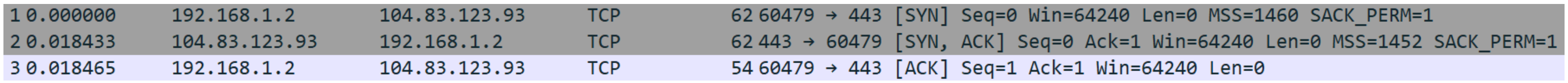
Erda Ymeri 632415 Nicola Coltelli 581087 Tommaso Lencioni 560309

Gruppo 9 – Assignment 1 – fast.com speed test

The capture filter used in Wireshark to filter only the packets coming from the fast.com website is **host fast.com** . This is contained inside capture fast_capture.pcap.

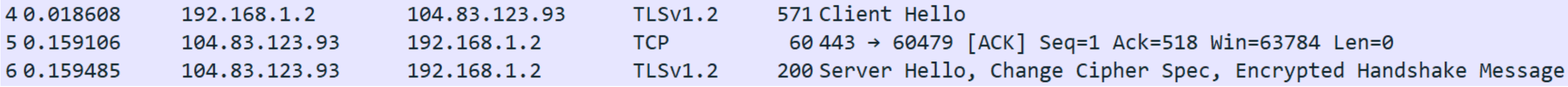
All the packets sent to us have the MAC address of our home router.

The first packets received from the Wireshark capture filter are packets from the three-way handshake TCP used in the Transport layer.



```
1 0.000000     192.168.1.2      104.83.123.93    TCP     62 60479 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1
2 0.018433     104.83.123.93    192.168.1.2      TCP     62 443 → 60479 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1452 SACK_PERM=1
3 0.018465     192.168.1.2      104.83.123.93    TCP     54 60479 → 443 [ACK] Seq=1 Ack=1 Win=64240 Len=0
```

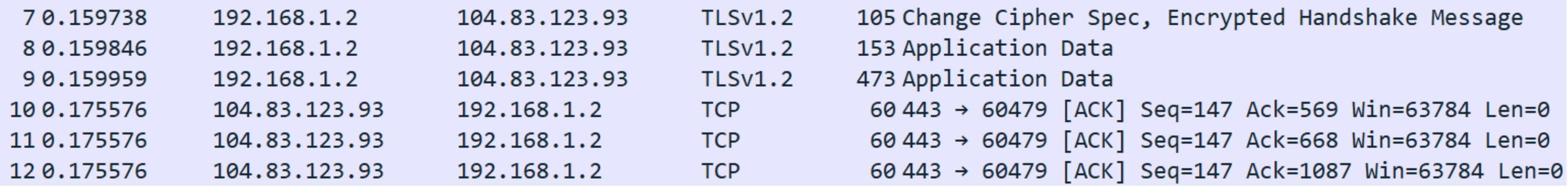
The 4th and the 6th packets are the, respectively, the Client Hello and the Server Hello (they are TLS packets encapsulated in TCP segments).



```
4 0.018608     192.168.1.2      104.83.123.93    TLSv1.2    571 Client Hello
5 0.159106     104.83.123.93    192.168.1.2      TCP         60 443 → 60479 [ACK] Seq=1 Ack=518 Win=63784 Len=0
6 0.159485     104.83.123.93    192.168.1.2      TLSv1.2    200 Server Hello, Change Cipher Spec, Encrypted Handshake Message
```

The 6th packet also contains a flag that communicates that we already established a secure connection with the server. Having this flag set makes us spare some packets exchange for establishing the secure connection from nothing.

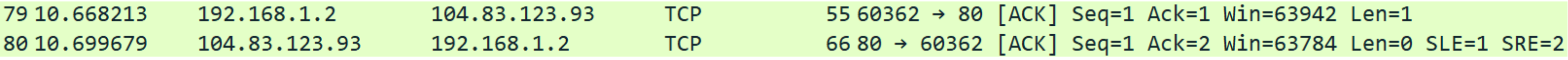
Between packet 7 and 9 we send data to the server to finish the TLS setup from the client side, and between packet 10 and 12 the server responds with empty ACKs; finally, in packets 13 and 14 there is a last setup exchange between us and the server, and so we finish setting the TLS connection up. (maybe insert the screens even for 13 and 14)



```
 7 0.159738    192.168.1.2      104.83.123.93    TLSv1.2    105 Change Cipher Spec, Encrypted Handshake Message
 8 0.159846    192.168.1.2      104.83.123.93    TLSv1.2    153 Application Data
 9 0.159959    192.168.1.2      104.83.123.93    TLSv1.2    473 Application Data
10 0.175576    104.83.123.93    192.168.1.2      TCP         60 443 → 60479 [ACK] Seq=147 Ack=569 Win=63784 Len=0
11 0.175576    104.83.123.93    192.168.1.2      TCP         60 443 → 60479 [ACK] Seq=147 Ack=668 Win=63784 Len=0
12 0.175576    104.83.123.93    192.168.1.2      TCP         60 443 → 60479 [ACK] Seq=147 Ack=1087 Win=63784 Len=0
```

We suppose that the 15th packet with 1506 length is the first download packet to test our download speed. The packet has been split into two packets (15 and 17) as suggested by Wireshark, and we can see the reassembled result by examining packet 17. After that the server keeps sending data and we reply to those packets with empty ACKs (es. Packets 16 and 18).

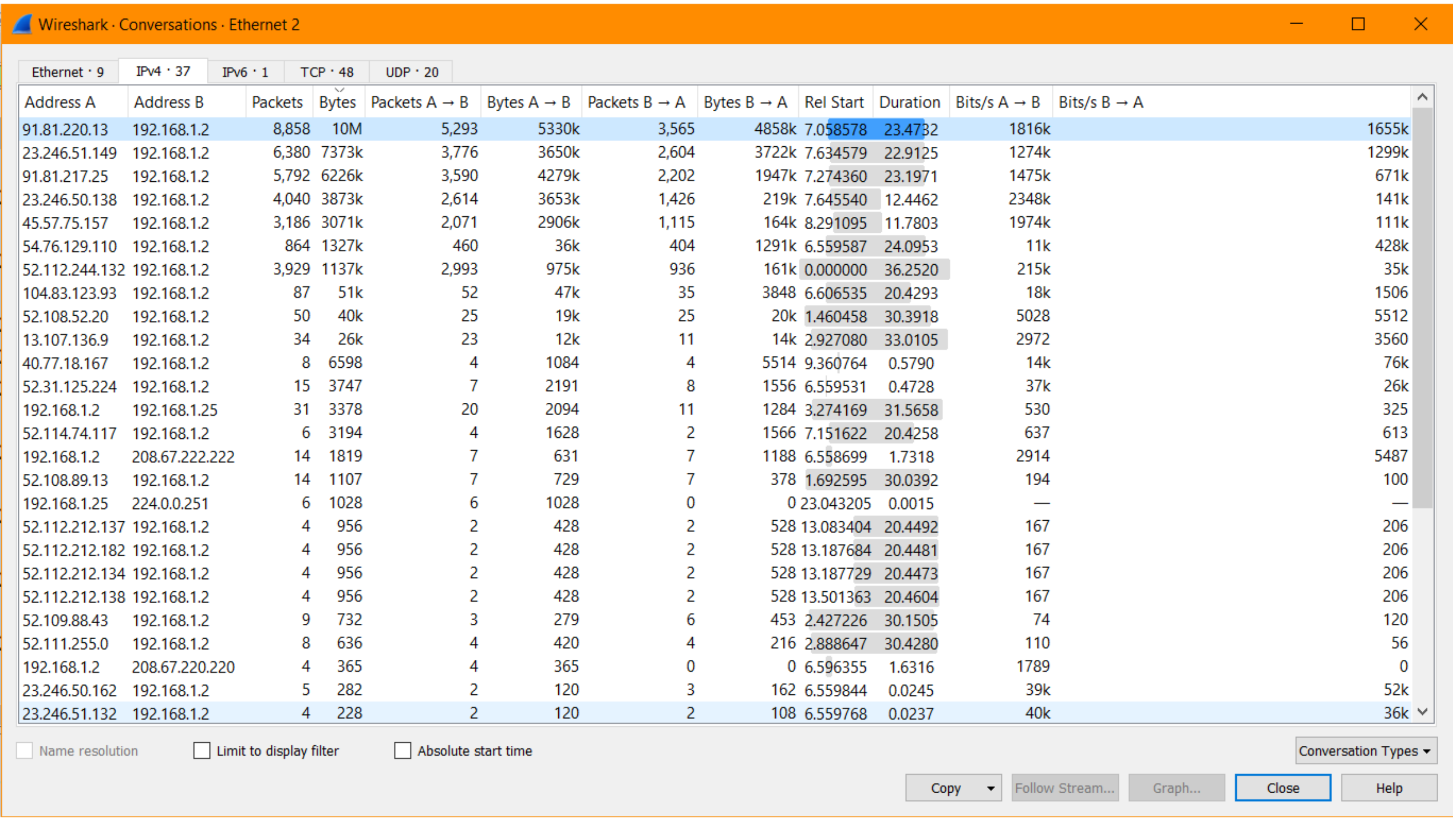
We can see that a TCP segment is sent as keep alive on HTTP protocol (port 80 instead of 443 used by HTTP over TLS) because we accidentally left the capture going beyond the TCP timeout (approximately 10 seconds past the last ACK segment).



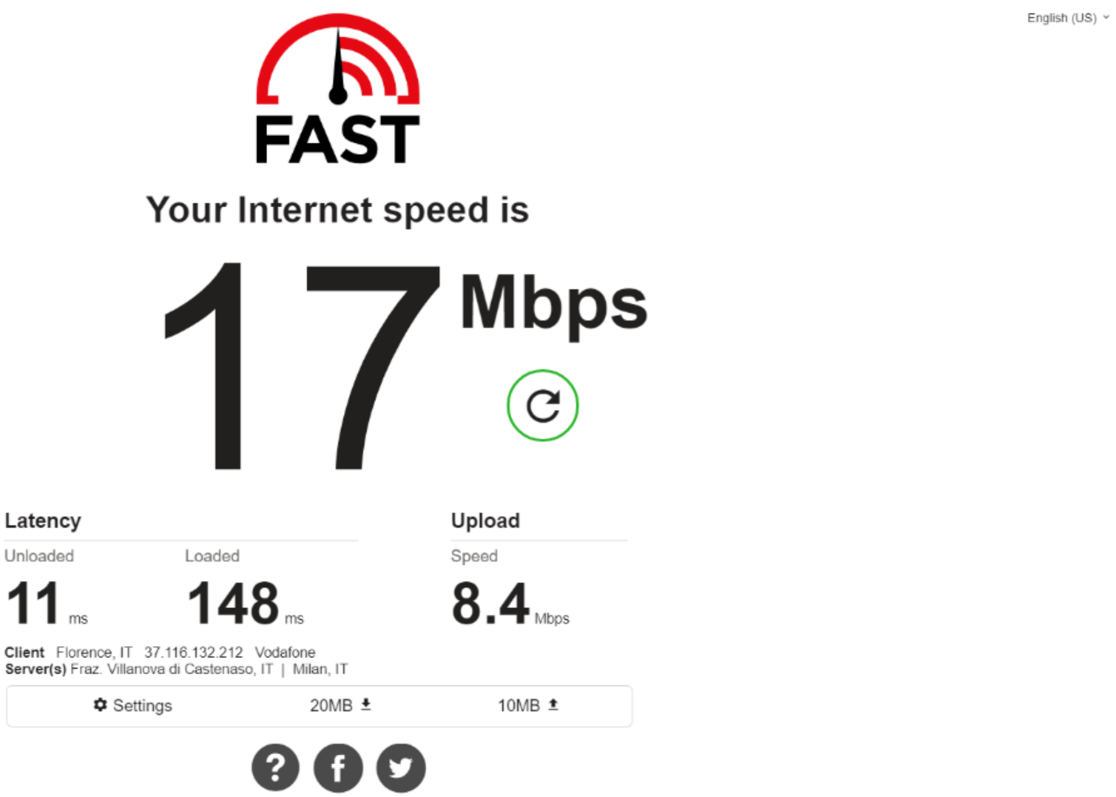
```
79 10.668213   192.168.1.2      104.83.123.93    TCP     55 60362 → 80 [ACK] Seq=1 Ack=1 Win=63942 Len=1
80 10.699679   104.83.123.93    192.168.1.2      TCP     66 80 → 60362 [ACK] Seq=1 Ack=2 Win=63784 Len=0 SLE=1 SRE=2
```

The data packets that we see in the capture are not the only ones downloaded from the server because the server starts some parallel connections to transfer more data at the same time. We can't see those connections in the capture because the capture filter was set on **host www.fast.com,** thus the parallel connections' IP addresses are filtered out since their hostname is not ‘fast.com’. We could not find a filter that was able to capture only the packets from fast.com and the parallel connections’ packets since we don’t know their IPs in advance. We sorted the conversations by dimension because we guessed they were the ones coming from fast.com, and then we made sure by doing some DNS queries and we checked if they were Netflix IPs.

We noticed that by summing up the dimension of the packets coming from fast.com and the parallel connection we obtained around 30MB, which corresponds to the amount of data fast.com says it downloaded and uploaded. We could keep track of the number of bytes in the Conversation window (20MB down + 10MB up).



Address A	Address B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
91.81.220.13	192.168.1.2	8,858	10M	5,293	5330k	3,565	4858k	7.058578	23.4732	1816k	1655k
23.246.51.149	192.168.1.2	6,380	7373k	3,776	3650k	2,604	3722k	7.634579	22.9125	1274k	1299k
91.81.217.25	192.168.1.2	5,792	6226k	3,590	4279k	2,202	1947k	7.274360	23.1971	1475k	671k
23.246.50.138	192.168.1.2	4,040	3873k	2,614	3653k	1,426	219k	7.645540	12.4462	2348k	141k
45.57.75.157	192.168.1.2	3,186	3071k	2,071	2906k	1,115	164k	8.291095	11.7803	1974k	111k
54.76.129.110	192.168.1.2	864	1327k	460	36k	404	1291k	6.559587	24.0953	11k	428k
52.112.244.132	192.168.1.2	3,929	1137k	2,993	975k	936	161k	0.000000	36.2520	215k	35k
104.83.123.93	192.168.1.2	87	51k	52	47k	35	3848	6.606535	20.4293	18k	1506
52.108.52.20	192.168.1.2	50	40k	25	19k	25	20k	1.460458	30.3918	5028	5512
13.107.136.9	192.168.1.2	34	26k	23	12k	11	14k	2.927080	33.0105	2972	3560
40.77.18.167	192.168.1.2	8	6598	4	1084	4	5514	9.360764	0.5790	14k	76k
52.31.125.224	192.168.1.2	15	3747	7	2191	8	1556	6.559531	0.4728	37k	26k
192.168.1.2	192.168.1.25	31	3378	20	2094	11	1284	3.274169	31.5658	530	325
52.114.74.117	192.168.1.2	6	3194	4	1628	2	1566	7.151622	20.4258	637	613
192.168.1.2	208.67.222.222	14	1819	7	631	7	1188	6.558699	1.7318	2914	5487
52.108.89.13	192.168.1.2	14	1107	7	729	7	378	1.692595	30.0392	194	100
192.168.1.25	224.0.0.251	6	1028	6	1028	0	0	23.043205	0.0015	—	—
52.112.212.137	192.168.1.2	4	956	2	428	2	528	13.083404	20.4492	167	206
52.112.212.182	192.168.1.2	4	956	2	428	2	528	13.187684	20.4481	167	206
52.112.212.134	192.168.1.2	4	956	2	428	2	528	13.187729	20.4473	167	206
52.112.212.138	192.168.1.2	4	956	2	428	2	528	13.501363	20.4604	167	206
52.109.88.43	192.168.1.2	9	732	3	279	6	453	2.427226	30.1505	74	120
52.111.255.0	192.168.1.2	8	636	4	420	4	216	2.888647	30.4280	110	56
192.168.1.2	208.67.220.220	4	365	4	365	0	0	6.596355	1.6316	1789	0
23.246.50.162	192.168.1.2	5	282	2	120	3	162	6.559844	0.0245	39k	52k
23.246.51.132	192.168.1.2	4	228	2	120	2	108	6.559768	0.0237	40k	36k



Your Internet speed is

17 Mbps

Latency

Unloaded 11 ms

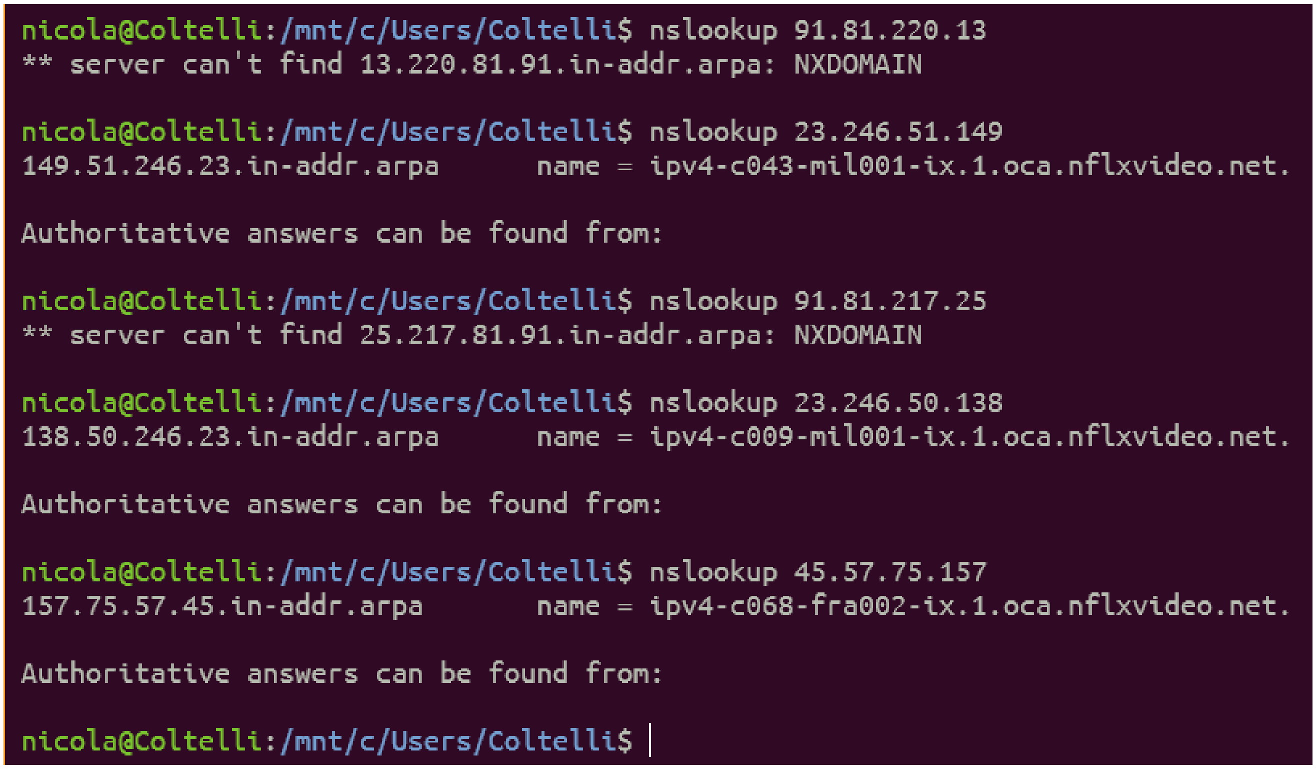
Loaded 148 ms

Upload

Speed 8.4 Mbps

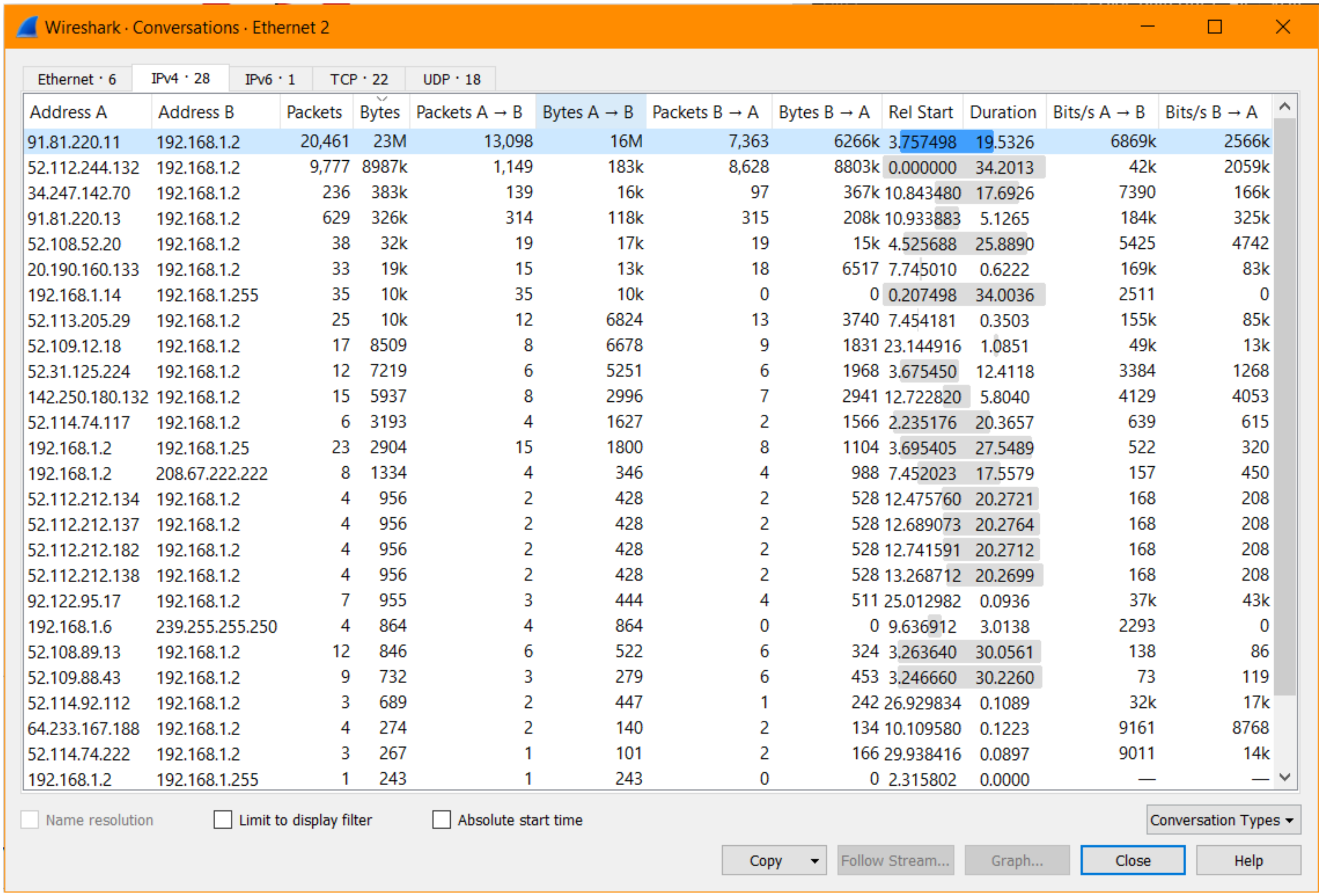
Client Florence, IT 37.116.132.212 Vodafone
Server(s) Fraz. Villanova di Castenaso, IT | Milan, IT

We made sure those were fast.com’s IPs by making a DNS interrogation using **nslookup**, and for some of them we obtained that they are Netflix hosts.



```
nicola@Coltelli:/mnt/c/Users/Coltelli$ nslookup 91.81.220.13
** server can't find 13.220.81.91.in-addr.arpa: NXDOMAIN

nicola@Coltelli:/mnt/c/Users/Coltelli$ nslookup 23.246.51.149
149.51.246.23.in-addr.arpa      name = ipv4-c043-mil001-ix.1.oca.nflxvideo.net.

Authoritative answers can be found from:

nicola@Coltelli:/mnt/c/Users/Coltelli$ nslookup 91.81.217.25
** server can't find 25.217.81.91.in-addr.arpa: NXDOMAIN

nicola@Coltelli:/mnt/c/Users/Coltelli$ nslookup 23.246.50.138
138.50.246.23.in-addr.arpa      name = ipv4-c009-mil001-ix.1.oca.nflxvideo.net.

Authoritative answers can be found from:

nicola@Coltelli:/mnt/c/Users/Coltelli$ nslookup 45.57.75.157
157.75.57.45.in-addr.arpa       name = ipv4-c068-fra002-ix.1.oca.nflxvideo.net.

Authoritative answers can be found from:

nicola@Coltelli:/mnt/c/Users/Coltelli$
```

We noticed that on fast.com under Show more info -> Settings we can specify the max Parallel connections. Setting it to 1 we connected once again and, this time all the data is transferred inside a single conversation (IP 91.81.220.11).



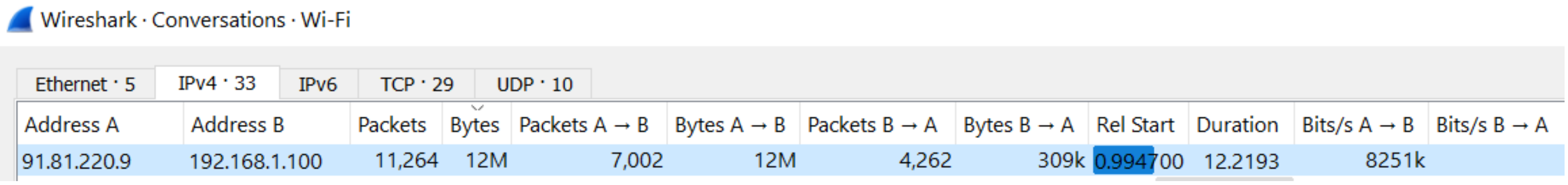
Address A	Address B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
91.81.220.11	192.168.1.2	20,461	23M	13,098	16M	7,363	6266k	3.757498	19.5326	6869k	2566k
52.112.244.132	192.168.1.2	9,777	8987k	1,149	183k	8,628	8803k	0.000000	34.2013	42k	2059k
34.247.142.70	192.168.1.2	236	383k	139	16k	97	367k	10.843480	17.6926	7390	166k
91.81.220.13	192.168.1.2	629	326k	314	118k	315	208k	10.933883	5.1265	184k	325k
52.108.52.20	192.168.1.2	38	32k	19	17k	19	15k	4.525688	25.8890	5425	4742
20.190.160.133	192.168.1.2	33	19k	15	13k	18	6517	7.745010	0.6222	169k	83k
192.168.1.14	192.168.1.255	35	10k	35	10k	0	0	0.207498	34.0036	2511	0
52.113.205.29	192.168.1.2	25	10k	12	6824	13	3740	7.454181	0.3503	155k	85k
52.109.12.18	192.168.1.2	17	8509	8	6678	9	1831	23.144916	1.0851	49k	13k
52.31.125.224	192.168.1.2	12	7219	6	5251	6	1968	3.675450	12.4118	3384	1268
142.250.180.132	192.168.1.2	15	5937	8	2996	7	2941	12.722820	5.8040	4129	4053
52.114.74.117	192.168.1.2	6	3193	4	1627	2	1566	2.235176	20.3657	639	615
192.168.1.2	192.168.1.25	23	2904	15	1800	8	1104	3.695405	27.5489	522	320
192.168.1.2	208.67.222.222	8	1334	4	346	4	988	7.452023	17.5579	157	450
52.112.212.134	192.168.1.2	4	956	2	428	2	528	12.475760	20.2721	168	208
52.112.212.137	192.168.1.2	4	956	2	428	2	528	12.689073	20.2764	168	208
52.112.212.182	192.168.1.2	4	956	2	428	2	528	12.741591	20.2712	168	208
52.112.212.138	192.168.1.2	4	956	2	428	2	528	13.268712	20.2699	168	208
92.122.95.17	192.168.1.2	7	955	3	444	4	511	25.012982	0.0936	37k	43k
192.168.1.6	239.255.255.250	4	864	4	864	0	0	9.636912	3.0138	2293	0
52.108.89.13	192.168.1.2	12	846	6	522	6	324	3.263640	30.0561	138	86
52.109.88.43	192.168.1.2	9	732	3	279	6	453	3.246660	30.2260	73	119
52.114.92.112	192.168.1.2	3	689	2	447	1	242	26.929834	0.1089	32k	17k
64.233.167.188	192.168.1.2	4	274	2	140	2	134	10.109580	0.1223	9161	8768
52.114.74.222	192.168.1.2	3	267	1	101	2	166	29.938416	0.0897	9011	14k
192.168.1.2	192.168.1.255	1	243	1	243	0	0	2.315802	0.0000	—	—

□ Name resolution □ Limit to display filter □ Absolute start time

Initially we observed that from our computer were uploaded small packets (varying 100-700 length, instead of the ~1500 of the download packets) to test the upload speed.
However, after discovering the use of parallel connections we saw that we sent the declared amount of data indeed. You can check their dimension with the display filter (**ip.src == 192.168.1.2 and tcp.len > 0**). We concluded that this is due to the difference in bandwidth between download and upload, with upload being notoriously smaller.

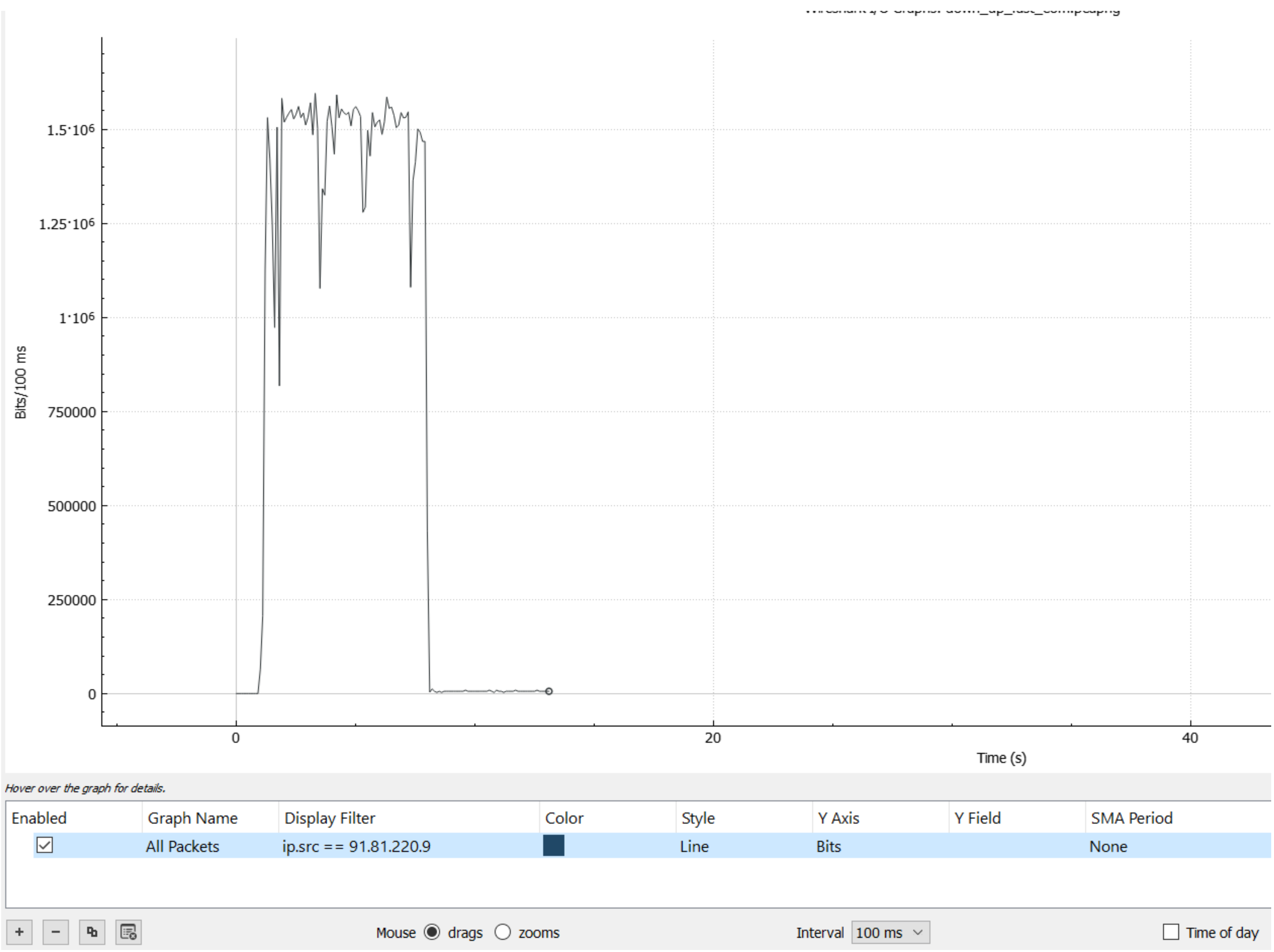
We can’t estimate the latency using Wireshark because it gives us the timestamps of arrival of the server packets and departure of our packets. We would need, instead, the timestamp of departure of the packets but that information is known only by fast.com. We suppose that fast.com estimates our latency by measuring the difference between the timestamp of departure of a packet and the timestamp of arrival of the corresponding ACK we sent back.

To understand how fast.com measures download, we started a run with just one parallel connection. We can see that the data received from the address 91.81.220.9 sums up to 12MB (96Mb). The download spikes last 8 seconds of measurement so if we want to establish an average value, we can divide it by 8, resulting in 12Mb/s, exactly the estimation done by fast.com. This is contained inside capture tcp_capture.pcap.



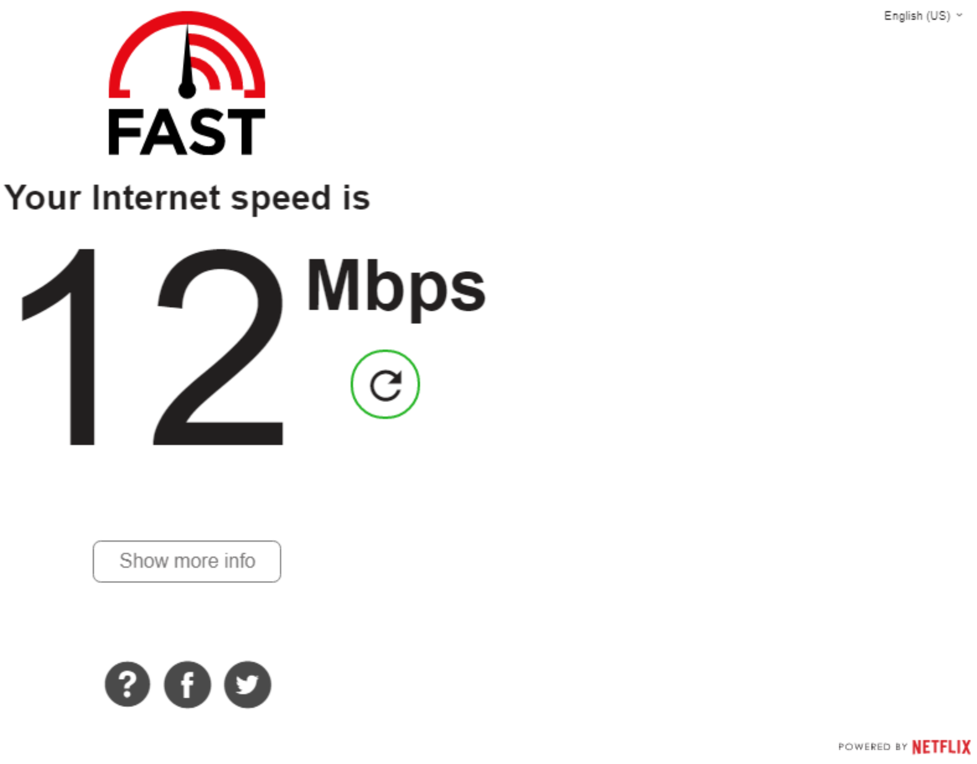
Wireshark · Conversations · Wi-Fi

Address A	Address B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
91.81.220.9	192.168.1.100	11,264	12M	7,002	12M	4,262	309k	0.994700	12.2193	8251k	



Hover over the graph for details.

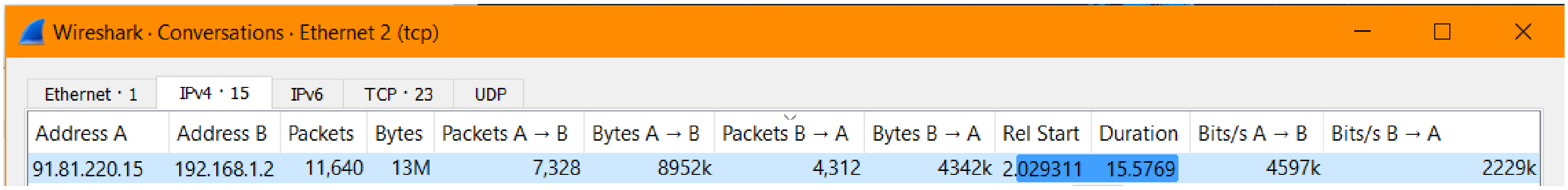
Enabled	Graph Name	Display Filter	Color	Style	Y Axis	Y Field	SMA Period
☑	All Packets	ip.src == 91.81.220.9	■	Line	Bits		None



Your Internet speed is

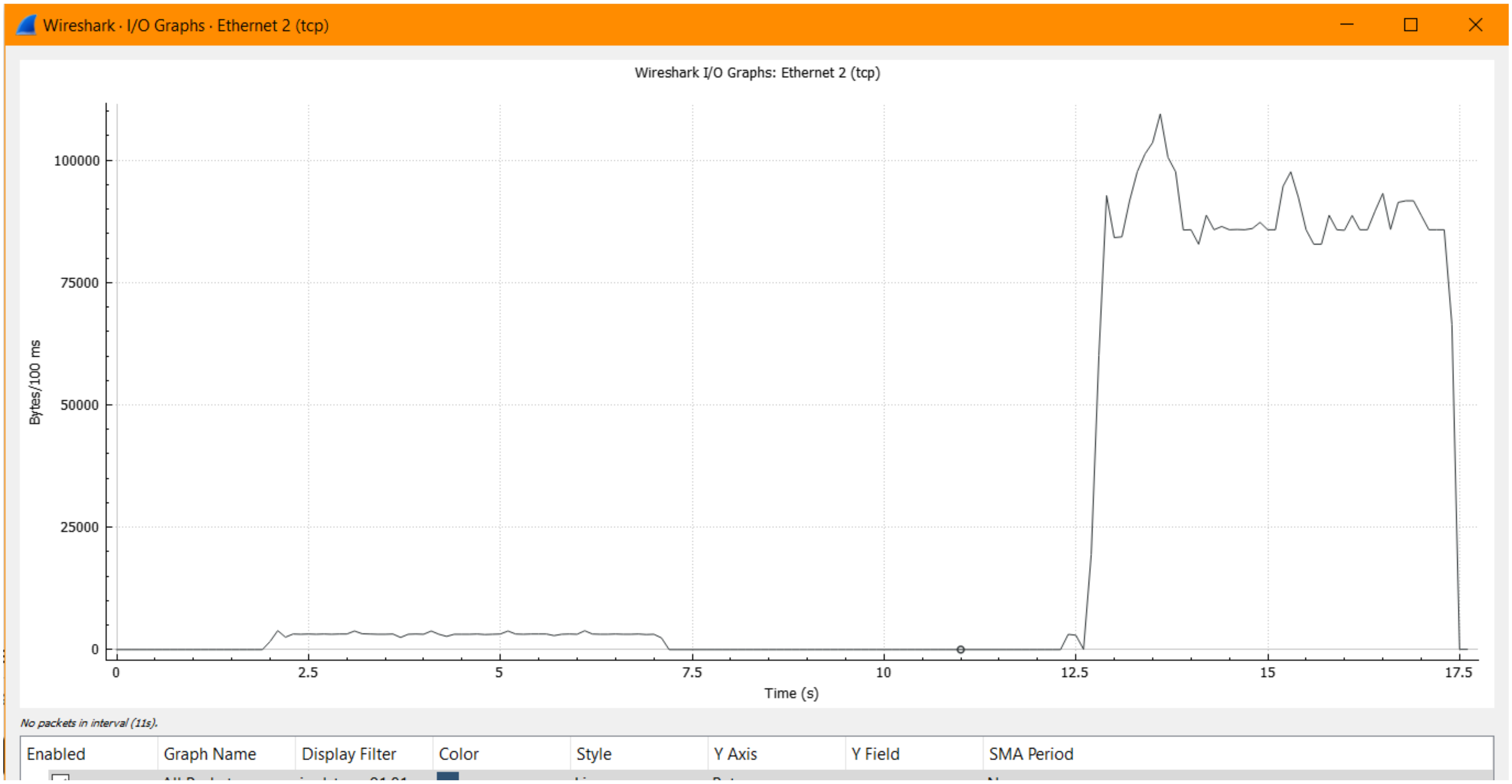
12 Mbps

We did the same for upload in a different capture, receiving 4342KB (4.24MB, 33.92Mb). The upload spike lasts for 4.9 seconds (just as we specified in fast.com settings) giving us an upload speed of 33.92/4.9 = 6.92Mb/s, which is very close to what fast.com reports (7.0Mb/s).



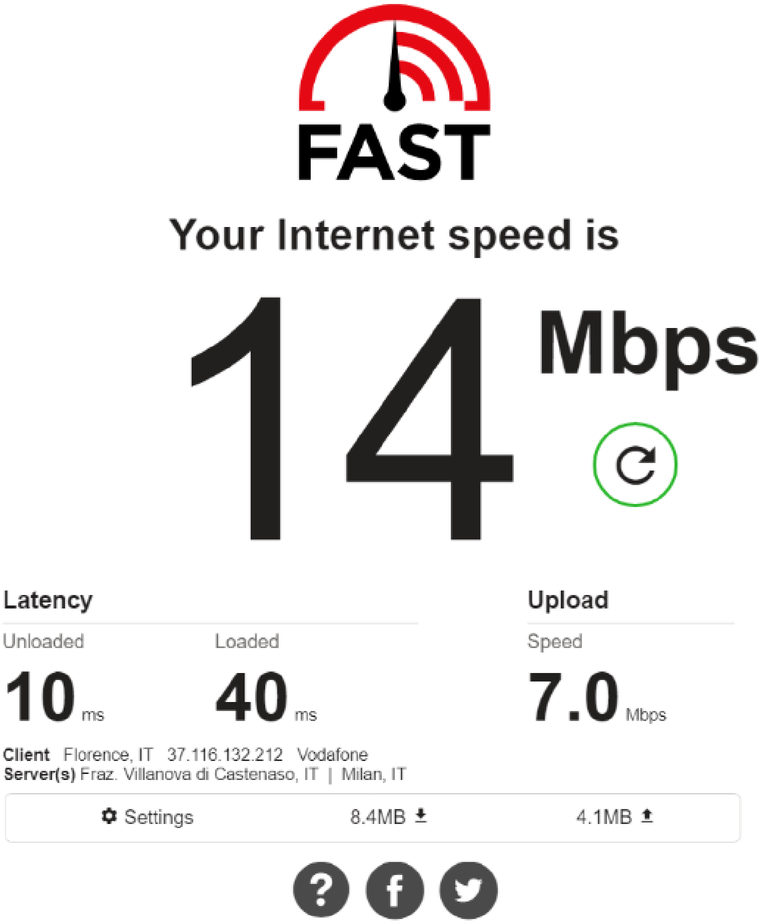
Address A	Address B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
91.81.220.15	192.168.1.2	11,640	13M	7,328	8952k	4,312	4342k	2.029311	15.5769	4597k	2229k

Erda Ymeri 632415 Nicola Coltelli 581087 Tommaso Lencioni 560309


Wireshark · I/O Graphs · Ethernet 2 (tcp)
Wireshark I/O Graphs: Ethernet 2 (tcp)
Bytes/100 ms
100000
75000
50000
25000
0
0
2.5
5
7.5
10
12.5
15
17.5
Time (s)

No packets in interval (11s).

Enabled	Graph Name	Display Filter	Color	Style	Y Axis	Y Field	SMA Period



Your Internet speed is

14 Mbps

Latency

Unloaded 10 ms

Loaded 40 ms

Upload

Speed 7.0 Mbps

Client Florence, IT 37.116.132.212 Vodafone
Server(s) Fraz. Villanova di Castenaso, IT | Milan, IT

Erda Ymeri 632415 Nicola Coltelli 581087 Tommaso Lencioni 560309

Pcaps files:

fast_capture.pcap => capture filter: **host fast.com** ; fast.com settings: **default**

tcp_capture.pcap => capture filter: **tcp** ; fast.com settings: **one parallel connection**

The screenshots also refer to the other 2 captures made with the same settings, so we decided not to include them.