Relazione 4

Analisi del goodput

Mattia Chiarle 269969, Michele Ferrero 268542, Gabriele Ferro 268510

Gruppo 22 Anno 2021/2022

Introduzione

L'esperienza consiste nell'analisi del goodput, della rete locale stabilita, in diversi casi in base al tipo di protocollo di livello trasporto adottato (TCP o UDP), al tipo di canale utilizzato (FULL o HALF duplex) e al numero e alla direzione delle comunicazioni effettuate contemporaneamente.
Per ogni caso vengono prima formulate delle previsioni che tengono conto, a livello teorico, delle implementazioni dei protocolli e successivamente vengono comparate le ipotesi fatte con i risultati effettivi del test.

Il test viene svolto tramite il programma "nttcp" il quale raccoglie ed elabora i dati riguardanti il transfer rate della connessione considerata. Nella fase di setup viene avviato il programma sulla macchina da contattare tramite il comando "nttcp -i" che permette al programma di comportarsi da daemon e al dispositivo di essere visto e contattato da altri host per effettuare test su di esso.
L'output del comando mostra due righe, una per l'host locale "l" e una per quello contattato "1", in cui vengono illustrati i risultati del test, tra cui i campi più rilevanti sono: il numero di byte inviati/ricevuti, il tempo in secondi della trasmissione/ricezione dei byte, il tempo di elaborazione da parte della CPU, il goodput reale misurato della comunicazione tra i due dispositivi calcolato come "$G = \frac{Bytes\ Inviati}{Tempo}$", la velocità in Mb/s di elaborazione della CPU e il numero di chiamate alla CPU

I casi analizzati, sia per TCP che per UDP, sono:

- A - FULL duplex e singolo flusso: H1 → H2
- B - HALF duplex e singolo flusso: H1 → H2
- E - FULL duplex e flusso multiplo al ricevitore: H1 → H2 e H3 → H2
- F - HALF duplex e flusso multiplo al ricevitore: H1 → H2 e H3 → H2
- G - FULL duplex e flusso multiplo dal trasmettitore: H1 → H2 e H1 → H3
- H - HALF duplex e flusso multiplo dal trasmettitore: H1 → H2 e H1 → H3

Per ogni caso viene valutato il goodput, la probabilità di collisione e la percentuale di perdita dei pacchetti a livello applicazione.
Nei casi in cui sia previsto di utilizzare un canale HALF duplex, vengono effettuati i test per ogni combinazione possibile di canali per quanto riguarda la sua posizione.

In generale la misura di velocità affidabile sarà sempre quella del ricevitore: il trasmettitore tende infatti a sovrastimare, in quanto effettua la misura a livello 7 e quindi vede solo la velocità di generazione dei pacchetti e non la loro effettiva velocità di trasmissione. Trascura, di conseguenza, sia il tempo necessario per attraversare la pila protocollare sia soprattutto il tempo di accodamento. Questo effetto sarà rilevante soprattutto negli scenari più critici con UDP.

Per scegliere la dimensione del payload si aprono due scenari differenti in base al protocollo usato: Con TCP risulta ininfluente la scelta effettuata purché la dimensione dei blocchi non sia troppo piccola (altrimenti diventa la CPU il collo di bottiglia), mentre per UDP la dimensione del payload è fondamentale e questo verrà dimostrato durante l'esempio A.
Il protocollo TCP, infatti, manda sempre pacchetti grandi una MSS, ottimizzando il più possibile la trasmissione mentre UDP aggiunge unicamente l'intestazione senza effettuare raggruppamenti.
Altro importante parametro è la durata dell'esperimento, questo in ambedue i casi deve essere almeno pari a 10 secondi per rendere ininfluenti i tempi di accodamento nella pila protocollare.

Altre importanti formule per la stima del goodput sono:

- $G = \eta * C$ (con G=goodput) dove η è l'efficienza stimata
- $\eta_{TCP,FD} = \frac{Payload\ Length}{Numero\ di\ bytes\ trasmessi\ per\ un\ payload}$

Caso A

Nel primo caso, data la configurazione utilizzata, non si prevedono collisioni (grazie ai canali full duplex) né perdite sia con TCP sia con UDP. Di conseguenza, l'efficienza attesa per TCP risulta essere

$$\eta_{TCP,FD} = \frac{1448}{1448+20_{header\ TCP}+20_{header\ IP}+38_{header\ ethernet}+12_{bit\ di\ opzioni}} = 0.941$$

Si ipotizza dunque che nttcp rilevi una velocità di circa 9.41 Mb/s. È stato quindi eseguito un test inviando 8000 blocchi da 1460 B. In effetti con questi valori nttcp genera 1460*8000*8=93.440.000 bit, che con la capacità di 10 Mb/s verranno trasmessi in più di 9.34 secondi (a causa dell'overhead dei vari header).
Analizzando i risultati tramite l'output di ifconfig e di nttcp si nota in effetti come H2 riceva lo stesso numero di byte di H1, confermando l'ipotesi iniziale di assenza di perdite. Inoltre il contatore delle collisioni in A rimane 0 dopo l'esecuzione del comando, confermando anche la seconda ipotesi. Infine, la velocità misurata è 9.4089, ottenendo un errore trascurabile rispetto al valore atteso.
Con UDP l'efficienza attesa invece è

$$\eta_{UDP,FD} = \frac{1472}{1472+8_{header\ UDP}+20_{header\ IP}+38_{header\ ethernet}} = 0.957$$

La velocità attesa è di conseguenza circa 9.57 Mb/s. Si nota in particolare come UDP, in quanto è un protocollo meno robusto, richiede meno byte di intestazione e di conseguenza permette di avere un'efficienza teorica più alta. Tuttavia non ottimizza la dimensione dei pacchetti: mentre TCP manda sempre un pacchetto di dimensione MSS a prescindere dalla grandezza dei blocchi, UDP manda sempre un blocco per volta (eventualmente frammentando). Per analizzare questo comportamento sono stati effettuati due test, uno con blocchi lunghi 1472 B (in modo da massimizzare il goodput) e uno con blocchi da 100 B.
Con il primo test è stata ottenuta una velocità di 9.5679 Mb/s, coerente con quanto atteso, mentre nel secondo 6.0225 Mb/s, con un errore del 37%. In entrambi i casi il numero di blocchi è stato scelto in modo da mantenere la durata dell'esperimento di circa 10 secondi (sono stati usati rispettivamente 8.000 e 100.000 blocchi). Inoltre, si nota un comportamento particolare inaspettato: a livello teorico, vista la velocità e il livello fisico usato, ci si aspetta di non avere perdite neanche con UDP.
La realtà è però diversa: nel primo test è misurata una perdita pari al 20% mentre nel secondo si è ridotta al 4%. Queste perdite sono dovute al fatto che UDP non implementa un meccanismo di controllo di flusso o congestione, l'applicazione genera di conseguenza i pacchetti alla massima velocità, riempiendo la coda della scheda di rete (in quanto l'invio è più lento della generazione). Quando il buffer è pieno, ethernet lo comunica a IP, che lo comunica a UDP che lo comunica all'applicazione ed in questo lasso di tempo tutti i pacchetti inviati (a causa del ritardo nella comunicazione) vengono persi.
Le perdite sono più accentuate nel primo caso in quanto la dimensione dei pacchetti è maggiore (1472 B di payload), e di conseguenza nello stesso tempo si perdono molti più byte rispetto all'altro caso ed in più il tempo di trasmissione minore permette di svuotare più velocemente la coda. Se fossero presenti frammentazioni inoltre la situazione peggiorerebbe ancora. Se venissero ad esempio inviati 4432 (pari a 4500-60 bytes di IP e 8 di UDP) B di payload ethernet sarebbero necessari 3 frammenti; tuttavia, se anche solo uno dei tre venisse perso è come se tutti e tre fossero stati persi in quanto IP non riesce a ricostruire il pacchetto originale e si triplicherebbe la perdita.

Caso B

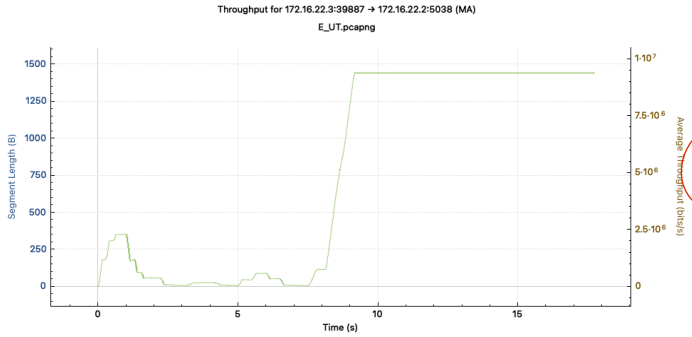
Per quanto riguarda l'analisi con TCP il valore atteso per ciascuna delle configurazioni è pari a:

$$\eta_{TCP,FD} = \frac{1448}{1448+20_{header\ TCP}+20_{header\ IP}+38_{header\ ethernet}+12_{bit\ di\ opzioni}+90_{dimensione\ ACK}} = 0.89$$

Questo valore non tiene conto di eventuali collisioni aventi probabilità teorica molto bassa (pari a $\frac{T_p}{T_{tx}+T_p} \simeq 4 \cdot 10^{-6}$).

Nonostante il modello ideale appena descritto, effettuando i test si osserva un comportamento inaspettato: i goodput risultano molto minori delle previsioni e sono inoltre presenti molte collisioni. Effettivamente la probabilità di collisione calcolata precedentemente si basa sull'ipotesi errata che i due host effettuino la trasmissione in momenti casuali, mentre in questo test la trasmissione avviene continuamente da parte del client nttcp. Di conseguenza, quando la trasmissione del pacchetto precedente termina, lo switch vuole inviare il pacchetto successivo mentre H2 l'ACK relativo al pacchetto appena ricevuto. Entrambi gli host iniziano quindi a trasmettere contemporaneamente, rendendo la probabilità di collisione non trascurabile. Il caso peggiore, sia a livello teorico sia a livello pratico, è proprio il caso 2, perché non vi è nessun ritardo da quando il ricevitore vuole inviare l'ACK a quando lo switch vuole inviare il prossimo pacchetto. Il caso 1 è il migliore in quanto il comportamento precedente risulta molto attenuato: stavolta H2 ha un canale FD e riesce quindi a inviare subito l'ACK, il quale arriva allo switch con un po' di ritardo (dovuto al tempo di accodamento nello switch, al tempo di trasmissione e a quello di propagazione). Quando lo switch vuole trasmetterlo al trasmettitore vede già il canale occupato. Questa desincronizzazione permette di ridurre molto il numero di collisioni rispetto agli altri casi, e il rallentamento è dovuto in parte anche al ritardo nella ricezione degli ACK (che causano ritardi nello scorrimento della finestra di trasmissione).
Infine, il caso 3 va leggermente meglio del caso 2 in quanto la presenza dei due canali HD va a ridurre la probabilità di collisione: poiché possono esserci collisioni su entrambi i canali cadono in parte i discorsi sulla sincronizzazione fatti precedentemente, andando a migliorare leggermente le perdite riscontrate. I risultati sono comunque molto simili al caso 2.
Per UDP valgono le stesse considerazioni presenti nel caso A: H2 non deve infatti inviare nessun ACK a H1, e di conseguenza il passaggio da FD a HD non comporta modifiche al goodput

Caso E



In questo caso con TCP ci si aspetta di riscontrare il fenomeno della TCP fairness: come conseguenza del controllo di congestione infatti a livello teorico due flussi TCP inviati sullo stesso canale tendono a dividersi equamente la velocità di trasmissione per limitare le perdite. I test hanno confermato questa ipotesi: ogni host ha registrato infatti una velocità di 4.71 Mb/s, ottenendo una velocità complessiva di 9.42 Mb/s praticamente coincidente con quella attesa di 9.41 Mb/s. Non sono presenti collisioni (grazie ai canali FD) né perdite (grazie a TCP).
Con UDP invece ci si aspetta nuovamente una velocità dimezzata, non grazie ai controlli di congestione ma a causa delle perdite nella pila protocollare e nello switch: quest'ultimo è limitato dai canali a cui è connesso, infatti in input riceve complessivamente da due canali distinti 20 Mb/s, mentre in output può al più inviare 10 Mb/s. Ci si aspetterebbe quindi una percentuale di perdita del 50% (statisticamente si perdono la metà dei pacchetti in ogni flusso), questa tuttavia nella pratica risulta leggermente più bassa (circa 41%) a causa dei pause frame. I pause frame sono dei messaggi che possono essere inviati dai dispositivi (switch o host) quando non riescono a stare dietro alla velocità di ricezione attuale: inviando questi pacchetti impongono al trasmettitore di fermarsi momentaneamente. Vista la ricezione si è cercato di disabilitare il pause frame come consigliato, e infatti i test successivi non lo includeranno.
Inoltre, la velocità risulta leggermente migliore di quanto atteso (4.81 Mb/s al posto di 4.785 Mb/s) ma comunque con un errore molto piccolo (circa 0.5%). Questa discrepanza è probabilmente dovuta a un leggero disallineamento nell'invio del comando da parte dei due host: in questo modo inizialmente uno dei due host trasmette alla massima velocità permessa (10 Mb/s) in quanto è l'unico a trasmettere. Dopo poco tempo arriva anche il secondo host e si iniziano a perdere molti pacchetti, e quindi il goodput massimo si divide tra i due host statisticamente in parti uguali.

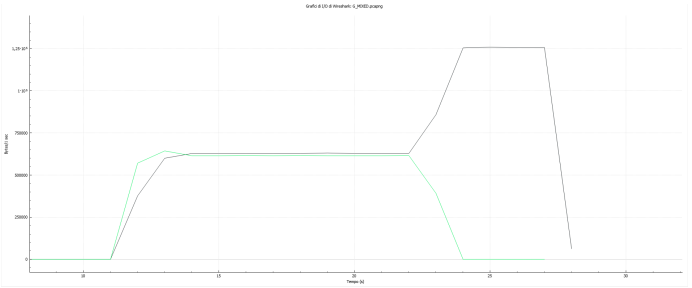
Infine, la trasmissione partita prima termina leggermente prima, consentendo all'altra di inviare dati alla massima velocità. Per questo motivo i goodput risultano leggermente superiori rispetto al risultato teorico atteso.
Come ultimo caso è stato eseguito un test in cui H1 invia dati con UDP e H2 con TCP. Come atteso UDP sovrasta TCP, in quanto non si cura del controllo di congestione mentre TCP, per cercare di ridurre le perdite, diminuisce la dimensione della finestra di trasmissione fino a portarla a 0. Di conseguenza UDP ha una velocità stimata vicina al valore ideale (9.01 Mb/s) ma leggermente inferiore: questo è dovuto al maggior numero di perdite nelle fasi iniziali, in cui TCP prova a trasmettere prima di dimensionare come atteso la finestra di congestione (e quindi di trasmissione) a 0
Non appena termina la trasmissione UDP, TCP può procedere in modo esattamente analogo al punto A. Di conseguenza, come si può notare dal grafico per la prima metà della trasmissione avrà goodput quasi nullo, mentre nella seconda metà circa 9.4 Mb/s. Si nota inoltre come per TCP, a causa del goodput iniziale non sempre nullo, la misura ottenuta non sarà esattamente $\frac{9,4}{2}$Mb/s (ovvero 4.7 Mb/s) ma sarà leggermente più veloce (5.3 Mb/s).

Caso F

In questo caso durante l'utilizzo del protocollo TCP si ottiene che il ramo in half duplex introduce collisioni tra i pacchetti inviati e gli ack ricevuti dall'host contattato come già visto nel caso B. Dato che sia H3 sia H2 comunicano con H1, se il ramo HD si trova nel tratto tra lo switch e H1 si ottiene un goodput limitato dall'alta probabilità di collisione tra i pacchetti. Il caso migliore osservato è quello nel caso in cui il tratto HD sia posizionato tra H2 (o H3) e lo switch: questo è dovuto alla desincronizzazione già analizzata precedentemente. Inoltre, si nota come poiché solo un trasmettitore ha un canale HD, questo ci impiegherà più tempo a completare la trasmissione rispetto all'altro a causa delle collisioni. L'altro host sarà quindi più veloce, completando la trasmissione in meno tempo e ottenendo un goodput migliore. Quando la sua trasmissione termina l'host più lento può però sfruttare tutta la banda, migliorando la sua misura (nonostante le collisioni siano ancora presenti). Complessivamente quindi si otterrà una media superiore al goodput teorico atteso, e l'host col canale FD avrà una velocità media migliore. Infine, il caso in cui tutti i canali sono half duplex risulta avere goodput minore rispetto agli altri casi, e ciò è dovuto alla necessità di andare a gestire i due flussi e gli ACK da essi derivanti con tutti canali HD.
Utilizzando UDP le considerazioni rimangono le stesse del caso E già descritto poiché non sono previsti pacchetti di conferma nel protocollo e quindi non risultano limitazioni derivanti dall'utilizzo di un canale half duplex.

Il caso misto analizzato, in cui H2 trasmette in TCP e H3 trasmette in UDP, utilizza tutti i canali in half duplex. Si nota che l'andamento è lo stesso del test analizzato al caso E (sempre per il caso misto) dove TCP riduce la dimensione della propria finestra fino ad azzerarla, per via del controllo di congestione applicato dal protocollo, e a seguito del termine della trasmissione dei pacchetti di tipo UDP la finestra incrementa nuovamente la sua dimensione fino ad arrivare a MSS. In questo caso, a differenza di quello con i canali full duplex, TCP si ritrova a dover gestire delle collisioni, portando ad una stima del goodput inferiore rispetto al corrispettivo test con canale full duplex. La misura ottenuta risulta leggermente superiore alla previsione (ovvero la velocità di TCP HD dimezzata) per il goodput iniziale non nullo analizzato precedentemente. Si nota inoltre come UDP abbia molte più perdite rispetto al caso E, che comportano una diminuzione del goodput ottenuto (circa 7.5 Mb/s): questo è dovuto al fatto che oltre alle perdite nella pila protocollare e nello switch si aggiunge anche un numero di collisioni non trascurabile con gli ACK di TCP, che causa una diminuzione dei bytes ricevuti e una conseguente diminuzione del goodput.

Caso G



In questo caso l'host H1 invia pacchetti ad H2 e H3 in parallelo. Nel caso di TCP ci aspettiamo una suddivisione equa tra H2 e H3 avendo quindi un goodput medio pari a 4,705 Mb/s ma nella realtà quest'ultimo è maggiore. Il motivo è dato dalla non equipartizione perfetta della velocità: infatti un host è leggermente più veloce dell'altro (trasmettono rispettivamente a circa 4.8 Mb/s e 4.6 Mb/s) e quindi finisce anticipatamente; fino a quel momento il goodput medio è esattamente pari a 4.7 Mb/s come atteso. Successivamente l'host rimanente ha un picco di goodput, che alza il suo goodput misurato a 4.7 Mb/s giustificando il risultato leggermente scostato dalla teoria.
Nel secondo caso, usando UDP, ci si aspetterebbe una suddivisione del goodput tra le due connessioni, avendo esattamente una perdita pari al 50%, nella realtà però troviamo come valore medio 4.5 Mb/s: ciò è dovuto alla perdita da parte di un host di due pacchetti di terminazione su 3, aggiungendo 2 secondi di ritardo che riducono il suo goodput (in assenza di questi ultimi si sarebbe ottenuto il valore corretto).

Per quanto riguarda il caso misto, nel grafico si può notare come l'host H1 che trasmette vada a dividere equamente la banda tra UDP (verde) e TCP (nero). Questo è dovuto al fatto che statisticamente i pacchetti delle due connessioni si alternano, e risulta quindi irrilevante il protocollo usato. Tuttavia TCP, al contrario di UDP, implementa un meccanismo di ritrasmissione in caso di pacchetti persi; di conseguenza, UDP finisce di trasmettere prima rispetto a TCP, aumentando il goodput di TCP nei secondi finali e portandolo dai 4,705 Mb/s attesi a 5,96 Mb/s.

Caso H

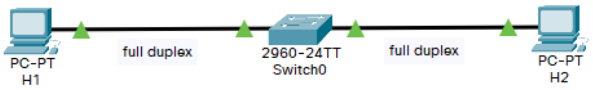
L'ultimo caso analizzato segue le stesse indicazioni del caso G utilizzando però almeno un canale di tipo half duplex. Il caso con UDP, come nei casi precedenti, non subisce sostanziali modifiche nel suo andamento, mentre utilizzando TCP si verificano le stesse problematiche descritte anche negli altri casi per quanto riguarda le collisioni.
Il caso peggiore analizzato in TCP risulta essere quello in cui il canale tra lo switch e i ricevitori è impostato su half duplex per la non desincronizzazione analizzata precedentemente. Inoltre, il caso con tutti i canali HD risulta molto più variabile rispetto ai casi B e F. Questo è dovuto al fatto che sono presenti tre punti in cui possono avvenire collisioni in maniera quasi casuale: al contrario degli altri casi non è detto che quando H2 o H3 vogliono inviare un ACK anche lo switch voglia inviargli un pacchetto, in quanto le sue trasmissioni coinvolgono in maniera alternata H2 e H3. Di conseguenza si possono avere più o meno collisioni in base al contesto, e questo causa un'indeterminatezza anche sui risultati (che oscillano tra 4.2 Mb/s e 4.6 Mb/s nei vari test effettuati). Questo spiega anche perché il caso peggiore in H sia comunque migliore dei casi peggiori in B e F: come detto precedentemente non sempre quando H2 o H3 vogliono mandare un ACK anche lo switch vuole mandargli un pacchetto, e ciò diminuisce la probabilità di collisione aumentando il goodput (che in questo caso è in media 4.4 Mb/s per ogni ricevitore).

Nel caso misto si verifica una situazione particolare (figura H.1 in appendice): ci si aspetterebbe infatti una situazione molto simile a quella trovata nel caso G ma con goodput minori a causa delle collisioni. Nella pratica effettivamente sono presenti molte collisioni dovute ai canali HD; mentre però UDP procede come al solito (in quanto non si cura delle perdite) TCP fatica molto. Addirittura, l'apertura della connessione con TCP richiede 0.7 secondi, e anche nell'invio dei pacchetti si nota come gli ACK arrivino sempre circa un secondo dopo all'invio dei pacchetti a cui sono riferiti. Questo ritardo degli ACK blocca di conseguenza TCP, in quanto non può far scorrere la sua finestra di trasmissione a causa della mancata ricezione degli ACK. Si ottiene quindi con UDP un valore molto vicino a quello atteso per una trasmissione con solo UDP (8.8 Mb/s) ma leggermente inferiore in quanto TCP continua comunque a inviare, seppur lentamente (oltre alle collisioni presenti a causa degli ACK). Quando UDP termina TCP si sblocca, e si riceve un grande numero di ACK consecutivi (di cui molti addirittura duplicati, a conferma dei grandi problemi di TCP) riferiti a tutti i pacchetti non ancora confermati che erano stati inviati. Al termine si ottiene per TCP una velocità stimata di 4.91 Mb/s, che è leggermente superiore al valore atteso (metà della velocità rispetto al caso TCP HD) in quanto per metà del tempo (ovvero quando c'è anche UDP) TCP trasmette molto poco ma trasmette comunque un po', aumentando di conseguenza il goodput finale ottenuto.

- Molto bene
- chiara e completa
- bene le parti opzionali 11/10

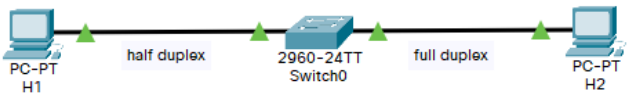
Appendice

A. Configurazione

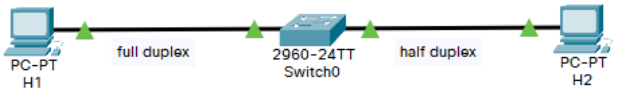


B. Configurazione

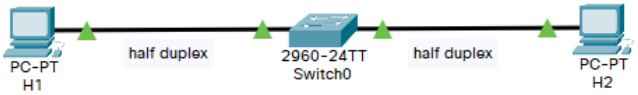
1.



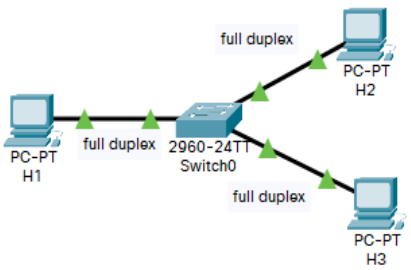
2.



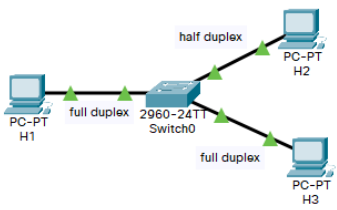
3.

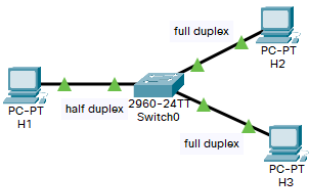


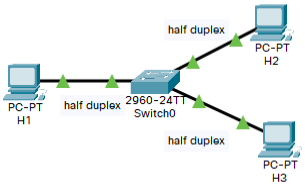
E. Configurazione



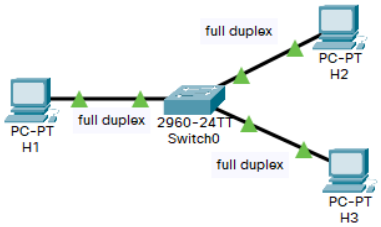
F. Configurazione



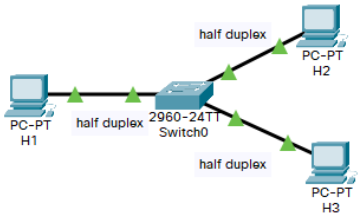


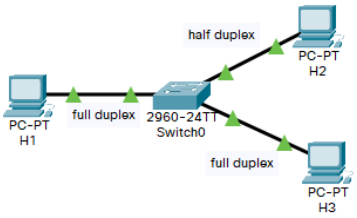


G. Configurazione



H. Configurazione





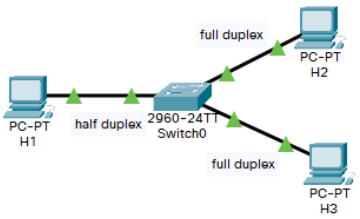
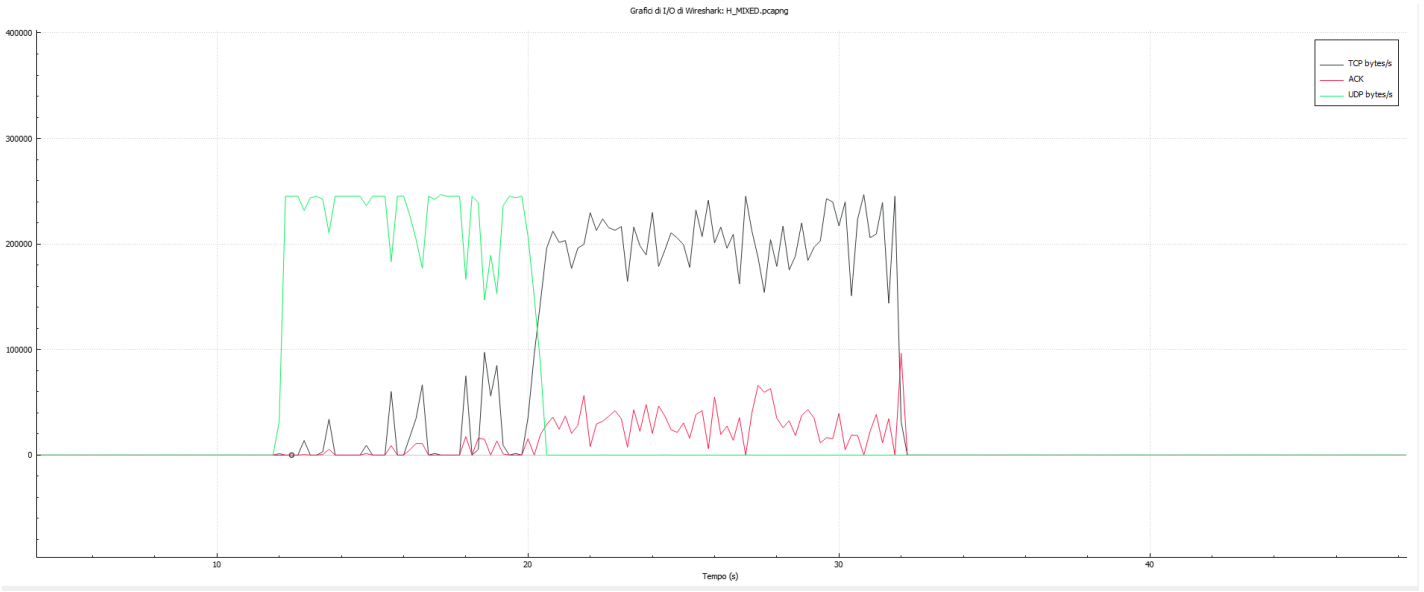


Grafico H.1



Verde UDP, nero TCP, rosso arrivo degli ACK (con fattore 100 per migliore visibilità)

Tabella di riferimento

Test	Average TCP Goodput per flow		Collision probability		Loss at the application layer	
	Prediction	Observed	Prediction	Observed	Prediction	Observed
A	9,41	9,4089	NO	NO	NO	NO
B1 (HF)	8,9	8,17	0,0004%	1,94%	NO	NO
B2 (FH)	8,9	6,94	0,0004%	31,03%	NO	NO
B3 (HH)	8,9	7,20	0,0004%	25,37%	NO	NO
E	4,705	4,71	NO	NO	NO	NO
F1 (FHF)	4,45	5	Circa come B1	2,40%	NO	NO
F2 (FFH)	<4,45	4,40	Circa come B2	1,06%	NO	NO
F3 (HHH)	<<4.45	3,55	Circa come B3	10,72%	NO	NO
G	4,705	4,75	NO	NO	NO	NO
H1 (FFH)	4,45	4,4	Circa come B1	1,81%	NO	NO
H2 (HHF)	<4.45	4,7	Circa come B2	6,60%	NO	NO
H3 (HHH)	<<4.45	4,2	Circa come B3	19,74%	NO	NO
H3 (HHH)	<<4.45	4,6	Circa come B3	25,16%	NO	NO
Test	Average UDP Goodput per flow		Collision probability		Loss at the application layer	
	Prediction	Observed	Prediction	Observed	Prediction	Observed
A1	9,57	9,568	NO	NO	NO	19,78%
A2	6,024	6,023	NO	NO	NO	4,10%
B1 (HF/FH)	9,57	9,56	NO	NO	Come A1	19,78%
B3 (HH)	9,57	9,56	NO	NO	Come A1	19,82%
E	4,785	4,81	NO	NO	50%	41,69%
F (FFH)	4,785	4,8	NO	NO	50%	41,59%
G	4,785	4,5	NO	NO	50%	41,65%
H1	4,785	4,7786	NO	NO	50%	41,23%
Test	Average UDP Goodput per flow		Average TCP Goodput per flow			
	Prediction	Observed	Prediction	Observed		
E	9,57	9,01	4,705	5,3		
F (HHH)	9,57	7,25	4,45	5,05		
G	4,785	4,87	4,705	5,96		
H	>4.785	8,81	<4.705	4,91		