Wireshark visualization TIPS & tricks TOP10

Supplemental files
http://www.ikeriri.ne.jp/sharkfest/
and official site later

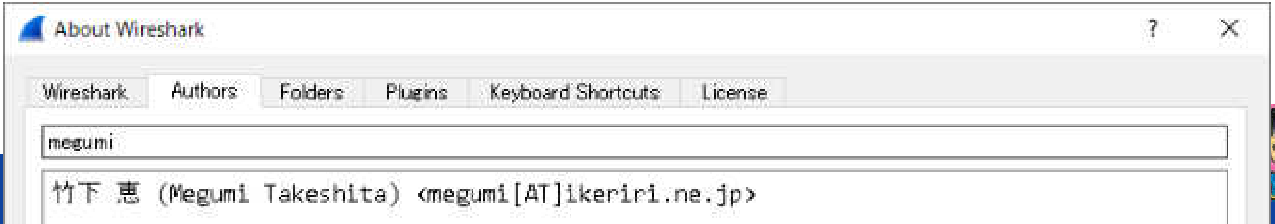
Megumi Takeshita

Packet Otaku, ikeriri network service

Megumi Takeshita, ikeriri network service

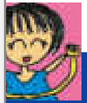
- Former CACE technologies reseller in 2008
- Founder, ikeriri network service co., ltd
- Wrote 10+ books about Wireshark
- Reseller of Riverbed Technology and other capture hardware/software in Japan
- Attending all Sharkfest
- One of contributor of Wireshark

  Translate Wireshark into Japanese



megumi

竹下 恵 (Megumi Takeshita) <megumi[AT]ikeriri.ne.jp>



Visualization TIPS and TRICKS TOP10

#1 Flow Graph
#2 New Map
#3 TCP Stream Graph
#4 RTP Graph
#5 IO Graph
#6 Copy table values as CSV

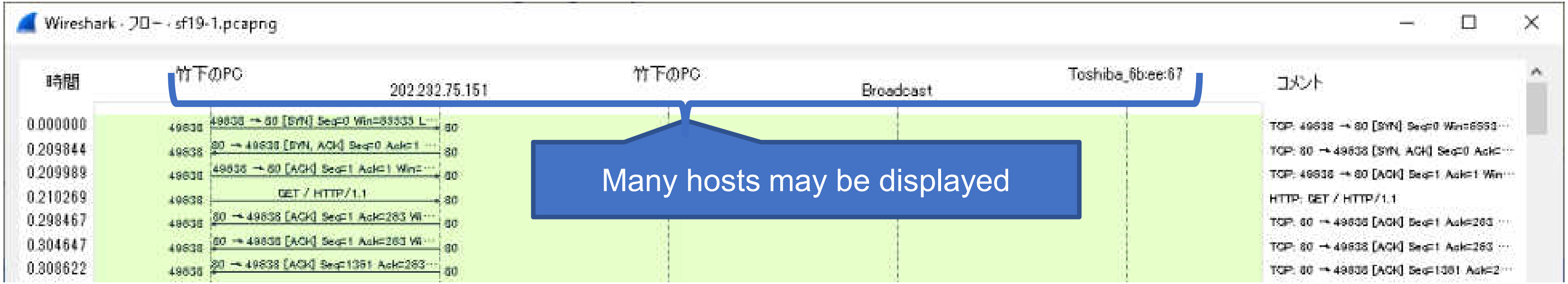
Part1
Wireshark

#7 Create statistics using tshark
#8 Collect fields for Visualization
#9 Export Packet dissection to JSON
#10 Splunk

Part2
tshark

#1 Flow Graph with Conversation Filter

- If you want to grab sequence, retransmission, and fragmentation between hosts, Flow Graph is a good idea to visualize packets.
- Open trace file “sf19-1.pcapng” and choose Statistics > Flow Graph to create Flow Graph



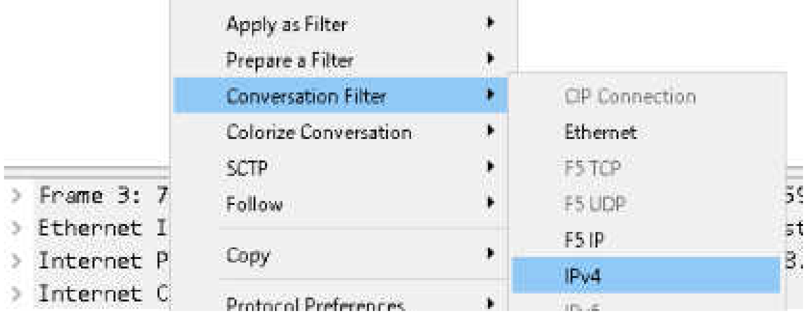
Wireshark · フロー · sf19-1.pcapng

```
時間        竹下のPC              竹下のPC        Toshiba_6b:ee:67    コメント
                 202.232.75.151        Broadcast
0.000000  49838  49838 → 80 [SYN] Seq=0 Win=83333 L…  80      TCP: 49838 → 80 [SYN] Seq=0 Win=6553…
0.209844  49838  80 → 49838 [SYN, ACK] Seq=0 Ack=1 …  80      TCP: 80 → 49838 [SYN, ACK] Seq=0 Ack=…
0.209989  49838  49838 → 80 [ACK] Seq=1 Ack=1 Win=…  80       TCP: 49838 → 80 [ACK] Seq=1 Ack=1 Win…
0.210269  49838  GET / HTTP/1.1  80                            HTTP: GET / HTTP/1.1
0.298467  49838  80 → 49838 [ACK] Seq=1 Ack=283 Wi…  80       TCP: 80 → 49838 [ACK] Seq=1 Ack=283 …
0.304647  49838  80 → 49838 [ACK] Seq=1 Ack=283 Wi…  80       TCP: 80 → 49838 [ACK] Seq=1 Ack=283 …
0.308622  49838  80 → 49838 [ACK] Seq=1381 Ack=283…  80       TCP: 80 → 49838 [ACK] Seq=1381 Ack=2…
```

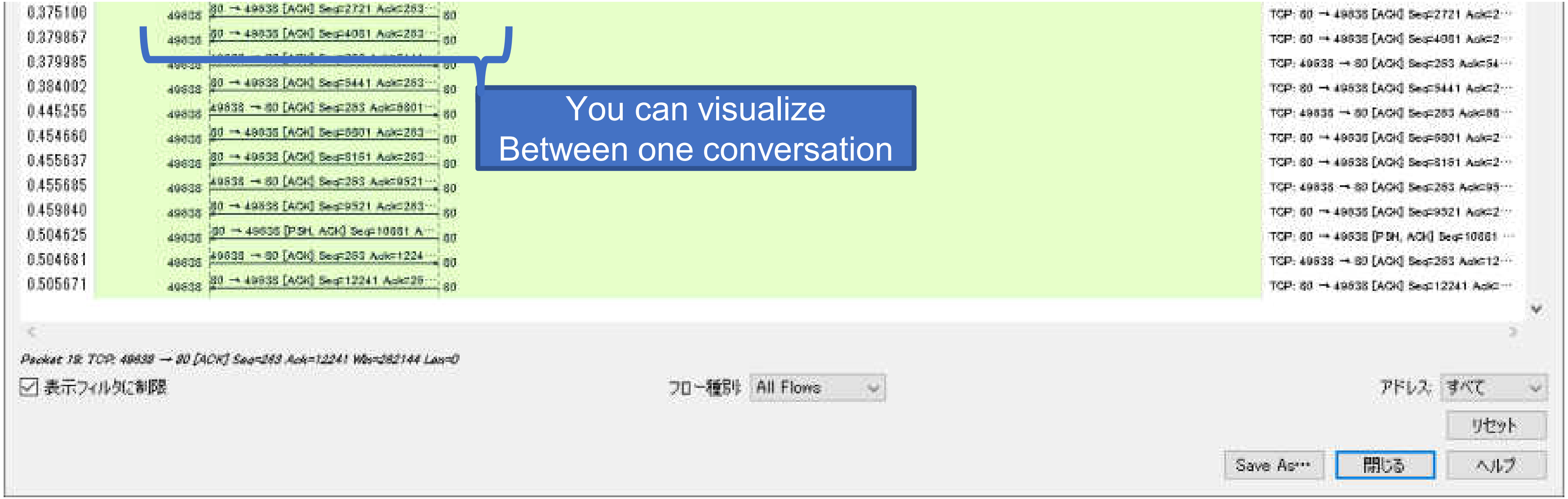
Many hosts may be displayed

#1 Flow Graph with Conversation Filter

- Wireshark shows Flow Graph of all packets, there may be tons of hosts in a Flow Graph, so use conversation filter to focus between 2 hosts you want to.
- Choose a packet that you want to visualize conversation, right click to choose “Conversation Filter” > “IPv4” to set display filter.
- Then click
  Statistics > FlowGraph



#1 Flow Graph with Conversation Filter



```
0.375100  49638  80 → 49638 [ACK] Seq=2721 Ack=263···   80   TCP: 80 → 49638 [ACK] Seq=2721 Ack=2···
0.379867  49638  80 → 49638 [ACK] Seq=4081 Ack=263···   80   TCP: 80 → 49638 [ACK] Seq=4081 Ack=2···
0.379985  49638                                         80   TCP: 49638 → 80 [ACK] Seq=263 Ack=54···
0.384002  49638  80 → 49638 [ACK] Seq=5441 Ack=263···   80   TCP: 80 → 49638 [ACK] Seq=5441 Ack=2···
0.445255  49638  49638 → 80 [ACK] Seq=263 Ack=6801···   80   TCP: 49638 → 80 [ACK] Seq=263 Ack=68···
0.454660  49638  80 → 49638 [ACK] Seq=6801 Ack=263···   80   TCP: 80 → 49638 [ACK] Seq=6801 Ack=2···
0.455637  49638  80 → 49638 [ACK] Seq=8161 Ack=263···   80   TCP: 80 → 49638 [ACK] Seq=8161 Ack=2···
0.455685  49638  49638 → 80 [ACK] Seq=263 Ack=9521···   80   TCP: 49638 → 80 [ACK] Seq=263 Ack=95···
0.459840  49638  80 → 49638 [ACK] Seq=9521 Ack=263···   80   TCP: 80 → 49638 [ACK] Seq=9521 Ack=2···
0.504625  49638  80 → 49638 [PSH, ACK] Seq=10881 A···   80   TCP: 80 → 49638 [PSH, ACK] Seq=10881 ···
0.504681  49638  49638 → 80 [ACK] Seq=263 Ack=1224···   80   TCP: 49638 → 80 [ACK] Seq=263 Ack=12···
0.505671  49638  80 → 49638 [ACK] Seq=12241 Ack=26···   80   TCP: 80 → 49638 [ACK] Seq=12241 Ack=···

Packet 19: TCP: 49638 → 80 [ACK] Seq=263 Ack=12241 Win=262144 Len=0
```

☑ 表示フィルタに制限

フロー種別: All Flows

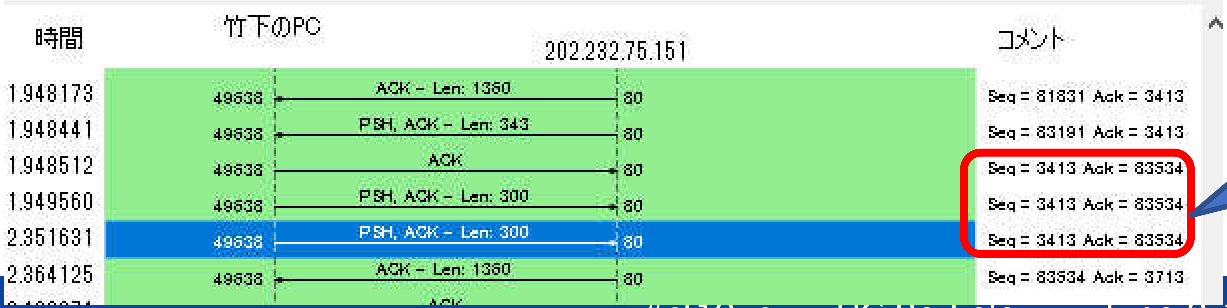
アドレス: すべて

You can visualize
Between one conversation

- Check “Limit to display filter” to limit conversation.
- You can visualize Flow Graph between 2 hosts.

#1 Flow Graph with Conversation Filter

- If you want to see flow of TCP level connection,
- Choose a packet and right click “Conversation Filter” > “TCP”, then select Statistics > Flow Graph, click “Limit to display filter” and change flow type as TCP.
- Time 2.351631 shows TCP retransmission and you can also check the same Seq / Ack numbers.

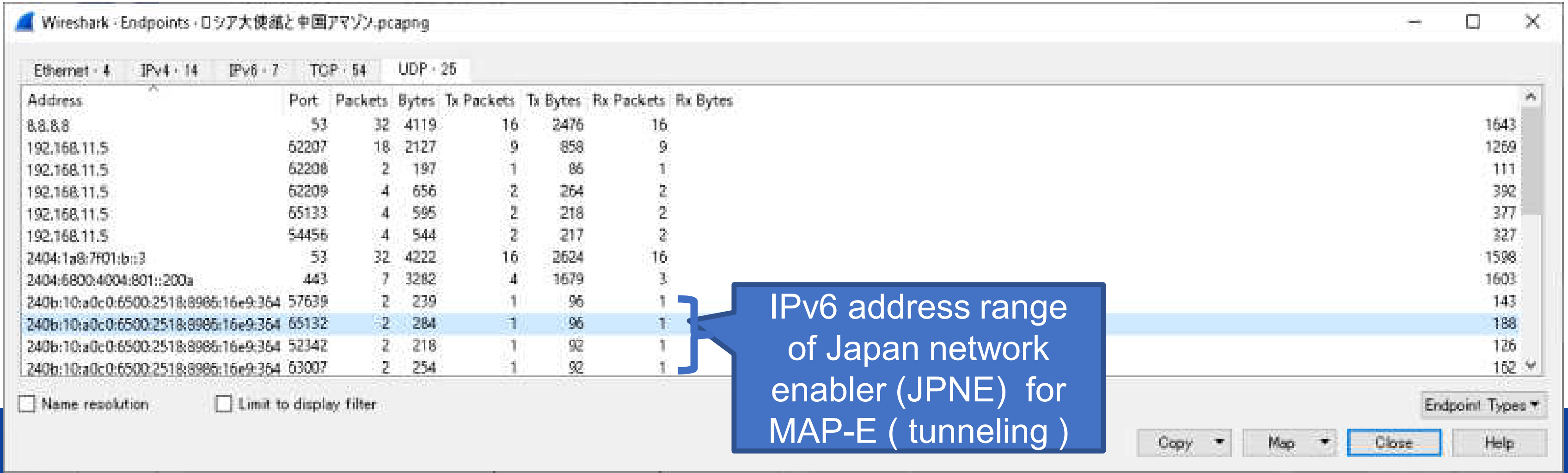


時間	竹下のPC		202.232.75.151	コメント
1.948173	49638	ACK - Len: 1360	80	Seq = 81831 Ack = 3413
1.948441	49638	PSH, ACK - Len: 343	80	Seq = 83191 Ack = 3413
1.948512	49638	ACK	80	Seq = 3413 Ack = 83534
1.949560	49638	PSH, ACK - Len: 300	80	Seq = 3413 Ack = 83534
2.351631	49638	PSH, ACK - Len: 300	80	Seq = 3413 Ack = 83534
2.364125	49638	ACK - Len: 1360	80	Seq = 83534 Ack = 3713
		ACK		

Same Seq / Ack says The sent segment is still not ACKed and receive no segment yet.

#2 New Map

- Wireshark 3.x revived Map function and we can visualize traffic by Map using Endpoints plugin.
- Open “sf19-2.pcapng” and click Statistics > Endpoints > UDP tab, then click Map > open in browser



Wireshark · Endpoints · ロシア大使館と中国アマゾン.pcapng

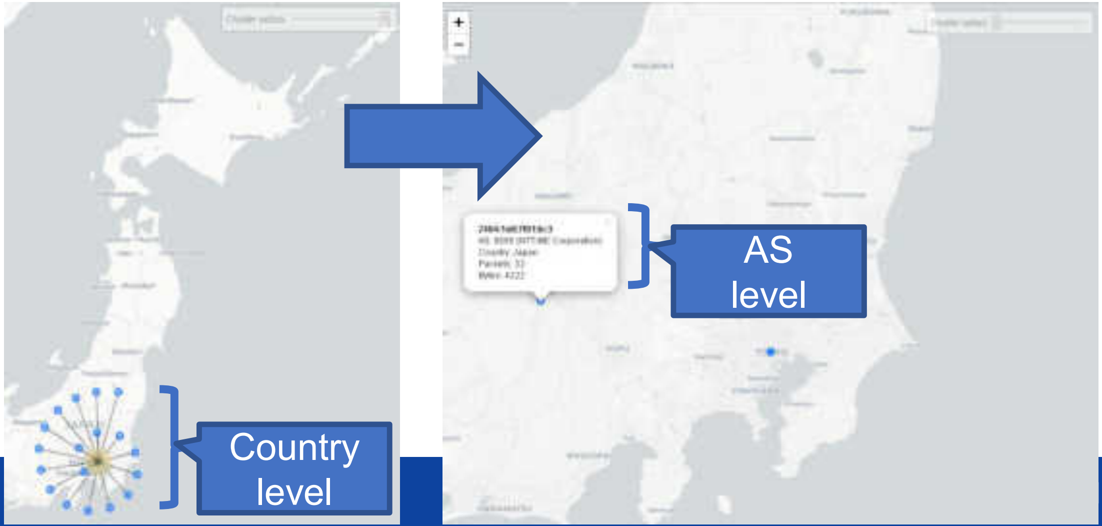
Address	Port	Packets	Bytes	Tx Packets	Tx Bytes	Rx Packets	Rx Bytes
8.8.8.8	53	32	4119	16	2476	16	1643
192.168.11.5	62207	18	2127	9	858	9	1269
192.168.11.5	62208	2	197	1	86	1	111
192.168.11.5	62209	4	656	2	264	2	392
192.168.11.5	65133	4	595	2	218	2	377
192.168.11.5	54456	4	544	2	217	2	327
2404:1a8:7f01:b::3	53	32	4222	16	2624	16	1598
2404:6800:4004:801::200a	443	7	3282	4	1679	3	1603
240b:10:a0c0:6500:2518:8986:16e9:364	57639	2	239	1	96	1	143
240b:10:a0c0:6500:2518:8986:16e9:364	65132	2	284	1	96	1	188
240b:10:a0c0:6500:2518:8986:16e9:364	52342	2	218	1	92	1	126
240b:10:a0c0:6500:2518:8986:16e9:364	63007	2	254	1	92	1	162

☐ Name resolution ☐ Limit to display filter

IPv6 address range of Japan network enabler (JPNE) for MAP-E (tunneling)

#2 New Map

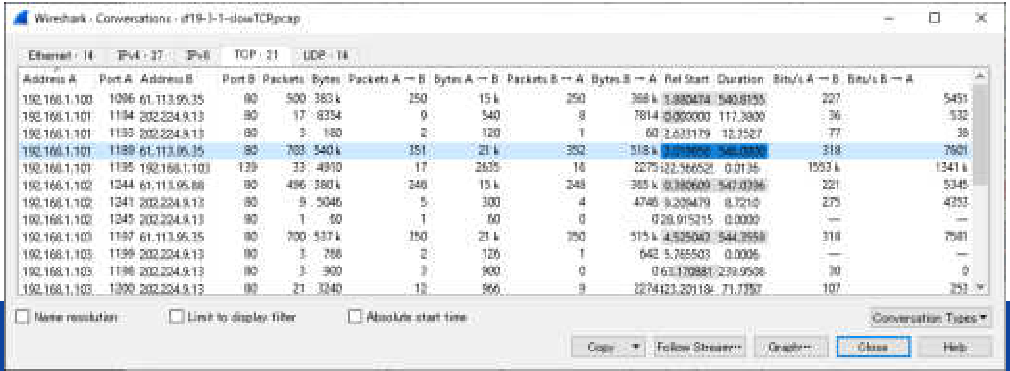
- Set “Cluster radius” slider to the right edge (max), then click blue dot to see UDP in entire Japan area.
- Set “Cluster radius” slider to the left edge (min), so you can find each address grouped by AS number.



This is a good way to understand traffic by L4 protocols geometrically, such as country and AS.

#3 TCP Stream Graph

- Wireshark can list up all TCP/UDP connection using Conversation table, so you can pick up slow connection, create 5 types of TCP Stream Graph to visualize socket.
- Open “sf19-3.pcapng”, click Statistics >Conversation > TCP tab to list all TCP sockets and check Duration column grey bar. (you can also sort the column)

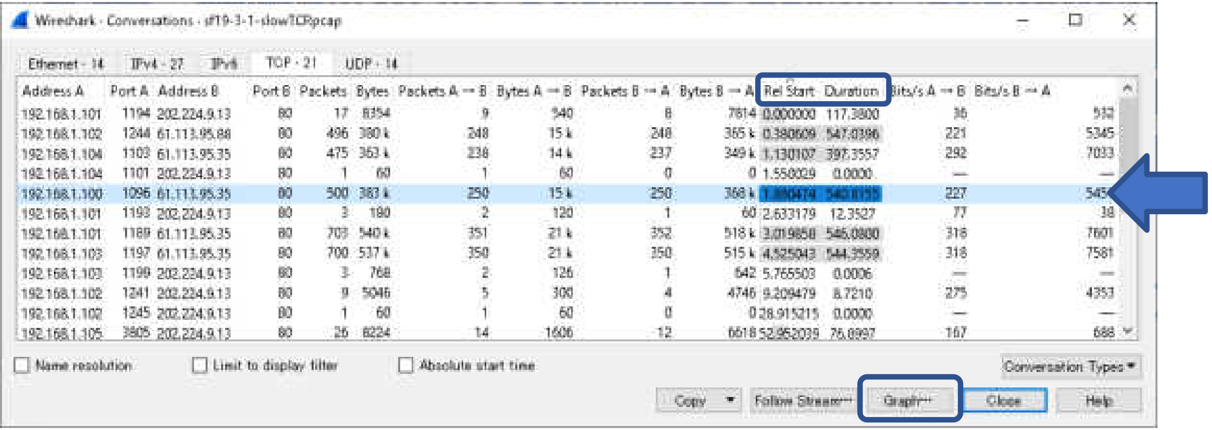


Wireshark · Conversations · sf19-3-1-slowTCP.pcap

Address A	Port A	Address B	Port 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
192.168.1.100	1096	61.113.95.35	80	500	383 k	250	15 k	250	368 k	5.880474	540.8155	227	5451
192.168.1.101	1184	202.224.9.13	80	17	8354	9	540	8	7814	0.000000	117.3800	36	532
192.168.1.101	1193	202.224.9.13	80	3	180	2	120	1	60	2.633179	12.2527	77	38
192.168.1.101	1189	61.113.95.35	80	703	540 k	351	21 k	352	518 k	[illegible]	[illegible]	318	7601
192.168.1.101	1195	192.168.1.103	139	33	4910	17	2635	16	2275	22.366652	0.0136	1553 k	1341 k
192.168.1.102	1244	61.113.95.88	80	496	380 k	248	15 k	248	365 k	0.380609	547.0396	221	5345
192.168.1.102	1241	202.224.9.13	80	9	5046	5	300	4	4746	9.209479	8.7210	275	4353
192.168.1.102	1245	202.224.9.13	80	1	60	1	60	0	0	28.915215	0.0000	—	—
192.168.1.103	1197	61.113.95.35	80	700	537 k	350	21 k	350	515 k	4.525040	544.3958	318	7581
192.168.1.103	1199	202.224.9.13	80	3	768	2	126	1	642	5.765503	0.0006	—	—
192.168.1.103	1198	202.224.9.13	80	3	900	3	900	0	0	63.170881	239.9508	30	0
192.168.1.103	1200	202.224.9.13	80	21	3240	12	966	9	2274	123.201184	71.7357	107	253

#3 TCP Stream Graph

- Pick up the conversation which took 546.0800 duration.
- Sort again with Rel Start and count the stream ID (TCP stream starts with 0, and this connection is 4)
- Confirm the direction (from B to A : downstream)

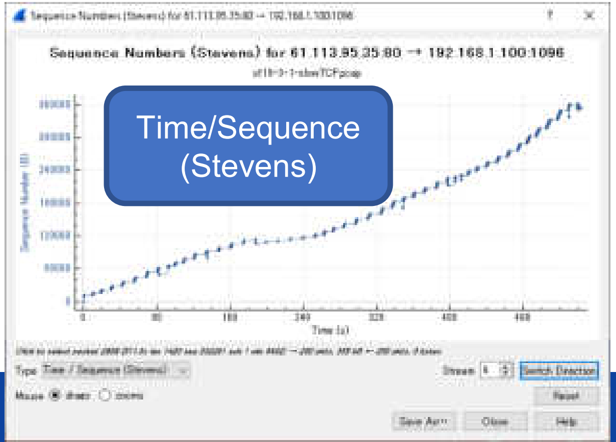


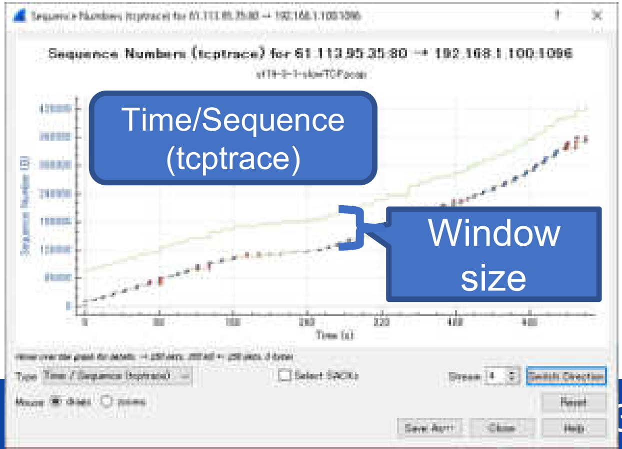
Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
192.168.1.101	1194	202.224.9.13	80	17	8354	9	540	8	7814	0.000000	117.3800	36	532
192.168.1.102	1244	61.113.95.88	80	496	380 k	248	15 k	248	365 k	0.380609	547.0396	221	5345
192.168.1.104	1103	61.113.95.35	80	475	363 k	238	14 k	237	349 k	1.130107	397.3557	292	7033
192.168.1.104	1101	202.224.9.13	80	1	60	1	60	0	0	1.550029	0.0000	—	—
192.168.1.100	1096	61.113.95.35	80	500	383 k	250	15 k	250	368 k	1.880474	540.8155	227	545
192.168.1.101	1193	202.224.9.13	80	3	180	2	120	1	60	2.633179	12.3527	77	38
192.168.1.101	1189	61.113.95.35	80	703	540 k	351	21 k	352	518 k	3.019858	546.0800	318	7601
192.168.1.103	1197	61.113.95.35	80	700	537 k	350	21 k	350	515 k	4.525043	544.3559	318	7581
192.168.1.103	1199	202.224.9.13	80	3	768	2	126	1	642	5.765503	0.0006	—	—
192.168.1.102	1241	202.224.9.13	80	9	5046	5	300	4	4746	9.209479	8.7210	275	4353
192.168.1.102	1245	202.224.9.13	80	1	60	1	60	0	0	28.915215	0.0000	—	—
192.168.1.105	3805	202.224.9.13	80	26	8224	14	1606	12	6618	52.952039	76.0997	167	688

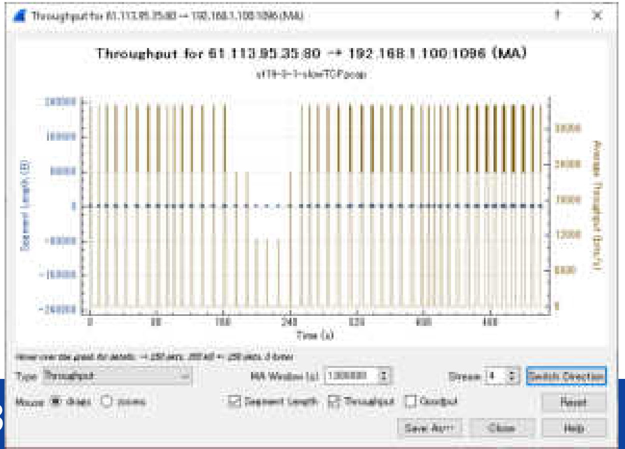
We look for this slow TCP connection (tcp.stream eq 4)
Press Graph after you find stream index

#3 TCP Stream Graph

- Press Graph button to visualize TCP steam
  Time / Sequence (Stevens) : understand stagnation
  Time / Sequence (tcptrace) : understand stagnation as well as window size
  Throughput : understand theoretical performance and segment length
- You can drag/zoom, and refer each packet number according to Wireshark main screen.

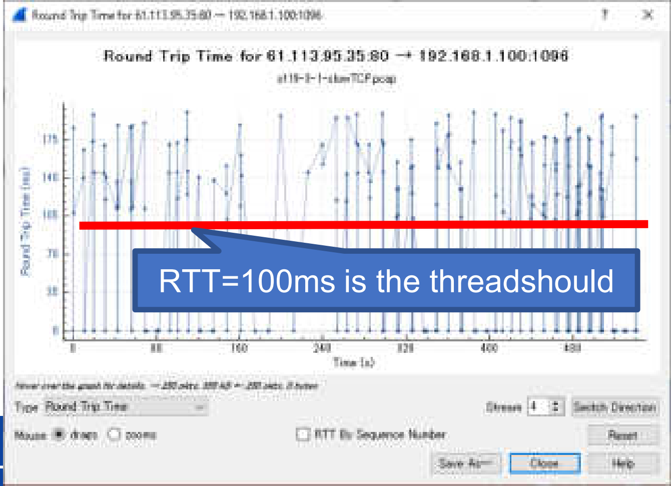






#3 TCP Stream Graph

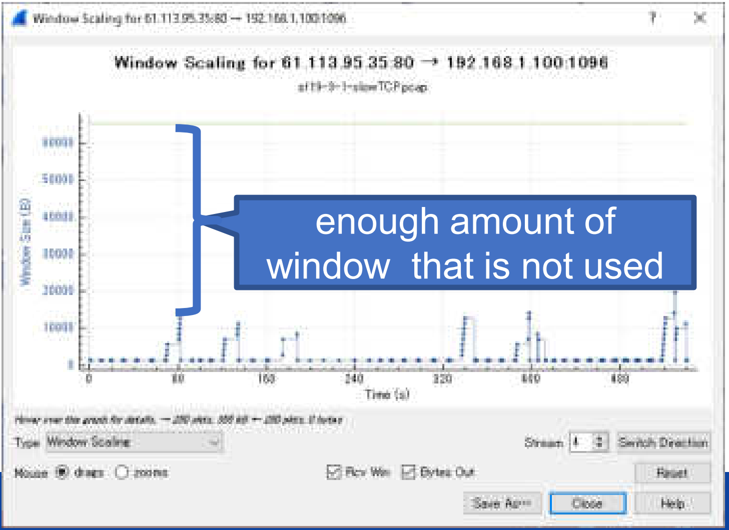
- Create Round Trip Time Graph and check RTT
  Ave. **RTT<1ms** Fast Intranet (may not think about TCP window mechanism but you still need to think of **Delayed ACK (40ms)**, Nagle and so on.
  RTT<50ms Extranet or Domestic Internet (You may think of Retransmission
  RTT>100ms International Internet or long range WAN links.
  (You must think of TCP receive window control a.k.a LFN)
- Average round trip time is about **100ms** in this time (the plot of 0 ms just says there are no packet) so let's create Window scaling Graph to determine TCP RWIN



RTT=100ms is the threadshould

#3 TCP Stream Graph

- TCP window control mechanism works in big RTT environment (it takes long time to ACK, so we need buffer for efficient conversation.).
- There are enough margins of TCP window size (Green RWIN vs Blue bytes out)
- This trace file was capture in old phone WAN link (128kbps) slow RTT and narrow bandwidth
- You may think about TCP RWIN in **LFN** (Large Fat Network)


Window Scaling for 61.113.95.35:80 → 192.168.1.100:1096

Window Scaling for 61.113.95.35:80 → 192.168.1.100:1096

sf19-3-1-slowTCP.pcap

enough amount of window that is not used

Hover over the graph for details. — 280 pkts, 800 kB — 280 pkts, 0 bytes

Type: Window Scaling

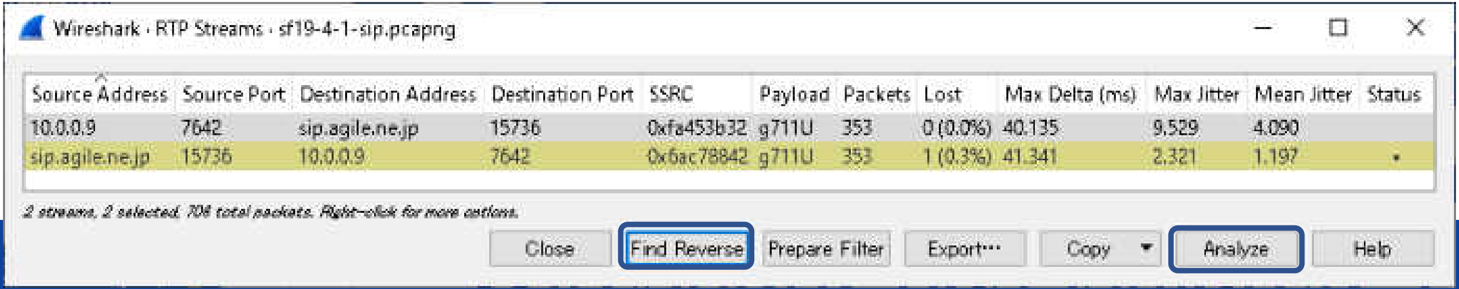
Stream: 4

Mouse ● drags ○ zooms

☑ Rcv Win ☑ Bytes Out

#4 RTP Graph

- Wireshark has Telephony menu to analyze VoIP, SIP/RTP/RTCP packets and you can also listen too.
- Open “sf19-4.pcapng” and click Telephony > RTP Streams, and click a row of RTP stream and press Find Reverse to select forward and reverse streams. (or Shift + Click to select multiple rows)
- Press Analyze button to see both direction at a glance

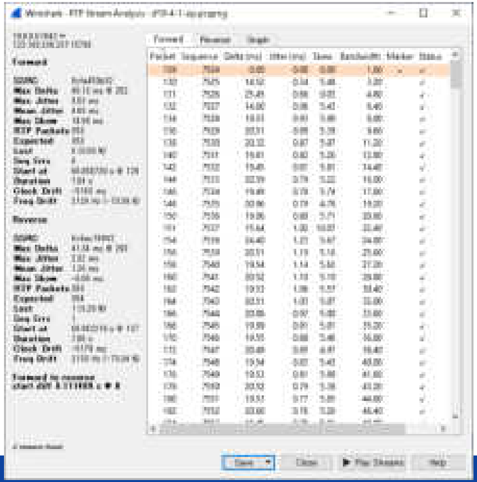


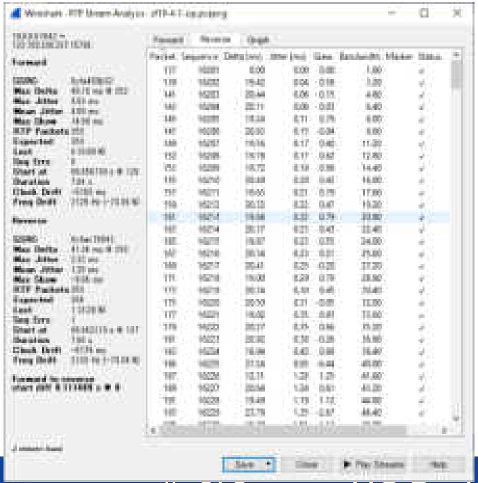
Source Address	Source Port	Destination Address	Destination Port	SSRC	Payload	Packets	Lost	Max Delta (ms)	Max Jitter	Mean Jitter	Status
10.0.0.9	7642	sip.agile.ne.jp	15736	0xfa453b32	g711U	353	0 (0.0%)	40.135	9.529	4.090	
sip.agile.ne.jp	15736	10.0.0.9	7642	0x6ac78842	g711U	353	1 (0.3%)	41.341	2.321	1.197	•

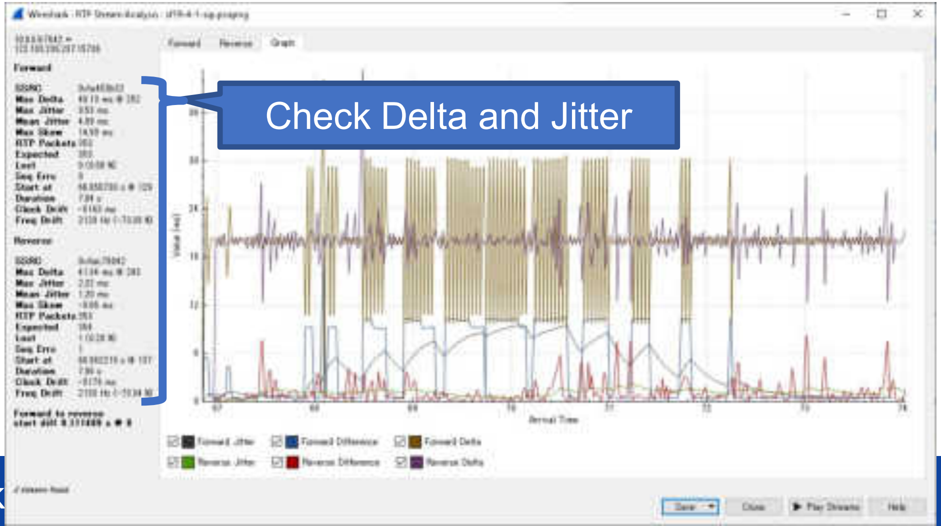
2 streams, 2 selected, 704 total packets. Right-click for more options.

#4 RTP Graph

- Select Forward and Reverse tabs to investigate stream.
- Visualize RTP at a glance to press Graph tab.
  Delta: **<150ms OK <400ms Alert >400ms NG**
  Jitter: **20ns – 1 micro sec.** (as the case may be by Human)







Check Delta and Jitter

#5 IO Graph

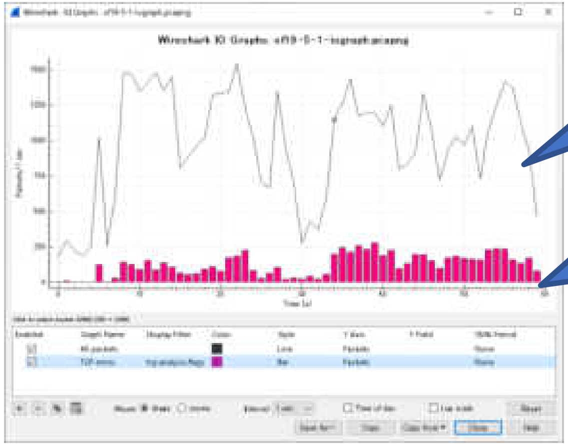
- IO graph is common method to visualize traffic, selecting adequate Y axis is very important.

1. Packet count graph : set Y axis by packets
2. Bandwidth graph : set Y axis by bits per seconds
3. Field value graph : choose math function to match.
4. Response time graph : set Y fields as http.time, etc.

- Open “sf19-5.pcapng” wireless trace file, and change profile to “customized IO Graph”
- Click Statistics > IO Graph

#5 IO Graph

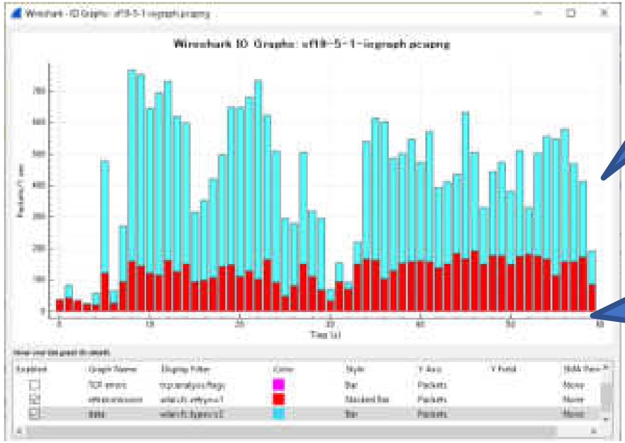
- What **style** is good for IO Graph ?
- If you want show the **movement of the value**, set **Line** is good idea, and if you want to show the **ratio of partition**, use **Stacked Bar** and **Bar**.



Enabled	Graph Name	Display Filter	Color	Style	Y Axis	Y Field	SMA Period
☑	All packets		■	Line	Packets		None
☑	TCP errors	tcp.analysis.flags	■	Bar	Packets		None

Line is good for understanding the movement

Bar is easy to understanding the value at the moment



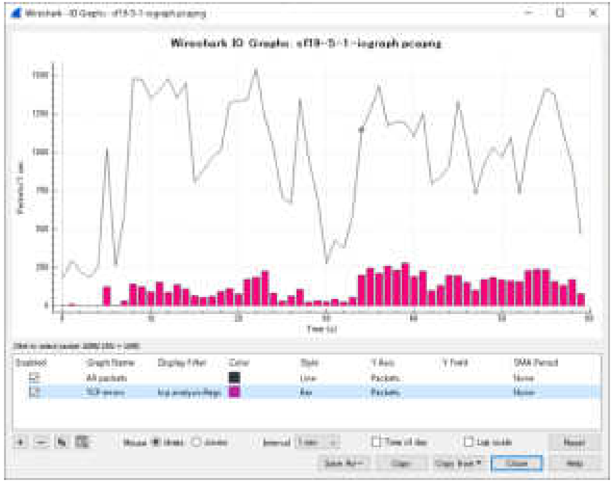
Enabled	Graph Name	Display Filter	Color	Style	Y Axis	Y Field	SMA Period
□	TCP errors	tcp.analysis.flags	■	Bar	Packets		None
☑	[illegible]	wlan.fc.retry==1	■	Stacked Bar	Packets		None
☑	[illegible]	wlan.fc.type==2	■	Bar	Packets		None

Stacked bar at the latter (backward)

Stacked bar at the first (front)

#5 IO Graph

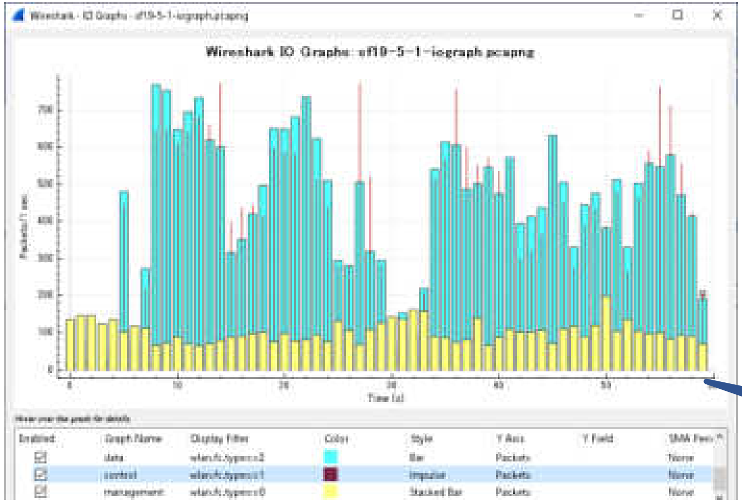
Set Y axis as packets to visualize counting frame by time. For example Wireshark shows all packets vs TCP error packets (default), Line is used by all packets, Bar is used for TCP error packets



Enabled	Graph Name	Display Filter	Color	Style	Y Axis	Y Field	SMA Period
☑	All packets		■	Line	Packets		None
☑	TCP errors	tcp.analysis.flags	■	Bar	Packets		None

"set Y axis by packets" can visualize counting frame by time it is good for understanding the ration of error, retransmission and frame types.

#5 IO Graph



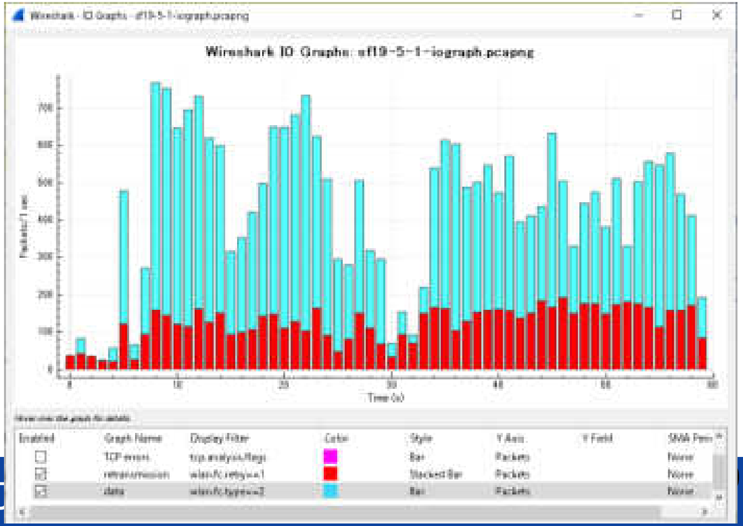
Wireshark · IO Graphs · sf19-5-1-iograph.pcapng
Wireshark IO Graphs: sf19-5-1-iograph.pcapng
Packets / 1 sec
0
100
200
300
400
500
600
700
0
10
20
30
40
50
Time (s)
Hover over the graph for details.

Enabled	Graph Name	Display Filter	Color	Style	Y Axis	Y Field	SMA Peri
☑	data	wlan.fc.type==2	🟦	Bar	Packets		None
☑	control	wlan.fc.type==1	🟥	Impulse	Packets		None
☑	management	wlan.fc.type==0	🟨	Stacked Bar	Packets		None

Left Graph shows the ratio of wireless frame types, such as management, control and data. You can understand the status of Wi-Fi

Stacked Bar for counting

Right Graph shows the ratio of data frame and retransmitted data frame. (wlan.fc.retry==1)

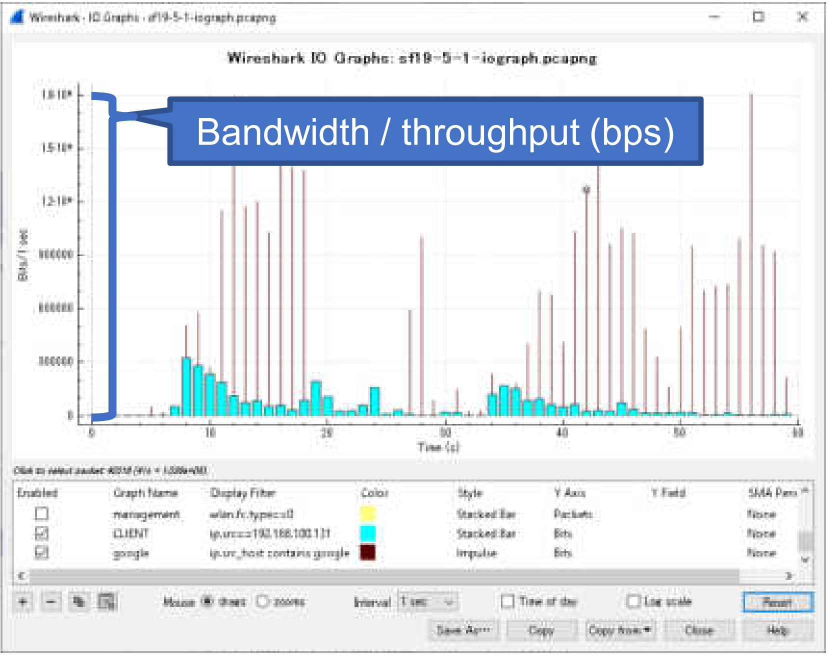


Wireshark · IO Graphs · sf19-5-1-iograph.pcapng
Wireshark IO Graphs: sf19-5-1-iograph.pcapng
Packets / 1 sec
0
100
200
300
400
500
600
700
0
10
20
30
40
50
60
Time (s)
Hover over the graph for details.

Enabled	Graph Name	Display Filter	Color	Style	Y Axis	Y Field	SMA Peri
□	TCP errors	tcp.analysis.flags	🟪	Bar	Packets		None
☑	retransmission	wlan.fc.retry==1	🟥	Stacked Bar	Packets		None
☑	data	wlan.fc.type==2	🟦	Bar	Packets		None

#5 IO Graph

2. Set Y axis by bits per seconds to visualize bandwidth


Wireshark · IO Graphs · sf19-5-1-iograph.pcapng
Wireshark IO Graphs: sf19-5-1-iograph.pcapng
1.8·10⁶
1.5·10⁶
1.2·10⁶
900000
600000
300000
0
Bits / sec
0
10
20
30
40
50
60
Time (s)
Click to select packet 4038 (41s = 1.008e+06)

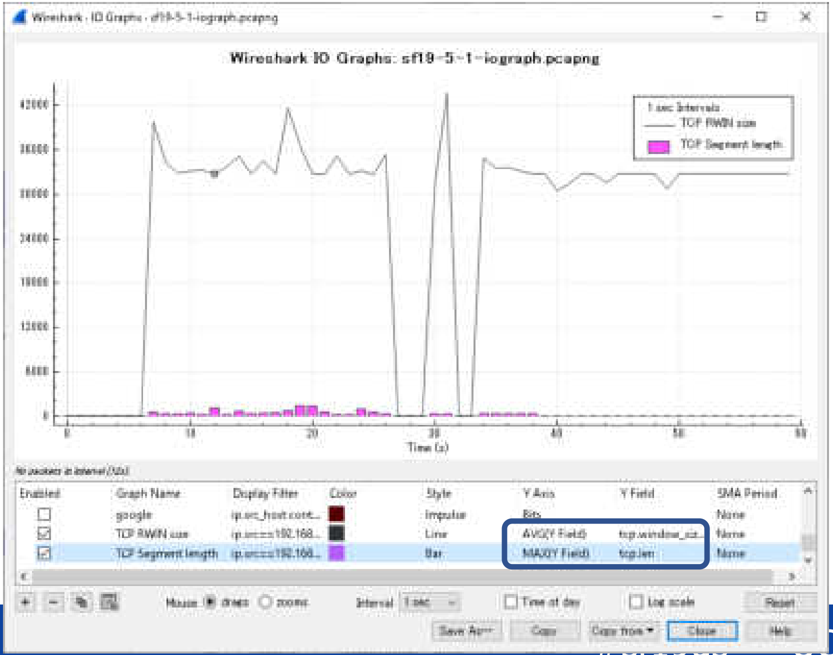
Bandwidth / throughput (bps)

Enabled	Graph Name	Display Filter	Color	Style	Y Axis	Y Field	SMA Period
□	management	wlan.fc.type==0	🟨	Stacked Bar	Packets		None
☑	CLIENT	ip.src==192.168.100.131	🟦	Stacked Bar	Bits		None
☑	google	ip.src_host contains google	🟥	Impulse	Bits		None

- set Y axis by bits
- Read Y axis as 10^6 Mbps
- Compare CLIENT (ip.addr==192.168.100.135) and Google traffic (ip.src_host contains google)
- "set Y axis by bits" IO Graph is good for throughput

#5 IO Graph

3. set Y field and choose math function to visualize specific field value

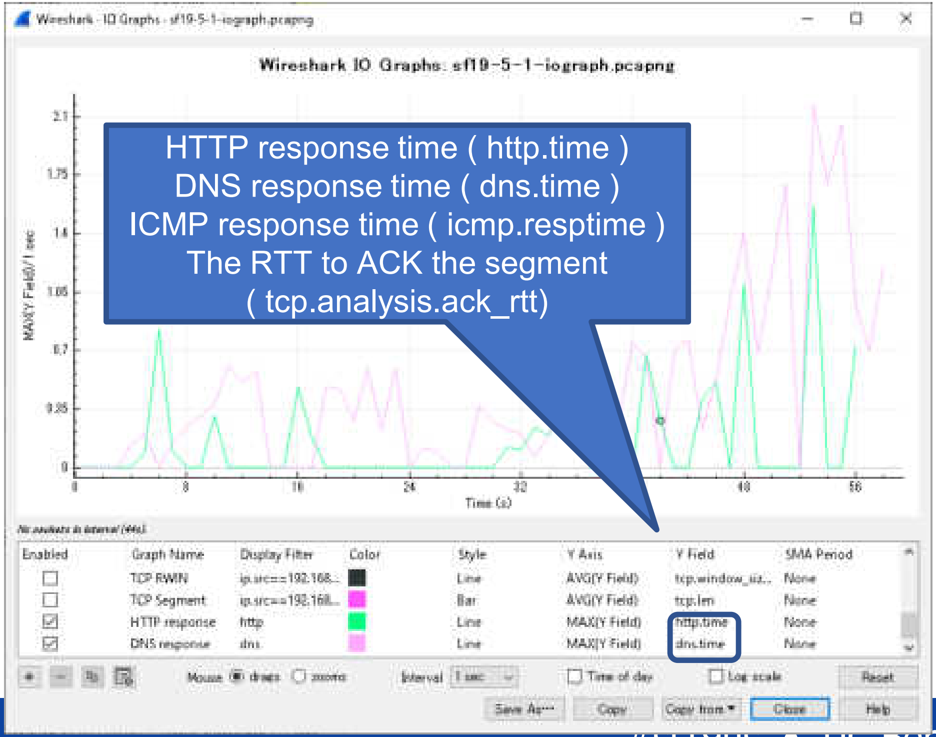

Wireshark · IO Graphs · sf19-5-1-iograph.pcapng
Wireshark IO Graphs: sf19-5-1-iograph.pcapng
42000
36000
30000
24000
18000
12000
6000
0
0
10
20
30
40
50
60
Time (s)
1 sec Intervals
TCP RWIN size
TCP Segment length
No packets in interval (32s)

Enabled	Graph Name	Display Filter	Color	Style	Y Axis	Y Field	SMA Period
□	google	ip.src_host cont…	■	Impulse	Bits		None
☑	TCP RWIN size	ip.src==192.168…	■	Line	AVG(Y Field)	tcp.window_si…	None
☑	TCP Segment length	ip.src==192.168…	■	Bar	MAX(Y Field)	tcp.len	None

- Add two items of TCP RWIN and TCP segment.
- Set Y Field as the average of calculated RWIN (tcp.window_size), and maximum of segment length (tcp.len)
- Check TCP is OK or not.

#5 IO Graph

4. Response time graph : set Y fields as http.time, etc.



Wireshark · IO Graphs · sf19-5-1-iograph.pcapng
Wireshark IO Graphs: sf19-5-1-iograph.pcapng
MAX(Y Field)/1 sec
2.1
1.75
1.4
1.05
0.7
0.35
0
0
8
16
24
32
40
56
Time (s)
No packets in interval (44s).

HTTP response time (http.time)
DNS response time (dns.time)
ICMP response time (icmp.resptime)
The RTT to ACK the segment
(tcp.analysis.ack_rtt)

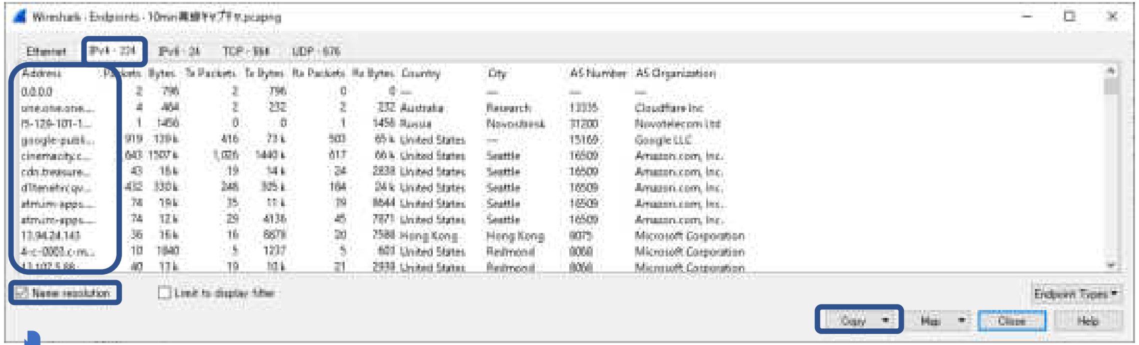
Enabled	Graph Name	Display Filter	Color	Style	Y Axis	Y Field	SMA Period
☐	TCP RWIN	ip.src==192.168...	■	Line	AVG(Y Field)	tcp.window_siz...	None
☐	TCP Segment	ip.src==192.168...	■	Bar	AVG(Y Field)	tcp.len	None
☑	HTTP response	http	■	Line	MAX(Y Field)	http.time	None
☑	DNS response	dns	■	Line	MAX(Y Field)	dns.time	None

- Add two items of HTTP DNS response time.
- Set Y fields as Maximum of http.time and dns.time.
- Compare response time.
- Not HTTP but DNS is the problem at the worst case.

#6 Copy table values as CSV

- IO Graph is very useful to visualize traffic, but it uses only Time as X axis… We want to visualize traffic using various dimension except for Time.
- Wireshark has various plugin table for statistics. You can copy table values as CSV, then utilize them to Excel as Histograms and so on
- Open “sf19-6.pcapng” that contains 10 mins wireless client traffic. And choose Statics > Endpoints, and select IPv4 tab. Also check “Resolve Network Address” from name resolution.

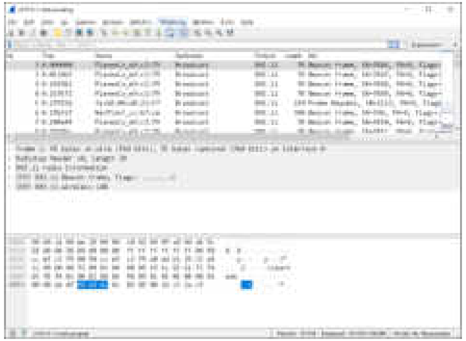
#6 Copy table values as CSV



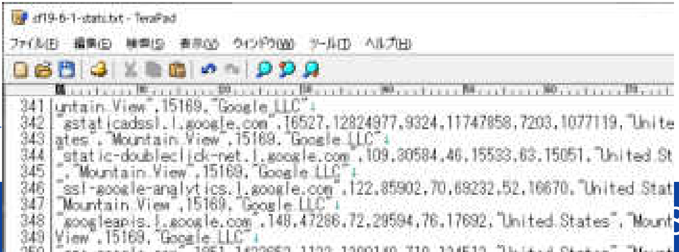
Wireshark · Endpoints · 10min長時間キャプチャ.pcapng

Address	Packets	Bytes	Tx Packets	Tx Bytes	Rx Packets	Rx Bytes	Country	City	AS Number	AS Organization
0.0.0.0	2	796	2	796	0	0	—	—	—	—
one.one.one...	4	464	2	232	2	232	Australia	Research	13335	Cloudflare Inc
f5-129-101-1...	1	1456	0	0	1	1456	Russia	Novosibirsk	31200	Novotelecom Ltd
google-publi...	919	139 k	416	73 k	503	65 k	United States	—	15169	Google LLC
cinemacity.c...	1,643	1507 k	1,026	1440 k	617	66 k	United States	Seattle	16509	Amazon.com, Inc.
cdn.treasure...	43	16 k	19	14 k	24	2838	United States	Seattle	16509	Amazon.com, Inc.
d1lenehrcqv...	432	330 k	248	305 k	184	24 k	United States	Seattle	16509	Amazon.com, Inc.
atmum-apps...	74	19 k	35	11 k	39	8644	United States	Seattle	16509	Amazon.com, Inc.
atmum-apps...	74	12 k	29	4136	45	7871	United States	Seattle	16509	Amazon.com, Inc.
13.94.24.143	36	15 k	16	6679	20	7588	Hong Kong	Hong Kong	8075	Microsoft Corporation
4-c-0003.c-m...	10	1840	5	1237	5	603	United States	Redmond	8068	Microsoft Corporation
13.107.5.88	40	17 k	19	10 k	21	2938	United States	Redmond	8068	Microsoft Corporation

☑ Name resolution □ Limit to display filter



- How about visualize host name as X axis ?
  Check “Name resolution” and press Copy “as CSV” and paste them to “sf19-6-1.txt”

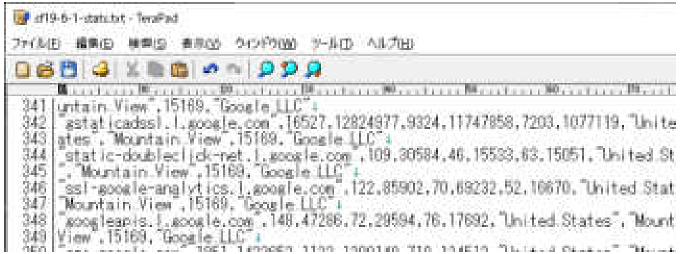


sf19-6-1-stats.txt - TeraPad

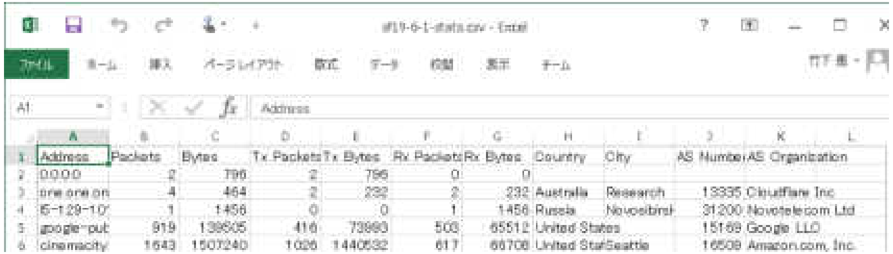
ファイル(F) 編集(E) 検索(S) 表示(V) ウィンドウ(W) ツール(T) ヘルプ(H)

```
untain View",15169,"Google LLC"
"gstaticadssl.l.google.com",16527,12824977,9324,11747858,7203,1077119,"Unite
ates","Mountain View",15169,"Google LLC"
"static-doubleclick-net.l.google.com",109,30584,46,15533,63,15051,"United St
","Mountain View",15169,"Google LLC"
"ssl-google-anglytics.l.google.com",122,85902,70,69232,52,16670,"United Stat
"Mountain View",15169,"Google LLC"
"googleapis.l.google.com",148,47286,72,29594,76,17692,"United States","Mount
View",15169,"Google LLC"
```

#6 Copy table values as CSV



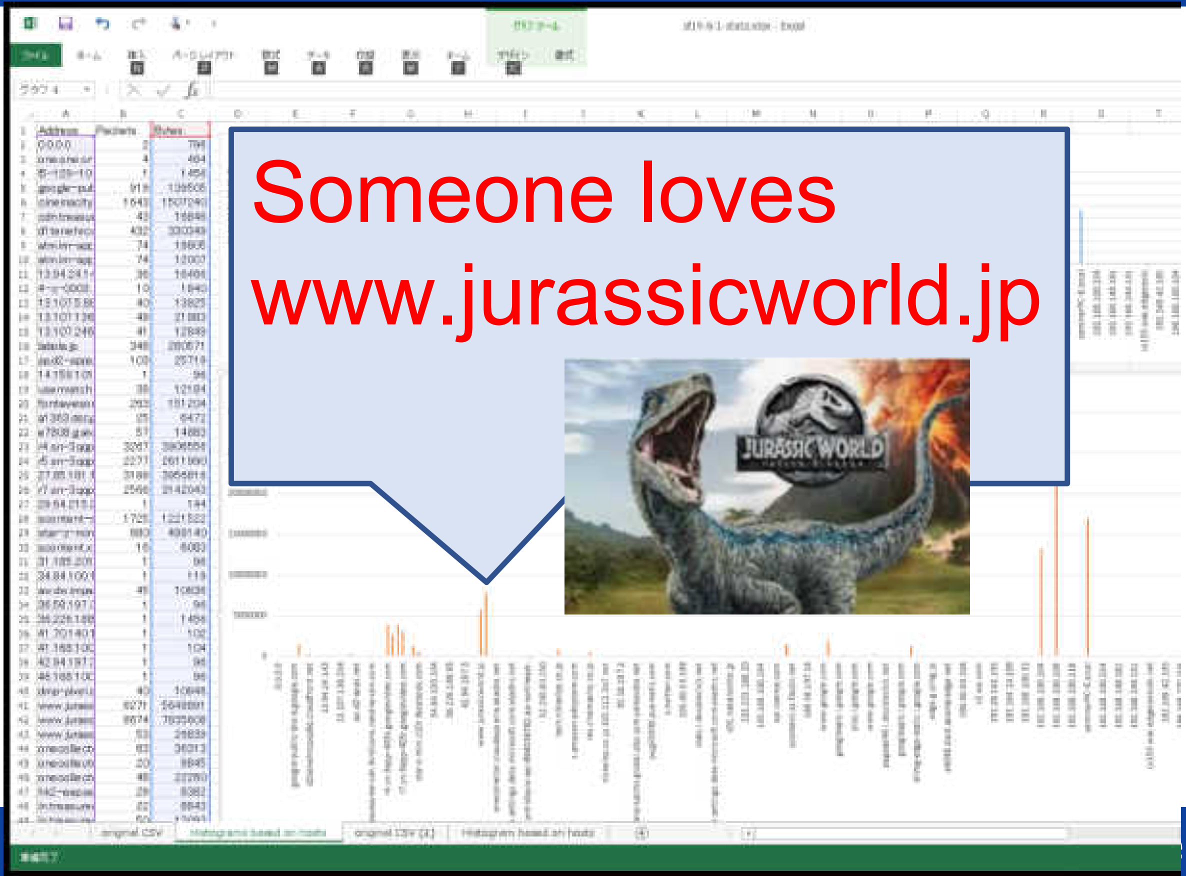
```
untain View",15169,"Google LLC"↓
"gstaticadssl.l.google.com",16527,12824977,9324,11747858,7203,1077119,"Unite
ates","Mountain View",15169,"Google LLC"↓
"static-doubleclick-net.l.google.com",109,30684,46,15533,63,15051,"United St
","Mountain View",15169,"Google LLC"↓
"ssl-google-anglytics.l.google.com",122,85902,70,69232,52,16670,"United Stat
"Mountain View",15169,"Google LLC"↓
"googleapis.l.google.com",148,47286,72,29594,76,17692,"United States","Mount
View",15169,"Google LLC"↓
```



	A	B	C	D	E	F	G	H	I	J	K	L
1	Address	Packets	Bytes	Tx Packets	Tx Bytes	Rx Packets	Rx Bytes	Country	City	AS Number	AS Organization	
2	0.0.0.0	2	796	2	796	0	0					
3	one.one.on	4	464	2	232	2	232	Australia	Research	13335	Cloudflare Inc	
4	5-129-10	1	1456	0	0	1	1456	Russia	Novosibirsk	31200	Novotelecom Ltd	
5	google-pub	919	139505	416	73993	503	65512	United States		15169	Google LLC	
6	cinemacity	1643	1507240	1026	1440532	617	66708	United Stat	Seattle	16509	Amazon.com, Inc.	

- Change extentions from txt to csv, start visualization using Microsoft Excel or other apps.
- In this case, using Excel to create a new sheet.
- Open sample visualization example table file “sf19-6-1.xlsx”

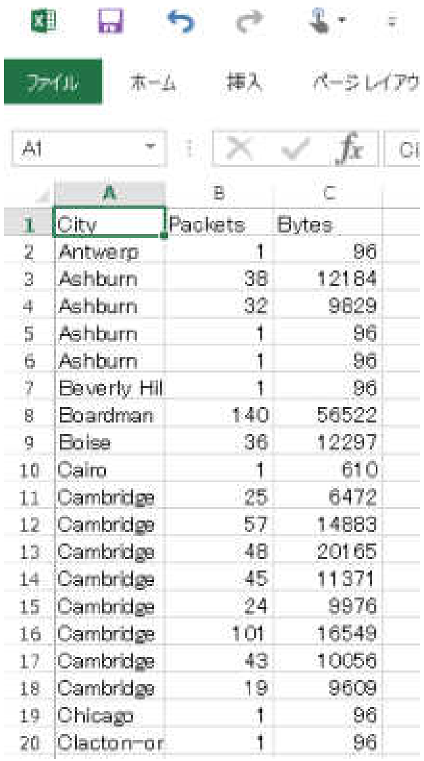
#6 Copy table values as CSV



Someone loves www.jurassicworld.jp

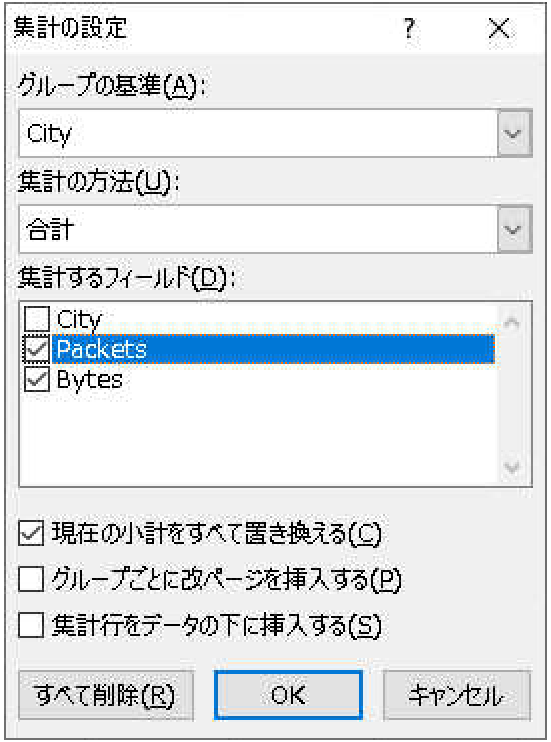
- Just cut Address, Packets, and Bytes Rows, then paste another tab. Then Insert > Graph to create Hisograms
- Someone loves Jurassic World movie.

#6 Copy table values as CSV



	A	B	C
1	City	Packets	Bytes
2	Antwerp	1	96
3	Ashburn	38	12184
4	Ashburn	32	9829
5	Ashburn	1	96
6	Ashburn	1	96
7	Beverly Hil	1	96
8	Boardman	140	56522
9	Boise	36	12297
10	Cairo	1	610
11	Cambridge	25	6472
12	Cambridge	57	14883
13	Cambridge	48	20165
14	Cambridge	45	11371
15	Cambridge	24	9976
16	Cambridge	101	16549
17	Cambridge	43	10056
18	Cambridge	19	9609
19	Chicago	1	96
20	Clacton-or	1	96

- Copy CSV to another sheet, edit rows following City, Packets and Bytes. Then group by City name, clicking Data > subtotal
- Set group by City, count by Total of Packets and Bytes rows, then press OK



集計の設定

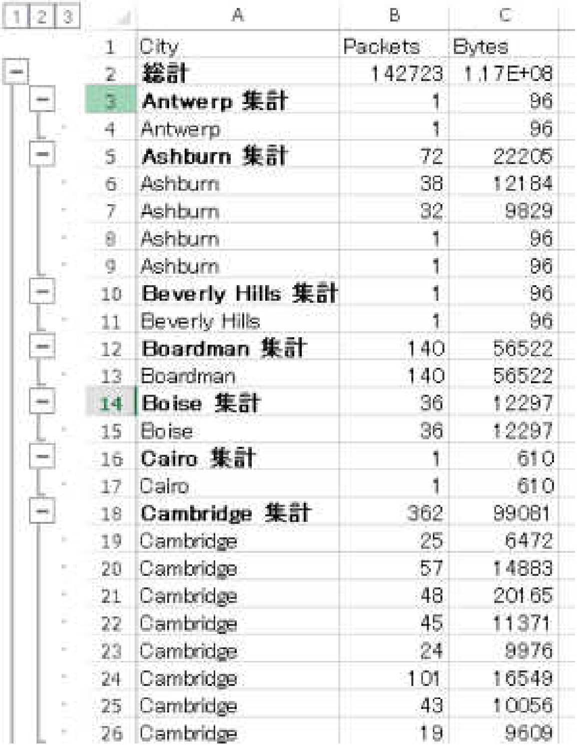
グループの基準(A): City

集計の方法(U): 合計

集計するフィールド(D):
□ City
☑ Packets
☑ Bytes

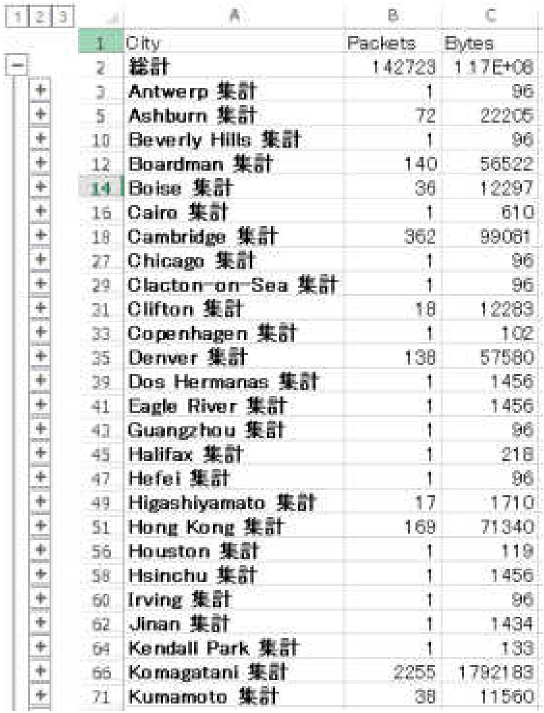
☑ 現在の小計をすべて置き換える(C)
□ グループごとに改ページを挿入する(P)
□ 集計行をデータの下に挿入する(S)

#6 Copy table values as CSV



	A	B	C
1	City	Packets	Bytes
2	**総計**	142723	1.17E+08
3	**Antwerp 集計**	1	96
4	Antwerp	1	96
5	**Ashburn 集計**	72	22205
6	Ashburn	38	12184
7	Ashburn	32	9829
8	Ashburn	1	96
9	Ashburn	1	96
10	**Beverly Hills 集計**	1	96
11	Beverly Hills	1	96
12	**Boardman 集計**	140	56522
13	Boardman	140	56522
14	**Boise 集計**	36	12297
15	Boise	36	12297
16	**Cairo 集計**	1	610
17	Cairo	1	610
18	**Cambridge 集計**	362	99081
19	Cambridge	25	6472
20	Cambridge	57	14883
21	Cambridge	48	20165
22	Cambridge	45	11371
23	Cambridge	24	9976
24	Cambridge	101	16549
25	Cambridge	43	10056
26	Cambridge	19	9609

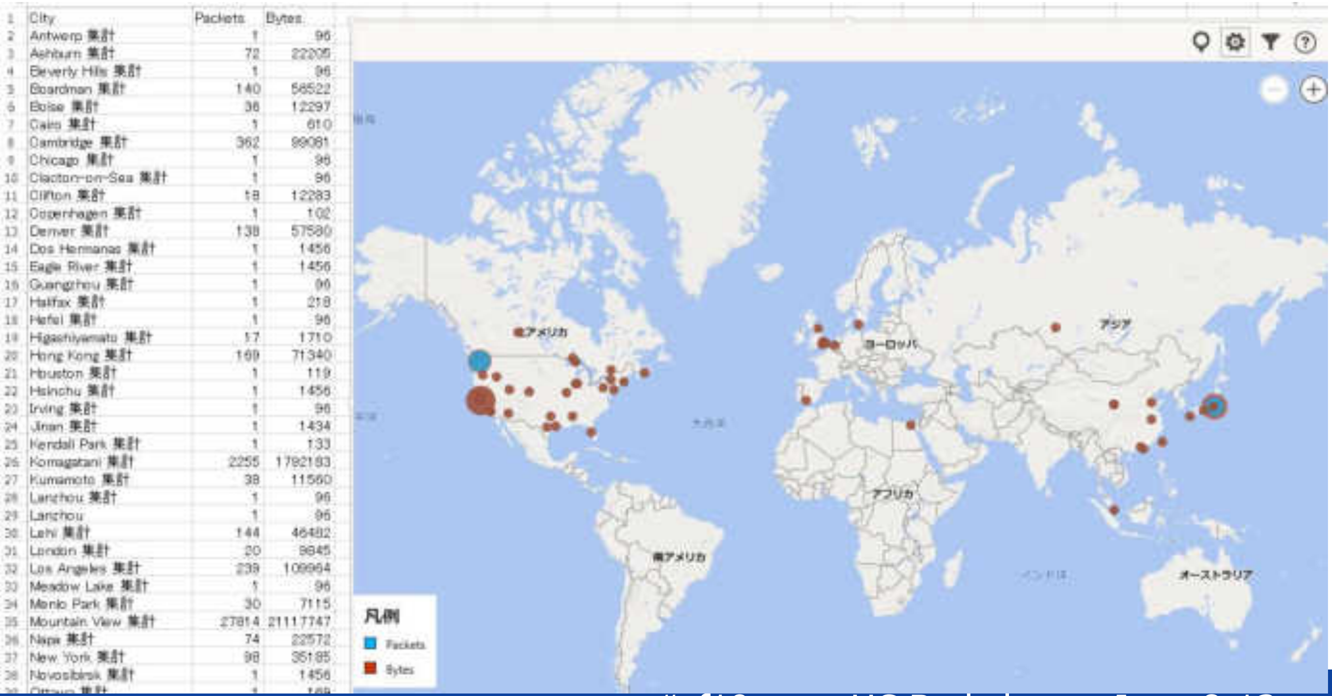
- Press left side group button [2].
- Copy City, Packets, and Bytes row subtotaled by City and paste values into another sheet.
- Edit some cells to limit top 100 data



	A	B	C
1	City	Packets	Bytes
2	**総計**	142723	1.17E+08
3	**Antwerp 集計**	1	96
5	**Ashburn 集計**	72	22205
10	**Beverly Hills 集計**	1	96
12	**Boardman 集計**	140	56522
14	**Boise 集計**	36	12297
16	**Cairo 集計**	1	610
18	**Cambridge 集計**	362	99081
27	**Chicago 集計**	1	96
29	**Clacton-on-Sea 集計**	1	96
31	**Clifton 集計**	18	12283
33	**Copenhagen 集計**	1	102
35	**Denver 集計**	138	57580
39	**Dos Hermanas 集計**	1	1456
41	**Eagle River 集計**	1	1456
43	**Guangzhou 集計**	1	96
45	**Halifax 集計**	1	218
47	**Hefei 集計**	1	96
49	**Higashiyamato 集計**	17	1710
51	**Hong Kong 集計**	169	71340
56	**Houston 集計**	1	119
58	**Hsinchu 集計**	1	1456
60	**Irving 集計**	1	96
62	**Jinan 集計**	1	1434
64	**Kendall Park 集計**	1	133
66	**Komagatani 集計**	2255	1792183
71	**Kumamoto 集計**	38	11560

#6 Copy table values as CSV

- Create another tab and copy City, Bytes and Packets.

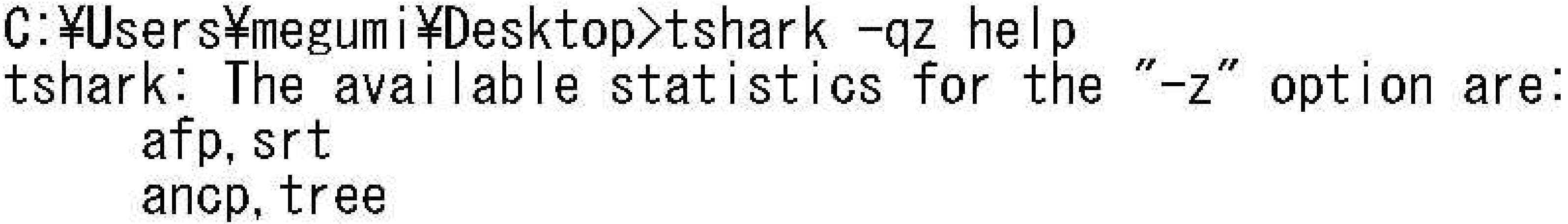


City	Packets	Bytes
Antwerp 集計	1	96
Ashburn 集計	72	22205
Beverly Hills 集計	1	96
Boardman 集計	140	58522
Boise 集計	36	12297
Cairo 集計	1	610
Cambridge 集計	362	99081
Chicago 集計	1	96
Clacton-on-Sea 集計	1	96
Clifton 集計	18	12283
Copenhagen 集計	1	102
Denver 集計	138	57580
Dos Hermanas 集計	1	1456
Eagle River 集計	1	1456
Guangzhou 集計	1	96
Halifax 集計	1	218
Hefei 集計	1	96
Higashiyamato 集計	17	1710
Hong Kong 集計	169	71340
Houston 集計	1	119
Hsinchu 集計	1	1456
Irving 集計	1	96
Jinan 集計	1	1434
Kendall Park 集計	1	133
Komagatani 集計	2255	1782183
Kumamoto 集計	38	11560
Lanzhou 集計	1	96
Lanzhou	1	96
Lehi 集計	144	46482
London 集計	20	9845
Los Angeles 集計	239	109964
Meadow Lake 集計	1	96
Menlo Park 集計	30	7115
Mountain View 集計	27814	21117747
Napa 集計	74	22572
New York 集計	98	35185
Novosibirsk 集計	1	1456
Ottawa 集計	1	169

- Insert Graph > Bing Map
- Press Filter button to set data region.
- You can see packets and data in Map

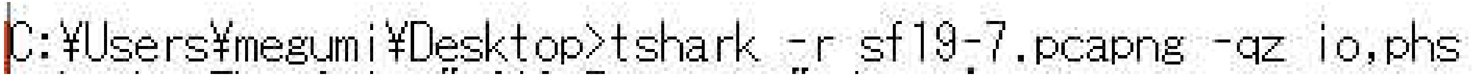
#7 Create statistics using tshark

- Tshark is a CLI version of Wireshark, so tshark can use some statistic plugin with –qz option. Check online help with “tshark –qz help”



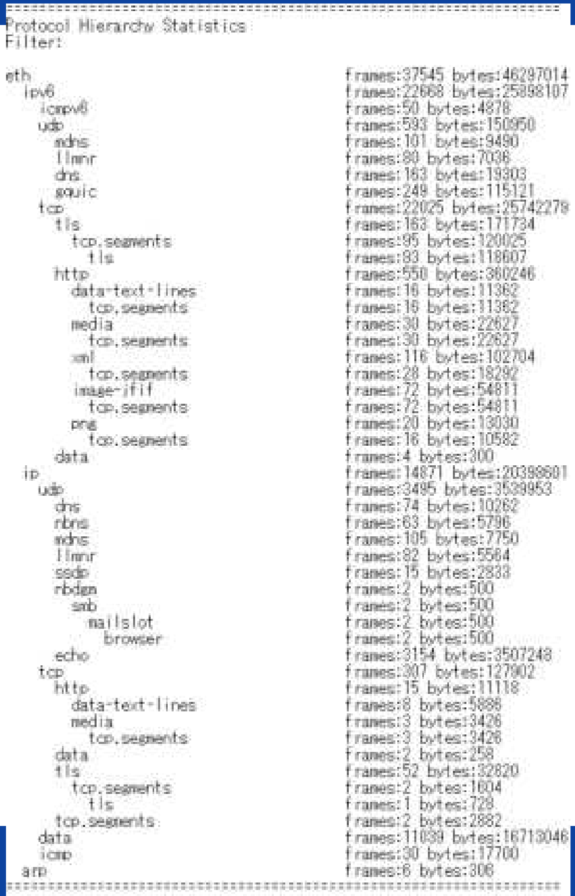
```
C:¥Users¥megumi¥Desktop>tshark -qz help
tshark: The available statistics for the "-z" option are:
     afp,srt
     ancp,tree
```

- The option of protocol hierarchy statistics chart is “io,phs” so open “sf19-7.pcapng” with “-qz io,phs”



```
C:¥Users¥megumi¥Desktop>tshark -r sf19-7.pcapng -qz io,phs
```

#7 Create statistics using tshark



```
===================================================================
Protocol Hierarchy Statistics
Filter:

eth                                      frames:37545 bytes:46297014
  ipv6                                   frames:22668 bytes:25858107
    icmpv6                               frames:50 bytes:4878
    udp                                  frames:593 bytes:150950
      mdns                               frames:101 bytes:9490
      llmnr                              frames:80 bytes:7036
      dns                                frames:163 bytes:19303
      gquic                              frames:249 bytes:115121
    tcp                                  frames:22025 bytes:25742279
      tls                                frames:163 bytes:171734
        tcp.segments                     frames:95 bytes:120025
          tls                            frames:83 bytes:118607
      http                               frames:550 bytes:360246
        data-text-lines                  frames:16 bytes:11362
          tcp.segments                   frames:16 bytes:11362
        media                            frames:30 bytes:22627
          tcp.segments                   frames:30 bytes:22627
        xml                              frames:116 bytes:102704
          tcp.segments                   frames:28 bytes:18292
        image-jfif                       frames:72 bytes:54811
          tcp.segments                   frames:72 bytes:54811
        png                              frames:20 bytes:13030
          tcp.segments                   frames:16 bytes:10582
      data                               frames:4 bytes:300
  ip                                     frames:14871 bytes:20399601
    udp                                  frames:3495 bytes:3529953
      dns                                frames:74 bytes:10262
      nbns                               frames:63 bytes:5796
      mdns                               frames:105 bytes:7750
      llmnr                              frames:82 bytes:5564
      ssdp                               frames:15 bytes:2833
      nbdgm                              frames:2 bytes:500
        smb                              frames:2 bytes:500
          mailslot                       frames:2 bytes:500
            browser                      frames:2 bytes:500
      echo                               frames:3154 bytes:3507248
    tcp                                  frames:307 bytes:127902
      http                               frames:15 bytes:11118
        data-text-lines                  frames:8 bytes:5888
        media                            frames:3 bytes:3426
          tcp.segments                   frames:3 bytes:3426
      data                               frames:2 bytes:258
      tls                                frames:52 bytes:32820
        tcp.segments                     frames:2 bytes:1604
          tls                            frames:1 bytes:728
      tcp.segments                       frames:2 bytes:2882
    data                                 frames:11039 bytes:16713046
    icmp                                 frames:30 bytes:17700
  arp                                    frames:6 bytes:306
===================================================================
```

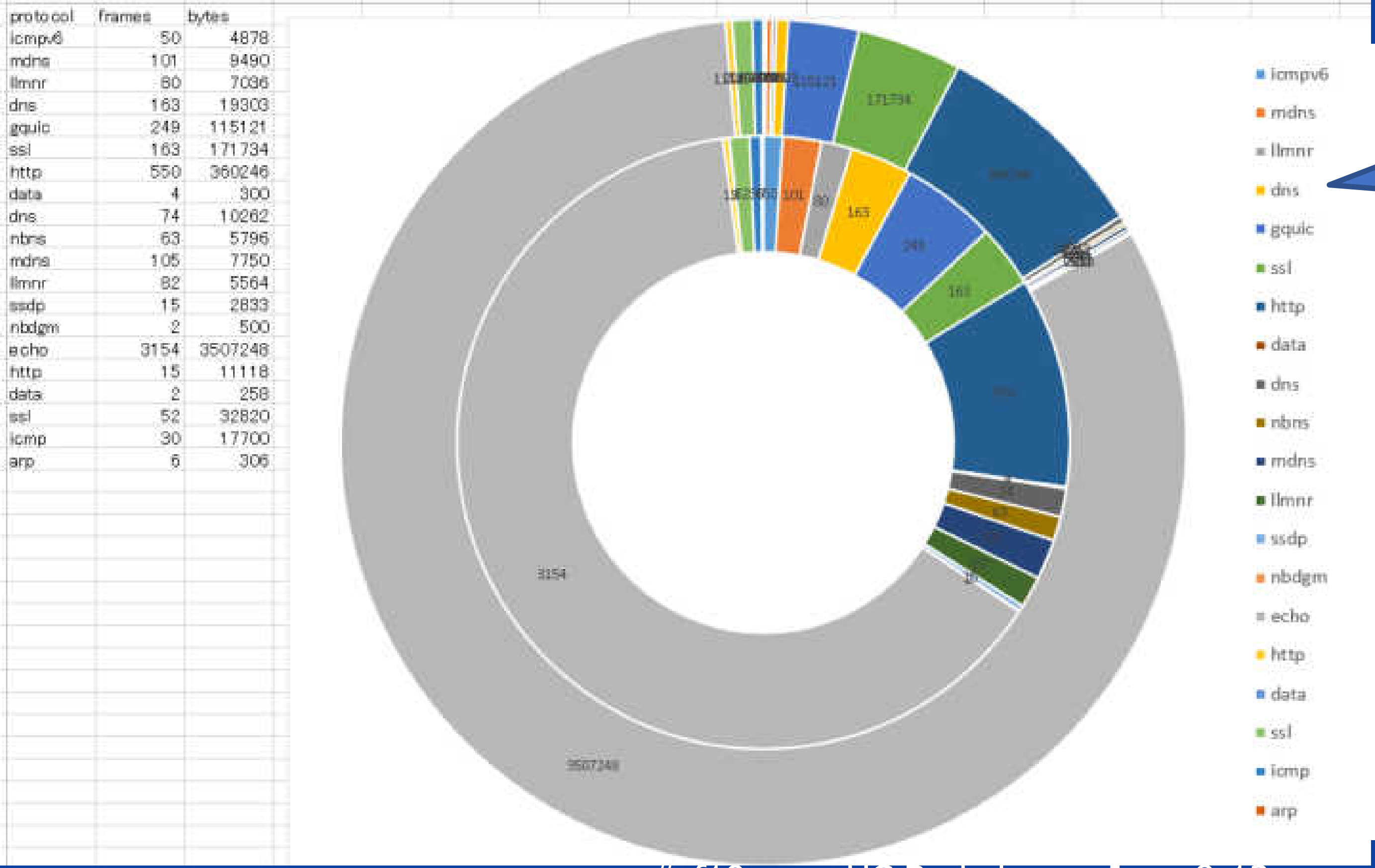
- We got protocol hierarchy statistics of all protocols in text format.
- For making pie chart we need to process text data to match CSV.
- Remove “frames:” and “bytes:” using sed -e 's/frames://' -e 's/bytes://` using bash
- Redirect output stream as phs.csv

tshark -r sf19-7.pcapng -qz io,phs | sed -e 's/frames://' -e 's/bytes://' >> phs.csv

#7 Create statistics using tshark

- tshark -r sf19-7.pcapng -qz io,phs | sed -e 's/frames://' -e 's/bytes://' >> phs.csv
  you also may use “tr -d ' ‘” to remove space character
- Open csv in Excel and create a new sheet and copy from original data and remove unnecessary lines.
- Set Data>Delimiter as space and add header line
- Insert Graph > Donut Pie Chart and customize color, size, index, title, etc
- Finally we can find UDP echo is the majority

#7 Use tshark to create statistics



protocol	frames	bytes
icmpv6	50	4878
mdns	101	9490
llmnr	80	7036
dns	163	19303
gquic	249	115121
ssl	163	171734
http	550	360246
data	4	300
dns	74	10262
nbns	63	5796
mdns	105	7750
llmnr	82	5564
ssdp	15	2833
nbdgm	2	500
echo	3154	3507248
http	15	11118
data	2	258
ssl	52	32820
icmp	30	17700
arp	6	306

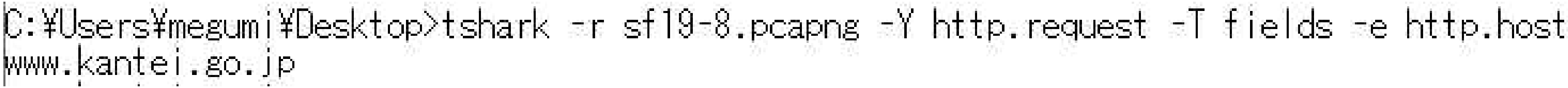
Nice donut,
Isn't it ?

#8 Collect fields for Visualization

- Tshark is a CLI version of Wireshark, as well as nice data processing tool for visualization from trace files.
- Check –T option and you can pick up any fields of dissector from trace file like –T fields –e ip.src
- This time we want to collect host information of http open “sf19-8.pcapng” using tshark
  and collect http.host field information as below

```
tshark –r sf19-8.pcapng –Y http.request  –T fields
–e http.host ( use –R read filter if huge trace file )
```

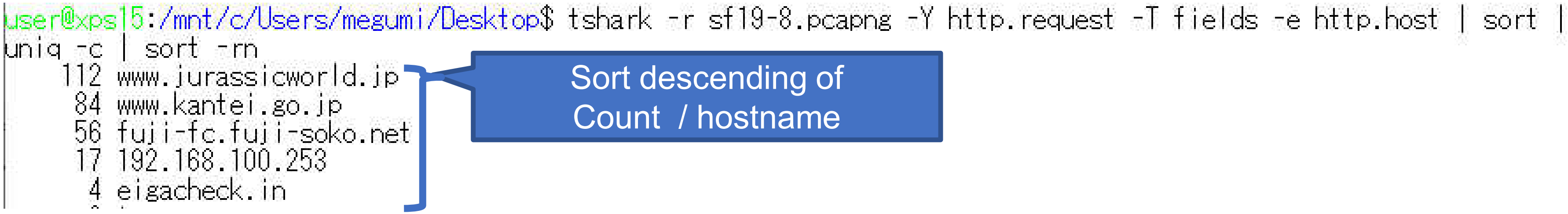
#8 Collect fields for Visualization



```
C:¥Users¥megumi¥Desktop>tshark -r sf19-8.pcapng -Y http.request -T fields -e http.host
www.kantei.go.jp
```

- The output contains host header information in each http request, start data processing for visualization
- At first we need bash and the typical technics below sort an output stream, then count the same line, and sort again for descending for top list
  tshark -r sf19-8.pcapng -Y http.request -T fields -e http.host **| sort | uniq -c | sort –rn**
  (sort alphabetically and count duplications)

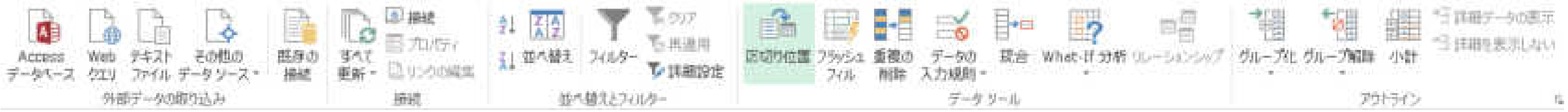
#8 Collect fields for Visualization



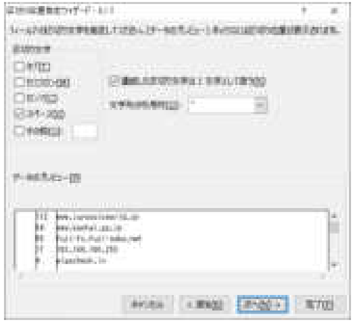
```
user@xps15:/mnt/c/Users/megumi/Desktop$ tshark -r sf19-8.pcapng -Y http.request -T fields -e http.host | sort |
uniq -c | sort -rn
    112 www.jurassicworld.jp
     84 www.kantei.go.jp
     56 fuji-fc.fuji-soko.net
     17 192.168.100.253
      4 eigacheck.in
```

Sort descending of
Count / hostname

- Redirecting the output stream as csv
  tshark -r sf19-8.pcapng -Y http.request -T fields
  -e http.host | sort | uniq -c | sort -rn >> hostlist.csv
- Open CSV file and set delimiter using Excel



#8 Collect fields for Visualization


区切り位置指定ウィザード - 2/3

フィールドの区切り文字を指定してください。[データのプレビュー] ボックスには区切り位置が表示されます。

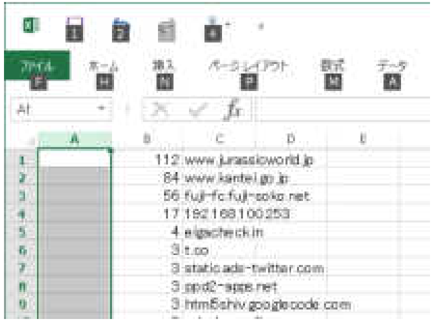
区切り文字
- □ タブ(T)
- □ セミコロン(M)
- □ カンマ(C)
- ☑ スペース(S)
- □ その他(O)

☑ 連続した区切り文字は1文字として扱う(R)

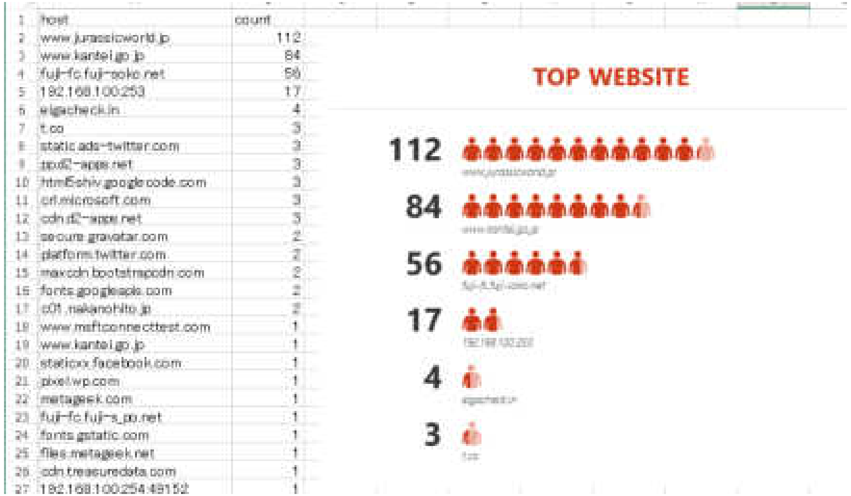
文字列の引用符(Q): "

データのプレビュー(P)

```
112  www.jurassicworld.jp
84   www.kantei.go.jp
56   fuji-fc.fuji-soko.net
17   192.168.100.253
4    eigacheck.in
```



	A	B	C	D	E
1		112 www.jurassicworld.jp			
2		84 www.kantei.go.jp			
3		56 fuji-fc.fuji-soko.net			
4		17 192.168.100.253			
5		4 eigacheck.in			
6		3 t.co			
7		3 static.ads-twitter.com			
8		3 jp.d2-apps.net			
9		3 html5shiv.googlecode.com			

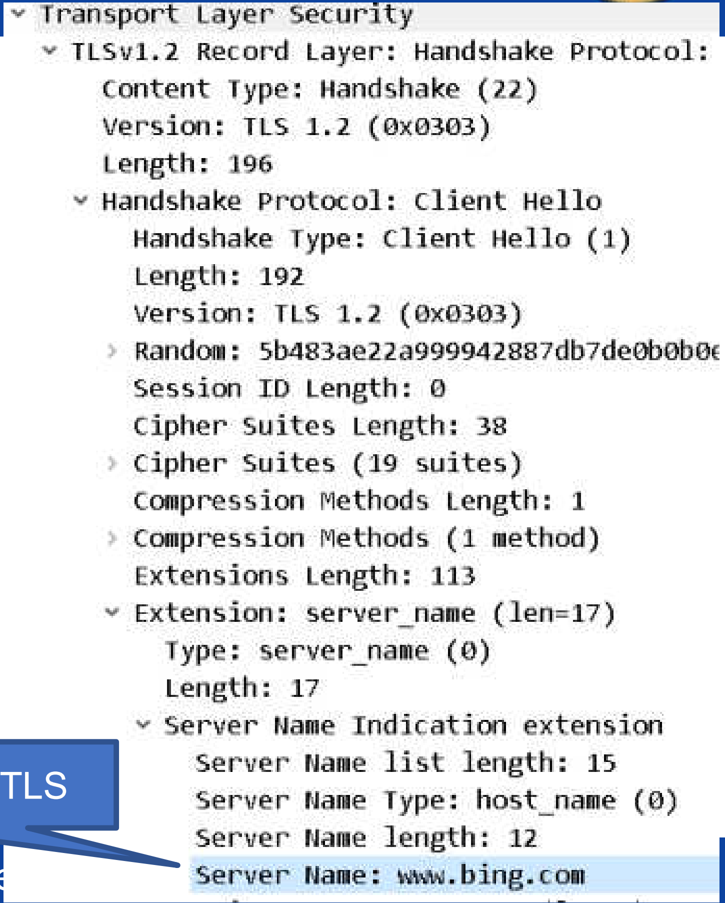


	host	count
2	www.jurassicworld.jp	112
3	www.kantei.go.jp	84
4	fuji-fc.fuji-soko.net	56
5	192.168.100.253	17
6	eigacheck.in	4
7	t.co	3
8	static.ads-twitter.com	3
9	jp.d2-apps.net	3
10	html5shiv.googlecode.com	3
11	crl.microsoft.com	3
12	cdn.d2-apps.net	3
13	secure.gravatar.com	2
14	platform.twitter.com	2
15	maxcdn.bootstrapcdn.com	2
16	fonts.googleapis.com	2
17	c01.nakanohito.jp	2
18	www.msftconnecttest.com	1
19	www.kantei.go.jp	1
20	staticxx.facebook.com	1
21	pixel.wp.com	1
22	metageek.com	1
23	fuji-fc.fuji-s_po.net	1
24	fonts.gstatic.com	1
25	files.metageek.net	1
26	cdn.treasuredata.com	1
27	192.168.100.254:49152	1

- Set delimiter as space and create a new sheet,
- copy and paste host and count rows into new sheet.
- Insert People graph and save as topwebsite.xlsx

#8 Collect fields for Visualization

- How about TLS ?
- Client Hello messages may contain host name as one of extensions (tls.handshake.type== 1)
- Server name fields locates in one of extentions in Client Hello (ssl.handshake.extensions_ server_name)



```
˅ Transport Layer Security
  ˅ TLSv1.2 Record Layer: Handshake Protocol:
      Content Type: Handshake (22)
      Version: TLS 1.2 (0x0303)
      Length: 196
    ˅ Handshake Protocol: Client Hello
        Handshake Type: Client Hello (1)
        Length: 192
        Version: TLS 1.2 (0x0303)
      › Random: 5b483ae22a999942887db7de0b0b0e
        Session ID Length: 0
        Cipher Suites Length: 38
      › Cipher Suites (19 suites)
        Compression Methods Length: 1
      › Compression Methods (1 method)
        Extensions Length: 113
      ˅ Extension: server_name (len=17)
          Type: server_name (0)
          Length: 17
        ˅ Server Name Indication extension
            Server Name list length: 15
            Server Name Type: host_name (0)
            Server Name length: 12
            Server Name: www.bing.com
```

Server Name fields in TLS

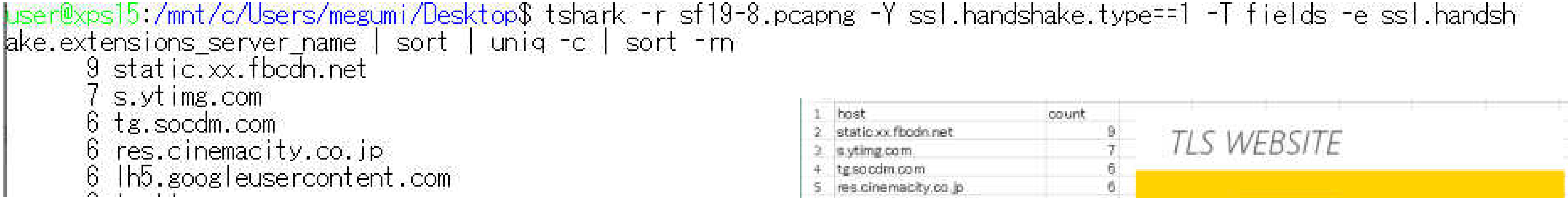
#8 Collect fields for Visualization

- Set display filter as “tls.handshake.type == 1” and collect fields of “ssl.handshake.extensions_server_name” in trace. So use the command in bash to create csv

```
tshark -r sf19-8.pcapng -Y ssl.handshake.type == 1
-T fields –e ssl.handshake.extensions_server_name
 | sort | uniq -c | sort -rn >> tlshostlist.csv
```

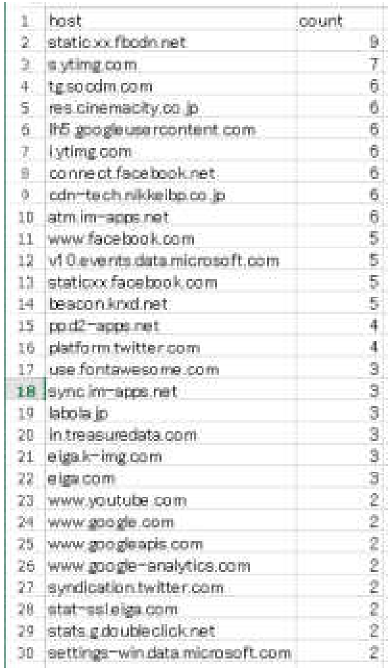
- Note: sometimes only old filter string is accepted, so we use ssl display filter word instead of tls in tshark

#8 Collect fields for Visualization

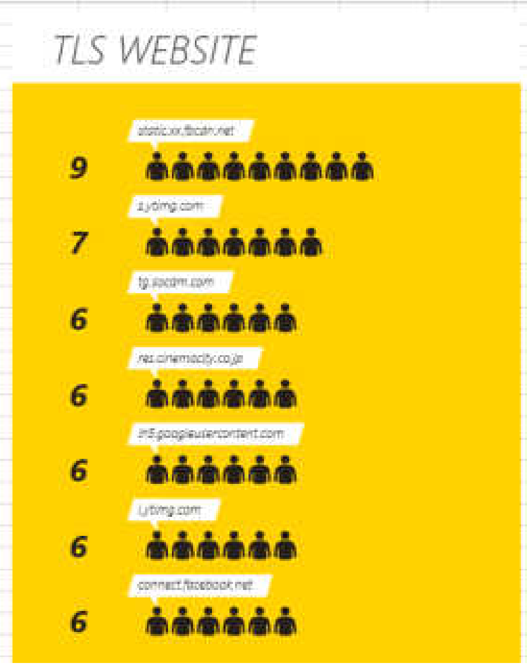


```
user@xps15:/mnt/c/Users/megumi/Desktop$ tshark -r sf19-8.pcapng -Y ssl.handshake.type==1 -T fields -e ssl.handsh
ake.extensions_server_name | sort | uniq -c | sort -rn
      9 static.xx.fbcdn.net
      7 s.ytimg.com
      6 tg.socdm.com
      6 res.cinemacity.co.jp
      6 lh5.googleusercontent.com
```

- If you got blank in server name, there is no host information in Client Hello
- Create People Graph in the same way and save file as toptlssite.xlsx

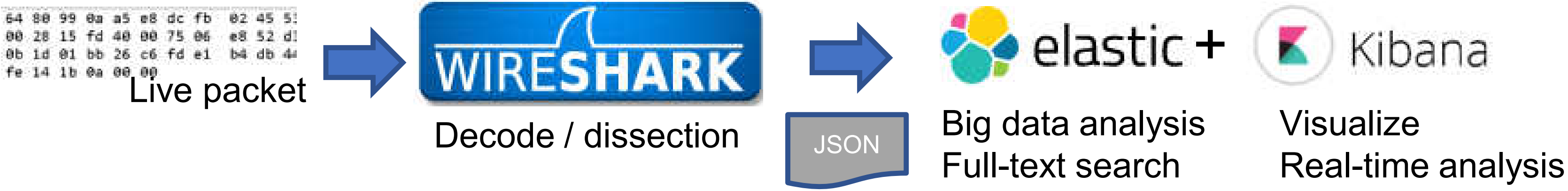


host	count
static.xx.fbcdn.net	9
s.ytimg.com	7
tg.socdm.com	6
res.cinemacity.co.jp	6
lh5.googleusercontent.com	6
i.ytimg.com	6
connect.facebook.net	6
cdn-tech.nikkeibp.co.jp	6
atm.im-apps.net	6
www.facebook.com	5
v10.events.data.microsoft.com	5
staticxx.facebook.com	5
beacon.krxd.net	5
pp.d2-apps.net	4
platform.twitter.com	4
use.fontawesome.com	3
sync.im-apps.net	3
labola.jp	3
in.treasuredata.com	3
eiga.k-img.com	3
eiga.com	3
www.youtube.com	2
www.google.com	2
www.googleapis.com	2
www.google-analytics.com	2
syndication.twitter.com	2
stat-ssl.eiga.com	2
stats.g.doubleclick.net	2
settings-win.data.microsoft.com	2



#9 Export Packet dissection to JSON

- I talked about Visualization using Elastic Search and Kibana from json file from Wireshark at Sharkfest’17


64 80 99 0a a5 e8 dc fb 02 45 5
00 28 15 fd 40 00 75 06 e8 52 d
0b 1d 01 bb 26 c6 fd e1 b4 db 4
fe 14 1b 0a 00 00
WIRESHARK
JSON
elastic +
Kibana

Live packet

Decode / dissection

Big data analysis
Full-text search

Visualize
Real-time analysis

- Wireshark 3.x / tshark now support many options to output json file from trace file and live capture.
- -T json / jsonraw / ek (Elastic search Kibana)
  and we can also use –G elastic-mapping and
  --elastic-mapping-filter <protocols> option

#9 Export Packet dissection to JSON

elastic + Kibana

Big data analysis
Full-text search

Visualize
Real-time analysis

Setup Elastic and Kibana environments

Elasticsearch

スケーラブルでRESTfulな検索・分析エンジン。

Kibana

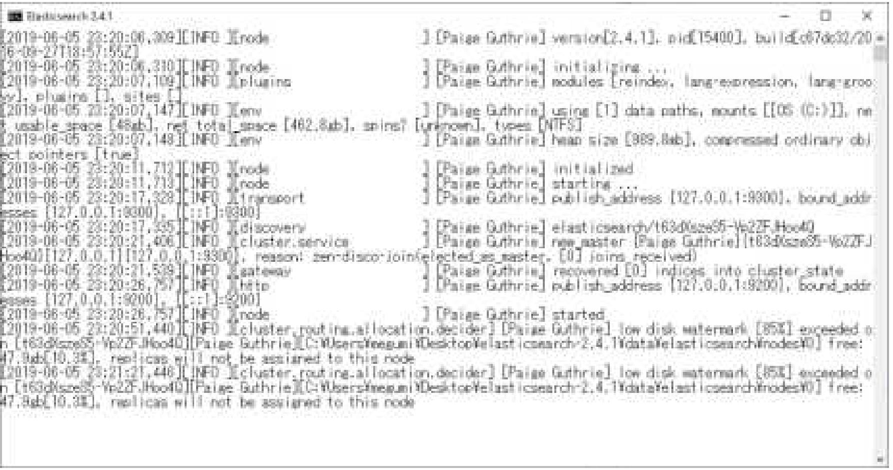
データを可視化。Stackを操作。

1.Check your machine supports Java
C:¥Users¥megumi>set | find "JAVA"
JAVA_HOME=C:¥Program Files¥Java¥jre1.8.0_212
2.Access https://www.elastic.co/jp/downloads
3.Download Elastic search, Kibana
4.Extract zip and open each bin folder
5.Execute elasticsearch.bat
6.Check “started” in command prompt
7.Open http://localhost:9200
8.Execute kibana.bat
9.Check “Kibana index ready”in prompt
10.Open http://localhost:5601

#9 Export Packet dissection to JSON

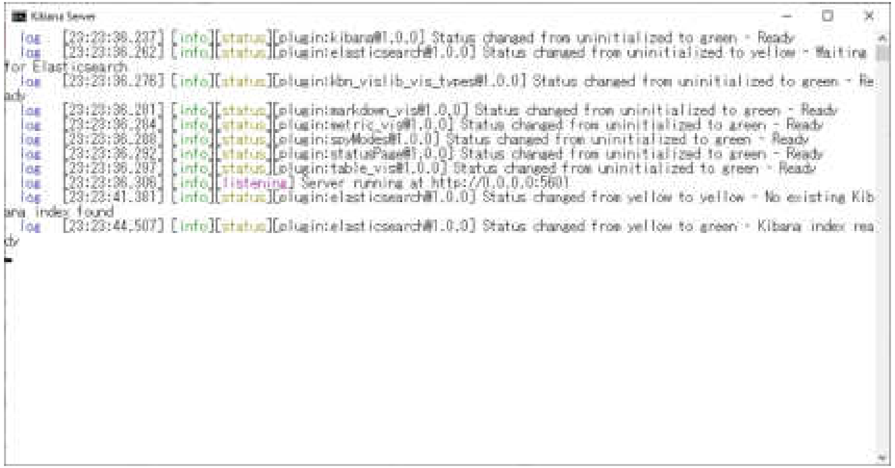
http://localhost:9200 http://localhost:5601

- This time I used old set of Elastic + Kibana elasticsearch-2.4.1 and kibana-4.6.1-windows-x86



Elasticsearch 2.4.1

```
[2019-06-05 23:20:06,309][INFO ][node                     ] [Paige Guthrie] version[2.4.1], pid[15400], build[c67dc32/20
16-09-27T18:57:55Z]
[2019-06-05 23:20:06,310][INFO ][node                     ] [Paige Guthrie] initializing ...
[2019-06-05 23:20:07,109][INFO ][plugins                  ] [Paige Guthrie] modules [reindex, lang-expression, lang-groo
vy], plugins [], sites []
[2019-06-05 23:20:07,147][INFO ][env                      ] [Paige Guthrie] using [1] data paths, mounts [[OS (C:)]], ne
t usable_space [48gb], net total_space [462.8gb], spins? [unknown], types [NTFS]
[2019-06-05 23:20:07,148][INFO ][env                      ] [Paige Guthrie] heap size [989.8mb], compressed ordinary obj
ect pointers [true]
[2019-06-05 23:20:11,712][INFO ][node                     ] [Paige Guthrie] initialized
[2019-06-05 23:20:11,713][INFO ][node                     ] [Paige Guthrie] starting ...
[2019-06-05 23:20:17,328][INFO ][transport                ] [Paige Guthrie] publish_address {127.0.0.1:9300}, bound_addr
esses {127.0.0.1:9300}, {[::1]:9300}
[2019-06-05 23:20:17,335][INFO ][discovery                ] [Paige Guthrie] elasticsearch/t63dXsze55-Vp22FJHoo4Q
[2019-06-05 23:20:21,406][INFO ][cluster.service          ] [Paige Guthrie] new_master {Paige Guthrie}{t63dXsze55-Vp22FJ
Hoo4Q}{127.0.0.1}{127.0.0.1:9300}, reason: zen-disco-join(elected_as_master, [0] joins received)
[2019-06-05 23:20:21,539][INFO ][gateway                  ] [Paige Guthrie] recovered [0] indices into cluster_state
[2019-06-05 23:20:26,757][INFO ][http                     ] [Paige Guthrie] publish_address {127.0.0.1:9200}, bound_addr
esses {127.0.0.1:9200}, {[::1]:9200}
[2019-06-05 23:20:26,757][INFO ][node                     ] [Paige Guthrie] started
[2019-06-05 23:20:51,440][INFO ][cluster.routing.allocation.decider] [Paige Guthrie] low disk watermark [85%] exceeded o
n [t63dXsze55-Vp22FJHoo4Q][Paige Guthrie][C:¥Users¥megumi¥Desktop¥elasticsearch-2.4.1¥data¥elasticsearch¥nodes¥0] free:
47.9gb[10.3%], replicas will not be assigned to this node
[2019-06-05 23:21:21,446][INFO ][cluster.routing.allocation.decider] [Paige Guthrie] low disk watermark [85%] exceeded o
n [t63dXsze55-Vp22FJHoo4Q][Paige Guthrie][C:¥Users¥megumi¥Desktop¥elasticsearch-2.4.1¥data¥elasticsearch¥nodes¥0] free:
47.9gb[10.3%], replicas will not be assigned to this node
```



Kibana Server

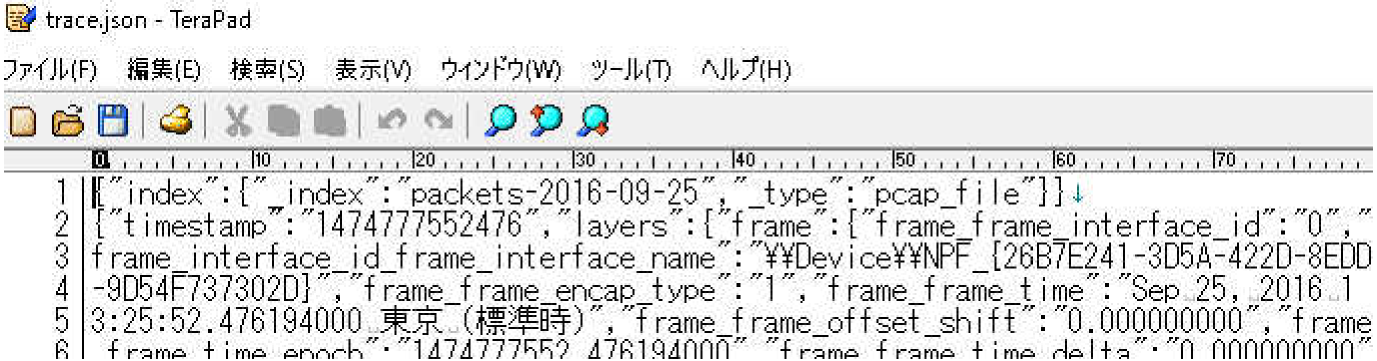
```
  log   [23:23:36.237] [info][status][plugin:kibana@1.0.0] Status changed from uninitialized to green - Ready
  log   [23:23:36.262] [info][status][plugin:elasticsearch@1.0.0] Status changed from uninitialized to yellow - Waiting
for Elasticsearch
  log   [23:23:36.276] [info][status][plugin:kbn_vislib_vis_types@1.0.0] Status changed from uninitialized to green - Re
ady
  log   [23:23:36.281] [info][status][plugin:markdown_vis@1.0.0] Status changed from uninitialized to green - Ready
  log   [23:23:36.284] [info][status][plugin:metric_vis@1.0.0] Status changed from uninitialized to green - Ready
  log   [23:23:36.288] [info][status][plugin:spyModes@1.0.0] Status changed from uninitialized to green - Ready
  log   [23:23:36.292] [info][status][plugin:statusPage@1.0.0] Status changed from uninitialized to green - Ready
  log   [23:23:36.297] [info][status][plugin:table_vis@1.0.0] Status changed from uninitialized to green - Ready
  log   [23:23:36.306] [info][listening] Server running at http://0.0.0.0:5601
  log   [23:23:41.381] [info][status][plugin:elasticsearch@1.0.0] Status changed from yellow to yellow - No existing Kib
ana index found
  log   [23:23:44.507] [info][status][plugin:elasticsearch@1.0.0] Status changed from yellow to green - Kibana index rea
dy
```

#9 Export Packet dissection to JSON

Export packet dissection to JSON (Elastic + Kibana) format from sf19-9.pcapng

tshark –r sf19-9.pcapng –T ek > trace.json

Open editor and check json file



```
{"index":{"_index":"packets-2016-09-25","_type":"pcap_file"}}↓
{"timestamp":"1474777552476","layers":{"frame":{"frame_frame_interface_id":"0","
frame_interface_id_frame_interface_name":"¥¥Device¥¥NPF_{26B7E241-3D5A-422D-8EDD
-9D54F737302D}","frame_frame_encap_type":"1","frame_frame_time":"Sep 25, 2016 1
3:25:52.476194000 東京 (標準時)","frame_frame_offset_shift":"0.000000000","frame
_frame_time_epoch":"1474777552.476194000","frame_frame_time_delta":"0.000000000"
```

#9 Export Packet dissection to JSON

OPTION

If you we want to create json file including only tcp and ip header, we can use –e tcp –e ip

tshark –r sf19-9.pcapng –T ek –e tcp –e ip

Check output to confirm the json file contains only tcp and ip header information. Also –j/-J

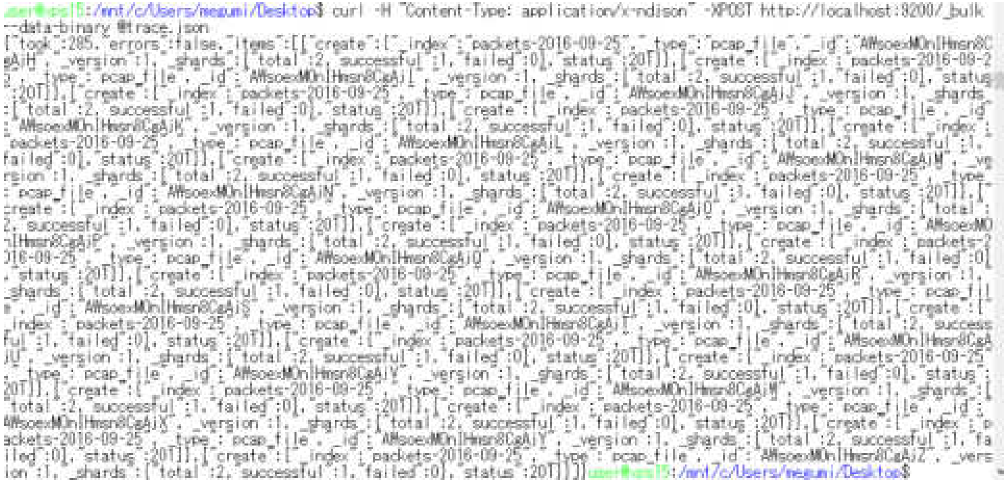
```
 -j <protocolfilter>      protocols layers filter if -T ek|pdml|json selected
                         (e.g. "ip ip.flags text", filter does not expand child
                         nodes, unless child is specified also in the filter)
  -J <protocolfilter>      top level protocol filter if -T ek|pdml|json selected
                         (e.g. "http tcp", filter which expands all child nodes)
```

#9 Export Packet dissection to JSON

Put trace.json into Elastic

curl -H "Content-Type: application/x-ndjson" -XPOST http://localhost:9200/_bulk --data-binary @trace.json

Check “successful”

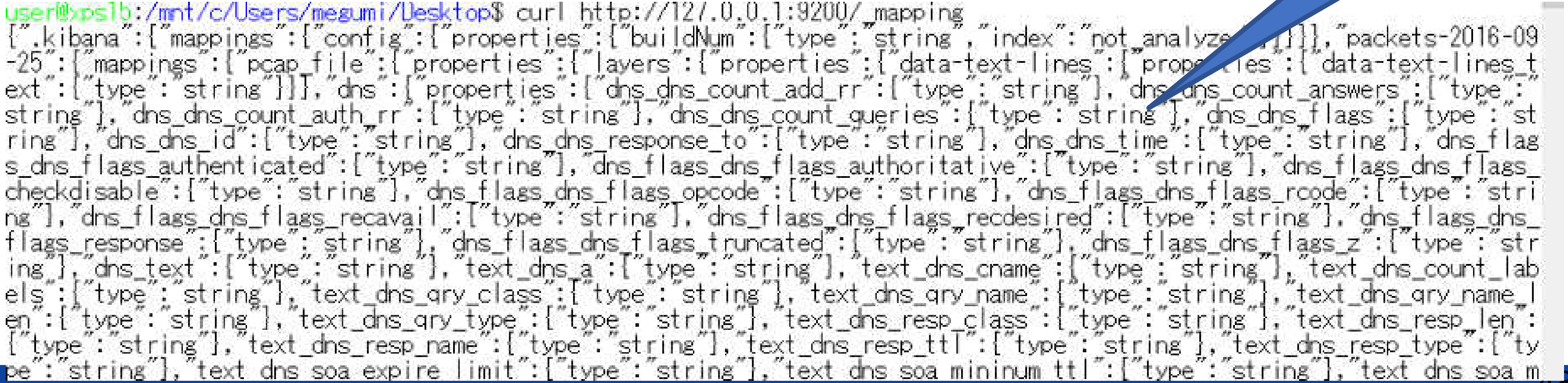


```
user@xps15:/mnt/c/Users/megumi/Desktop$ curl -H "Content-Type: application/x-ndjson" -XPOST http://localhost:9200/_bulk
--data-binary @trace.json
{"took":285,"errors":false,"items":[{"create":{"_index":"packets-2016-09-25","_type":"pcap_file","_id":"AWsoexMOnlHmsn8C
gAiH","_version":1,"_shards":{"total":2,"successful":1,"failed":0},"status":201}},{"create":{"_index":"packets-2016-09-2
5","_type":"pcap_file","_id":"AWsoexMOnlHmsn8CgAiI","_version":1,"_shards":{"total":2,"successful":1,"failed":0},"status
":201}},{"create":{"_index":"packets-2016-09-25","_type":"pcap_file","_id":"AWsoexMOnlHmsn8CgAiJ","_version":1,"_shards"
:{"total":2,"successful":1,"failed":0},"status":201}},{"create":{"_index":"packets-2016-09-25","_type":"pcap_file","_id"
:"AWsoexMOnlHmsn8CgAiK","_version":1,"_shards":{"total":2,"successful":1,"failed":0},"status":201}},{"create":{"_index":
"packets-2016-09-25","_type":"pcap_file","_id":"AWsoexMOnlHmsn8CgAiL","_version":1,"_shards":{"total":2,"successful":1,
"failed":0},"status":201}},{"create":{"_index":"packets-2016-09-25","_type":"pcap_file","_id":"AWsoexMOnlHmsn8CgAiM","_ve
rsion":1,"_shards":{"total":2,"successful":1,"failed":0},"status":201}},{"create":{"_index":"packets-2016-09-25","_type"
:"pcap_file","_id":"AWsoexMOnlHmsn8CgAiN","_version":1,"_shards":{"total":2,"successful":1,"failed":0},"status":201}},{"
create":{"_index":"packets-2016-09-25","_type":"pcap_file","_id":"AWsoexMOnlHmsn8CgAiO","_version":1,"_shards":{"total":
2,"successful":1,"failed":0},"status":201}},{"create":{"_index":"packets-2016-09-25","_type":"pcap_file","_id":"AWsoexMO
nlHmsn8CgAiP","_version":1,"_shards":{"total":2,"successful":1,"failed":0},"status":201}},{"create":{"_index":"packets-2
016-09-25","_type":"pcap_file","_id":"AWsoexMOnlHmsn8CgAiQ","_version":1,"_shards":{"total":2,"successful":1,"failed":0}
,"status":201}},{"create":{"_index":"packets-2016-09-25","_type":"pcap_file","_id":"AWsoexMOnlHmsn8CgAiR","_version":1,
"_shards":{"total":2,"successful":1,"failed":0},"status":201}},{"create":{"_index":"packets-2016-09-25","_type":"pcap_fil
e","_id":"AWsoexMOnlHmsn8CgAiS","_version":1,"_shards":{"total":2,"successful":1,"failed":0},"status":201}},{"create":{
"_index":"packets-2016-09-25","_type":"pcap_file","_id":"AWsoexMOnlHmsn8CgAiT","_version":1,"_shards":{"total":2,"success
ful":1,"failed":0},"status":201}},{"create":{"_index":"packets-2016-09-25","_type":"pcap_file","_id":"AWsoexMOnlHmsn8CgA
iU","_version":1,"_shards":{"total":2,"successful":1,"failed":0},"status":201}},{"create":{"_index":"packets-2016-09-25"
,"_type":"pcap_file","_id":"AWsoexMOnlHmsn8CgAiV","_version":1,"_shards":{"total":2,"successful":1,"failed":0},"status":
201}},{"create":{"_index":"packets-2016-09-25","_type":"pcap_file","_id":"AWsoexMOnlHmsn8CgAiW","_version":1,"_shards":{
"total":2,"successful":1,"failed":0},"status":201}},{"create":{"_index":"packets-2016-09-25","_type":"pcap_file","_id":
"AWsoexMOnlHmsn8CgAiX","_version":1,"_shards":{"total":2,"successful":1,"failed":0},"status":201}},{"create":{"_index":"p
ackets-2016-09-25","_type":"pcap_file","_id":"AWsoexMOnlHmsn8CgAiY","_version":1,"_shards":{"total":2,"successful":1,"fa
iled":0},"status":201}},{"create":{"_index":"packets-2016-09-25","_type":"pcap_file","_id":"AWsoexMOnlHmsn8CgAiZ","_vers
ion":1,"_shards":{"total":2,"successful":1,"failed":0},"status":201}}]}user@xps15:/mnt/c/Users/megumi/Desktop$
```

#9 Export Packet dissection to JSON

- We success putting json file into Elastic, but data schema (term mapping in Elastic) is not correct//
- curl http://127.0.0.1:9200/_mapping
  all fields types are recognized as “string”

All fields types are string



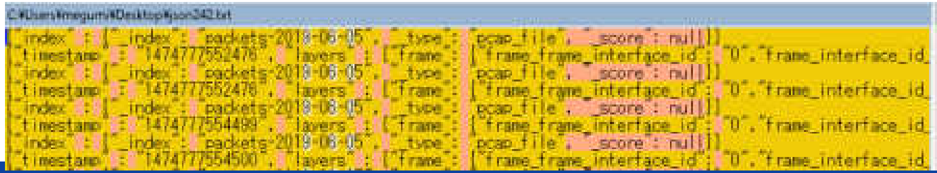
```
user@xps15:/mnt/c/Users/megumi/Desktop$ curl http://127.0.0.1:9200/_mapping
{".kibana":{"mappings":{"config":{"properties":{"buildNum":{"type":"string","index":"not_analyzed"}}}}},"packets-2016-09
-25":{"mappings":{"pcap_file":{"properties":{"layers":{"properties":{"data-text-lines":{"properties":{"data-text-lines_t
ext":{"type":"string"}}},"dns":{"properties":{"dns_dns_count_add_rr":{"type":"string"},"dns_dns_count_answers":{"type":"
string"},"dns_dns_count_auth_rr":{"type":"string"},"dns_dns_count_queries":{"type":"string"},"dns_dns_flags":{"type":"st
ring"},"dns_dns_id":{"type":"string"},"dns_dns_response_to":{"type":"string"},"dns_dns_time":{"type":"string"},"dns_flag
s_dns_flags_authenticated":{"type":"string"},"dns_flags_dns_flags_authoritative":{"type":"string"},"dns_flags_dns_flags_
checkdisable":{"type":"string"},"dns_flags_dns_flags_opcode":{"type":"string"},"dns_flags_dns_flags_rcode":{"type":"stri
ng"},"dns_flags_dns_flags_recavail":{"type":"string"},"dns_flags_dns_flags_recdesired":{"type":"string"},"dns_flags_dns_
flags_response":{"type":"string"},"dns_flags_dns_flags_truncated":{"type":"string"},"dns_flags_dns_flags_z":{"type":"str
ing"},"dns_text":{"type":"string"},"text_dns_a":{"type":"string"},"text_dns_cname":{"type":"string"},"text_dns_count_lab
els":{"type":"string"},"text_dns_qry_class":{"type":"string"},"text_dns_qry_name":{"type":"string"},"text_dns_qry_name_l
en":{"type":"string"},"text_dns_qry_type":{"type":"string"},"text_dns_resp_class":{"type":"string"},"text_dns_resp_len":
{"type":"string"},"text_dns_resp_name":{"type":"string"},"text_dns_resp_ttl":{"type":"string"},"text_dns_resp_type":{"ty
pe":"string"},"text_dns_soa_expire_limit":{"type":"string"},"text_dns_soa_mininum_ttl":{"type":"string"},"text_dns_soa_m
```

#9 Export Packet dissection to JSON

- When you create json file using tshark / Wireshark, there are problems about mismatch of database schema (a.k.a. “mapping” in Elastic)
  When you upgrade Wireshark and some protocol dissector is updated or modified,
  the output json file format may be changed.

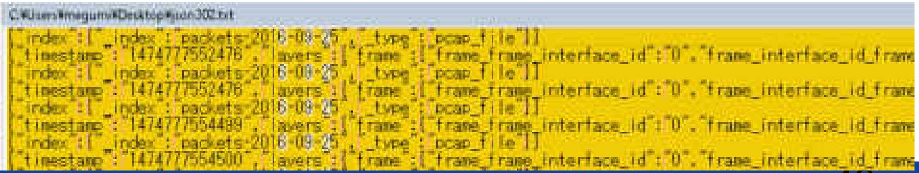
```
C:¥Users¥megumi¥Desktop>tshark --version
TShark (Wireshark) 2.4.2 (v2.4.2-0-gb6c63ae086)
tshark -T ek -r stream.pcapng >> json242.txt
```

```
C:¥Users¥megumi¥Desktop>tshark --version
TShark (Wireshark) 3.0.2 (v3.0.2-0-g621ed351d5c9)
tshark -T ek -r stream.pcapng >> json302.txt
```



C:¥Users¥megumi¥Desktop¥json242.txt

```
{"index" : {"_index" : "packets-2019-06-05", "_type" : "pcap_file", "_score" : null}}
{"timestamp" : "1474777552476", "layers" : {"frame" : {"frame_frame_interface_id" : "0","frame_interface_id_
{"index" : {"_index" : "packets-2019-06-05", "_type" : "pcap_file", "_score" : null}}
{"timestamp" : "1474777552476", "layers" : {"frame" : {"frame_frame_interface_id" : "0","frame_interface_id_
{"index" : {"_index" : "packets-2019-06-05", "_type" : "pcap_file", "_score" : null}}
{"timestamp" : "1474777554499", "layers" : {"frame" : {"frame_frame_interface_id" : "0","frame_interface_id_
{"index" : {"_index" : "packets-2019-06-05", "_type" : "pcap_file", "_score" : null}}
{"timestamp" : "1474777554500", "layers" : {"frame" : {"frame_frame_interface_id" : "0","frame_interface_id_
```



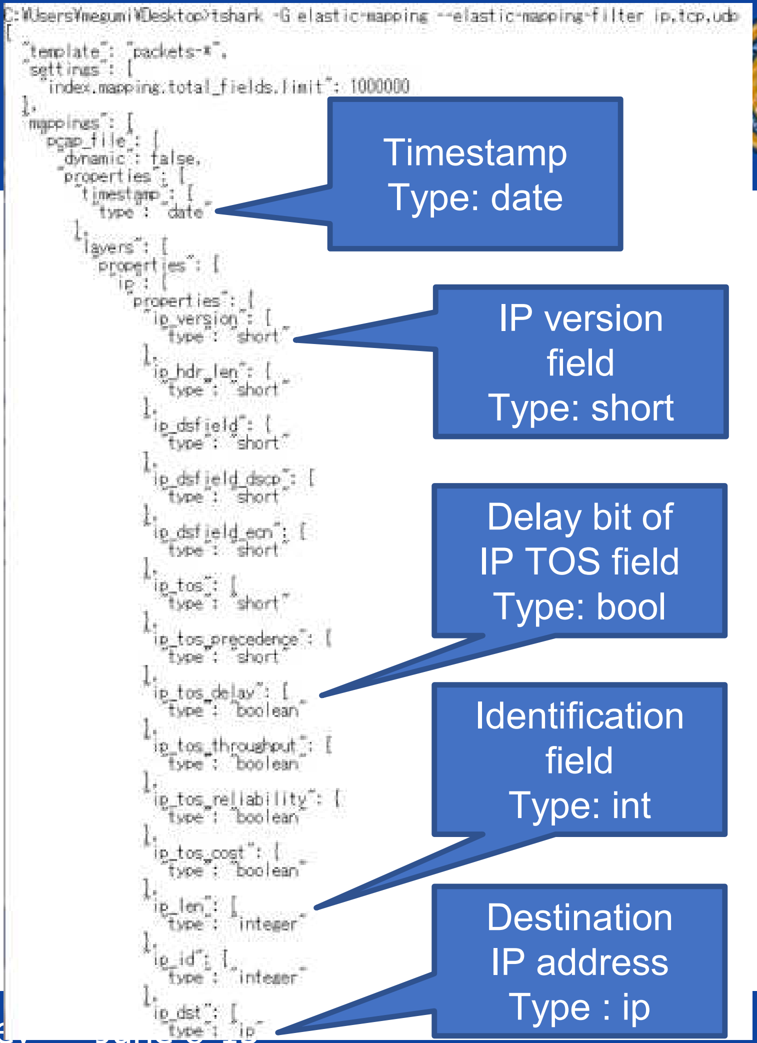
C:¥Users¥megumi¥Desktop¥json302.txt

```
{"index":{"_index":"packets-2016-09-25","_type":"pcap_file"}}
{"timestamp":"1474777552476","layers":{"frame":{"frame_frame_interface_id":"0","frame_interface_id_frame
{"index":{"_index":"packets-2016-09-25","_type":"pcap_file"}}
{"timestamp":"1474777552476","layers":{"frame":{"frame_frame_interface_id":"0","frame_interface_id_frame
{"index":{"_index":"packets-2016-09-25","_type":"pcap_file"}}
{"timestamp":"1474777554499","layers":{"frame":{"frame_frame_interface_id":"0","frame_interface_id_frame
{"index":{"_index":"packets-2016-09-25","_type":"pcap_file"}}
{"timestamp":"1474777554500","layers":{"frame":{"frame_frame_interface_id":"0","frame_interface_id_frame
```

#9 Export Packet dissection to JSON

- We can create adequate Elastic mapping file semi-automatically using tshark
- If we want to create flow based schema information including ip, tcp and udp

tshark -G elastic-mapping --elastic-mapping-filter ip,tcp,udp > mapping.json



```
C:¥Users¥megumi¥Desktop>tshark -G elastic-mapping --elastic-mapping-filter ip,tcp,udp
{
  "template": "packets-*",
  "settings": {
    "index.mapping.total_fields.limit": 1000000
  },
  "mappings": {
    "pcap_file": {
      "dynamic": false,
      "properties": {
        "timestamp": {
          "type": "date"
        },
        "layers": {
          "properties": {
            "ip": {
              "properties": {
                "ip_version": {
                  "type": "short"
                },
                "ip_hdr_len": {
                  "type": "short"
                },
                "ip_dsfield": {
                  "type": "short"
                },
                "ip_dsfield_dscp": {
                  "type": "short"
                },
                "ip_dsfield_ecn": {
                  "type": "short"
                },
                "ip_tos": {
                  "type": "short"
                },
                "ip_tos_precedence": {
                  "type": "short"
                },
                "ip_tos_delay": {
                  "type": "boolean"
                },
                "ip_tos_throughput": {
                  "type": "boolean"
                },
                "ip_tos_reliability": {
                  "type": "boolean"
                },
                "ip_tos_cost": {
                  "type": "boolean"
                },
                "ip_len": {
                  "type": "integer"
                },
                "ip_id": {
                  "type": "integer"
                },
                "ip_dst": {
                  "type": "ip"
```

Timestamp
Type: date

IP version field
Type: short

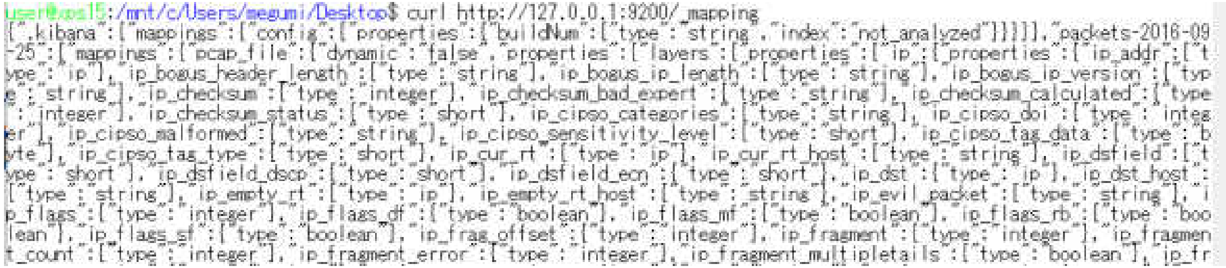
Delay bit of IP TOS field
Type: bool

Identification field
Type: int

Destination IP address
Type : ip

#9 Export Packet dissection to JSON

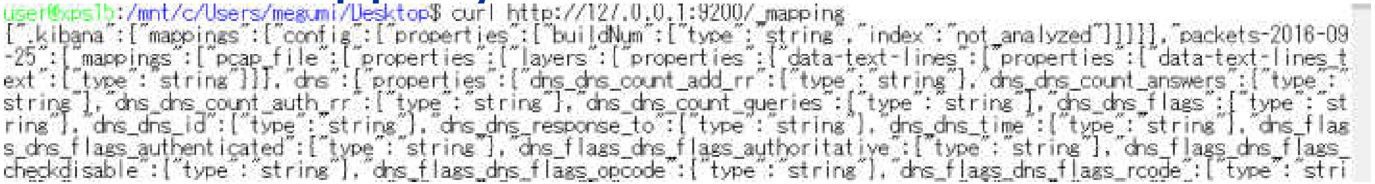
- We need to delete all data and schema
  curl -XDELETE http://localhost:9200/*
- Then put mapping information into Elastic
  curl -H "Content-Type: application/x-ndjson" -XPOST http://localhost:9200/packets-2016-09-25 --data-binary @mapping.json
- Check mapping “curl http://127.0.0.1:9200/_mapping”



```
user@xps15:/mnt/c/Users/megumi/Desktop$ curl http://127.0.0.1:9200/_mapping
{".kibana":{"mappings":{"config":{"properties":{"buildNum":{"type":"string","index":"not_analyzed"}}}}},"packets-2016-09
-25":{"mappings":{"pcap_file":{"dynamic":"false","properties":{"layers":{"properties":{"ip":{"properties":{"ip_addr":{"t
ype":"ip"},"ip_bogus_header_length":{"type":"string"},"ip_bogus_ip_length":{"type":"string"},"ip_bogus_ip_version":{"typ
e":"string"},"ip_checksum":{"type":"integer"},"ip_checksum_bad_expert":{"type":"string"},"ip_checksum_calculated":{"type
":"integer"},"ip_checksum_status":{"type":"short"},"ip_cipso_categories":{"type":"string"},"ip_cipso_doi":{"type":"integ
er"},"ip_cipso_malformed":{"type":"string"},"ip_cipso_sensitivity_level":{"type":"short"},"ip_cipso_tag_data":{"type":"b
yte"},"ip_cipso_tag_type":{"type":"short"},"ip_cur_rt":{"type":"ip"},"ip_cur_rt_host":{"type":"string"},"ip_dsfield":{"t
ype":"short"},"ip_dsfield_dscp":{"type":"short"},"ip_dsfield_ecn":{"type":"short"},"ip_dst":{"type":"ip"},"ip_dst_host":
{"type":"string"},"ip_empty_rt":{"type":"ip"},"ip_empty_rt_host":{"type":"string"},"ip_evil_packet":{"type":"string"},"i
p_flags":{"type":"integer"},"ip_flags_df":{"type":"boolean"},"ip_flags_mf":{"type":"boolean"},"ip_flags_rb":{"type":"boo
lean"},"ip_flags_sf":{"type":"boolean"},"ip_frag_offset":{"type":"integer"},"ip_fragment":{"type":"integer"},"ip_fragmen
t_count":{"type":"integer"},"ip_fragment_error":{"type":"integer"},"ip_fragment_multipletails":{"type":"boolean"},"ip_fr
```

#9 Export Packet dissection to JSON

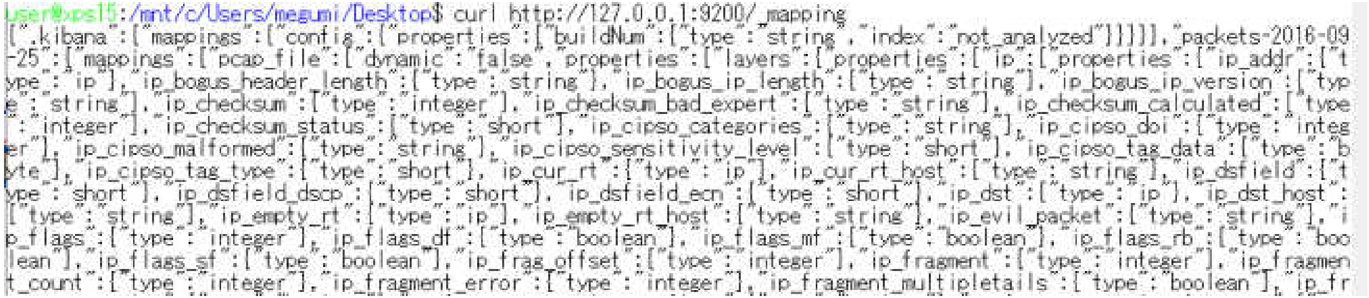
- Without mapping file



```
user@xps15:/mnt/c/Users/megumi/Desktop$ curl http://127.0.0.1:9200/_mapping
{".kibana":{"mappings":{"config":{"properties":{"buildNum":{"type":"string","index":"not_analyzed"}}}}}},"packets-2016-09
-25":{"mappings":{"pcap_file":{"properties":{"layers":{"properties":{"data-text-lines":{"properties":{"data-text-lines_t
ext":{"type":"string"}}},"dns":{"properties":{"dns_dns_count_add_rr":{"type":"string"},"dns_dns_count_answers":{"type":"
string"},"dns_dns_count_auth_rr":{"type":"string"},"dns_dns_count_queries":{"type":"string"},"dns_dns_flags":{"type":"st
ring"},"dns_dns_id":{"type":"string"},"dns_dns_response_to":{"type":"string"},"dns_dns_time":{"type":"string"},"dns_flag
s_dns_flags_authenticated":{"type":"string"},"dns_flags_dns_flags_authoritative":{"type":"string"},"dns_flags_dns_flags_
checkdisable":{"type":"string"},"dns_flags_dns_flags_opcode":{"type":"string"},"dns_flags_dns_flags_rcode":{"type":"stri
```

- With mapping file “mapping.json”

curl -H "Content-Type: application/x-ndjson" -XPOST http://localhost:9200/packets-2016-09-25 --data-binary @mapping.json



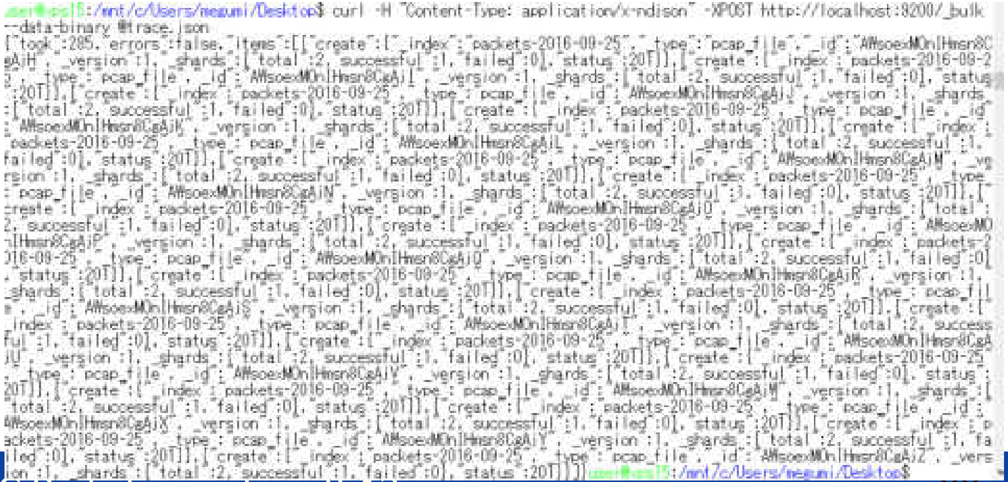
```
user@xps15:/mnt/c/Users/megumi/Desktop$ curl http://127.0.0.1:9200/_mapping
{".kibana":{"mappings":{"config":{"properties":{"buildNum":{"type":"string","index":"not_analyzed"}}}}}},"packets-2016-09
-25":{"mappings":{"pcap_file":{"dynamic":"false","properties":{"layers":{"properties":{"ip":{"properties":{"ip_addr":{"t
ype":"ip"},"ip_bogus_header_length":{"type":"string"},"ip_bogus_ip_length":{"type":"string"},"ip_bogus_ip_version":{"typ
e":"string"},"ip_checksum":{"type":"integer"},"ip_checksum_bad_expert":{"type":"string"},"ip_checksum_calculated":{"type
":"integer"},"ip_checksum_status":{"type":"short"},"ip_cipso_categories":{"type":"string"},"ip_cipso_doi":{"type":"integ
er"},"ip_cipso_malformed":{"type":"string"},"ip_cipso_sensitivity_level":{"type":"short"},"ip_cipso_tag_data":{"type":"b
yte"},"ip_cipso_tag_type":{"type":"short"},"ip_cur_rt":{"type":"ip"},"ip_cur_rt_host":{"type":"string"},"ip_dsfield":{"t
ype":"short"},"ip_dsfield_dscp":{"type":"short"},"ip_dsfield_ecn":{"type":"short"},"ip_dst":{"type":"ip"},"ip_dst_host":
{"type":"string"},"ip_empty_rt":{"type":"ip"},"ip_empty_rt_host":{"type":"string"},"ip_evil_packet":{"type":"string"},"i
p_flags":{"type":"integer"},"ip_flags_df":{"type":"boolean"},"ip_flags_mf":{"type":"boolean"},"ip_flags_rb":{"type":"boo
lean"},"ip_flags_sf":{"type":"boolean"},"ip_frag_offset":{"type":"integer"},"ip_fragment":{"type":"integer"},"ip_fragmen
t_count":{"type":"integer"},"ip_fragment_error":{"type":"integer"},"ip_fragment_multipletails":{"type":"boolean"},"ip_fr
```

#9 Export Packet dissection to JSON

Put trace.json again into Elastic

curl -H "Content-Type: application/x-ndjson" -XPOST http://localhost:9200/_bulk --data-binary @trace.json

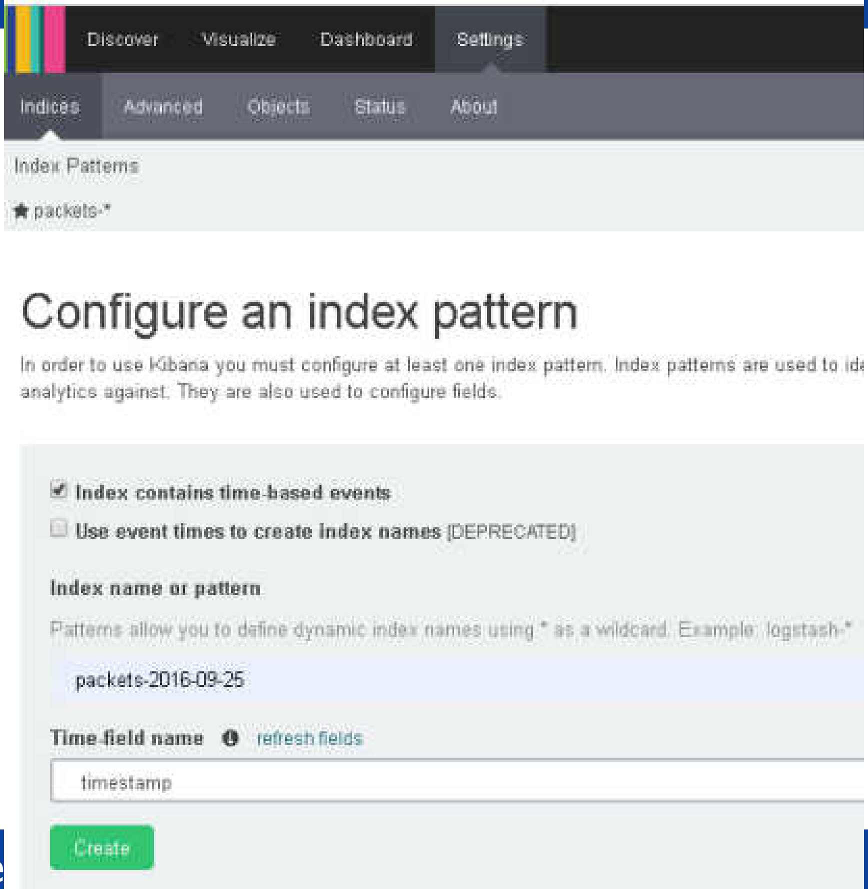
Check “successful”



```
user@pc15:/mnt/c/Users/megumi/Desktop$ curl -H "Content-Type: application/x-ndjson" -XPOST http://localhost:9200/_bulk
--data-binary @trace.json
{"took":285,"errors":false,"items":[{"create":{"_index":"packets-2016-09-25","_type":"pcap_file","_id":"AWsoexMOnIHmsn8C
gAiH","_version":1,"_shards":{"total":2,"successful":1,"failed":0},"status":201}},{"create":{"_index":"packets-2016-09-2
5","_type":"pcap_file","_id":"AWsoexMOnIHmsn8CgAiI","_version":1,"_shards":{"total":2,"successful":1,"failed":0},"status
":201}},{"create":{"_index":"packets-2016-09-25","_type":"pcap_file","_id":"AWsoexMOnIHmsn8CgAiJ","_version":1,"_shards"
:{"total":2,"successful":1,"failed":0},"status":201}},{"create":{"_index":"packets-2016-09-25","_type":"pcap_file","_id"
:"AWsoexMOnIHmsn8CgAiK","_version":1,"_shards":{"total":2,"successful":1,"failed":0},"status":201}},{"create":{"_index":
"packets-2016-09-25","_type":"pcap_file","_id":"AWsoexMOnIHmsn8CgAiL","_version":1,"_shards":{"total":2,"successful":1,
"failed":0},"status":201}},{"create":{"_index":"packets-2016-09-25","_type":"pcap_file","_id":"AWsoexMOnIHmsn8CgAiM","_ve
rsion":1,"_shards":{"total":2,"successful":1,"failed":0},"status":201}},{"create":{"_index":"packets-2016-09-25","_type"
:"pcap_file","_id":"AWsoexMOnIHmsn8CgAiN","_version":1,"_shards":{"total":2,"successful":1,"failed":0},"status":201}},{"
create":{"_index":"packets-2016-09-25","_type":"pcap_file","_id":"AWsoexMOnIHmsn8CgAiO","_version":1,"_shards":{"total":
2,"successful":1,"failed":0},"status":201}},{"create":{"_index":"packets-2016-09-25","_type":"pcap_file","_id":"AWsoexMO
nIHmsn8CgAiP","_version":1,"_shards":{"total":2,"successful":1,"failed":0},"status":201}},{"create":{"_index":"packets-2
016-09-25","_type":"pcap_file","_id":"AWsoexMOnIHmsn8CgAiQ","_version":1,"_shards":{"total":2,"successful":1,"failed":0
},"status":201}},{"create":{"_index":"packets-2016-09-25","_type":"pcap_file","_id":"AWsoexMOnIHmsn8CgAiR","_version":1,
"_shards":{"total":2,"successful":1,"failed":0},"status":201}},{"create":{"_index":"packets-2016-09-25","_type":"pcap_fil
e","_id":"AWsoexMOnIHmsn8CgAiS","_version":1,"_shards":{"total":2,"successful":1,"failed":0},"status":201}},{"create":{"
_index":"packets-2016-09-25","_type":"pcap_file","_id":"AWsoexMOnIHmsn8CgAiT","_version":1,"_shards":{"total":2,"success
ful":1,"failed":0},"status":201}},{"create":{"_index":"packets-2016-09-25","_type":"pcap_file","_id":"AWsoexMOnIHmsn8CgA
iU","_version":1,"_shards":{"total":2,"successful":1,"failed":0},"status":201}},{"create":{"_index":"packets-2016-09-25"
,"_type":"pcap_file","_id":"AWsoexMOnIHmsn8CgAiV","_version":1,"_shards":{"total":2,"successful":1,"failed":0},"status":
201}},{"create":{"_index":"packets-2016-09-25","_type":"pcap_file","_id":"AWsoexMOnIHmsn8CgAiW","_version":1,"_shards":{
"total":2,"successful":1,"failed":0},"status":201}},{"create":{"_index":"packets-2016-09-25","_type":"pcap_file","_id":
"AWsoexMOnIHmsn8CgAiX","_version":1,"_shards":{"total":2,"successful":1,"failed":0},"status":201}},{"create":{"_index":"p
ackets-2016-09-25","_type":"pcap_file","_id":"AWsoexMOnIHmsn8CgAiY","_version":1,"_shards":{"total":2,"successful":1,"fa
iled":0},"status":201}},{"create":{"_index":"packets-2016-09-25","_type":"pcap_file","_id":"AWsoexMOnIHmsn8CgAiZ","_vers
ion":1,"_shards":{"total":2,"successful":1,"failed":0},"status":201}}]}user@pc15:/mnt/c/Users/megumi/Desktop$
```

#9 Export Packet dissection to JSON

- Its time to use Kibana ! http://localhost:5601
- Set index pattern as packets-2016-09-25 (may work packets-*)
- Set Time-filed name as timestamp (type:date)
- Click “Create”



Configure an index pattern

In order to use Kibana you must configure at least one index pattern. Index patterns are used to ide analytics against. They are also used to configure fields.

☑ **Index contains time-based events**

□ **Use event times to create index names** [DEPRECATED]

Index name or pattern

Patterns allow you to define dynamic index names using * as a wildcard. Example: logstash-*

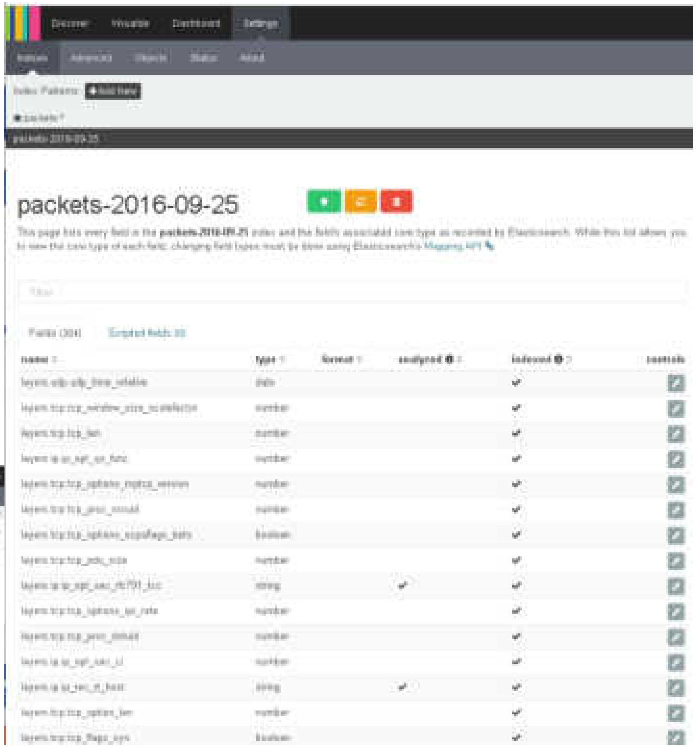
packets-2016-09-25

Time-field name refresh fields

timestamp

#9 Export Packet dissection to JSON

- Check mapping is correctly assigned as adequate type.
  layers.udp.udp_time_relative as date
  layers.tcp.tcp_window_size_scalefactor as number
  layers.tcp.tcp_options_scpsflags_bets as bool
  etc.
- Enjoy Visualization

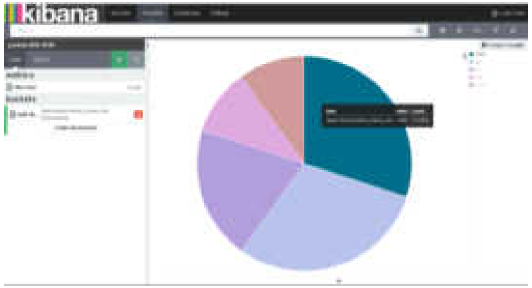


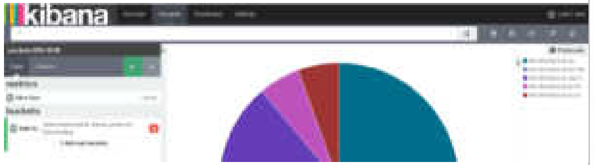
packets-2016-09-25

This page lists every field in the **packets-2016-09-25** index and the field's associated core type as recorded by Elasticsearch. While this list allows you to view the core type of each field, changing field types must be done using Elasticsearch's Mapping API

Fields (304) Scripted fields (0)

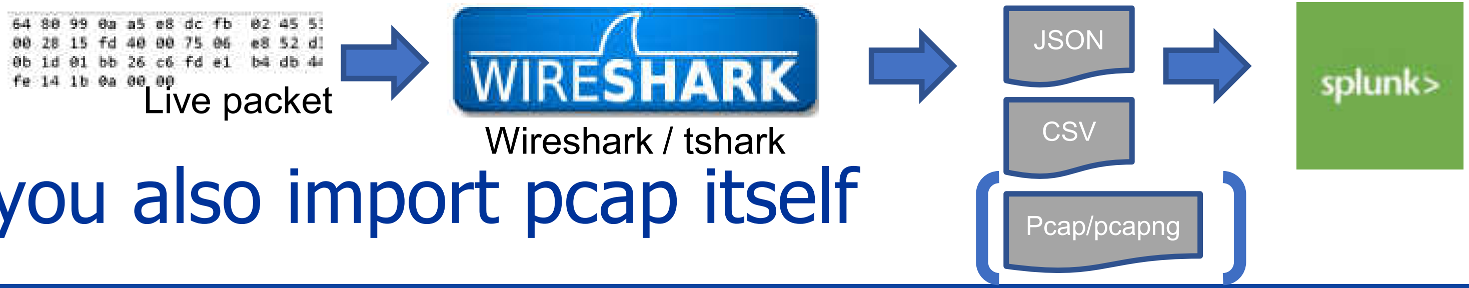
name	type	format	analyzed	indexed	controls
layers.udp.udp_time_relative	date			✓	✎
layers.tcp.tcp_window_size_scalefactor	number			✓	✎
layers.tcp.tcp_len	number			✓	✎
layers.ip.ip_opt_qs_func	number			✓	✎
layers.tcp.tcp_options_mptcp_version	number			✓	✎
layers.tcp.tcp_[illegible]	number			✓	✎
layers.tcp.tcp_options_scpsflags_bets	boolean			✓	✎
layers.tcp.tcp_pdu_size	number			✓	✎
layers.ip.ip_opt_sec_rfc791_tcc	string		✓	✓	✎
layers.tcp.tcp_options_qs_rate	number			✓	✎
layers.tcp.tcp_[illegible]	number			✓	✎
layers.ip.ip_opt_sec_cl	number			✓	✎
layers.ip.ip_[illegible]	string		✓	✓	✎
layers.tcp.tcp_option_len	number			✓	✎
layers.tcp.tcp_flags_syn	boolean			✓	✎





#10 Splunk

- Splunk is one of big data processing tools for visualizing trace files via CSV or JSON https://splunkbase.splunk.com/app/2748/
- We can use free if the data size is under 500MB in Windows / Linux / macOS environments
- There are two major way to convert pcap/pcapng

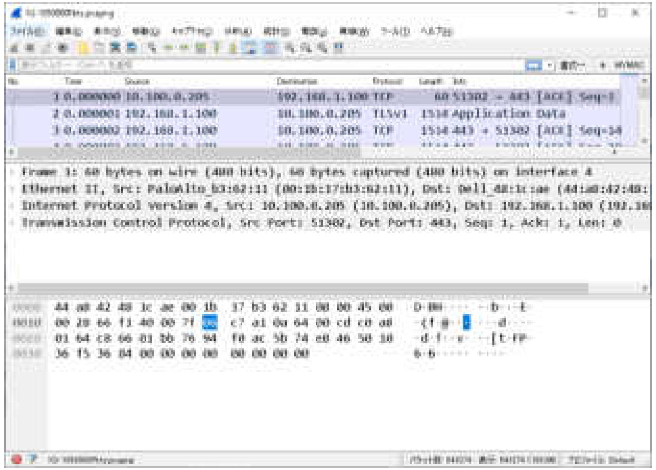


```
64 80 99 0a a5 e8 dc fb  02 45 5
00 28 15 fd 40 00 75 06  e8 52 d
0b 1d 01 bb 26 c6 fd e1  b4 db 4
fe 14 1b 0a 00 00
```

you also import pcap itself

#10 Splunk

- There are sample trace files including huge packets. (https://www.bettydubois.com/sharkfest19)
- I use 1G trace (1G-1050000Pkts.pcapng) that contains about 1 million packets
- Open the file in Wireshark (recommend with ReadFilter or light profile for huge file) and Export Packet Dissections to export CSV which contains just a packet summary information)

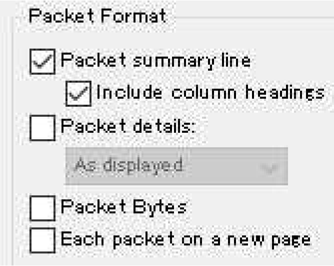


```
No.  Time     Source          Destination     Protocol Length Info
  1 0.000000 10.100.0.205    192.168.1.100   TCP        60 51302 → 443 [ACK] Seq=1
  2 0.000001 192.168.1.100   10.100.0.205    TLSv1    1514 Application Data
  3 0.000002 192.168.1.100   10.100.0.205    TCP      1514 443 → 51302 [ACK] Seq=14
```

```
> Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 4
> Ethernet II, Src: PaloAlto_b3:62:11 (00:1b:17:b3:62:11), Dst: Dell_48:1c:ae (44:a8:42:48:
> Internet Protocol Version 4, Src: 10.100.0.205 (10.100.0.205), Dst: 192.168.1.100 (192.16
> Transmission Control Protocol, Src Port: 51302, Dst Port: 443, Seq: 1, Ack: 1, Len: 0
```

```
0000  44 a8 42 48 1c ae 00 1b  17 b3 62 11 08 00 45 00   D·BH······b···E·
0010  00 28 66 f1 40 00 7f 06  c7 a1 0a 64 00 cd c0 a8   ·(f·@·······d····
0020  01 64 c8 66 01 bb 76 94  f0 ac 5b 74 e0 46 50 10   ·d·f··v···[t·FP·
0030  36 f5 36 04 00 00 00 00  00 00 00 00               6·6·········
```

#10 Splunk



Packet Format

☑ Packet summary line

☑ Include column headings

☐ Packet details:

As displayed

☐ Packet Bytes

☐ Each packet on a new page

In this case, we use the default information of packet summary pane, such as Numbers, Source, Destination, Protocol, Length and Info
Though you can off course customize them

Using tshark is also a good way to handle big trace files,

```
tshark -r 1G-1050000Pkts.pcapng -T text
>> 1G-1050000Pkts.csv
```

or you can use –T json for your customized dissector fields information (with –e or –j or –J options)

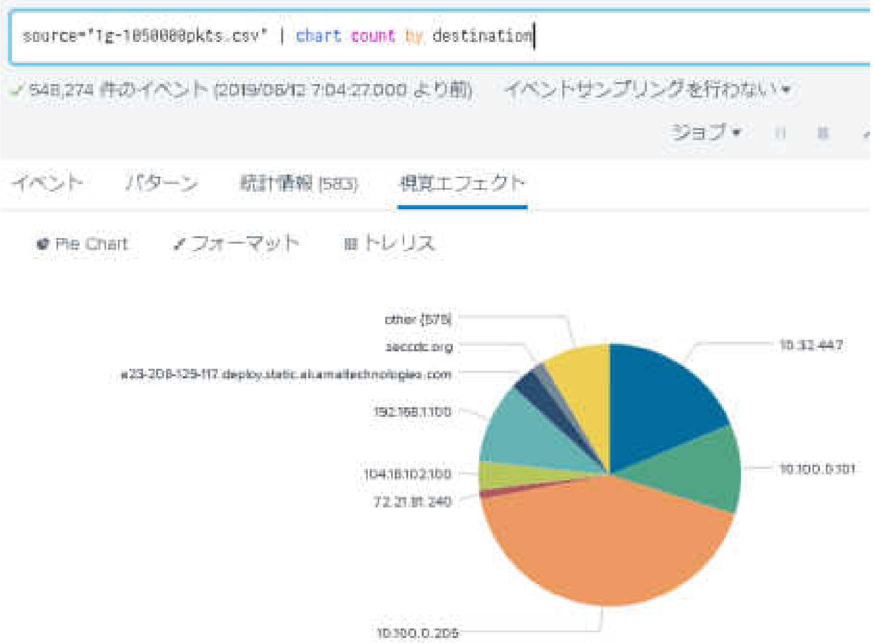
#10 Splunk

- So input 1G-1050000Pkts.csv into Splunk, set fields name and indexes are created automatically
- I'll not talk about Splunk in detail , there are tons of documents and samples you can refer
- Open splunk page and login (http://localhost:8000/)

1. Click [Search and Reporting] in Left pane
2. Choose time range as all terms
3. Type "source="1g-1050000pkts.csv" | chart count by destination and set style as pie chart

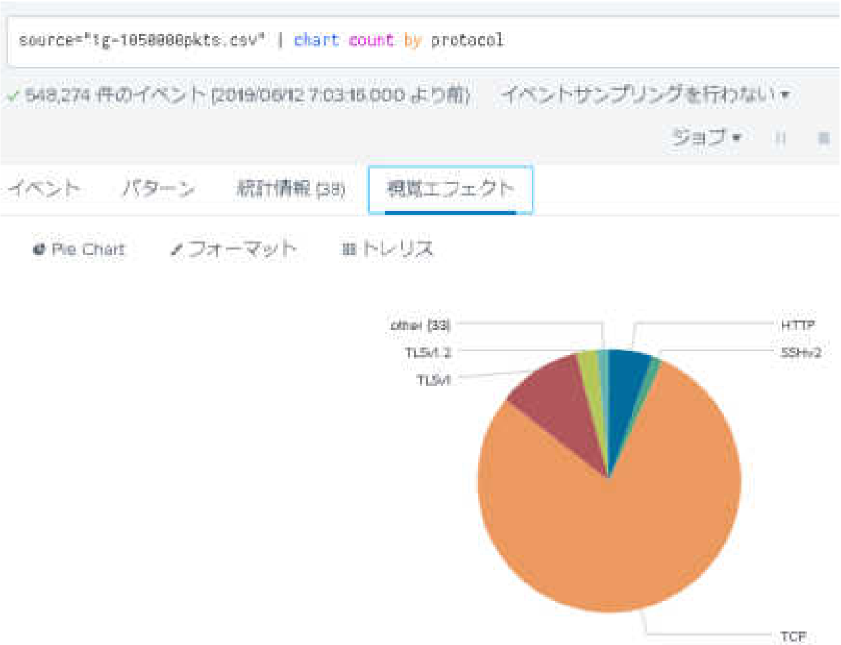
#10 Splunk

source="1g-1050000pkts.csv" | chart count by destination



```
source="1g-1050000pkts.csv" | chart count by destination
```

✓ 548,274 件のイベント (2019/06/12 7:04:27.000 より前)　イベントサンプリングを行わない ▾



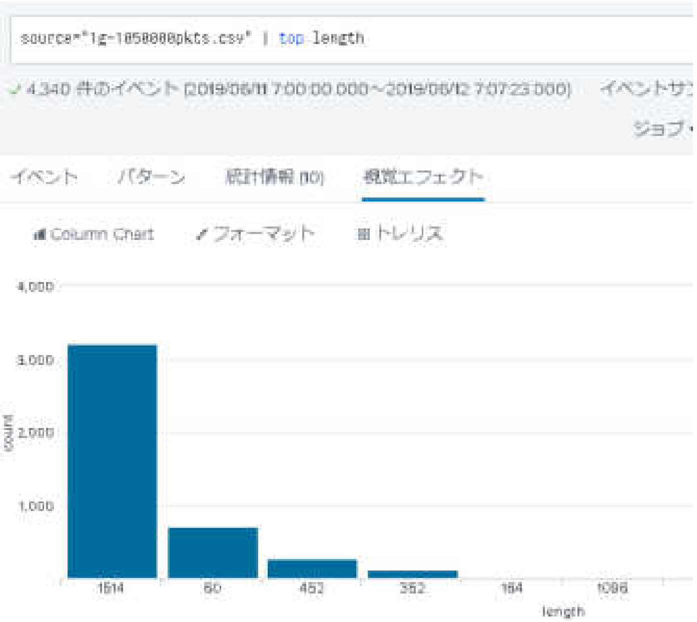
```
source="1g-1050000pkts.csv" | chart count by protocol
```

✓ 548,274 件のイベント (2019/06/12 7:03:16.000 より前)　イベントサンプリングを行わない ▾

source="1g-1050000pkts.csv" | chart count by protocol

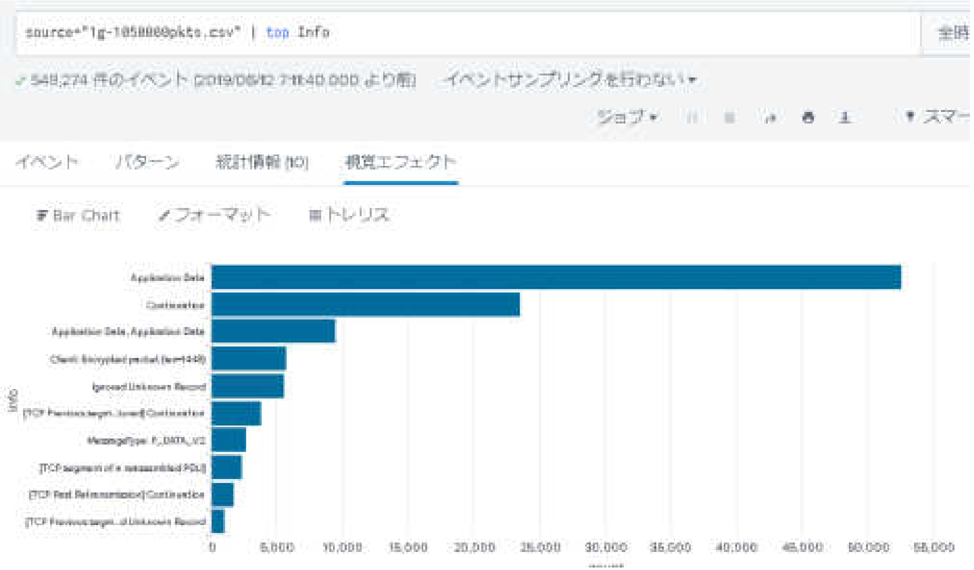
#10 Splunk

source="1g-1050000pkts.csv" | top length



```
source="1g-1050000pkts.csv" | top length
```

✓ 4,340 件のイベント (2019/06/11 7:00:00.000～2019/06/12 7:07:23.000)



```
source="1g-1050000pkts.csv" | top Info
```

✓ 548,274 件のイベント (2019/06/12 7:11:40.000 より前)

全時
イベントサンプリングを行わない ▾
ジョブ ▾
スマー
イベント
パターン
統計情報 (10)
視覚エフェクト
Bar Chart
フォーマット
トレリス
Application Data
Continuation
Application Data, Application Data
Client Encrypted packet (len=1448)
Ignored Unknown Record
[TCP Previous segm...ured] Continuation
MessageType: P_DATA_V2
[TCP segment of a reassembled PDU]
[TCP Fast Retransmission] Continuation
[TCP Previous segm...d Unknown Record
Info
0
5,000
10,000
15,000
20,000
25,000
30,000
35,000
40,000
45,000
50,000
55,000
count

source="1g-1050000pkts.csv" | top Info

Think Visually

- You have finished visualization of trace file, then its turn to think visually.
- Stop looking each frame in detail, Look over the traffic visually.
- You may find a new clue which you have never found !!
- USE WIRESHARK and THINK VISUALLY

Thank you for attending !!

Please complete the SharkFest'19 US app-based survey

Supplemental file

http://www.ikeriri.ne.jp/sharkfest

ikeriri network service

http://www.ikeriri.ne.jp