

**Aufgabe 2 und Aufgabe 3.2:**

Zusammen mit einem Kommilitonen haben wir einen UDPRing aus zwei Personen hergestellt.

```
Token: seq=9, #members=2 (136.199.25.167, 59402) (136.199.106.246, 58277)
Token: seq=11, #members=2 (136.199.25.167, 59402) (136.199.106.246, 58277)
Token: seq=13, #members=2 (136.199.25.167, 59402) (136.199.106.246, 58277)
Token: seq=15, #members=2 (136.199.25.167, 59402) (136.199.106.246, 58277)
Token: seq=17, #members=2 (136.199.25.167, 59402) (136.199.106.246, 58277)
```

Herausforderung dabei war es, die richtigen Ports freizuschalten und ein Forwarding im Router einzurichten. Ohne dies hat die Firewall die UDP-Pakete abgefangen und der Ring ist nicht funktionsfähig gewesen.

In Wireshark konnten wir anhand obiger Abbildung die Pakete identifizieren:

9730	1109.694418	136.199.106.246	136.199.25.167	UDP	140	58277 → 59402	Len=98
9731	1109.895072	136.199.106.246	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1	
9732	1110.891413	136.199.25.167	136.199.106.246	UDP	140	59402 → 58277	Len=98

**Aufgabe 3.1:**

Aufgabe ist es, den Umgang mit **WireShark** zu lernen.

Als Einstieg habe ich dafür einen Ring mit 4 Instanzen (lokal) aufgebaut:

```
Token: seq=482, #members=4 (192.168.178.175, 54605) (192.168.178.175, 51000) (192.168.178.175, 57179) (192.168.178.175, 57180)
```

Abb. 1.1

Dieser läuft fehlerfrei und jede Sekunde wird ein Token von einer Instanz zu einer anderen Instanz geschickt. Dies lässt sich in **WireShark** an folgender Stelle erkennen:

udp							
No.	Time	Source	Destination	Protocol	Length	Info	
1	0.000000	192.168.178.175	192.168.178.175	UDP	209	51000 → 57179	Len=177
6	1.001874	192.168.178.175	192.168.178.175	UDP	209	57179 → 57180	Len=177
9	2.003752	192.168.178.175	192.168.178.175	UDP	209	57180 → 54605	Len=177
12	3.006992	192.168.178.175	192.168.178.175	UDP	209	54605 → 51000	Len=177
21	4.009583	192.168.178.175	192.168.178.175	UDP	209	51000 → 57179	Len=177
22	5.015737	192.168.178.175	192.168.178.175	UDP	209	57179 → 57180	Len=177
23	6.020881	192.168.178.175	192.168.178.175	UDP	209	57180 → 54605	Len=177

Abb. 1.2

Ab dem in Rot unterlegten Eintrag sind die folgenden drei Einträge zugehörig zum UDP-Ring. Erkennen lässt sich dies an den „Source“ und „Destination“ Adressen. Zu jedem Eintrag gibt es die „Info“, welcher „Source Port“ und welcher „Destination Port“ verwendet wird.

Werden die Einträge Aus Abb. 1.1 und Abb. 2.1 verglichen, so fällt auf, dass hier IP-Adressen und Portnummern exakt übereinstimmen.

In Abb. 1.1 schickt 192.168.178.175 über Port 54605 einen Token zu gleicher IP mit Port 51000:

```
12 3.006992 192.168.178.175 192.168.178.175 UDP 209 54605 → 51000 Len=177
```

Von 51000 geht es dann nach 57179 usw. :

21 4.009583 192.168.178.175 192.168.178.175 UDP 209 51000 → 57179 Len=177

Bis der Ring geschlossen wird, indem Port 57180 an 54605 sendet:

23 6.020881 192.168.178.175 192.168.178.175 UDP 209 57180 → 54605 Len=177

Damit die Suche nach den relevanten Einträgen schneller durchgeführt werden kann, wurde in *Abb. 1.2* bereits nach „udp“ gefiltert. Eine weitere Möglichkeit besteht darin, speziell nach einem Eintrag zu suchen, im folgenden Beispiel nach der bekannten IP-Adresse:

udp						
Paketliste Schmal & breit Groß- / Kleinschreibung beachten Anzeigefilter ip.addr == 192.168.178.175						
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.178.175	192.168.178.175	UDP	209	51000 → 57179 Len=177
6	1.001874	192.168.178.175	192.168.178.175	UDP	209	57179 → 57180 Len=177

Eine weitere Spezifikation wäre die Suche nach den Ports. Aus *Abb. 1.1* wissen wir, welche Ports beteiligt sind. Wenn bspw. Einträge von Port 51000 nach 57179 gesucht werden, so kann die folgende Syntax verwendet werden:

udp						
Paketliste Schmal & breit Groß- / Kleinschreibung beachten Anzeigefilter udp.srcport == 51000 && udp.dstport == 57179						
No.	Time	Source	Destination	Protocol	Length	Info
28	6.753157	fe80::5597:80f0:...	ff02::fb	MDNS	97	Standard query 0x0000 PTR _spotify-connect._tcp.lo
29	6.753328	fe80::d63d:dcd:7...	ff02::fb	MDNS	97	Standard query 0x0000 PTR _spotify-connect._tcp.lo
30	7.033958	192.168.178.175	192.168.178.175	UDP	209	54605 → 51000 Len=177
31	8.049289	192.168.178.175	192.168.178.175	UDP	209	51000 → 57179 Len=177
32	8.972831	192.168.56.1	239.255.255.250	SSDP	157	M-SEARCH * HTTP/1.1
33	8.972951	192.168.69.24	239.255.255.250	SSDP	157	M-SEARCH * HTTP/1.1
34	8.973056	192.168.178.175	239.255.255.250	SSDP	157	M-SEARCH * HTTP/1.1
35	9.050878	192.168.178.175	192.168.178.175	UDP	209	57179 → 57180 Len=177
38	10.055003	192.168.178.175	192.168.178.175	UDP	209	57180 → 54605 Len=177
39	11.067802	192.168.178.175	192.168.178.175	UDP	209	54605 → 51000 Len=177
40	12.079737	192.168.178.175	192.168.178.175	UDP	209	51000 → 57179 Len=177
53	13.093894	192.168.178.175	192.168.178.175	UDP	209	57179 → 57180 Len=177
54	13.807759	192.168.56.1	192.168.56.255	UDP	76	57621 → 57621 Len=44
55	13.807861	192.168.69.24	192.168.69.255	UDP	76	57621 → 57621 Len=44
56	13.807919	192.168.178.175	192.168.178.255	UDP	76	57621 → 57621 Len=44
57	14.109137	192.168.178.175	192.168.178.175	UDP	209	57180 → 54605 Len=177
58	15.113854	192.168.178.175	192.168.178.175	UDP	209	54605 → 51000 Len=177
61	16.126452	192.168.178.175	192.168.178.175	UDP	209	51000 → 57179 Len=177

Die gefundenen Einträge werden dann schwarz hinterlegt.