Eseguendo il comando si ottiene un riscontro di quanto ipotizzato al punto precedente. In particolare eseguendo 10 test si ottengono i risultati in Figura 1. Si può notare che il bitrate del sender è legermente sovrastimato rispetto a quello del receiver: ciò accade perchè i $\Delta T$ di sender e receiver sono differenti, il primo calcola il tempo trascorso dall'invio del primo pacchetto all'invio dell'ultimo, mentre il secondo usa tempi più grandi poichè deve attendere la ricezione. Si ottiene quindi una velocità media di 9.58 Mbit/s al sender e 9.39 Mbit/s al receiver.

```
lo@ubu:~/Downloads$ cat TCPPC.dat | grep " sender\| receiver"
[  5]   0.00-10.00  sec  11.5 MBytes  9.62 Mbits/sec    0             sender
[  5]   0.00-10.00  sec  11.2 MBytes  9.41 Mbits/sec                  receiver
[  5]   0.00-10.00  sec  11.5 MBytes  9.62 Mbits/sec    0             sender
[  5]   0.00-10.00  sec  11.2 MBytes  9.40 Mbits/sec                  receiver
[  5]   0.00-10.00  sec  11.5 MBytes  9.62 Mbits/sec    0             sender
[  5]   0.00-10.00  sec  11.2 MBytes  9.39 Mbits/sec                  receiver
[  5]   0.00-10.00  sec  11.5 MBytes  9.62 Mbits/sec    0             sender
[  5]   0.00-10.00  sec  11.2 MBytes  9.40 Mbits/sec                  receiver
[  5]   0.00-10.00  sec  11.4 MBytes  9.57 Mbits/sec    0             sender
[  5]   0.00-10.00  sec  11.2 MBytes  9.40 Mbits/sec                  receiver
[  5]   0.00-10.00  sec  11.4 MBytes  9.57 Mbits/sec    0             sender
[  5]   0.00-10.00  sec  11.2 MBytes  9.39 Mbits/sec                  receiver
[  5]   0.00-10.00  sec  11.3 MBytes  9.52 Mbits/sec    0             sender
[  5]   0.00-10.00  sec  11.1 MBytes  9.35 Mbits/sec                  receiver
[  5]   0.00-10.00  sec  11.5 MBytes  9.62 Mbits/sec    0             sender
[  5]   0.00-10.00  sec  11.2 MBytes  9.42 Mbits/sec                  receiver
[  5]   0.00-10.00  sec  11.3 MBytes  9.47 Mbits/sec    0             sender
[  5]   0.00-10.00  sec  11.1 MBytes  9.33 Mbits/sec                  receiver
[  5]   0.00-10.00  sec  11.4 MBytes  9.57 Mbits/sec    0             sender
[  5]   0.00-10.00  sec  11.2 MBytes  9.41 Mbits/sec                  receiver
```

Figura 1

## -1.1 Test TCP - Singolo flusso, half duplex

In questo caso è possibile predirre l'efficenza della rete tenendo in considerazione il funzionamento di TCP: il protocollo prevede infatti l'invio di un ACK di dimensioni 20+20+38, una volta che il receiver ha ricevuto il pacchetto.

$$\eta = \frac{MSS}{MSS + 20 + 32 + 38 + (20 + 20 + 38)} \simeq 0.89 \tag{1}$$

Essendo il canale condiviso per trasmissione e ricezione possono verificarsi collisioni a livello fisico che porterebbero a una diminuzione del goodput calcolato. Eseguendo il test si ottengono le seguenti velocità di trasmissione

Come ipotizzato la velocità è piu' lenta di quella calcolata per via delle collisioni, gestite direttamente dalla scheda di rete, visualizzabili con *ifconfig* prima e dopo il comando *iperf3*

## -1.2 Test TCP - Singolo fulsso, full duplex, reverse mode

```
Connecting to host 192.168.1.196, port 5201
Reverse mode, remote host 192.168.1.196 is sending
[  5] local 192.168.1.210 port 49392 connected to 192.168.1.196 port 5201
[ ID] Interval           Transfer     Bitrate
[  5]   0.00-1.00   sec   847 KBytes  6.94 Mbits/sec
[  5]   1.00-2.00   sec  1.13 MBytes  9.46 Mbits/sec
[  5]   2.00-3.00   sec  1.13 MBytes  9.47 Mbits/sec
[  5]   3.00-4.00   sec  1.13 MBytes  9.48 Mbits/sec
[  5]   4.00-5.00   sec  1.13 MBytes  9.47 Mbits/sec
[  5]   5.00-6.00   sec  1.13 MBytes  9.48 Mbits/sec
[  5]   6.00-7.00   sec  1.13 MBytes  9.48 Mbits/sec
[  5]   7.00-8.00   sec  1.13 MBytes  9.49 Mbits/sec
[  5]   8.00-9.00   sec  1.13 MBytes  9.47 Mbits/sec
[  5]   9.00-10.00  sec  1.13 MBytes  9.49 Mbits/sec
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate
[  5]   0.00-10.00  sec  11.1 MBytes  9.33 Mbits/sec                  sender
[  5]   0.00-10.00  sec  11.0 MBytes  9.22 Mbits/sec                  receiver
```

Figura 2: TCP PC -R

Facendo riferimento al setup precendente si esegue il comando *iperf3 -c 192.168.1.196 -R* al client, in questo modo è il client a mandare un primo messaggio al server per richiedere la connessione inversa, scambiando così i ruoli di client e server. In questo scenario la velocità del client è molto maggiore di quella del server, per questo motivo interverranno il controllo di congestione e di flusso per adattare, nei primi istanti della connessione, la velocità di *sender* e *receiver* per evitare congestione e ritrasmissioni che diminuirebbero il goodput della rete.

In figura 2 e' possibile notare l'intervento del controllo di congestione guardando la quantita' di dati scambiati nel primo secondo rispetto agli istanti successivi. Eseguendo il comando 10 volte e' possibile ricavare le velocità medie di sender e receiver, rispettivamente 9.33 e 9.20.

## -1.3 Test UDP - Singolo flusso, full duplex

L'efficenza di UDP è massima se si riescono a scambiare la maggiore quantità di bit senza che avvenga frammentazione del pacchetto, ovvero quando $S + 8 + 20 = MTU$, quindi $S_{MAX} = 1472$

Dalla (2) si ricava che

$$\eta = \frac{1472}{1472 + 8 + 20 + 38} \simeq 0,957 \rightarrow V_{TXMAX} = C \cdot \eta = 9,57Mb/s \tag{2}$$

Nel caso del singolo flusso full duplex non ci aspettiamo problemi a livello fisico, nè problemi di congestione della rete. Poichè UDP è un protocollo inaffidabile e non aspetta conferme dall'interlocutore, l'efficenza per connessioni half duplex rimane la stessa di quella calcolata precedentemente. La velocità ottenuta risulta essere maggiore di quella TCP per via delle dimensioni ridotte dell'header UDP

```
lo@ubu:~/Documents/LabInt/Lab4$ cat UDP_PC.dat | grep " sender\| receiver"
[  5]   0.00-10.00  sec  15.0 MBytes  12.6 Mbits/sec  0.000 ms  0/10760 (0%)  sender
[  5]   0.00-10.00  sec  11.4 MBytes  9.55 Mbits/sec  2.044 ms  2533/10708 (24%)  receiver
[  5]   0.00-10.00  sec  15.0 MBytes  12.6 Mbits/sec  0.000 ms  0/10760 (0%)  sender
[  5]   0.00-10.00  sec  11.4 MBytes  9.55 Mbits/sec  2.015 ms  2534/10710 (24%)  receiver
[  5]   0.00-10.00  sec  15.0 MBytes  12.6 Mbits/sec  0.000 ms  0/10760 (0%)  sender
[  5]   0.00-10.00  sec  11.4 MBytes  9.55 Mbits/sec  2.038 ms  2533/10708 (24%)  receiver
[  5]   0.00-10.00  sec  15.0 MBytes  12.6 Mbits/sec  0.000 ms  0/10760 (0%)  sender
[  5]   0.00-10.00  sec  11.4 MBytes  9.55 Mbits/sec  2.033 ms  2534/10710 (24%)  receiver
[  5]   0.00-10.00  sec  15.0 MBytes  12.6 Mbits/sec  0.000 ms  0/10760 (0%)  sender
[  5]   0.00-10.00  sec  11.4 MBytes  9.55 Mbits/sec  2.021 ms  2532/10706 (24%)  receiver
[  5]   0.00-10.00  sec  15.0 MBytes  12.6 Mbits/sec  0.000 ms  0/10760 (0%)  sender
[  5]   0.00-10.00  sec  11.4 MBytes  9.55 Mbits/sec  1.820 ms  2534/10710 (24%)  receiver
[  5]   0.00-10.00  sec  15.0 MBytes  12.6 Mbits/sec  0.000 ms  0/10760 (0%)  sender
[  5]   0.00-10.00  sec  11.4 MBytes  9.55 Mbits/sec  2.010 ms  2533/10708 (24%)  receiver
[  5]   0.00-10.00  sec  15.0 MBytes  12.6 Mbits/sec  0.000 ms  0/10760 (0%)  sender
[  5]   0.00-10.00  sec  11.4 MBytes  9.55 Mbits/sec  1.978 ms  2533/10708 (24%)  receiver
[  5]   0.00-10.00  sec  15.0 MBytes  12.6 Mbits/sec  0.000 ms  0/10750 (0%)  sender
[  5]   0.00-10.00  sec  11.4 MBytes  9.54 Mbits/sec  2.118 ms  2531/10703 (24%)  receiver
[  5]   0.00-10.00  sec  15.0 MBytes  12.6 Mbits/sec  0.000 ms  0/10760 (0%)  sender
[  5]   0.00-10.00  sec  11.4 MBytes  9.55 Mbits/sec  1.751 ms  2534/10710 (24%)  receiver
```

Figura 3: UDP PC

A conferma della maggiore efficenza del protocllo UDP eseguendo i 10 test si ottiene una velocità media di 12.6 al sender e 9.56 al receiver. Poichè non viene limitato il bitrate di generazione dei pacchetti (opzione *-b 0*) e il sender li genera ad una velocità maggiore di 10 Mb/s una parte dei pacchetti creata non può essere smaltita dalla pila protocollare del sender, in questo caso il 24%.

## -1.4 Risultati

## -1.5 Test UDP - Singolo flusso, full duplex, reverse mode

```
laboratorio@laboratorio:~$ iperf3 -c 192.168.1.196 -u -b 0 -R
Connecting to host 192.168.1.196, port 5201
Reverse mode, remote host 192.168.1.196 is sending
[  5] local 192.168.1.210 port 36376 connected to 192.168.1.196 port 5201
[ ID] Interval           Transfer     Bitrate         Jitter    Lost/Total Datagrams
[  5]   0.00-1.00   sec  1.13 MBytes  9.50 Mbits/sec  1.641 ms  63566/64379 (99%)
[  5]   1.00-2.00   sec  1.13 MBytes  9.50 Mbits/sec  1.602 ms  78989/79802 (99%)
[  5]   2.00-3.00   sec  1.13 MBytes  9.52 Mbits/sec  1.566 ms  74661/75476 (99%)
[  5]   3.00-4.00   sec  1.14 MBytes  9.55 Mbits/sec  1.636 ms  73455/74273 (99%)
[  5]   4.00-5.00   sec  1.13 MBytes  9.52 Mbits/sec  1.604 ms  77375/78190 (99%)
[  5]   5.00-6.00   sec  1.14 MBytes  9.55 Mbits/sec  1.609 ms  78063/78881 (99%)
[  5]   6.00-7.00   sec  1.14 MBytes  9.55 Mbits/sec  1.582 ms  76264/77082 (99%)
[  5]   7.00-8.00   sec  1.14 MBytes  9.55 Mbits/sec  1.590 ms  76040/76858 (99%)
[  5]   8.00-9.00   sec  1.14 MBytes  9.55 Mbits/sec  1.658 ms  77777/78595 (99%)
[  5]   9.00-10.00  sec  1.14 MBytes  9.54 Mbits/sec  1.560 ms  74951/75768 (99%)
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Jitter    Lost/Total Datagrams
[  5]   0.00-10.00  sec  1.04 GBytes   892 Mbits/sec  0.000 ms  0/759304 (0%)  sender
[  5]   0.00-10.00  sec  11.4 MBytes  9.53 Mbits/sec  1.560 ms  751141/759304 (99%)  receiver

iperf Done.
```

Figura 4: UDP PC -R

Essendo 1000 Mb/s la velocità tra il server e lo switch ci aspettiamo di avere una situazione di *bottleneck* poichè il client riceve solo a 10 Mb/s. Come si puo' notare dalla Figura 4 infatti abbiamo il 99% di pacchetti persi a causa della congestione della rete dovuta al riempimento della coda presente nello switch.