

Rechnernetze, Übungsblatt 1, Sommer 2024

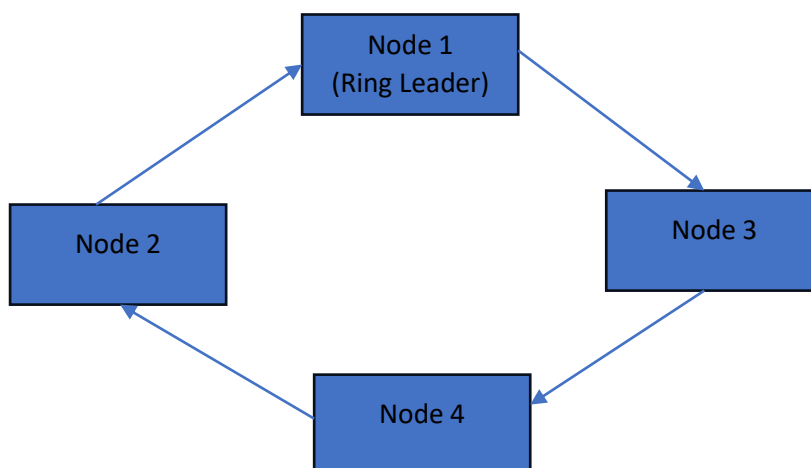
Zu Aufgabe 2

Zum Aufbau eines möglichst großen Rings unternahm ich mit meinen Kommilitonen verschiedene Versuche vor Ort an der Uni. Dies gestaltete sich allerdings als schwierig, da ein Verbindungsaufbau per UDP im Uninetzwerk nicht (bzw. nur in einem kurzen Ausnahmefall) funktionierte. Die Firewall zumindest zweier Rechner konnte ich als Fehlerquelle ausschließen, da sie in meinem Heimnetzwerk bereits einen Ring gebildet hatten. Auch das Einstellen des Uninetzwerks als privates Netz zwecks Sichtbarkeit der Rechner schuf keine Abhilfe. Ebenso wenig von Erfolg gekrönt waren mehrere Versuche, einen Ring in einem Hotspot aufzubauen. Als Hotspot-Geräte fungierten hierbei verschiedene Smartphones und Laptops.

Die Programmausgaben node1.txt bis node4.txt dokumentieren den Bau eines Rings in meinem Heimnetzwerk bestehend aus vier Knoten verteilt auf zwei Rechnern.

- node1.txt dokumentiert die Ausgabe des Ring Leaders (IP-Adresse: 192.168.2.108, Port: 50126)
- Mit dem Ring Leader wurde nun ein zweiter Knoten auf demselben Rechner verbunden (IP-Adresse: 192.168.2.108, Port: 59811), dessen Protokollausgabe in node2.txt dokumentiert ist.
- Anschließend wurde Node 3 (IP-Adresse: 192.168.2.198, Port: 50664) zwischen dem Ring Leader und Node 2 eingefügt.
- Schlussendlich wurde Node 4 (IP-Adresse: 192.168.2.198, Port: 60632) zwischen Node 3 und Node 2 eingefügt.

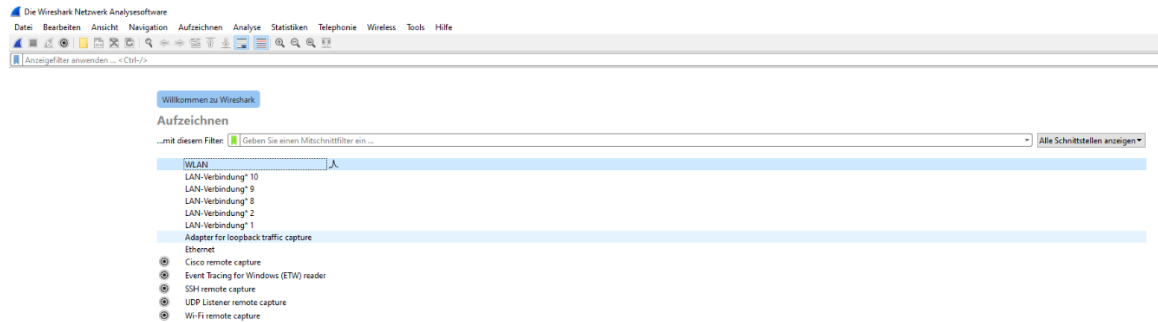
Wie sich anhand der Sequenz der Tokens nachvollziehen lässt, sah der Ring schlussendlich also folgendermaßen aus:



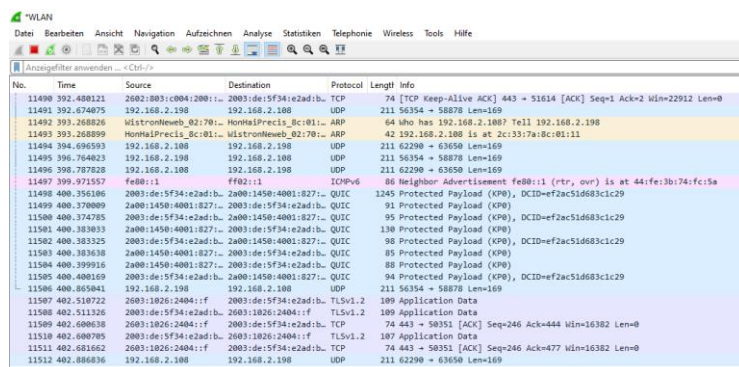
Zu Aufgabe 3

Wie in Aufgabe 2 nutzte ich auch für diese Aufgabe einen Ring bestehend aus vier Knoten verteilt auf zwei Rechner mit den IP-Adressen 192.168.2.108 und 192.168.2.198. (Da es sich um einen neuen Ring handelt, weichen die Ports allerdings von Aufgabe 2 ab.)

Nach dem Starten von Wireshark wählte ich zunächst die Netzwerkverbindung „WLAN“, um die Pakete des Token Rings mitzuschneiden. (Im Falle eines lokalen Token-Rings auf einem einzigen Rechner hätte ich „Adapter for loopback traffic capture“ wählen müssen.)



Da nun alle Pakete, die meine WLAN-Verbindung gesendet wurden, angezeigt wurden, war es ziemlich unübersichtlich und unkomfortabel, die Pakete des Token-Rings herauszusuchen.



No.	Time	Source	Destination	Protocol	Length	Info
11490	392.480121	2002:803:c004:200::1	2003:de:5f34:e2ad:b...	TCP	74	[TCP Keep-Alive ACK] 443 → 51614 [ACK] Seq=1 Ack=2 Win=22912 Len=0
11491	392.674075	192.168.2.198	192.168.2.108	UDP	211	56354 → 58878 Len=169
11492	393.268826	WlstronNeweb_02:79::	MonNaIPrecis_8c:01::	ARP	64	Who has 192.168.2.108? Tell 192.168.2.198
11493	393.268899	MonNaIPrecis_8c:01::	WlstronNeweb_02:79::	ARP	42	192.168.2.108 is at 2c:33:7a:8c:01:11
11494	394.696593	192.168.2.108	192.168.2.198	UDP	211	62290 → 63650 Len=169
11495	396.764023	192.168.2.198	192.168.2.108	UDP	211	56354 → 58878 Len=169
11496	396.767828	192.168.2.108	192.168.2.198	UDP	211	62290 → 63650 Len=169
11497	399.971557	fe80::1	ff02::1	ICMPv6	86	Neighbor Advertisement fe80::1 (rtr, ovr) is at 44:fe:3b:74:fc:5a
11498	400.356186	2003:de:5f34:e2ad:b...	2a00:1450:4001:827::	QUIC	1245	Protected Payload (XPR), DCID=ef2ac51d683c1c29
11499	400.370009	2a00:1450:4001:827::	2003:de:5f34:e2ad:b...	QUIC	91	Protected Payload (XPR)
11500	400.374785	2003:de:5f34:e2ad:b...	2a00:1450:4001:827::	QUIC	95	Protected Payload (XPR), DCID=ef2ac51d683c1c29
11501	400.383033	2a00:1450:4001:827::	2003:de:5f34:e2ad:b...	QUIC	130	Protected Payload (XPR)
11502	400.383325	2003:de:5f34:e2ad:b...	2a00:1450:4001:827::	QUIC	98	Protected Payload (XPR), DCID=ef2ac51d683c1c29
11503	400.383638	2a00:1450:4001:827::	2003:de:5f34:e2ad:b...	QUIC	85	Protected Payload (XPR)
11504	400.399916	2a00:1450:4001:827::	2003:de:5f34:e2ad:b...	QUIC	88	Protected Payload (XPR)
11505	400.400169	2003:de:5f34:e2ad:b...	2a00:1450:4001:827::	QUIC	94	Protected Payload (XPR), DCID=ef2ac51d683c1c29
11506	400.865041	192.168.2.198	192.168.2.108	UDP	211	56354 → 58878 Len=169
11507	402.518722	2003:1026:2404::f	2003:de:5f34:e2ad:b...	TLSv1.2	109	Application Data
11508	402.511326	2003:de:5f34:e2ad:b...	2003:1026:2404::f	TLSv1.2	189	Application Data
11509	402.600638	2003:1026:2404::f	2003:de:5f34:e2ad:b...	TCP	74	443 → 58351 [ACK] Seq=246 Ack=444 Win=16382 Len=0
11510	402.600705	2003:de:5f34:e2ad:b...	2003:1026:2404::f	TLSv1.2	187	Application Data
11511	402.681662	2003:1026:2404::f	2003:de:5f34:e2ad:b...	TCP	74	443 → 58351 [ACK] Seq=246 Ack=477 Win=16382 Len=0
11512	402.886836	192.168.2.198	192.168.2.108	UDP	211	62290 → 63650 Len=169

Daher verwendete ich einen Displayfilter, sodass nur noch UDP-Pakete angezeigt wurden, die von der IP-Adresse 192.168.2.108 an 192.168.2.198 oder umgekehrt gesendet wurden. Hierzu gab ich in die Filtermaske folgendes ein: *udp and (ip.src==192.168.2.108 or ip.src==192.168.2.198) and (ip.dst==192.168.2.108 or ip.dst==192.168.2.198)*

Somit wurden nur noch Pakete des Token-Rings angezeigt. Beim Ansehen der Pakete bzw. deren Hex Dumps ließ sich nachvollziehen, dass jeweils der Header des UDP-Datagramms (auf der obigen Abbildung grün markiert) ab dem 35. Byte begann und 64 bit also 8 Bytes umfasste. Danach folgten die ASCII-codierten Nutzdaten. Bei den Daten vor dem Header des Datagramms handelte es sich um den IPv4-Header des jeweiligen Pakets.