Introduzione

In questo documento viene assunto che gli istanti di tempo da considerare su richiesta della traccia vadano modificati prima di essere usati su Wireshark. Il programma infatti considera $t_0 = 0s$ come l'istante in cui avviene la ricezione del primo pacchetto e non l'inizio della simulazione, che chiameremo tempo **globale**. Nelle domande verranno comunque specificati i calcoli necessari per questa trasformazione. Nella trattazione delle domande, il punto **A1** è stato trattato una singola volta, per evitare delle ripetizioni all'interno del documento.

Analisi delle configurazioni

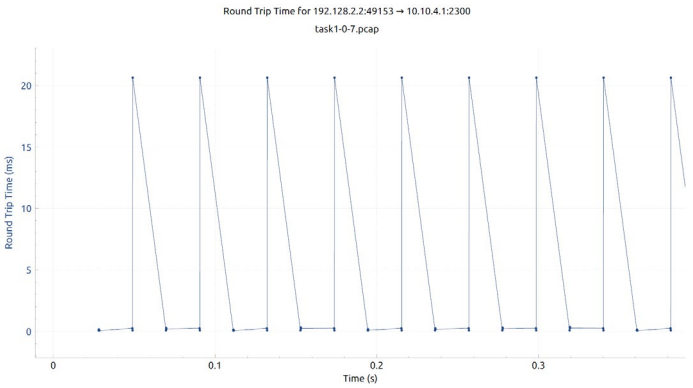
A1) Nella rete analizzata abbiamo riconosciuto essere composta da due connessioni CSMA, ed una unione tra una connessione di tipo stella basata su connessioni Point-To-Point, ed una connessione di tipo ad anello basata su Point-To-Point, che unisce i nodi esterni della connessione a stella. La prima connessione CSMA riguarda i nodi n0, n1, n2; la seconda i nodi n7, n8, n9. La connessione a stella ha come nodo centrale "hub" il nodo n5 ed i seguenti nodi esterni n3, n4, n6, n7. La connessione ad anello comprende i nodi n3, n4, n6 e n7. I nodi n2 e n3 sono connessi tramite una Point-To-Point.

Configurazione 0

La **configurazione 0** richiede di mandare dei pacchetti TCP dal nodo n9 al nodo n5 quindi c'è un unico stream di comunicazione. Data la richiesta di inviare pacchetti con dimensione pari a 1'300bytes, i messaggi vengono divisi in tre pacchetti dal programma Wireshark, con rispettive dimensioni pari a 536bytes, 536bytes e 228bytes. A questi vanno aggiunti 70bytes di header per ogni pacchetto.

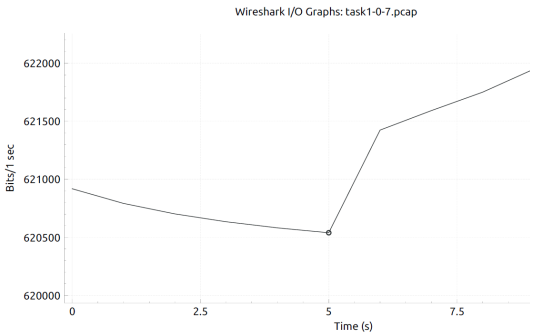
A2) Utilizzando i file di ASCII tracing generati dal codice, abbiamo potuto seguire il percorso intrapreso dai pacchetti. Questi partono da n9, con IP 192.128.2.2, passano per il nodo n7, con IP 192.128.2.3, e si fermano al nodo n5, con IP 10.10.4.1, come desiderato. Importante sottolineare che i pcap file generati sono vuoti, tranne per quello che corrisponde al nodo 7 ("task1-0-7.pcap"), effettivamente toccato dal percorso dei pacchetti.

A3) Abbiamo calcolato il **RTT** (Round Trip Time) facendo la differenza tra il tempo di arrivo di un pacchetto ACK e il tempo di partenza del pacchetto corrispondente. Il risultato calcolato, pari a 20.6 millisecondi, corrisponde al grafico del RTT ottenuto grazie agli strumenti di analisi del programma Wireshark. I picchi del grafico, costanti per l'intera durata della simulazione, corrispondono al momento in cui viene eseguito il campionamento; inoltre, i bytes trasmessi in ogni intervallo sono 3200bytes.



A4) In questa configurazione è presente un bottleneck (collo di bottiglia) nella connessione CSMA tra i nodi n7, n8 e n9, dato che la velocità di connessione è più lenta (30Mbps) rispetto a quella della connessione star (80Mbps).

C01) Per calcolare il throughput istantaneo abbiamo preso in considerazione un frame casuale (composto da 4 pacchetti divisi a causa dell'MMS). Considerando la differenza di tempo tra il pacchetto che lo precede e quello che lo succede, abbiamo così calcolato il throughput in un istante di tempo



$$\frac{1580bytes}{0{,}02035s} = 77'\,641bytes/s = 621'\,130bit/s$$

Il risultato corrisponde al grafico del throughput istantaneo che abbiamo ricavato grazie alle funzioni statistiche di Wireshark.

C02) Il throughput medio del flusso TCP al tempo $t = 4s$ è:

$$throughput\ medio = n\ bit\ trasmessi/tempo\ totale$$

Leggendo il numero di byte ricevuti in Wireshark al secondo relativo $t = 4s - 3{,}015s = 0{,}985s$. In questo intervallo sono stati inviati 76'010bytes; di conseguenza, il throughput medio al secondo 4 è $77'\,167bytes/s$, che trasformato in bit risulta

$$617'\,340bit/s \sim 617kbit/s$$

C03) Seguendo lo stesso procedimento per il tempo globale $t = 7s$, che corrisponde al tempo relativo $t = 7s - 3{,}02s = 3{,}980s$, possiamo vedere che i byte scambiati sono 308'570bytes. Di conseguenza il throughput medio è $77'\,530bytes/s$, che trasformato in bit risulta

$$620'\,241bit/s = 620kbit/s$$

Possiamo notare come il throughput è aumentato di 3kbit/s con il passare del tempo.

C04) Per calcolare il ritardo totale della simulazione abbiamo utilizzato il parametro "EpochTime" su Wireshark che ci permette di vedere il tempo effettivo in cui è ricevuto un pacchetto. (L'inizio della simulazione viene considerato come il 01/01/1970 01:00:00). Sottraendo il tempo del primo pacchetto inviato, con flag SYN, a quello dell'ultimo otteniamo la durata effettiva della simulazione. Possiamo ottenere così il ritardo dividendo per la durata prevista dalla specifica.

$$\frac{15{,}000083 - 3{,}006055}{12} = \frac{11{,}994028}{12} = 0{,}9995$$

Il ritardo è quindi pari a 0,9995s. Dato che i pacchetti inviati sono 2'602, possiamo ottenere il ritardo medio per pacchetto, che corrisponde a $3{,}84 * 10^{-4}s$.

Configurazione 1

La configurazione 1 ha due stream TCP di comunicazione. Il primo manda dei pacchetti di dimensione 2'500bytes dal nodo n9 al nodo n5, il secondo invia pacchetti di 5'000bytes dal nodo n8 al nodo n0. Come per la configurazione precedente, la grandezza dell'header è pari a 70bytes.

A2) Usando i file di ASCII tracing generati dal codice il percorso dei pacchetti è:

- **Stream 1:** Questi partono da n9, con IP 192.128.2.2, passano per il nodo n7, con IP 192.128.2.3, e si fermano al nodo n5, con IP 10.10.4.1.

- **Stream 2:** I pacchetti partono da n8 con IP 192.128.2.1, passano per il nodo n7, con IP 192.128.2.3, passano per n5 con IP 10.10.4.1, poi in n3 con IP 10.1.1.2. A questo punto si muovono in n2 IP 192.128.1.3 e raggiungono così n0 con IP 192.128.1.1.

A3) Utilizzando un procedimento analogo a quello usato nella configurazione 0, abbiamo calcolato gli RTT corrispondenti a ciascuno degli stream per ognuno dei nodi associati a un pcap file. I calcoli sono stati riportati in questa tabella e corrispondono con quanto ottenuto dai grafici su Wireshark. Anche in questo caso i valori dei picchi RTT sono costanti per tutta la durata della trasmissione.

	RTT
N7 stream 1	39,5ms
N7 stream 2	79,0ms
N3 stream 2	78,9ms
N0 stream 2	78,2ms

A4) Come per la configurazione precedente, è presente un rallentamento causato dalle differenze di "datarate" tra le varie connessioni. In particolare, entrambe le connessioni CSMA sono particolarmente più lente di quelle Point To Point. Inoltre, il nodo n7, rappresenta un punto critico della configurazione in quanto questo è condiviso tra più tipologie di connessione. Soprattutto in questa configurazione l'alto numero di stream di dati può causare un ritardo di accodamento notevole che potrebbe portare, nei casi peggiori, ad una perdita di pacchetti.

C11) Per poter calcolare i due throughput abbiamo separato i due stream usando i seguenti filtri nel file pcap:

- "ip.src == 10.10.4.1 || ip.src == 192.128.2.2" per lo **stream 1**.
- "ip.src == 192.128.1.1 || ip.src == 192.128.2.1" per lo **stream 2**.

Throughput medio del primo stream

Per questo stream, l'intervallo di tempo da considerare è di $t = 10s$ poiché la comunicazione inizia al tempo globale **5s** e termina al tempo globale **15s**. Su Wireshark questi vengono registrati come $t_0 = 2{,}99s$ e $t_1 = 12{,}99s$. Dato che in questo intervallo il numero di byte trasmessi è 753'540bytes, il throughput del primo stream risulta

$$\frac{753540bytes}{10s} = 75'\,354bytes/s$$

Che corrisponde a $602'\,832bit/s = 603kbit/s$.

Throughput medio del secondo stream

Per il secondo stream l'intervallo di tempo da considerare è di 7s; per Wireshark sono a $t_0 = 0s$ e $t_1 = 6{,}99s$. Il numero di byte scambiati è 526'570bytes, per un throughput medio di

$$\frac{526'570bytes}{6{,}99s} = 75'\,235bytes/s = 601'\,880bit/s$$

C12) Per ottenere il flusso dati desiderato secondo le condizioni chieste, abbiamo utilizzato il seguente filtro sul file pcap del nodo n7: "(ip.src == 10.10.4.1 || ip.src == 192.128.2.2) && frame.time_relative < 4". Abbiamo così selezionato i pacchetti relativi al primo flusso ricevuto

prima del tempo globale $t = 6s$, pari al tempo relativo $t = 4s$. I bytes trasmessi in questo periodo sono 296'520bytes, per un throughput medio di $74'\,136 bytes/s$. In bit equivale a

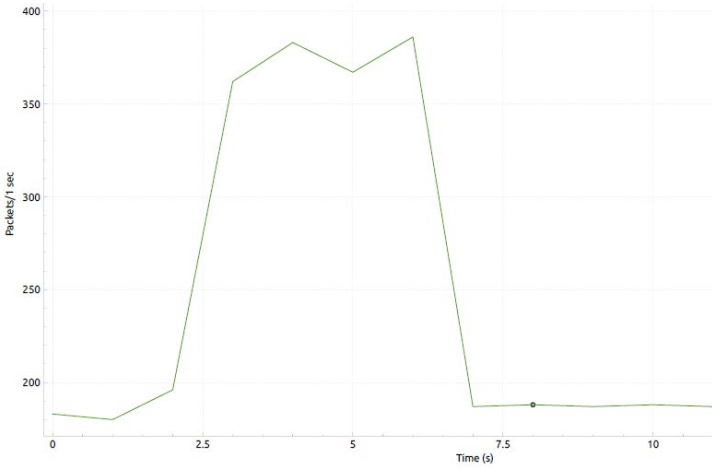
$$593'\,040 bit/s \sim 593 kbit/s$$

C13) Analogamente a quanto visto nel punto precedente, abbiamo modificato la condizione relativa al "frame.time_relative" per selezionare i pacchetti precedenti al tempo globale $t = 8s$, corrispondente al tempo relativo $t = 6s$. I byte trasmessi in questo intervallo sono 447'770bytes, per un throughput medio di $74'\,628 bytes/s$. In bit equivale a

$$597'\,026 bit/s \sim 597 kbit/s$$

Risulta importante sottolineare che c'è stato un aumento di throughput di circa $4 kbit/s$.

C14) Per ritardo di accodamento intendiamo il tempo medio impiegato da un pacchetto per lasciare la coda di un nodo. In un dato istante, il primo pacchetto subirà un ritardo di accodamento pari a 0, mentre l'ultimo dovrà aspettare che tutti i pacchetti nella coda siano inviati. Questo ritardo è direttamente collegato all'intensità di traffico, che descrive la quantità di pacchetti che sono in uscita da un determinato nodo in un istante di tempo:



$$La/R$$

"R" corrisponde alla velocità di trasmissione, "a" è la velocità media di arrivo dei pacchetti in un nodo e "L" è la dimensione media di ogni pacchetto. Il grafico qui a destra rappresenta la quantità di pacchetti in entrata nel nodo 7 in funzione del tempo. Si può quindi calcolare l'intensità di traffico in un determinato istante in modo da stimare il ritardo di accodamento. È importante sottolineare che, se l'intensità di traffico supera 1, le code dei nodi cresceranno sempre più fino a causare perdita di pacchetti.

Configurazione 2

La configurazione 2 è composta da tre stream di comunicazione. Il primo è uno stream UdpEcho che manda 5 pacchetti con intervalli di 2 secondi e dimensione 2560bytes dal nodo **n8** al nodo **n2**. Il secondo è uno stream UDP SinkOnOff che manda pacchetti di dimensione 3000bytes dal nodo **n8** al nodo **n0** e l'ultimo è uno stream TCP SinkOnOff che invia pacchetti di 3000 bytes dal nodo **n9** al nodo **n5**.

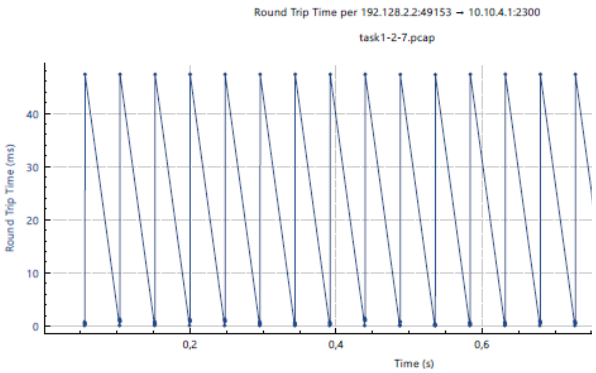
A2) Usando i file di ASCII tracing generati dal codice, il percorso dei pacchetti è il seguente:

Stream 1. I pacchetti partono dal nodo n8 con IP 192.128.2.1, passano per n7 con IP 192.128.2.3, attraversano n5 con IP 10.10.4.1, passano in n3 con IP 10.1.1.2 ed arrivano a n2, con IP 192.128.1.3.

Stream 2. I pacchetti partono dal nodo n8 con IP 192.128.2.1, passano per n7 con IP 192.128.2.3, attraversano n5 con IP 10.10.4.1, passano in n3 con IP 10.1.1.2, attraversano n2 con IP 192.128.1.3 e raggiungono n0 con IP 192.128.1.1.

Stream 3. I pacchetti partono dal nodo n9 con IP 192.128.2.2, passano per n7 con IP 192.128.2.3 e si fermano al nodo n5 con IP 10.10.4.1.

A3) Usiamo un procedimento analogo a quello della configurazione precedente per calcolare il Round Trip Time, ricordando che può essere calcolato solo per i pacchetti che usano il protocollo TCP, ovvero solo per lo stream 3. Il risultato è 47,3ms che corrisponde a quanto ottenuto dal grafico di Wireshark.



A4) Come per le configurazioni precedenti, è presente un rallentamento causato dalle differenze di "datarate" tra le varie connessioni. In particolare, entrambe le connessioni CSMA sono particolarmente più lente di quelle Point To Point. Similmente alla configurazione precedente, sia il nodo n7 che il nodo n2 sono punti critici del sistema. In particolare, si può venire a creare una coda ingestibile nel nodo n2 nel caso delle connessioni che hanno come destinazione i nodi n0 e n1. Questo avviene poiché la connessione CSMA che li collega è decisamente più lenta rispetto alla connessione Point To Point che collega n2 e n3.

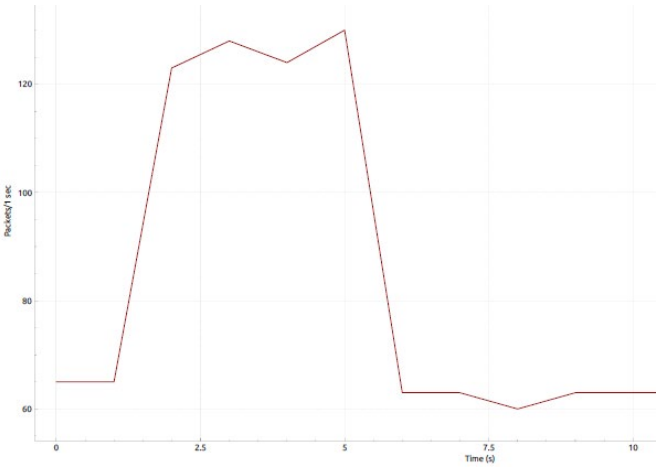
C21) Calcoliamo il throughput medio a tempo globale $t = 5s$, che corrisponde al tempo relativo $t = 2s$, dell'unico flusso TCP presente in questa configurazione. Usando lo stesso metodo usato per i punti precedenti, abbiamo applicato il seguente filtro su Wireshark in modo da prendere in considerazione solo i pacchetti desiderati "(ip.src == 192.128.2.2 || ip.src ==10.10.4.1) && frame.time_relative < 2". In questa configurazione risultano 41 pacchetti da 390bytes, 124 da 70bytes e 205 da 606bytes, per un totale di 25'900bytes. Il throughput medio risulta quindi $12'\,950 bytes/s$, ovvero:

$$103'\,600 bit/s \sim 104 kbit/s$$

C22) Per calcolare il tempo globale a $t = 7s$, pari al tempo relativo $t = 4s$, dello stream SinkOnOff abbiamo usato lo stesso metodo del punto precedente. Applicando i filtri di su Wireshark troviamo la presenza di 83 pacchetti da 390bytes, 250 da 70bytes e 415 da 606bytes, per un totale di 301'360bytes. Di conseguenza, il throughput medio risulta quindi $75'\,340 bytes/s$, ovvero:

$$602'\,720 bit/s \sim 603 kbit/s$$

Notiamo quindi un importante aumento nel throughput medio. Infatti il throughput al secondo 7 risulta essere quasi 6 volte maggiore di quello all'istante 5. La loro differenza è $499'\,120 bit/s$.



C23) Come per la configurazione precedente, mostriamo nel grafico a destra il numero di pacchetti in arrivo sul nodo n7 in funzione del tempo. È possibile notare che la frequenza di pacchetti è molto minore rispetto a quella della configurazione precedente. Possiamo spiegare questo risultato, notando che il primo stream UDPEcho manda solo 5 pacchetti ogni 2 secondi. Inoltre, il protocollo TCP ha integrato un sistema di controllo della congestione, che si assicura di ridurre il numero di pacchetti inviati per evitare malfunzionamenti (ovvero perdita di pacchetti) nella rete.

Team 25

Lavoro compiuto dal Team 25, composto da:

- Simone Federico Laganà – 1946083
- Filippo Guerra – 1931976
- Giulio Di Gregorio – 1943235
- Marika Fuccio – 1962183