POLITECNICO DI TORINO

A.A. 2019/2020

Report 3:

Performance test with off-the-shelf hardware

Student:

Simone Galota - 233727 – Gruppo 16

Teacher:

Prof. Marco Mellia

Course:

Laboratorio di Internet

Scenario A: Home network experiment

In questo primo scenario gli esperimenti vengono svolti nel seguente testbed: (figura in appendice)

- Un pc con Ubuntu-live in esecuzione da pen-drive (H1: 172.16.9.1/27);
- Un Mac su cui gira MacOS Catalina (H2: 172.16.9.1/27);
- Le due macchine sono connesse da un cavo ethernet cat.6; la connessione tramite Mac e cavo di rete avviene tramite USB con un adattatore USB-Ethernet.

Tramite il tool "ethtool", da Ubuntu, la velocità del collegamento fisico è stata impostata a 10Mb/s in full duplex con auto-negoziazione attiva. Per quanto riguarda l'adattatore, sicuramente fa Store&Forward. Si hanno comunque informazioni limitate riguardo al suo funzionamento, dato che è stato acquistato sul web con il criterio del minor costo da un produttore non propriamente noto. Quello che si sa è che l'USB è di tipo 3.0 e quindi dovrebbe supportare velocità nell'ordine dei Gb/s. Quindi, per i dati che si hanno, la velocità del collo di bottiglia, cioè la velocità della rete, è di 10 Mb/s.
Tuttavia, non si hanno informazioni specifiche riguardo alla velocità, al tipo di connessione e alla modalità di trasferimento dei dati tra CPU a scheda di rete.

Previsione goodput atteso

Si definisce Goodput il rapporto tra la dimensione dei dati utili a livello applicazione e il tempo di trasmissione (Byte/sec). Tale rapporto sarà sicuramente minore della velocità del canale fisico, poiché solo una percentuale dei dati effettivamente trasmessi è ritenuta utile. Vi sono infatti numerosi fattori (analizzati in seguito) che incidono nel calcolo del Goodput. Tra questi, gli overhead di ogni livello (che possono essere definiti dati di servizio) che vengono aggiunti al pacchetto trasmesso man mano che attraversa la pila protocollare.
Dunque, è importante parlare di efficienza di trasmissione. L'efficienza η è definita come il rapporto tra la dimensione dei dati utili a livello 7 (S) e la dimensione dei dati a livello 1 (S + n, dove n è la somma degli overhead di L4, L3, L2). Però, quello che ci interessa ai fini dell'esperimento in questione, è l'efficienza massima η_{max} .

- Nel caso di UDP, la dimensione di S che massimizza l'efficienza è: **S = 1500(MTU) – 8 (UDP) – 20 (IP) = 1472 byte.** Se S > 1472 il pacchetto viene segmentato, se è minore viene inviato interamente e non si ha la massima efficienza. Per cui, $\eta_{max,UDP} = \frac{1472}{1472 + 8\,(UDP) + 20\,(IP) + 38\,(ETH)} = 0{,}957$. Ciò significa che il 5% dei dati totali trasmessi non è utile a livello 7.

  Invece, nel caso di TCP, poiché è un protocollo stream-oriented, si trasmette solo per segmenti (prodotti automaticamente da TCP) di dimensione **MSS = 1500(MTU) – 20 (TCP) – 20 (IP) = 1460 byte (se non ci sono opzioni).** Quindi, la dimensione S che massimizza l'efficienza con TCP è **S = MSS**.
  $\eta_{max,TCP} = \frac{1460}{1460 + 20\,(TCP) + 20\,(IP) + 38\,(ETH)} = 0{,}949$

  È importante sapere che quello che fa TCP quando S < MSS, dipende da come è fatta l'interfaccia applicazione-TCP.
  Un' importante differenza con tra TCP e UDP è che:
  - UDP manda blocchi di messaggi (segmenta solo se necessario → S > MSS);
  - Invece, TCP prende messaggi e, qualsiasi sia la loro dimensione, si organizza per creare e inviare segmenti di lunghezza fissa MSS; di conseguenza in TCP il numero di "calls" al ricevitore è uguale al totale dei segmenti ricevuti.

- Per quanto concerne i fattori che possono ridurre il Goodput in questo scenario:
  - L1: non mi aspetto errori di trasmissione, perché si ha un collegamento lungo meno di 1 metro, in un ambiente controllato nel quale si trovano solo le due macchine per effettuare l'esperimento.
  - L2: non mi aspetto collisioni perché il cavo è configurato in modalità full duplex e quindi trasmissione e ricezione avvengono separatamente, inoltre il mezzo di trasmissione è condiviso solo dai due pc.
  - L2: a causa della differenza di velocità tra l'adattatore Eth-USB che fa S&F e il cavo Eth, potrebbe esserci congestione a valle dell'adattatore. Poiché i dati entrano da USB con una velocità nell'ordine dei Gb/s ed escono a 10 Mb/s. Ciò che influenza i pacchetti persi usando UDP è la quantita di dati che può contenere il buffer dell'adattatore.
  - L3: In questo scenario non c'è un router di livello 3.
  - L4: Per la natura affidabile del protocollo, con TCP non mi aspetto perdite di pacchetti, ma ci potrebbero essere ritrasmissioni che portano via del tempo. Ci sarà sicuramente una fase di segnalazione (3-way handshake) e i controlli di flusso e congestione che fanno sì che non si perdano pacchetti. Con UDP potrebbe succedere che si perdano perché è un protocollo message-oriented e best-effort.
  - L7: Non mi aspetto rallentamenti al livello dell'applicazione NTTCP, né da parte del sistema operativo, né da parte delle CPU, che comunque genera pacchetti nell'ordine dei Gb/s.

- Dunque, in relazione alle efficienze massime calcolate e alla velocità minima complessiva del collegamento fisico descritto, il Goodput che ci si aspetta dagli esperimenti svolti in questo scenario sarebbe:
  - $G_{TCP} = \eta_{max,TCP} * V_C = 0{,}949 * 10\ Mb/s = 9{,}49\ Mb/s$
  - $G_{UDP} = \eta_{max,UDP} * V_C = 0{,}957 * 10\ Mb/s = 9{,}57\ Mb/s$

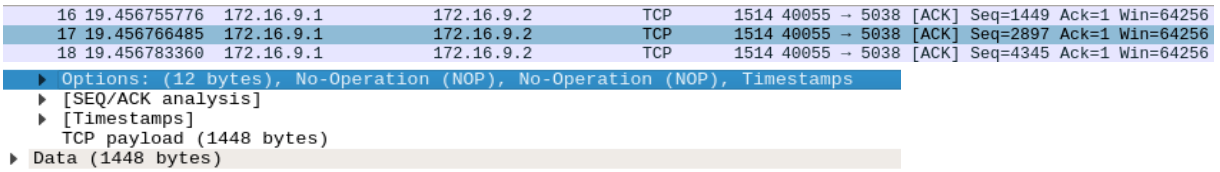
Gli esperimenti sono stati impostati in modo da garantire l'affidabilità e la consistenza delle misurazioni, e in maniera tale da avere una durata congrua a seconda del caso, così da rendere tempi di elaborazione, propagazione, arp-request e 3-way handshake trascurabili.

Test 1.a: Single flow – TCP: H1 → H2

Durante la prova le funzionalità avanzate della scheda di rete di H1 sono state disattivate, in modo che non influissero sulle misurazioni e sulle segmentazioni dei pacchetti. Il test è stato eseguito effettuando tentativi del tipo:

```
(base) laboratorio@laboratorio:~$ nttcp -T -n 100 -l 1000  172.16.9.2
     Bytes  Real s   CPU s Real-MBit/s  CPU-MBit/s   Calls  Real-C/s   CPU-C/s
l   100000    0.03    0.00     23.8351   1248.0499     100   2979.38  156006.2
1   100000    0.08    0.00      9.4138    782.0137     139   1635.64  135874.9
(base) laboratorio@laboratorio:~$ nttcp -T -n 1000 -l 5000  172.16.9.2
     Bytes  Real s   CPU s Real-MBit/s  CPU-MBit/s   Calls  Real-C/s   CPU-C/s
l  5000000    4.21    0.01      9.5097   3093.1024    1000    237.74   77327.6
1  5000000    4.25    0.04      9.4149    939.4302    3454    812.98   81119.8
(base) laboratorio@laboratorio:~$ nttcp -T -n 2048 -l 5000  172.16.9.2
     Bytes  Real s   CPU s Real-MBit/s  CPU-MBit/s   Calls  Real-C/s   CPU-C/s
l 10240000    8.65    0.03      9.4656   2961.1422    2048    236.64   74028.6
1 10240000    8.70    0.09      9.4149    897.0849    7073    812.88   77454.6
```

Dai risultati si evince come la misura riportata dal ricevitore sia la più affidabile, perché più vicina alla previsione. Mentre quella misurata al trasmettitore si discosta dal valore ipotizzato, a volte (se il tempo di esecuzione dell'applicazione è troppo breve) risulta anche maggiore della velocità fisica del canale. Tale anomalia è giustificata dal fatto che il tempo al trasmettitore è sottostimato, con una conseguente sovrastima della velocità. Questo perché non è un tempo di trasmissione, bensì un tempo di generazione dei pacchetti da parte della CPU, anche in considerazione del fatto che, in questo scenario, la CPU genera pacchetti molto più velocemente di quanto vengano trasmessi (*schema di calcolo dei tempi in appendice*). Maggiore sarà la durata dell'esecuzione, minore sarà la differenza tra tempo di generazione e tempo di trasmissione.
Tuttavia, a fronte di un bit rate previsto di 9,49 Mb/s, si ha una misura di 9,41 Mb/s. È possibile spiegare questa gap guardando la traccia dei pacchetti su wireshark.



Dalla cattura si nota che in ogni pacchetto TCP inviato sono presenti 12 bytes di opzioni (2 bytes di NOP e 10 bytes di timestamps). Per cui il payload del pacchetto, cioè la lunghezza S dei dati a livello 7, non è più di 1460 bytes, ma di 1448. Ne deriva che il calcolo dell'efficienza massima e del Goodput diventano:

$$\eta_{max,TCP} = \frac{1448}{1448 + 12\,(TCP-opt) + 20\,(TCP) + 20\,(IP) + 38\,(ETH)} = 0{,}941$$

$$G_{TCP} = \eta_{max,TCP} * V_C = 0{,}941 * 10\ Mb/s = 9{,}41\ Mb/s$$

Possiamo notare inoltre come la dimensione totale del pacchetto catturato da wireshark non sia di 1538, ma di 1514 byte. Ciò è dovuto al fatto che wireshark effettua la cattura a livello 2, tra LLC e MAC, perciò i 24 byte mancanti di intestazione ethernet sono aggiunti dallo strato più esterno (MAC), dopo che lo sniffer cattura il pacchetto.

Test 1.b: Single flow – TCP: H2 → H1

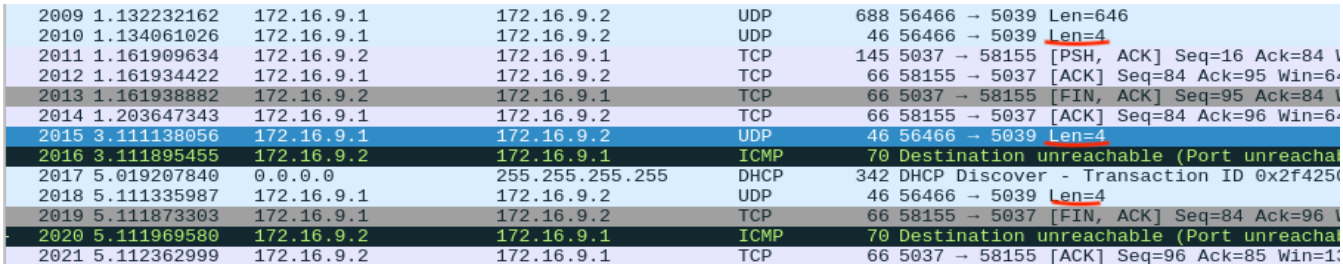
Per questo test possono essere fatte le stesse considerazioni del test precedente. Anche su H2 sono state disattivate le funzionalità avanzate della scheda di rete.
Non si evidenziano particolari anomalie, a parte un leggerissimo gap (0,01-0,02) rispetto al valore atteso 9,41 Mbit/s. Ciò è dovuto probabilmente al tempo perso per qualche ritrasmissione, a causa di qualche pacchetto perso a livello dell'adattatore.

Test 2.a: Single flow – UDP: H1 → H2

Come nel Test 1, prima di avviare l'esperimento le due macchine si scambiano dei pacchetti TCP per "mettersi d'accordo" sulle modalità (e.g. decisione della porta UDP da utilizzare per ricevere i pacchetti). Da evidenziare è il fatto che in UDP la velocità al ricevitore è calcolata solo sui dati effettivamente ricevuti e non su quelli persi (come è ovvio che sia).
Il risultato corrisponde alle previsioni (9,57 Mbit/s) se la dimensione di S = MSS e non si hanno nemmeno perdite di pacchetti. Però, facendo numerosi tentativi con vari valori di S, si nota che si verificano perdite di pacchetti quando S < 646 bytes o S > 1472 bytes. In particolare, se S > MSS, la perdita risulta più ingente, perché la PDU sarà soggetta a frammentazione. Pertanto, la probabilità di perdere un intero pacchetto aumenta. Infatti, basta che non sia trasmesso correttamente un solo frammento e tutto viene perso per impossibilità di corretta ricostruzione. Tale perdita è probabilmente dovuta a qualche tipo

di congestione al trasmettitore, nel collegamento tra CPU e scheda di rete (che non conosciamo); dato che, nella direzione di svolgimento dell'esperimento, non ci può essere congestione a livello dell'adattatore.

Osservando la colonna "Calls" dell'output, si nota che al trasmettitore vi sono sempre 3 chiamate in più rispetto al parametro 'n' passato al comando 'nttcp' che indica il numero di pacchetti da trasmettere. Perché? In TCP, sapendo che arriveranno tutti i pacchetti trasmessi, per misurare il tempo di fine trasmissione basta controllare quando arriverà l'ultimo (n-esimo) pacchetto, mentre in UDP non è detto che l'n-esimo pacchetti arrivi. Per cui il trasmettitore invia, ciascuno a distanza di 2 sec, 3 pacchetti UDP di lunghezza 4 byte, così da far sapere al ricevitore quando termina la finestra temporale delta_T di trasmissione. Ne vengono inviati 3 perché, se la probabilità di perdersene uno, da parte del RX, è di 10^-5, quella di perderne 3 è di 10^-5x10^-5x10^-5 = 10^-15. Che ingegneristicamente vuol dire zero. Di conseguenza è sicuro che, ricevendone uno dei tre, possa misurare l'istante di tempo correttamente.



2009	1.132232162	172.16.9.1	172.16.9.2	UDP	688 56466 → 5039 Len=646
2010	1.134061026	172.16.9.1	172.16.9.2	UDP	46 56466 → 5039 Len=4
2011	1.161909634	172.16.9.2	172.16.9.1	TCP	145 5037 → 58155 [PSH, ACK] Seq=16 Ack=84
2012	1.161934422	172.16.9.1	172.16.9.2	TCP	66 58155 → 5037 [ACK] Seq=84 Ack=95 Win=64
2013	1.161938882	172.16.9.2	172.16.9.1	TCP	66 5037 → 58155 [FIN, ACK] Seq=95 Ack=84
2014	1.203647343	172.16.9.1	172.16.9.2	TCP	66 58155 → 5037 [ACK] Seq=84 Ack=96 Win=64
2015	3.111138056	172.16.9.1	172.16.9.2	UDP	46 56466 → 5039 Len=4
2016	3.111895455	172.16.9.2	172.16.9.1	ICMP	70 Destination unreachable (Port unreacha
2017	5.019207840	0.0.0.0	255.255.255.255	DHCP	342 DHCP Discover - Transaction ID 0x2f425
2018	5.111335987	172.16.9.1	172.16.9.2	UDP	46 56466 → 5039 Len=4
2019	5.111873303	172.16.9.1	172.16.9.2	TCP	66 58155 → 5037 [FIN, ACK] Seq=84 Ack=96
2020	5.111969580	172.16.9.2	172.16.9.1	ICMP	70 Destination unreachable (Port unreacha
2021	5.112362999	172.16.9.2	172.16.9.1	TCP	66 5037 → 58155 [ACK] Seq=96 Ack=85 Win=1

Test 2.b: Single flow – UDP: H2 → H1

Anche in questo caso, se S = MSS, la velocità al ricevitore è quella ipotizzata. Ciò che invece è da evidenziare è una consistente perdita di pacchetti se l'esperimento ha una durata ragionevole (5-10s). L'esperimento svolto in questa direzione (H2 → H1) è sicuramente soggetto a un collo di bottiglia causato dalla presenza dell'adattatore ETH-USB. Perché, come già accennato, i pacchetti vengono prodotti dalla CPU di H2 e trasmessi all'adattatore con velocità dell'ordine dei Gb/s. Quest'ultimo fa S&F per ritrasmettere, stavolta con velocità di 10 Mb/s, ad H1. Quindi è evidente che, una volta che la coda di pacchetti è piena e ricevendone molti di più di quelli trasmessi, quelli in eccesso vengano persi.

Tuttavia, è importante sottolineare che il valore "CPU s" sia minore di "Real s", e che "Real-Mbit/s" sia minore di "CPU-Mbit/s", altrimenti il collo di bottiglia non è nella rete, ma nell'applicazione.

Scenario B: Internet experiment

Nel secondo scenario gli esperimenti vengono svolti nel seguente testbed:

- Un Pc con Ubuntu live in esecuzione da pen-drive che funge da client;
- Un server connesso in rete a 1 Gb/s dal Politecnico di Torino, denominato bigdatadb.polito.it.

Il client è collegato a internet tramite un access-point domestico con un cavo ethernet cat.6a settato a 100 Mb/s. La connessione è di tipo FTTC, la cui banda in uplink è intorno ai 21 Mb/s, in downlink intorno ai 54 Mb/s. Tali valori non sono comunque da ritenere fissi e affidabili al 100%, poiché la qualità della linea dipende da vari fattori, quali il margine di rumore e l'attenuazione, sia nel tratto tra DSLAM←→Modem che nell'impianto dell'abitazione. A seguito di alcune prove, è possibile ipotizzare che i valori di up e down siano soggetti ad un'oscillazione di +/- 1Mb/s.
Tutti i test sono stati effettuati con la presenza del solo client interessato all'interno della LAN, così da lasciare tutta la banda per il pc su cui gira Ubuntu.

Previsione goodput atteso

Per quanto riguarda la previsione del goodput, bisogna considerare il fatto che gli esperimenti sono svolti in uno scenario incerto e non totalmente sotto controllo, in quanto non si sa con precisione a cosa i pacchetti vadano incontro una volta usciti dalla LAN. Dunque, abbiamo solo informazioni inerenti al nostro testbed, nel quale la velocità del collo di bottiglia è sicuramente quella imposta dal router sia in up che in down. Nella LAN non ci aspettiamo collisioni, congestioni o errori di trasmissione. Per il calcolo delle efficienze massime, che si calcolano allo stesso modo dello scenario precedente, considereremo solo i dati del nostro setup.

$$\eta_{max,UDP} = \frac{1472}{1472 + 8\,(UDP) + 20\,(IP) + 38\,(ETH)} = 0{,}957$$

$$\eta_{max,TCP} = \frac{1448}{1448 + 12(TCP - opt) + 20\,(TCP) + 20\,(IP) + 38\,(ETH)} = 0{,}941$$

Dunque, il goodput atteso è:

- TCP:
  - In trasmissione, quindi in upload: $G_{TCP} = \eta_{max,TCP} * V_{UP} = 0{,}941 * 21\,Mb/s = 19{,}76\,Mb/s$
  - In ricezione, quindi in download: $G_{TCP} = \eta_{max,TCP} * V_{DOWN} = 0{,}941 * 54\,Mb/s = 50{,}81\,Mb/s$

- UDP:
  - In trasmissione, quindi in upload: $G_{UDP} = \eta_{max,UDP} * V_{UP} = 0{,}957 * 21\,Mb/s = 20{,}10\,Mb/s$
  - In ricezione, quindi in download: $G_{UDP} = \eta_{max,UDP} * V_{DOWN} = 0{,}957 * 54\,Mb/s = 51{,}68\,Mb/s$

Test 1.a: Single flow – TCP: H1 → H2

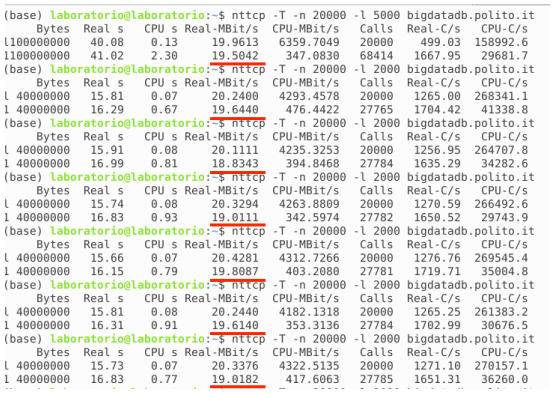
Per lo svolgimento di questo test, nel calcolo dell'efficienza massima si è già tenuto conto dei 12 bytes di opzioni presenti in ogni pacchetto TCP inviato.
Il test è stato eseguito 7 volte: 6 volte con durata di circa 16s, 1 volta con durata di 40s. Le velocità misurate variano leggermente dal valore atteso, ma si possono considerare comunque affidabili, in relazione anche ai fattori variabili che incidono sulla qualità della linea citati sopra.

Valore minimo: 18,83 Mb/s – Valore massimo: 19,80 Mb/s – Valore medio: 19,34 Mb/s

Test 1.b: Single flow – TCP: H2 → H1

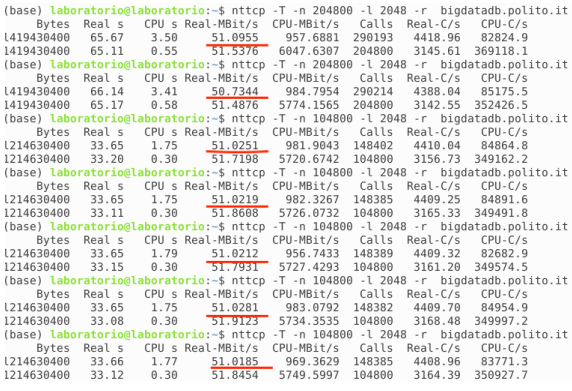
Per effettuare questo test con l'applicazione nttcp è stato necessario aprire in entrata la porta TCP/UDP 5038 nella quale il server avvia la trasmissione dei dati al client, in modo da consentire al NAT della nostra LAN di inoltrare il traffico della porta scelta all'ip corretto.
Come nel caso precedente, possiamo fare le stesse considerazioni sui valori ottenuti dal test svolto. Da notare che la durata di questo test, per avere misure affidabili e consistenti, è di 30 s almeno (rispetto ai 15s del test 1.a). È giustificata, probabilmente, da un numero più alto di ritrasmissioni dovute alla congestione causata dal server che invia dati con velocità di 1 Gb/s e il nostro access-point riceve a 54 Mb/s.

Valore minimo: 50,73 Mb/s – Valore massimo: 51,09 Mb/s – Valore medio: 50,98 Mb/s



```
(base) laboratorio@laboratorio:~$ nttcp -T -n 20000 -l 5000 bigdatadb.polito.it
     Bytes  Real s   CPU s Real-MBit/s  CPU-MBit/s   Calls  Real-C/s   CPU-C/s
l100000000   40.08    0.13     19.9613   6359.7049   20000    499.03  158992.6
l100000000   41.02    2.30     19.5042    347.0830   68414   1667.95   29681.7
(base) laboratorio@laboratorio:~$ nttcp -T -n 20000 -l 2000 bigdatadb.polito.it
     Bytes  Real s   CPU s Real-MBit/s  CPU-MBit/s   Calls  Real-C/s   CPU-C/s
l 40000000   15.81    0.07     20.2400   4293.4578   20000   1265.00  268341.1
l 40000000   16.29    0.67     19.6440    476.4422   27765   1704.42   41338.8
(base) laboratorio@laboratorio:~$ nttcp -T -n 20000 -l 2000 bigdatadb.polito.it
     Bytes  Real s   CPU s Real-MBit/s  CPU-MBit/s   Calls  Real-C/s   CPU-C/s
l 40000000   15.91    0.08     20.1111   4235.3253   20000   1256.95  264707.8
l 40000000   16.99    0.81     18.8343    394.8468   27784   1635.29   34282.6
(base) laboratorio@laboratorio:~$ nttcp -T -n 20000 -l 2000 bigdatadb.polito.it
     Bytes  Real s   CPU s Real-MBit/s  CPU-MBit/s   Calls  Real-C/s   CPU-C/s
l 40000000   15.74    0.08     20.3294   4263.8809   20000   1270.59  266492.6
l 40000000   16.83    0.93     19.0111    342.5974   27782   1650.52   29743.9
(base) laboratorio@laboratorio:~$ nttcp -T -n 20000 -l 2000 bigdatadb.polito.it
     Bytes  Real s   CPU s Real-MBit/s  CPU-MBit/s   Calls  Real-C/s   CPU-C/s
l 40000000   15.66    0.07     20.4281   4312.7266   20000   1276.76  269545.4
l 40000000   16.15    0.79     19.8087    403.2080   27781   1719.71   35004.8
(base) laboratorio@laboratorio:~$ nttcp -T -n 20000 -l 2000 bigdatadb.polito.it
     Bytes  Real s   CPU s Real-MBit/s  CPU-MBit/s   Calls  Real-C/s   CPU-C/s
l 40000000   15.81    0.08     20.2440   4182.1318   20000   1265.25  261383.2
l 40000000   16.31    0.91     19.6140    353.3136   27784   1702.99   30676.5
(base) laboratorio@laboratorio:~$ nttcp -T -n 20000 -l 2000 bigdatadb.polito.it
     Bytes  Real s   CPU s Real-MBit/s  CPU-MBit/s   Calls  Real-C/s   CPU-C/s
l 40000000   15.73    0.07     20.3376   4322.5135   20000   1271.10  270157.1
l 40000000   16.83    0.77     19.0182    417.6063   27785   1651.31   36260.0
```

TEST 1.a



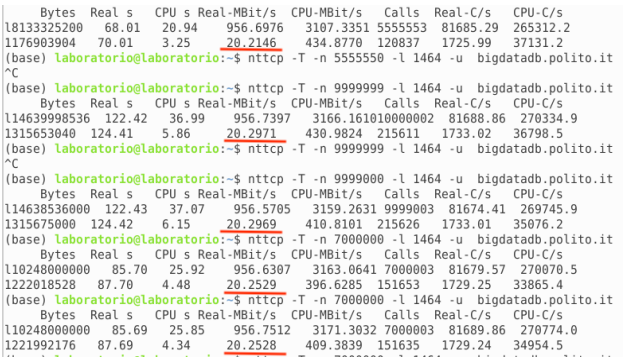
```
(base) laboratorio@laboratorio:~$ nttcp -T -n 204800 -l 2048 -r  bigdatadb.polito.it
     Bytes  Real s   CPU s Real-MBit/s  CPU-MBit/s   Calls  Real-C/s   CPU-C/s
l419430400   65.67    3.50     51.0955    957.6881  290193   4418.96   82824.9
l419430400   65.11    0.55     51.5376   6047.6307  204800   3145.61  369118.1
(base) laboratorio@laboratorio:~$ nttcp -T -n 204800 -l 2048 -r  bigdatadb.polito.it
     Bytes  Real s   CPU s Real-MBit/s  CPU-MBit/s   Calls  Real-C/s   CPU-C/s
l419430400   66.14    3.41     50.7344    984.7954  290214   4388.04   85175.5
l419430400   65.17    0.58     51.4876   5774.1565  204800   3142.55  352426.5
(base) laboratorio@laboratorio:~$ nttcp -T -n 104800 -l 2048 -r  bigdatadb.polito.it
     Bytes  Real s   CPU s Real-MBit/s  CPU-MBit/s   Calls  Real-C/s   CPU-C/s
l214630400   33.65    1.75     51.0251    981.9043  148402   4410.04   84864.8
l214630400   33.20    0.30     51.7198   5720.6742  104800   3156.73  349162.2
(base) laboratorio@laboratorio:~$ nttcp -T -n 104800 -l 2048 -r  bigdatadb.polito.it
     Bytes  Real s   CPU s Real-MBit/s  CPU-MBit/s   Calls  Real-C/s   CPU-C/s
l214630400   33.65    1.75     51.0219    982.3267  148385   4409.25   84891.6
l214630400   33.11    0.30     51.8608   5726.0732  104800   3165.33  349491.8
(base) laboratorio@laboratorio:~$ nttcp -T -n 104800 -l 2048 -r  bigdatadb.polito.it
     Bytes  Real s   CPU s Real-MBit/s  CPU-MBit/s   Calls  Real-C/s   CPU-C/s
l214630400   33.65    1.79     51.0212    956.7433  148389   4409.32   82682.9
l214630400   33.15    0.30     51.7931   5727.4293  104800   3161.20  349574.5
(base) laboratorio@laboratorio:~$ nttcp -T -n 104800 -l 2048 -r  bigdatadb.polito.it
     Bytes  Real s   CPU s Real-MBit/s  CPU-MBit/s   Calls  Real-C/s   CPU-C/s
l214630400   33.65    1.75     51.0281    983.0792  148382   4409.70   84954.9
l214630400   33.08    0.30     51.9123   5734.3535  104800   3168.48  349997.2
(base) laboratorio@laboratorio:~$ nttcp -T -n 104800 -l 2048 -r  bigdatadb.polito.it
     Bytes  Real s   CPU s Real-MBit/s  CPU-MBit/s   Calls  Real-C/s   CPU-C/s
l214630400   33.66    1.77     51.0185    969.3629  148385   4408.96   83771.3
l214630400   33.12    0.30     51.8454   5749.5997  104800   3164.39  350927.7
```

TEST 1.b

Test 2.a: Single flow – UDP: H1 → H2

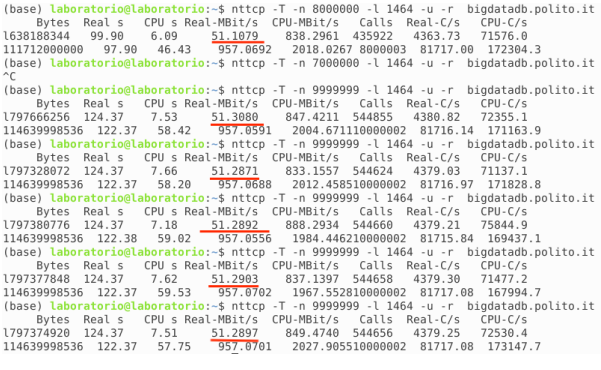
Da una prova preliminare, settando la dimensione dei dati S = MSS= 1472 bytes, si hanno riscontri anomali, in particolare il numero di pacchetti arrivati a destinazione è troppo basso (in ricezione, a volte anche 0), e di conseguenza si hanno misure sottostimate, in totale disaccordo con la previsione.
Tale anomalia non è stata riscontrata con TCP, dato che, come detto, TCP riceve dal livello applicazione un pacchetto di qualsiasi dimensione, e produce lui stesso i segmenti.
Il motivo di questo problema risiede nel fatto che al DSLAM la MTU di livello 2 non è di 1500 bytes, bensì di 1492. Conseguentemente, la MSS non è più di 1472, ma di 1464 bytes e quindi UDP segmenta e la probabilità di perdere tutto il pacchetto è del 50% (in questo caso specifico, con 2 segmenti). Nel test con TCP non era 1448, bensì 1440 bytes. Tuttavia, non variano i rapporti che determinano le efficienze massime nei due protocolli.

In questo test ci aspettiamo perdite, quindi per avere una misura affidabile si è optato di far durare il test almeno 60/70s (alcuni anche di 120s), inviando un'ingente quantità di dati.
In 5 tentativi, si registra un valore compatibile con la previsione di 20,2 Mb/s, il che vuol dire che al momento dei test vi era una velocità effettiva in uplink di 21,2 Mb/s: 0,957*21,2= 20,2 Mb/s. Può essere considerato comunque un risultato consistente.



```
     Bytes  Real s   CPU s Real-MBit/s  CPU-MBit/s   Calls  Real-C/s   CPU-C/s
l8133325200   68.01   20.94    956.6976   3107.3351 5555553  81685.29  265312.2
l176903904   70.01    3.25     20.2146    434.8770  120837   1725.99   37131.2
(base) laboratorio@laboratorio:~$ nttcp -T -n 5555550 -l 1464 -u  bigdatadb.polito.it
^C
(base) laboratorio@laboratorio:~$ nttcp -T -n 9999999 -l 1464 -u  bigdatadb.polito.it
     Bytes  Real s   CPU s Real-MBit/s  CPU-MBit/s   Calls  Real-C/s   CPU-C/s
l14639998536  122.42   36.99    956.7397   3166.161010000002  81688.86  270334.9
l1315653040  124.41    5.86     20.2971    430.9824  215611   1733.02   36798.5
(base) laboratorio@laboratorio:~$ nttcp -T -n 9999999 -l 1464 -u  bigdatadb.polito.it
^C
(base) laboratorio@laboratorio:~$ nttcp -T -n 9999000 -l 1464 -u  bigdatadb.polito.it
     Bytes  Real s   CPU s Real-MBit/s  CPU-MBit/s   Calls  Real-C/s   CPU-C/s
l14638536000  122.43   37.07    956.5705   3159.2631 9999003  81674.41  269745.9
l1315675000  124.42    6.15     20.2969    410.8101  215626   1733.01   35076.2
(base) laboratorio@laboratorio:~$ nttcp -T -n 7000000 -l 1464 -u  bigdatadb.polito.it
     Bytes  Real s   CPU s Real-MBit/s  CPU-MBit/s   Calls  Real-C/s   CPU-C/s
l10248000000   85.70   25.92    956.6307   3163.0641 7000003  81679.57  270070.5
l1222018528   87.70    4.48     20.2529    396.6285  151653   1729.25   33865.4
(base) laboratorio@laboratorio:~$ nttcp -T -n 7000000 -l 1464 -u  bigdatadb.polito.it
     Bytes  Real s   CPU s Real-MBit/s  CPU-MBit/s   Calls  Real-C/s   CPU-C/s
l10248000000   85.69   25.85    956.7512   3171.3032 7000003  81689.86  270774.0
l1221992176   87.69    4.34     20.2528    409.3839  151635   1729.24   34954.5
```

TEST 2.a



```
(base) laboratorio@laboratorio:~$ nttcp -T -n 8000000 -l 1464 -u -r  bigdatadb.polito.it
     Bytes  Real s   CPU s Real-MBit/s  CPU-MBit/s   Calls  Real-C/s   CPU-C/s
l638188344   99.90    6.09     51.1079    838.2961  435922   4363.73   71576.0
111712000000   97.90   46.43    957.0692   2018.0267 8000003  81717.00  172304.3
(base) laboratorio@laboratorio:~$ nttcp -T -n 7000000 -l 1464 -u -r  bigdatadb.polito.it
^C
(base) laboratorio@laboratorio:~$ nttcp -T -n 9999999 -l 1464 -u -r  bigdatadb.polito.it
     Bytes  Real s   CPU s Real-MBit/s  CPU-MBit/s   Calls  Real-C/s   CPU-C/s
l797666256  124.37    7.53     51.3080    847.4211  544855   4380.82   72355.1
114639998536  122.37   58.42    957.0591   2004.671110000002  81716.14  171163.9
(base) laboratorio@laboratorio:~$ nttcp -T -n 9999999 -l 1464 -u -r  bigdatadb.polito.it
     Bytes  Real s   CPU s Real-MBit/s  CPU-MBit/s   Calls  Real-C/s   CPU-C/s
l797328072  124.37    7.66     51.2871    833.1557  544624   4379.03   71137.1
114639998536  122.37   58.20    957.0688   2012.458510000002  81716.97  171828.8
(base) laboratorio@laboratorio:~$ nttcp -T -n 9999999 -l 1464 -u -r  bigdatadb.polito.it
     Bytes  Real s   CPU s Real-MBit/s  CPU-MBit/s   Calls  Real-C/s   CPU-C/s
l797380776  124.37    7.18     51.2892    888.2934  544660   4379.21   75844.9
114639998536  122.38   59.02    957.0556   1984.446210000002  81715.84  169437.1
(base) laboratorio@laboratorio:~$ nttcp -T -n 9999999 -l 1464 -u -r  bigdatadb.polito.it
     Bytes  Real s   CPU s Real-MBit/s  CPU-MBit/s   Calls  Real-C/s   CPU-C/s
l797377848  124.37    7.62     51.2903    837.1397  544658   4379.30   71477.2
114639998536  122.37   59.53    957.0702   1967.552810000002  81717.08  167994.7
(base) laboratorio@laboratorio:~$ nttcp -T -n 9999999 -l 1464 -u -r  bigdatadb.polito.it
     Bytes  Real s   CPU s Real-MBit/s  CPU-MBit/s   Calls  Real-C/s   CPU-C/s
l797374920  124.37    7.51     51.2897    849.4740  544656   4379.25   72530.4
114639998536  122.37   57.75    957.0701   2027.905510000002  81717.08  173147.7
```

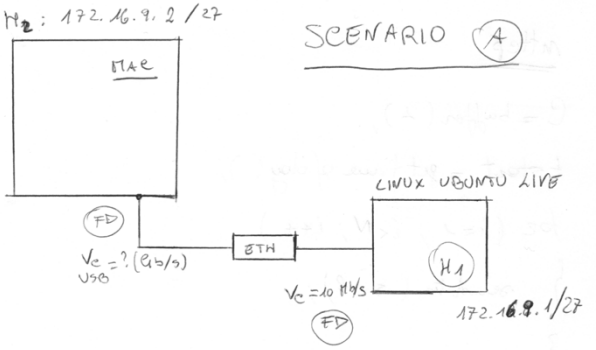
TEST 2.b

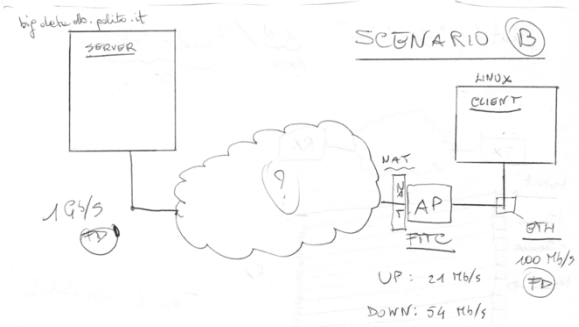
Test 2.b: Single flow – UDP: H2 → H1

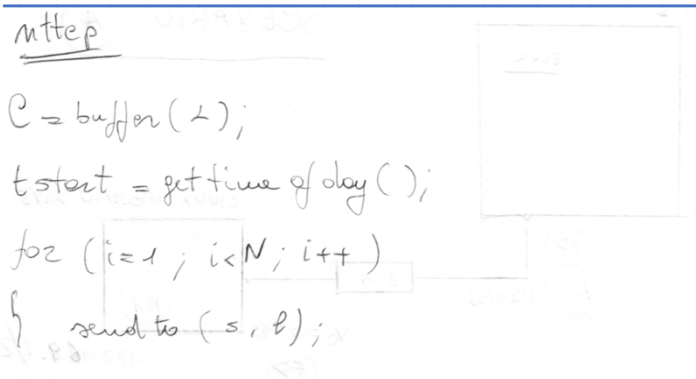
In quest'ultimo caso sono stati effettuati 5 test di durata 120s circa. Il risultato può considerarsi consistente, è stata misurata una velocità di 51,29 Mb/s in tutte le prove. Quest'ultimo valore differisce leggermente dal goodput ipotizzato, ma può considerarsi comunque affidabile, per i motivi già ampiamente citati (probabilmente la V_{DOWN} non era proprio 54 Mb/s, ma qualcosa in meno 53,6 Mb/). Come nel caso del test 2.a ci potrebbe essere congestione all'ingresso del router (ma anche nel percorso effettuato dai pacchetti) che, nel caso di UDP, determina perdita e non ritrasmissione.

In conclusione, possiamo affermare che, in linea di massima, previsione e valori sperimentali misurati nelle varie prove siano coerenti tra di loro.

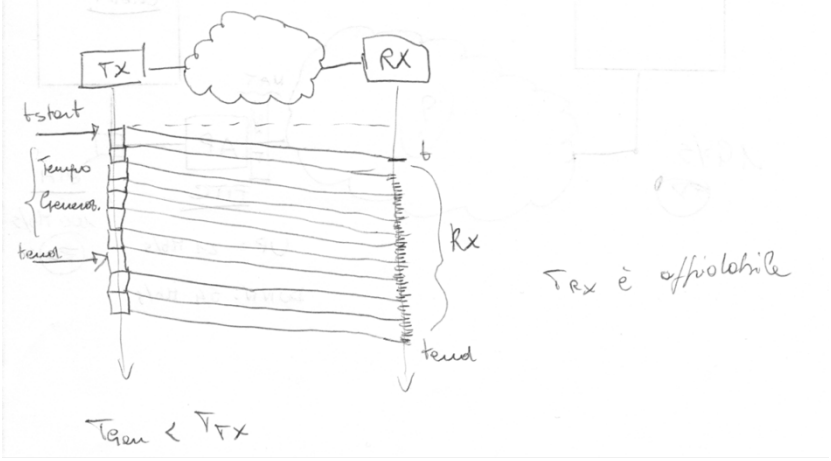
Appendice:







Script in pseudo-codice che descrive quale potrebbe essere il meccanismo per il calcolo della finestra temporale ΔT su cui nttcp calcola la velocità di trasmissione.



Schema rappresentativo dello script