Report Gruppo 46 – Homework 3

Membri:

- Lorenzo Pannacci - 1948926 (leader)
- Matteo Pannacci - 1948942
- Liwen Zheng - 1915940
- Matteo Terrinoni - 1883939

Assunzioni:

1. Nella configurazione 2 viene chiesto di inserire in alcuni segmenti la somma dei numeri di matricola come intero. Abbiamo svolto la consegna inserendo in uno spazio di 2560 bytes tutti zeri tranne i primi 4, che contengono il numero rappresentato come INT.

Ritroviamo il numero (7697747) su Wireshark in esadecimale come “53 75 75 00”, questo perché il sistema operativo utilizza una rappresentazione numerica in Little Endian (le prime cifre sono le meno significative).

2. Quando parleremo di bottleneck non considereremo il data-rate a livello di client in quanto non è parte della rete ma interno all’applicazione stessa.

3. Poiché nella domanda C01 non è specificato su quale istante calcolare il throughput istantaneo, su suggerimento della prof.ssa Cuomo si è calcolato il throughput istantaneo su due pacchetti scelti arbitrariamente.

4. Nelle domande che chiedono il throughput mostreremo sia il throughput complessivo sia quello legato soltanto ai messaggi dello stato applicativo in quanto la definizione data e lezione e quella usata da Wireshark divergono.

5. I file .pcap generati dal programma si riferiscono ad il link l0 per n3, l2 per n5 e csma2 per n6 in quanto sono quelli considerati più significativi per tutte le configurazioni.

6. Calcoleremo il ritardo di trasferimento come la differenza tra il tempo di ricezione dell’ultimo pacchetto da parte del server e quello di invio del primo pacchetto da parte del client.

A1) Individuare le varie topologie note che compongono la rete.

La rete descritta nell'homework è composta dalle seguenti topologie:

- Topologia BUS composta dai nodi n0, n1, n2 (CSMA link di sinistra, che indicheremo con **csma1**)
- Topologia BUS composta dai nodi n6, n7, n8 (CSMA link di destra, che indicheremo con **csma2**)
- Topologia Point-To-Point tra i nodi n1, n3 (indicata con **l0**)
- Topologia Point-To-Point tra i nodi n3, n6 (indicata con **l1**)
- Topologia Point-To-Point tra i nodi n4, n5 (indicata con **l2**)
- Topologia Point-To-Point tra i nodi n5, n6 (indicata con **l3**)

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 seconda dello stato del canale e della topologia può seguire un percorso diverso, è importante quindi evidenziare eventuali differenze al variare della configurazione e i filtri utilizzati.

Configurazione 0:

Per la comunicazione da n4 a n2 abbiamo "***n4-n5-n6-n3-n1-n2***" (figura 1).

Infatti:

- task1-0-n6.pcap: vuoto (nessun pacchetto, punto di vista di csma2)
- task1-0-n5.pcap: con il filtro "*ip.addr == 10.0.3.1*" si hanno tutti i pacchetti
- task1-0-n3.pcap: come il precedente

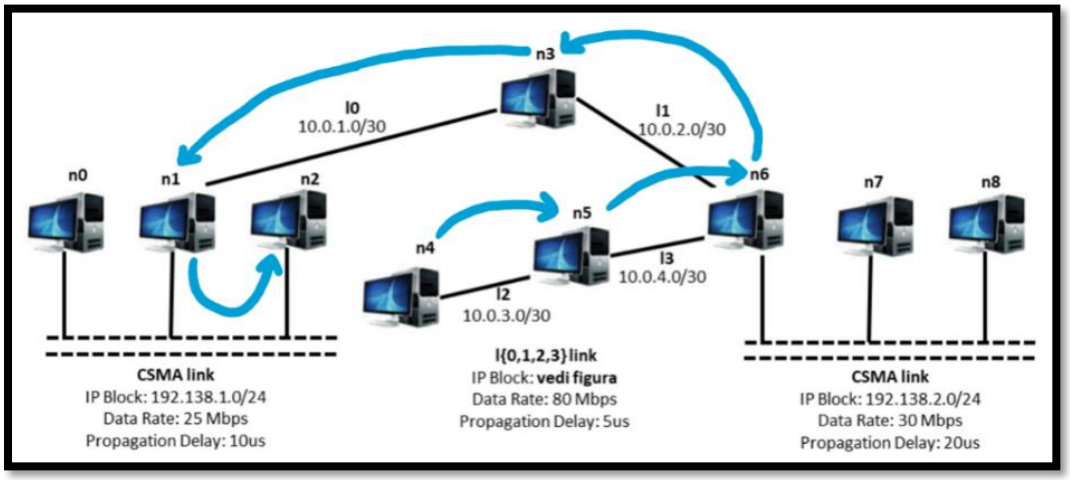


Figura 1: percorso pacchetti configurazione 0

Configurazione 1:

Abbiamo (figura 2):

- Per la comunicazione da n4 a n0 (c1) "***n4-n5-n6-n3-n1-n0***"
- Per la comunicazione da n8 a n2 (c2) "***n8-n6-n3-n1-n2***"

Infatti:

- task1-1-n6.pcap
  - Con il filtro "*ip.addr == 192.138.2.3*" ci sono i pacchetti di c1
  - Con il filtro "*ip.addr == 10.0.3.1*" non ci sono i pacchetti di c2
- task1-1-n5.pcap
  - Con il filtro "*ip.addr == 192.138.2.3*" non ci sono i pacchetti di c1
  - Con il filtro "*ip.addr == 10.0.3.1*" ci sono i pacchetti di c2
- task1-1-n3.pcap
  - Con i filtri precedenti distinguiamo c1 e c2

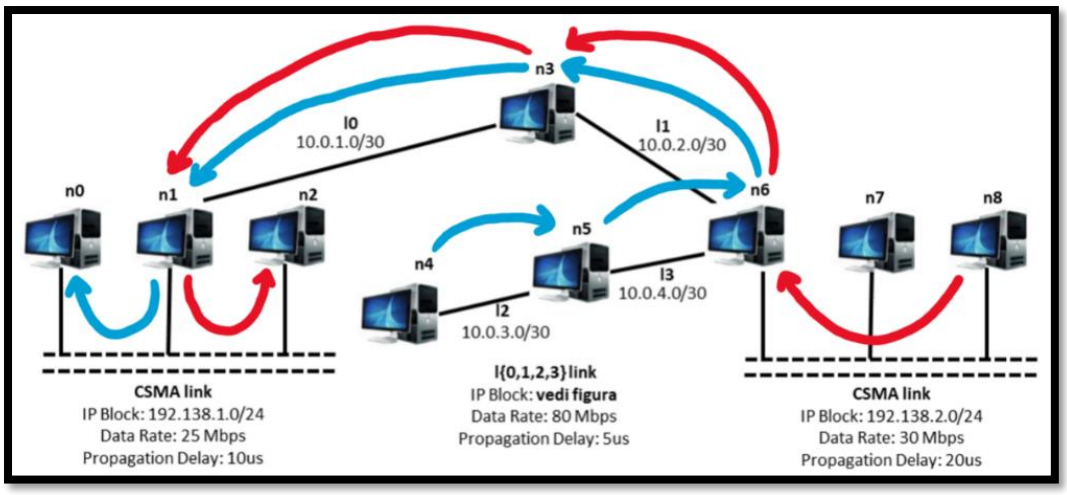


Figura 2: percorso pacchetti configurazione 1

Configurazione 2:

Abbiamo (figura 3):

- Per la comunicazione da n8 a n2 (c1) "***n8-n6-n3-n1-n2***"
- Per la comunicazione da n4 a n2 (c2) "***n4-n5-n6-n3-n1-n2***"
- Per la comunicazione da n7 a n0 (c3) "***n7-n6-n3-n1-n0***"

Infatti:

- task1-2-n6.pcap
  - filtro "ip.addr == 192.138.2.3" ci sono pacchetti di c1
  - filtro "ip.addr == 10.0.3.1" non ci sono pacchetti di c2
  - filtro "ip.addr == 192.138.2.2" ci sono pacchetti di c3
- task1-2-n5.pcap
  - filtro "ip.addr == 10.0.3.1" ci sono pacchetti di c2
  - filtro "ip.addr == 192.138.2.2 || ip.addr == 192.138.2.3" non ci pacchetti di c1 e c3
- task1-2-n3.pcap
  - con i filtri precedenti vediamo c1, c2 e c3

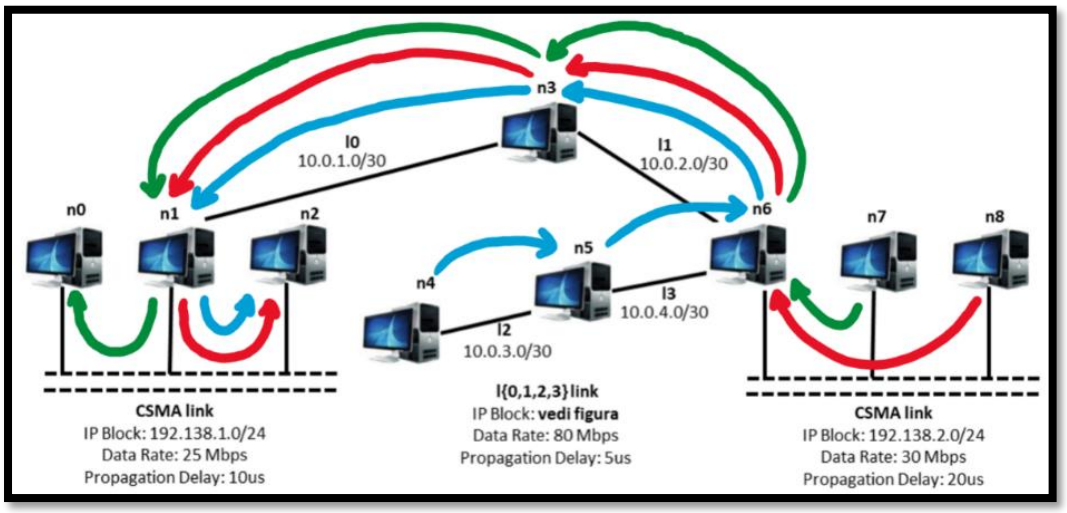


Figura 3: percorso pacchetti configurazione 2

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

Wireshark mette a disposizione i grafici dell'RTT (*Statistics -> TCP Stream Graphs -> Round trip time*).

Configurazione 0:

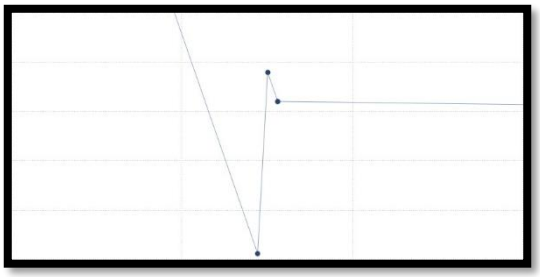


Figura 5: particolare del grafico di figura 4

Figura 4: grafico dell'RTT della connessione da n4 a n2 della configurazione 0

Alcuni pacchetti hanno RTT ~0,5ms ed altri ~24ms. Questo perché gli ACK vengono mandati ogni due pacchetti ma i pacchetti in ogni segmento sono dispari (3 in questo caso). Un pacchetto rimane in attesa finché non arriva il prossimo segmento, dopo ~23ms dall'arrivo del precedente.

Il pacchetto "avanzato" insieme ai prossimi 3 dà un numero pari, questo garantisce un RTT basso per tutti i pacchetti del nuovo segmento. Un pacchetto ogni 2 segmenti ha un RTT maggiorato.

Configurazione 1:

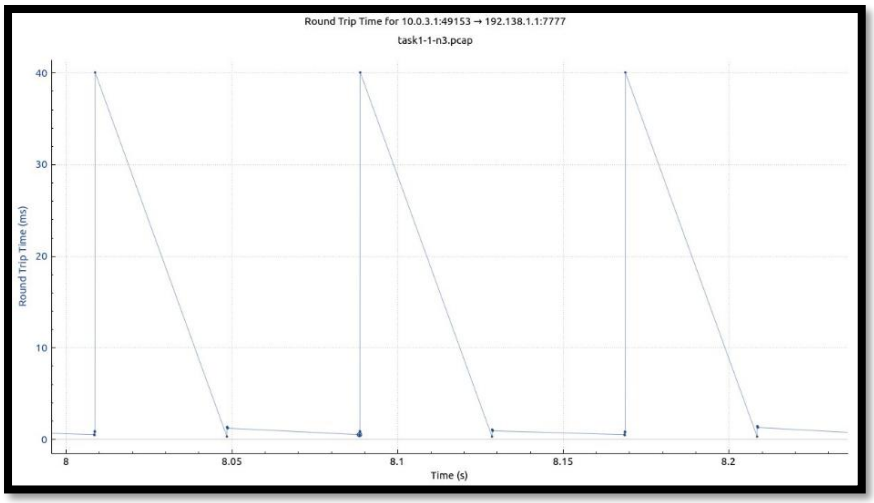


Figura 6: grafico dell'RTT da n4 a n0 della configurazione 1

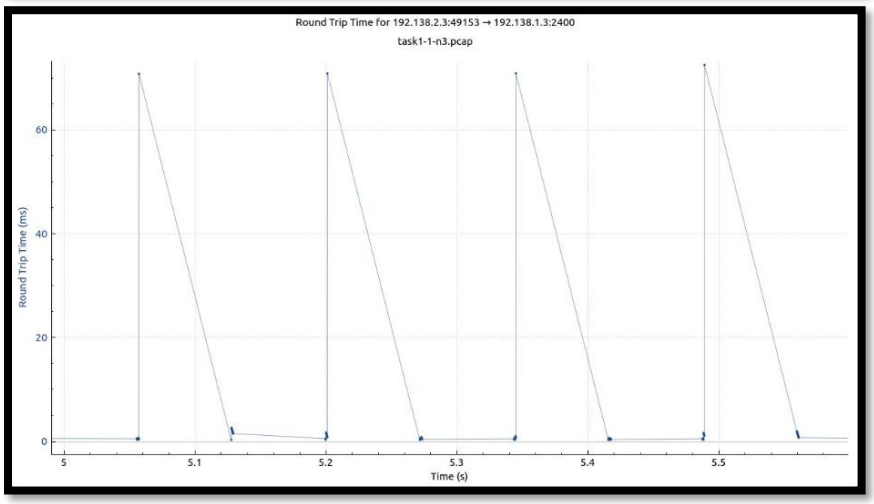


Figura 7: grafico dell'RTT da n8 a n2 della configurazione 1

Il comportamento è sostanzialmente lo stesso, qui i segmenti vengono divisi in 5 e 9 pacchetti. Notiamo che l'RTT maggiorato è proporzionale alla dimensione del segmento.

Configurazione 2:

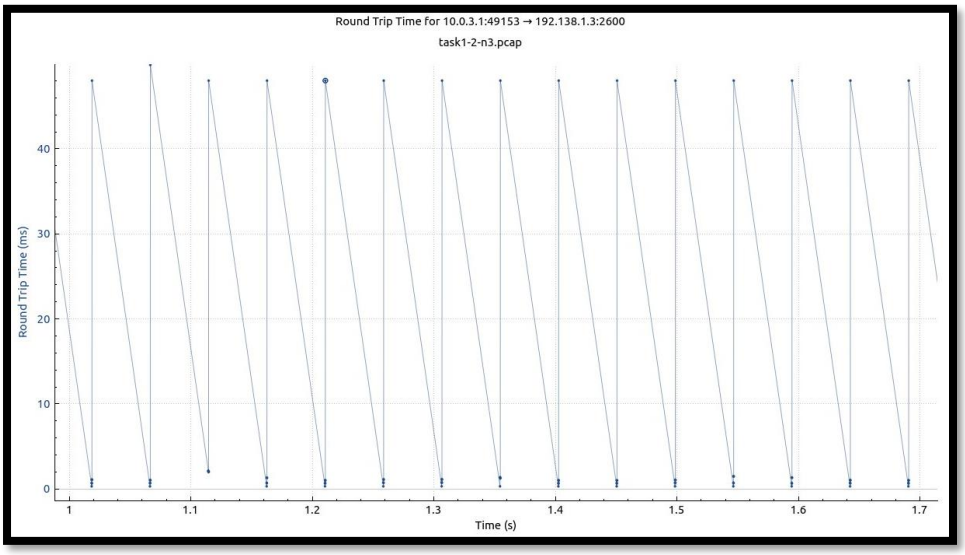


Figura 8: grafico dell'RTT da n4 a n2 della configurazione 2

Il grafico ha una forma diversa, infatti i segmenti vengono divisi in 6 pacchetti. Abbiamo un fenomeno simile al precedente: nel conteggio dei segmenti per inviare l'ACK viene contato anche l'ultimo dei 3 pacchetti del three-way-handshake.

In questo caso abbiamo un pacchetto con RTT maggiorato per ogni segmento.

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

Configurazione 0:

Abbiamo un singolo flusso. Dalla figura 1 notiamo che il collegamento con data-rate minimo lungo il percorso è csma1, che pone 25Mbps come limite superiore alla connessione. Ciò non influisce sul throughput effettivo della comunicazione in quanto il data-rate del client è molto inferiore (accade lo stesso nelle altre configurazioni).

Configurazione 1:

Abbiamo due flussi TCP e che condividono alcuni link (figura 2). TCP garantisce il controllo di congestione, ci aspettiamo nel caso di risorse limitate una suddivisione equa tra i flussi. Il bottleneck è anche in questo caso il csma1 in quanto è il collegamento con il data-rate minore ed è utilizzato da entrambe le comunicazioni.

Configurazione 2:

Abbiamo due flussi UDP ed uno TCP (figura 3). Dato che UDP non fornisce meccanismi di controllo di congestione nel caso la rete abbia un traffico elevato la comunicazione TCP verrebbe "strozzata" dalle altre due.

Il bottleneck continua ad essere csma1 per gli stessi motivi della configurazione precedente.

C01) Calcolare il throughput istantaneo del flusso TCP.

Pacchetto 1 (no. 49, 50, 51):

- n4 invia a n2 1500 byte divisi in 3 frame ognuno con 54 byte di header
- Inizio trasmissione: 3,27428s
- Fine trasmissione: 3,27524s
- Intervallo: 0,00096 s
- Throughput istantaneo (totale): 1662 byte / 0,00096s = **13.85 Mbps**
- Throughput istantaneo (dati): 1500 byte / 0,00096s = **12.5 Mbps**

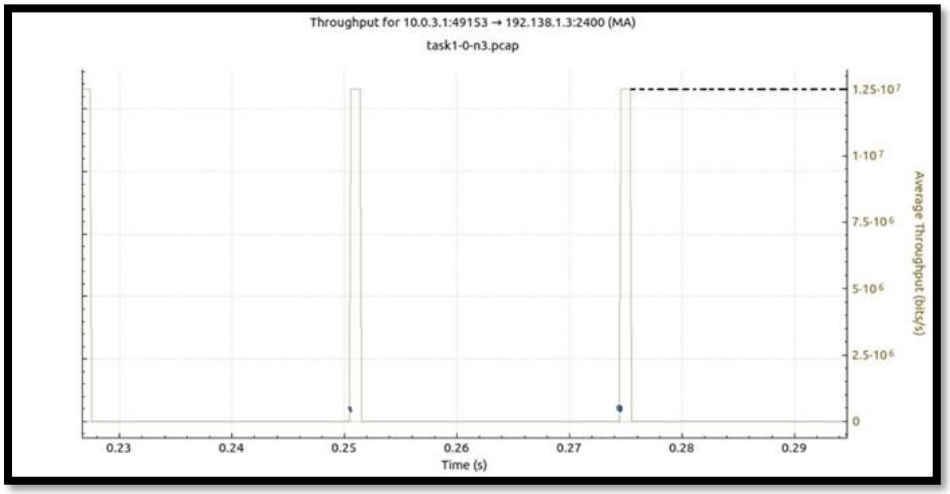


Figura 9: throughput istantaneo del pacchetto 1. MA window = 0,00096s

Pacchetto 2 (no. 1287, 1288, 1289):

- n4 invia n2 1500 byte divisi in 3 frame ognuno con 54 byte di header
- Inizio trasmissione: 9,87429s
- Fine trasmissione: 9,87513s
- Intervallo: 0.00084s
- Throughput istantaneo (totale): 1662 byte / 0,00084s = **15.83 Mbps**
- Throughput istantaneo (dati): 1500 byte / 0,00084s = **14.2 Mbps**

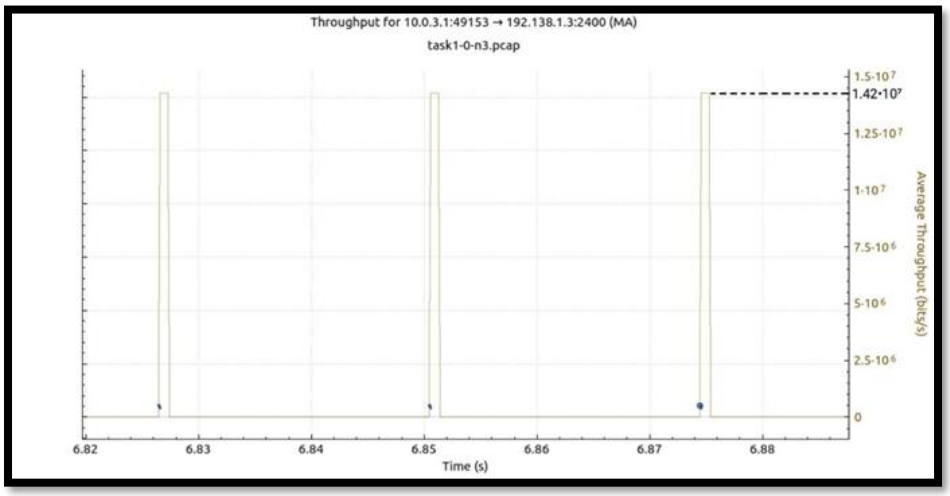


Figura 10: throughput istantaneo del pacchetto 2. MA window = 0,00084s

Notiamo (figure 9 e 10) che il throughput istantaneo ha dei picchi negli istanti della trasmissione dei pacchetti mentre è nullo in ogni altro momento.

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

Consideriamo l'intervallo da 3.0s a 4.0s. Il datarate del client è di 500 kbps (valore di default) di messaggi a livello applicativo. Il collegamento con datarate minimo nel percorso ha 25Mbps (csma1). Ci aspettiamo 500 kbps di throughput (più header)

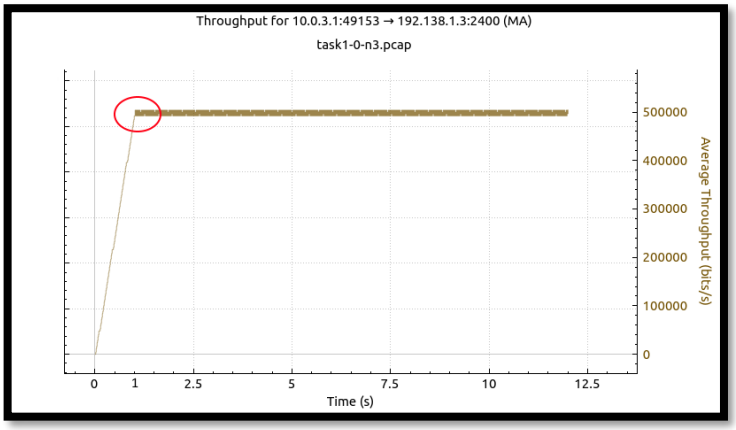
Analizziamo con Wireshark (task1-0-n3.pcap)

- Con "*frame.time_relative < 1*" selezioniamo i pacchetti da 3.0s a 4.0s
- Usando *Statistics -> Conversations* nella sezione *TCP·1* abbiamo:

Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
10.0.3.1	49153	192.138.1.3	2400	188	71660	125	68254	63	3406	0	0,995185	548.673,864658329	27.379,8339002296

- Totale trasferito: 71.660 byte (573.280 bit)
- Throughput medio: **573.280 bit/s** (poiché si chiede t=4.0s consideriamo 1s e non 0,995s)

Dal logging del programma vediamo che dei 71.660 byte, 61.500 byte (492.000 bit) sono di messaggi, ottenendo quindi **492.000 bit/s** di throughput considerando solo i messaggi a livello applicativo (come nei grafici Wireshark, figure 11 e 12)



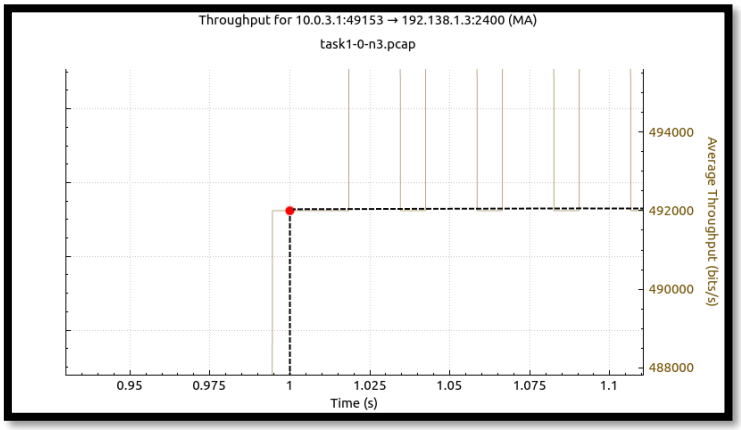


Figure 11 e 12: Throughput medio, finestre di 1s, configuration 0, visto da n3. t=0s nel grafico equivale a t=3.0s nella simulazione.

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

Consideriamo l'intervallo da 3.0s a 7.0s, con le stesse assunzioni di prima.

Analizziamo con Wireshark (task1-0-n3.pcap)

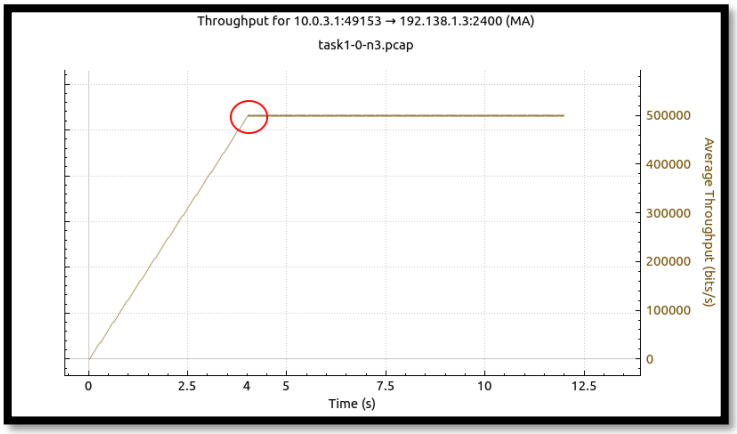
- Con "*frame.time_relative < 4*" selezioniamo i pacchetti da 3.0s a 7.0s
- Usando *Statistics -> Conversations* nella sezione *TCP·1* abbiamo:

Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
10.0.3.1	49153	192.138.1.3	2400	750	289508	500	276004	250	13504	0	3,994963	552.703,992502558	27.042,0527048686

- Totale trasferito: 289.508 byte (2.316.064 bit)
- Throughput medio: 2.316.064 / 4 = **579.016 bit/s**

Dal logging del programma vediamo che dei 289.508 byte, 249.000 byte (1.992.000 bit) sono di messaggi, ottenendo quindi **498.000 bit/s** di throughput considerando solo i messaggi a livello applicativo (come nei grafici Wireshark, figure 13 e 14).

Il throughput medio è leggermente maggiore quello di C02. Infatti, mentre da 3.0s a 4.0s il 3-way-handshake abbassa il throughput medio, da 4.0s a 7.0s si lavora a regime. Notiamo che 1.992.000 – 492.000 = 1.500.000 bit, ovvero un throughput medio di 500.000 bit/s (dei messaggi) che è esattamente il massimo possibile per il client.



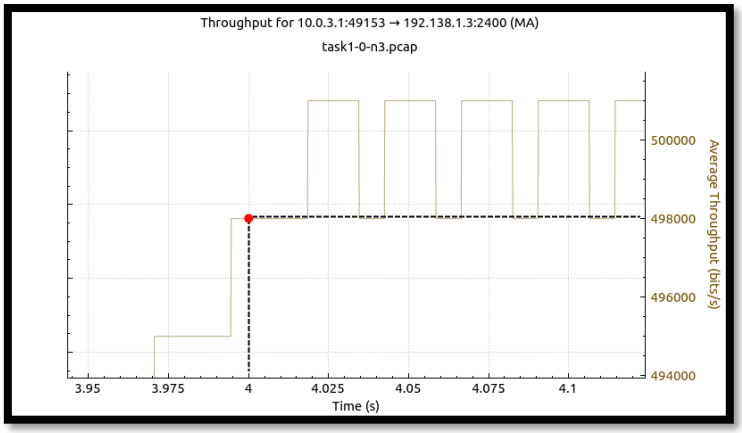


Figure 13 e 14: Throughput medio, finestre di 4s, configuration 0, visto da n3. t=0s nel grafico equivale a t=3.0s nella simulazione.

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

Calcoliamo il ritardo di trasferimento (Rt) come la differenza tra il tempo di ricezione dell'ultimo pacchetto (Tf) e quello di invio del primo (Ti).

Il primo è il SYN che n4 manda a n2 per iniziare la comunicazione (three-way-handshake) mentre l'ultimo è il pacchetto che n4 manda a n2 per confermare la ricezione del pacchetto FIN che questi gli ha inviato.

Supponiamo il primo venga inviato all'inizio della comunicazione, Ti = 3s.

Tf possiamo stimarlo: con Wireshark vediamo che il pacchetto passa da n3 in t = 15,00018s, dobbiamo però considerare quando arriva a n2.

Sapendo che il primo ACK viene mandato subito dopo l'arrivo del primo pacchetto informativo leggiamo nei log quando questi arriva a n2 (3.03474s) e lo sottraiamo al tempo in cui l'ACK raggiunge n3 (3,034783s), ottenendo 0,000043s, che è il tempo che impiega un ACK a passare da n2 a n3 o viceversa.

Tf = 15,00018 + 0,000043 = 15,000223s

Da cui:

Rt = Tf – Ti = 15,000223 – 3 = **12,000223s**

C11) Calcolare il throughput medio dei flussi TCP.

Il datarate dei client è di 500 kbps ciascuno. Il collegamento con datarate minimo lungo il percorso ha 25Mbps (csma1), ci aspettiamo 500 kbps di throughput ciascuno (più header), senza che uno influisca sul throughput dell'altro data la larga banda disponibile.

Analizziamo il flusso n4->n0 (task1-1-n3.pcap)

- Con "*ip.addr == 10.0.3.1*" ne selezioniamo i pacchetti
- Usando *Statistics -> Conversations* nella sezione *TCP·1* abbiamo:

Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
10.0.3.1	49153	192.138.1.1	7777	1874	723704	1249	689950	625	33754	2,997906	10,000148	551.951,831112899	27.002,8003585546

- Totale trasferito: 723.704 byte (5.789.632 bit).
- Throughput medio (totale): 5.789.632 / 10 = **578.963 bit/s**
- Throughput messaggi (dal log): 622.500 * 8 / 10 = **498.000 bit/s**

Analizziamo il flusso n8->n2 (task1-1-n3.pcap)

- Con "*ip.addr == 192.138.2.3*" ne selezioniamo i pacchetti
- Usando *Statistics -> Conversations* nella sezione *TCP·1* abbiamo:

Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
192.138.2.3	49153	192.138.1.3	2400	1302	502316	868	478876	434	23440	0	6,998108	547.434,820954463	26.795,8139542859

- Totale trasferito: 502.316 byte (4.018.528 bit).
- Throughput medio (totale): 4.018.528 / 7 = **574.075 bit/s**
- Throughput messaggi (dal log): 432.000 * 8 / 7 = **493.714 bit/s**

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

Consideriamo il flusso n8->n2 da 2.0s a 6.0s. Le assunzioni sono analoghe alle precedenti.

Analizziamo con Wireshark (task1-1-n3.pcap)

- Con "*ip.addr == 192.138.2.3 && frame.time_relative < 4*" ne selezioniamo i pacchetti
- Usando *Statistics -> Conversations* nella sezione *TCP·1* abbiamo:

Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
192.138.2.3	49153	192.138.1.3	2400	746	287792	497	274342	249	13450	0	3,977668	551.764,501210256	27.051,0258774739

- Totale trasferito: 287.792 byte (2.302.336 bit).
- Throughput medio (totale): 2.302.336 / 4 = **575.584 bit/s**
- Throughput messaggi (dal log): 247.500 * 8 / 4 = **495.000 bit/s**

Il flusso è l'unico nella rete da 2.0s a 5.0s mentre da 5.0s a 6.0s condivide i canali l1, l0 e csma1 con il flusso n4->n0. Ciò non influenza significativamente il throughput perché il collegamento con datarate minore è da 25Mbps mentre i due canali usano complessivamente meno di 1,2Mbps.

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

Consideriamo il flusso n8->n2 da 2.0s a 8.0s. Le assunzioni sono analoghe alle precedenti.

Analizziamo con Wireshark (task1-1-n3.pcap)

- Con "*ip.addr == 192.138.2.3 && frame.time_relative < 6*" ne selezioniamo i pacchetti
- Usando *Statistics -> Conversations* nella sezione *TCP·1* abbiamo:

Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
192.138.2.3	49153	192.138.1.3	2400	1124	434204	749	413950	375	20254	0	5,993682	552.515,131767084	27.033,7999246539

- Totale trasferito: 434.204 byte (3.473.632 bit).
- Throughput medio (totale): 3.473.632 / 6 = **578.939 bit/s**
- Throughput messaggi (dal log): 373.500 * 8 / 6 = **498.000 bit/s**

Il throughput medio risulta leggermente maggiore per lo stesso motivo descritto in C03.

Usando il logging vediamo:

- Throughput medio a regime con n8->n2 con l'unico flusso (da 3.0s a 5.0s): (184500 - 58500) * 8 / 2 = 504.000 bit/s
- Throughput medio a regime con entrambi i flussi (da 5.0s a 8.0s): (373500 - 184500) * 8 / 3 = 504.000 bit/s

Sembra che la presenza del flusso n4->n0 non influenzi significativamente il throughput.

C14) [Extra] Ritardo di accodamento vs congestione: Disegnare un grafico che mostri il ritardo di accodamento in funzione del livello di congestione in rete

Definiamo ritardo di accodamento quel tempo che un pacchetto impiega in un elemento di rete aspettando che i pacchetti arrivati precedentemente vengano elaborati e trasmessi. In generale è un valore non prevedibile, ma possiamo calcolarne il valore medio in funzione dell'intensità di traffico.

Definiamo invece il livello di congestione (o intensità di traffico) come La/R, con:

- L: lunghezza di un pacchetto (bit/pacchetto)
- a: tasso medio di arrivo dei pacchetti (pacchetto/secondo)
- R: velocità di trasmissione in uscita (bit/s)

Tanto più La/R si avvicina ad 1 (ovvero il numero di bit in entrata ed uscita è simile) tanto più il ritardo di accodamento medio è elevato. La curva segue un andamento iperbolico con l'asintoto in La/R = 1.

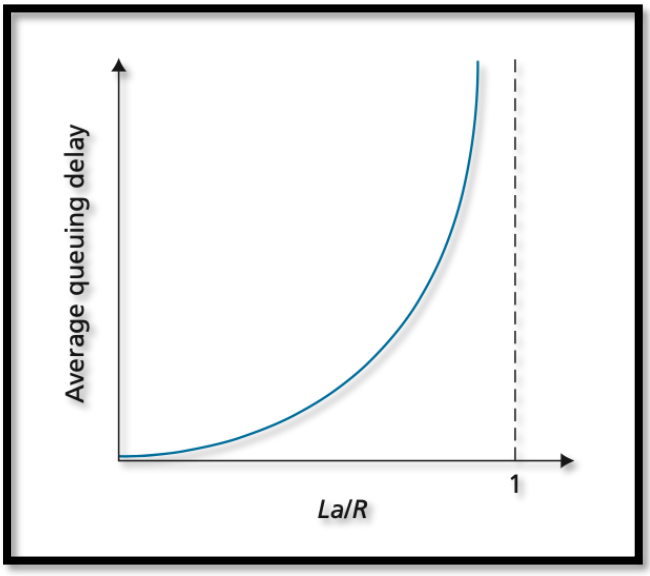


Figura 15: grafico rapporto ritardo di accodamento e livello di congestione

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

Consideriamo il flusso n4->n2 da 3.0s a 5.0s. Il datarate dei client è di 500 kbps ciascuno. Il collegamento con datarate minimo lungo il percorso ha 25Mbps (csma1). L'unico altro flusso presente è n8->n2 che invia pacchetti ogni 2 secondi a partire da 3.0s.

Analizziamo con Wireshark (task1-2-n3.pcap)

- Con "*ip.addr == 10.0.3.1 && frame.time_relative < 2*" selezioniamo i pacchetti d'interesse
- Usando *Statistics -> Conversations* nella sezione *TCP·1* abbiamo:

Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
10.0.3.1	49153	192.138.1.3	2600	372	143096	248	136396	124	6700	0	1,979647	551.193,217780745	27.075,5341735167

- Totale trasferito: 143.096 byte (1.144.768 bit).
- Throughput medio (totale): 1.144.768 / 2 = **572.384 bit/s**
- Throughput messaggi (dal log): 123.000 * 8 / 2 = **492.000 bit/s**

Il percorso del flusso è analogo a quello del flusso della configurazione 0, il rapporto payload/header (1500:162 = 3000:324) è lo stesso ed infatti, analizzando i log, vediamo che i throughput medi dopo stessi intervalli di tempo sono uguali.

Possiamo dedurre che il flusso n8->n2 non influenza il throughput del flusso considerato.

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

Consideriamo il flusso n4->n2 da 3.0s a 7.0s. Le assunzioni sono analoghe alle precedenti, ma prendiamo in considerazione anche il flusso n7->n0 che inizia a 5.0s.

Analizziamo con Wireshark (task1-2-n3.pcap)

- Con "*ip.addr == 10.0.3.1 && frame.time_relative < 4*" selezioniamo i pacchetti d'interesse
- Usando *Statistics -> Conversations* nella sezione *TCP·1* abbiamo:

Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
10.0.3.1	49153	192.138.1.3	2600	750	289508	500	276004	250	13504	0	3,996372	552.509,125777079	27.032,5184942743

- Totale trasferito: 289.508 byte (2.316.064 bit).
- Throughput medio (totale): 2.316.064 / 4 = **579.016 bit/s**
- Throughput messaggi (dal log): 249.000 * 8 / 4 = **498.000 bit/s**

Come nella risposta C21 notiamo l'analogia con il flusso della configurazione 0 e l'uguaglianza dei log negli stessi intervalli di tempo.

Possiamo dedurre che i flussi n8->n2 e n7->n0 non influenzano il throughput del flusso TCP.