Margherita Merialdo 1936363, Giuseppe Misuraca 1944376, Luca Vitaterna 1933782, Somma delle matricole = 5814521

TASK 1

1) Tutti i frame ricevono acknowledgement? Spiegare perché.

In generale secondo il protocollo Wi-Fi tutti i frame ricevono ack, esclusi i messaggi che non hanno un destinatario di cui assicurarsi la ricezione. Un esempio sono i pacchetti broadcast di arp request e beacon (se infrastructured), rispettivamente i primi possono non ricevere riscontro se non è presente l'ip cercato, altrimenti basta ARPreply come verifica; i secondi vengono ritrasmessi di continuo quindi a prescindere non ci interessa che vengano ricevuti correttamente.

Se inoltre si verifica una collisione, i frame interessati ovviamente non riceveranno ack. Questo diventa meno probabile con l'uso di RTS/CTS.

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

Leggendo il testo si può sospettare una collisione a t=2s, quando n3 e n4 dovrebbero inviare contemporaneamente un pacchetto UDP a n0. Analizzando la sezione Packets di NetAnim però non se ne osservano.

Questo perché n4 ha già comunicato con n0, ne conosce già il MAC e non deve inviare ARP req, ma solo il pac UDP che ha ritardo di elaborazione di circa 50us (valore stimato vedendo che l'invio in NetAnim ha quasi sempre questo scarto dal tempo impostato in ns3).
N3, invece, deve ancora inviare ARP req, per la quale abbiamo stimato (come sopra) un ritardo di elaborazione di 3050us. Quando sarà pronto a trasmettere, noterà che n4 lo sta già facendo e usando il CSMA/CA riproverà a trasmettere dopo un tempo di backoff.

	From Id	To Id	Tx	Meta
44	4	0	2.00005	UDP 49153 > 20
45	0	1	2.00486	Wifi CTL_ACK RA:00:00:00:00:00:05
46	0	2	2.00486	Wifi CTL_ACK RA:00:00:00:00:00:05
47	0	3	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	2	2.00521	UDP 20 > 49153
51	0	3	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	1	2.01002	Wifi CTL_ACK RA:00:00:00:00:00:01
55	4	2	2.01002	Wifi CTL_ACK RA:00:00:00:00:00:01
56	4	0	2.01002	Wifi CTL_ACK RA:00:00:00:00:00:01
57	3	4	2.01052	Arp request SMac: 00:00:00:00:00:04 DMac: ff:ff:ff:ff:ff:ff SrcIp : 192
58	3	1	2.01052	Arp request SMac: 00:00:00:00:00:04 DMac: ff:ff:ff:ff:ff:ff SrcIp : 192

fig 0: evidenza di mancata collisione

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?

Si modifica la soglia sulla dimensione dei pacchetti oltre la quale viene attivato il servizio di handshake RTS/CTS:

```
UintegerValue ctsThr = (useRtsCts ? UintegerValue(100) : UintegerValue(2200));
Config::SetDefault("ns3::WifiRemoteStationManager::RtsCtsThreshold", ctsThr);
```

Se impostiamo useRtsCts true, i pacchetti di dimensione sopra i 100 bytes usufruiranno di tale servizio; con useRtsCts false la soglia la impostiamo alta cosicché nessun pacchetto si serva di RTS/CTS.

Riduce le collisioni poiché il sender “prenota” il canale prima di trasmettere anziché affidarsi ad un accesso casuale.
Sender invia un RTS verso receiver che quando lo riceve invia a tutti un CTS (dopo un SIFS) autorizzando solo il sender a trasmettere.
Gli altri non saranno autorizzati e resteranno idle per t=NAV (tempo da attendere prima di riprovare accesso al mezzo, considerando i vari DIFS e SIFS).

4) Forzare l’uso di RTS/CTS nella rete utilizzando il parametro useRtsCts: Ci sono delle collisioni adesso? Quali sono i benefici di RTS/CTS? Dove si possono trovare ed analizzare le informazioni relative al Network Allocation Vector?

Non ci sono collisioni neanche attivando RTS/CTS, perché le ARPreq non sono precedute da RTS. Quindi n4 per poter inviare l’UDP riesce a mandare un RTS prima che n3 abbia elaborato l’ARP req. Per cui n3 troverà il canale occupato.

Utilizzando la procedura RTS/CTS le collisioni vengono minimizzate. Si evitano quelle tra frame di dati; potrebbero ancora verificarsi tra i pacchetti RTS. É comunque un beneficio perché i pacchetti RTS sono molto piccoli quindi le collisioni sono meno probabili e fanno perdere meno info.

Le informazioni sul NAV si trovano analizzando la voce Duration dei pacchetti RTS o CTS nei pcap notando che nei secondi esso è minore come la teoria suggerisce.

5) Calcolare il throughput medio complessivo delle applicazioni

Rts/Cts On

n3 <-> n0 vengono inviati 2304 byte in un intervallo di tempo pari a 1.9921s, si ottiene un throughput = $\frac{2304byte \cdot 8}{1.9921s} = 9252,547bit/s$

n4 <-> n0 vengono inviati 2304 byte in un intervallo di 1.0010s, si ottiene un throughput pari a:

$$\frac{2304byte \cdot 8}{1.0010s} = 18413,586bit/s$$

Rts/Cts Off

n3 <-> n0 vengono inviati 2304 byte in un intervallo di 1.9927s, si ottiene un throughput pari a:

$$\frac{2304byte \cdot 8}{1.9927s} = 9249,761bit/s$$

n4 <-> n0 vengono inviati 2304 byte in un intervallo di 1.0003s, si ottiene un throughput pari a:

$$\frac{2304byte \cdot 8}{1.0003} = 18426,472bit/s$$

Dati ottenuti esaminando il pcap con wireshark nella sezione : Statistics>Conversations
L'uso dell'handshake modifica i tempi.

TASK 2

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

Gli AP sono dei dispositivi che si interfacciano con gli host, gestiscono accesso al mezzo trasmissivo e si comportano come bridge verso altre reti esterne.
L'AP invia periodicamente beacon per segnalare la propria presenza e comunicare informazioni riguardo la gestione della comunicazione.
L'AP riceve le richieste di associazione degli host, manda un ACK e poi un ASSOCIATION RESPONSE il cui parametro STATUS CODE indica se la richiesta è stata approvata o no. In seguito all'approvazione è istanziato un association id che identifica il nodo all'interno della rete (nodo non più identificato solo da mac). Per il resto del tempo media le comunicazioni tra coppie di station ricevendo e rilanciandone i messaggi.

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

Un beacon frame contiene tutte le informazioni che una base station necessita per rilevare un AP, sincronizzarsi impostando i parametri e trasmettere frame in quella rete.
Per analizzare i beacon frame usiamo il seguente filtro, su tutti i file pcap generati:

wlan.fc.type_subtype == 0x8

0 indica la tipologia la categoria dei *management frames*, 8 la sottocategoria dei beacon frame.

- BSSID : il MAC address di AP
- Capabilities Information : info sui servizi offerti dalla rete
- Beacon interval : tempo che intercorre fra invii successivi di beacon
- SSID parameter set : nome che identifica la rete wifi
- Supported rates : data rates supportati (nel caso ve ne fossero molti Extended support rates che ne specifica altri)
- DS : il canale dove è attivo AP
- ERP presente solo in reti che supportano la 802.11g, per retrocompatibilità con station aventi versioni meno recenti della 802.11g

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

Ipotizziamo collisioni nei momenti in cui più nodi comunicano contemporaneamente con l' AP (n5) cioè durante le association e a t = 4s. Si verificano le seguenti collisioni:
RTS off:

- l'ass. request di n1 (fig1) non riesce ad arrivare all'AP ma solo a n0. Verrà infatti ritrasmessa a 0.124789s.

	From Id	To Id	Tx	
31	5	0	0.122697	Wifi CTL_ACK RA:00:00:00:00:00:05
32	1	0	0.123056	Wifi MGT_ASSOCIATION_REQUEST FromDS: 0 toDS: 0 DA: 00:00:00:
33	5	4	0.123768	
34	5	2	0.123768	
35	5	3	0.123768	
36	5	1	0.123768	
37	5	0	0.123768	
38	3	4	0.124394	Wifi CTL_ACK RA:00:00:00:00:00:01
39	3	0	0.124394	Wifi CTL_ACK RA:00:00:00:00:00:01
40	3	5	0.124394	Wifi CTL_ACK RA:00:00:00:00:00:01
41	3	1	0.124394	Wifi CTL_ACK RA:00:00:00:00:00:01
42	3	2	0.124394	Wifi CTL_ACK RA:00:00:00:00:00:01
43	1	2	0.124789	Wifi MGT_ASSOCIATION_REQUEST FromDS: 0 toDS: 0 DA: 00:00:00:
44	1	0	0.124789	Wifi MGT_ASSOCIATION_REQUEST FromDS: 0 toDS: 0 DA: 00:00:00:
45	1	4	0.124789	Wifi MGT_ASSOCIATION_REQUEST FromDS: 0 toDS: 0 DA: 00:00:00:

fig 1: tentativo fallito di associazione riga 32. Ritrasmissione (riga 43)

- quando n3 e n4 inviano un UDP. Essendo n4 molto vicino all'AP (fig 2), riesce ad inviarglielo, invece n3 riesce a mandarlo solo a n0 (fig 3) ed è costretto a ritrasmetterlo a 4.02068s.

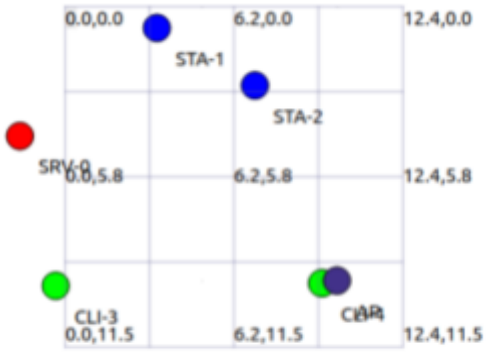

fig 2: posizione dei nodi a t = 4s

	From Id	To Id	Tx	
511	5	1	3.95079	Wifi MGT_BEACON FromDS: 0 toDS: 0 DA: ff:ff:ff:ff:ff:ff SA: 00:00:0
512	5	0	3.95079	Wifi MGT_BEACON FromDS: 0 toDS: 0 DA: ff:ff:ff:ff:ff:ff SA: 00:00:0
513	4	5	4.00003	UDP 49153 > 21
514	3	0	4.00003	UDP 49153 > 21
515	4	2	4.00003	UDP 49153 > 21
516	5	4	4.00484	Wifi CTL_ACK RA:00:00:00:00:00:06
517	5	2	4.00484	Wifi CTL_ACK RA:00:00:00:00:00:06
518	5	3	4.00484	Wifi CTL_ACK RA:00:00:00:00:00:06
519	5	1	4.00484	Wifi CTL_ACK RA:00:00:00:00:00:06
520	5	0	4.00484	Wifi CTL_ACK RA:00:00:00:00:00:06
521	5	4	4.00517	UDP 49153 > 21
522	5	2	4.00517	UDP 49153 > 21
523	5	3	4.00517	UDP 49153 > 21
524	5	1	4.00517	UDP 49153 > 21
525	5	0	4.00517	UDP 49153 > 21

fig 3: tentativo fallito di n3 di inviare UDP (riga 514)

RTS on: collidono gli stessi punti, ma a t=4 lo fanno gli RTS:

- le ass. req. non superano il threshold quindi non cambia nulla.
- n3 non riceve il CTS (fig 4) dall'AP quindi fa backoff e ritrasmette in seguito.

Quindi RTS non risolve queste collisioni perché non sono dovute a hidden terminal. Ricordiamo che le collisioni RTS sono meno dannose.

	From Id	To Id	Tx	
590	5	3	3.95079	Wifi MGT_BEACON FromDS: 0 toDS: 0 DA: ff:ff:ff:ff:ff:ff SA: 00:00:
591	5	1	3.95079	Wifi MGT_BEACON FromDS: 0 toDS: 0 DA: ff:ff:ff:ff:ff:ff SA: 00:00:
592	5	0	3.95079	Wifi MGT_BEACON FromDS: 0 toDS: 0 DA: ff:ff:ff:ff:ff:ff SA: 00:00:
593	4	5	4.00003	Wifi CTL_RTS RA:00:00:00:00:00:01 TA:00:00:00:00:00:06
594	3	0	4.00003	Wifi CTL_RTS RA:00:00:00:00:00:01 TA:00:00:00:00:00:05
595	4	2	4.00003	Wifi CTL_RTS RA:00:00:00:00:00:01 TA:00:00:00:00:00:06
596	5	4	4.00039	Wifi CTL_CTS RA:00:00:00:00:00:06
597	5	2	4.00039	Wifi CTL_CTS RA:00:00:00:00:00:06
598	5	3	4.00039	Wifi CTL_CTS RA:00:00:00:00:00:06
599	5	1	4.00039	Wifi CTL_CTS RA:00:00:00:00:00:06
600	5	0	4.00039	Wifi CTL_CTS RA:00:00:00:00:00:06
601	4	5	4.0007	UDP 49153 > 21
602	4	2	4.0007	UDP 49153 > 21
603	4	1	4.0007	UDP 49153 > 21
604	4	3	4.0007	UDP 49153 > 21

fig 4: tentativo fallito di n3 di mandare RTS (riga 594)