Homework 2

Team 23

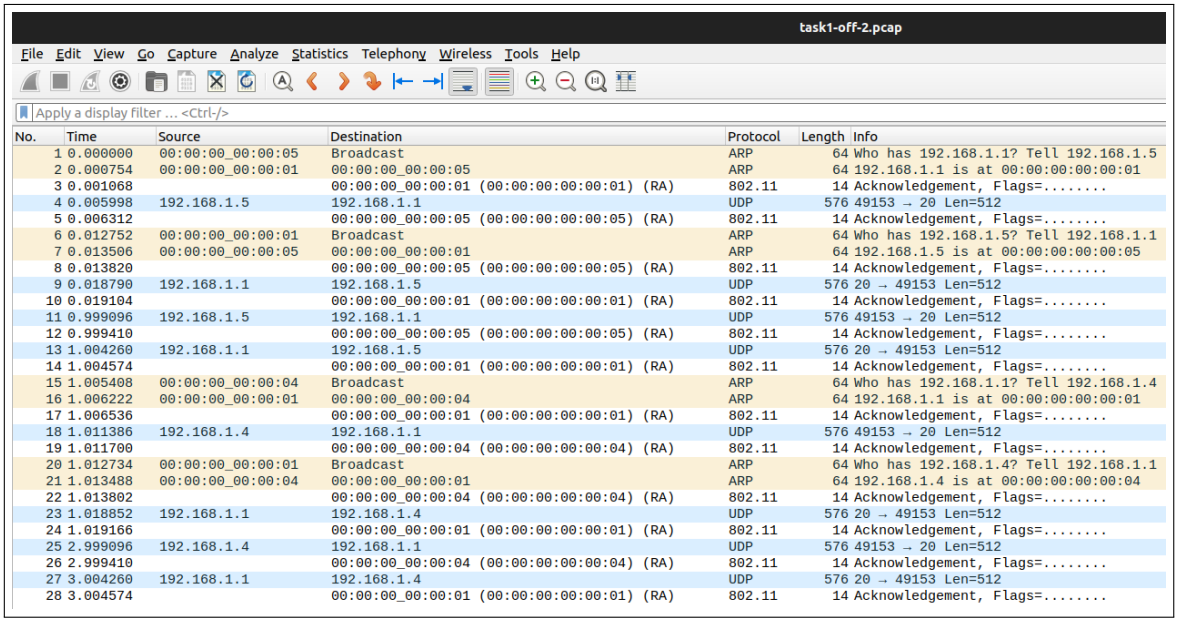
Concari, Laura 1890490
Di Chiara, Alessandro 1938462
Lamacchia, Anna Lucia 1933472
Longiarù, Gianmarco 1965768

December 16, 2022

Task 1

1. Tutti i frame ricevono l'acknowledgement? Spiegare perché.

Tutti i frame inviati ricevono l'acknowledgement, ad eccezione delle richieste ARP. Tali richieste ricevono l'ACK solo quando il mittente riceve il frame ARP reply, ma non alla ricezione da parte del destinatario della ARP request.
La richiesta ARP viene effettuata dal sender per individuare l'indirizzo MAC del nodo receiver di cui conosce unicamente l'indirizzo IP. Quando il receiver riceve un frame Arp reply contenente l'indirizzo MAC cercato, il sender invia un ACK per confermare la corretta ricezione dell'informazione.
In Wireshark si nota come ad ogni richiesta ARP o invio di un pacchetto UDP da parte di un nodo corrisponda un ACK inviato subito dopo la corretta ricezione del dato. (Fig. 1).
In NetAnim si osserva come ogni richiesta ARP o invio di un pacchetto UDP abbia come mittenti e destinatari tutti i nodi della rete, essendo un'infrastruttura ad-hoc, i quali rispondono tutti con un ACK (Fig. 2).



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	00:00:00_00:00:05	Broadcast	ARP	64	Who has 192.168.1.1? Tell 192.168.1.5
2	0.000754	00:00:00_00:00:01	00:00:00_00:00:05	ARP	64	192.168.1.1 is at 00:00:00:00:00:01
3	0.001068		00:00:00_00:00:01 (00:00:00:00:00:01) (RA)	802.11	14	Acknowledgement, Flags=........
4	0.005998	192.168.1.5	192.168.1.1	UDP	576	49153 → 20 Len=512
5	0.006312		00:00:00_00:00:05 (00:00:00:00:00:05) (RA)	802.11	14	Acknowledgement, Flags=........
6	0.012752	00:00:00_00:00:01	Broadcast	ARP	64	Who has 192.168.1.5? Tell 192.168.1.1
7	0.013506	00:00:00_00:00:05	00:00:00_00:00:01	ARP	64	192.168.1.5 is at 00:00:00:00:00:05
8	0.013820		00:00:00_00:00:05 (00:00:00:00:00:05) (RA)	802.11	14	Acknowledgement, Flags=........
9	0.018790	192.168.1.1	192.168.1.5	UDP	576	20 → 49153 Len=512
10	0.019104		00:00:00_00:00:01 (00:00:00:00:00:01) (RA)	802.11	14	Acknowledgement, Flags=........
11	0.999096	192.168.1.5	192.168.1.1	UDP	576	49153 → 20 Len=512
12	0.999410		00:00:00_00:00:05 (00:00:00:00:00:05) (RA)	802.11	14	Acknowledgement, Flags=........
13	1.004260	192.168.1.1	192.168.1.5	UDP	576	20 → 49153 Len=512
14	1.004574		00:00:00_00:00:01 (00:00:00:00:00:01) (RA)	802.11	14	Acknowledgement, Flags=........
15	1.005408	00:00:00_00:00:04	Broadcast	ARP	64	Who has 192.168.1.1? Tell 192.168.1.4
16	1.006222	00:00:00_00:00:01	00:00:00_00:00:04	ARP	64	192.168.1.1 is at 00:00:00:00:00:01
17	1.006536		00:00:00_00:00:01 (00:00:00:00:00:01) (RA)	802.11	14	Acknowledgement, Flags=........
18	1.011386	192.168.1.4	192.168.1.1	UDP	576	49153 → 20 Len=512
19	1.011700		00:00:00_00:00:04 (00:00:00:00:00:04) (RA)	802.11	14	Acknowledgement, Flags=........
20	1.012734	00:00:00_00:00:01	Broadcast	ARP	64	Who has 192.168.1.4? Tell 192.168.1.1
21	1.013488	00:00:00_00:00:04	00:00:00_00:00:01	ARP	64	192.168.1.4 is at 00:00:00:00:00:04
22	1.013802		00:00:00_00:00:04 (00:00:00:00:00:04) (RA)	802.11	14	Acknowledgement, Flags=........
23	1.018852	192.168.1.1	192.168.1.4	UDP	576	20 → 49153 Len=512
24	1.019166		00:00:00_00:00:01 (00:00:00:00:00:01) (RA)	802.11	14	Acknowledgement, Flags=........
25	2.999096	192.168.1.4	192.168.1.1	UDP	576	49153 → 20 Len=512
26	2.999410		00:00:00_00:00:04 (00:00:00:00:00:04) (RA)	802.11	14	Acknowledgement, Flags=........
27	3.004260	192.168.1.1	192.168.1.4	UDP	576	20 → 49153 Len=512
28	3.004574		00:00:00_00:00:01 (00:00:00:00:00:01) (RA)	802.11	14	Acknowledgement, Flags=........

Figure 1: File task1-off-2.pcap - Analisi pacchetti Wireshark

	From Id	To Id	Tx	Meta
1	4	3	1.00505	Arp request SMac: 00:00:00:00:00:05 DMac: ff:ff:ff:ff:ff:ff SrcIp : 192.168.1.5 DstIp : 192.168.1.1
2	4	0	1.00505	Arp request SMac: 00:00:00:00:00:05 DMac: ff:ff:ff:ff:ff:ff SrcIp : 192.168.1.5 DstIp : 192.168.1.1
3	4	1	1.00505	Arp request SMac: 00:00:00:00:00:05 DMac: ff:ff:ff:ff:ff:ff SrcIp : 192.168.1.5 DstIp : 192.168.1.1
4	4	2	1.00505	Arp request SMac: 00:00:00:00:00:05 DMac: ff:ff:ff:ff:ff:ff SrcIp : 192.168.1.5 DstIp : 192.168.1.1
5	0	1	1.0058	Arp reply SMac: 00:00:00:00:00:01 DMac: 00:00:00:00:00:05 SrcIp : 192.168.1.1 DstIp : 192.168.1.5
6	0	3	1.0058	Arp reply SMac: 00:00:00:00:00:01 DMac: 00:00:00:00:00:05 SrcIp : 192.168.1.1 DstIp : 192.168.1.5
7	0	4	1.0058	Arp reply SMac: 00:00:00:00:00:01 DMac: 00:00:00:00:00:05 SrcIp : 192.168.1.1 DstIp : 192.168.1.5
8	0	2	1.0058	Arp reply SMac: 00:00:00:00:00:01 DMac: 00:00:00:00:00:05 SrcIp : 192.168.1.1 DstIp : 192.168.1.5
9	4	3	1.00652	Wifi CTL_ACK RA:00:00:00:00:00:01
10	4	0	1.00652	Wifi CTL_ACK RA:00:00:00:00:00:01
11	4	1	1.00652	Wifi CTL_ACK RA:00:00:00:00:00:01
12	4	2	1.00652	Wifi CTL_ACK RA:00:00:00:00:00:01
13	4	3	1.00695	UDP 49153 > 20
14	4	0	1.00695	UDP 49153 > 20
15	4	1	1.00695	UDP 49153 > 20
16	4	2	1.00695	UDP 49153 > 20
17	0	1	1.01176	Wifi CTL_ACK RA:00:00:00:00:00:05
18	0	3	1.01176	Wifi CTL_ACK RA:00:00:00:00:00:05
19	0	4	1.01176	Wifi CTL_ACK RA:00:00:00:00:00:05
20	0	2	1.01176	Wifi CTL_ACK RA:00:00:00:00:00:05
21	0	1	1.0178	Arp request SMac: 00:00:00:00:00:01 DMac: ff:ff:ff:ff:ff:ff SrcIp : 192.168.1.1 DstIp : 192.168.1.5
22	0	3	1.0178	Arp request SMac: 00:00:00:00:00:01 DMac: ff:ff:ff:ff:ff:ff SrcIp : 192.168.1.1 DstIp : 192.168.1.5
23	0	4	1.0178	Arp request SMac: 00:00:00:00:00:01 DMac: ff:ff:ff:ff:ff:ff SrcIp : 192.168.1.1 DstIp : 192.168.1.5
24	0	2	1.0178	Arp request SMac: 00:00:00:00:00:01 DMac: ff:ff:ff:ff:ff:ff SrcIp : 192.168.1.1 DstIp : 192.168.1.5
25	4	3	1.01856	Arp reply SMac: 00:00:00:00:00:05 DMac: 00:00:00:00:00:01 SrcIp : 192.168.1.5 DstIp : 192.168.1.1
26	4	0	1.01856	Arp reply SMac: 00:00:00:00:00:05 DMac: 00:00:00:00:00:01 SrcIp : 192.168.1.5 DstIp : 192.168.1.1
27	4	1	1.01856	Arp reply SMac: 00:00:00:00:00:05 DMac: 00:00:00:00:00:01 SrcIp : 192.168.1.5 DstIp : 192.168.1.1
28	4	2	1.01856	Arp reply SMac: 00:00:00:00:00:05 DMac: 00:00:00:00:00:01 SrcIp : 192.168.1.5 DstIp : 192.168.1.1
29	0	1	1.01927	Wifi CTL_ACK RA:00:00:00:00:00:05
30	0	3	1.01927	Wifi CTL_ACK RA:00:00:00:00:00:05
31	0	4	1.01927	Wifi CTL_ACK RA:00:00:00:00:00:05
32	0	2	1.01927	Wifi CTL_ACK RA:00:00:00:00:00:05

	From Id	To Id	Tx	Meta
33	0	1	1.01974	UDP 20 > 49153
34	0	3	1.01974	UDP 20 > 49153
35	0	4	1.01974	UDP 20 > 49153
36	0	2	1.01974	UDP 20 > 49153
37	4	3	1.02455	Wifi CTL_ACK RA:00:00:00:00:00:01
38	4	0	1.02455	Wifi CTL_ACK RA:00:00:00:00:00:01
39	4	1	1.02455	Wifi CTL_ACK RA:00:00:00:00:00:01
40	4	2	1.02455	Wifi CTL_ACK RA:00:00:00:00:00:01
41	4	3	2.00005	UDP 49153 > 20
42	4	0	2.00005	UDP 49153 > 20
43	4	1	2.00005	UDP 49153 > 20
44	4	2	2.00005	UDP 49153 > 20
45	0	1	2.00486	Wifi CTL_ACK RA:00:00:00:00:00:05
46	0	3	2.00486	Wifi CTL_ACK RA:00:00:00:00:00:05
47	0	2	2.00486	Wifi CTL_ACK RA:00:00:00:00:00:05
48	0	4	2.00486	Wifi CTL_ACK RA:00:00:00:00:00:05
49	0	1	2.00521	UDP 20 > 49153
50	0	3	2.00521	UDP 20 > 49153
51	0	2	2.00521	UDP 20 > 49153
52	0	4	2.00521	UDP 20 > 49153
53	4	3	2.01002	Wifi CTL_ACK RA:00:00:00:00:00:01
54	4	0	2.01002	Wifi CTL_ACK RA:00:00:00:00:00:01
55	4	1	2.01002	Wifi CTL_ACK RA:00:00:00:00:00:01
56	4	2	2.01002	Wifi CTL_ACK RA:00:00:00:00:00:01
57	3	4	2.01046	Arp request SMac: 00:00:00:00:00:04 DMac: ff:ff:ff:ff:ff:ff SrcIp : 192.168.1.4 DstIp : 192.168.1.1
58	3	0	2.01046	Arp request SMac: 00:00:00:00:00:04 DMac: ff:ff:ff:ff:ff:ff SrcIp : 192.168.1.4 DstIp : 192.168.1.1
59	3	1	2.01046	Arp request SMac: 00:00:00:00:00:04 DMac: ff:ff:ff:ff:ff:ff SrcIp : 192.168.1.4 DstIp : 192.168.1.1
60	3	2	2.01046	Arp request SMac: 00:00:00:00:00:04 DMac: ff:ff:ff:ff:ff:ff SrcIp : 192.168.1.4 DstIp : 192.168.1.1
61	0	1	2.01127	Arp reply SMac: 00:00:00:00:00:01 DMac: 00:00:00:00:00:04 SrcIp : 192.168.1.1 DstIp : 192.168.1.4
62	0	3	2.01127	Arp reply SMac: 00:00:00:00:00:01 DMac: 00:00:00:00:00:04 SrcIp : 192.168.1.1 DstIp : 192.168.1.4
63	0	2	2.01127	Arp reply SMac: 00:00:00:00:00:01 DMac: 00:00:00:00:00:04 SrcIp : 192.168.1.1 DstIp : 192.168.1.4
64	0	4	2.01127	Arp reply SMac: 00:00:00:00:00:01 DMac: 00:00:00:00:00:04 SrcIp : 192.168.1.1 DstIp : 192.168.1.4
65	3	4	2.01199	Wifi CTL_ACK RA:00:00:00:00:00:01
66	3	0	2.01199	Wifi CTL_ACK RA:00:00:00:00:00:01
67	3	1	2.01199	Wifi CTL_ACK RA:00:00:00:00:00:01
68	3	2	2.01199	Wifi CTL_ACK RA:00:00:00:00:00:01

	From Id	To Id	Tx	Meta
69	3	4	2.01234	UDP 49153 > 20
70	3	0	2.01234	UDP 49153 > 20
71	3	1	2.01234	UDP 49153 > 20
72	3	2	2.01234	UDP 49153 > 20
73	0	1	2.01715	Wifi CTL_ACK RA:00:00:00:00:00:04
74	0	3	2.01715	Wifi CTL_ACK RA:00:00:00:00:00:04
75	0	2	2.01715	Wifi CTL_ACK RA:00:00:00:00:00:04
76	0	4	2.01715	Wifi CTL_ACK RA:00:00:00:00:00:04
77	0	1	2.01778	Arp request SMac: 00:00:00:00:00:01 DMac: ff:ff:ff:ff:ff:ff SrcIp : 192.168.1.1 DstIp : 192.168.1.4
78	0	3	2.01778	Arp request SMac: 00:00:00:00:00:01 DMac: ff:ff:ff:ff:ff:ff SrcIp : 192.168.1.1 DstIp : 192.168.1.4
79	0	2	2.01778	Arp request SMac: 00:00:00:00:00:01 DMac: ff:ff:ff:ff:ff:ff SrcIp : 192.168.1.1 DstIp : 192.168.1.4
80	0	4	2.01778	Arp request SMac: 00:00:00:00:00:01 DMac: ff:ff:ff:ff:ff:ff SrcIp : 192.168.1.1 DstIp : 192.168.1.4
81	3	4	2.01854	Arp reply SMac: 00:00:00:00:00:04 DMac: 00:00:00:00:00:01 SrcIp : 192.168.1.4 DstIp : 192.168.1.1
82	3	0	2.01854	Arp reply SMac: 00:00:00:00:00:04 DMac: 00:00:00:00:00:01 SrcIp : 192.168.1.4 DstIp : 192.168.1.1
83	3	1	2.01854	Arp reply SMac: 00:00:00:00:00:04 DMac: 00:00:00:00:00:01 SrcIp : 192.168.1.4 DstIp : 192.168.1.1
84	3	2	2.01854	Arp reply SMac: 00:00:00:00:00:04 DMac: 00:00:00:00:00:01 SrcIp : 192.168.1.4 DstIp : 192.168.1.1
85	0	1	2.01925	Wifi CTL_ACK RA:00:00:00:00:00:04
86	0	3	2.01925	Wifi CTL_ACK RA:00:00:00:00:00:04
87	0	2	2.01925	Wifi CTL_ACK RA:00:00:00:00:00:04
88	0	4	2.01925	Wifi CTL_ACK RA:00:00:00:00:00:04
89	0	1	2.01981	UDP 20 > 49153
90	0	3	2.01981	UDP 20 > 49153
91	0	2	2.01981	UDP 20 > 49153
92	0	4	2.01981	UDP 20 > 49153
93	3	4	2.02462	Wifi CTL_ACK RA:00:00:00:00:00:01
94	3	0	2.02462	Wifi CTL_ACK RA:00:00:00:00:00:01
95	3	1	2.02462	Wifi CTL_ACK RA:00:00:00:00:00:01
96	3	2	2.02462	Wifi CTL_ACK RA:00:00:00:00:00:01
97	3	4	4.00005	UDP 49153 > 20
98	3	0	4.00005	UDP 49153 > 20
99	3	2	4.00005	UDP 49153 > 20
100	3	1	4.00005	UDP 49153 > 20
101	0	1	4.00486	Wifi CTL_ACK RA:00:00:00:00:00:04
102	0	2	4.00486	Wifi CTL_ACK RA:00:00:00:00:00:04
103	0	3	4.00486	Wifi CTL_ACK RA:00:00:00:00:00:04
104	0	4	4.00486	Wifi CTL_ACK RA:00:00:00:00:00:04
105	0	1	4.00521	UDP 20 > 49153
106	0	2	4.00521	UDP 20 > 49153
107	0	3	4.00521	UDP 20 > 49153
108	0	4	4.00521	UDP 20 > 49153
109	3	4	4.01002	Wifi CTL_ACK RA:00:00:00:00:00:01
110	3	0	4.01002	Wifi CTL_ACK RA:00:00:00:00:00:01
111	3	2	4.01002	Wifi CTL_ACK RA:00:00:00:00:00:01
112	3	1	4.01002	Wifi CTL_ACK RA:00:00:00:00:00:01

Figure 2: File wireless-task1-rts-off.xml - Analisi pacchetti NetAnim

2. Vi sono delle collisioni nella rete? Spiegare perché. Come sei arrivato a questa conclusione?

Non ci sono collisioni, poiché ogni pacchetto inviato riceve sempre un ACK di conferma.
Se così non fosse, seguendo il protocollo CSMA/CA, il nodo mittente farebbe una ritrasmissione del pacchetto dopo un backoff time fino alla ricezione di un ACK, a conferma della corretta ricezione del pacchetto.
Utilizzando il filtro wlan.fc.retry==1 si osserva come non ci siano ritrasmissioni di pacchetti UDP, chiara dimostrazione dell'assenza di collisioni (Fig. 3).

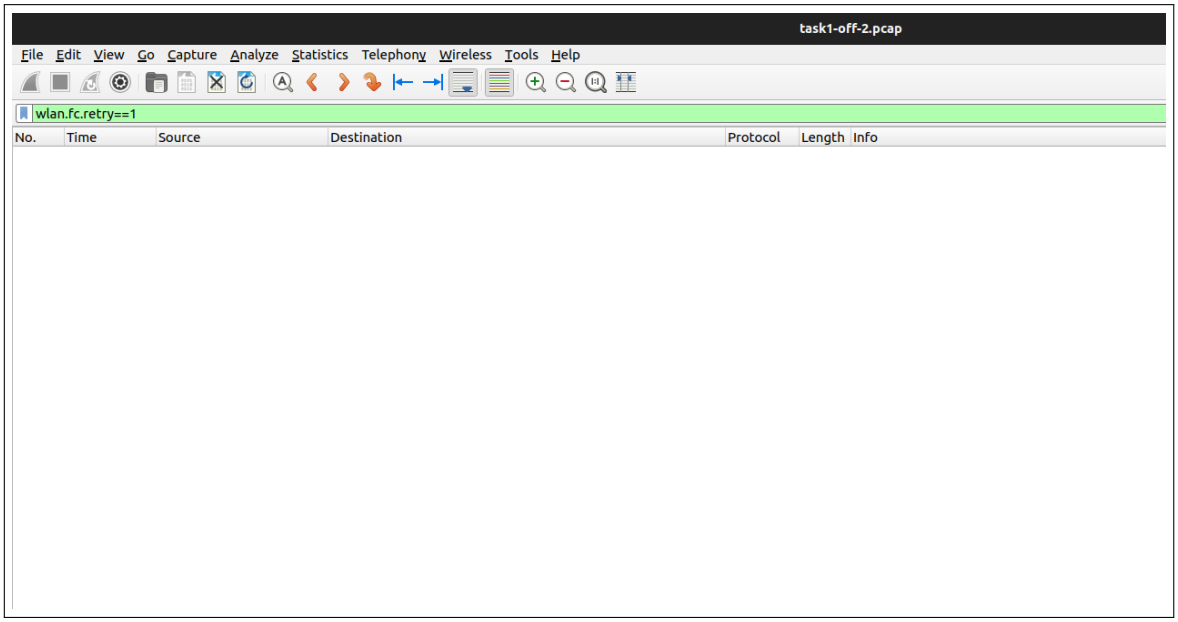


Figure 3: File task1-off-2.pcap - Assenza di collisioni

3. Come si può forzare i nodi ad utilizzare la procedura di handshake RTS/CTS vista in classe? Qual è il ragionamento dietro questa procedura?

La procedura di two-way-handshake di RTS/CTS è attivata quando la dimensione di un pacchetto supera una determinata threshold. Riducendo il valore della threshold al di sotto della packet size configurata, ogni nodo dovrà attenersi alla procedura di scambio determinata da RTS/CTS.
Il valore della threshold è impostato modificando i parametri della configurazione della simulazione ns-3 attraverso le seguenti righe di codice.

```
UintegerValue ctsThreshold = (useRtsCts ? UintegerValue(100) : UintegerValue(2346));
Config::SetDefault("ns3::WifiRemoteStationManager::RtsCtsThreshold", ctsThreshold);
```

Se la variabile useRtsCts ha valore `true` viene forzato l'uso della procedura RTS/CTS per tutti i pacchetti la cui dimensione supera 100 byte, valore di threshold definito nel codice.

4. Forzare l'uso di RTS/CTS nella rete utilizzando il parametro useRtsCts:

Ci sono delle collisioni adesso? Non ci sono collisioni, poiché ogni nodo prima di trasmettere deve essere autorizzato da un clearance il quale assicura che non ci siano altre trasmissioni in corso (Fig. 4).

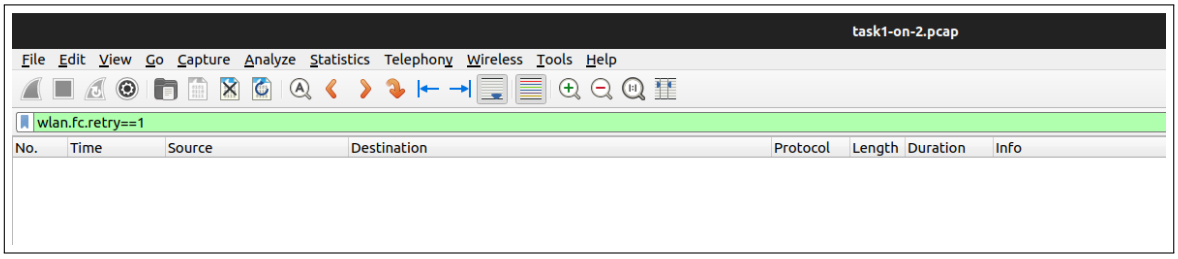


Figure 4: File task1-on-2.pcap - Assenza di collisioni

Quali sono i benefici di RTS/CTS? RTS/CTS permette di evitare collisioni silenziando le comunicazioni di tutti i nodi al di fuori dei due autorizzati alla trasmissione.
Infatti, il nodo mittente invia prima un RTS (Request to send) per verificare che il canale sia libero. Solo quando il destinatario invia come risposta un CTS (Clear to send), il nodo mittente può cominciare a trasmettere, avendo la sicurezza di trovare il canale libero.
Permette, inoltre, di ovviare al problema del terminale nascosto, in cui due nodi collegati ad uno stesso Access Point rilevano il solo AP, ma non la presenza dell'altro nodo, generando collisioni nell'AP. Senza meccanismi come il RTS/CTS un continuo invio dello stesso pacchetto da parte di un nodo porterebbe ad annullare il throughput dell'altro nodo che vorrebbe comunicare con l'AP.
RTS/CTS può tuttavia causare il problema del terminale esposto. Un nodo C, che potrebbe comunicare senza generare interferenze con un nodo D, viene bloccato dalla ricezione di un CTS. Questo CTS viene inoltrato da un altro nodo B, che si trova nel suo range di comunicazione, il quale sta comunicando con un nodo A, al di fuori del range di C e D.

Dove si possono trovare ed analizzare le informazioni relative al Network Allocation Vector? Le informazioni sul Network Allocation Vector (NAV) sono presenti nel campo Duration (2 byte) del frame del pacchetto RTS o CTS, in cui viene specificata la durata della finestra di comunicazione riservata. Infatti, i nodi in ascolto sul mezzo wireless, leggono il campo Duration e impostano il NAV che indica al nodo quanto deve attendere per accedere al mezzo (Fig.5).

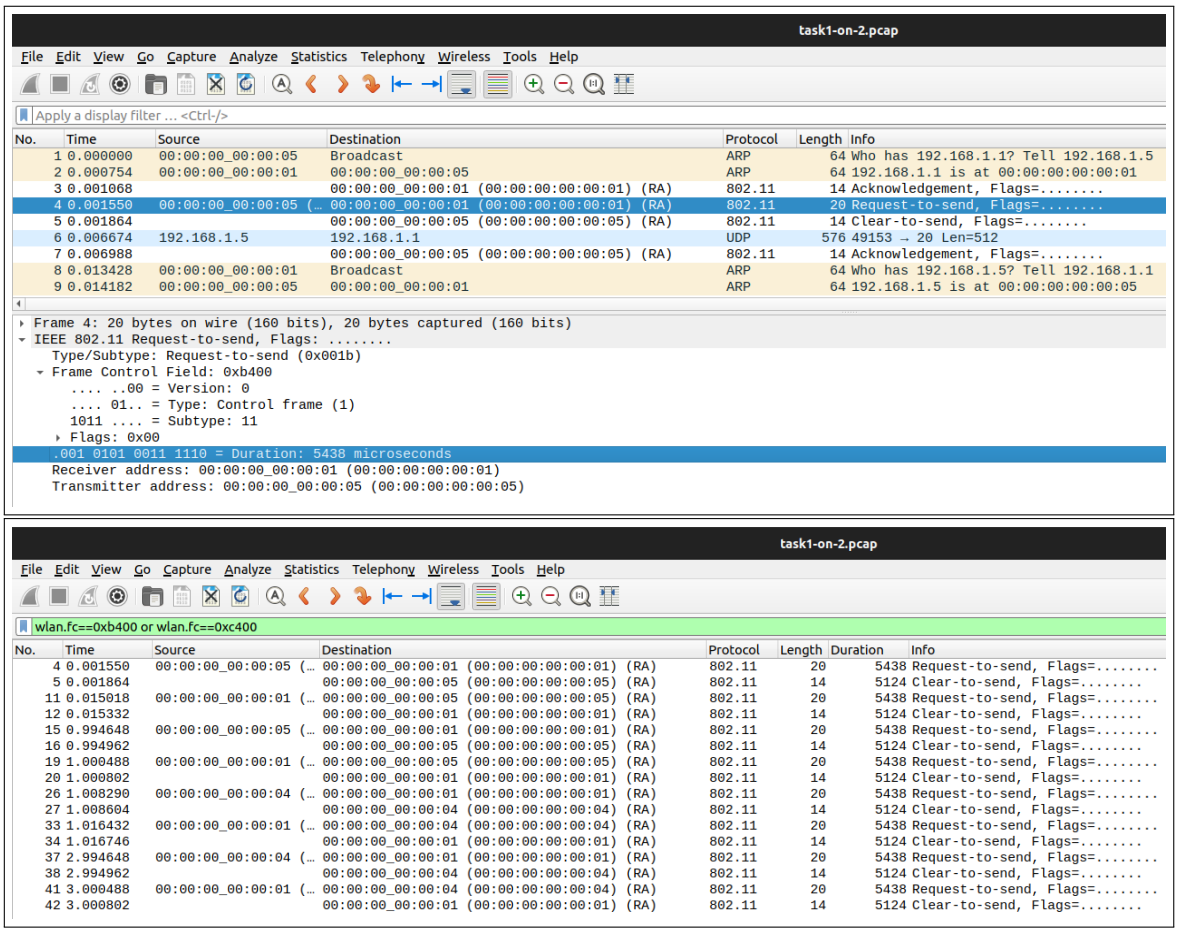


Figure 5: File task1-on-2.pcap - Filtro wlan.fc==0xb400 per RTS e wlan.fc==0xc400 per CTS

5. Calcolare il throughput medio complessivo delle applicazioni

Il goodput è calcolato considerando i dati (payload) che il livello di applicazione fornisce al livello di trasporto. Il resto dei dati contenuti nel pacchetto inviato sono inseriti dai livelli sottostanti e costituiscono il throughput totale del trasferimento. I nodi 3, con IP 192.168.1.4, e 4, con IP 192.168.1.5, inviano ciascuno 2 pacchetti con payload di dimensione 512 byte (Fig. 6).

$$T_{a\to b} = \frac{Bytes(bits)}{t_{stop} - t_{start}} \tag{1}$$

$$T_{3\to 0} = \frac{512 \cdot 2 \cdot 8}{2.999896 - 1.011386} \frac{bit}{s} \approx 4119 \frac{bit}{s} \tag{2}$$

$$T_{4\to 0} = \frac{512 \cdot 2 \cdot 8}{0.999896 - 0.005998} \frac{bit}{s} \approx 8242 \frac{bit}{s} \tag{3}$$

Pur inviando lo stesso numero di byte (1024), il nodo 4 ha un tempo totale di trasmissione pari circa alla metà del tempo di trasmissione del nodo 3.
Considerando che l'applicazione è su UdpEchoServer e UdpEchoClient, bisogna aspettarsi che dopo

la trasmissione dei pacchetti dal client al server, quest'ultimo faccia echo, il che significa che il server inoltra nuovamente i pacchetti al nodo che glieli ha inizialmente inviati. Per questo motivo, anche la trasmissione dal server al client avrà un throughput che sarà uguale a quello relativo all'invio dal client al server, non essendoci state state collisioni.
Si nota, inoltre, come in questo caso l'utilizzo della procedura di handshake RTS/CTS sia vantaggioso, considerando che per entrambe le applicazioni il tempo di trasmissione diminuisce (Fig. 7), facendo aumentare di conseguenza il throughput.

$$T_{3\to 0} = \frac{512 \cdot 2 \cdot 8}{2.999772 - 1.013414} \frac{bit}{s} \approx 4124 \frac{bit}{s} \tag{4}$$

$$T_{4\to 0} = \frac{512 \cdot 2 \cdot 8}{0.999772 - 0.006674} \frac{bit}{s} \approx 8248 \frac{bit}{s} \tag{5}$$

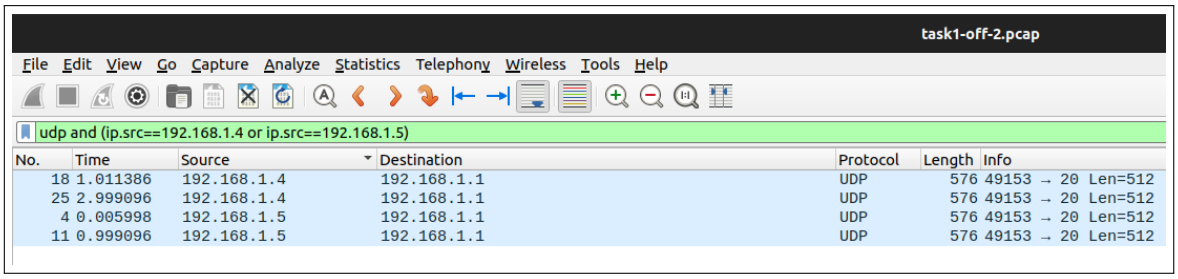


Figure 6: File task1-off-2.pcap - Pacchetti UDP inviati senza RTS/CTS

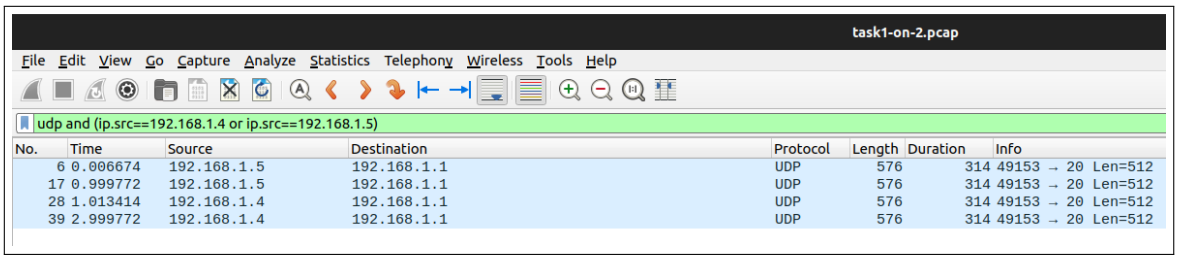


Figure 7: File task1-on-2.pcap - Pacchetti UDP inviati con RTS/CTS

Task 2

1. Spiegare il comportamento dell'AP. Cosa succede fin dal primo momento dell'inizio della simulazione?

Dal primo istante della simulazione l'AP trasmette periodicamente, in una struttura BSS, i beacon frame ai nodi della rete, i quali, fornendo determinate informazioni, annunciano la presenza di una rete wireless LAN permettendo la sincronizzazione dei nodi (Fig. 8).
Dopo l'invio del primo beacon frame, ogni nodo invia un pacchetto di Association Request all'AP, con indirizzo MAC 00:00:00:00:00:06, chiedendo di unirsi alla rete che gli è stata comunicata attraverso il beacon frame, ricevendo, successivamente, un ACK di conferma e poi una Association Response (Fig. 9).
Se la rete è composta da un numero elevato di nodi l'invio dei beacon frame richiede un tempo elevato, causando un calo di performance.

	From Id	To Id	Tx	Meta
1	5	4	0.041406	Wifi MGT_BEACON FromDS: 0 toDS: 0 DA: ff:ff:ff:ff:ff:ff SA: 00:00:00:00:00:06 BSSId: 00:00:00:00:00:06
2	5	3	0.041406	Wifi MGT_BEACON FromDS: 0 toDS: 0 DA: ff:ff:ff:ff:ff:ff SA: 00:00:00:00:00:06 BSSId: 00:00:00:00:00:06
3	5	2	0.041406	Wifi MGT_BEACON FromDS: 0 toDS: 0 DA: ff:ff:ff:ff:ff:ff SA: 00:00:00:00:00:06 BSSId: 00:00:00:00:00:06
4	5	1	0.041406	Wifi MGT_BEACON FromDS: 0 toDS: 0 DA: ff:ff:ff:ff:ff:ff SA: 00:00:00:00:00:06 BSSId: 00:00:00:00:00:06
5	5	0	0.041406	Wifi MGT_BEACON FromDS: 0 toDS: 0 DA: ff:ff:ff:ff:ff:ff SA: 00:00:00:00:00:06 BSSId: 00:00:00:00:00:06
6	4	3	0.120924	Wifi MGT_ASSOCIATION_REQUEST FromDS: 0 toDS: 0 DA: 00:00:00:00:00:06 SA: 00:00:00:00:00:05 BSSId: 00:00:00:00:00:06 SSid: 7728192
7	4	5	0.120924	Wifi MGT_ASSOCIATION_REQUEST FromDS: 0 toDS: 0 DA: 00:00:00:00:00:06 SA: 00:00:00:00:00:05 BSSId: 00:00:00:00:00:06 SSid: 7728192
8	4	1	0.120924	Wifi MGT_ASSOCIATION_REQUEST FromDS: 0 toDS: 0 DA: 00:00:00:00:00:06 SA: 00:00:00:00:00:05 BSSId: 00:00:00:00:00:06 SSid: 7728192
9	4	0	0.120924	Wifi MGT_ASSOCIATION_REQUEST FromDS: 0 toDS: 0 DA: 00:00:00:00:00:06 SA: 00:00:00:00:00:05 BSSId: 00:00:00:00:00:06 SSid: 7728192
10	4	2	0.120924	Wifi MGT_ASSOCIATION_REQUEST FromDS: 0 toDS: 0 DA: 00:00:00:00:00:06 SA: 00:00:00:00:00:05 BSSId: 00:00:00:00:00:06 SSid: 7728192
11	5	4	0.121582	Wifi CTL_ACK RA:00:00:00:00:00:05
12	5	3	0.121582	Wifi CTL_ACK RA:00:00:00:00:00:05
13	5	2	0.121582	Wifi CTL_ACK RA:00:00:00:00:00:05
14	5	1	0.121582	Wifi CTL_ACK RA:00:00:00:00:00:05
15	5	0	0.121582	Wifi CTL_ACK RA:00:00:00:00:00:05
16	5	4	0.121914	
17	5	3	0.121914	
18	5	2	0.121914	
19	5	1	0.121914	
20	5	0	0.121914	

Figure 8: File wireless-task2-rts-off.xlm - Trasmissione delle beacon frame iniziali (Riga vuota equivale a Association Response)

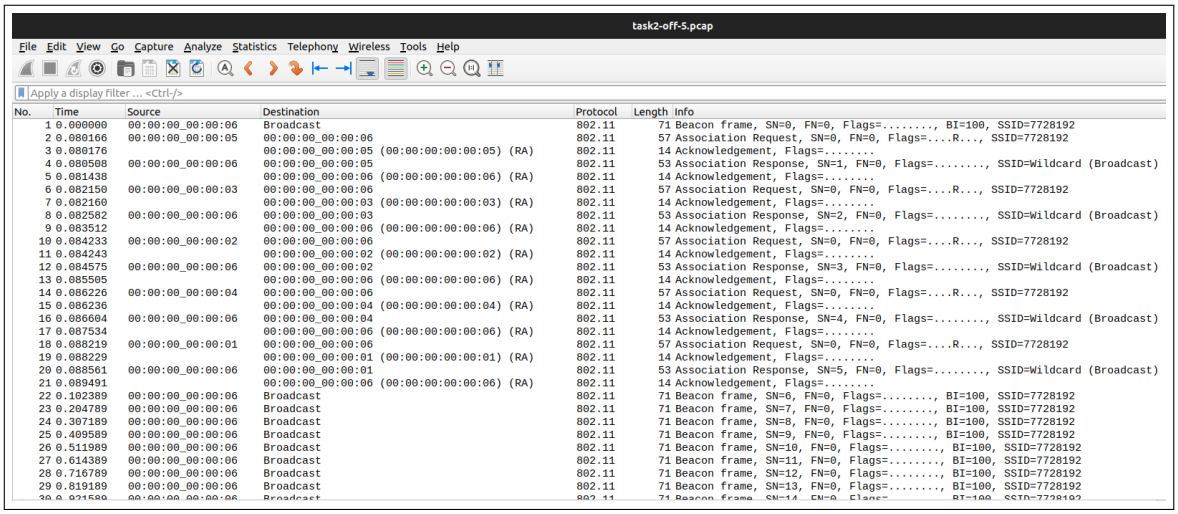


Figure 9: File task2-off-5.pcap - Evidenzia la presenza di Association Response

2. Analizzare il beacon frame. Quali sono le sue parti più rilevanti? Specificare il filtro Wireshark ed il file utilizzati per l'analisi.

Utilizzando il filtro `wlan.fc.type subtype==0x0008` su Wireshark otteniamo tutti i beacon frame inviati durante la trasmissione (Fig. 10).
Campi rilevanti da considerare sono:

- Source: mostra che è l'Access Point con indirizzo MAC 00:00:00:00:00:06 ad inviare la beacon frame
- Destination: fa notare che questo frame viene inviato a tutti i nodi collegati all'AP in broadcast, con indirizzo MAC FF:FF:FF:FF:FF:FF.

Il beacon frame è formato da fixed e tagged parameters.
Campi più importanti dei fixed parameters:

- Timestamp = rappresenta il numero di microsecondi durante i quali l'AP è stato attivo permettendo ai nodi collegati ad esso di sincronizzarsi.
- Beacon Interval = tempo in secondi che intercorre tra l'invio dei vari beacon frame.
- IBSS = campo significativo contenuto nella sezione Capabilities Information che specifica se il beacon frame è trasmesso in una rete ad-hoc o con infrastruttura BSS.

Campi più importanti dei tagged parameters:

- SSID = specifica che i frame vengono inviati ai soli nodi appartenenti alla rete con tale identifier.

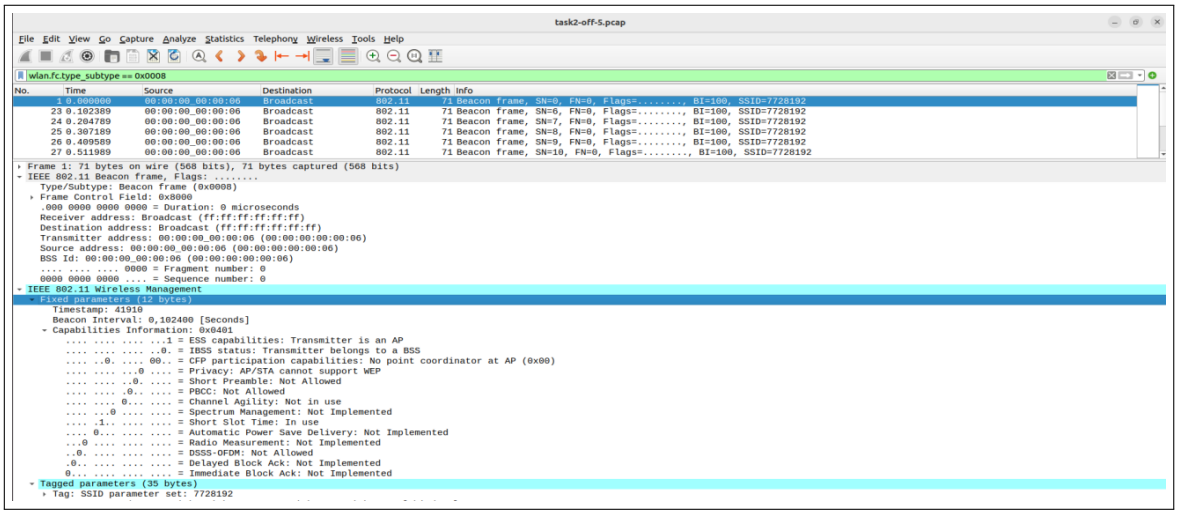


Figure 10: File task2-off-5.pcap - Campi importanti del beacon frame

3. Come per il Task 1, forzare l'uso di RTS/CTS nella rete utilizzando il parametro useRtsCts:

Ci sono delle collisioni adesso? Spiegare il perché. Prima dell'attivazione della handshake RTS/CTS si verifica una collisione. Infatti, utilizzando il filtro `wlan.fc.retry==1` si nota che al secondo 3.983609 è avvenuta la trasmissione del pacchetto UDP No. 113, che risulta però essere ritrasmesso, come si può notare dall'informazione Retry settata ad 1 nella sezione IEEE 802.11/Frame Control Field/Flags del frame (Fig. 11). Questa ritrasmissione è dovuta ad una collisione causata dall'invio verso l'Access Point di due frame diversi nello stesso istante di tempo (Fig. 12).
Dopo l'attivazione della handshake RTS/CTS per il controllo del flusso di trasmissione, non si verificano collisioni (Fig. 13). Infatti, ogni pacchetto viene inviato dal client all'Access Point il quale lo trasmette al server di destinazione. Ogni volta che il pacchetto viene trasmesso, il nodo di destinazione, che in questo caso può essere l'AP o il server, invia al nodo mittente un ACK per la conferma della corretta ricezione dei dati. Infatti, come già spiegato precedentemente, la procedura RTS/CTS serve proprio ad evitare collisioni poiché i nodi sender prenotano la trasmissione nel canale inviando una RTS e cominciano ad inviare solo quando ricevono la conferma che il canale sia libero attraverso una CTS.

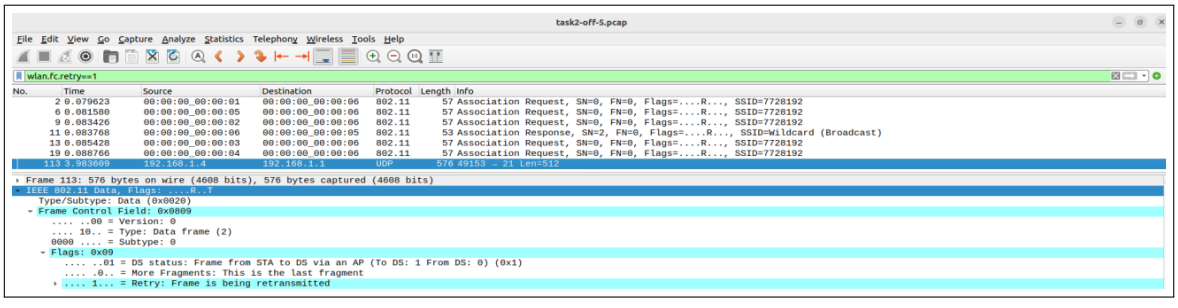


Figure 11: File task2-off-5.pcap - Ritrasmissione del pacchetto 113

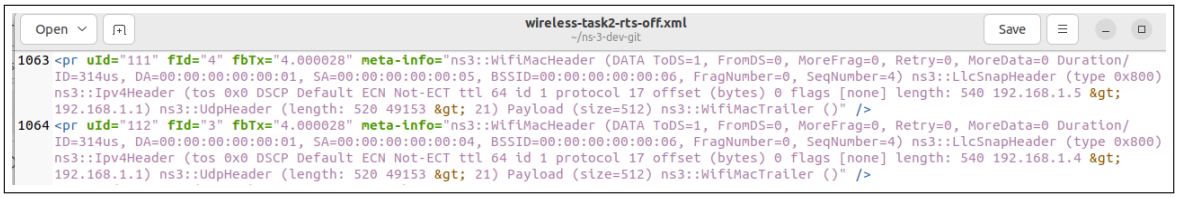


Figure 12: File wireless-task2-rts-on - Collisione

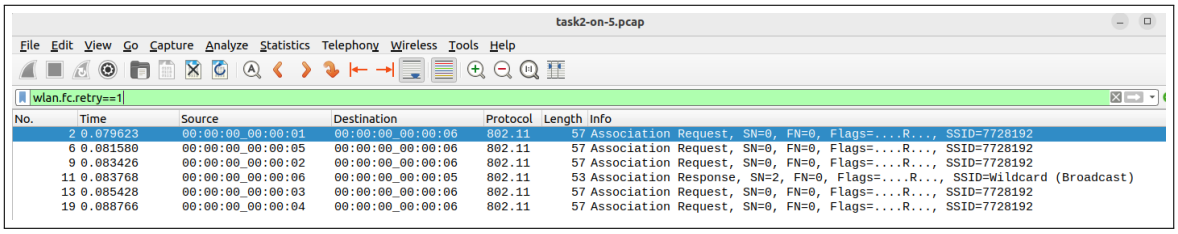


Figure 13: File task2-on-5.pcap - Assenza di collisione