计算机网络PJ

21302010042
侯斌洋

一、浏览器测量

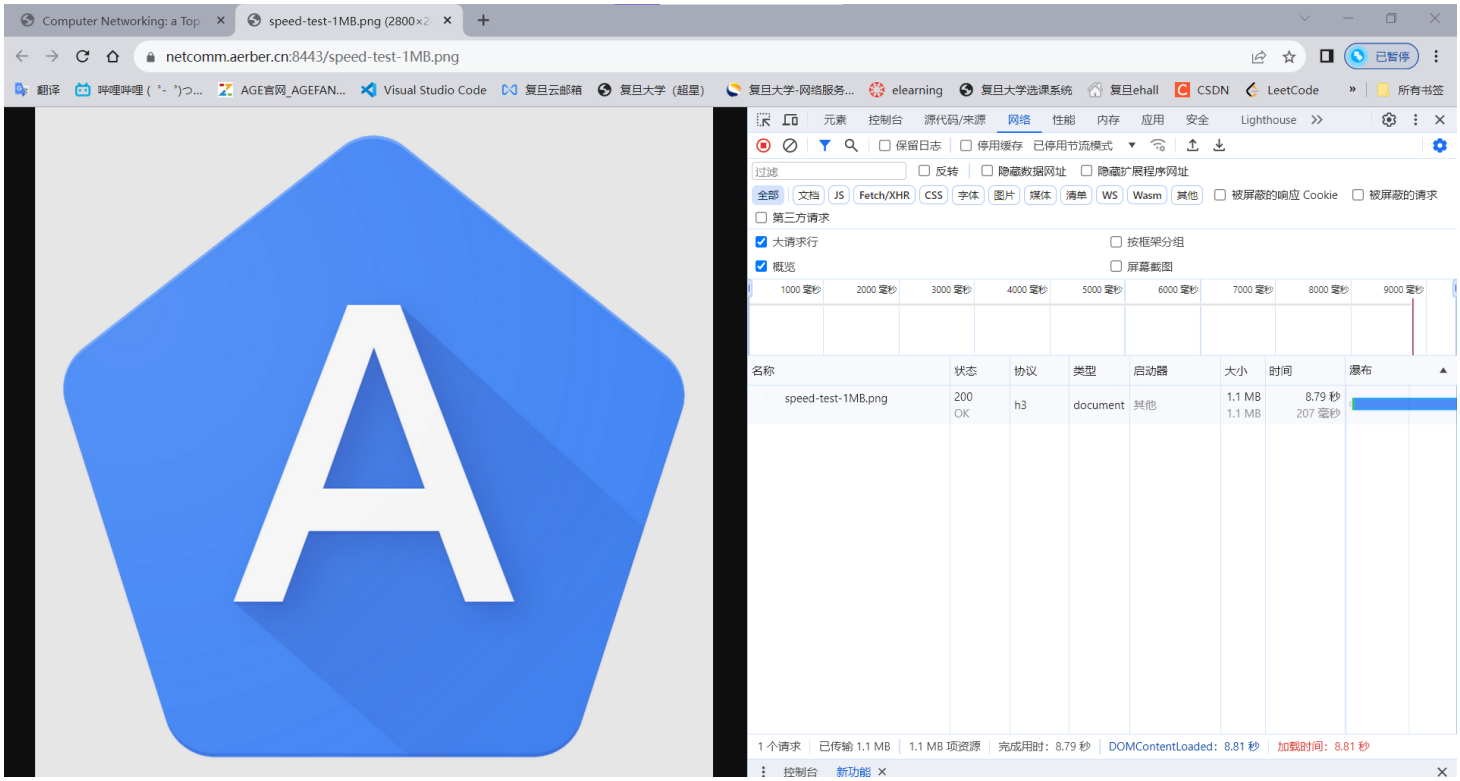
（1）环境：windows

（2）工具：Chrome 120.0.6099.71（正式版本） （64 位）

（3）测量对象：对于图片，测量资源加载时间，对于html页面，测量页面加载时间。加载时间均以chrome控制台网络页下显示的时间为准。

（4）测量过程截图：

测量过程较长，在此只列出一个例子。如下所示，统计的时间即为右下角红字的加载时间。测量过程中显示协议来确保正确性。因为h3在最开始需要h1.1的alt-src响应头，故在测量像照片这样的单个资源而不是html页面时，需要先访问同端口的html文件让浏览器知道后续可以使用h3。在用chrome浏览器测量h3时这点尤为重要，特此说明。



（5）测量结果

为减小误差，测量结果为测5次取平均值，在Excel中记录，结果在 result.xlsx 的chrome工作表中。

（注：在此处及下面提到的各种文件都在src目录中）

HTTP1.1

HTTP1.1	结果1/s	结果2/s	结果3/s	结果4/s	结果5/s	平均结果/s
https://111.229.132.28:80/speed-test-100KB.png	1.03	2.09	2.46	2.67	2.87	2.224
https://111.229.132.28:80/speed-test-1MB.png	30.83	29.32	22.58	34.19	27.31	28.846
https://111.229.132.28:80/speed-test-5MB.jpg	114.00	120.00	114.00	138.00	108.00	118.800
https://111.229.132.28:80/speed-test-small.html	5.59	6.72	7.55	4.70	9.96	6.904
https://111.229.132.28:80/speed-test-medium.html	7.99	9.42	8.63	11.75	10.51	9.660
https://111.229.132.28:80/speed-test-large.html	168.00	204.00	150.00	132.00	228.00	176.400

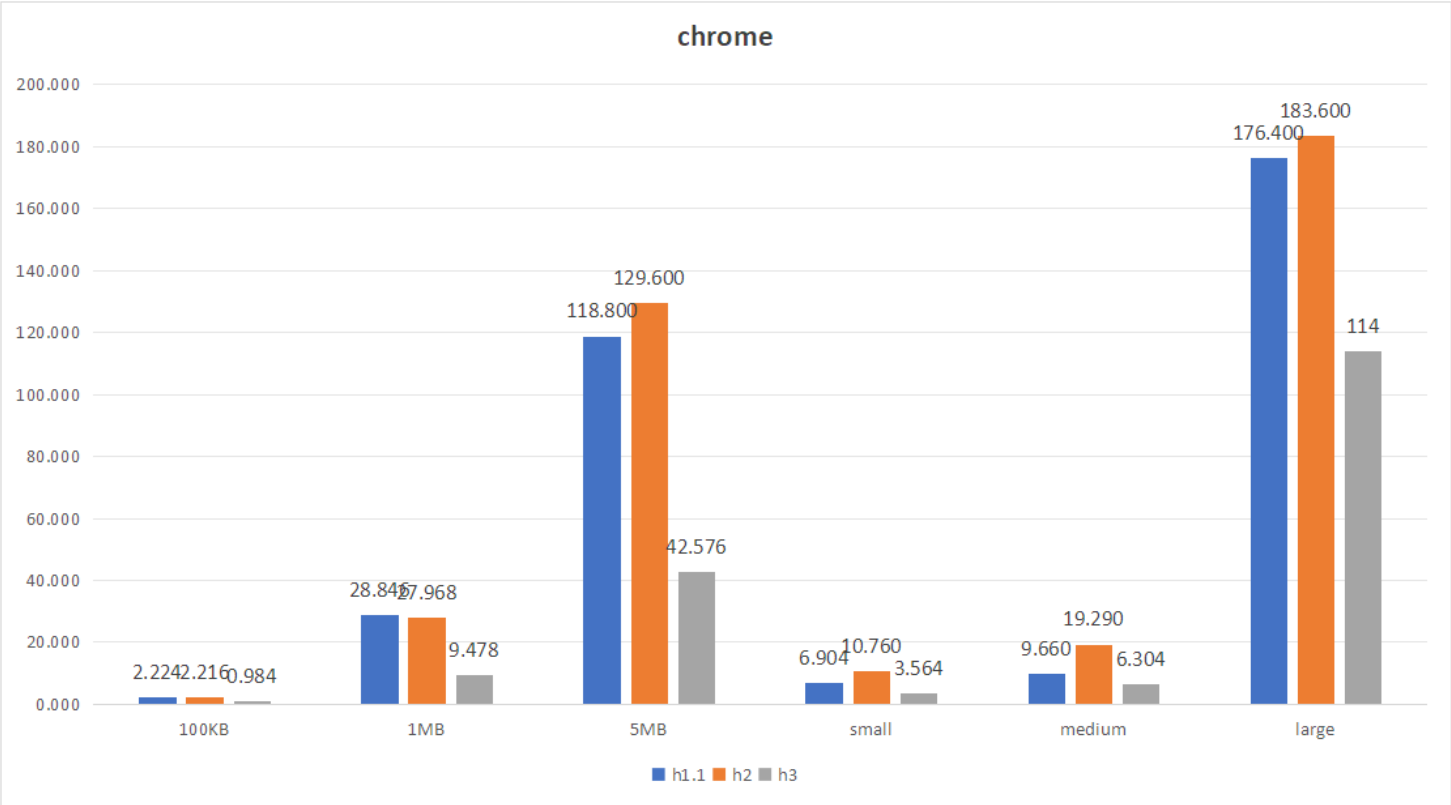
HTTP2

HTTP2.0	结果1/s	结果2/s	结果3/s	结果4/s	结果5/s	平均结果/s
https://111.229.132.28:443/speed-test-100KB.png	2.57	1.44	1.86	1.85	3.36	2.216
https://111.229.132.28:443/speed-test-1MB.png	20.77	39.57	31.28	23.36	24.86	27.968
https://111.229.132.28:443/speed-test-5MB.jpg	132.00	138.00	114.00	132.00	132.00	129.600
https://111.229.132.28:443/speed-test-small.html	10.78	12.15	11.77	8.74	10.36	10.760
https://111.229.132.28:443/speed-test-medium.html	21.60	17.64	19.14	18.34	19.73	19.290
https://111.229.132.28:443/speed-test-large.html	204.00	186.00	198	162.00	168.00	183.600

HTTP3

HTTP3.0	结果1/s	结果2/s	结果3/s	结果4/s	结果5/s	平均结果/s
https://netcomm.aerber.cn:8443/speed-test-100KB.png	1.28	0.86	0.62	1.31	0.85	0.984
https://netcomm.aerber.cn:8443/speed-test-1MB.png	9.17	9.13	9.40	10.68	9.01	9.478
https://netcomm.aerber.cn:8443/speed-test-5MB.jpg	42.67	42.02	42.82	42.23	43.14	42.576
https://netcomm.aerber.cn:8443/speed-test-small.html	3.59	3.42	3.79	3.39	3.63	3.564
https://netcomm.aerber.cn:8443/speed-test-medium.html	6.09	6.13	6.13	6.17	7.00	6.304
https://netcomm.aerber.cn:8443/speed-test-large.html	114	114	114	114	114	114

（6）分析



以上为平均时间对比图。可以看出，无论是请求单个文件还是html页面，h1.1和h2的性能十分相近，且h2会略慢一点。h3的性能则明显要高很多，在请求大文件和大页面时尤为显著，速度大致为h1.1和h2的两倍。由此看出，在从h1.1到h2的更迭中性能并没有提升，而从h2到h3的过程中性能显著提升。浏览器采用QUIC的确能在很大程度上提升网页访问速度。

二、命令行发送请求

（1）环境：ubuntu22.04

（2）工具：

h1.1: curl

h2: curl

h3: ymuski/curl-http3 (in docker)

（3）测量对象：从输入curl命令开始到命令执行结束的时间，即为完整获取文件的时间。注意到对于html文件，不会加载该页面中的其他资源，仅加载该html文件。以及请求各种端口时的响应头。

（4）测量过程截图：

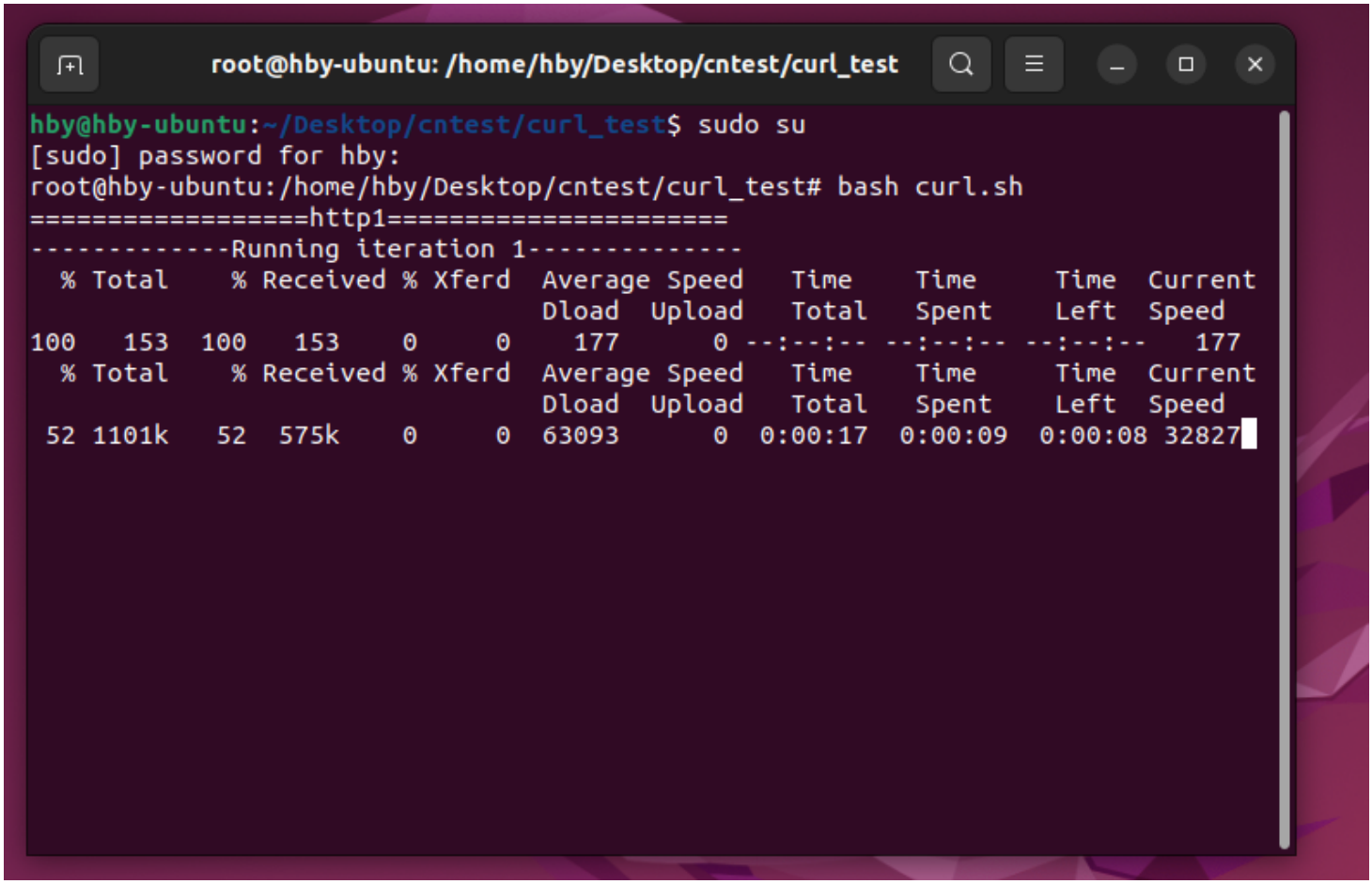
由于涉及的命令较多，测量时间相对教长，故写了个bash脚本来进行运行（curl.sh），如下所示：

```
#!/bin/bash
num_iterations=5
sync
echo ==================http1====================== >result.txt
echo ==================http1======================

for ((i=1; i<=$num_iterations; i++)); do
    echo "-------------Running iteration $i--------------" >> result.txt
    echo "-------------Running iteration $i--------------"
    # 1 start
    start_time=$(date +%s.%N)
    curl -k --http1.1 https://111.229.132.28:80/speed-test-100KB.jpg --output test.png
    wait
    end_time=$(date +%s.%N)
    execution_time=$(echo "$end_time - $start_time" | bc)
    echo "http1.1---100KB: $execution_time seconds" >> result.txt
    wait
    # end

    # 2 start
    start_time=$(date +%s.%N)
    curl -k --http1.1 https://111.229.132.28:80/speed-test-1MB.png --output test.png
    wait
    end_time=$(date +%s.%N)
    execution_time=$(echo "$end_time - $start_time" | bc)
    echo "http1.1---1MB: $execution_time seconds" >> result.txt
    wait
    # end

    # 3 start
    start_time=$(date +%s.%N);
    curl -k --http1.1 https://111.229.132.28:80/speed-test-5MB.jpg --output test.jpg
    wait
    end_time=$(date +%s.%N)
    execution_time=$(echo "$end_time - $start_time" | bc)
    echo "http1.1---5MB: $execution_time seconds" >> result.txt
    wait
    # end

    # 4 start
    start_time=$(date +%s.%N)
    curl -k --http1.1 https://111.229.132.28:80/speed-test-small.html --output test.html
    wait
```

```
    end_time=$(date +%s.%N)
    execution_time=$(echo "$end_time - $start_time" | bc)
    echo "http1.1---small: $execution_time seconds" >> result.txt
    wait
    # end

    # 5 start
    start_time=$(date +%s.%N)
    curl -k --http1.1 https://111.229.132.28:80/speed-test-medium.html --output test.html
    wait
    end_time=$(date +%s.%N)
    execution_time=$(echo "$end_time - $start_time" | bc)
    echo "http1.1---medium: $execution_time seconds" >> result.txt
    wait
    # end

    # 6 start
    start_time=$(date +%s.%N)
    curl -k --http1.1 https://111.229.132.28:80/speed-test-large.html --output test.html
    wait
    end_time=$(date +%s.%N)
    execution_time=$(echo "$end_time - $start_time" | bc)
    echo "http1.1---large: $execution_time seconds" >> result.txt
    wait
    # end

    echo "------------------------------------------------" >> result.txt
    echo "------------------------------------------------"
    wait
done

echo ================================================ >>result.txt
echo ================================================
wait

# head
echo ====================http1==================== 1>head.txt;
echo ====================http1====================
curl -k --http1.1 https://111.229.132.28:80/speed-test-100KB.png --head 1>>head.txt;
wait
curl -k --http1.1 https://111.229.132.28:80/speed-test-1MB.png --head 1>>head.txt;
wait
curl -k --http1.1 https://111.229.132.28:80/speed-test-5MB.jpg --head 1>>head.txt;
```

```
wait
curl -k --http1.1 https://111.229.132.28:80/speed-test-small.html --head 1>>head.txt;
wait
curl -k --http1.1 https://111.229.132.28:80/speed-test-medium.html --head 1>>head.txt;
wait
curl -k --http1.1 https://111.229.132.28:80/speed-test-large.html --head 1>>head.txt;
wait
echo ========================================== 1>>head.txt;
echo ==========================================
```

以上为对于 http1.1的测试时间和响应头的命令。其他对于h2和h3的测量命令与此类似，可详见文件。



（5）测量结果

为减小误差，测量结果为测5次取平均值，在Excel中记录，结果在result.xlsx的curl工作表中（curl.sh运行结果在m_result.txt中）。

HTTP1.1

HTTP1.1	结果1/s	结果2/s	结果3/s	结果4/s	结果5/s	平均结果/s
curl -k --http1.1 https://111.229.132.28:80/speed-test-100KB.png --output test.png	0.91	0.88	0.87	0.89	0.88	0.886
curl -k --http1.1 https://111.229.132.28:80/speed-test-1MB.png --output test.png	26.78	23.05	34.32	24.23	23.63	26.402
curl -k --http1.1 https://111.229.132.28:80/speed-test-5MB.jpg --output test.jpg	135.96	127.19	97.19	124.44	121.68	121.292
curl -k --http1.1 https://111.229.132.28:80/speed-test-small.html --output test.html	0.89	0.88	0.87	0.87	0.88	0.878
curl -k --http1.1 https://111.229.132.28:80/speed-test-medium.html --output test.html	3.00	1.28	2.11	3.06	2.04	2.298
curl -k --http1.1 https://111.229.132.28:80/speed-test-large.html --output test.html	5.22	6.42	2.15	2.54	6.43	4.552

HTTP2

HTTP2.0	结果1/s	结果2/s	结果3/s	结果4/s	结果5/s	平均结果/s
curl -k --http2 https://111.229.132.28:443/speed-test-100KB.png --output test.png	3.54	3.27	1.67	6.41	1.46	3.270
curl -k --http2 https://111.229.132.28:443/speed-test-1MB.png --output test.png	28.78	18.38	33.59	23.59	32.07	27.282
curl -k --http2 https://111.229.132.28:443/speed-test-5MB.jpg --output test.jpg	129.37	123.99	105.23	120.20	130.40	121.838
curl -k --http2 https://111.229.132.28:443/speed-test-small.html --output test.html	0.83	0.83	0.83	0.83	0.84	0.832
curl -k --http2 https://111.229.132.28:443/speed-test-medium.html --output test.html	2.34	1.29	2.78	2.48	2.32	2.242
curl -k --http2 https://111.229.132.28:443/speed-test-large.html --output test.html	2.07	9.85	2.26	10.60	2.91	5.538

HTTP3

HTTP3.0	结果1/s	结果2/s	结果3/s	结果4/s	结果5/s	平均结果/s
docker run -it --rm ymuski/curl-http3 curl https://netcomm.aerber.cn:8443/speed-test-100KB.png --http3-only --output test.png	1.58	1.34	1.87	1.92	1.83	1.708
docker run -it --rm ymuski/curl-http3 curl https://netcomm.aerber.cn:8443/speed-test-1MB.png --http3-only --output test.png	11.34	10.91	10.07	10.18	10.94	10.688
docker run -it --rm ymuski/curl-http3 curl https://netcomm.aerber.cn:8443/speed-test-5MB.jpg --http3-only --output test.jpg	44.77	44.45	44.45	43.39	43.80	44.172
docker run -it --rm ymuski/curl-http3 curl https://netcomm.aerber.cn:8443/speed-test-small.html --http3-only --output test.html	1.30	1.43	1.43	2.57	2.05	1.756
docker run -it --rm ymuski/curl-http3 curl https://netcomm.aerber.cn:8443/speed-test-medium.html --http3-only --output test.html	1.58	1.60	1.60	1.47	1.77	1.604
docker run -it --rm ymuski/curl-http3 curl https://netcomm.aerber.cn:8443/speed-test-large.html --http3-only --output test.html	2.70	3.99	3.99	2.52	2.58	3.156

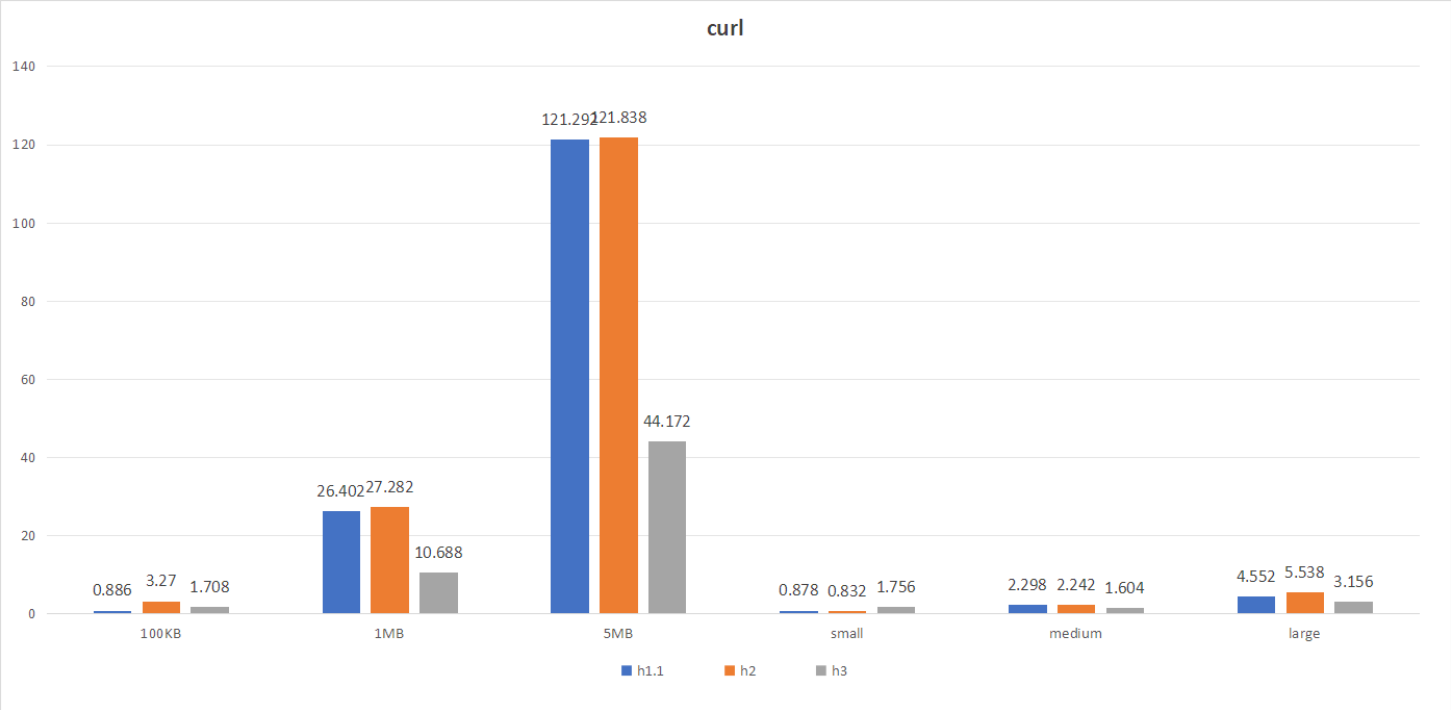
head

以h1.1的head为例，h2和h3的可见 m_head.txt。

```
==================http1==================
HTTP/1.1 200 OK
Server: nginx/1.25.2
Date: Thu, 07 Dec 2023 12:15:46 GMT
Content-Type: image/png
Content-Length: 98763
Last-Modified: Mon, 23 Oct 2023 07:53:25 GMT
Connection: keep-alive
ETag: "65362675-181cb"
Accept-Ranges: bytes

HTTP/1.1 200 OK
Server: nginx/1.25.2
Date: Thu, 07 Dec 2023 12:15:48 GMT
Content-Type: image/png
Content-Length: 1128358
Last-Modified: Mon, 23 Oct 2023 07:53:22 GMT
Connection: keep-alive
ETag: "65362672-1137a6"
Accept-Ranges: bytes

HTTP/1.1 200 OK
Server: nginx/1.25.2
Date: Thu, 07 Dec 2023 12:15:49 GMT
Content-Type: image/jpeg
Content-Length: 5048759
Last-Modified: Mon, 23 Oct 2023 07:53:25 GMT
Connection: keep-alive
ETag: "65362675-4d09b7"
Accept-Ranges: bytes

HTTP/1.1 200 OK
Server: nginx/1.25.2
Date: Thu, 07 Dec 2023 12:15:50 GMT
Content-Type: text/html
Content-Length: 5885
Last-Modified: Mon, 23 Oct 2023 08:02:38 GMT
Connection: keep-alive
ETag: "6536289e-16fd"
Accept-Ranges: bytes

HTTP/1.1 200 OK
Server: nginx/1.25.2
```

```
Date: Thu, 07 Dec 2023 12:15:51 GMT
Content-Type: text/html
Content-Length: 92550
Last-Modified: Mon, 23 Oct 2023 07:55:47 GMT
Connection: keep-alive
ETag: "65362703-16986"
Accept-Ranges: bytes

HTTP/1.1 200 OK
Server: nginx/1.25.2
Date: Thu, 07 Dec 2023 12:15:53 GMT
Content-Type: text/html
Content-Length: 229404
Last-Modified: Mon, 23 Oct 2023 08:12:02 GMT
Connection: keep-alive
ETag: "65362ad2-3801c"
Accept-Ranges: bytes

==========================================
```

(6) 分析



以上为平均时间对比图。由于html文件自身比较小，故使用命令行请求时差距不大。在对于大文件的请求上，则与使用浏览器得到的结论类似。h3速度大致为h1.1和h2的两倍。故在命令行上请求文件时采用QUIC也能在很大程度上提升网页访问速度。

三、wireshark抓包

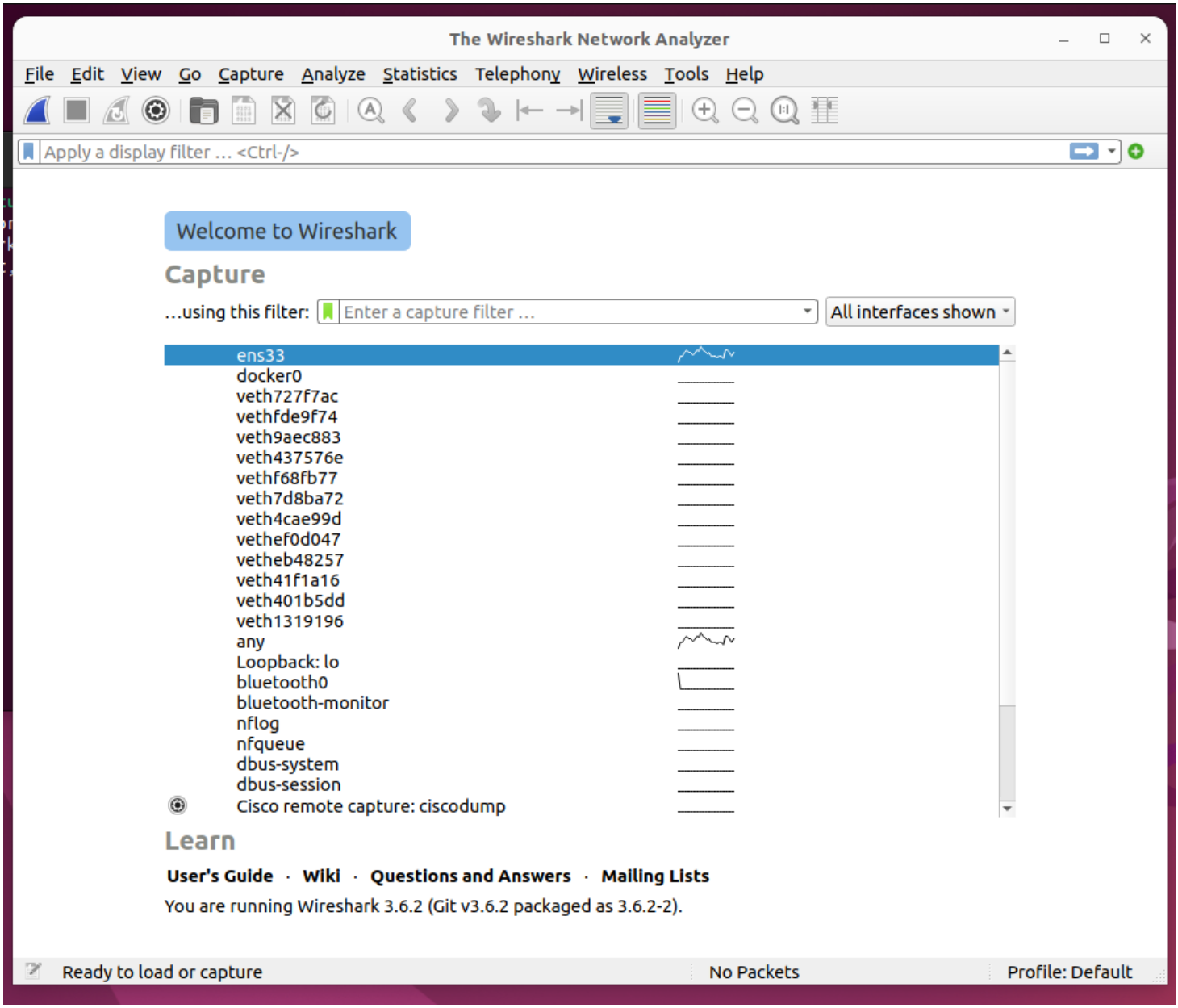
（1）环境：ubuntu22.04

（2）工具：wireshark

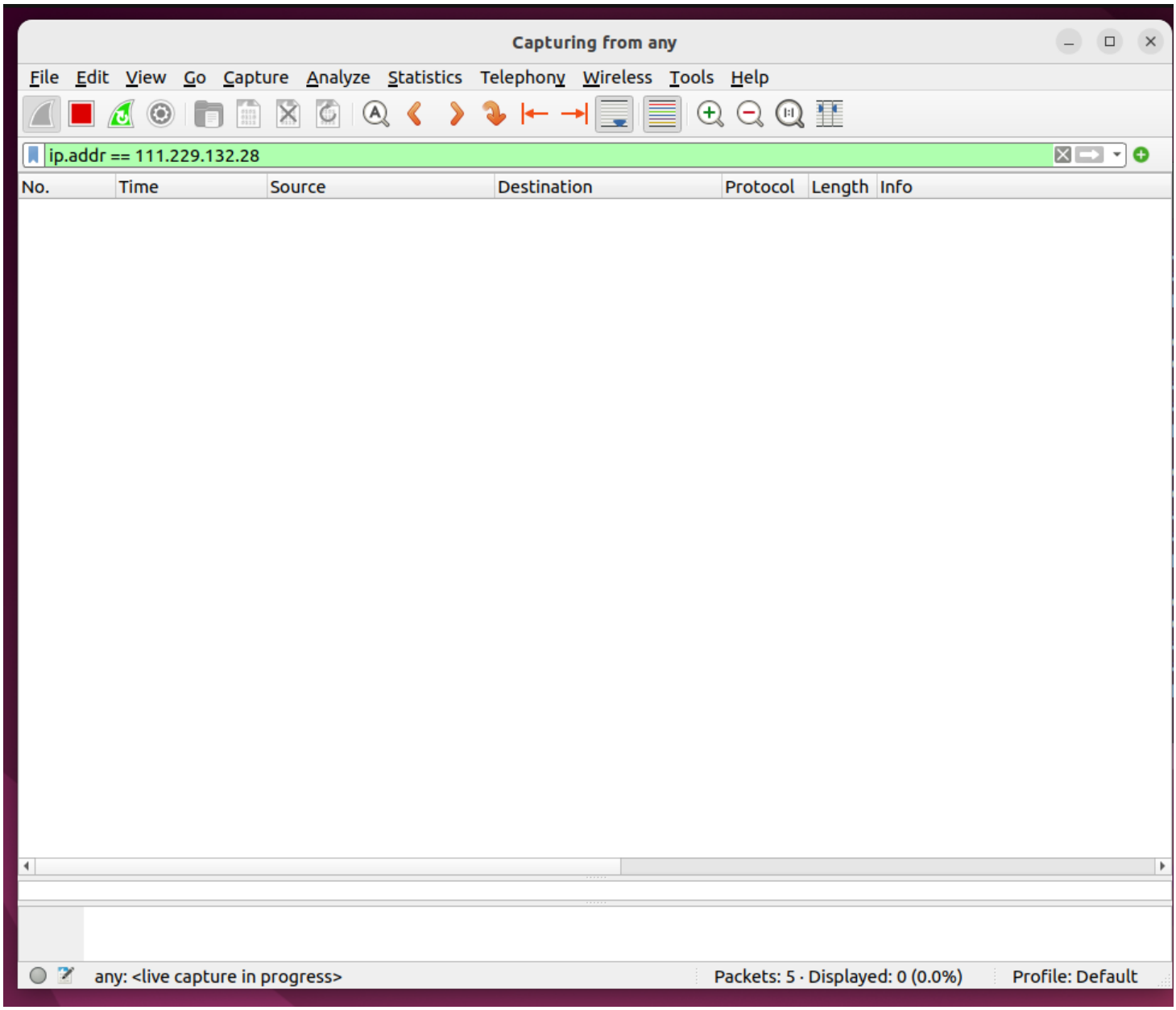
（3）测量对象：由于主要目的为抓取HTTP请求过程中的网络数据包，识别在此过程中涉及到的网络协议，并对相关协议进行分析。故对于每个协议均以请求100KB.png文件的head为示例来进行分析。

（4）测量过程截图：

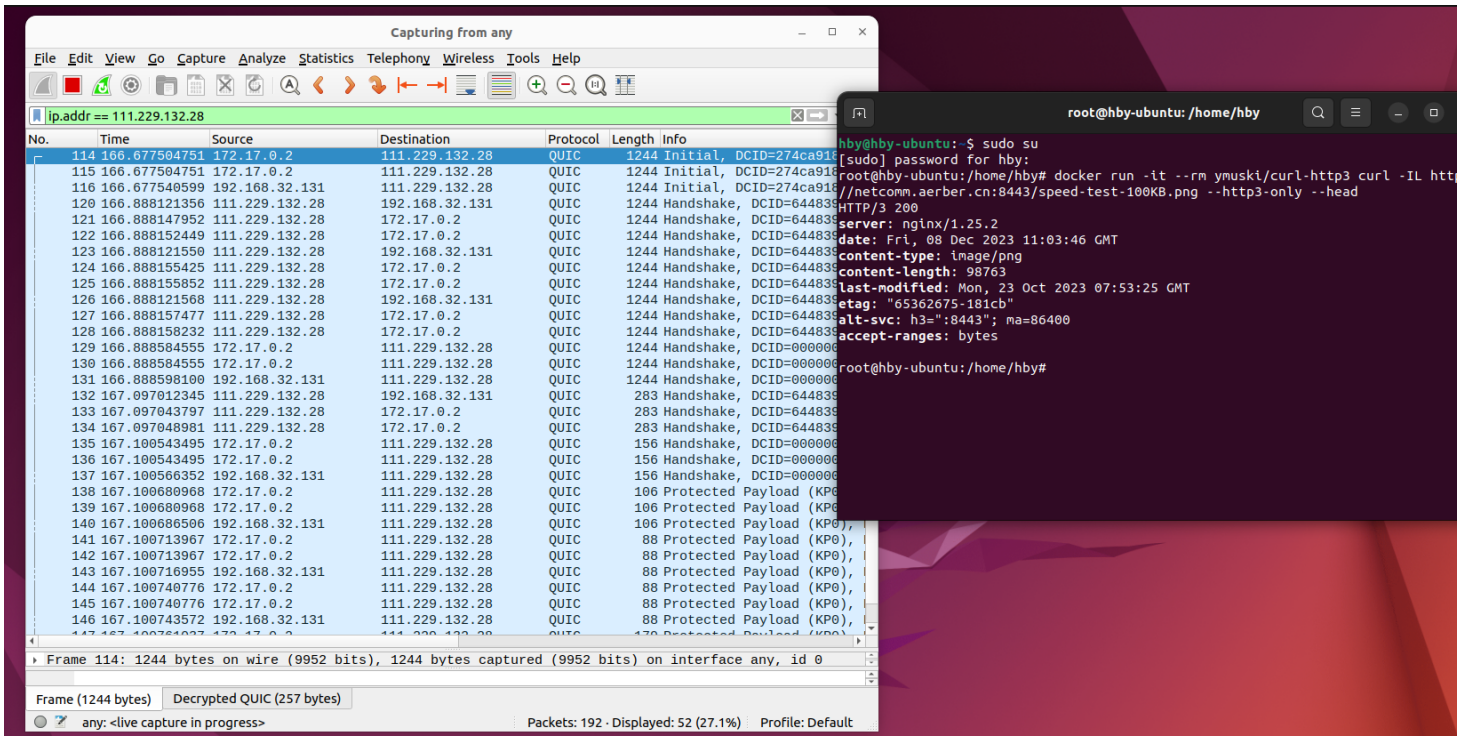
如下在ubuntu中打开wireshark。可选择any或ens33网卡来进行检测



选择any并设置过滤器规则，这里设置 ip.addr == 111.229.132.28 过滤发往或来自 111.229.132.28 的包。



另起一个终端并发送h3请求便可捕获包并分析。

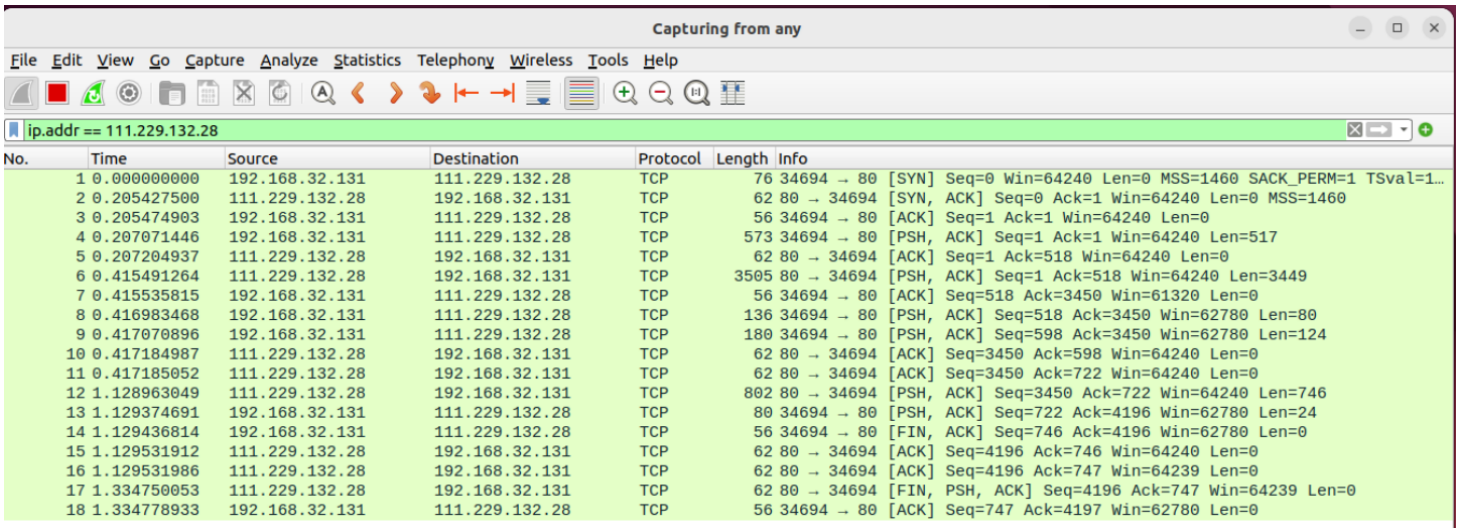


（5）测量结果及分析

在 result.xlsx 的wireshark工作表中。

HTTP1.1

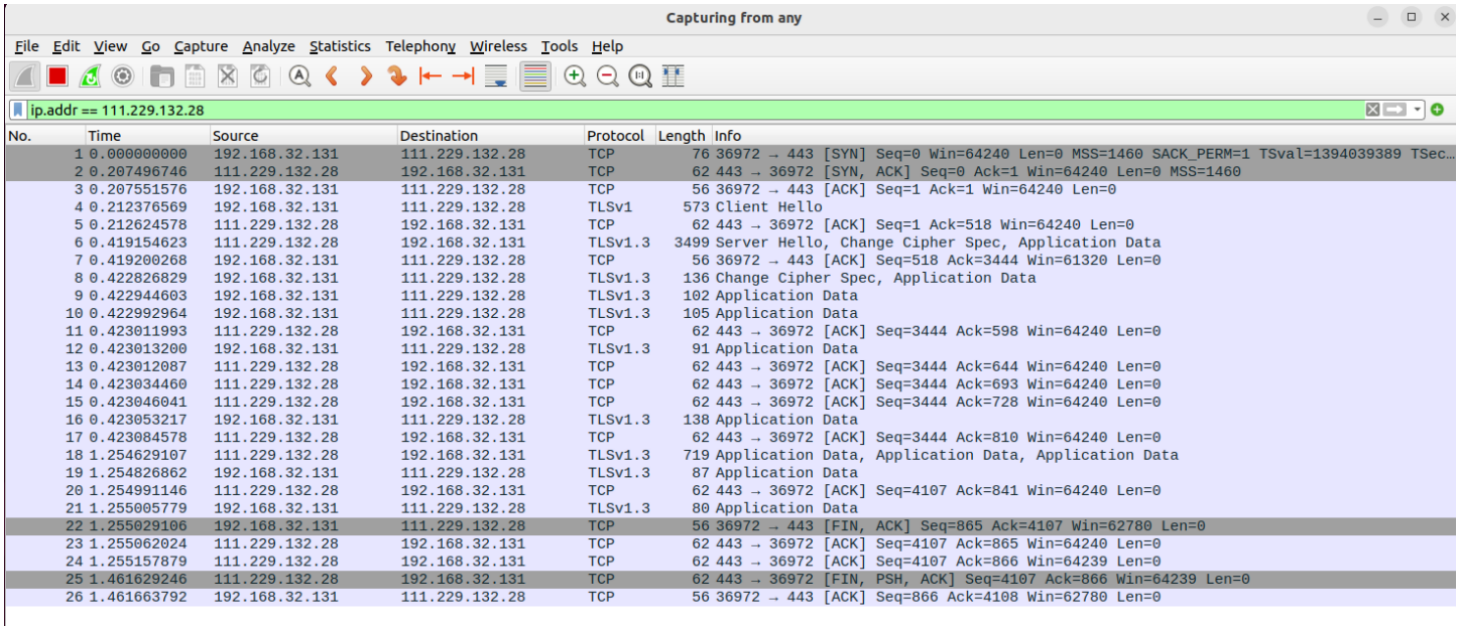
curl -k --http1.1 https://111.229.132.28:80/speed-test-100KB.png --head



可以看出http1.1使用的传输层协议为TCP。SYN标志位用于建立连接。c（client）首先向s（server）发送SYN报文请求建立连接，s再以SYN和ACK回应，c收到ACK再以ACK回应，这样c/s之间的连接就建立起来了。ACK标志位用于确认收到数据，PSH用于推送数据，c/s之间使用这两种标志进行数据通信。FIN用于结束连接，c/s双方都确认了FIN后连接终止。

HTTP2

curl -k --http2 https://111.229.132.28:443/speed-test-100KB.png --head

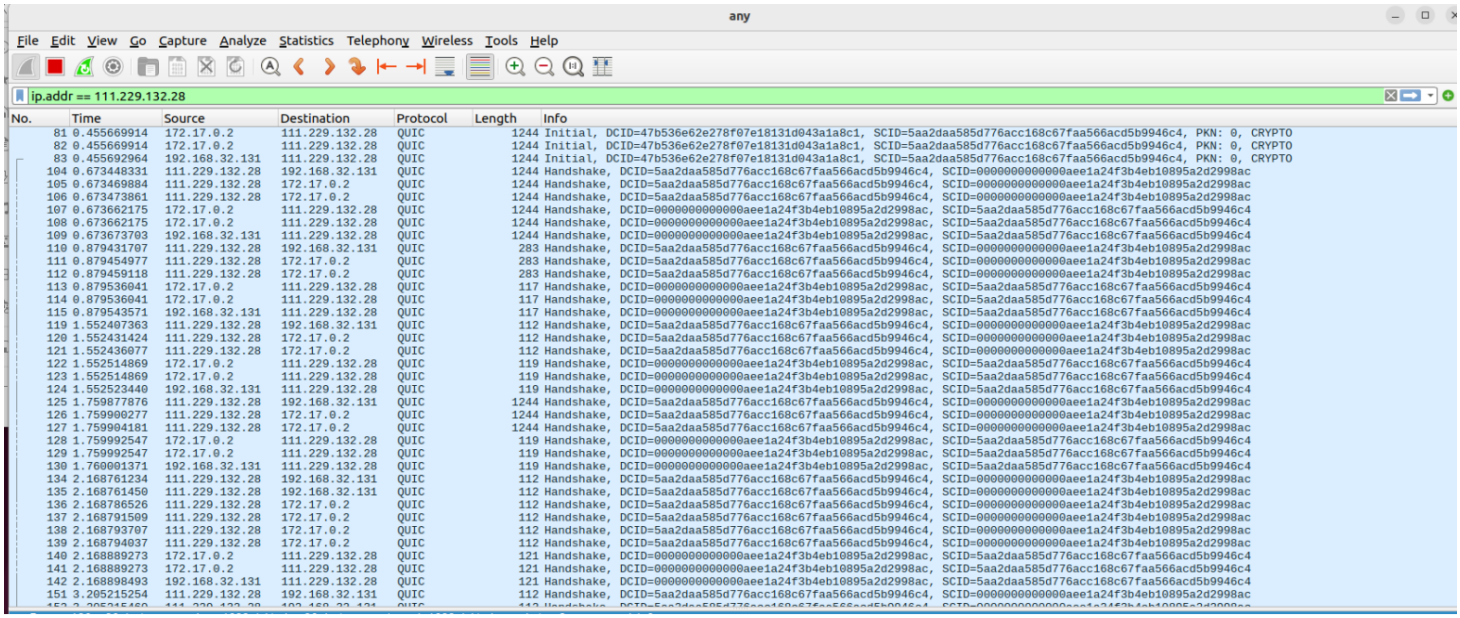


No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.32.131	111.229.132.28	TCP	76	36972 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=1394039389 TSec…
2	0.207496746	111.229.132.28	192.168.32.131	TCP	62	443 → 36972 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
3	0.207551576	192.168.32.131	111.229.132.28	TCP	56	36972 → 443 [ACK] Seq=1 Ack=1 Win=64240 Len=0
4	0.212376569	192.168.32.131	111.229.132.28	TLSv1	573	Client Hello
5	0.212624578	111.229.132.28	192.168.32.131	TCP	62	443 → 36972 [ACK] Seq=1 Ack=518 Win=64240 Len=0
6	0.419154623	111.229.132.28	192.168.32.131	TLSv1.3	3499	Server Hello, Change Cipher Spec, Application Data
7	0.419200268	192.168.32.131	111.229.132.28	TCP	56	36972 → 443 [ACK] Seq=518 Ack=3444 Win=61320 Len=0
8	0.422826829	192.168.32.131	111.229.132.28	TLSv1.3	136	Change Cipher Spec, Application Data
9	0.422944603	192.168.32.131	111.229.132.28	TLSv1.3	102	Application Data
10	0.422992964	192.168.32.131	111.229.132.28	TLSv1.3	105	Application Data
11	0.423011993	111.229.132.28	192.168.32.131	TCP	62	443 → 36972 [ACK] Seq=3444 Ack=598 Win=64240 Len=0
12	0.423013200	192.168.32.131	111.229.132.28	TLSv1.3	91	Application Data
13	0.423012087	111.229.132.28	192.168.32.131	TCP	62	443 → 36972 [ACK] Seq=3444 Ack=644 Win=64240 Len=0
14	0.423034460	111.229.132.28	192.168.32.131	TCP	62	443 → 36972 [ACK] Seq=3444 Ack=693 Win=64240 Len=0
15	0.423046041	111.229.132.28	192.168.32.131	TCP	62	443 → 36972 [ACK] Seq=3444 Ack=728 Win=64240 Len=0
16	0.423053217	192.168.32.131	111.229.132.28	TLSv1.3	138	Application Data
17	0.423084578	111.229.132.28	192.168.32.131	TCP	62	443 → 36972 [ACK] Seq=3444 Ack=810 Win=64240 Len=0
18	1.254629107	111.229.132.28	192.168.32.131	TLSv1.3	719	Application Data, Application Data, Application Data
19	1.254826862	192.168.32.131	111.229.132.28	TLSv1.3	87	Application Data
20	1.254991146	111.229.132.28	192.168.32.131	TCP	62	443 → 36972 [ACK] Seq=4107 Ack=841 Win=64240 Len=0
21	1.255005779	192.168.32.131	111.229.132.28	TLSv1.3	80	Application Data
22	1.255029106	192.168.32.131	111.229.132.28	TCP	56	36972 → 443 [FIN, ACK] Seq=865 Ack=4107 Win=62780 Len=0
23	1.255062024	111.229.132.28	192.168.32.131	TCP	62	443 → 36972 [ACK] Seq=4107 Ack=865 Win=64240 Len=0
24	1.255157879	111.229.132.28	192.168.32.131	TCP	62	443 → 36972 [ACK] Seq=4107 Ack=866 Win=64239 Len=0
25	1.461629246	111.229.132.28	192.168.32.131	TCP	62	443 → 36972 [FIN, PSH, ACK] Seq=4107 Ack=866 Win=64239 Len=0
26	1.461663792	192.168.32.131	111.229.132.28	TCP	56	36972 → 443 [ACK] Seq=866 Ack=4108 Win=62780 Len=0

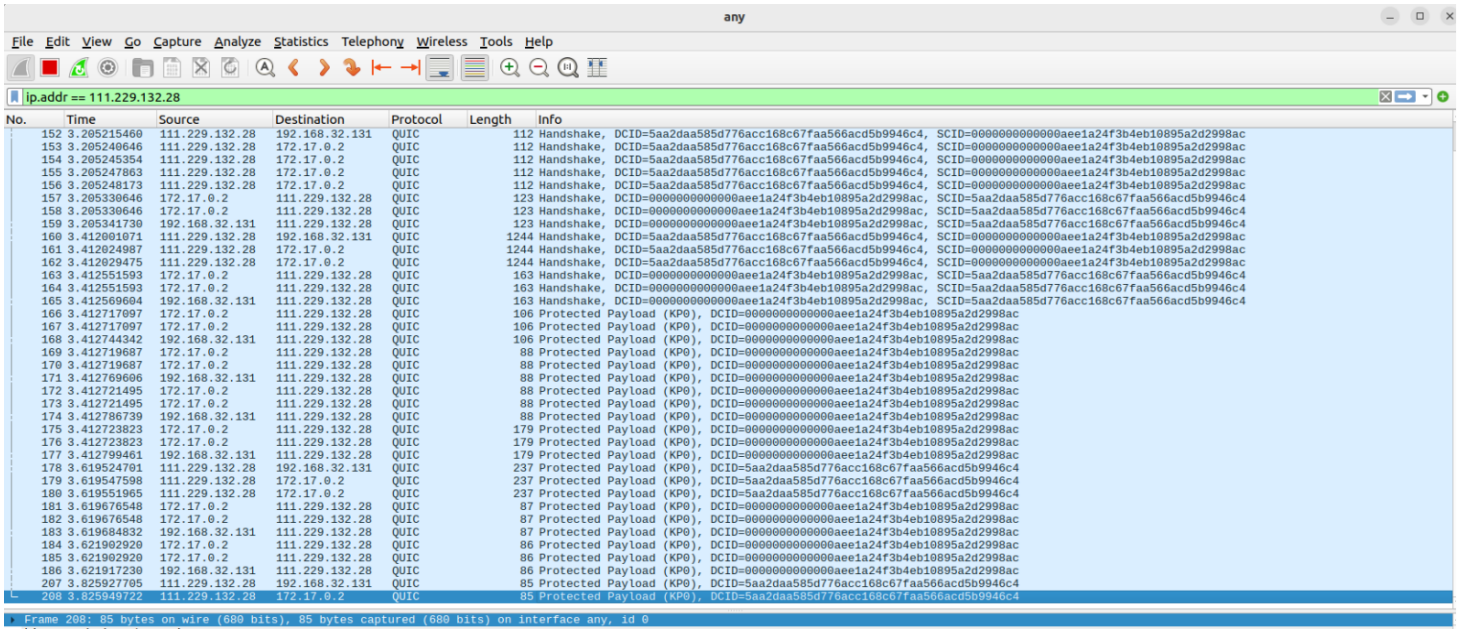
可以看出http2使用的传输层协议为TCP，TLSv1，TLSv1.3。TLS位于 OSI 模型的安全层，用于在应用层和传输层之间提供安全性。TLS协议通常建立在TCP之上，通过对数据进行加密和认证来保护通信的隐私和完整性。http2使用了TLS对数据进行了加密。在使用TCP建立连接后，http2的双方还要再进行一次TLS层的hello的握手。之后的带有数据的报文均通过TLS传递，ACK和FIN此类不包含数据的报文则可继续通过TCP传输。最后终止连接也是使用TCP的FIN标志。

HTTP3

docker run -it --rm ymuski/curl-http3 curl -IL https://netcomm.aerber.cn:8443/speed-test-100KB.png --http3-only --head



No.	Time	Source	Destination	Protocol	Length	Info
81	0.455669914	172.17.0.2	111.229.132.28	QUIC	1244	Initial, DCID=47b536e62e278f07e18131d043a1a8c1, SCID=5aa2daa585d776acc168c67faa566acd5b9946c4, PKN: 0, CRYPTO
82	0.455669914	172.17.0.2	111.229.132.28	QUIC	1244	Initial, DCID=47b536e62e278f07e18131d043a1a8c1, SCID=5aa2daa585d776acc168c67faa566acd5b9946c4, PKN: 0, CRYPTO
83	0.455692964	192.168.32.131	111.229.132.28	QUIC	1244	Initial, DCID=47b536e62e278f07e18131d043a1a8c1, SCID=5aa2daa585d776acc168c67faa566acd5b9946c4, PKN: 0, CRYPTO
104	0.673448331	111.229.132.28	192.168.32.131	QUIC	1244	Handshake, DCID=5aa2daa585d776acc168c67faa566acd5b9946c4, SCID=0000000000000aee1a24f3b4eb10895a2d2998ac
105	0.673469884	111.229.132.28	172.17.0.2	QUIC	1244	Handshake, DCID=5aa2daa585d776acc168c67faa566acd5b9946c4, SCID=0000000000000aee1a24f3b4eb10895a2d2998ac
106	0.673473861	111.229.132.28	172.17.0.2	QUIC	1244	Handshake, DCID=5aa2daa585d776acc168c67faa566acd5b9946c4, SCID=0000000000000aee1a24f3b4eb10895a2d2998ac
107	0.673662175	172.17.0.2	111.229.132.28	QUIC	1244	Handshake, DCID=0000000000000aee1a24f3b4eb10895a2d2998ac, SCID=5aa2daa585d776acc168c67faa566acd5b9946c4
108	0.673662175	172.17.0.2	111.229.132.28	QUIC	1244	Handshake, DCID=0000000000000aee1a24f3b4eb10895a2d2998ac, SCID=5aa2daa585d776acc168c67faa566acd5b9946c4
109	0.673673703	192.168.32.131	111.229.132.28	QUIC	1244	Handshake, DCID=0000000000000aee1a24f3b4eb10895a2d2998ac, SCID=5aa2daa585d776acc168c67faa566acd5b9946c4
110	0.879431707	111.229.132.28	192.168.32.131	QUIC	283	Handshake, DCID=5aa2daa585d776acc168c67faa566acd5b9946c4, SCID=0000000000000aee1a24f3b4eb10895a2d2998ac
111	0.879454977	111.229.132.28	172.17.0.2	QUIC	283	Handshake, DCID=5aa2daa585d776acc168c67faa566acd5b9946c4, SCID=0000000000000aee1a24f3b4eb10895a2d2998ac
112	0.879459118	111.229.132.28	172.17.0.2	QUIC	283	Handshake, DCID=5aa2daa585d776acc168c67faa566acd5b9946c4, SCID=0000000000000aee1a24f3b4eb10895a2d2998ac
113	0.879536041	172.17.0.2	111.229.132.28	QUIC	117	Handshake, DCID=0000000000000aee1a24f3b4eb10895a2d2998ac, SCID=5aa2daa585d776acc168c67faa566acd5b9946c4
114	0.879536041	172.17.0.2	111.229.132.28	QUIC	117	Handshake, DCID=0000000000000aee1a24f3b4eb10895a2d2998ac, SCID=5aa2daa585d776acc168c67faa566acd5b9946c4
115	0.879543571	192.168.32.131	111.229.132.28	QUIC	117	Handshake, DCID=0000000000000aee1a24f3b4eb10895a2d2998ac, SCID=5aa2daa585d776acc168c67faa566acd5b9946c4
119	1.552497363	111.229.132.28	192.168.32.131	QUIC	112	Handshake, DCID=5aa2daa585d776acc168c67faa566acd5b9946c4, SCID=0000000000000aee1a24f3b4eb10895a2d2998ac
120	1.552431424	111.229.132.28	172.17.0.2	QUIC	112	Handshake, DCID=5aa2daa585d776acc168c67faa566acd5b9946c4, SCID=0000000000000aee1a24f3b4eb10895a2d2998ac
121	1.552436077	111.229.132.28	172.17.0.2	QUIC	112	Handshake, DCID=5aa2daa585d776acc168c67faa566acd5b9946c4, SCID=0000000000000aee1a24f3b4eb10895a2d2998ac
122	1.552514869	172.17.0.2	111.229.132.28	QUIC	119	Handshake, DCID=0000000000000aee1a24f3b4eb10895a2d2998ac, SCID=5aa2daa585d776acc168c67faa566acd5b9946c4
123	1.552514869	172.17.0.2	111.229.132.28	QUIC	119	Handshake, DCID=0000000000000aee1a24f3b4eb10895a2d2998ac, SCID=5aa2daa585d776acc168c67faa566acd5b9946c4
124	1.552523440	192.168.32.131	111.229.132.28	QUIC	119	Handshake, DCID=0000000000000aee1a24f3b4eb10895a2d2998ac, SCID=5aa2daa585d776acc168c67faa566acd5b9946c4
125	1.759877876	111.229.132.28	192.168.32.131	QUIC	1244	Handshake, DCID=5aa2daa585d776acc168c67faa566acd5b9946c4, SCID=0000000000000aee1a24f3b4eb10895a2d2998ac
126	1.759900277	111.229.132.28	172.17.0.2	QUIC	1244	Handshake, DCID=5aa2daa585d776acc168c67faa566acd5b9946c4, SCID=0000000000000aee1a24f3b4eb10895a2d2998ac
127	1.759904181	111.229.132.28	172.17.0.2	QUIC	1244	Handshake, DCID=5aa2daa585d776acc168c67faa566acd5b9946c4, SCID=0000000000000aee1a24f3b4eb10895a2d2998ac
128	1.759992547	172.17.0.2	111.229.132.28	QUIC	119	Handshake, DCID=0000000000000aee1a24f3b4eb10895a2d2998ac, SCID=5aa2daa585d776acc168c67faa566acd5b9946c4
129	1.759992547	172.17.0.2	111.229.132.28	QUIC	119	Handshake, DCID=0000000000000aee1a24f3b4eb10895a2d2998ac, SCID=5aa2daa585d776acc168c67faa566acd5b9946c4
130	1.760001371	192.168.32.131	111.229.132.28	QUIC	119	Handshake, DCID=0000000000000aee1a24f3b4eb10895a2d2998ac, SCID=5aa2daa585d776acc168c67faa566acd5b9946c4
134	2.168761234	111.229.132.28	192.168.32.131	QUIC	112	Handshake, DCID=5aa2daa585d776acc168c67faa566acd5b9946c4, SCID=0000000000000aee1a24f3b4eb10895a2d2998ac
135	2.168761450	111.229.132.28	192.168.32.131	QUIC	112	Handshake, DCID=5aa2daa585d776acc168c67faa566acd5b9946c4, SCID=0000000000000aee1a24f3b4eb10895a2d2998ac
136	2.168786526	111.229.132.28	172.17.0.2	QUIC	112	Handshake, DCID=5aa2daa585d776acc168c67faa566acd5b9946c4, SCID=0000000000000aee1a24f3b4eb10895a2d2998ac
137	2.168791509	111.229.132.28	172.17.0.2	QUIC	112	Handshake, DCID=5aa2daa585d776acc168c67faa566acd5b9946c4, SCID=0000000000000aee1a24f3b4eb10895a2d2998ac
138	2.168793707	111.229.132.28	172.17.0.2	QUIC	112	Handshake, DCID=5aa2daa585d776acc168c67faa566acd5b9946c4, SCID=0000000000000aee1a24f3b4eb10895a2d2998ac
139	2.168794037	111.229.132.28	172.17.0.2	QUIC	112	Handshake, DCID=5aa2daa585d776acc168c67faa566acd5b9946c4, SCID=0000000000000aee1a24f3b4eb10895a2d2998ac
140	2.168889273	172.17.0.2	111.229.132.28	QUIC	121	Handshake, DCID=0000000000000aee1a24f3b4eb10895a2d2998ac, SCID=5aa2daa585d776acc168c67faa566acd5b9946c4
141	2.168889273	172.17.0.2	111.229.132.28	QUIC	121	Handshake, DCID=0000000000000aee1a24f3b4eb10895a2d2998ac, SCID=5aa2daa585d776acc168c67faa566acd5b9946c4
142	2.168898493	192.168.32.131	111.229.132.28	QUIC	121	Handshake, DCID=0000000000000aee1a24f3b4eb10895a2d2998ac, SCID=5aa2daa585d776acc168c67faa566acd5b9946c4
151	3.205215254	111.229.132.28	192.168.32.131	QUIC	112	Handshake, DCID=5aa2daa585d776acc168c67faa566acd5b9946c4, SCID=0000000000000aee1a24f3b4eb10895a2d2998ac



可以看出http3使用的传输层协议为QUIC。QUIC（Quick UDP Internet Connections）是一个基于UDP的传输层协议，旨在提供更快的连接建立和更低的延迟。Initial是QUIC协议中建立新连接时的初始阶段。在这个阶段，客户端和服务器之间进行初始通信以建立连接。Handshake是QUIC协议中的安全建立连接的阶段。在这个阶段，客户端和服务器交换加密信息以建立安全的通信渠道。Protected Payload是在QUIC连接建立并完成握手后传输的数据。这些数据受到加密保护，并通过QUIC通道进行传输。QUIC协议Handshake过程基于TLS 1.3，数据传输过程基于UDP。最后QUIC关闭连接也是通过Protected Payload进行。

四、使用弱网工具

（1）环境：windows

（2）工具：

Chrome 120.0.6099.71（正式版本） （64 位）

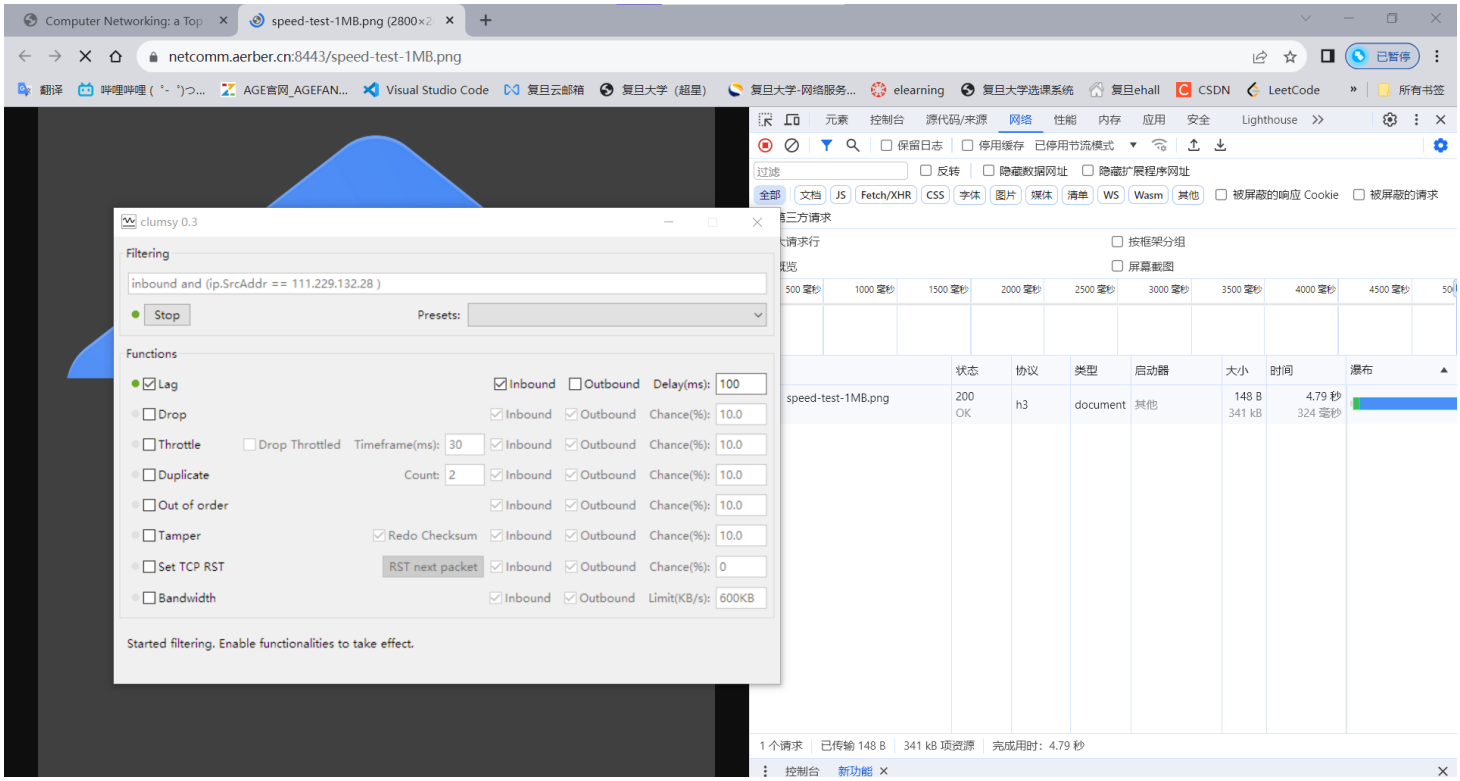
clumsy 0.3

（3）测量对象：不同端口下的1MB.png文件的加载时间。分别测量以下7种情况：

1. 正常情况
2. 延迟100ms
3. 10%节流
4. 10%重复
5. 10%乱序
6. 限制带宽60KB/s
7. 10%损坏

（4）测量过程截图：

如下所示：clumsy中的选项一闪一闪✨即为正在模拟弱网。



（5）测量结果

为减小误差，测量结果为测3次取平均值，在Excel中记录，结果在 result.xlsx 的using_chumsy工作表中。

HTTP1.1

HTTP1.1	结果1/s	结果2/s	结果3/s	平均结果/s
normal	29.96	31.54	29.93	30.48
lag-100ms	66	43.55	47.24	52.26
drop-10%	120	120	120	120.00
throttle-10%	52.13	50.94	38.51	47.19
duplicate-10%	45.01	41.41	35.27	40.56
out-of-order-10%	44.56	41.93	42.7	43.06
bandwidth-60KB/s	41.16	37.15	37.23	38.51
tamper-10%	失败	失败	失败	#DIV/0!

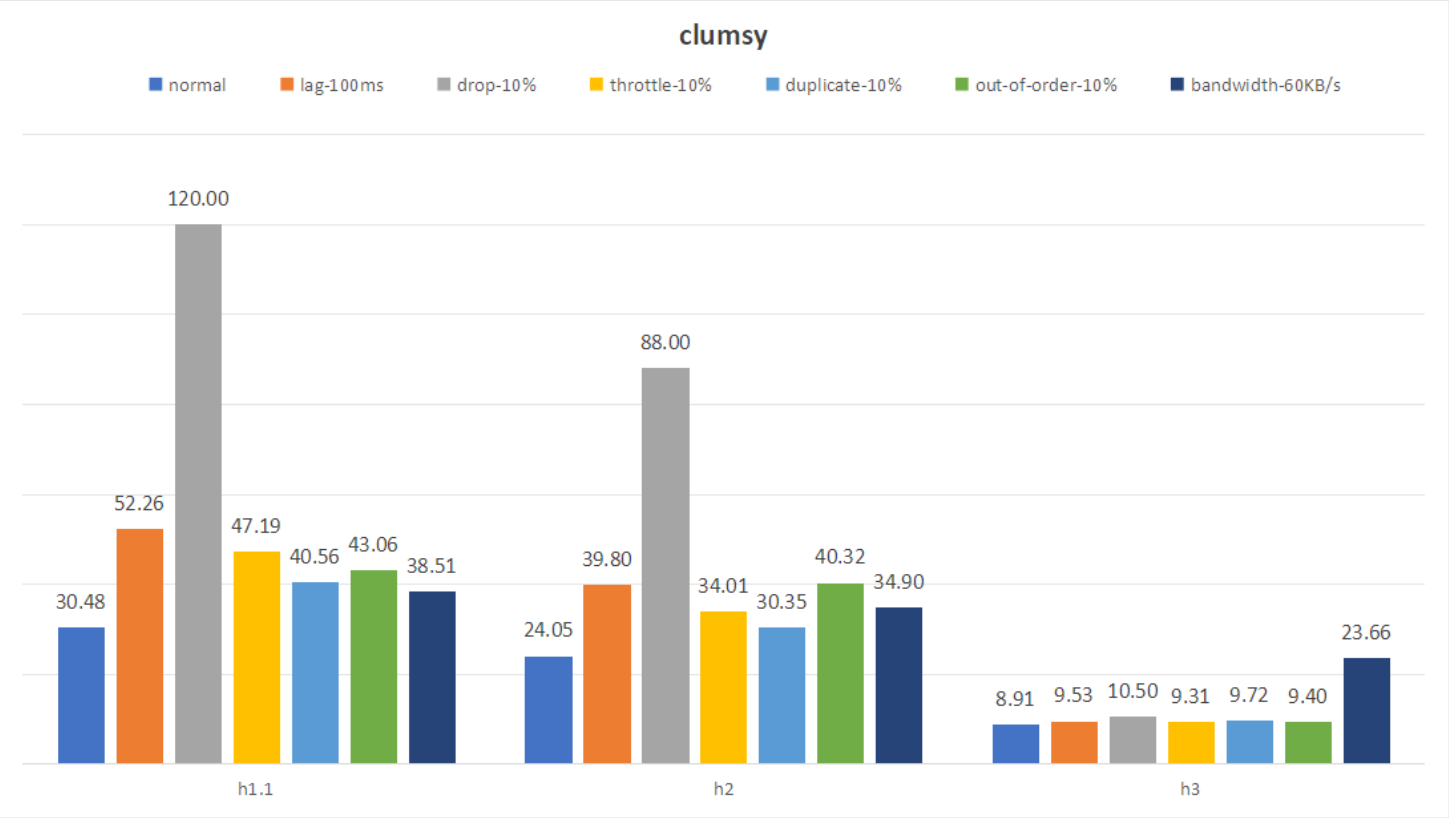
HTTP2

HTTP2	结果1/s	结果2/s	结果3/s	平均结果/s
normal	22.11	25.59	24.45	24.05
lag-100ms	41.48	41.83	36.09	39.80
drop-10%	84	90	90	88.00
throttle-10%	37.07	35.43	29.54	34.01
duplicate-10%	31.08	31.91	28.07	30.35
out-of-order-10%	47.45	34.99	38.53	40.32
bandwidth-60KB/s	36.57	35.53	32.59	34.90
tamper-10%	失败	失败	失败	#DIV/0!

HTTP3

HTTP3	结果1/s	结果2/s	结果3/s	平均结果/s
normal	9.01	8.81	8.90	8.91
lag-100ms	9.11	9.89	9.59	9.53
drop-10%	10.53	10.12	10.84	10.50
throttle-10%	9.12	9.63	9.17	9.31
duplicate-10%	9.21	10.89	9.06	9.72
out-of-order-10%	9.34	9.63	9.22	9.40
bandwidth-60KB/s	22.68	25.39	22.91	23.66
tamper-10%	失败	失败	失败	#DIV/0!

（6）分析



如上图所示：normal为正常情况，在每簇的最左侧，用来进行对比。可以看出，h3在弱网环境中表现是最好的，在不限制带宽的情况下，10%的包错误和100ms的延迟对其影响都不大；而对于h1.1和h2来说，二者表现比较类似。100ms的延迟就已经对二者的传输速度影响比较大了，而10%的丢包对于传输速度的影响尤为巨大，所用时间甚至多了3-4倍。其他类型的10%包错误对其均有不同程度的影响，但都没有延迟100ms和丢包10%那样显著。而在限制带宽为60KB的情况下，这个带宽限制已经低于服务器提供的带宽，故三种协议传输速度均有所下降。其中h3对于带宽最为敏感，推测是由于h3对于带宽的利用率更高，因为文件大小为1MB，h3所用时间最接近数据传输时间。个人感觉h1.1,h2二者与h3的区别主要是由TCP和UDP的区别导致的，在进行测量的过程中，h3结果稳定性要显著高于h1.1和h2，速度也要快上不少，推测是由于UDP没有拥塞控制机制，故表现要更好。