=0	
2958...	793.863323	8.241.88.124	192.168.178.51	TCP	1506	https(443) → 62762	[ACK]	Seq=83654266	Ack=65523	Win=65088	Len=1452	[TCP segment of a reassembled PDU]
2958...	793.863716	8.241.88.124	192.168.178.51	TCP	1506	https(443) → 62762	[ACK]	Seq=83655718	Ack=65523	Win=65088	Len=1452	[TCP segment of a reassembled PDU]
2958...	793.863720	8.241.88.124	192.168.178.51	TCP	1506	https(443) → 62762	[ACK]	Seq=83657170	Ack=65523	Win=65088	Len=1452	[TCP segment of a reassembled PDU]
2958...	793.863752	192.168.178.51	8.241.88.124	TCP	54	62762 → https(443)	[ACK]	Seq=65523	Ack=83658622	Win=65340	Len=0	
2958...	793.863946	8.241.88.124	192.168.178.51	TCP	1506	https(443) → 62762	[ACK]	Seq=83658622	Ack=65523	Win=65088	Len=1452	[TCP segment of a reassembled PDU]
2958...	793.863988	192.168.178.51	8.241.88.124	TCP	54	62762 → https(443)	[ACK]	Seq=65523	Ack=83660074	Win=65340	Len=0	
2958...	793.864013	8.241.88.124	192.168.178.51	TCP	1506	https(443) → 62762	[ACK]	Seq=83660074	Ack=65523	Win=65088	Len=1452	[TCP segment of a reassembled PDU]
2958...	793.866598	8.241.88.124	192.168.178.51	TCP	1506	https(443) → 62762	[ACK]	Seq=83661526	Ack=65523	Win=65088	Len=1452	[TCP segment of a reassembled PDU]
2958...	793.866598	8.241.88.124	192.168.178.51	TCP	1506	https(443) → 62762	[ACK]	Seq=83662978	Ack=65523	Win=65088	Len=1452	[TCP segment of a reassembled PDU]
2958...	793.866642	8.241.88.124	192.168.178.51	TLv1.3	1506	Application Data	[TCP segment of a reassembled PDU]					
2958...	793.866643	8.241.88.124	192.168.178.51	TCP	1506	https(443) → 62762	[ACK]	Seq=83665882	Ack=65523	Win=65088	Len=1452	[TCP segment of a reassembled PDU]
2958...	793.866644	8.241.88.124	192.168.178.51	TCP	1506	https(443) → 62762	[ACK]	Seq=83667334	Ack=65523	Win=65088	Len=1452	[TCP segment of a reassembled PDU]
2958...	793.866700	192.168.178.51	8.241.88.124	TCP	54	62762 → https(443)	[ACK]	Seq=65523	Ack=83668786	Win=65340	Len=0	
2958...	793.866872	8.241.88.124	192.168.178.51	TCP	1506	https(443) → 62762	[ACK]	Seq=83668786	Ack=65523	Win=65088	Len=1452	[TCP segment of a reassembled PDU]
2958...	793.867018	8.241.88.124	192.168.178.51	TCP	1506	https(443) → 62762	[ACK]	Seq=83670238	Ack=65523	Win=65088	Len=1452	[TCP segment of a reassembled PDU]
2958...	793.867038	192.168.178.51	8.241.88.124	TCP	54	62762 → https(443)	[ACK]	Seq=65523	Ack=83671690	Win=65340	Len=0	
2958...	793.867370	8.241.88.124	192.168.178.51	TCP	1506	https(443) → 62762	[ACK]	Seq=83671690	Ack=65523	Win=65088	Len=1452	[TCP segment of a reassembled PDU]
2958...	793.867373	8.241.88.124	192.168.178.51	TCP	1506	https(443) → 62762	[ACK]	Seq=83673142	Ack=65523	Win=65088	Len=1452	[TCP segment of a reassembled PDU]
2958...	793.867374	8.241.88.124	192.168.178.51	TCP	1506	https(443) → 62762	[ACK]	Seq=83674594	Ack=65523	Win=65088	Len=1452	[TCP segment of a reassembled PDU]

Die TCP-Window Größe ändert sich mit der Zeit mehrfach und wechselt zwischen 65088, 65535 und 63640 hin und her.

ip.addr == 8.241.88.124 && tcp.analysis.window_update												
Paketliste Schmal & breit Groß- / Kleinschreibung beachten Anzeigefilter												
	Time	Source	Destination	Protocol	Length	Info						
33272	85.401546	192.168.178.51	8.241.88.124	TCP	54	[TCP Window Update] 62762 → https(443) [ACK] Seq=13879 Ack=15460958 Win=65340 Len=0						
1389...	297.095700	192.168.178.51	8.241.88.124	TCP	54	[TCP Window Update] 62761 → https(443) [ACK] Seq=44882 Ack=58032841 Win=65340 Len=0						
1567...	329.335955	192.168.178.51	8.241.88.124	TCP	54	[TCP Window Update] 62761 → https(443) [ACK] Seq=53610 Ack=73374394 Win=65340 Len=0						
1569...	329.424716	192.168.178.51	8.241.88.124	TCP	54	[TCP Window Update] 62761 → https(443) [ACK] Seq=53610 Ack=73589290 Win=65340 Len=0						
1633...	338.904043	192.168.178.51	8.241.88.124	TCP	54	[TCP Window Update] 62761 → https(443) [ACK] Seq=56802 Ack=78499878 Win=65340 Len=0						
1738...	360.277756	192.168.178.51	8.241.88.124	TCP	54	[TCP Window Update] 62761 → https(443) [ACK] Seq=62833 Ack=80701073 Win=65340 Len=0						
2668...	742.721101	192.168.178.51	8.241.88.124	TCP	54	[TCP Window Update] 62762 → https(443) [ACK] Seq=51340 Ack=59269052 Win=65340 Len=0						
3324...	853.945111	192.168.178.51	8.241.88.124	TCP	54	[TCP Window Update] 62762 → https(443) [ACK] Seq=82322 Ack=112724221 Win=65340 Len=0						

Mit einem weiteren Filter lassen sich die Pakete finden, bei denen das TCP-Window geupdated wurde.

# Aufgabe 11

## Ablauf einer TELNET-Verbindung zur IP-Adresse 141.62.66.207

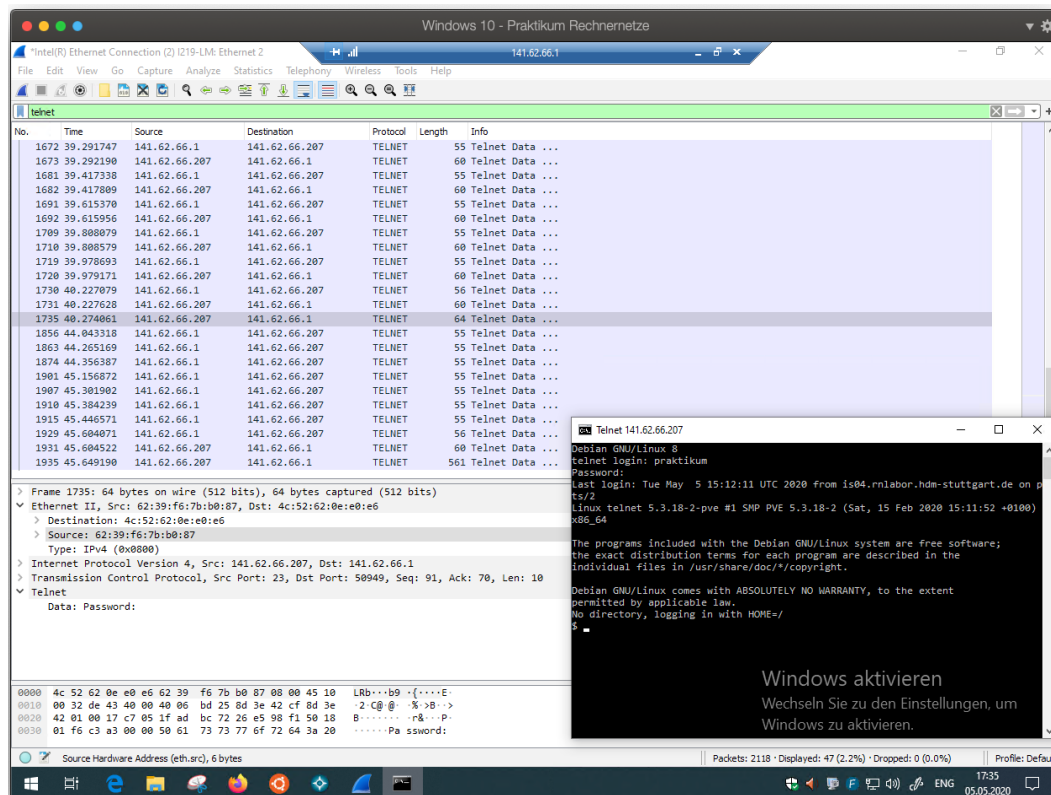
Wir haben folgenden Ablauf befolgt, um TELNET Traffic aufzuzeichnen:

1. Wireshark Capture starten
2. Prompt geöffnet
3. telnet 141.62.66.207
4. login: praktikum, password: versuch eintippen
5. Prompt geschlossen
6. Wireshark Capture gestoppt
7. Nach telnet suchen

## Können Sie Passwörter im Wireshark-Trace identifizieren?

Mit Wireshark kann man den gesamten Traffic bei TELNET mitlesen, da der Traffic nicht verschlüsselt übertragen wird.

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



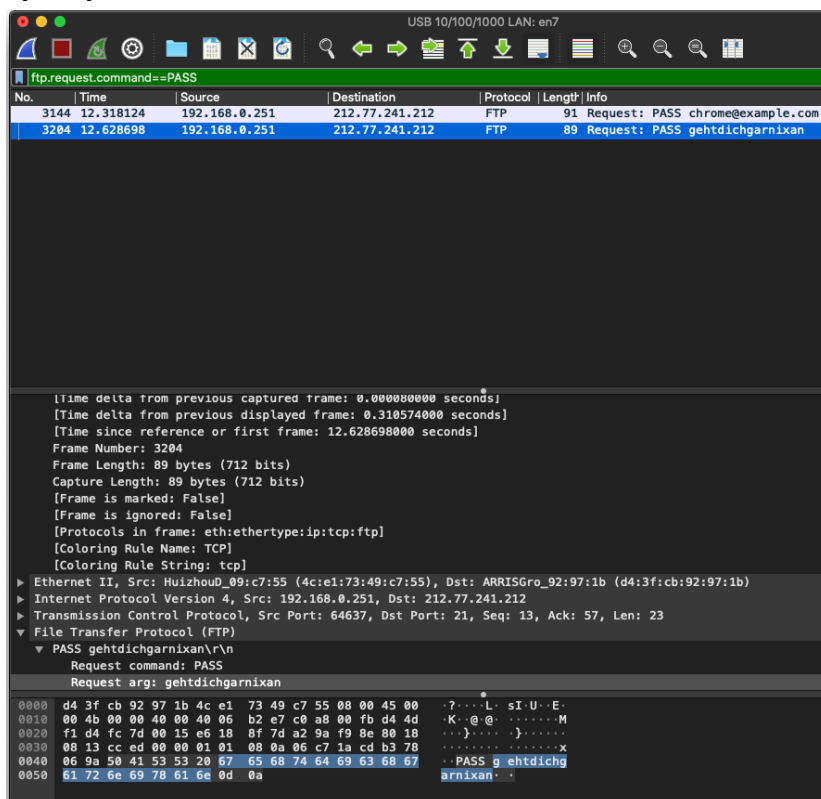
Ab der markierten Zeile, wird jeder Buchstabe des Passworts übertragen. Eine SSH Verbindung dagegen ist sicher ;)

Bernd Maier (bm075), Yannick Möller (ym018), Rebecca Mombrei (rm048), Michael Vanhee (mv068)

## Aufgabe 12

Entwickeln, testen und dokumentieren Sie Wireshark-Filter zur Lösung folgender Aufgaben. Ihr Filter soll folgende Anforderungen erfüllen

- nur IP-Pakete, deren TTL größer ist als ein von Ihnen sinnvoll gewählter Referenzwert  
**IP > 64** (der maximale Wert beträgt 255)
- nur IP-Pakete, die fragmentiert sind  
**ip.fragment**
- beim login-Versuch auf ftp.bellevue.de mit von Ihnen wählbaren Account-Daten nur Rahmen herausfiltern, die das gewählte Passwort im Ethernet-Datenfeld enthalten  
**ftp.request.command==PASS** filtert nur FTP-Rahmen, welche das Passwort beinhalten.



- nur den Port 80-Verkehr zu Ihrer IP-Adresse (ankommend und abgehend)  
**ip.addr == 192.168.178.51 && tcp.port == 80**
- nur Pakete mit einer IP-Multicast-Adresse  
**ip.dst == 224.0.0.0/4**