Anno Accademico 2021/2022

Laboratorio di Internet

Politecnico di Torino

Report III

Misure di goodput

Gruppo 14

Andreas Brummer (s270332)

Alessandro Ciullo (s269589)

Andrea Scamporrino (s270971)

1 Configurazione di rete

Nell'effettuare i test é stata usata la seguente configurazione di rete:

Indirizzo di rete	172.16.14.0
Indirizzo di broadcast	172.16.14.63
NetMask	255.255.255.192

Nome host	Indirizzo IP
H1	172.16.14.1/26
H2	172.16.14.2/26
H3	172.16.14.3/26

Tabella 1.1: Configurazione di rete

I vari host erano collegati tramite uno switch e su tutte le linee la velocità impostata era di 10 Mb/s. Per effettuare le misure di goodput è stato usato nttcp. Per usare il programma in questione basta lanciare l'eseguibile sulla macchina locale e su una macchina partner. Sulla macchina partner(server) nttcp deve essere avviato con l'opzione -i. Sull'host locale è sufficiente digitare nttcp seguito dall'indirizzo IP del server. Per impostazione predefinita, il programma trasferisce 2048 buffer di 4KByte di lunghezza (per un totale di 8 MByte) all'host partner. Su entrambi i lati verranno misurate le prestazioni ed i risultati (sia remoti che locali) saranno riportati sul lato locale. Eventualmente è possibile modificare quasi tutti i parametri della trasmissione tramite le opzioni da riga di comando.

2 Per ogni scenario

Quando gli esperimenti vengono effettuati i tempi vengono misurati sia a lato ricevitore che trasmettitore. Nel caso del trasmettitore avremo una sovrastima mentre nel caso del ricevitore la misura sarà più accurata. Inoltre nel caso di una connessione TCP prima di inviare i dati viene aperta una connessione. Per rendere il tempo di apertura di una connessione trascurabile rispetto ai tempi in gioco nell'esperimento i parametri -l e -n sono stati configurati in modo da avere pacchetti di dimensione massima per il canale fisico utilizzato (in questo caso ethernet) e una durata dell'esperimento di circa 10 secondi. Per ottenere il numero N di byte totali, conoscendo la velocità ed il tempo desiderato usiamo la seguente formula:

$$N = \frac{C \cdot \Delta T}{8} = \frac{10Mb/s \cdot 10s}{8} = 12,5MB$$

Per conoscere il numero di pacchetti da inviare invece basta dividere N per la grandezza di un pacchetto (1538). Infine come grandezza del pacchetto usiamo la MSS. I parametri usati nel caso di TCP sono $l = 1448$ ed $n = 8127$. Per UDP invece abbiamo $l = 1472$ ed $n = 8127$.

Tutti i dati riportati sono stati catturati mediante l'uso dei due script in appendice 8.3, adattati allo scenario tramite i parametri opzionali di nttcp. Gli script in questione eseguono i test richiesti un numero N di volte e calcolano la media tra i risultati ottenuti.

3 Scenario A

In questa sezione analizziamo il caso in cui entrambi gli host H1 ed H2 abbiano negoziato con lo switch una connessione Full Duplex a 10Mbps.

3.1 TCP

In questo primo caso osserviamo il comportamento della rete quando nttcp viene eseguito in modalità TCP dall'host H1 verso l'host H2, che svolge la funzione di server.

I canali essendo full duplex sono esenti da collisioni. Essendo TCP un protocollo affidabile tutti i pacchetti vengono ricevuti correttamente da H2 e non sono quindi presenti perdite. Quando una connessione TCP viene completata, di default, viene istanziato un "TCP Kernel socket buffer" bloccante; non sono quindi possibili perdite nella pila protocollare e la velocità di scrittura sul socket si allinea a quella di trasmissione sul canale. Il protocollo di trasporto in questione fa inoltre uso di un sistema di frammentazione votato all'efficienza, scomponendo i dati ricevuti tenendo conto del proprio header, dell'header IP e della MSS in modo da ottimizzare il goodput.

$$G \leq C \cdot \eta_{TCP,FD} = C \cdot \frac{1448}{1448 + 32_{TCP} + 20_{IP} + 38_{ETH}} \simeq 9,41\ Mb/s \tag{1}$$

Osservando i risultati ottenuti nella tabella 8.1 si può notare come il Goodput sia sovrastimato al trasmettitore superando anche il Goodput massimo atteso calcolato nell'equazione 1. Questo fenomeno si presenta poichè al trasmettitore è misurato il tempo di generazione dei pacchetti, ignorando il tempo di attesa all'interno del buffer della scheda di rete. Contrariamente al ricevitore la misura è effettuata correttamente poiché le misurazioni contengono intrinsecamente anche il tempo trascorso nei buffer sulla linea e come possiamo osservare coincide quasi perfettamente col valore di Goodput atteso.

3.2 UDP

In questo seconda esperienza osserviamo il comportamento della rete quando nttcp viene eseguito in modalità UDP dall'host H1 verso l'host H2.

Anche in questo scenario non sono presenti collisioni per via della presenza di soli canali Full Duplex. Questa volta, però, il protocollo di trasporto utilizzato è di tipo "connection-less" e non garantisce nessuna affidabilità nella ricezione dei pacchetti. Il kernel linux ha un approccio molto più aggressivo sui socket UDP, e cerca di passare i pacchetti in essi contenuti ai livelli inferiori al massimo della velocità concessa , nell'ordine dei Gbps, causando la saturazione del buffer della scheda di rete, che trasmette a soli 10Mbps, e la conseguente perdita di pacchetti nella pila protocollare[1]. Ciò comporta una netta sovrastima del Goodput al trasmettitore, il quale assume di aver inoltrato nella rete un quantitativo di dati sensibilmente superiore a quello che di fatto viene trasmesso sul canale fisico. Al ricevitore vengono invece osservati solo i pacchetti effettivamente trasferiti e la misurazione viene quindi effettuata correttamente.

É rilevante alla nostra analisi ricordare che UDP non esegue una segmentazione ottimale sul payload ricevuto, ma essendo un protocollo "best-effort", si limita ad aggiungere, senza fare considerazioni aggiuntive, il suo header a ciasciun pacchetto ricevuto e a passarlo al livello successivo. Se il payload associato ad ogni pacchetto è minore di 1472 Byte (su canale ethernet) il rendimento diminuisce; se, invece, il payload associato a ciascun pacchetto è maggiore di 1472 il protocollo di rete IP svolgerà la frammentazione: per ciascun frammento perso nella pila protocollare il ricevitore scarta ogni frammento ad esso collegato comportando perdite sempre più significative al crescere del payload associato al pacchetto UDP.

Nella nostra esperienza è stato usato un payload di dimensioni ottimali. L'intestazione UDP, di soli 8byte, permette un Goodput superiore a quello ottenuto utilizzando TCP, che come possiamo constatare osservando i risultati in figura 8.2 si allinea quasi perfettamente con il valore previsto mediante la seguente formula:

$$G \leq C \cdot \eta_{UDP,FD} = C \cdot \frac{1448}{1448 + 8_{UDP} + 20_{IP} + 38_{ETH}} \simeq 9,57\ Mb/s \tag{2}$$

Osservando le catture tramite il software "Wireshark" abbiamo potuto constatare che la negoziazione iniziale avviene tramite una connessione affidabile TCP mentre la fase di chiusura, non potendo far affidamento sul numero di dati ricevuti (per via delle perdite), viene sancita dall'invio di 3 pacchetti contenenti 4 byte di payload, grazie ai quali possiamo anche giustificare le 3 system call aggiuntive che vengono effettuate al trasmettitore durante i nostri esperimenti.

[1] I messaggi per il controllo di flusso inviati nella pila protocollare non sono abbastanza veloci da impedire il fenomeno

4 Scenario B

In questo caso il canale tra H1 e lo switch viene impostato come half duplex.

4.1 TCP

Quando usiamo una connessione TCP su canale half-duplex dobbiamo tener conto degli ACK che transitano in direzione opposta ad i pacchetti. Questi prenderanno una parte della banda riducendo il goodput misurato. Se ipotizziamo che un ACK viene inviato ogni volta che un pacchetto è ricevuto l'efficienza può essere calcolata come segue:

$$G \leq C \cdot \eta_{TCP,HD} = C \cdot \frac{1448}{1448 + 32_{TCP} + 20_{IP} + 38_{ETH} + 84_{ACK}} \simeq 8,93\ Mb/s \tag{3}$$

4.1.1 Trasmettitore in Half-Duplex

Configurando il canale che si interpone tra il trasmettitore e lo switch in Half-Duplex si ottengono i risultati mostrati in tabella 8.1. Il goodput misurato correttamente al ricevitore è notabilmente inferiore a quello calcolato nell'equazione 3. Ciò è causato dalle collisioni, che come mostrato in tabella 8.1 si verificano con una non trascurabile probabilità del 1,43% in questo scenario. Seppur questa possibilità sia bassa, poiché i cavi in gioco sono corti ed i tempi di propagazione sono trascurabili rispetto a quelli di collisione, è decisamente superiore al valore atteso (nell'ordine di 10^{-6}). Questo fenomeno è causato dal tempismo con cui gli "acknowledge" vengono inoltrati, ovvero nell'istante in cui la trasmissione di un pacchetto termina e si tenta di inviare il successivo causando, talvolta, collisioni. Le collisioni scatenano, inoltre, l'invio dei "pause frame", utilizzati per regolare il flusso sul canale e che inficiano sul goodput misurato.

4.1.2 Ricevitore in Half-Duplex

In questo scenario il canale in Half-Duplex si interpone tra lo switch e il ricevitore. I risultati ottenuti, seppur simili, si discostano dall'esperienza precedente per il numero di collisioni aumentato. Ciò si verifica poiché il ricevitore comunica direttamente sul canale Half-duplex e gli acknowledge, non subendo il ritardo causato dallo S&F dello switch, tendono a collidere più spesso.

4.2 UDP

Nel caso di una connessione UDP il modello non cambia poiché non si fa uso degli ACK ed i dati viaggiano in una sola direzione. Le poche collisioni che si verificano sono causate da pacchetti esterni al nostro esperimento. Ne consegue inoltre che nessuna variazione avviene quando il canale in half duplex si trova tra il trasmettitore e lo switch o tra il ricevitore e lo switch.

5 Scenario E

In questo scenario i dati vengono inviati da due dispositivi verso un unico ricevitore. I canali sono tutti impostati in full duplex.

5.1 TCP

In questa esperienza abbiamo utilizzato due connessioni TCP. Come abbiamo potuto osservare, nel transitorio iniziale, entrambi gli host tentano di trasmettere al massimo della velocità concessa sul proprio canale, inviando allo switch un flusso totale di dati pari a 20 Mbps che lo switch incanala verso H2 alla velocità negoziata di 10Mbps. Quando il buffer dello switch viene, per forza di cose, saturato circa il 50% del flusso ricevuto sarà perso allo switch. Ciò genera l'invio di ACK duplicati da parte del ricevitore e innesca il processo di controllo di congestione caratteristico di TCP, che si occupa di suddividere la capacità equamente tra H1 ed H3. Il goodput misurato per ciascuna connessione è maggiore della meta del goodput totale atteso

(9,41) poiché è impossibile avviare nello stesso istante di tempo i due processi nttcp ottenendo così misure forvianti.

5.2 UDP

In questa esperienza abbiamo utilizzato due connessioni UDP. La premessa è la stessa del caso precedente, ma data l'assenza del controllo di congestione, le perdite osservate superano il 50% (50% teorico più un quantitativo perso nella pila protocollare) per tutta la durata dell'esperienza. Osservando i valori di goodput misurati e i grafici prodotti tramite wireshark, vedi figura 8.3, si deduce che il flusso di dati che raggiungono il ricevitore sia distribuito equamente tra i due trasmettitori.

Il goodput osservato per ciascuna connessione è inferiore a quello atteso (9,56/2) a differenza di quanto visto per TCP: possiamo ipotizzare che, non essendoci nessun controllo di congestione, la linea venga sovraccaricata e il controllo di flusso Ethernet interviene inoltrando dei pause frame per limitare la velocità di trasmissione.

5.3 TCP & UDP

In questa esperienza i trasmettitori H2 e H3 usano rispettivamente una connessione TCP ed UDP. Anche in questo scenario la premessa è la stessa, ma il controllo di congestione subentra solo per una delle due connessioni. Il flusso UDP continua invece senza limitazioni a 10Mbps, soffocando la connessione TCP la cui velocità di trasmissione tende allo zero. Una volta terminata la trasmissione UDP, riprende il flusso di di dati TCP ad una velocità di 10Mbps sul canale ormai libero.

Si misura, quindi, un goodput per la connessione UDP leggermente più basso del valore teorico nel caso di un canale libero, per via del breve transitorio iniziale in cui TCP contribuisce alla saturazione della rete. Per quanto concerne la connessione TCP, il goodput è molto inferiore a quello ideale in caso di canale completamente libero poichè per quasi tutta la durata della connessione UDP non vengono inoltrati pacchetti e di conseguenza all'incirca raddoppia il tempo necessario al completamento della trasmissione. Il goodput è però anche significativamente superiore a 4,7: poichè UDP subisce consistenti perdite e la connessione termina prima, diminuendo quindi i tempi di attesa per TCP, fenomeno che si verificherà anche in tutti gli altri casi "misti".

6 Scenario F

Questo scenario è simile al precedente, ma il canale tra il ricevitore e lo switch viene impostato in half duplex.

6.1 TCP

Anche in questo caso, come nel precedente ci aspettiamo una simmetria tra i flussi inviati da H2 ed H3, come si vede in figura 8.4. Le collisioni, come potevamo immaginare, quasi raddoppiano rispetto al caso 4.1.2 poiché in numero totale di dati inviati, e di ACK generati, è anch'esso raddoppiato. Ciò è causa di una significativa riduzione del goodput misurato, come possiamo osservare dalla tabella 8.1 nel sommario.

6.2 UDP

Questo caso non differisce dallo scenario equivalente nel caso E (full duplex) in quanto il ricevitore non invia alcun pacchetto di riscontro ed il canale viene usufruito in maniera mono-direzionale.

6.3 TCP & UDP

In questo scenario si ripetono le considerazioni fatte per l'esperienza equivalente 5.3. Il canale che collega lo switch al ricevitore è però in Half duplex questa volta e perciò si verificano un numero di collisioni non trascurabili che causa la diminuzione del goodput per entrambe le connessioni. Gli ACK, e soprattutto gli

ACK duplicati vengono trasmessi durante tutta la durata dell'esperimento generando collisioni anche con i pacchetti UDP e così motivandone un netto incremento rispetto al caso 4.1.2.

Anche in questo caso le numerose perdite che affliggono UDP favoriscono il goodput di TCP rispetto al valore atteso.

7 Scenario G

In questo scenario, a differenza dei precedenti sarà H1 a trasmettere verso entrambi gli host H2 e H3 e tutti i canali della rete sono impostati in full duplex.

Nei test svolti la somma dei goodput delle due connessioni aperte in contemporanea tende al valore teorico atteso, seppur con delle piccole variazioni dovute principalmente alla non perfetta coincidenza tra gli istanti in cui i due processi vengono avviati, le connessioni stabiliti e/o terminate.

7.1 TCP

Avendo entrambi i processi la stessa priorità, vengono schedulati secondo lo stesso algoritmo; ne consegue che i due flussi sono simmetrici ed equamente divisi nella capacità del canale. Diversamente dallo scenario precedente però il quantitativo di dati ricevuti allo switch non raddoppia e il flusso in entrata resta limitato a 10Mbps.

7.2 UDP

Anche quando entrambe le connessioni sono UDP lo scheduler si comporta come già ipotizzato e i flussi vengono quindi divisi equamente. Il kernel ha però un maggior numero di sockets da cui attingere dati, e per via dell'approccio aggressivo con cui avviene nel caso UDP, si hanno perdite pari a circa il doppio di quelle osservate nel caso 3.2.

7.3 TCP & UDP

In questo scenario, a differenza dei casi visti prima, UDP non prevale su TCP,come si può notare dal grafico 8.5, poiché i due flussi hanno la stessa priorità dal punto di vista dello scheduler che li arbitra. Le differenze in goodput osservabili sono causate soltanto dal diverso valore del rendimento e dall'ordine in cui i processi vengono avviati.

La connessione UDP è, anche questa volta, soggetta a perdite pari a quasi il doppio del caso 3.2 poiché deve condividere il buffer di invio della scheda di rete con la connessione TCP attiva. Inoltre a causa di tali perdite i flussi TCP ed UDP inviano un quantitativo totale di dati diverso, aumentando così il valore di goodput TCP osservato.

8 Scenario H

In quest'ultimo caso impostiamo il canale tra il trasmettitore e lo switch in half duplex.

8.1 TCP

Questo scenario è molto simile all'esperienza equivalente precedente, ma il canale in half duplex è causa di un numero significativo di collisioni che comportano, quindi, un abbassamento del goodput rispetto al caso 7.1.

8.2 UDP

Come osservato più volte, i test su connessioni puramente UDP, non sono influenzati dall'impostazione del duplex sul canale, si ottiene quindi un esperienza identica allo scenario 7.2.

8.3 TCP & UDP

Per questo scenario le premesse sono le stesse di quelle del caso equivalente in full duplex, ma come era già avvenuto nello scenario 6.3 la presenza di collisioni causa una non trascurabile diminuzione del goodput totale.

✓

– molto ben fatta e completa

11/10

Appendice

Sommario TCP

Test	Average TCP Goodput per flow		Collision probability		Loss at the application layer	
	Prediction	Observed	Prediction	Observed	Prediction	Observed
A	9,41	9,41	No	No	No	No
B (tx HD)	8,93	8.,11	Trascurabili	1,43%	No	No
B (rx HD)	8,93	8,32	Trascurabili	1,64 %	No	No
E(H2)	4,7	5,07	No	0%	No	No
E(H3)	4,7	4,89	No	0 %	No	No
F(H2)	4,46	4,37	Trascurabili	2,30%	No	No
F(H3)	4,46	4,28	Trascurabili	2,30 %	No	No
G(H2)	4,7	4,71	No	0%	No	No
G(H3)	4,7	4,88	No	0%	No	No
H(H2)	4,46	4,71	Trascurabili	2,43%	No	No
H(H3)	4,46	4,42	Trascurabili	2,43%	No	No

Tabella 8.1: Riepilogo degli esperimenti sul goodput TCP

Sommario UDP

Test	Average UDP Goodput per flow		Collision probability		Loss at the application layer (MByte)	
	Prediction	Observed	Prediction	Observed	Prediction	Observed
A	9,57	9,57	No	0%	No	19,9%(2,39)
B	9,57	9,52	No	0,01%	No	20,06% (2,40)
E(H2)	4,78	4,64	No	0%	50 %	57% (6,82)
E(H3)	4,78	4,14	No	0%	50 %	59,28% (7,09)
F(H2)	4,78	4,13	No	0,02 %	50 %	59,53% (7,12)
F(H3)	4,78	4,13	No	0,02 %	50 %	56,9% (6,81)
G(H2)	4,78	4,79	No	0 %	No	41,88% (5,01)
G(H3)	4,78	4,33	No	0 %	No	41,88% (5,01)
H(H2)	4,78	4,59	No	0.02%	No	41,72% (4,99)
H(H3)	4,78	4,81	No	0.02%	No	41,72% (4,99)

Tabella 8.2: Riepilogo degli esperimenti sul goodput UDP

Sommario TCP e UDP

Test	Average Goodput per flow		Collision probability	
	Prediction	Observed	Prediction	Observed
E(TCP)	4,7	5,27	No	0%
E(UDP)	9,56	8,97	No	0 %
F(TCP)	4,46	4,67	Trascurabili	1,17%
F(UDP)	9,56	8,55	Trascurabili	1,17%
G(TCP)	4,7	5,883	No	0 %
G(UDP)	4,78	5,087	No	0%
H(TCP)	4,46	4,81	Trascurabili	2,14%
H(UDP)	4,78	5,75	Trascurabili	2,14%

Tabella 8.3: Riepilogo degli esperimenti sul goodput TCP e UDP

Grafici

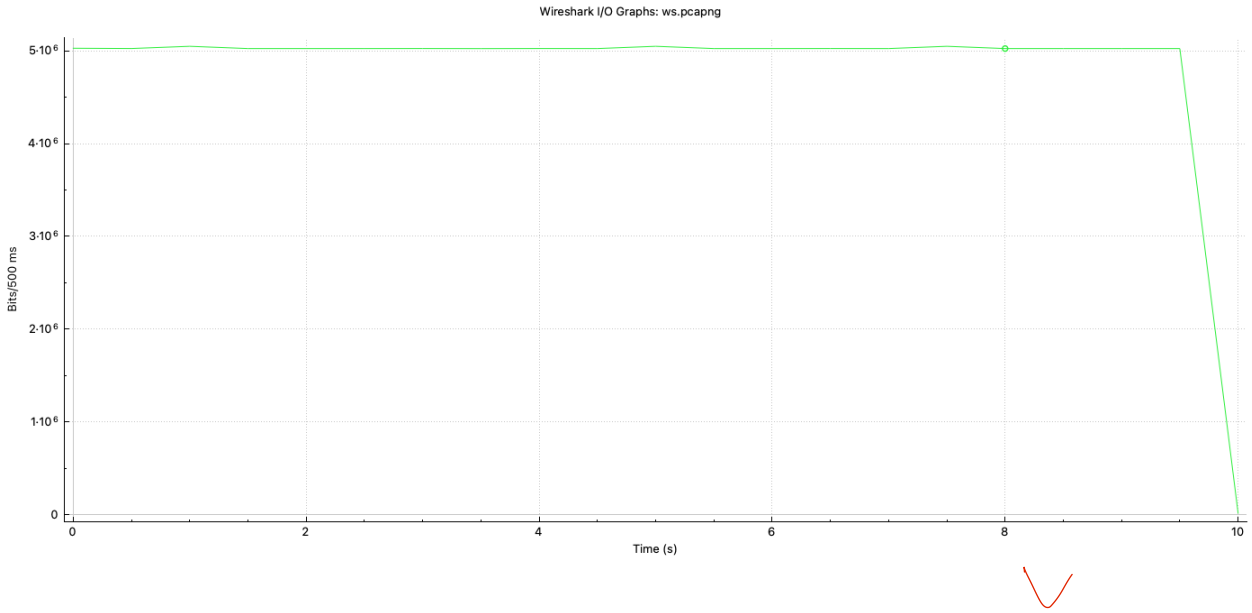

Figura 8.1: Grafico scenario A con TCP

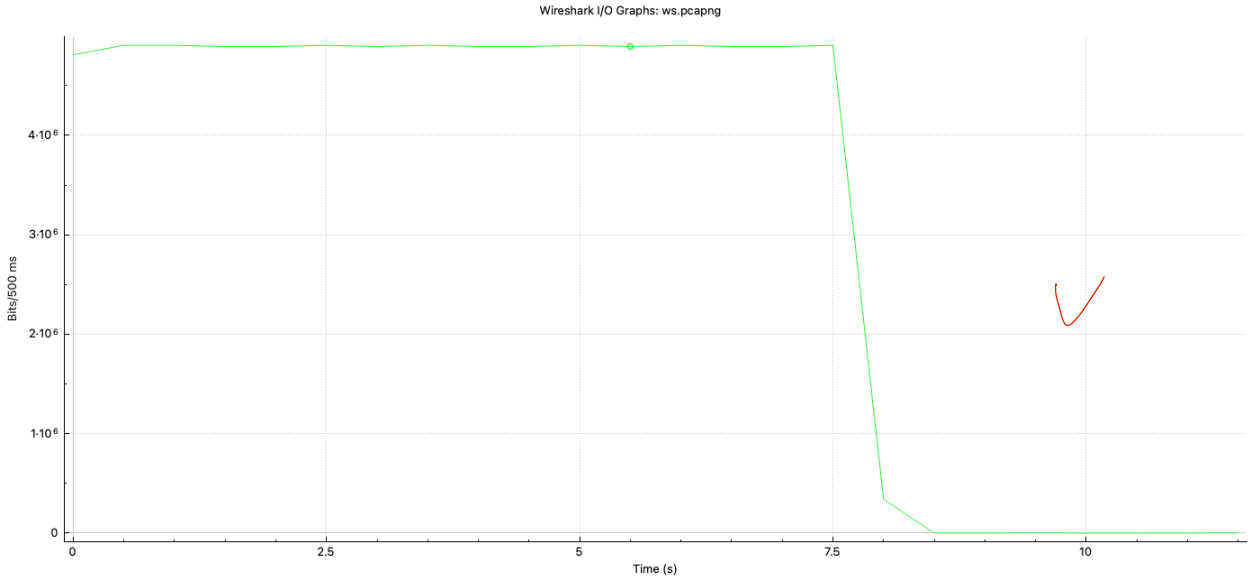

Figura 8.2: Grafico scenario B con UDP e half-duplex al ricevitore

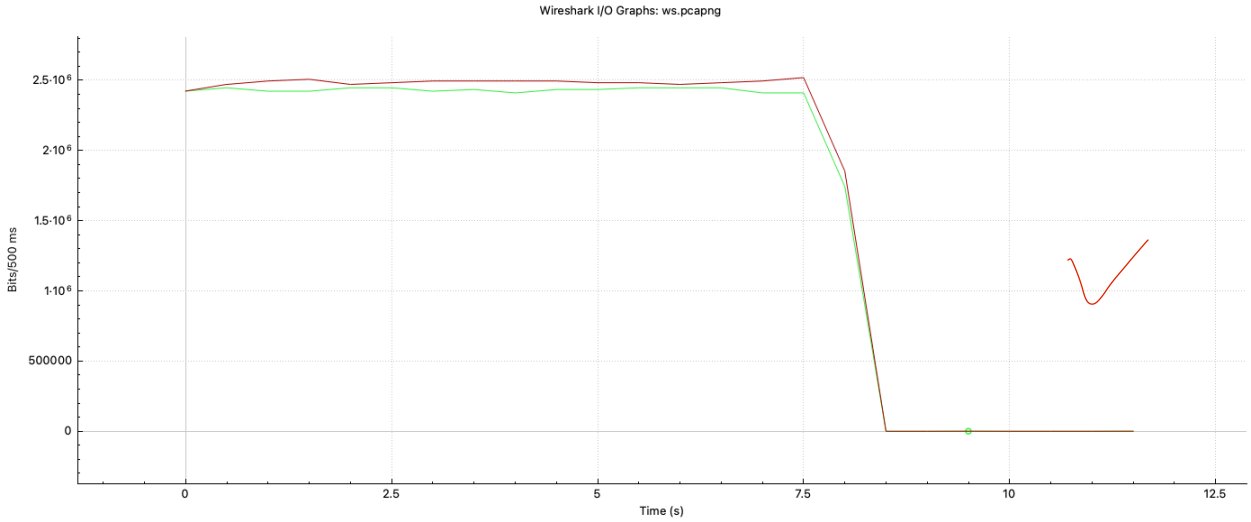

Figura 8.3: Grafico scenario E con UDP

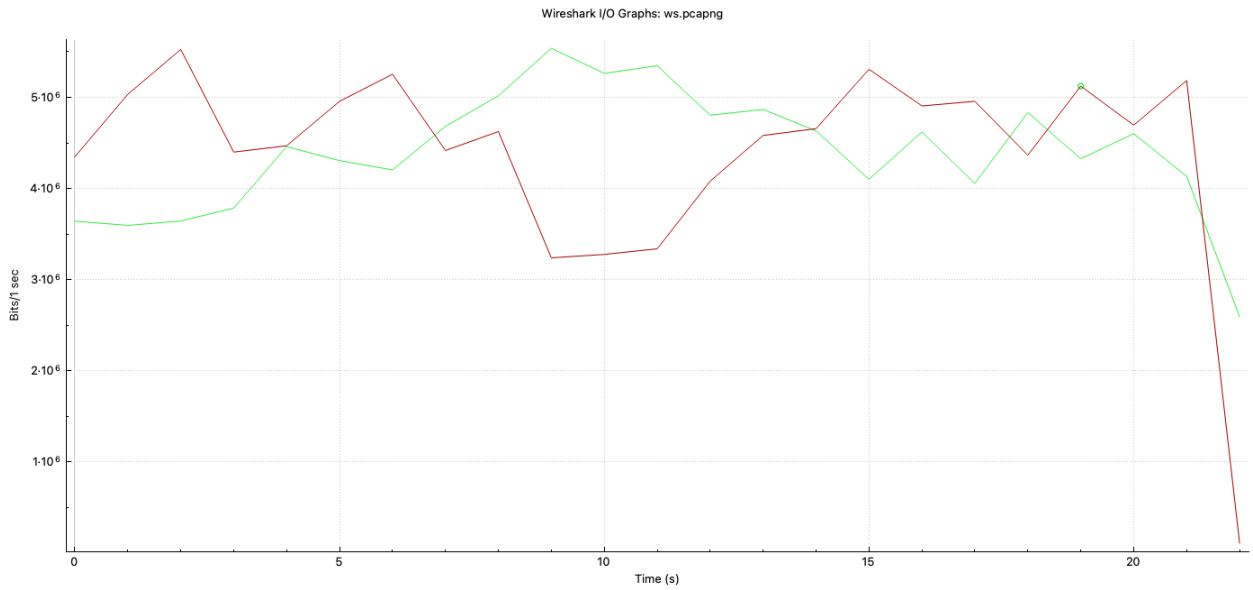

Figura 8.4: Grafico scenario F con TCP

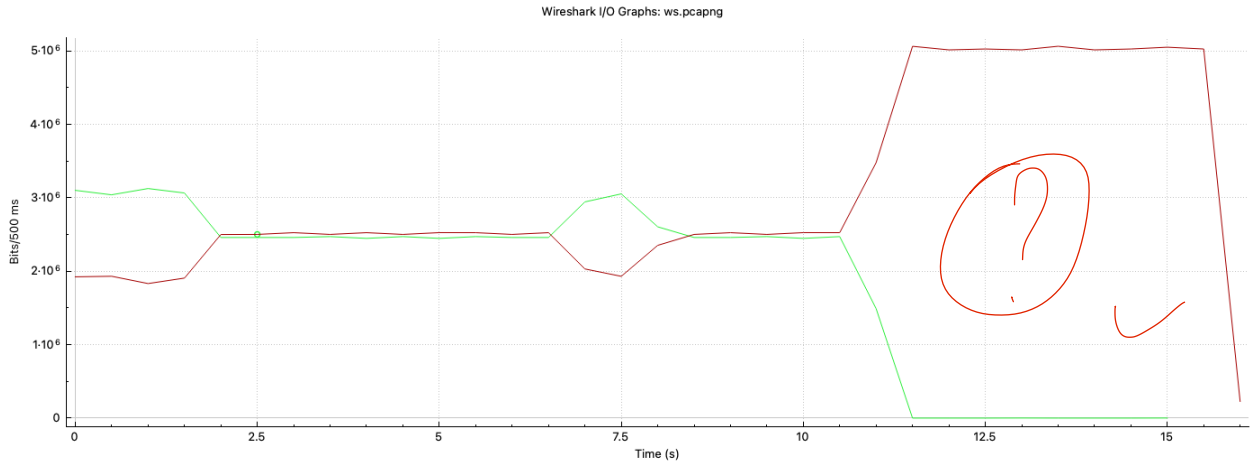

Figura 8.5: Grafico scenario G con TCP e UDP

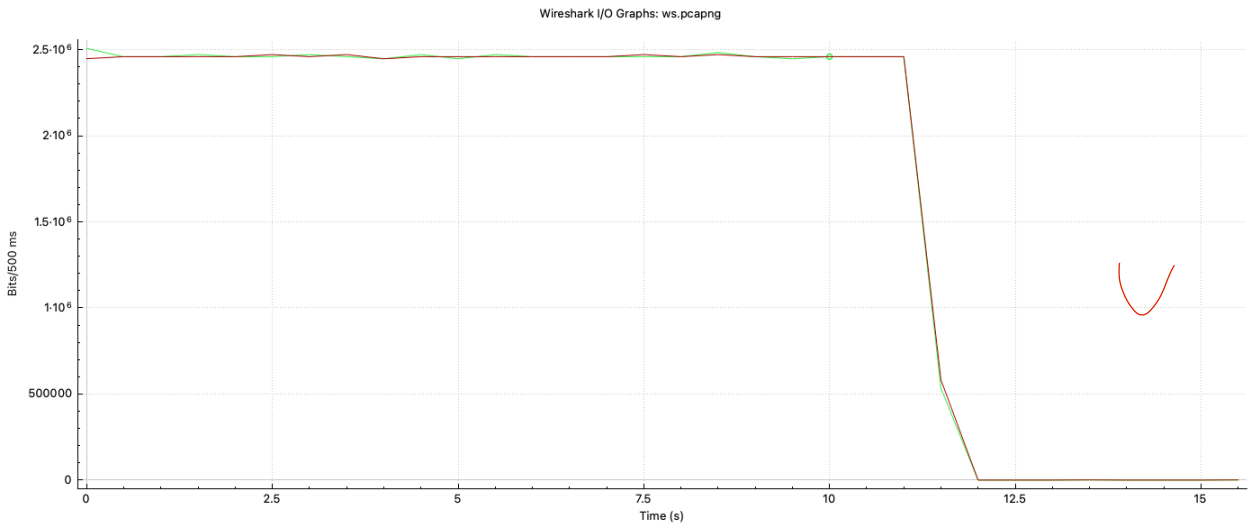

Figura 8.6: Grafico scenario H con UDP

Script

```
#bin\bash
#
H1=172.16.14.1
H2=172.16.14.2
H3=172.16.14.3

rm -f  data.dat dataRx.dat dataTx.dat results.dat

N=3

Gtx=0
Grx=0
Col=0
Loss=0

RANGE=$(seq 1 1 $N)

for k in $RANGE; do

tmp1=$(ifconfig eth0 | grep collisions| tr -s " " | cut -d " " -f12)
nttcp -n 8127 -l 1448 $H2 >> data.dat
tmp2=$(ifconfig eth0 | grep collisions| tr -s " " | cut -d " " -f12)
let Col=Col+tmp2-tmp1

echo  >> data.dat

Tx=$(cat data.dat | tail -n 3 | head -n 1 | tr -s " ")
Rx=$(cat data.dat | tail -n 2 | head -n 1 | tr -s " ")

Gtx=$(echo "$Gtx + $(echo $Tx | cut -d " " -f5)" | bc)
Grx=$(echo "$Grx + $(echo $Rx | cut -d " " -f5)" | bc)

Loss=$(echo "$Loss + $(echo $Tx | cut -d " " -f2) - $(echo $Rx | cut -d " " -f2)" | bc)

echo $Tx >> dataTx.dat
echo $Rx >> dataRx.dat
done

Gtx=$(echo "scale=3; $Gtx/$N" | bc)
Grx=$(echo "scale=3; $Grx/$N" | bc)
Col=$(echo "scale=3; $Col/$N" | bc)
Loss=$(echo "scale=3; $Loss/$N" | bc)

echo "Vel media TX | Vel media RX | Collisions | Loss" >> results.dat
echo "$Gtx          $Grx          $Col            $Loss" >> results.dat
cat results.dat
```

Figura 8.7: Script per la raccolta dei dati tra due host

```
#bin\bash
H1=172.16.14.1
H2=172.16.14.2
H3=172.16.14.3

rm -f  dataH2.dat dataRxH2.dat dataTxH2.dat resultsH2.dat dataH3.dat dataRxH3.dat dataTxH3.
    dat resultsH3.dat

N=3
GtxH2=0
GrxH2=0
GtxH3=0
```

```
GrxH3=0
Col=0
LossH2=0
LossH3=0

RANGE=$(seq 1 1 $N)

for k in $RANGE; do

tmp1=$(ifconfig eth0 | grep collisions| tr -s " " | cut -d " " -f12)
nttcp -r -n 8127 -l 1448 $H2 >> dataH2.dat & PID1=$!
nttcp -r -n 8127 -l 1448 $H3 >> dataH3.dat & PID2=$!
wait $PID1
wait $PID2
tmp2=$(ifconfig eth0 | grep collisions| tr -s " " | cut -d " " -f12)
let Col=Col+tmp2-tmp1

echo  >> dataH2.dat
echo  >> dataH3.dat

TxH2=$(cat dataH2.dat | tail -n 3 | head -n 1 | tr -s " ")
RxH2=$(cat dataH2.dat | tail -n 2 | head -n 1 | tr -s " ")

TxH3=$(cat dataH3.dat | tail -n 3 | head -n 1 | tr -s " ")
RxH3=$(cat dataH3.dat | tail -n 2 | head -n 1 | tr -s " ")

GtxH2=$(echo "$GtxH2 + $(echo $TxH2 | cut -d " " -f5)" | bc)
GrxH2=$(echo "$GrxH2 + $(echo $RxH2 | cut -d " " -f5)" | bc)

GtxH3=$(echo "$GtxH3 + $(echo $TxH3 | cut -d " " -f5)" | bc)
GrxH3=$(echo "$GrxH3 + $(echo $RxH3 | cut -d " " -f5)" | bc)

LossH2=$(echo "$LossH2 + $(echo $TxH2 | cut -d " " -f2) - $(echo $RxH2 | cut -d " " -f2)" |
    bc)

LossH3=$(echo "$LossH3 + $(echo $TxH3 | cut -d " " -f2) - $(echo $RxH3 | cut -d " " -f2)" |
    bc)

echo $TxH2 >> dataTxH2.dat
echo $RxH2 >> dataRxH2.dat

echo $TxH3 >> dataTxH3.dat
echo $RxH3 >> dataRxH3.dat

done

GtxH2=$(echo "scale=3; $GtxH2/$N" | bc)
GrxH2=$(echo "scale=3; $GrxH2/$N" | bc)

GtxH3=$(echo "scale=3; $GtxH3/$N" | bc)
GrxH3=$(echo "scale=3; $GrxH3/$N" | bc)

Col=$(echo "scale=3; $Col/$N" | bc)

LossH2=$(echo "scale=3; $LossH2/$N" | bc)
LossH3=$(echo "scale=3; $LossH3/$N" | bc)

echo "Vel media TX | Vel media RX | Collisions | Loss---> H2" >> resultsH2.dat
echo "$GtxH2          $GrxH2          $Col            $LossH2" >> resultsH2.dat
cat resultsH2.dat

echo "Vel media TX | Vel media RX | Collisions | Loss---> H3" >> resultsH3.dat
echo "$GtxH3          $GrxH3          $Col            $LossH3" >> resultsH3.dat

cat resultsH3.dat
```

Listing 1: Script per la raccolta dei dati tra tre host