Domande per ogni configurazione:
A1) Individuare le varie topologie note che compongono la rete.

La rete proposta è caratterizzata da una topologia mista in cui si possono riconoscere:

- Una topologia ad albero in cui il nodo 3 è il nodo radice, n1 e n6 sono i figli e quest'ultimo è padre di n5 che a sua volta è padre del nodo foglia n4.
  Si noti che in questa topologia il fallimento di un nodo comporterebbe l'impossibilità di far comunicare fra di loro i nodi ad esso collegati. Nel nostro caso, se ciò dovesse accadere a n6 la comunicazione fra n3 e i figli di n6 sarebbe impossibilitata.

- Una sottorete con topologia lineare costituita da due collegamenti point to point tra n4, n5 ed n6. Il fallimento di n5 impedisce la comunicazione tra n4 ed n6;

- Due sottoreti con topologia a bus, riconoscibili perché i nodi n0-n1-n2 e n6-n7-n8 non sono collegati attraverso link point to point tra nodi, ma sono tutti collegati ad un link csma condiviso (csma è un protocollo del livello di collegamento tipico delle reti a bus).

A2) Ricostruzione del percorso dei pacchetti attraverso la rete di tutti i flussi simulati usando wireshark evidenziando i filtri utilizzati per isolare i singoli flussi dello strato di trasporto tra le tracce. Nota: Ogni configurazione a secondo dello stato del canale e della topologia può seguire un percorso diverso, è importante quindi evidenziare eventuali differenze al variare della configurazione e ai filtri utilizzati.

conf 0:

- n4-n2 segue il percorso l2 l3 l1 l0 csmalink, poiché analizzando i file pcap n5 n6 e n3 con filtro tcp && ip.src == 10.0.3.2 noto che tutti i pacchetti filtrati hanno come destinazione n2.
  Il percorso sopra riportato è ulteriormente confermato dal fatto che nel file pcap n6csma con il filtro tcp && ip.src == 10.0.3.2 non compaiano pacchetti, il che significa che questi non percorreranno quel link.

conf 1:

- n4-n0 applicando la medesima analisi di n4-n2 seguirà un percorso simile a questi, ma con dest pari a n0.

- n8-n0 percorrerá csmalink l1 l0 csmalink, infatti analizzando i file pcap n5 n6csma n3, con filtro tcp && ip.src == 192.138.2.3 tutti i pacchetti hanno destinazione n0.
  Siamo sicuri che non percorrerà altri link, poiché usando il filtro tcp && ip.src==192.138.2.3 il file task1-1-n6.pcap è privo di pacchetti.

conf 2:

- n8-n2 usando un approccio identico a n8-n0 seguirà un percorso simile a questi ma con ip dest uguale a 192.138.1.3.

- n4-n2 è esattamente uguale al flusso n4-n2 della conf 0

- n7-n0 ci aspettiamo che segua il percorso analogo a n8-n0, differendo soltanto per id src che sarà pari a n7. Ne abbiamo conferma con la solita strategia.

A3) Calcolo e grafico di round trip time (RTT) e commento.

I grafici sono tratti da Wireshark Statistics TCP Stream Graph sui file pcap più vicini al client.
I calcoli sono effettuati dai file tr del client.
Quindi ci aspettiamo di vedere sul grafico un valore leggermente inferiore in quanto Wireshark calcola RTT rispetto al nodo del pcap (t cattura ack - t cattura segm) ignorando il percorso client - nodopcap.

Config 0:

- Calcolo RTT sul pacchetto 400, con il file task1-0-n4.tr sottraendo i tempi di invio del pacchetto e di lettura del riscontro
  t (+) pacchetto 400 (seq=132001, len=536) : 5.14628s
  t (r) pacchetto 403 (ack = 132537) : 5.14752s
  RTT = 5.14752s - 5.14628s = 1.24ms (1.16ms in pcap)

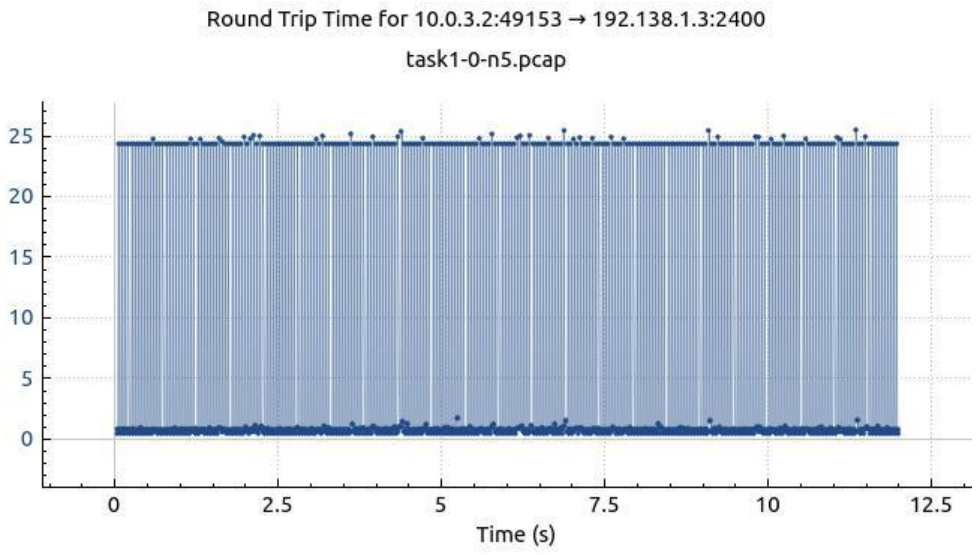

figura 1: grafico RTT configurazione 0

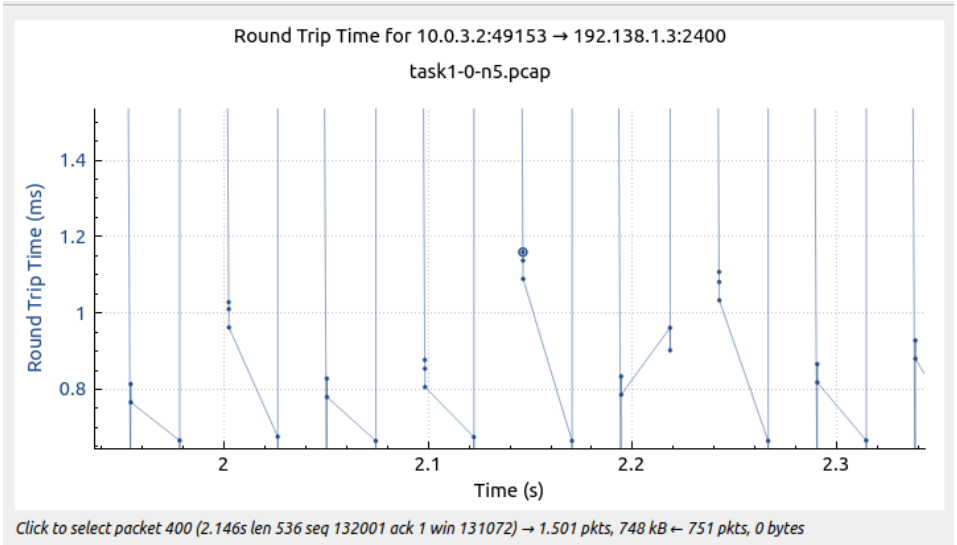

figura 2: grafico RTT configurazione 0, dettaglio pacchetto 400

Config 1:

- Calcolo RTT sul pacchetto 124 del flusso n4-n0 con il file task1-1-n4.tr sottraendo i tempi di invio del pacchetto e di lettura del riscontro
  t (+) pacchetto 124 (seq=40001, len=536): 5.68828s
  t (r) cattura pacchetto 126 (ack = 40537) : 5.68992s
  RTT = 5.68992s - 5.68828s = 1.64ms (1.57ms in pcap)

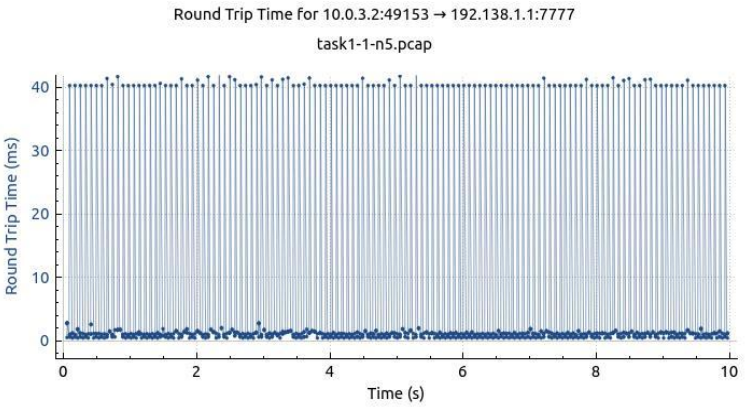

figura 3: grafico RTT configurazione 1, flusso n4-n0

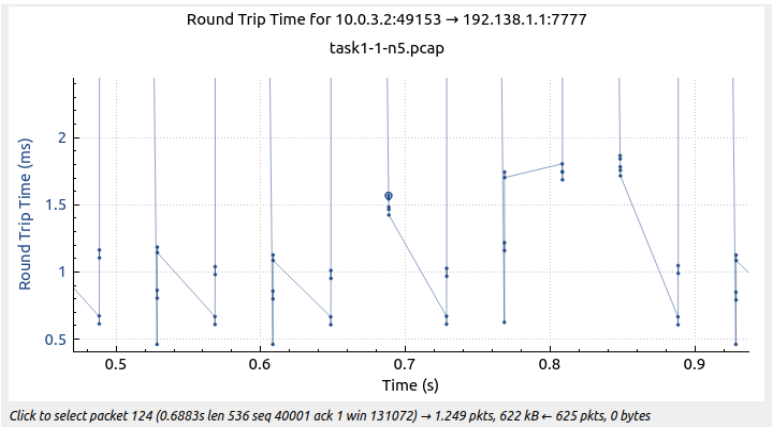

figura 4: grafico RTT configurazione 1, flusso n4-n0, dettaglio pacchetto 124

- Calcolo RTT sul pacchetto 18 del flusso n8-n2 con il file task1-1-n6csma.pcap sottraendo i tempi di cattura del pacchetto e del riscontro
  t cattura pacchetto 18 (seq=4289, len=212): 2.09547s
  t cattura pacchetto 21 (ack =4501) : 2.0980910s
  RTT = 2.090910s - 2.090168s = 0.74ms

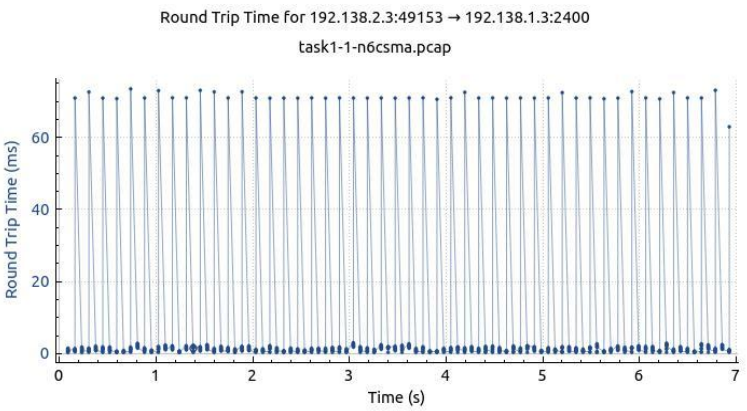

figura 5: grafico RTT configurazione 1, flusso n8-n2

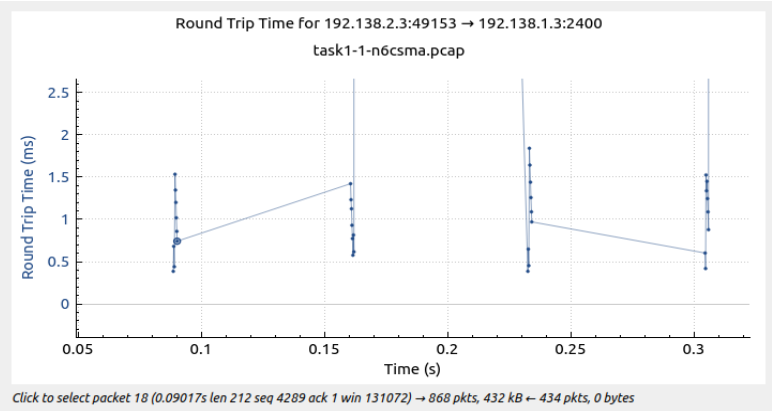

figura 6: grafico RTT configurazione 1, flusso n8-n2, dettaglio pacchetto 18

Config 2:

- Calcolo RTT sul pacchetto 67 del flusso n4-n2 con il file task1-0-n4.tr sottraendo i tempi di invio del pacchetto e di lettura del riscontro
  t partenza pacchetto 67 (seq=21001, len=536): 3.39428s
  t ritorno pacchetto 73 (ack = 21357) : 3.39663s

  RTT = 3.39663s - 3.39428s = 2.35ms (2.28ms in pcap)

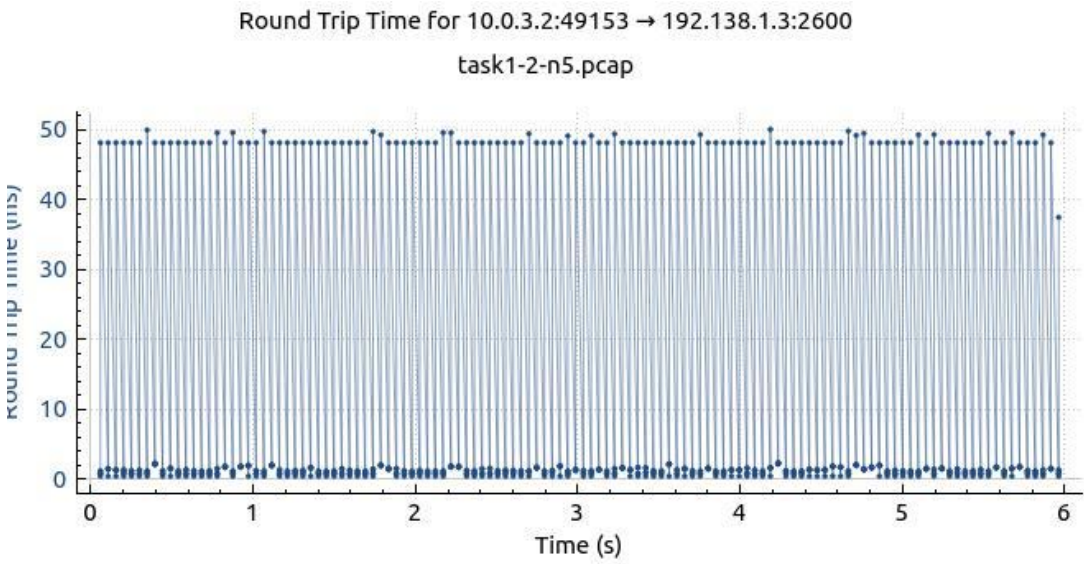

figura 7: grafico RTT configurazione 2

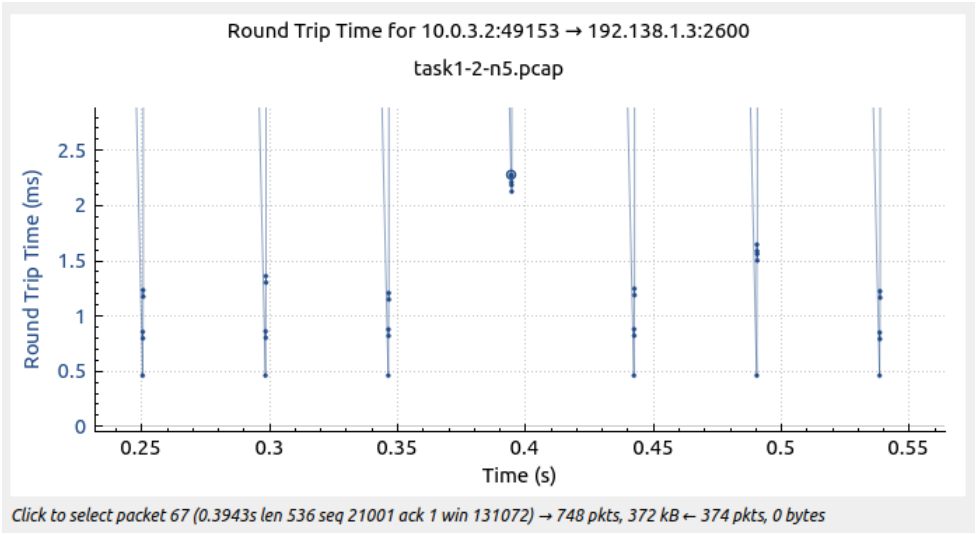

figura 8: grafico RTT configurazione 2, dettaglio pacchetto 67

A4) Vi sono dei bottleneck nella rete? Se si, individuare gli eventuali link e discutere eventuali contromisure e soluzioni.

I link point to point hanno tutti bitrate 80Mbps. I csma link hanno un data rate inferiore: 25Mbps tra n0, n1 ed n2 e 30Mbps tra n6, n7 ed n8.
Quindi in una comunicazione che attraversi sia link p2p che i csma link questi ultimi costituiranno un bottleneck, in particolare quello costituito da n0, n1 ed n2.

Infatti se la trasmissione fosse tra uno dei nodi n0 o n2 e uno dei nodi n7 o n8 il bottleneck sarebbe il csma link a cui appartiene il nodo n1 in quanto il suo data rate è inferiore all'altro csma link.

Dovremmo vederne gli effetti in tutte le comunicazioni della simulazione. In realtà nella simulazione, per via della natura delle applicazioni, non vengono quasi mai raggiunte velocità di trasmissione che saturino il bottleneck, quindi non ne vediamo gli effetti.

Possibili soluzioni al bottleneck:
-miglioramento della capacità di trasmissione dove si ha bottleneck, così da diminuire la disparità con i data rate degli altri link.
Ciò si può ottenere sostituendo le tecnologie usate con delle versioni più recenti, come l'uso della fibra nei cavi LAN.

Domande configurazione 0:

C01) Calcolare il throughput istantaneo del flusso TCP.

Lo calcoliamo considerando un singolo pacchetto in modo che l'intervallo di tempo sia sufficientemente basso da poter dare una buona stima del TH istantaneo.
Dal file task1-0-n3.pcap considero la differenza di tempo tra un segmento da 1500B (3 frame per un totale di 1662B) e il successivo:
t cattura pacchetto 193 (seq=63001) = 4.042441s
t cattura pacchetto 198 (seq=64501 (63001+1500)) = 4.066441s

$$THistantaneo = \frac{(1662byte * 8)}{(4.066441 - 4.042441)s} = 554000bps$$

Si può notare che il valore è in accordo con il grafico in figura 9, valido anche per le domande C02 e C03. Dal momento che il TH si mantiene all'incirca costante, non sorprende che quello istantaneo sia simile a quello medio. Per notare una differenza bisognerebbe prendere un intervallo di tempo talmente piccolo da includere solo un periodo di On del client.

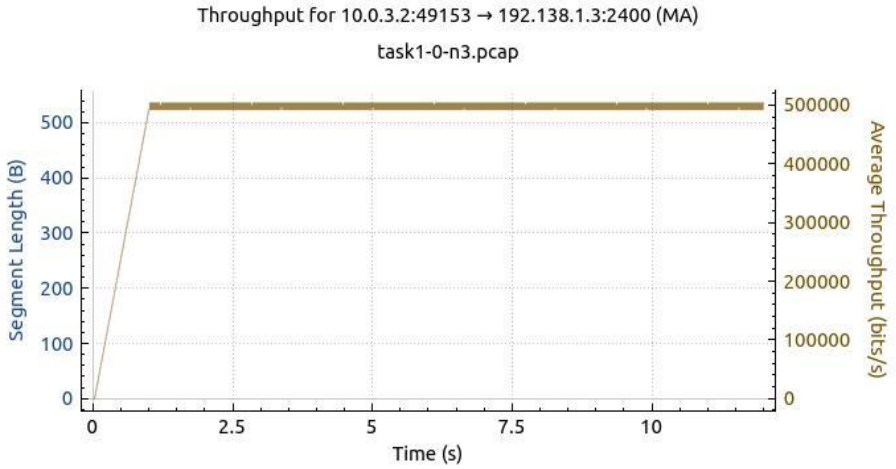

figura 9: grafico throughput tcp configurazione 0

C02) Calcolare il throughput medio del flusso TCP a tempo t=4.0s.

Per calcolare il throughput medio al tempo richiesto (t=4.0s), apriamo il file task1-0-n3.pcap impostando il filtro: frame.time_relative <= 1.0 && tcp && ip.src == 10.0.3.2
Sottolineiamo come il limite di tempo relativo usato sia 1s dato che Wireshark inizia a contare il tempo dall'invio dei pacchetti e non dall'avvio della simulazione.
Successivamento andando su Statistics > Conversation abbiamo ottenuto i seguenti dati:
- Packets A → B: 125
- Bytes A → B: 68254
- Duration: 0.99s

Per calcolare il throughput medio abbiamo diviso il numero di bit inviati per la durata dell'intervallo in cui si sono scambiati pacchetti (seguendo la definizione per cui il throughput medio è bit inviati diviso il tempo necessario per l'invio).

$$TH = \frac{68254byte \cdot 8}{0.99s} = 551547,47bit/s$$

C03) Calcolare il throughput medio del flusso TCP a tempo t=7.0s. Commentare eventuali cambiamenti rispetto a C02.

Per calcolare il throughput medio fino a t = 7s abbiamo usato l'identico metodo della domanda precedente, sullo stesso file .pcap, usando un filtro differente:
frame.time_relative <= 4.0 && tcp && ip.src == 10.0.3.2
I dati ottenuti su Wireshark sono i seguenti:
- Packets A → B: 500
- Bytes A → B: 276004
- Duration: 3.99s

$$TH = \frac{276004byte \cdot 8}{3.99s} = 553391,47bit/s$$

Confrontando i due throughput ottenuti possiamo notare che non c'è un'ampia differenza fra i due valori. Questo risultato si ottiene perché, come si può vedere dal grafico in figura 9, il valore del throughput dopo un determinato intervallo si assesta su un valore vicino a 0,5 Mbps.
Analizzando la documentazione del TCP OnOff Application abbiamo notato che il suo data rate di default è proprio 0.5 Mbps, non avendolo modificato ed essendo inferiore del bottleneck supponiamo che sia normale che il valore si assesti su questa cifra.

C04) Calcolare il ritardo di trasferimento complessivo di tutti i pacchetti inviati.

In task1-0-n4.tr notiamo che il primo pacchetto parte a t=3s, in task1-0-n2.tr l'ultimo pacchetto viene ricevuto a t=15,0002s, quindi complessivamente il trasferimento di tutti i pacchetti impiega 12,0002s.
Nei calcoli consideriamo anche handshake e chiusura e che l'OnOffApplication si interrompe periodicamente per mantenere un bitrate medio di 0.5Mbps.

Applicando su task1-0-n5.pcap il filtro (tcp && ip.src==10.0.3.2) osserviamo il ritardo di trasferimento come se la rete fosse stata sfruttata a pieno dall'applicazione:

Conversations: Bytes = 829kB
primo pack length = 58B
Trascuriamo rit di accodamento e di elaborazione. Assumiamo store and forward

rit di prop = (5*4+10)us
rit di trasm primo pack tutte le interfacce = (58*8bit)*4/80Mbps + 58*8bit/25Mbps
rit di trasm tutti i pack ultima interfaccia = (829˙000*8)bit/25Mbps
rit di trasm primo pack ultima interfaccia = (58*8)bit/25Mbps

$$rit_{trasf-compl} = (3 \cdot 10^{-5}s) + (4.176 \cdot 10^{-5}s) + (0,26528s) + (1,856 \cdot 10^{-5}s) = 0265s$$

Domande configurazione 1:

C11) Calcolare il throughput medio dei flussi TCP.

Calcolo del throughput medio dei seguenti flussi:

- -n8 n2: analizzando il pacchetto task1-1-n6csma.pcap e applicando il filtro: tcp && ip.src==192.138.2.3, abbiamo ottenuto:
  - Address A: 192.138.2.3
  - Port A: 49153
  - Address B: 192.138.1.3
  - Port B: 2400
  - Packets A → B: 868
  - Bytes A → B: 492764
  - Duration: 6.99s

$$TH = \frac{492764byte \cdot 8}{6.99s} = 563964bit/s$$

- -n4 n0: analizzando il pacchetto task1-1-n6.pcap e applicando il filtro: tcp && ip.src==10.0.3.2, si ha:
  - Address A: 10.0.3.2
  - Port A: 49153
  - Address B: 192.138.1.1
  - Port B: 7777
  - Packets A → B: 1249
  - Bytes A → B: 689950
  - Duration: 10.0s

  throughput medio del flusso da n4 a n0 per tutto l'intervallo in cui trasmettono

$$TH = \frac{689950byte \cdot 8}{10.0s} = 551960bit/s$$

C12) Calcolare il throughput medio del flusso TCP n8 verso n2 a tempo t=6s.

Come richiesto dalla domanda vogliamo calcolare il throughput medio al tempo richiesto (t=6.0s), per farlo apriamo il file taks1-1-n6csma.pcap impostando il filtro: frame.time_relative <= 4.0 && tcp && ip.src==192.138.2.3

Mettiamo in evidenza come il limite di tempo relativo usato sia di 4s dato che Wireshark inizia a contare il tempo dall'invio dei pacchetti e non dall'avvio della simulazione.

Successivamento andando su Statistics > Conversation abbiamo ottenuto i seguenti dati:

- Packets A → B: 497
- Bytes A → B: 282294
- Duration: 3.97s

Per calcolare il throughput medio abbiamo diviso il numero di bit inviati per l'intervallo di tempo richiesto, (seguendo la definizione per cui il throughput medio è bit inviati / tempo necessario per l'invio).

$$TH = \frac{282294byte \cdot 8}{3.97s} = 568854,40bit/s$$

C13) Calcolare il throughput medio del flusso TCP n8 verso n2 a tempo t=8s. Commentare eventuali cambiamenti rispetto a C12.

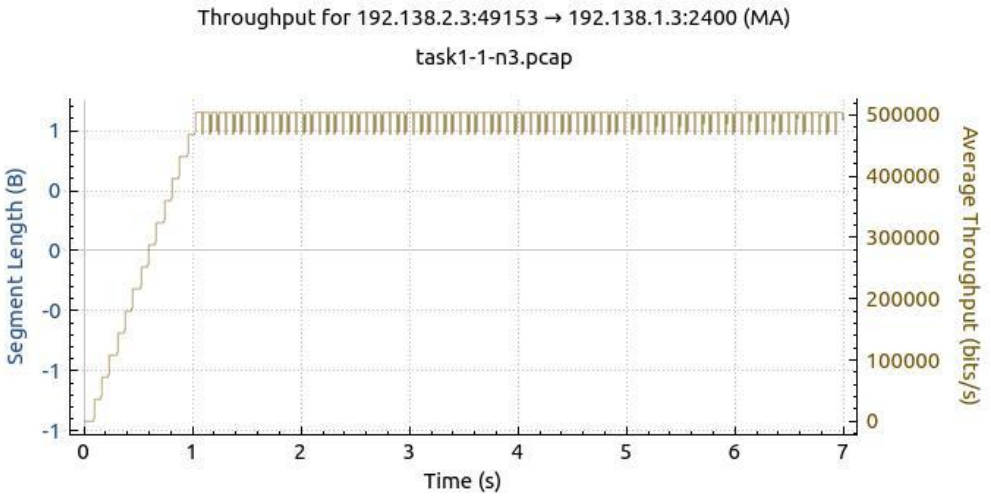

figura 10: grafico throughput configurazione 1

Per calcolare il throughput medio fino a t = 8s abbiamo usato l'identico metodo della domanda precedente, sullo stesso file .pcap, usando un filtro differente:
frame.time_relative <= 6.0 && tcp && ip.src == 192.138.2.3

I dati ottenuti su Wireshark sono i seguenti:
- Packets A → B: 749
- Bytes A → B: 425934
- Duration: 5.99s

Throughput del flusso TCP dal secondo 0 al secondo 8 della simulazione.

$$TH = \frac{425934byte \cdot 8}{5.99s} = 568860,10bit/s$$

Comparando i risultati dei throughput calcolati, anche in questo caso, osserviamo che non ci sia molta disparità tra i due valori. Questo risultato si ottiene perché, come si può vedere dal grafico in figura 10, il valore del throughput dopo un determinato intervallo si assesta su un valore vicino a 0,5 Mbps.
Come spiegato precedentemente ipotizziamo che ciò sia dovuto al data rate di default del TCP OnOff Application

Domande configurazione 2:

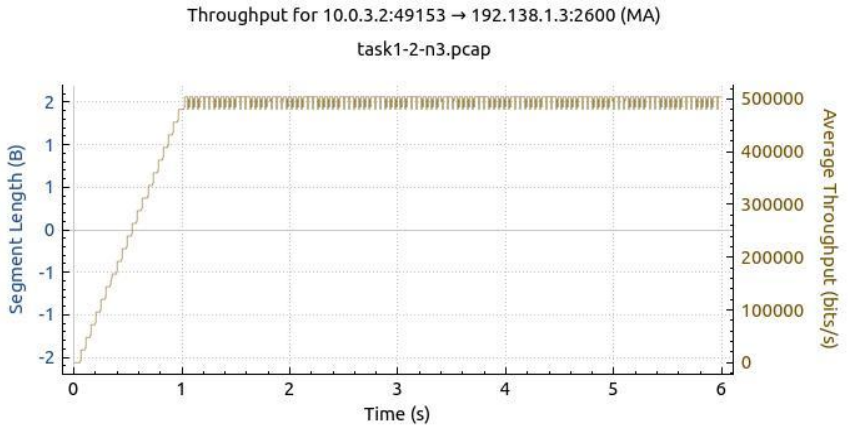

figura 11: grafico throughput configurazione 2

C21) Calcolare il throughput medio del flusso TCP a tempo t=5s.

Calcoliamo il throughput medio dall'inizio al tempo richiesto (t=5.0s), per farlo apriamo il file taks1-2-n6.pcap impostando il filtro: frame.time_relative <= 2.0 && tcp && ip.src == 10.0.3.2
Come già detto il tempo considerato per lo scambio di bytes sia solo di 2s dato che Wireshark avvia il tempo dal primo pacchetto inviato e non dall'avvio della simulazione.
Successivamento andando su Statistics > Conversation abbiamo ottenuto i seguenti dati:

- Packets A → B: 248
- Bytes A → B: 136396
- Duration: 1.97s

Per calcolare il throughput medio abbiamo diviso il numero di bit inviati per l'intervallo di tempo richiesto (seguendo la definizione per cui il throughput medio è byte inviati / tempo necessario per l'invio)

$$TH = \frac{136396byte \cdot 8}{1.97s} = 553892,38bit/s$$

C22) Calcolare il throughput medio del flusso TCP a tempo t=7s. Commentare eventuali cambiamenti rispetto a C21.

Per calcolare il throughput medio fino a t = 7s abbiamo usato l'identico metodo della domanda precedente, sullo stesso file .pcap, ma usando un filtro differente:
frame.time_relative <= 7.0 && tcp && ip.src == 32.0.3.2
I dati ottenuti su Wireshark sono i seguenti:

- Packets A → B: 500
- Bytes A → B: 276004
- Duration: 3.99s

$$TH = \frac{276004byte \cdot 8}{3.99s} = 553391,47bit/s$$

Come precedentemente riportato anche qui non riscontriamo cambiamenti significativi di throughput rispetto al calcolo scorso. Il grafico in figura 11 si attesta su un valore costante.