

Visualize application traffic using Wireshark



Extend your Wireshark expression with the intelligent use of the Graph function.

Materials: all trace files and profiles are here

<https://www.ikeriri.ne.jp/sharkfest/VisualizeApplicationTraffic.zip>

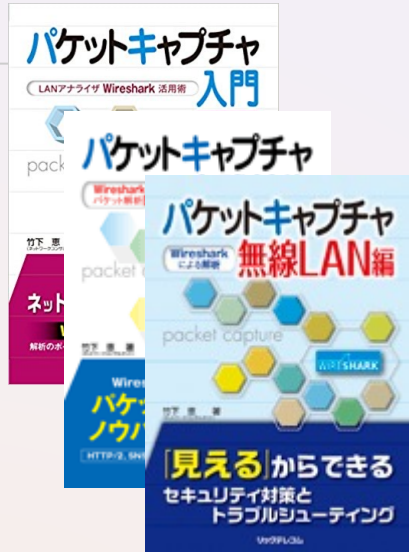


02 12:15-13:30 SGT

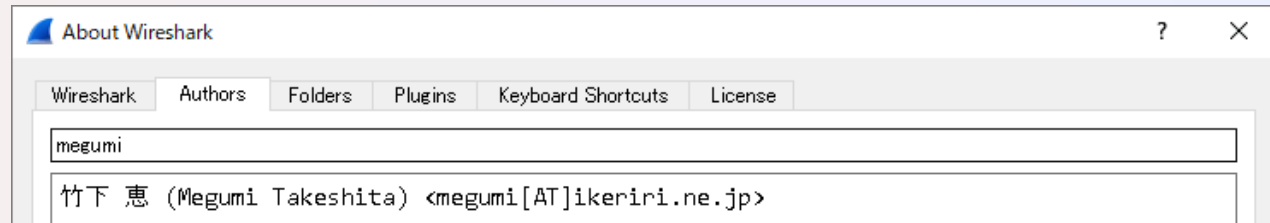
on 17th April 2023

Megumi Takeshita
Ikeriri network service

Megumi Takeshita, packet otaku



- Founder, ikeriri network service co., ltd
- Reseller of CACE technologies in 2008
- Worked SE/IS at BayNetwork, Nortel
- Wrote 10+ books about Wireshark
- Instruct Wireshark to JSDF and other company
- lecturer of CHUO university
- Reseller of packet capture / wireless-tools
- One of the contributors of Wireshark
- Translate Wireshark into Japanese



Session Details

We know Wireshark is good at trace file investigation with 3000+ over 3000 protocol dissectors, but it is also an excellent tool for visualizing traffic with a series of Graph functions. Megumi shows you TIPS and Tricks to use Wireshark as a graphical chart creation tool. Colourful charts tell you the traffic flow situation quickly, and those who are not users of Wireshark will also be satisfied.

Materials

all trace files and Wireshark profiles are here

<https://www.ikeriri.ne.jp/sharkfest/VisualizeApplicationTraffic.zip>

Visualize Application Traffic



- Wireshark has over 3k protocols that include many applications from HTTP to Gaming.
- We can utilize over 285k fields for visualizing application traffic.
- A most easy and common ways for visualization is the I/O graph with adequate display filter.
- Download materials and copy profile settings
<https://www.ikeriri.ne.jp/sharkfest/VisualizeApplicationTraffic.zip>

Copy Wireshark settings

- Open Wireshark, Help>About Wireshark and choose the Folders tab to find personal configuration
- “tshark -G folders” also works in CLI

About Wireshark

Wireshark	Authors	Folders	Plugins	Keyboard Shortcuts	Acknowledgments	License
Filter by path						
Name	Location	Typical Files				
"File" dialogs	C:\Users\TakeshitaMegumi\O...arkfest\SharkfestASIA2023\	capture files				
Global Extcap path	C:\Program Files\Wireshark\extcap	Extcap Plugins search path				
Global Lua Plugins	C:\Program Files\Wireshark\plugins	lua scripts				
Global Plugins	C:\Program Files\Wireshark\plugins\4.0	binary plugins				
Global configuration	C:\Program Files\Wireshark	dfilters, preferences, manuf, ...				
MIB/PIB path		SMI MIB/PIB search path				
MaxMind DB path	C:\ProgramData\GeoIP	MaxMind DB database search path				
MaxMind DB path	C:\GeoIP	MaxMind DB database search path				
Personal Extcap path	C:\Users\TakeshitaMegumi\...Roaming\Wireshark\extcap	Extcap Plugins search path				
Personal Lua Plugins	C:\Users\TakeshitaMegumi\...Roaming\Wireshark\plugins	lua scripts				
Personal Plugins	C:\Users\TakeshitaMegumi\...ing\Wireshark\plugins\4.0	binary plugins				
Personal configuration	C:\Users\TakeshitaMegumi\AppData\Roaming\Wireshark	dfilters, preferences, ethers, ...				
Program	C:\Program Files\Wireshark	program files				
System	C:\Program Files\Wireshark	ethers, ipxnets				
Temp	C:\Users\TAKESH~1\AppData\Local\Temp	untitled capture files				

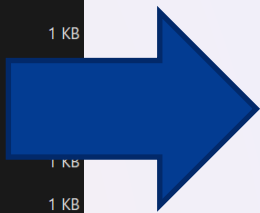
```
C:\Users\TakeshitaMegumi>tshark -G folders
Temp: C:\Users\TAKESH~1\AppData\Local\Temp
Personal configuration: C:\Users\TakeshitaMegumi\AppData\Roaming\Wireshark
Global configuration: C:\Program Files\Wireshark
System: C:\Program Files\Wireshark
Program: C:\Program Files\Wireshark
Personal Plugins: C:\Users\TakeshitaMegumi\AppData\Roaming\Wireshark\plugins\4.0
Global Plugins: C:\Program Files\Wireshark\plugins\4.0
Personal Lua Plugins: C:\Users\TakeshitaMegumi\AppData\Roaming\Wireshark\plugins
Global Lua Plugins: C:\Program Files\Wireshark\plugins
Extcap path: C:\Program Files\Wireshark\extcap
MaxMind database path: C:\ProgramData\GeoIP
MaxMind database path: C:\GeoIP
```

Personal Configuration

- Unzip and copy Wireshark configuration contents to our personal configuration folder
- There are Default and TCP/UDP application profiles

Wireshark configuration >

名前	更新日時	種類	サイズ
GeoIP	2023/04/06 17:55	ファイル フォルダー	
profiles	2023/04/06 17:55	ファイル フォルダー	
decode_as_entries	2023/04/06 17:55	ファイル	1 KB
dfilter_macros	2023/04/05 16:00	ファイル	
io_graphs	2023/04/05 17:54	ファイル	
language	2023/04/06 17:55	ファイル	1 KB
maxmind_db_paths	2023/04/06 17:54	ファイル	1 KB
preferences	2023/04/06 17:55	ファイル	227 KB
recent	2023/04/06 17:55	ファイル	4 KB
recent_common	2023/04/06 17:55	ファイル	6 KB

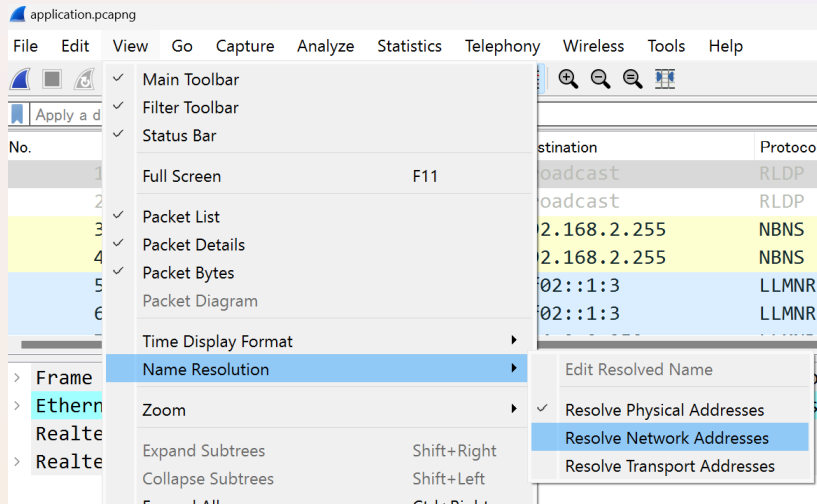


C:\Users\TakeshitaMegumi\AppData\Roaming\Wireshark

名前	更新日時
GeoIP	2023/03/23 20:08
profiles	2023/04/05 21:39
decode_as_entries	2023/04/06 17:55
dfilter_macros	2023/04/05 16:00
io_graphs	2023/04/05 17:54
language	2023/04/06 17:55
maxmind_db_paths	2023/04/06 17:54
preferences	2023/04/06 17:55
recent	2023/04/06 17:55
recent_common	2023/04/06 17:55

Unzip trace files

- Unzip tracefile.zip, there are three trace files,
application.pcapng (TCP/UDP applications)
application2.pcapng (HTTP/echo test traffic)
application3.pcapng (Youtube and Zoom traffic)



- open application.pcapng,
View>Name Resolution
>Resolve Network Address

Protocol Hierarchy Statistics and Multicast

Wireshark - Protocol Hierarchy Statistics - application.png

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s	PDUs
Frame	100.0	74657	100.0	70045272	3602 k	0	0	0	74657
Ethernet	100.0	74657	1.6	1143545	58 k	0	0	0	74657
Address Resolution Protocol	2.6	1948	0.1	86426	4444	1948	16426	4444	1948
Configuration Test Protocol (loopback)	0.2	124	0.0	5704	293	0	0	0	124
Data	0.2	124	0.0	5208	267	124	5208	267	124
Data	0.0	5	0.0	230	11	5	230	11	5
Internet Protocol Version 4	98.3	69650	2.0	1393304	71 k	0	0	0	69650
Internet Control Message Protocol	0.0	17	0.0	1566	80	0	0	0	17
Domain Name System	0.0	17	0.0	954	49	17	954	49	17
Internet Group Management Protocol	0.1	76	0.0	1056	54	76	1056	54	76
Transmission Control Protocol	74.1	53353	83.7	58610324	3014 k	44878	48793506	2509 k	53353
Data	0.1	106	0.1	54835	2820	106	54835	2820	106
Hypertext Transfer Protocol	0.1	73	0.7	486597	25 k	38	14002	720	73
Malformed Packet	0.0	4	0.0	0	0	4	0	0	4
Transport Layer Security	13.8	10292	80.3	56234919	2892 k	10290	51680429	2657 k	10643
User Datagram Protocol	19.0	14204	0.2	113632	5844	0	0	0	14204
Control And Provisioning of Wireless Access Points - Control	0.0	29	0.0	21460	1103	29	21460	1103	29
Data	0.2	129	0.1	52132	2681	129	52132	2681	129
Domain Name System	1.4	1056	0.1	70429	3622	1056	70429	3622	1056
Dynamic Host Configuration Protocol	0.0	33	0.0	9868	507	33	9868	507	33
Link-local Multicast Name Resolution	0.8	595	0.0	15987	822	595	15987	822	595
Multicast Domain Name System	1.4	1024	0.1	95333	4903	1024	95333	4903	1024
NetBIOS Datagram Service	0.0	20	0.0	3897	200	0	0	0	20
SMB (Server Message Block Protocol)	0.0	20	0.0	2257	116	0	0	0	20
NetBIOS Name Service	13.5	949	0.1	47738	2455	949	47738	2455	949
QUIC IETF	13.5	10050	11.7	8166400	420 k	10050	8156185	417 k	10109
Simple Network Management Protocol	0.0	21	0.0	1638	84	21	1638	84	21
Simple Service Discovery Protocol	0.4	298	0.1	52734	2712	298	52734	2712	298
Internet Protocol Version 6	1.9	1453	0.1	58120	2989	0	0	0	1453
Internet Control Message Protocol v6	0.1	109	0.0	3876	199	109	3876	199	109
Data	1.8	1344	0.0	10752	552	0	0	0	1344
DHCPv6	0.0	2	0.0	111	5	2	111	5	2
Link-local Multicast Name Resolution	0.7	553	0.0	14981	770	553	14981	770	553
Multicast Domain Name System	1.0	747	0.1	83106	4274	747	83106	4274	747
Simple Service Discovery Protocol	0.0	6	0.0	666	34	6	666	34	6
Realtek Layer 2 Protocol	2.0	1477	0.1	67942	3494	0	0	0	1477
Realtek Loop Detection Protocol	2.0	1477	0.1	67942	3494	1477	67942	3494	1477

No display filter.

Close Copy Help

- Statistics>Protocol Hierarchy Statistics and check the application protocols
- 93% is IPv4, IPv6 is few.
- There are TCP protocols, HTTP, TLS and others.
- Little UDP protocols, DNS, DHCP, SMB, QUIC and others
- Statistics>UDP Multicast Streams to check multicast

Wireshark - UDP Multicast Streams - application.png

Source Address	Source Port	Destination Address	Destination Port	Packets	Packets/s	Avg BW (bps)	Max BW (bps)	Max I
fe80::d071:2f8c:c51d	5353	ff02::fb	5353	2	0.02	109	0	1 / 1
fe80::cc5c111a:855:fe4	5353	ff02::fb	5353	196	1.27	964	0	1 / 1
fe80::cc5c111a:855:fe4	64122	ff02::1b	5355	2	4.92	3501	0	1 / 1
fe80::cc5c111a:855:fe4	63807	ff02::1b	5355	2	4.92	3502	0	1 / 1
fe80::cc5c111a:855:fe4	61186	ff02::1b	5355	2	4.87	3465	0	1 / 1
fe80::cc5c111a:855:fe4	59404	ff02::1b	5355	2	4.90	3486	0	1 / 1
fe80::cc5c111a:855:fe4	49219	ff02::1b	5355	2	4.89	3481	0	1 / 1
fe80::cc5c111a:855:fe4	54395	ff02::1b	5355	2	4.90	3491	0	1 / 1
fe80::cc5c111a:855:fe4	65121	ff02::1b	5355	2	4.90	3486	0	1 / 1
fe80::cc5c111a:855:fe4	57864	ff02::1b	5355	2	4.91	3495	0	1 / 1
fe80::cc5c111a:855:fe4	58734	ff02::1b	5355	2	4.93	3799	0	1 / 1
fe80::cc5c111a:855:fe4	59616	ff02::1b	5355	2	4.88	3477	0	1 / 1
fe80::cc5c111a:855:fe4	53927	ff02::1b	5355	2	4.89	3483	0	1 / 1
fe80::cc5c111a:855:fe4	55235	ff02::1b	5355	2	4.91	3494	0	1 / 1
fe80::cc5c111a:855:fe4	55672	ff02::1b	5355	2	4.88	3477	0	1 / 1
fe80::cc5c111a:855:fe4	50759	ff02::1b	5355	2	4.90	3487	0	1 / 1
fe80::cc5c111a:855:fe4	53888	ff02::1b	5355	2	4.90	3486	0	1 / 1

717 streams, avg bw: 22 Mbps, max bw: 82 Mbps, max burst: 17 / 100ms, max buffer: 624B

Burst measurement interval (ms): 100 Burst alarm threshold (packets): 50 Buffer alarm threshold (B): 10000

Stream empty speed (Kb/s): 5000 Total empty speed (Kb/s): 100000

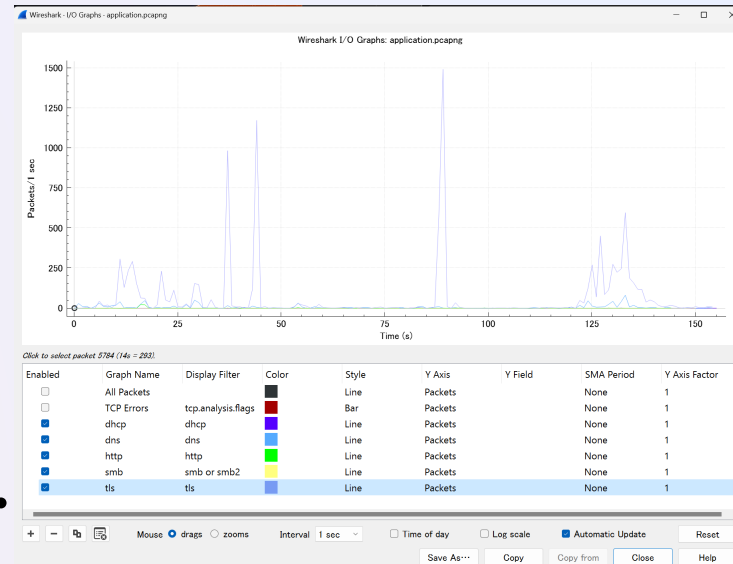
Display filter:

Copy Save as... Close

Basic Graph by application protocols



- The primary way to visualize the application traffic is using I/O Graph by types of application protocols
- Statistics>I/O Graph and check items on dhcp, dns, http, smb, tls
- The graph tells us each packet by application protocols with interval 1 second
- We can change Y axis from Packets to Bits to find bitrates.



Let's start with Visualization

Enabled	Graph Name	Display Filter	Color	Style	Y Axis	Y Field	SMA Period	Y Axis Factor
☐	dhcp	dhcp	Blue	Line	Packets		None	1
☐	dns	dns	Orange	Line	Packets		None	1
☐	http	http	Green	Line	Packets		None	1
☐	smb	smb or smb2	Purple	Line	Packets		None	1
☐	tls	tls	Red	Line	Packets		None	1

+

-

🔍

📄

Mouse ☒ drags ☐ zooms

Interval

1 sec

☐ Time of day ☐ Log scale ☒ Automatic Update

Reset

Save As...

Copy

Copy from ▾

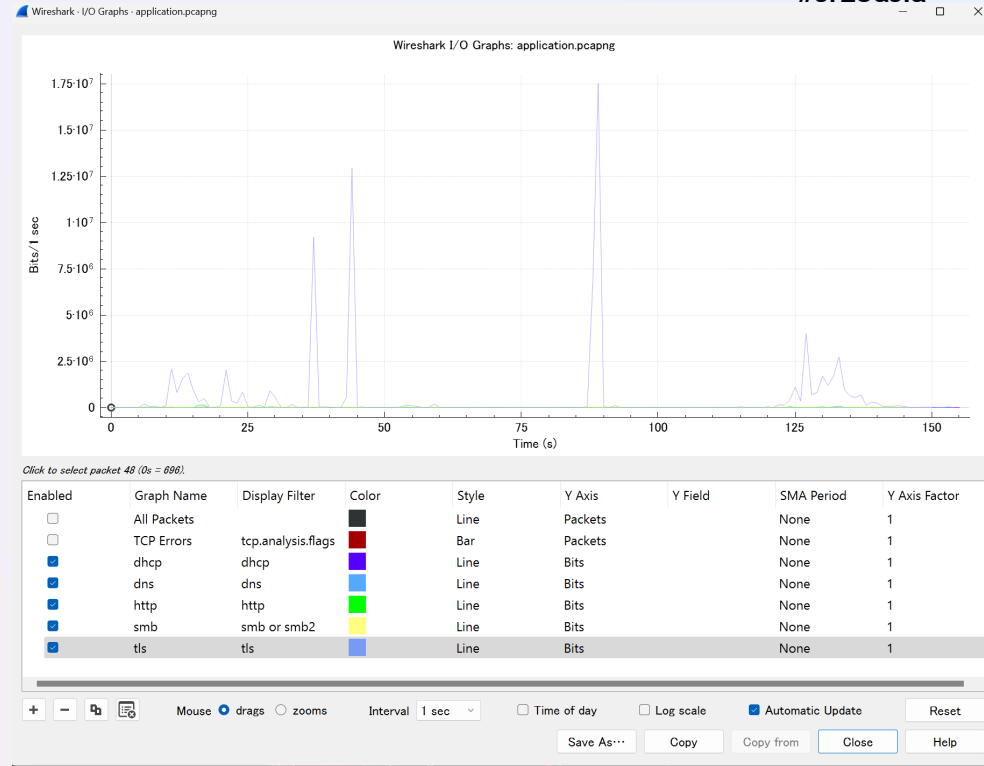
Close

Help

- Let's start Visualization, create an application traffic bandwidth graph. Select Statistics>I/O Graph
- add items those Graph Name as dhcp, dns, http, smb, tls, and each Display filter is the same, set Style as Line, set Y Axis as Bits, set interval as 1 sec

Application traffic bandwidth

- Change Y axis from Packts to Bit, and check Interval is 1 sec
- We can check bitrates by each application
 10^6 says Mbps
 10^9 says Gbps



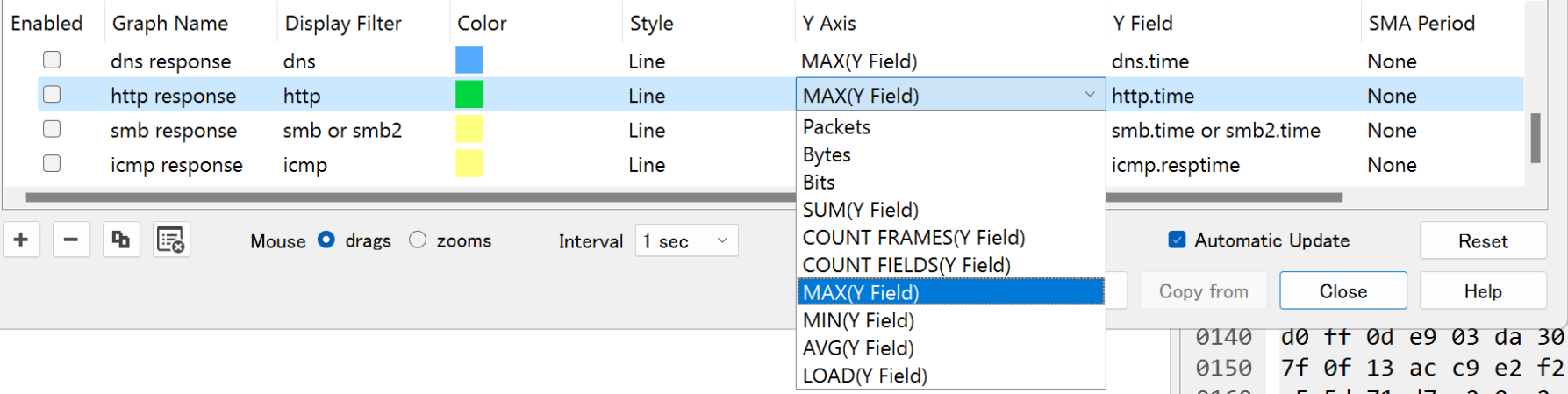
Response time by Application



- Some application dissector has the Wireshark Generated time fields[] that tell us response time.
- http.time field in the http response packet means the time between HTTP request and response.
- dns.time tell us the DNS response time and icmp.resptime and smb.time or smb2.time is also contains response time

```
▼ Hypertext Transfer Protocol
  > HTTP/1.1 200 OK\r\n
  > Content-Length: 14\r\n
    Date: Tue, 04 Apr 2023 22:19:19 GMT\r\n
    Connection: close\r\n
    Content-Type: text/plain\r\n
    Cache-Control: max-age=30, must-revalidate\r\n
    \r\n
    [HTTP response 1/1]
    [Time since request: 0.014466000 seconds]
    [Request in frame: 815]
    [Request URI: http://www.msftncsi.com/ncsi.txt]
    File Data: 14 bytes
```

Worst time within 1 seconds



Enabled	Graph Name	Display Filter	Color	Style	Y Axis	Y Field	SMA Period
☐	dns response	dns	Blue	Line	MAX(Y Field)	dns.time	None
☒	http response	http	Green	Line	MAX(Y Field)	http.time	None
☐	smb response	smb or smb2	Yellow	Line	Packets	smb.time or smb2.time	None
☐	icmp response	icmp	Yellow	Line	Bytes	icmp.resptime	None
					Bits		
					SUM(Y Field)		
					COUNT FRAMES(Y Field)		
					COUNT FIELDS(Y Field)		
					MIN(Y Field)		
					AVG(Y Field)		
					LOAD(Y Field)		

- Statistics>I/O Graph and check off all enabled select dns/http/smb/icmp response items enter Y fields as dns.time/http.time/smb.time or smb2.time/icmp.resptime and choose MAX (Y Field) in Y Axis listbox and enable these items.

Application/Service by specified IPv4/IPv6 address range

- Nowadays, most Internet application traffic use TLS and some Google service use QUIC.
- It is difficult to determine each application by protocol filter without keys or proxy server.
How about IPv4/IPv6 address?
- For example, Microsoft disclose Office 365 URLs and IP address ranges at the support site
<https://learn.microsoft.com/en-us/microsoft-365/enterprise/urls-and-ip-address-ranges?view=o365-worldwide>

IPv4/IPv6 address range of Office365 / Exchange endpoints

Office 365 URLs and IP address ranges

Article • 03/02/2023 • 9 minutes to read • [12 contributors](#) [Feedback](#)

In this article

[Exchange Online](#)
[SharePoint Online and OneDrive for Business](#)
[Skype for Business Online and Microsoft Teams](#)
[Microsoft 365 Common and Office Online](#)
[Related Topics](#)

Office 365 requires connectivity to the Internet. The endpoints below should be reachable for customers using Office 365 plans, including Government Community Cloud (GCC).

[Office 365 Worldwide \(+GCC\)](#) | [Office 365 operated by 21 Vianet](#) | [Office 365 U.S. Government DoD](#) | [Office 365 U.S. Government GCC High](#) |

Notes	Download	Use
Last updated: 03/29/2023 - Change Log subscription	Download: all required and optional destinations in one JSON formatted list .	Use: our proxy PAC files

- Download JSON formatted list and convert it into Wireshark display filter style
- Wireshark can use network addresses with / mask length like `ip.addr==13.107.6.152/31`, `ipv6.addr==2603:1006::/40`

Convert JSON IP range into Display filter strings

- Add ip.addr== or ipv6.addr== strings

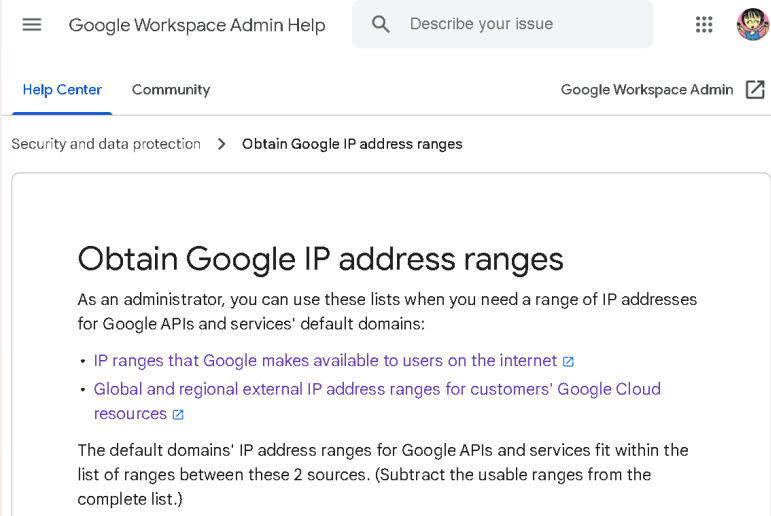
ip.addr==13.107.6.152/31 or ip.addr==13.107.18.10/31 or
 ip.addr==13.107.128.0/22 or ip.addr==23.103.160.0/20 or
 ip.addr==40.96.0.0/13 or ip.addr==40.104.0.0/15 or
 ip.addr==52.96.0.0/14 or ip.addr==131.253.33.215/32 or
 ip.addr==132.245.0.0/16 or ip.addr==150.171.32.0/22 or
 ip.addr==204.79.197.215/32 or ipv6.addr==2603:1006::/40 or
 ipv6.addr==2603:1016::/36 or ipv6.addr==2603:1026::/36 or
 ipv6.addr==2603:1036::/36 or ipv6.addr==2603:1046::/36 or
 ipv6.addr==2603:1056::/36 or ipv6.addr==2620:1ec:4::152/128 or
 ipv6.addr==2620:1ec:4::153/128 or ipv6.addr==2620:1ec:c::10/128 or
 ipv6.addr==2620:1ec:d::10/128 or ipv6.addr==2620:1ec:d::11/128 or
 ipv6.addr==2620:1ec:8f0::/46 or ipv6.addr==2620:1ec:900::/46 or
 ipv6.addr==2620:1ec:a92::152/128 or ipv6.addr==2620:1ec:a92::153/128

```
[
  {
    "id": 1,
    "serviceArea": "Exchange",
    "serviceAreaDisplayName": "Exchange Online",
    "urls": [
      "outlook.office.com",
      "outlook.office365.com"
    ],
    "ips": [
      "13.107.6.152/31",
      "13.107.18.10/31",
      "13.107.128.0/22",
      "23.103.160.0/20",
      "40.96.0.0/13",
      "40.104.0.0/15",
      "52.96.0.0/14",
      "131.253.33.215/32",
      "132.245.0.0/16",
      "150.171.32.0/22",
      "204.79.197.215/32",
      "2603:1006::/40",
      "2603:1016::/36",
      "2603:1026::/36",
      "2603:1036::/36",
      "2603:1046::/36",
      "2603:1056::/36",
      "2620:1ec:4::152/128",
      "2620:1ec:4::153/128",
      "2620:1ec:c::10/128",
      "2620:1ec:c::11/128",
      "2620:1ec:d::10/128",
      "2620:1ec:d::11/128",
      "2620:1ec:8f0::/46",
      "2620:1ec:900::/46",
      "2620:1ec:a92::152/128",
      "2620:1ec:a92::153/128"
    ],
    "tcpPorts": "80,443",
    "udpPorts": "443",
    "expressRoute": true,
    "category": "Optimize",
    "required": true
  },
]
```

Google services and customer's Google Cloud IPv4/IPv6 range



- Google also disclose the IP address range of Google service and end-user's Google Cloud
<https://support.google.com/a/answer/10026322?hl=en>



- Download the latest JSON lists and convert them into Wireshark Display Filter string styles
<https://www.gstatic.com/ipranges/goog.json>

```
{
  "syncToken": "1680660288691",
  "creationTime": "2023-04-04T19:04:48.691064",
  "prefixes": [
    {
      "pv4Prefix": "8.8.4.0/24"
    },
    {
      "pv4Prefix": "8.8.8.0/24"
    },
    {
      "pv4Prefix": "8.34.208.0/20"
    },
    {
      "pv4Prefix": "8.35.192.0/20"
    },
    {
      "pv4Prefix": "23.236.48.0/20"
    },
    {
      "pv4Prefix": "23.251.128.0/19"
    },
    {
      "pv4Prefix": "34.0.0.0/15"
    },
    {
      "pv4Prefix": "34.2.0.0/16"
    },
    {
      "pv4Prefix": "34.3.0.0/23"
    },
    {
      "pv4Prefix": "34.3.3.0/24"
    },
    {
      "pv4Prefix": "34.3.4.0/24"
    },
    {
      "pv4Prefix": "34.3.8.0/21"
    },
    {
      "pv4Prefix": "34.3.16.0/20"
    },
    {
      "pv4Prefix": "34.3.32.0/19"
    },
    {
      "pv4Prefix": "34.3.64.0/18"
    },
    {
      "pv4Prefix": "34.3.128.0/17"
    },
    {
      "pv4Prefix": "34.4.0.0/14"
    },
    {
      "pv4Prefix": "34.8.0.0/13"
    },
    {
      "pv4Prefix": "34.16.0.0/12"
    },
    {
      "pv4Prefix": "34.32.0.0/11"
    },
    {
      "pv4Prefix": "34.64.0.0/10"
    },
    {
      "pv4Prefix": "34.128.0.0/10"
    },
    {
      "pv4Prefix": "35.184.0.0/13"
    },
    {
      "pv4Prefix": "35.192.0.0/14"
    }
  ]
}
```

- Add ip.addr== or ipv6.addr== strings
 ip.addr==108.170.192.0/18 or ip.addr==108.177.0.0/17 or
 ip.addr==130.211.0.0/16 or ip.addr==136.112.0.0/12 or
 ip.addr==142.250.0.0/15 or ip.addr==146.148.0.0/17 or
 ip.addr==162.216.148.0/22 or ip.addr==162.222.176.0/21 or
 ip.addr==172.110.32.0/21 or ip.addr==172.217.0.0/16 or
 ip.addr==172.253.0.0/16 or ip.addr==173.194.0.0/16 or
 ip.addr==173.255.112.0/20 or ip.addr==192.158.28.0/22 or
 ip.addr==192.178.0.0/15 or ip.addr==193.186.4.0/24 or
 ip.addr==199.36.154.0/23 or ip.addr==199.36.156.0/24 or
 ip.addr==199.192.112.0/22 or ip.addr==199.223.232.0/21 or
 ip.addr==207.223.160.0/20 or ip.addr==208.65.152.0/22 or
 ip.addr==208.68.108.0/22 or ip.addr==208.81.188.0/22 or
 ip.addr==208.117.224.0/19 or ip.addr==209.85.128.0/17 or
 ip.addr==216.58.192.0/19 or ip.addr==216.73.80.0/20 or
 ip.addr==216.239.32.0/19 or ipv6.addr==2001:4860::/32 or
 ipv6.addr==2404:6800::/32 or ipv6.addr==2404:f340::/32 or
 ipv6.addr==2600:1900::/28 or ipv6.addr==2606:73c0::/32 or
 ipv6.addr==2607:f8b0::/32 or ipv6.addr==2620:11a:a000::/40 or
 ipv6.addr==2620:120:e000::/40 or ipv6.addr==2800:3f0::/32 or
 ipv6.addr==2a00:1450::/32 or ipv6.addr==2c0f:fb50::/32

Create a Display Filter macro. without any parameters

- As usual, we declare Display Filter macro with some parameters using \$1, \$2, ...
For example, the Name is “ver”
The text is “ip.version==\$1”
type \${ver:4} in Display filter
- We can also make a Display filter macro without any parameters, define long display filter strings for the use of some alias

Display Filter Macros	
Name	Text
ver	ip.version==\$1

Create each Display filter macro

Display Filter Macros

Name	Text
ver	ip.version==\$1
office365	ip.addr==13.107.6.152/31 or ip.addr==13.107.18.10/31 or ip.addr==13.107.128.0/22 or ip.addr==23.107.248.0/24
google	ip.addr==108.170.192.0/18 or ip.addr==108.177.0.0/17 or ip.addr==130.211.0.0/16 or ip.addr==136.114.0.0/16
twitter	ip.addr==69.195.160.0/24 or ip.addr==69.195.162.0/24 or ip.addr==69.195.163.0/24 or ip.addr==69.195.164.0/24
facebook	ip.addr==31.13.24.0/21 or ip.addr==31.13.64.0/19 or ip.addr==31.13.64.0/24 or ip.addr==31.13.69.0/24



<C:\Users\TakeshitaMegumi\AppData\Roaming\Wireshark\dfilter macros>

OK

からコピー

キャンセル

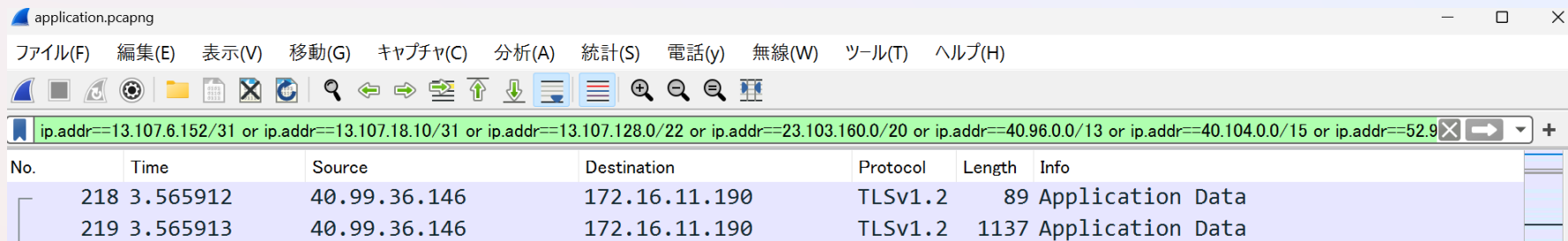
ヘルプ

- Create new Display filter macros without parameters
- Set IP address range for each application/service
- We set office365, google, facebook, and twitter

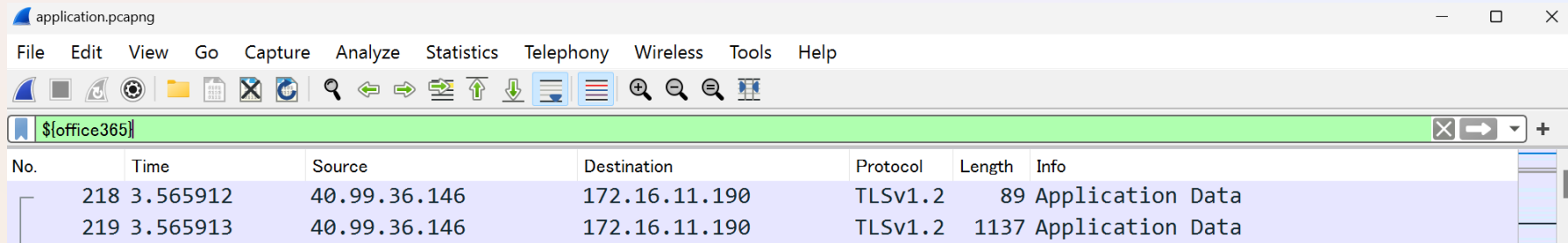
Use display filter macro in Wireshark



- Back to Wireshark, we use a long display filter to pick up office365 ip address range, like below



- Just type `${office365}` in the display filter, it works!!



Create application bandwidth graph

You didn't specify a field name. Click to select a portion of the graph.

Enabled	Graph Name	Display Filter	Color	Style	Y Axis	Y Field	SMA Period	Y Axis Factor
☒	office365	\${office365}		Bar	Bits		None	1
☒	google	\${google}		Bar	Bits		None	1
☒	twitter	\${office365}		Bar	Bits		None	1
☒	facebook	\${facebook}		Bar	Bits		None	1

+

-





Mouse ☐ drags ☒ zooms

Interval

1 sec

☐ Time of day

☐ Log scale

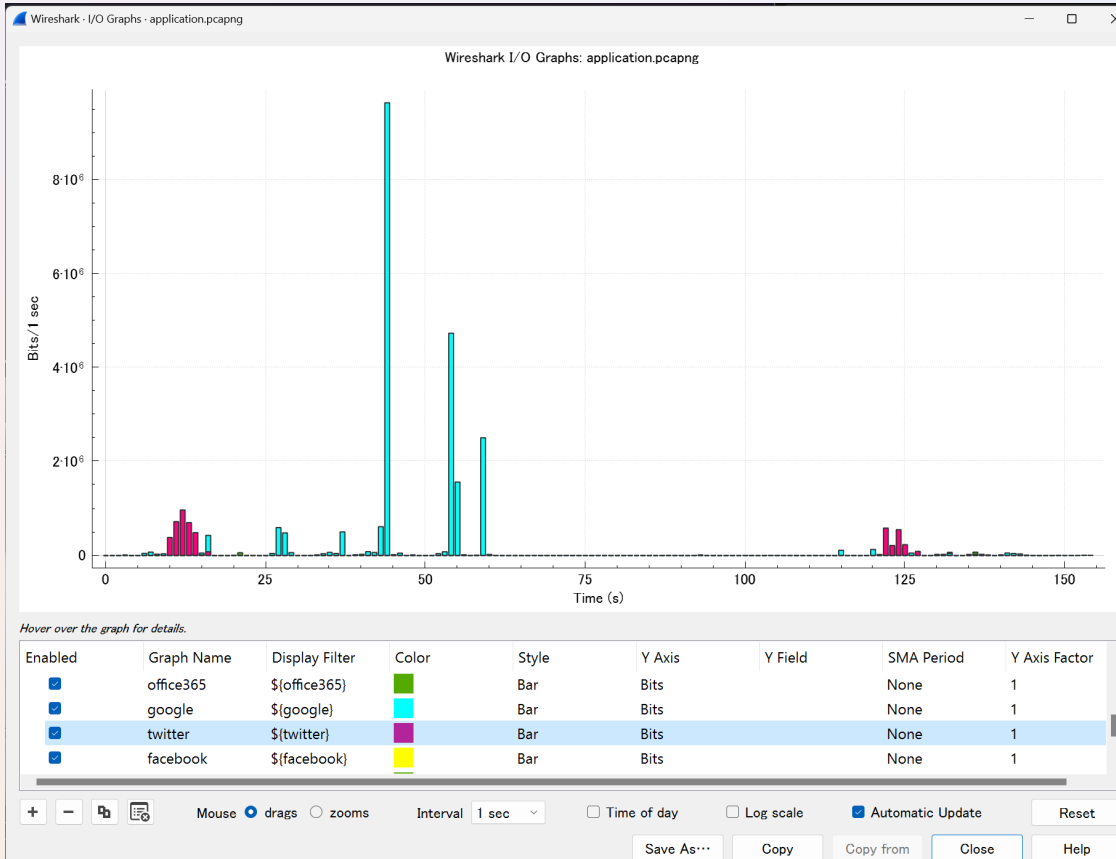
☒ Automatic Update

Reset

- Statistics>I/O Graph and check off all enabled
- Create items office365, google, twitter, facebook these filter is \${office365}, \${google}, \${twitter}, \${facebook} and sets Style as Bar, Y axis as Bit and set interval as 1 sec. Then enable these items

Visualize application by colour

- We can Visualize each application traffic by colour
- Google is the most used in this trace, and the max bandwidth is about 10Mbps

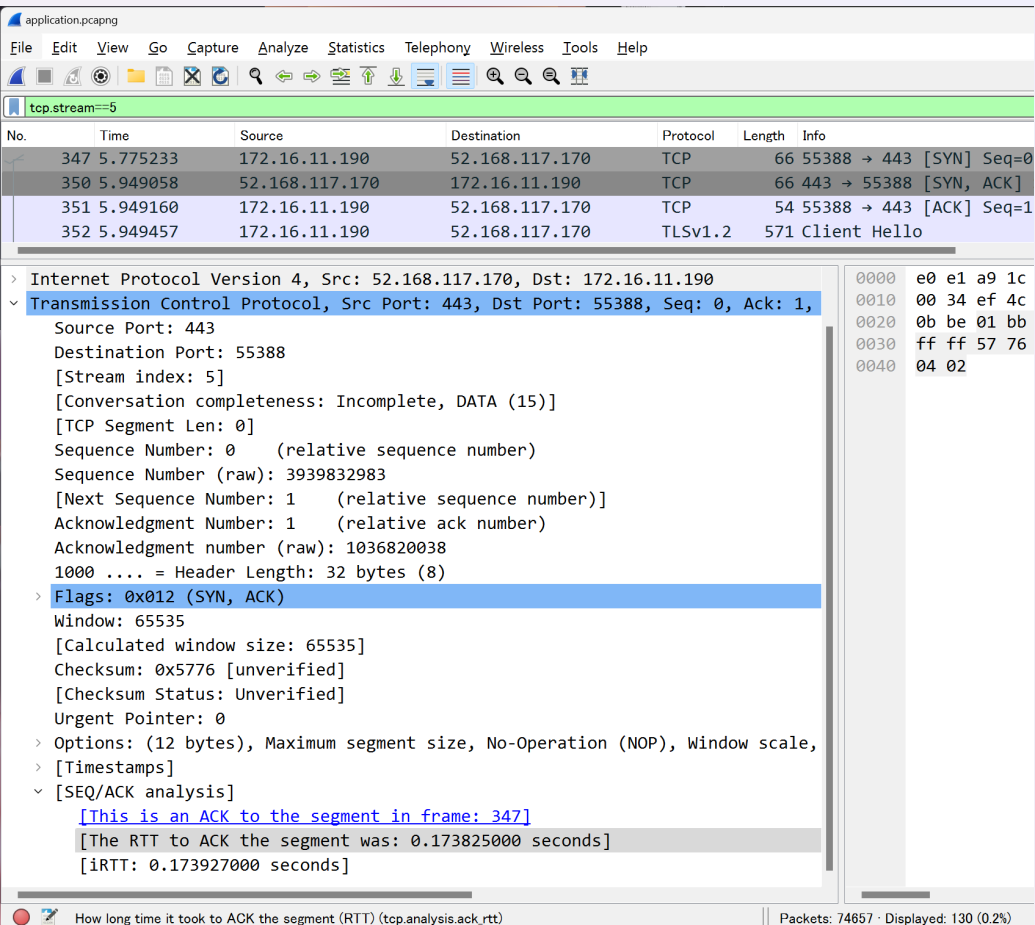


Response time of each application



- How about the response time of each service?
It is impossible to decrypt each TLS connection, (except you have a proxy server or server's key)
- So we try to use lower layer header information, TCP fields of each connection instead of upper layer.
- Let's check TCP connection, back to Wireshark, `tcp.stream==5` to filter some TCP connections, then look for a syn-ack packet of #350 frame.

How long time it took to ACK the segment



application.pcapng

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

tcp.stream==5

No.	Time	Source	Destination	Protocol	Length	Info
347	5.775233	172.16.11.190	52.168.117.170	TCP	66	55388 → 443 [SYN] Seq=0
350	5.949058	52.168.117.170	172.16.11.190	TCP	66	443 → 55388 [SYN, ACK]
351	5.949160	172.16.11.190	52.168.117.170	TCP	54	55388 → 443 [ACK] Seq=1
352	5.949457	172.16.11.190	52.168.117.170	TLSv1.2	571	Client Hello

Internet Protocol Version 4, Src: 52.168.117.170, Dst: 172.16.11.190

Transmission Control Protocol, Src Port: 443, Dst Port: 55388, Seq: 0, Ack: 1,

Source Port: 443
Destination Port: 55388
[Stream index: 5]
[Conversation completeness: Incomplete, DATA (15)]
[TCP Segment Len: 0]
Sequence Number: 0 (relative sequence number)
Sequence Number (raw): 3939832983
[Next Sequence Number: 1 (relative sequence number)]
Acknowledgment Number: 1 (relative ack number)
Acknowledgment number (raw): 1036820038
1000 = Header Length: 32 bytes (8)

Flags: 0x012 (SYN, ACK)
Window: 65535
[Calculated window size: 65535]
Checksum: 0x5776 [unverified]
[Checksum Status: Unverified]
Urgent Pointer: 0

Options: (12 bytes), Maximum segment size, No-Operation (NOP), Window scale,
[Timestamps]

[SEQ/ACK analysis]
[This is an ACK to the segment in frame: 347]
[The RTT to ACK the segment was: 0.173825000 seconds]
[iRTT: 0.173927000 seconds]

0000 e0 e1 a9 1c
0010 00 34 ef 4c
0020 0b be 01 bb
0030 ff ff 57 76
0040 04 02

How long time it took to ACK the segment (RTT) (tcp.analysis.ack_rtt)

Packets: 74657 · Displayed: 130 (0.2%)

- Wireshark collects each TCP round trip time to measure how long time it took to ACK the segment `tcp.analysis.ack_rtt`
- We can use an average time of 1 second to Visualize the application response.

The alternative of application response time

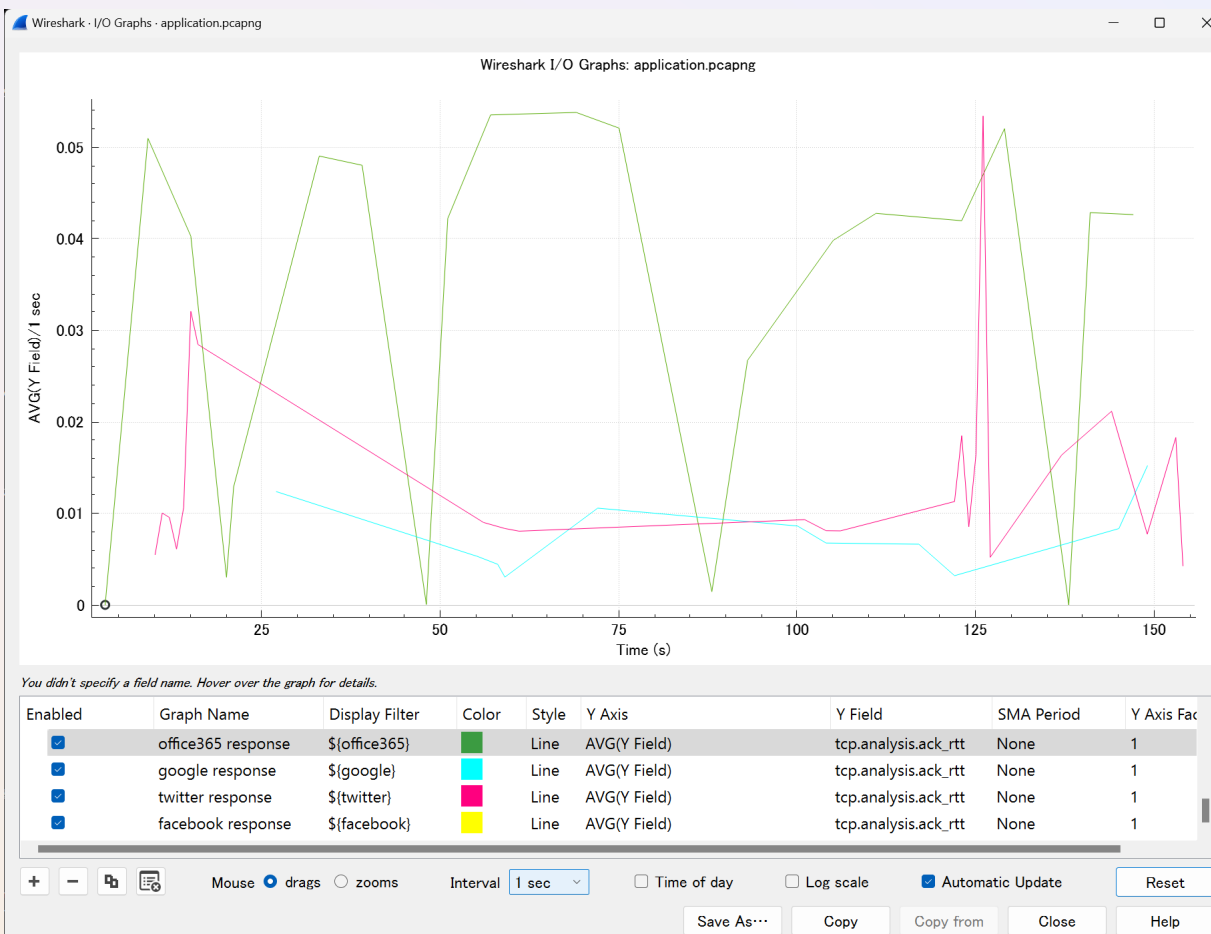
Enabled	Graph Name	Display Filter	Color	Style	Y Axis	Y Field	SMA Period	Y Axis Fac
☒	office365 response	\${office365}	Green	Line	AVG(Y Field)	tcp.analysis.ack_rtt	None	1
☐	google response	\${google}	Cyan	Line	Packets	tcp.analysis.ack_rtt	None	1
☐	twitter response	\${twitter}	Magenta	Line	Bytes	tcp.analysis.ack_rtt	None	1
☐	facebook response	\${facebook}	Yellow	Line	Bits	tcp.analysis.ack_rtt	None	1

Y Axis dropdown menu options: AVG(Y Field), COUNT FRAMES(Y Field), COUNT FIELDS(Y Field), MAX(Y Field), MIN(Y Field), SUM(Y Field).

Buttons: +, -, Zoom In, Zoom Out, Mouse (drags, zooms), Interval: 1 sec, Automatic Update (checked), Reset, Copy, Copy from, Close, Help.

- Statistics>I/O Graph and check off all enabled
- Create items office365 response, google response, twitter response, facebook response these filter is \${office365}, \${google}, \${twitter}, \${facebook} and sets Style as Line, Y axis as AVG(Y Field) and set Y Field as tcp.analysis.ack_rtt

Check App's Response by TCP RTT

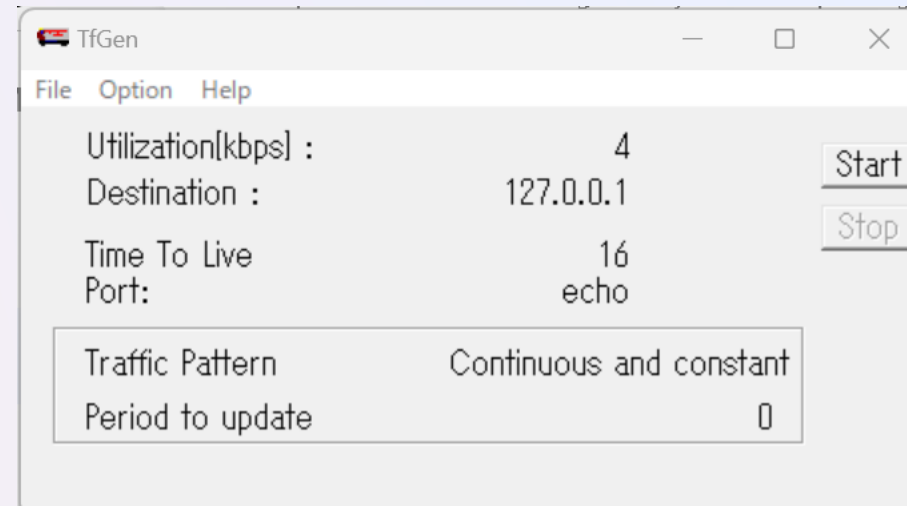


- Check each service response
- 1st Google
- 2nd Twitter
- 3rd Office365

Characteristics of UDP Application



- Open another trace file, application2.pcapng
- This trace contains TfGen, traffic generator packets, UDP echo (port 7) protocol as well as HTTP traffic.
- Change configuration profiles from default to UDP application
- Check the characteristics of UDP application.



UDP Application bandwidth graph

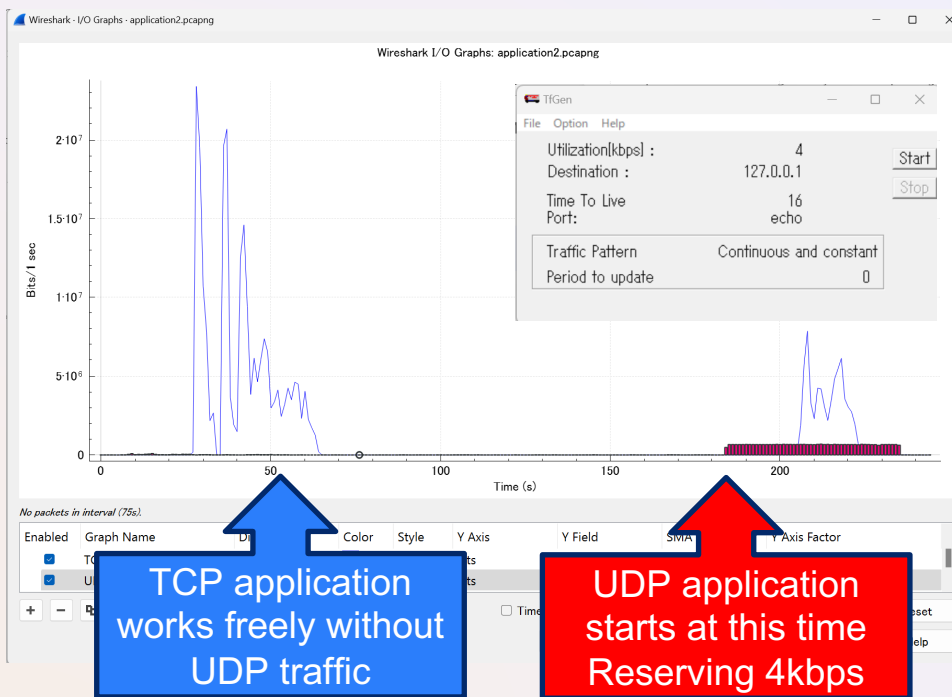


- Select Statistics>I/O graph and add an item as TCP application that display filter is “tcp”, set Style as Line, set Y Axis as Bits and add an item as UDP application that display filter is “udp”, set Style as Bar, Y Axis as Bits, then check enabled

Enabled	Graph Name	Display Filter	Color	Style	Y Axis	Y Field	SMA Period	Y Axis Factor
☒	TCP application	tcp		Line	Bits		None	1
☒	UDP application	udp		Bar	Bits		None	1

- This easy graph shows us the characteristics between TCP and UDP

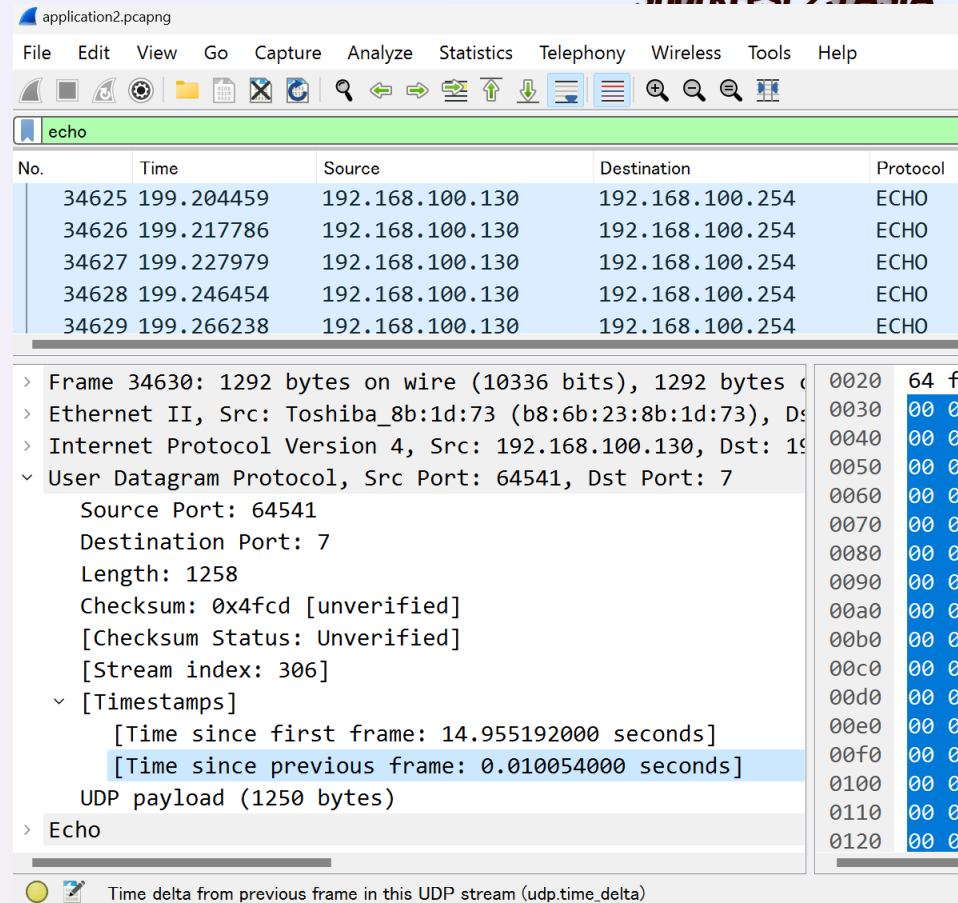
UDP uses fixed bandwidth prior to TCP at first then TCP bandwidth changes dynamically



- TCP application (http) works freely without UDP traffic at first (change output segment size, receive window size by round trip time)
- UDP application (echo) reserve fixed bandwidth 4kbps prior to TCP app.

Check UDP application (echo) frame rate

- Set the display filter as echo and check each packet.
- Check UDP application frame rate by Time delta from previous frame in this UDP stream (udp.time_delta)
- Also compare TCP RTT with UDP frame timing



application2.pcapng

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

echo

No.	Time	Source	Destination	Protocol
34625	199.204459	192.168.100.130	192.168.100.254	ECHO
34626	199.217786	192.168.100.130	192.168.100.254	ECHO
34627	199.227979	192.168.100.130	192.168.100.254	ECHO
34628	199.246454	192.168.100.130	192.168.100.254	ECHO
34629	199.266238	192.168.100.130	192.168.100.254	ECHO

Frame 34630: 1292 bytes on wire (10336 bits), 1292 bytes captured (10336 bits) on interface 0

Ethernet II, Src: Toshiba_8b:1d:73 (b8:6b:23:8b:1d:73), Dst: 08:00:27:00:00:00

Internet Protocol Version 4, Src: 192.168.100.130, Dst: 192.168.100.254

User Datagram Protocol, Src Port: 64541, Dst Port: 7

Source Port: 64541

Destination Port: 7

Length: 1258

Checksum: 0x4fcd [unverified]

[Checksum Status: Unverified]

[Stream index: 306]

[Timestamps]

[Time since first frame: 14.955192000 seconds]

[Time since previous frame: 0.010054000 seconds]

UDP payload (1250 bytes)

Echo

Time delta from previous frame in this UDP stream (udp.time_delta)

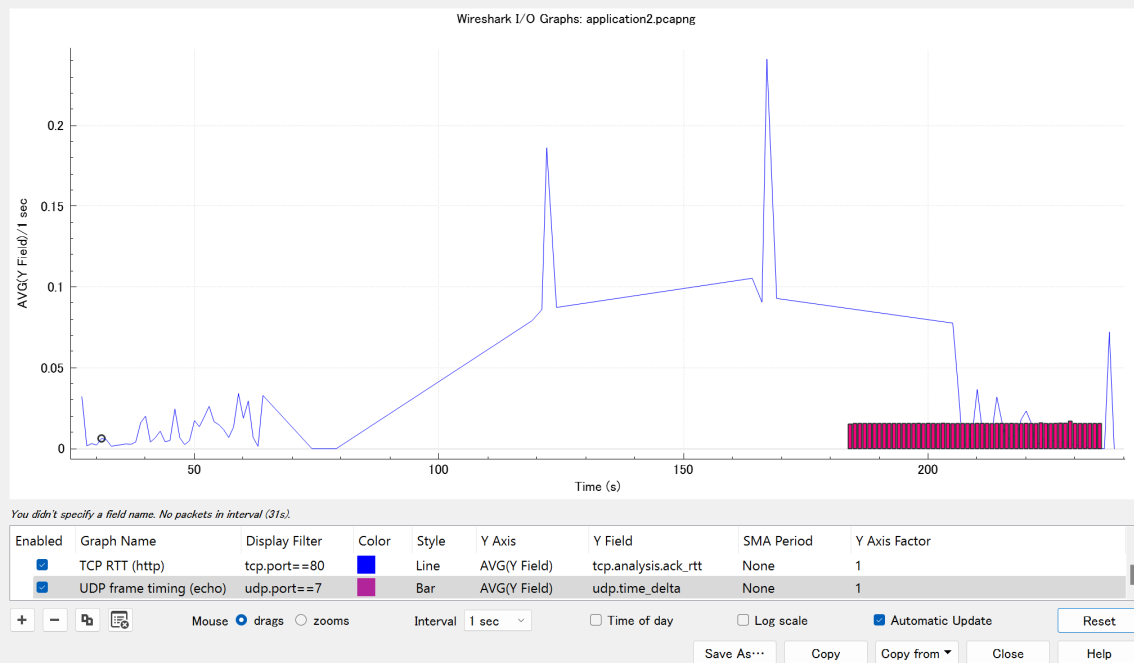
TCP RTT vs UDP frame timing

Enabled	Graph Name	Display Filter	Color	Style	Y Axis	Y Field	SMA Period	Y Axis Factor
☒	TCP RTT (http)	tcp.port==80	Blue	Line	AVG(Y Field)	tcp.analysis.ack_rtt	None	1
☒	UDP frame timing (echo)	udp.port==7	Purple	Bar	AVG(Y Field)	udp.time_delta	None	1

- Select Statistics>I/O graph and add an item as TCP RTT (http), set Display Filter as “tcp.port==80”, set Style as Line, set Y Axis as AVG(Y Field), set Y Field as “tcp.analysis.ack_rtt” and add an item as UDP frame timing (echo), set Display Filter as “udp.port==7”, set Style as Bar, set Y Axis as AVG(Y Field), set Y Field as “udp.time_delta”
- Enabled both items and reset the Graph

UDP sends packets on constant

Wireshark - I/O Graphs - application2.pcapng



- UDP application is prior and faster than TCP

- Blue line (TCP RTT) changes dynamically by the traffic status
- Red Bars (UDP frame timing) are almost the same height and lower than the blue line

Compare each UDP applications



- In this case, we compare Youtube with Zoom, both services are based on UDP protocols (another trace file: application3.pcapng)
- Youtube uses Google's QUIC, and Zoom uses a customised RTP protocol.
- UDP has no retransmission, no flow control, no mechanism of control bandwidth (application protocol does everything)
So first UDP application takes the bandwidth, the later TCP application suffers.

IPv4/IPv6 address range of Zoom

- Zoom network address blocks are below <https://support.zoom.us/hc/en-us/articles/201362683-Zoom-network-firewall-or-proxy-server-settings>



Zoom network firewall or proxy server settings

Last Updated: March 31, 2023

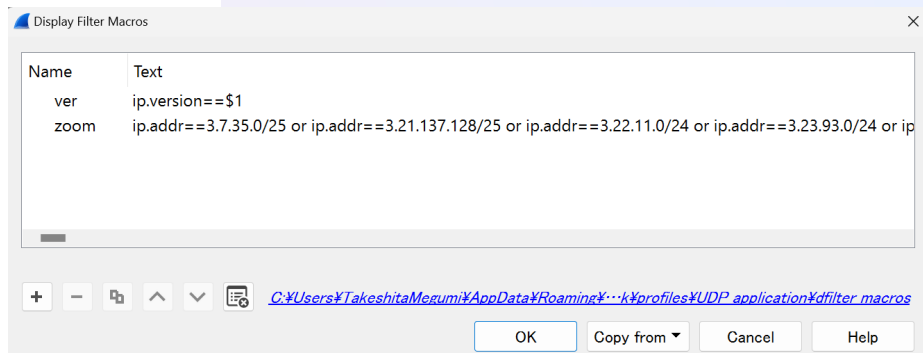
If your app stays in a "connecting" mode or has timed out due to **Network error**, please **try again** or **Can't connect to our service**, please **check your network connection and try again** issues, it could be related to your network connection, network firewall settings, or web security gateway settings.

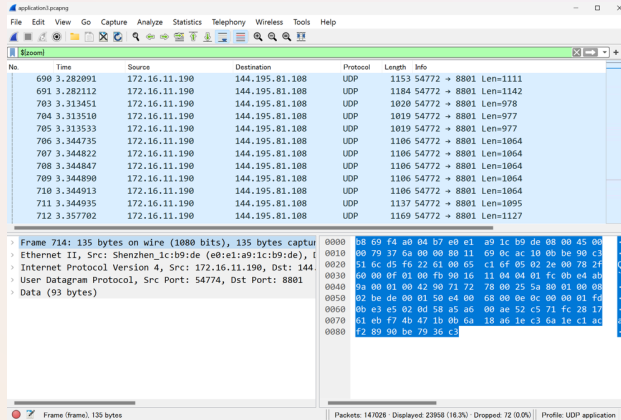
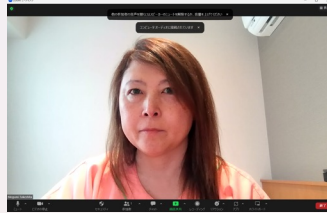
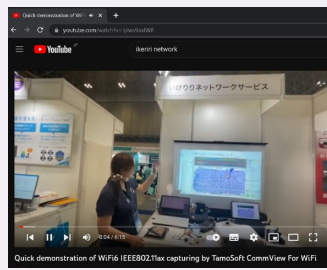
Note: Check your network connection by opening a browser and ensure that you can access <https://zoom.us>

This article covers:

- [Zoom firewall rules](#)
 - [Firewall rules for Zoom Phone](#)
 - [Firewall rules for Zoom Contact Center](#)
 - [Firewall rules for Zoom website](#)
 - [Firewall rules for certificate validation](#)
 - [Zoom CDN](#)
 - [Zoom Device Management \(ZDM\)](#)
 - [IP ranges txt files](#)
- [Proxy server](#)
- [Firewall rules for other Zoom services](#)

- Create Display Filter Macros of Zoom ip range





Let's Test the Application bandwidth and response

- First, Open Chrome and Play Youtube video 4k Full HD streaming using Google QUIC
- Second, Start video conference using Zoom
Zoom uses its customized RTP/RTCP.
- Capture and save trace file as application3.pcapng
- Open the file and create a bandwidth graph and UDP frame timing graph between Google and Zoom

Check Google QUIC traffic

- Statistics>Conversation and choose UDP with Name the resolution, sort traffic by the number of packets
- Select the 1st UDP conversation of 59.648MBytes
Address A Port A is 172.16.11.190:58274 (Client Chrome)
Address B Port B is rr3.sn-npoe7nek.googlevideo.com:443 (UDP port 443 means Google QUIC protocol by Youtube)
this traffic starts soon after starting the capture, duration is 51.7742, so this conversation (udp.stream==65) is Youtube

Wireshark · Conversations · application3.pcapng

Conversation Settings

☒ Name resolution

☐ Absolute start time

☐ Limit to display filter

Copy

Follow Stream...

Ethernet · 117

IPv4 · 127

IPv6 · 36

TCP · 99

UDP · 522

Address A	Port A	Address B	Port B	Packets	Bytes	Stream ID	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A →
172.16.11.190	58274	rr3.sn-npoe7nek.googlevideo.com	443	53,964	59.648 MiB	65	5,823	610.170 KiB	48,141	59.052 MiB	2.418833	51.7742	94.281 K
172.16.11.190	49906	144.195.87.230	8801	7,072	7.198 MiB	56	3,299	3.385 MiB	3,773	3.813 MiB	2.190351	50.5892	548.135 K
172.16.11.190	49909	144.195.87.230	8801	3,386	736.145 KiB	0	2,022	419.454 KiB	1,364	316.690 KiB	0.000000	52.7798	63.577 K
172.16.11.190	51401	nrt12s18-in-f6.1e100.net	443	1,473	1.650 MiB	22	136	11.237 KiB	1,337	1.639 MiB	0.679992	0.0809	1.085 M
172.16.11.190	61246	rr2.sn-oguesn6s.googlevideo.com	443	1,184	1.250 MiB	51	163	19.785 KiB	1,021	1.231 MiB	1.735490	21.7509	7.276 K
172.16.11.190	57662	rr2.sn-oguesn6s.googlevideo.com	443	465	481.091 KiB	520	85	9.151 KiB	380	471.939 KiB	54.093252	0.0971	753.765 K
172.16.11.190	58055	nrt12s35-in-f14.1e100.net	443	322	171.742 KiB	21	114	32.326 KiB	208	139.416 KiB	0.673120	41.1407	6.285 K
172.16.11.190	5353	224.0.0.251	5353	228	18.035 KiB	17	228	18.035 KiB	0	0.000000	0.000000	43.8188	2.202 K

Check Zoom video traffic

application3.pcapng

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

{zoom}

No.	Time	Source	Destination	Protocol	Length	Info
71035	50.936188	172.16.11.190	144.195.87.230	UDP	123	49906 → 8801 Len=81
71404	51.055959	144.195.87.230	172.16.11.190	UDP	133	8801 → 49906 Len=91
71405	51.056131	172.16.11.190	144.195.87.230	WireGu...	151	Transport Data, receiver
71406	51.064682	144.195.87.230	172.16.11.190	WireGu...	123	Transport Data, receiver

- Look for Zoom traffic, set Display filter as `{zoom}`
we can find UDP packets between Zoom Client
(172.16.11.190:49909) and Server(144.195.87.230:8801)
- Statistics>Conversation and choose UDP with Name resolution
- We can find 2nd UDP traffic of 7072 packets, 7.198MBytes,
this UDP stream (udp.stream==56) is Zoom video traffic.

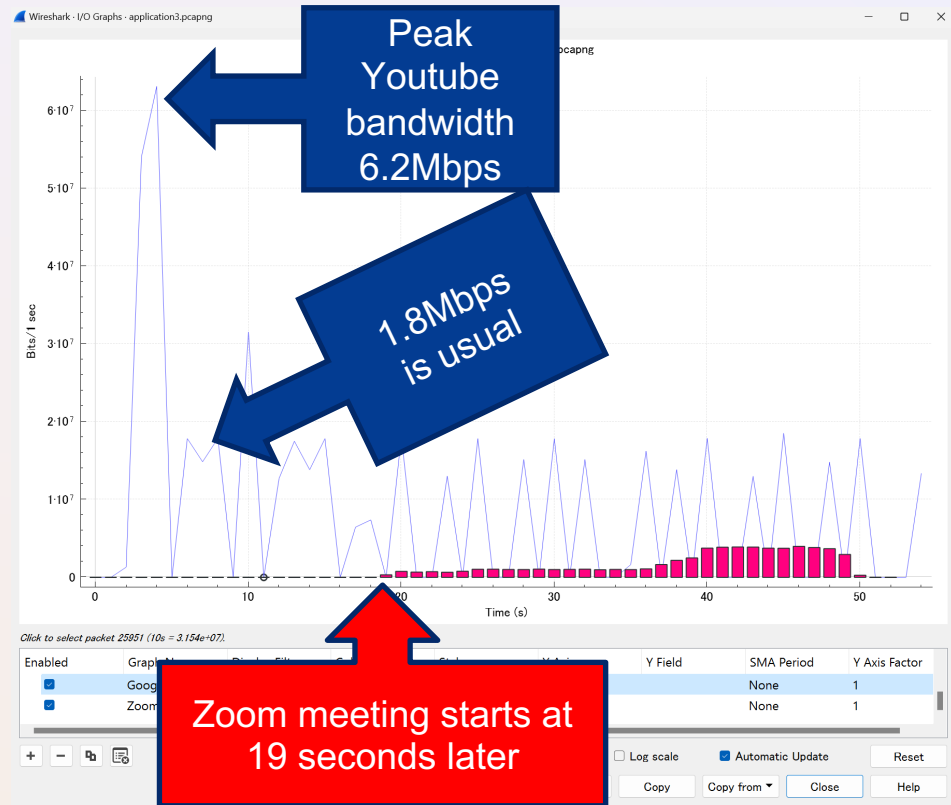
Bandwidth graph between Youtube and Zoom



Enabled	Graph Name	Display Filter	Color	Style	Y Axis	Y Field	SMA Period	Y Axis Factor
☒	Google bandwi...	udp.stream==65	Blue	Line	Bits		None	1
☒	Zoom bandwidth	udp.stream==56	Pink	Bar	Bits		None	1

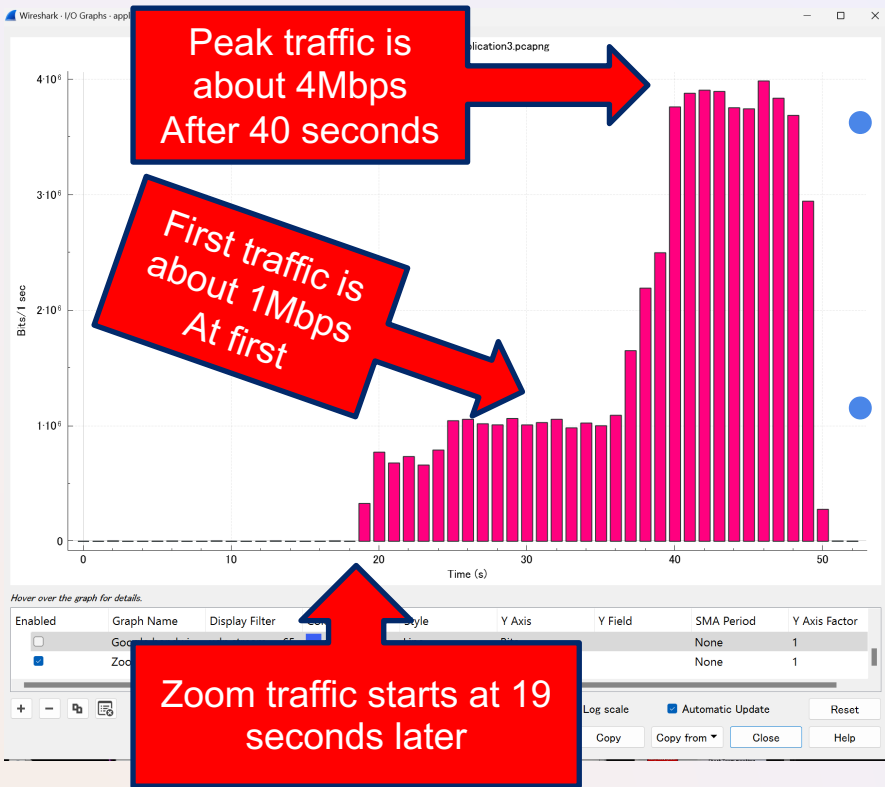
- Select Statistics>I/O graph and add an item as Google bandwidth, set Display Filter as `udp.stream==65`, set Style as Line, set Y Axis as Bits, and add an item as Zoom bandwidth, set Display Filter as `udp.stream==56`, set Style as Bar, set Y Axis as Bits, then check both items
- Make a bandwidth graph between Youtube and Zoom

Bandwidth usage between Google and Zoom



- This trace includes 4k 30fps Youtube video traffic at first
- Peak Google QUIC bandwidth is 62Mbps and 18Mbps usual
- Start Zoom meeting after 19 seconds later

Check off Google bandwidth and focus on Zoom



- Check off Google bandwidth item and focus on Zoom bandwidth
- Actual Zoom meeting traffic starts at 19 seconds later, and actual traffic is about 1Mbps at first, then speed up at 4Mbps after 40 seconds

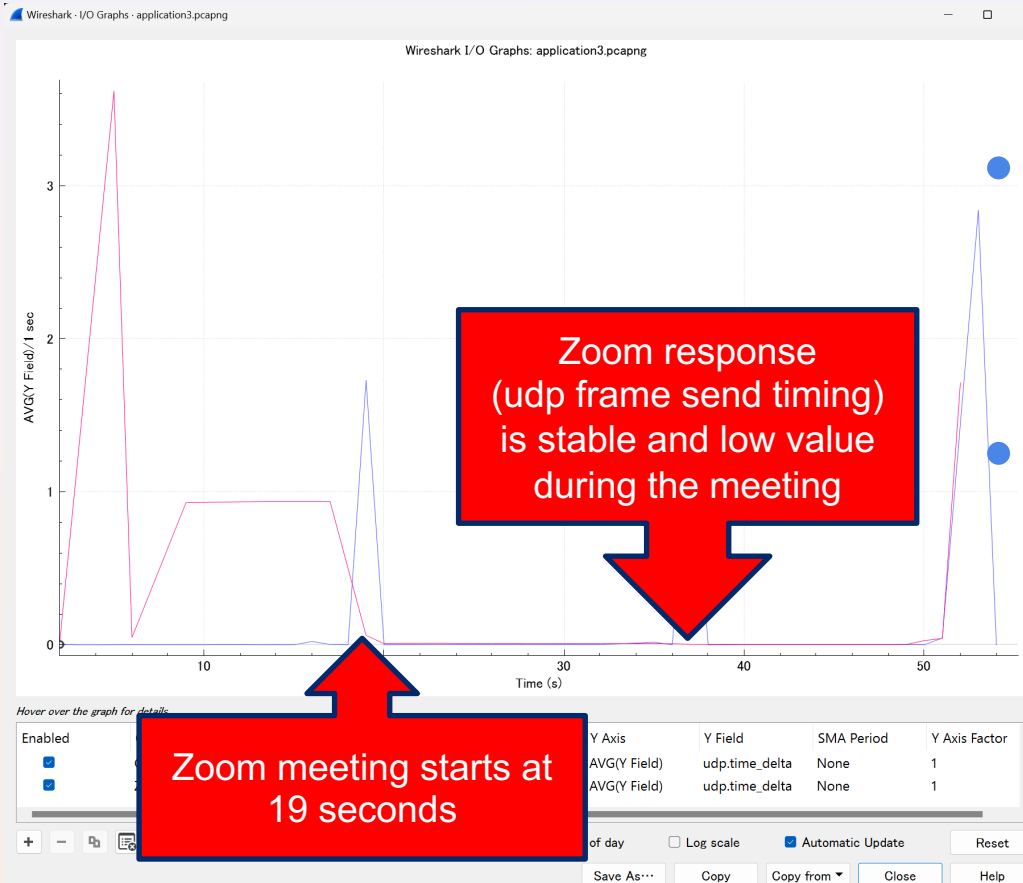
Create application response time



- Select Statistics>I/O graph and add an item as Google response, set Display Filter as `udp.stream==65`, set Style as Line, set Y Axis as AVG(Y Fields), set Y Field as `udp.time_delta` and add an item as Zoom response, set Display Filter as `udp.stream==56`, set Style as Line, set Y Axis as AVG(Y Fields), set Y Field as `udp.time_delta` , then check both items

Enabled	Graph Name	Display Filter	Color	Style	Y Axis	Y Field	SMA Period	Y Axis Factor
☐	Google response	<code>udp.stream==65</code>		Line	AVG(Y Field)	<code>udp.time_delta</code>	None	1
☐	Zoom response	<code>udp.stream==56</code>		Bar	AVG(Y Field)	<code>udp.time_delta</code>	None	1

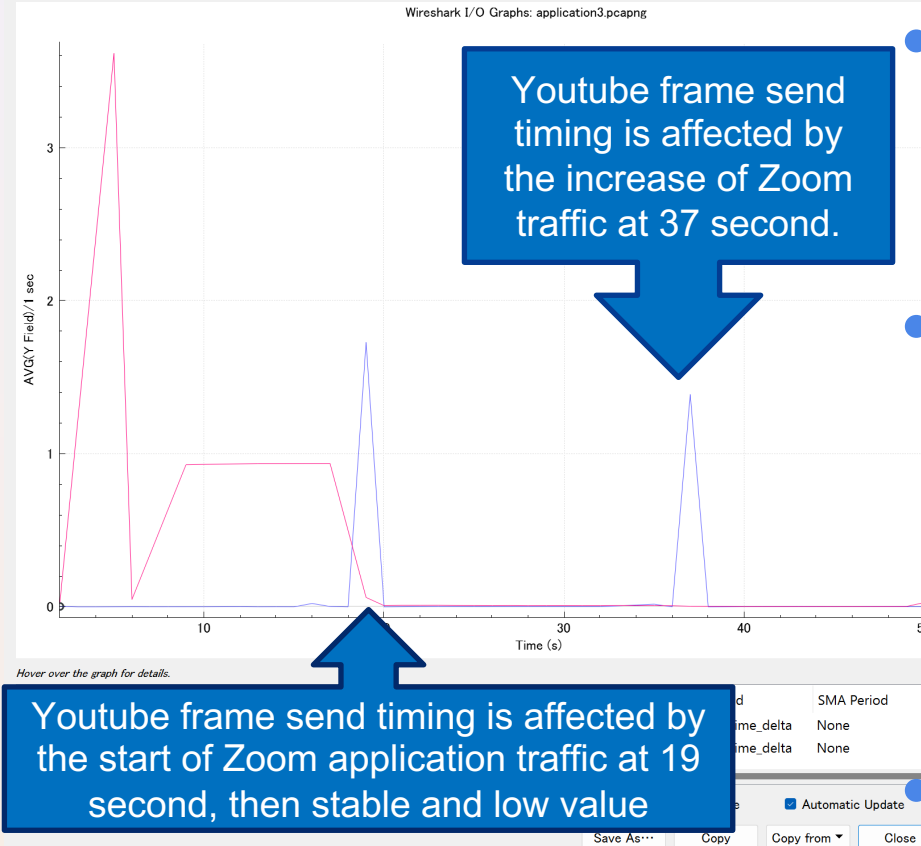
UDP frame send timing of Zoom application



Check the red line,
it means Zoom response
(udp frame send timing)
Zoom meeting starts at
19 seconds, then Zoom
response is low until 50
seconds at the end

UDP frame send timing of Youtube Video

Wireshark · I/O Graphs · application3.pcapng

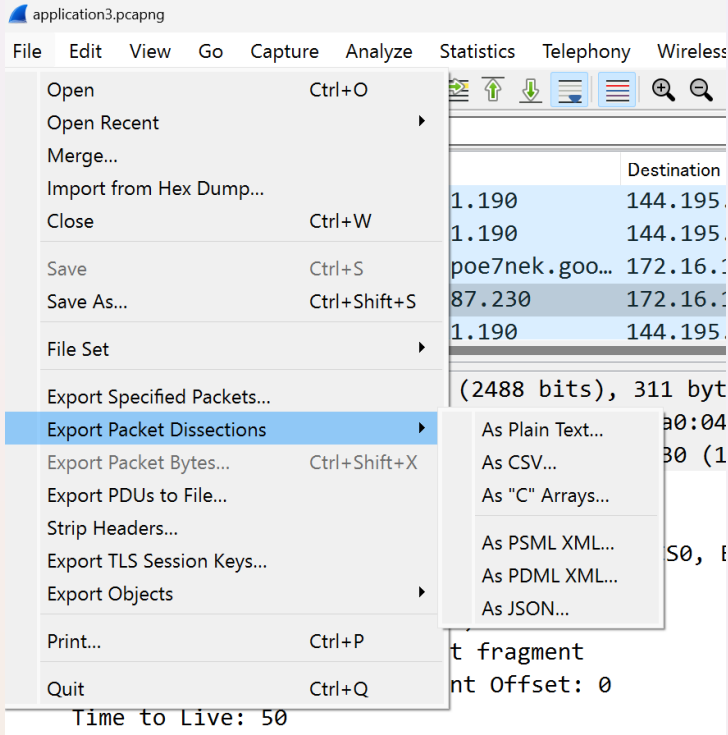


- Check the blue line, it means Youtube response (udp frame send timing)
- Youtube frame send timing is affected by the start of Zoom application traffic at 19 sec and the increase of Zoom traffic at 37 sec.
- But soon go back to normal

We can extend Wireshark Visualization by exporting the trace as JSON, CSV

SharkFest'23 ASIA
Singapore · Apr 17-19

#sf23asia

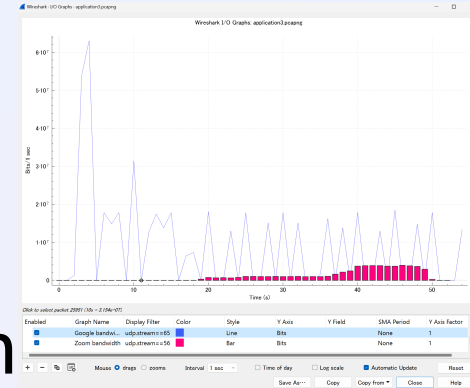


- File>Export Packet Dissections As CSV... or As JSON...
- We can use Excel to create Pie charts, Splunk or Erastic Search Kibana Grafana to create Data Analysis (or ChatGPT?)
- tshark is useful to create output data (-T and -e options)

Wireshark I/O Graph is also one of good tools for visualizing application traffic

SharkFest 23 ASIA
Singapore · Apr 17-19

- Excel, Splunk and other data analysis tools are good options for Visualization, but Wireshark I/O Graph is easy to use, no need for a complicated configuration
- We can try and get Visualization results within the Wireshark I/O graph easily.
- Use Wireshark I/O graph and Visualize, troubleshoot your application problems.



USE WIRESHARK

Thank you for watching.

Please complete the Google form survey

trace files and Wireshark profiles are here:

[https://www.ikeriri.ne.jp/sharkfest/
VisualizeApplicationTraffic.zip](https://www.ikeriri.ne.jp/sharkfest/VisualizeApplicationTraffic.zip)



ikeriri network service

<http://www.ikeriri.ne.jp>