

Assignment 1 (Gruppo 14)

A cura di: Favara Nicola (578003) e Fulginiti Innocenzo (594051)

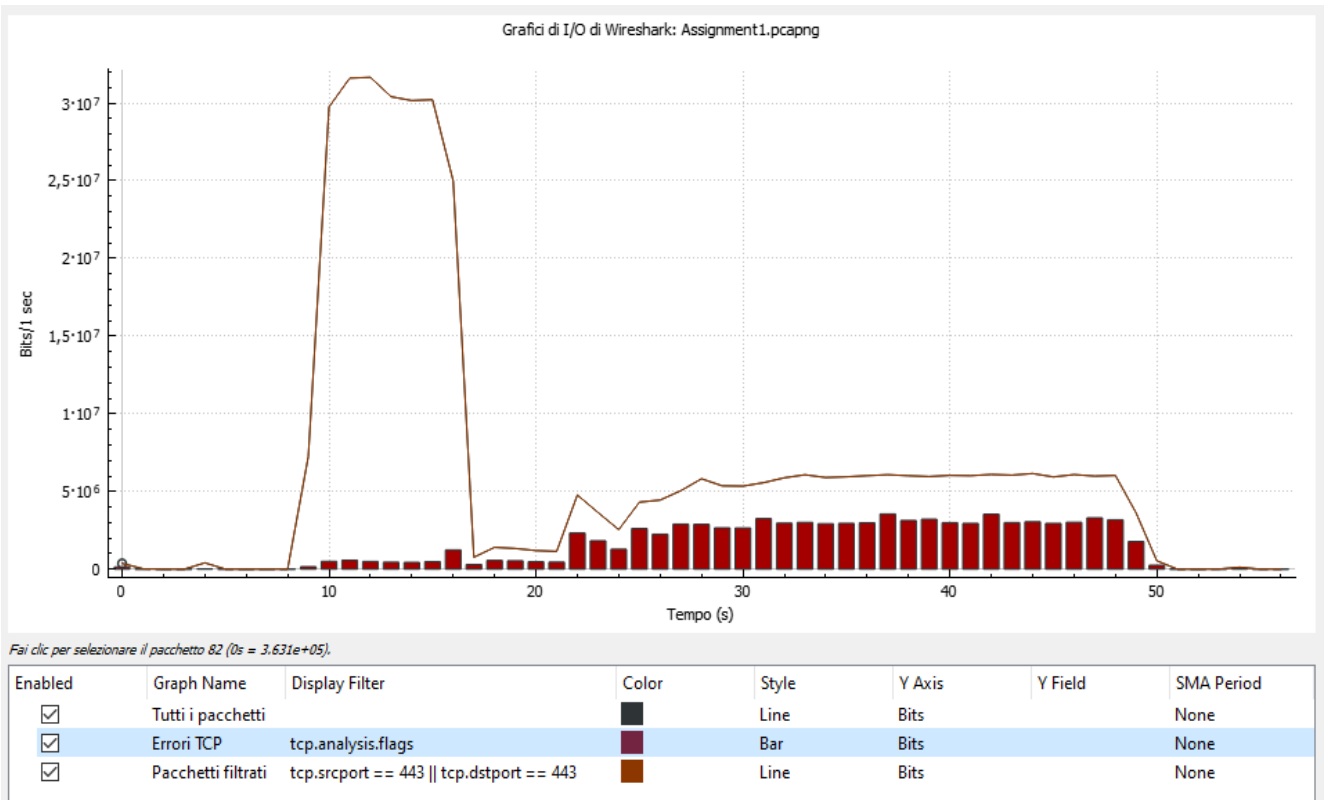
L'assignment chiedeva di analizzare il traffico catturato da uno speedtest con *Wireshark* e di verificare se i dati catturati siano simili a quelli di *fast.com*. Innanzitutto per evitare di catturare i pacchetti UDP (dato che il browser non manda pacchetti con protocollo UDP a livello di trasporto per lo *speedtest*) abbiamo avviato la cattura con il filtro: `!udp`.

Dopo aver atteso per qualche secondo, abbiamo avviato lo speed test di "*fast.com*". Una volta terminato abbiamo atteso ulteriormente dei secondi prima di interrompere la cattura.

Il risultato ottenuto è mostrato di fianco al paragrafo.



Andando sulla sezione "*Grafici I/O*", abbiamo visualizzato il seguente risultato:



Per osservare meglio i pacchetti dello speedtest, abbiamo usato il filtro:

```
tcp.srcport == 443 || tcp.dstport == 443
```

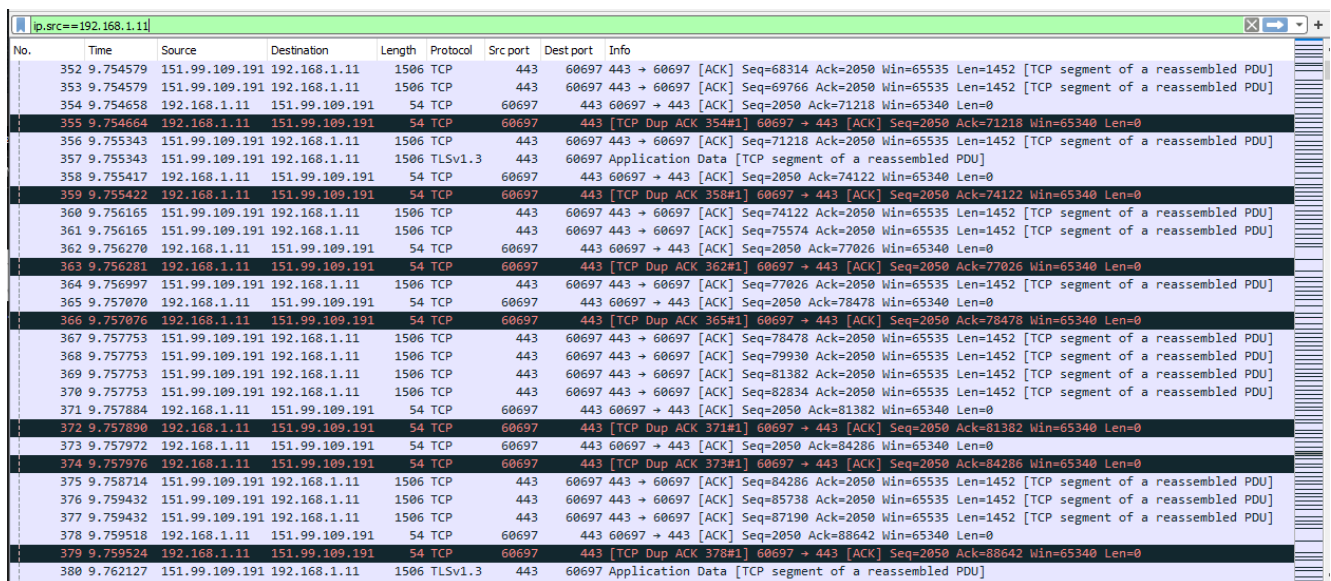
in modo tale da considerare solo i pacchetti TCP con protocollo applicativo HTTPS.

Dal grafo ottenuto abbiamo notato che, dal momento in cui è stato avviato lo *speedtest*, il numero di bps è aumentato rapidamente fino ad attestarsi ad un valore di circa $3 \cdot 10^7$ bps (~ 30 Mbps), per poi crollare immediatamente.

Dopo di che abbiamo osservato una piccola crescita che ha portato la funzione a stabilizzarsi di poco più di $5 \cdot 10^6$ bps (~ 5 Mbps) per poi tornare a valori minimi.

Confrontando i dati del grafico con quelli dello speed test abbiamo associato la parte della curva in cui viene raggiunto il valore massimo con la fase di download e la seconda parte con quella di upload.

Un'informazione che risalta subito è la presenza di diversi pacchetti contenenti errori che, come si può vedere dal grafico, aumentano di numero durante la fase di upload. Dal file pcap catturato possiamo notare che la maggior parte dei pacchetti di questo tipo sono: TCP *duplicate ACK*, TCP *retransmission* e TCP *window full*. I segmenti di riscontro duplicato e retransmission sembrano suggerire che i server, ai quali il nostro host è collegato durante lo *speedtest*, non rispondono mai. Infatti, se visualizziamo solo i pacchetti in uscita dal nostro computer, dopo ogni invio ne abbiamo uno con errore: inizialmente tutti di tipo TCP *duplicate ACK* durante la fase di download (infatti qui tutti i pacchetti sono di piccole dimensioni), poi quasi tutti di TCP *retransmission* durante la fase di upload (nella quale invece la dimensione dei pacchetti aumenta di molto).



The image shows a Wireshark packet capture window with the filter 'ip.src==192.168.1.1'. The table below represents the visible packets, which are all TCP segments originating from the local host (192.168.1.1) to a destination (192.168.1.11). The packets show various TCP flags and sequence numbers, indicating a connection in progress. Notably, several packets are marked as 'Duplicate ACK' or 'Retransmission', which are common in network tests like speedtest.

No.	Time	Source	Destination	Length	Protocol	Src port	Dest port	Info
352	9.754579	151.99.109.191	192.168.1.11	1506	TCP	443	60697	443 → 60697 [ACK] Seq=68314 Ack=2050 Win=65535 Len=1452 [TCP segment of a reassembled PDU]
353	9.754579	151.99.109.191	192.168.1.11	1506	TCP	443	60697	443 → 60697 [ACK] Seq=69766 Ack=2050 Win=65535 Len=1452 [TCP segment of a reassembled PDU]
354	9.754658	192.168.1.11	151.99.109.191	54	TCP	60697	443	60697 → 443 [ACK] Seq=2050 Ack=71218 Win=65340 Len=0
355	9.754664	192.168.1.11	151.99.109.191	54	TCP	60697	443	[TCP Dup ACK 354#1] 60697 → 443 [ACK] Seq=2050 Ack=71218 Win=65340 Len=0
356	9.755343	151.99.109.191	192.168.1.11	1506	TCP	443	60697	443 → 60697 [ACK] Seq=71218 Ack=2050 Win=65535 Len=1452 [TCP segment of a reassembled PDU]
357	9.755343	151.99.109.191	192.168.1.11	1506	TLSv1.3	443	60697	Application Data [TCP segment of a reassembled PDU]
358	9.755417	192.168.1.11	151.99.109.191	54	TCP	60697	443	60697 → 443 [ACK] Seq=2050 Ack=74122 Win=65340 Len=0
359	9.755422	192.168.1.11	151.99.109.191	54	TCP	60697	443	[TCP Dup ACK 358#1] 60697 → 443 [ACK] Seq=2050 Ack=74122 Win=65340 Len=0
360	9.756165	151.99.109.191	192.168.1.11	1506	TCP	443	60697	443 → 60697 [ACK] Seq=74122 Ack=2050 Win=65535 Len=1452 [TCP segment of a reassembled PDU]
361	9.756165	151.99.109.191	192.168.1.11	1506	TCP	443	60697	443 → 60697 [ACK] Seq=75574 Ack=2050 Win=65535 Len=1452 [TCP segment of a reassembled PDU]
362	9.756270	192.168.1.11	151.99.109.191	54	TCP	60697	443	60697 → 443 [ACK] Seq=2050 Ack=77026 Win=65340 Len=0
363	9.756281	192.168.1.11	151.99.109.191	54	TCP	60697	443	[TCP Dup ACK 362#1] 60697 → 443 [ACK] Seq=2050 Ack=77026 Win=65340 Len=0
364	9.756997	151.99.109.191	192.168.1.11	1506	TCP	443	60697	443 → 60697 [ACK] Seq=77026 Ack=2050 Win=65535 Len=1452 [TCP segment of a reassembled PDU]
365	9.757070	192.168.1.11	151.99.109.191	54	TCP	60697	443	60697 → 443 [ACK] Seq=2050 Ack=78478 Win=65340 Len=0
366	9.757076	192.168.1.11	151.99.109.191	54	TCP	60697	443	[TCP Dup ACK 365#1] 60697 → 443 [ACK] Seq=2050 Ack=78478 Win=65340 Len=0
367	9.757753	151.99.109.191	192.168.1.11	1506	TCP	443	60697	443 → 60697 [ACK] Seq=78478 Ack=2050 Win=65535 Len=1452 [TCP segment of a reassembled PDU]
368	9.757753	151.99.109.191	192.168.1.11	1506	TCP	443	60697	443 → 60697 [ACK] Seq=79930 Ack=2050 Win=65535 Len=1452 [TCP segment of a reassembled PDU]
369	9.757753	151.99.109.191	192.168.1.11	1506	TCP	443	60697	443 → 60697 [ACK] Seq=81382 Ack=2050 Win=65535 Len=1452 [TCP segment of a reassembled PDU]
370	9.757753	151.99.109.191	192.168.1.11	1506	TCP	443	60697	443 → 60697 [ACK] Seq=82834 Ack=2050 Win=65535 Len=1452 [TCP segment of a reassembled PDU]
371	9.757884	192.168.1.11	151.99.109.191	54	TCP	60697	443	60697 → 443 [ACK] Seq=2050 Ack=81382 Win=65340 Len=0
372	9.757890	192.168.1.11	151.99.109.191	54	TCP	60697	443	[TCP Dup ACK 371#1] 60697 → 443 [ACK] Seq=2050 Ack=81382 Win=65340 Len=0
373	9.757972	192.168.1.11	151.99.109.191	54	TCP	60697	443	60697 → 443 [ACK] Seq=2050 Ack=84286 Win=65340 Len=0
374	9.757976	192.168.1.11	151.99.109.191	54	TCP	60697	443	[TCP Dup ACK 373#1] 60697 → 443 [ACK] Seq=2050 Ack=84286 Win=65340 Len=0
375	9.758714	151.99.109.191	192.168.1.11	1506	TCP	443	60697	443 → 60697 [ACK] Seq=84286 Ack=2050 Win=65535 Len=1452 [TCP segment of a reassembled PDU]
376	9.759432	151.99.109.191	192.168.1.11	1506	TCP	443	60697	443 → 60697 [ACK] Seq=85738 Ack=2050 Win=65535 Len=1452 [TCP segment of a reassembled PDU]
377	9.759432	151.99.109.191	192.168.1.11	1506	TCP	443	60697	443 → 60697 [ACK] Seq=87190 Ack=2050 Win=65535 Len=1452 [TCP segment of a reassembled PDU]
378	9.759518	192.168.1.11	151.99.109.191	54	TCP	60697	443	60697 → 443 [ACK] Seq=2050 Ack=88642 Win=65340 Len=0
379	9.759524	192.168.1.11	151.99.109.191	54	TCP	60697	443	[TCP Dup ACK 378#1] 60697 → 443 [ACK] Seq=2050 Ack=88642 Win=65340 Len=0
380	9.762127	151.99.109.191	192.168.1.11	1506	TLSv1.3	443	60697	Application Data [TCP segment of a reassembled PDU]

Sempre dal confronto possiamo notare delle differenze tra i valori che ci viene fornito da "fast.com" (28 Mbps in download e 2 Mbps in upload) con quelli osservati su Wireshark.

Per quanto il download, ipotizziamo che questa differenza sia dovuta al fatto che fast non considera gli header dei vari livelli aggiunti ai pacchetti e quindi fornisce una stima un po' approssimata.

Per l'upload invece notiamo una differenza più netta (circa il doppio di quanto asserito da fast.com). Questo, secondo noi, è causato dai tanti pacchetti persi in questa fase, infatti, se si sottraggono i bps persi, da quelli totali otteniamo all'incirca lo stesso valore di quello individuato da fast.

Abbiamo anche notato che le connessioni che vengono stabilite durante lo speedtest sono principalmente con 5 host, come mostrato nell'immagine sottostante.

Address	Packets	Bytes	Tx Packets	Tx Bytes	Rx Packets	Rx Bytes
151.99.109.191	14.904	16M	7.402	9267k	7.502	7093k
45.57.73.140	9.867	11M	4.587	5186k	5.280	6317k
45.57.72.138	8.559	10M	4.023	4577k	4.536	5460k
23.246.50.159	5.059	4285k	2.757	4060k	2.302	225k
23.246.51.143	4.165	3474k	2.225	3215k	1.940	259k

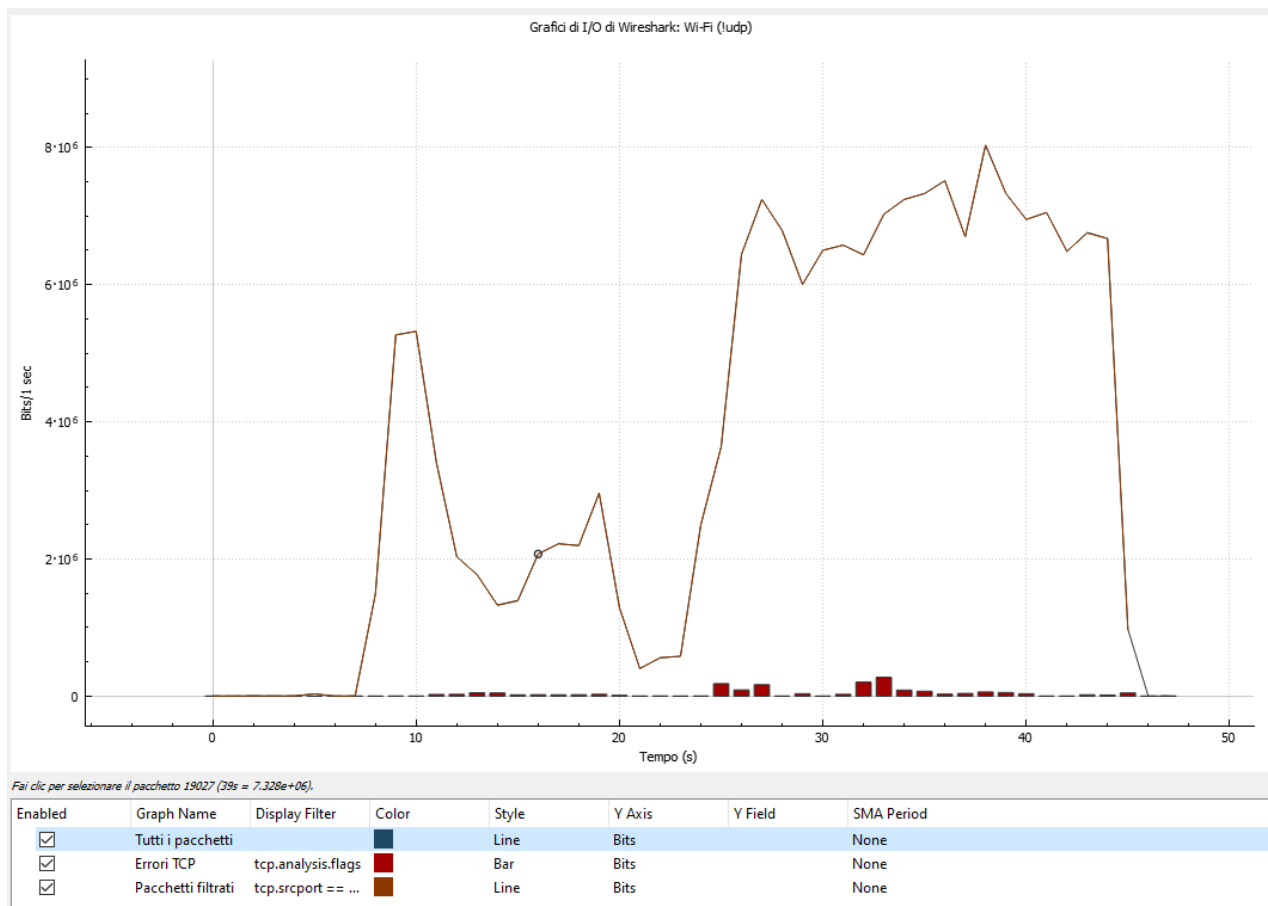
Essi sono quindi i 5 host con i quali il nostro pc crea delle connessioni durante l'intera operazione e che cercano di inviare più pacchetti possibili per riempire la banda e permettere al browser di vedere quanto al massimo essa può supportare.

Per un'ulteriore conferma del coinvolgimento di questi host sullo speedtest abbiamo notato che questi indirizzi risultavano presenti anche nella sezione *Network* degli strumenti sviluppatore di *Chrome*.

Contenuti aggiuntivi dopo il commento del professore

Abbiamo provato a ripetere il test utilizzando un'altra rete, questa volta i pacchetti con errore erano in numero nettamente inferiore e i dati di Wireshark e di Fast coincidevano maggiormente, soprattutto in upload che era la fase in cui si erano verificate le differenze principali durante il test precedente.





Questo ci fa ipotizzare che quei pacchetti di errore riscontrati non sono legati allo speed test ma più alla rete che si sta usando al momento e che minore è il numero di questo tipo di pacchetti, più i dati delle due parti si avvicinano.