SharkFest '18 US

Point and shoot packet

Shooting point of field in packet analysis

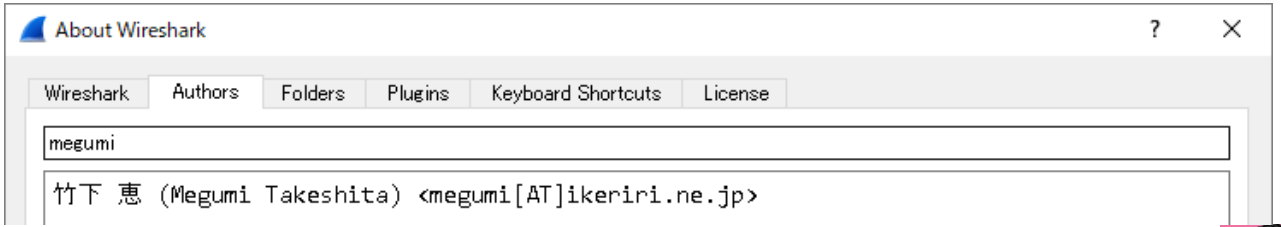
Supplemental trace files are
http://www.ikeriri.ne.jp/sharkfest/

Megumi Takeshita

Packet Otaku
ikeriri network service

Megumi Takeshita, ikeriri network service

- Founder, ikeriri network service co.,ltd
- Wrote 10+ books about Wireshark
- Reseller of Riverbed Technology (former CACE technologies) in Japan
- Attending all Sharkfest
- Translator of QT Wireshark into Japanese



Point and shoot packet

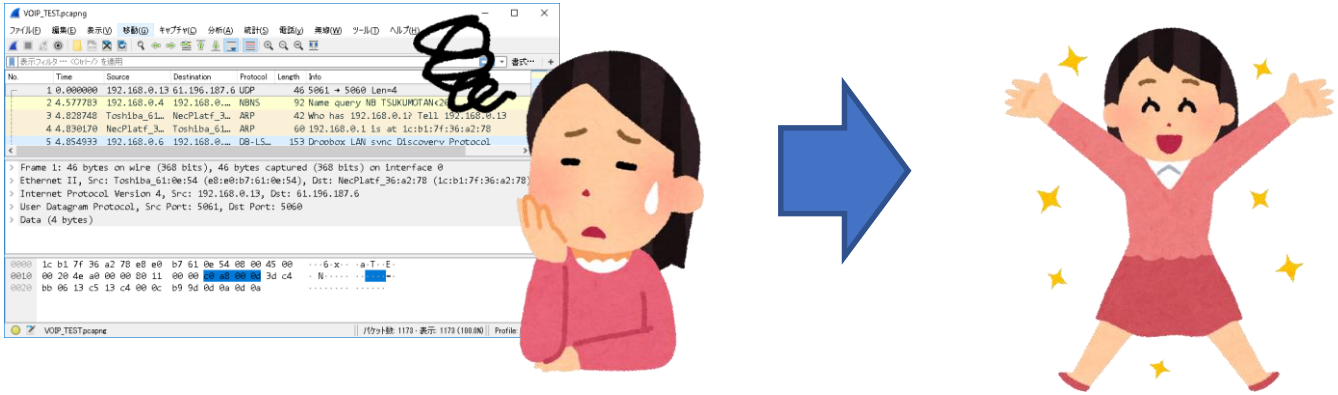
When you debug, troubleshoot, and inspect security issues using Wireshark, you may just look a glance of trace file, and watch each packet sequentially in detail.

In this session Megumi show you good ways to point and shoot packet, using display filters, graphs and tables. Each layer's header has important fields to analyze trace file. So you know shooting point of field in trace file.

This session show you alternative focus points of packet for your debugging, troubleshooting, and inspection.

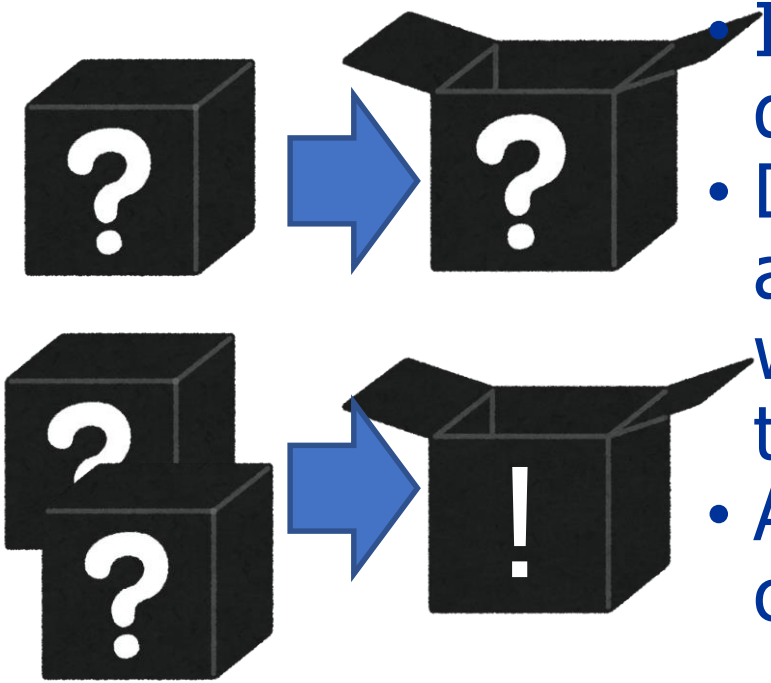
Point and Shoot packet

- There are trace files for debugging and troubleshooting, but what, how do we resolve ?
- Now show you the way to point the key in trace files, then shoot the trouble using Wireshark



- Note all trace files are modified and anonymized

Collect 2 trace files at least



- If you inspect a trace file so deeply, you may not find the key.
- Debugging and troubleshooting are kinds of black box test, so we should point from outside of the system using trace files.
- At least 2 trace files on different conditions are needed for starting

CASE1 CATV Box

- Customer's CATV boxes in the hotel
- Some boxes fails to start up
- Capture 2 trace files
  1_DHCP_SUCCESS.pcap
  1_DHCP_FAIL.pcap
- Let's start debugging

1_DHCP_FAIL.pcap

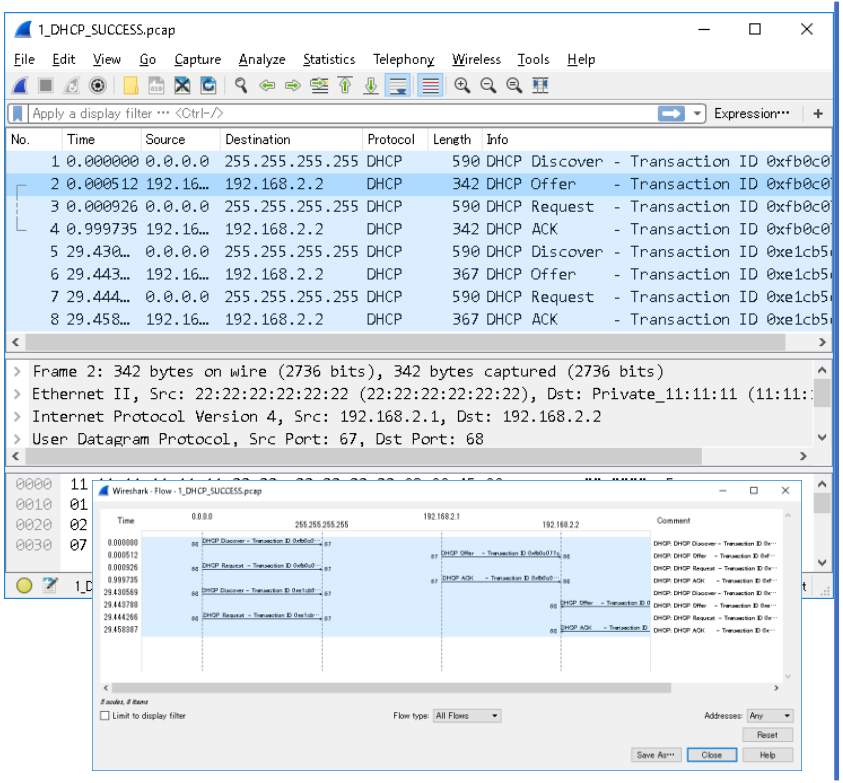
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

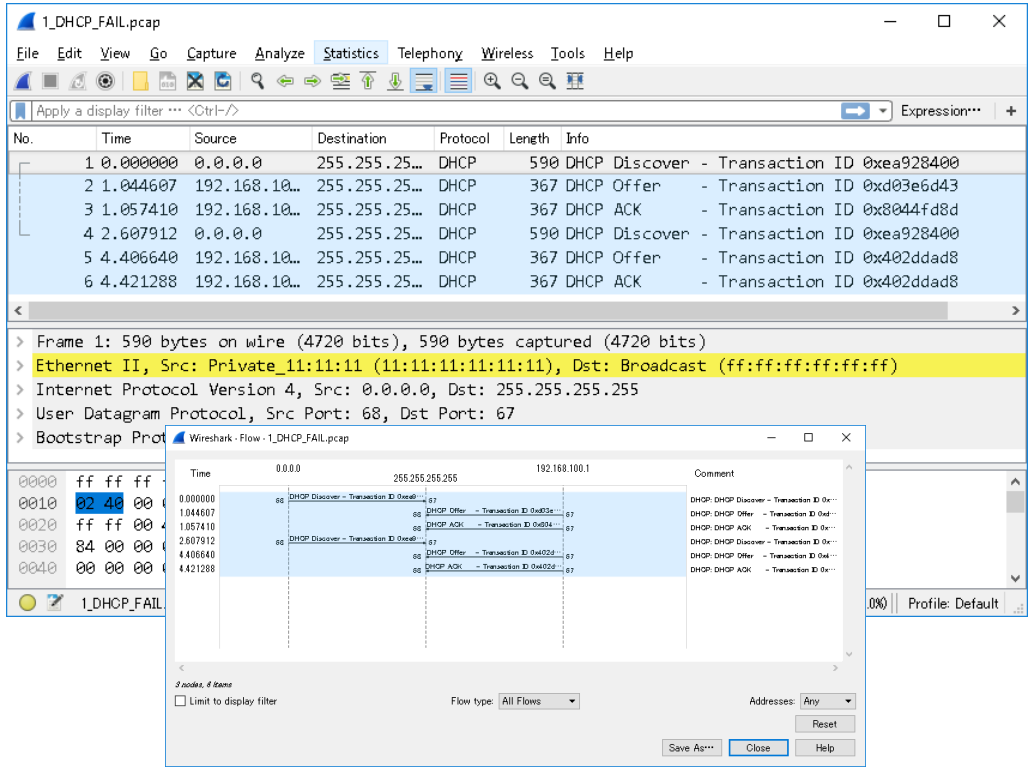
Apply a display filter ··· <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	0.0.0.0	255.255.255.255	DHCP	590	DHCP Discover - Transaction ID
2	1.044607	10.3.0.1	255.255.255.255	DHCP	367	DHCP Offer - Transaction ID
3	1.057410	10.3.0.1	255.255.255.255	DHCP	367	DHCP ACK - Transaction ID

Open 2 trace files

Recommendation: create Flow Graph of both



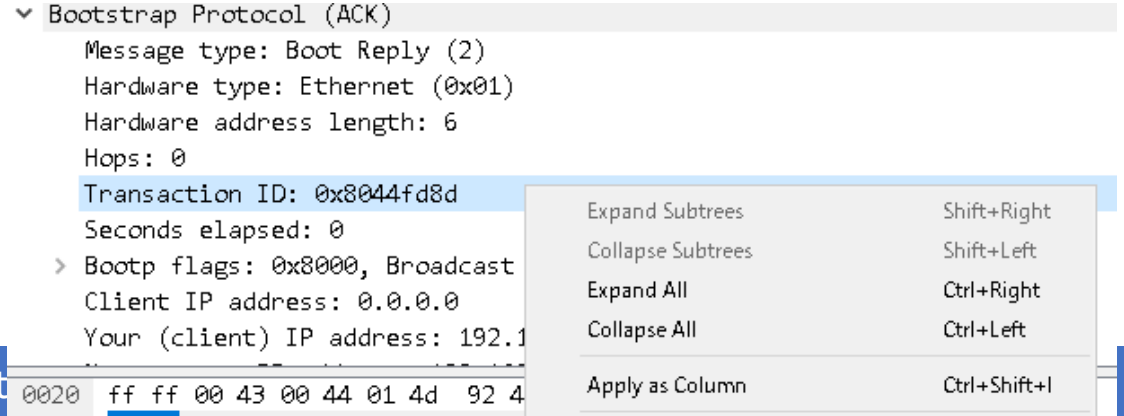


Where is the point ?

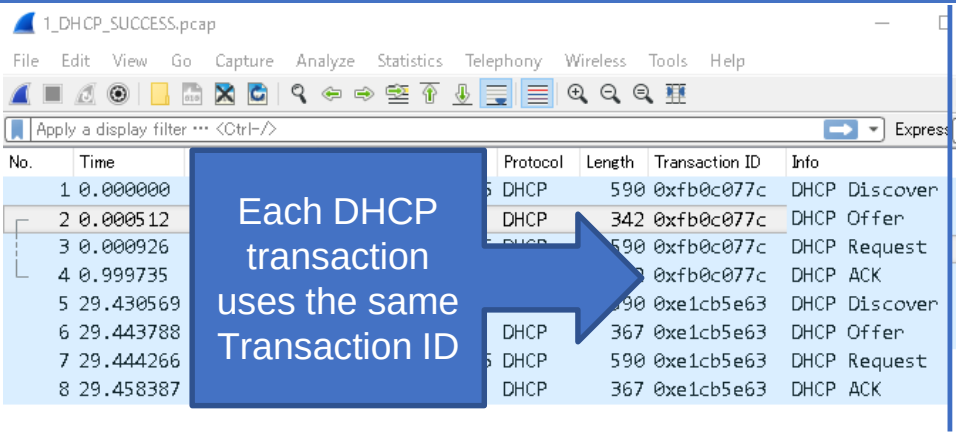
- DHCP is upgrade version of BOOTP by Microsoft, and RFC defines rough procedures. (Discover-Offer-Request-ACK)
- CATV box, router and small IoT devices uses smaller and incomplete protocol stack.
- Where is the point ? Destination address from packet list ? Umm please look Info and open each packet dissector of Bootstrap protocol.

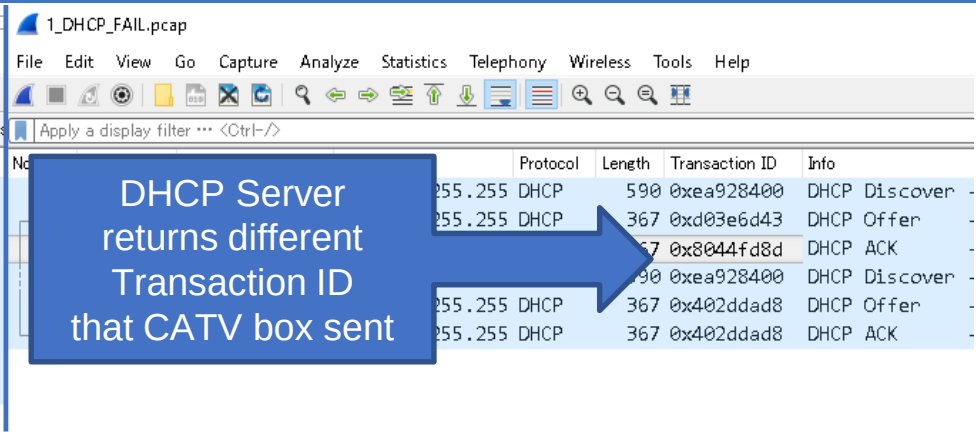
Transaction ID

- Transaction ID is the random number choosed by client. And we use the same transaction id in the specific DHCP process.
- Choose Bootstrap Protocol > Transaction ID and right click to Apply as Column
- Check both trace files



Shoot the Bug





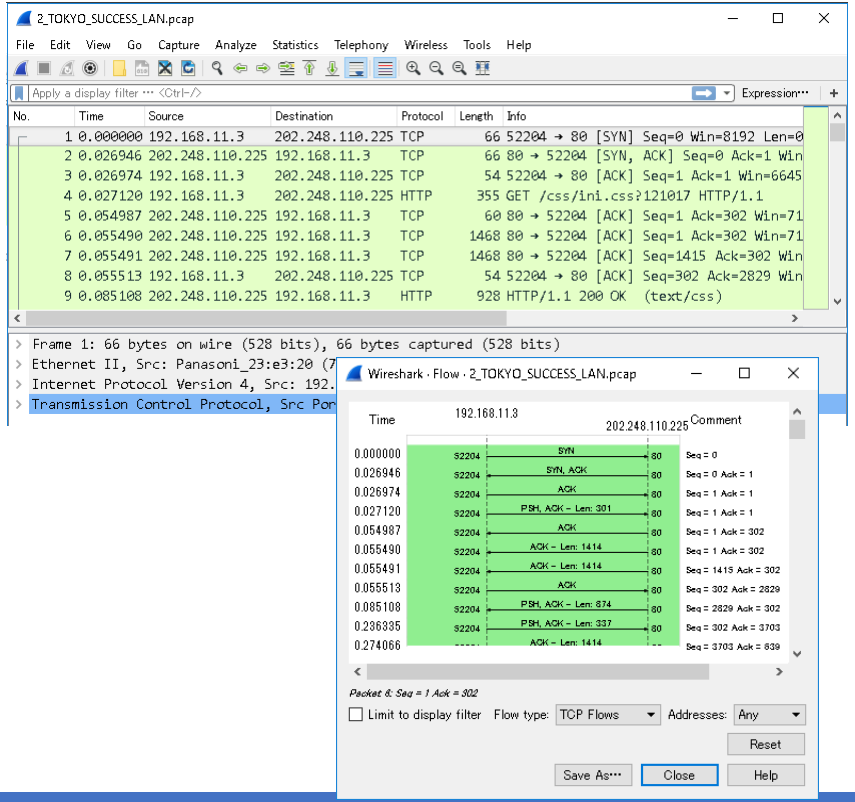
- DHCP server returns different transaction ID that CATV box sent. The problem is the DHCP server (broadband router) behavior is not appropriate.
- IoT devices uses poor dhcp software implementation

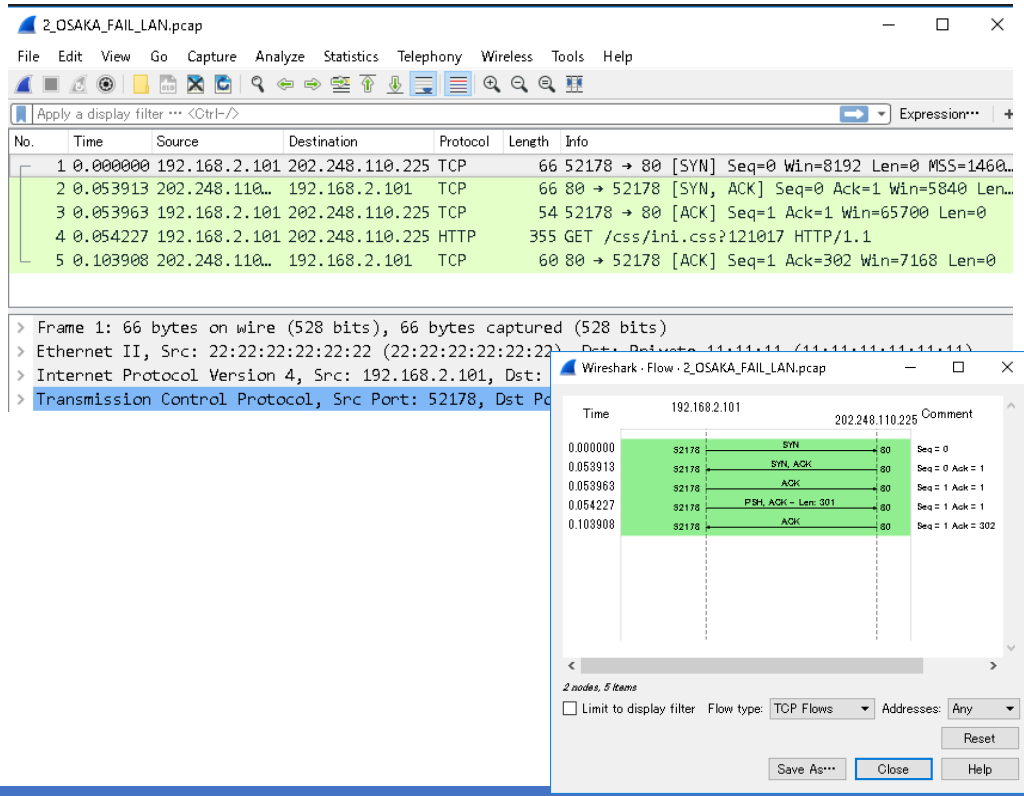
CASE2 Cannot see homepage

- A OSAKA user complains about the trouble some webpage cannot be displayed (the others can). for example, Google OK but Apple NG.
- TOKYO user says he never experienced such trouble in the same environments
- We capture packets in Osaka (2_OSAKA_FAIL_LAN.pcap) and Tokyo (2_TOKYO_SUCCESS_LAN.pcap)

Open 2 trace files

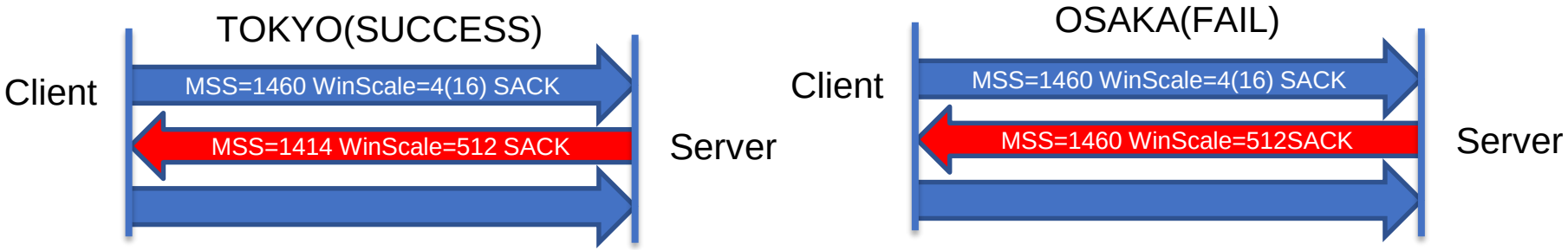
Recommendation: create TCP Flow Graph of both





Where is the point ?

- TCP connection starts with 3 way handshake and exchange parameters (MSS, SACK, Scaling etc.) in first two packet aka TCP negotiation .



- Tokyo’s Web Server uses MSS=1414, Osaka 1460

Maximum Segment Size

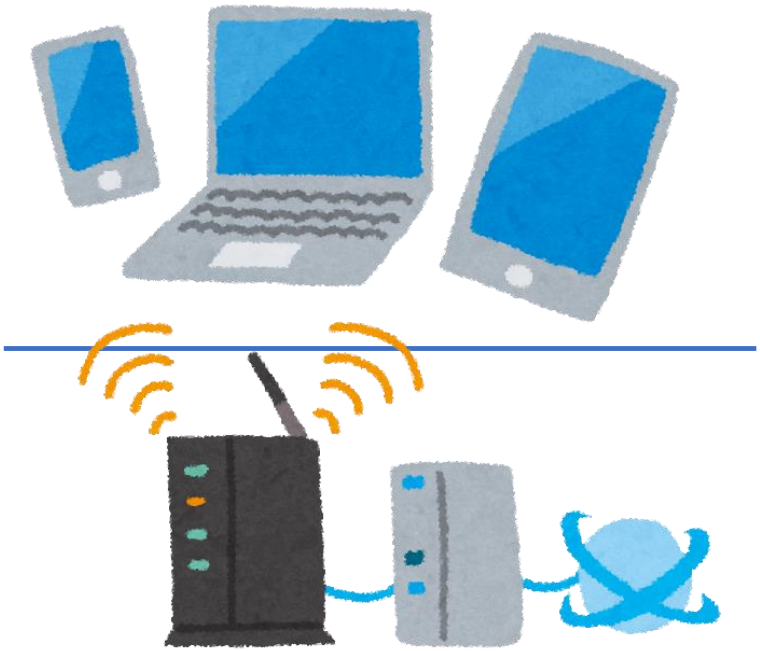
- MSS(Maximum Segment Size) is calculated
  MSS = IP MTU – IP Header (20) – TCP Header (20)

Datalink Header	IP MTU(Maximum Transfer Unit) Size		
	IP Header	TCP Header	MSS (Maximum Segment Size)

each IP and TCP header size without Options is 20

- But why Tokyo is OK and OSAKA is NG ?

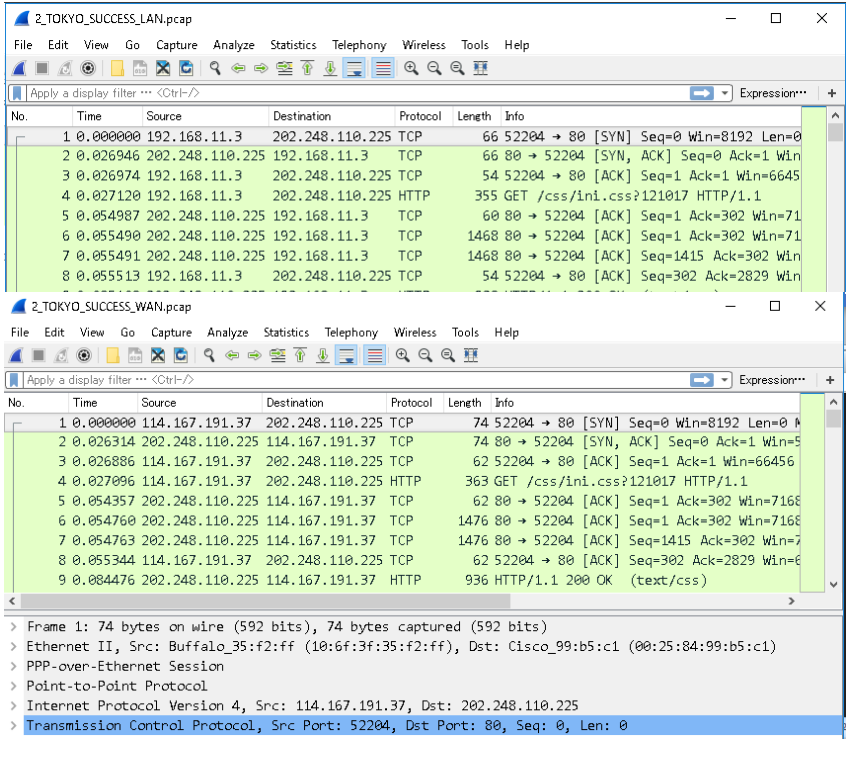
Tapping at both LAN and WAN

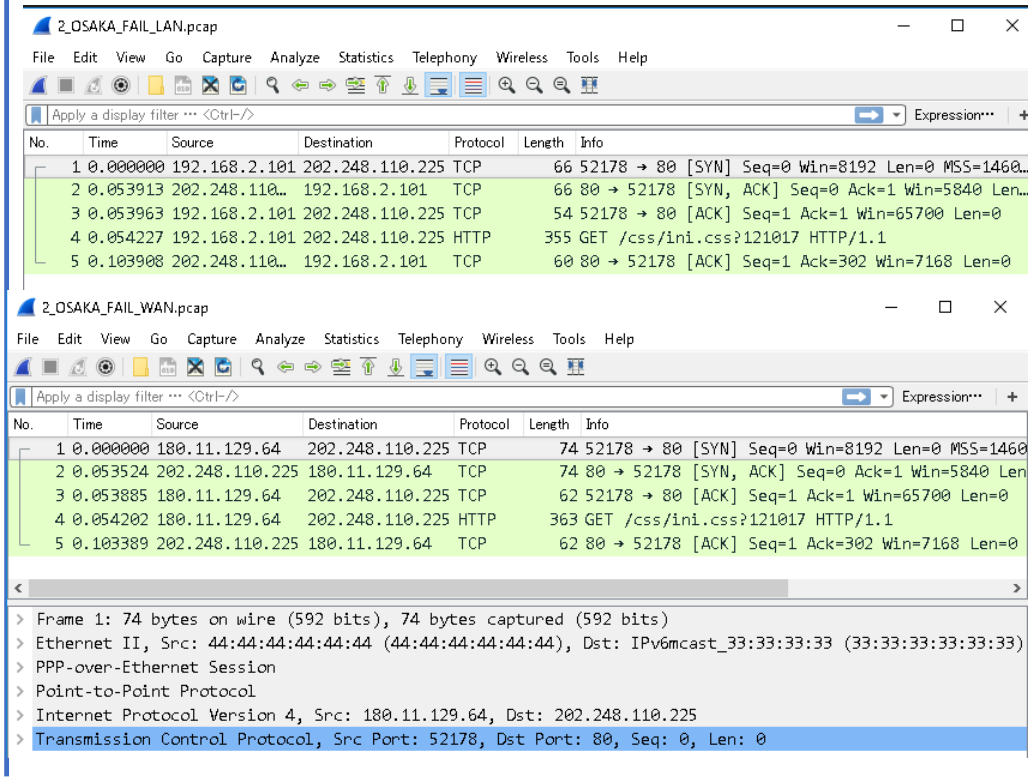


- To resolve internet connection problem, we need to capture at both boundary point of network. (LAN and WAN side)
- We need router side packets.
- Please add to open additional 2 trace files at the WAN side
  2_TOKYO_SUCCESS_WAN.pcap
  2_OSAKA_FAIL_WAN.pcap

Open additional 2 trace files

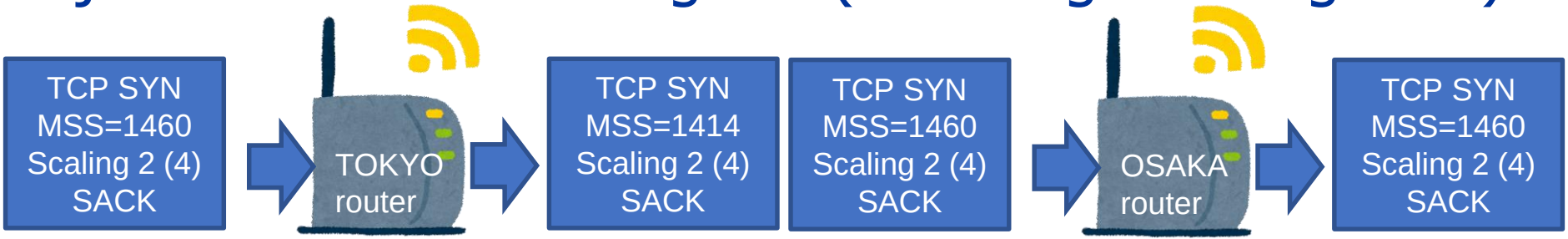
Recommendation: look at TCP negotiation carefully





Where is the point ?

- Both routers in Tokyo and Osaka rewrite Ethernet and IP header (ex. checksum), adds PPPoE and PPP header to connect to access point of the ISP (DSLAM (Digital Subscriber Line Access Multiplexer))
- Osaka router does not adjust MTU and MSS size, just route the IP datagram (including TCP segment)



The Point is in WAN side

#1 in TOKYO LAN
Original Client SYN

```
Transmission Control Protocol, Src Port: 52204, Dst Port: 80, Seq: 0, Len: 0
    Source Port: 52204
    Destination Port: 80
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 0    (relative sequence number)
    [Next sequence number: 0    (relative sequence number)]
    Acknowledgment number: 0
    1000 .... = Header Length: 32 bytes (8)
  > Flags: 0x002 (SYN)
    Window size value: 8192
    [Calculated window size: 8192]
    Checksum: 0x05ac [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  v Options: (12 bytes), Maximum segment size, No-Operation (NOP), Window scal
      TCP Option - Maximum segment size: 1460 bytes
      TCP Option - No-Operation (NOP)
      TCP Option - Window scale: 2 (multiply by 4)
      TCP Option - No-Operation (NOP)
      TCP Option - No-Operation (NOP)
      TCP Option - SACK permitted
  > [Timestamps]
```

#1 in TOKYO WAN
Router rewrites SYN

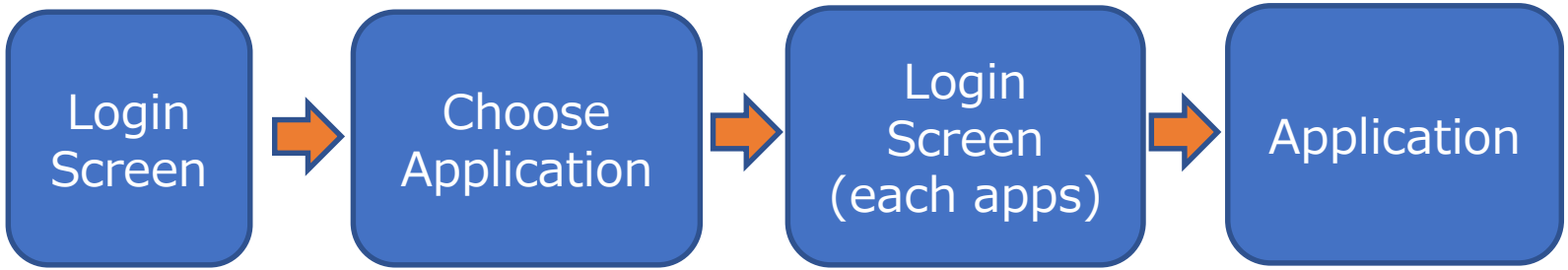
```
Transmission Control Protocol, Src Port: 52204, Dst Port: 80, Seq: 0, Len: 0
    Source Port: 52204
    Destination Port: 80
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 0    (relative sequence number)
    [Next sequence number: 0    (relative sequence number)]
    Acknowledgment number: 0
    1000 .... = Header Length: 32 bytes (8)
  > Flags: 0x002 (SYN)
    Window size value: 8192
    [Calculated window size: 8192]
    Checksum: 0xcd52 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  v Options: (12 bytes), Maximum segment size, No-Operation (NOP), Window scal
      TCP Option - Maximum segment size: 1414 bytes
      TCP Option - No-Operation (NOP)
      TCP Option - Window scale: 2 (multiply by 4)
      TCP Option - No-Operation (NOP)
      TCP Option - No-Operation (NOP)
      TCP Option - SACK permitted
  > [Timestamps]
```

MTU/MSS problem

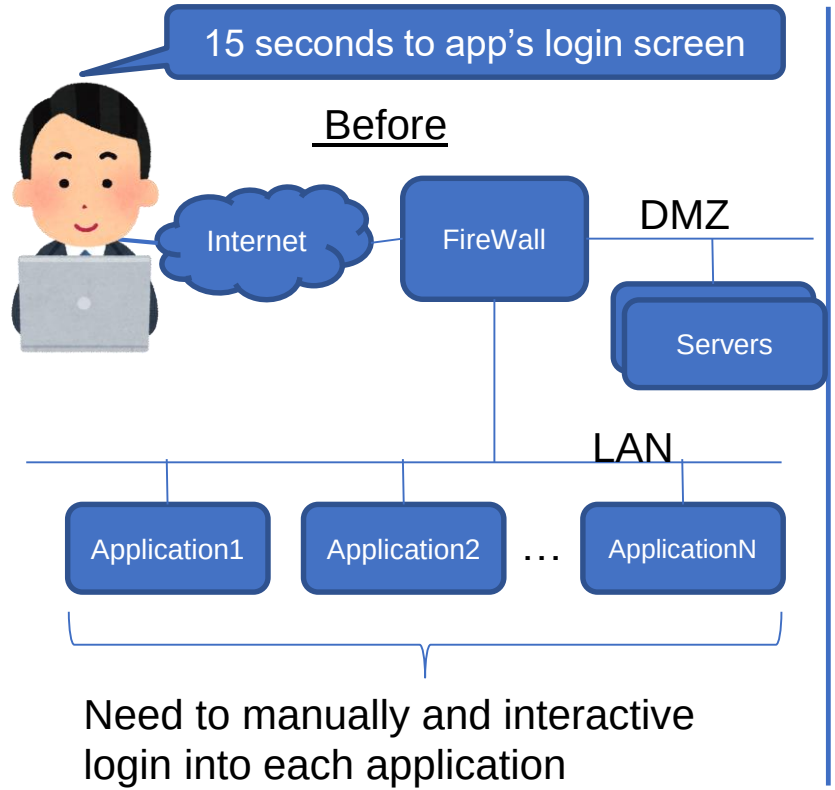
- UDP/IP application uses MTU to fragment each datagrams, and TCP uses MSS to split each segments. TCP sets MSS value in the negotiation.
- MTU and MSS problems are common in Internet connection using PPPoE and PPP datalink.
- We need to modify router (or host) parameters to match ISP's requirements.
- Some application automatically detect and adjust this problem. (ex. PMTU / NDP, and IPv6 !)

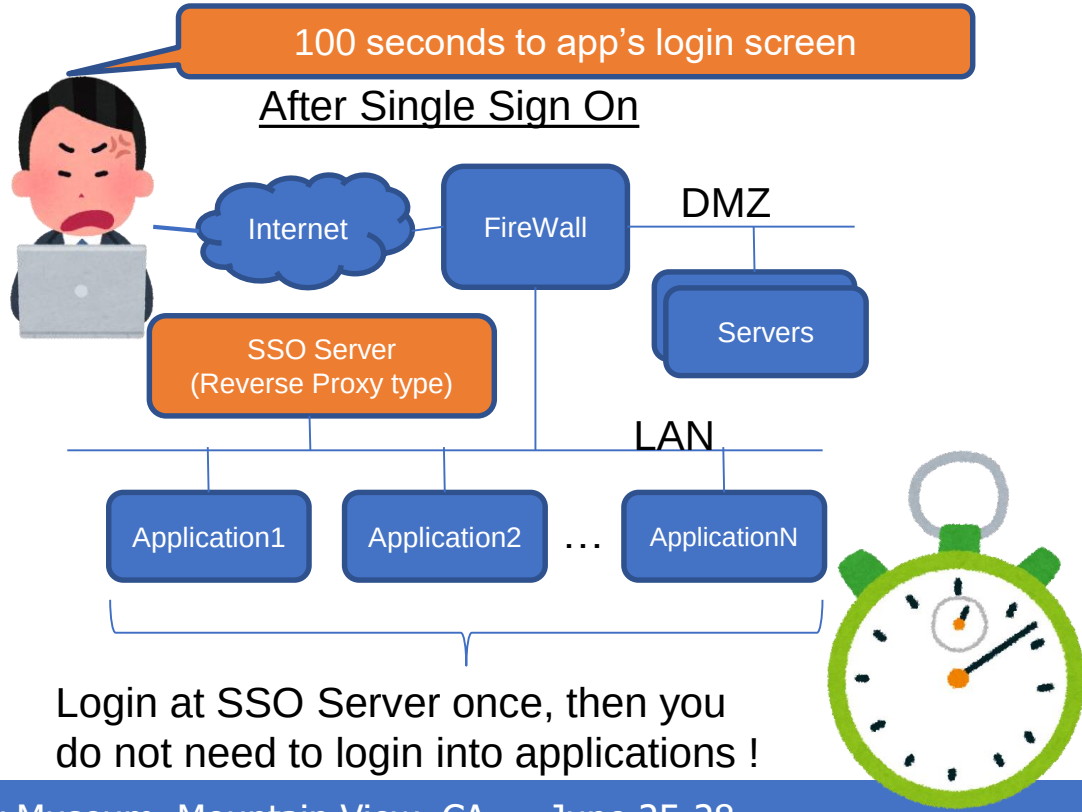
CASE3 Slow Single Sign On

- Many Web based application by 1500 users in enterprise system, each application need to login.
- The customer installed SSO server based on reverse proxy (HTTP proxy authenticates user's application)
- It took 15 seconds to login from remote, but it takes about 100 seconds after SSO, users are angry.



Network diagram



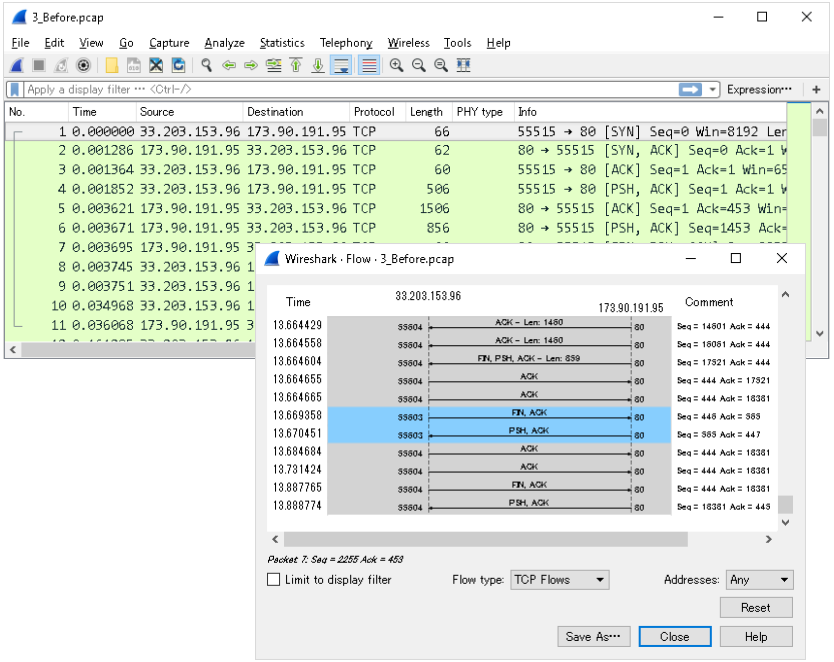


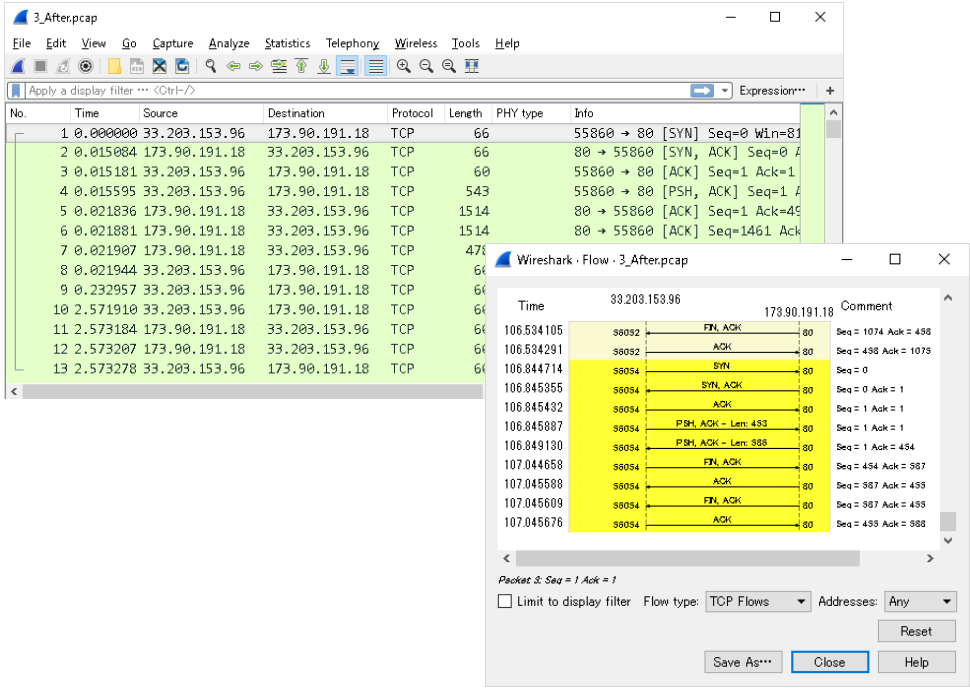
Capture at user point first

- To check repeatability, we need to capture at user-side point first (sometimes the issue is in the user specific environments, or just user's angry because of another reason.)
- Filter and remove other packet for minimizing and simplifying the trouble, use 2 (Before SSO / After SSO) trace files. (3_Before.pcap and 3_After.pcap)
  NOTE: sorry for inconvenience, all payload data is set to zero using pktanon for security reason
- If you cannot find the key, then we need to check the server-side point, so we can save times and money

Open 2 trace files

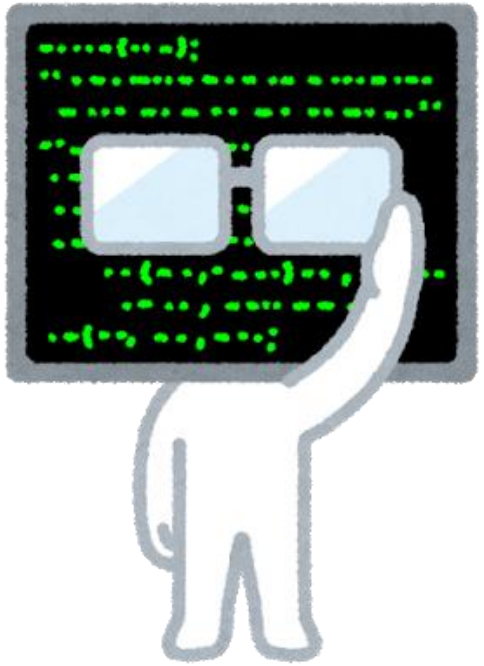
Recommendation: create TCP Flow Graph of both



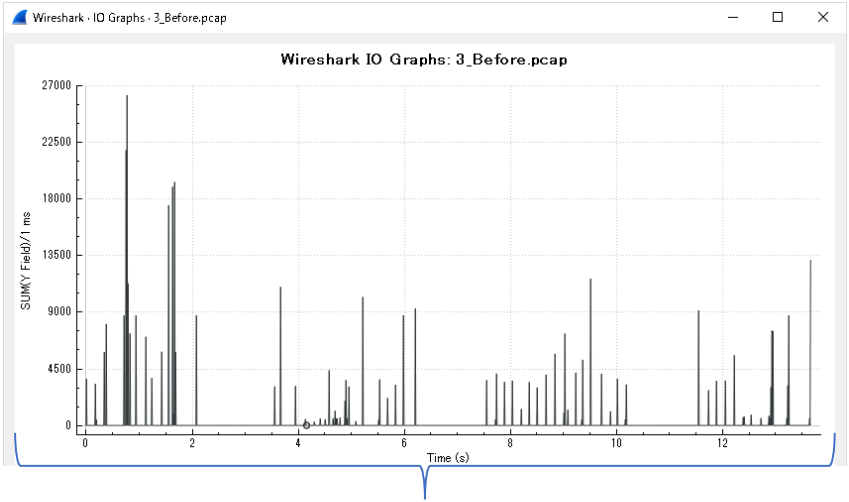


Visualizing TCP Data stream

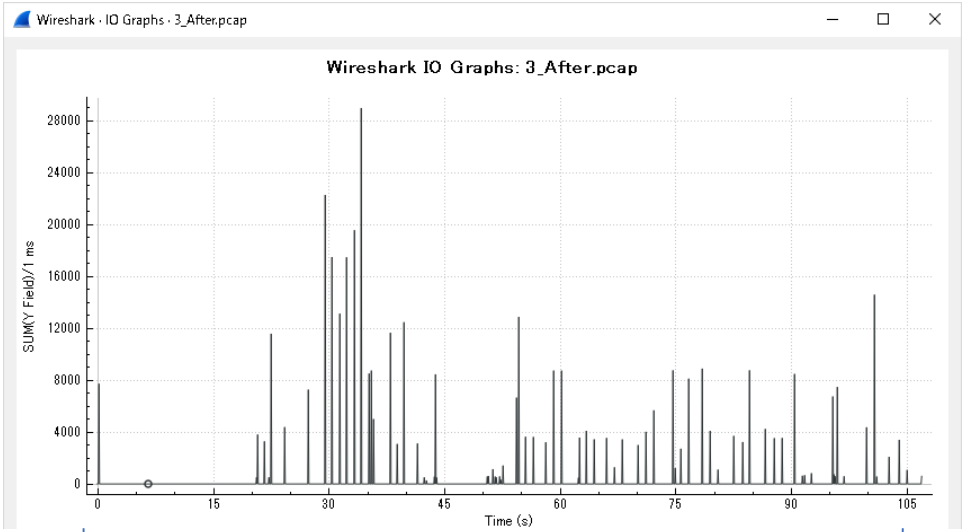
- To understand TCP activities, creating the TCP stream graph is a good idea, but there are many TCP streams in each trace file.
- Using tcp.analysis.bytes_in_flight to visualize the TCP by I/O graph (also check TCP pref.)
- Set Y Axis to SUM (Y Field) to set Y Field as tcp.analysis.bytes_in_flight
- TCP activity is easy to understand with 1ms interval in common internet infrastructure (Test ping to calculate latency at WAN side)



TCP analysis bytes_in_flight



About 14 seconds

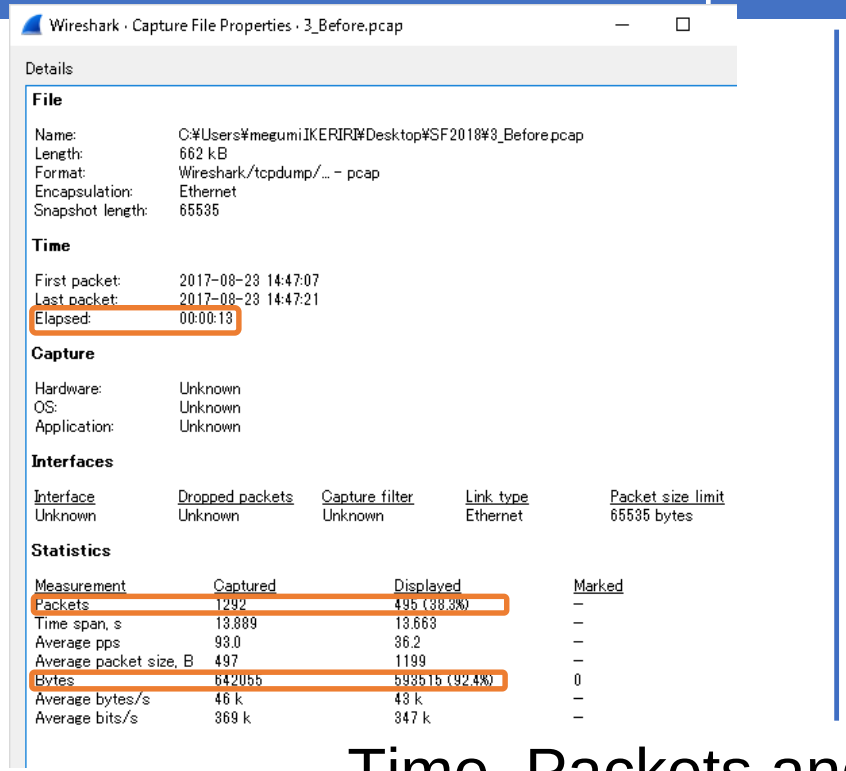


About 105 seconds in same procedure

You can visualize the latency and TCP activities, but Why ?

Capture File Properties

Check the Elapsed Time and Bytes



Wireshark · Capture File Properties · 3_Before.pcap

Details

File

Name:	C:¥Users¥megumi.IKERIRI¥Desktop¥SF2018¥3_Before.pcap
Length:	662 kB
Format:	Wireshark/tcpdump/... - pcap
Encapsulation:	Ethernet
Snapshot length:	65535

Time

First packet:	2017-08-23 14:47:07
Last packet:	2017-08-23 14:47:21
Elapsed:	00:00:13

Capture

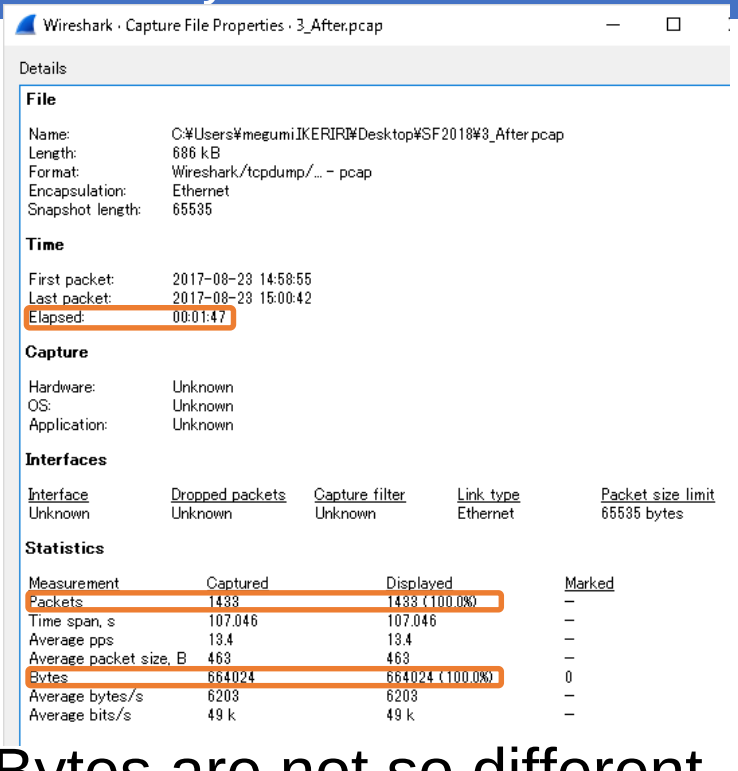
Hardware:	Unknown
OS:	Unknown
Application:	Unknown

Interfaces

Interface	Dropped packets	Capture filter	Link type	Packet size limit
Unknown	Unknown	Unknown	Ethernet	65535 bytes

Statistics

Measurement	Captured	Displayed	Marked
Packets	1292	495 (38.3%)	–
Time span, s	13.889	13.663	–
Average pps	93.0	36.2	–
Average packet size, B	497	1199	–
Bytes	642055	593515 (92.4%)	0
Average bytes/s	46 k	43 k	–
Average bits/s	369 k	347 k	–



Wireshark · Capture File Properties · 3_After.pcap

Details

File

Name:	C:¥Users¥megumi.IKERIRI¥Desktop¥SF2018¥3_After.pcap
Length:	686 kB
Format:	Wireshark/tcpdump/... - pcap
Encapsulation:	Ethernet
Snapshot length:	65535

Time

First packet:	2017-08-23 14:58:55
Last packet:	2017-08-23 15:00:42
Elapsed:	00:01:47

Capture

Hardware:	Unknown
OS:	Unknown
Application:	Unknown

Interfaces

Interface	Dropped packets	Capture filter	Link type	Packet size limit
Unknown	Unknown	Unknown	Ethernet	65535 bytes

Statistics

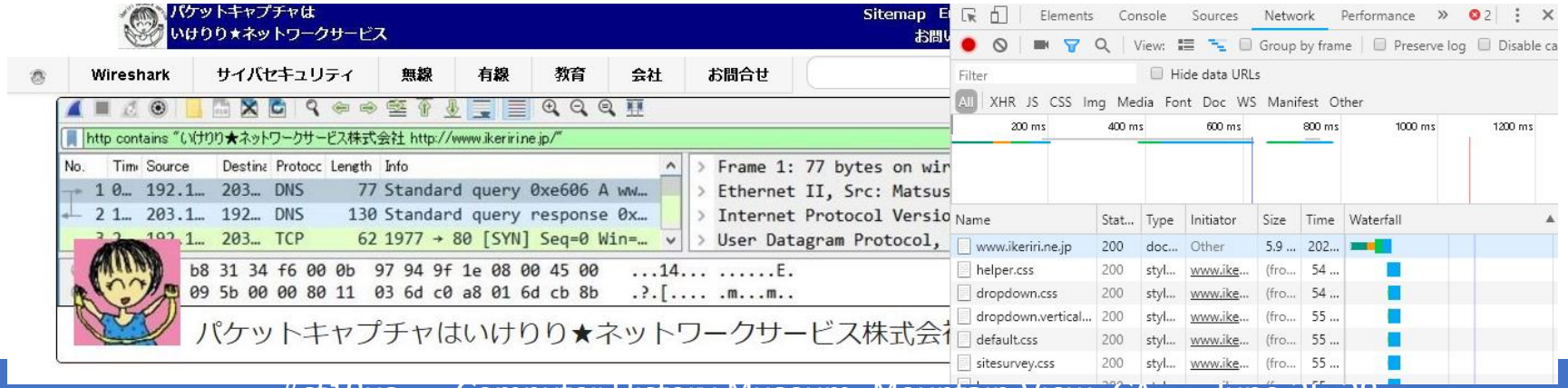
Measurement	Captured	Displayed	Marked
Packets	1433	1433 (100.0%)	–
Time span, s	107.046	107.046	–
Average pps	13.4	13.4	–
Average packet size, B	463	463	–
Bytes	664024	664024 (100.0%)	0
Average bytes/s	6203	6203	–
Average bits/s	49 k	49 k	–

Time, Packets and Bytes are not so different

Where is the point ?

Look into latency of each TCP connection

- Choose Statistics > Conversations and select TCP tab, Look into latency of each TCP connection.
- Sort ascending by Rel Start and check the grey line, the band means the time of each TCP stream like Network tab in Chrome Developer Tool (like Wireshark 3 !)



TCP conversation (sort by Rel Start)

Check the grey band and understand each TCP stream

Wireshark · Conversations · 3_Before.pcap

Ethernet · 1 | IPv4 · 1 | IPv6 | TCP · 90 | UDP

Add	Port	Add	Por	Pac	Bytes	Pac	Bytes	Pacl	Bytes	Rel Start	Duration
33...	55...	17...	80	11	3356	6	812	5	2544	0.000000	0.0361
33...	55...	17...	80	11	2929	6	799	5	2130	0.161285	0.0305
33...	55...	17...	80	9	1196	5	758	4	438	0.189310	0.0182
33...	55...	17...	80	14	7071	7	908	7	6163	0.329731	0.0813
33...	55...	17...	80	14	6432	7	894	7	5538	0.369324	0.0722
33...	55...	17...	80	15	7173	7	884	8	6289	0.709730	0.0911
33...	55...	17...	80	47	40 k	19	1601	28	38 k	0.740883	0.3101
33...	55...	17...	80	83	77 k	31	2328	52	74 k	0.741147	0.5438
33...	55...	17...	80	34	26 k	15	1368	19	25 k	0.756721	0.2320
33...	55...	17...	80	15	9480	7	889	8	8591	0.756907	0.1052
33...	55...	17...	80	15	8749	7	890	8	7859	0.757097	0.0899
33...	55...	17...	80	12	5157	6	817	6	4340	0.773238	0.0581
33...	55...	17...	80	28	19 k	13	1237	15	18 k	0.773423	0.1829
33...	55...	17...	80	11	3493	6	816	5	2677	0.773671	0.0424
33...	55...	17...	80	15	8787	7	877	8	7910	0.826308	0.0991
33...	55...	17...	80	17	10 k	8	942	9	9544	0.936625	0.1127
33...	55...	17...	80	15	9839	7	886	8	8953	1.123503	0.1137

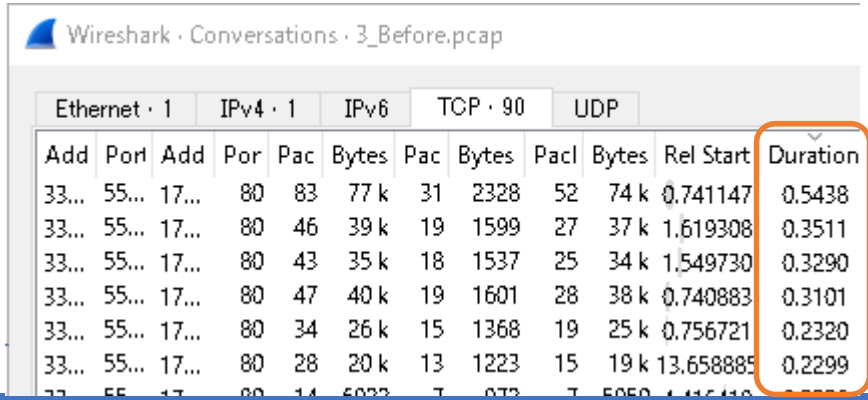
Wireshark · Conversations · 3_After.pcap

Ethernet · 1 | IPv4 · 1 | IPv6 | TCP · 95 | UDP

Adc	Por	Addre	Por	Pac	Bytes	Pac	Byt	Pac	Byte	Rel Start	Duration	Bits/s
3...	5...	173....	80	13	4601	7	909	6	3692	0.000000	2.5733	2825
3...	5...	173....	80	12	3567	6	866	6	2701	20.529745	0.3132	22 k
3...	5...	173....	80	11	2918	6	803	5	2115	21.544630	0.2134	30 k
3...	5...	173....	80	10	1289	6	866	4	423	22.107434	1.7480	3963
3...	5...	173....	80	14	7104	7	956	7	6148	22.432762	1.4353	5328
3...	5...	173....	80	14	6465	7	942	7	5523	24.164929	0.1281	58 k
3...	5...	173....	80	14	7146	7	932	7	6214	27.237261	5.9640	1250
3...	5...	173....	80	50	40 k	21	1...	29	38 k	29.376411	6.2081	2279
3...	5...	173....	80	86	77 k	33	2...	53	74 k	29.376938	6.4103	3114
3...	5...	173....	80	36	26 k	16	1...	20	25 k	30.342772	5.5335	2133
3...	5...	173....	80	17	9633	8	997	9	8636	30.343017	5.1162	1558
3...	5...	173....	80	17	8902	8	998	9	7904	30.343775	5.1019	1564
3...	5...	173....	80	13	5250	7	925	6	4325	31.295568	4.1495	1783
3...	5...	173....	80	30	20 k	14	1...	16	18 k	31.295839	5.5322	1944
3...	5...	173....	80	11	3526	6	864	5	2662	31.296070	4.1486	1666
3...	5...	173....	80	17	8940	8	985	9	7955	32.247220	4.8951	1609
3...	5...	173....	80	19	10 k	9	1...	10	9589	32.247489	4.4750	1877

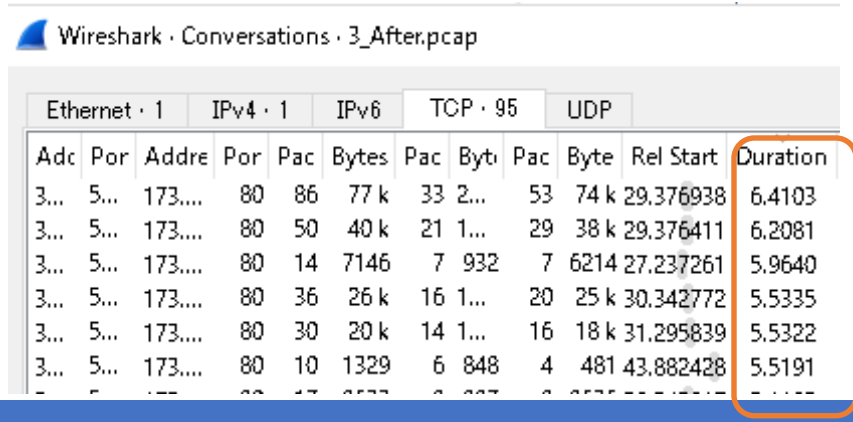
Each duration of TCP Stream

- Sorting descending by Duration Row, you can also understand the difference of TCP stream.
- Almost over 10 times slower than Before SSO.
- It is the key of the performance problem.


Wireshark · Conversations · 3_Before.pcap

Ethernet · 1 | IPv4 · 1 | IPv6 | TCP · 90 | UDP

Add	Port	Add	Por	Pac	Bytes	Pac	Bytes	Pacl	Bytes	Rel Start	Duration
33...	55...	17...	80	83	77 k	31	2328	52	74 k	0.741147	0.5438
33...	55...	17...	80	46	39 k	19	1599	27	37 k	1.619308	0.3511
33...	55...	17...	80	43	35 k	18	1537	25	34 k	1.549730	0.3290
33...	55...	17...	80	47	40 k	19	1601	28	38 k	0.740883	0.3101
33...	55...	17...	80	34	26 k	15	1368	19	25 k	0.756721	0.2320
33...	55...	17...	80	28	20 k	13	1223	15	19 k	13.658885	0.2299


Wireshark · Conversations · 3_After.pcap

Ethernet · 1 | IPv4 · 1 | IPv6 | TCP · 95 | UDP

Adc	Por	Addre	Por	Pac	Bytes	Pac	Byt	Pac	Byte	Rel Start	Duration
3...	5...	173....	80	86	77 k	33	2...	53	74 k	29.376938	6.4103
3...	5...	173....	80	50	40 k	21	1...	29	38 k	29.376411	6.2081
3...	5...	173....	80	14	7146	7	932	7	6214	27.237261	5.9640
3...	5...	173....	80	36	26 k	16	1...	20	25 k	30.342772	5.5335
3...	5...	173....	80	30	20 k	14	1...	16	18 k	31.295839	5.5322
3...	5...	173....	80	10	1329	6	848	4	481	43.882428	5.5191

Shoot the trouble

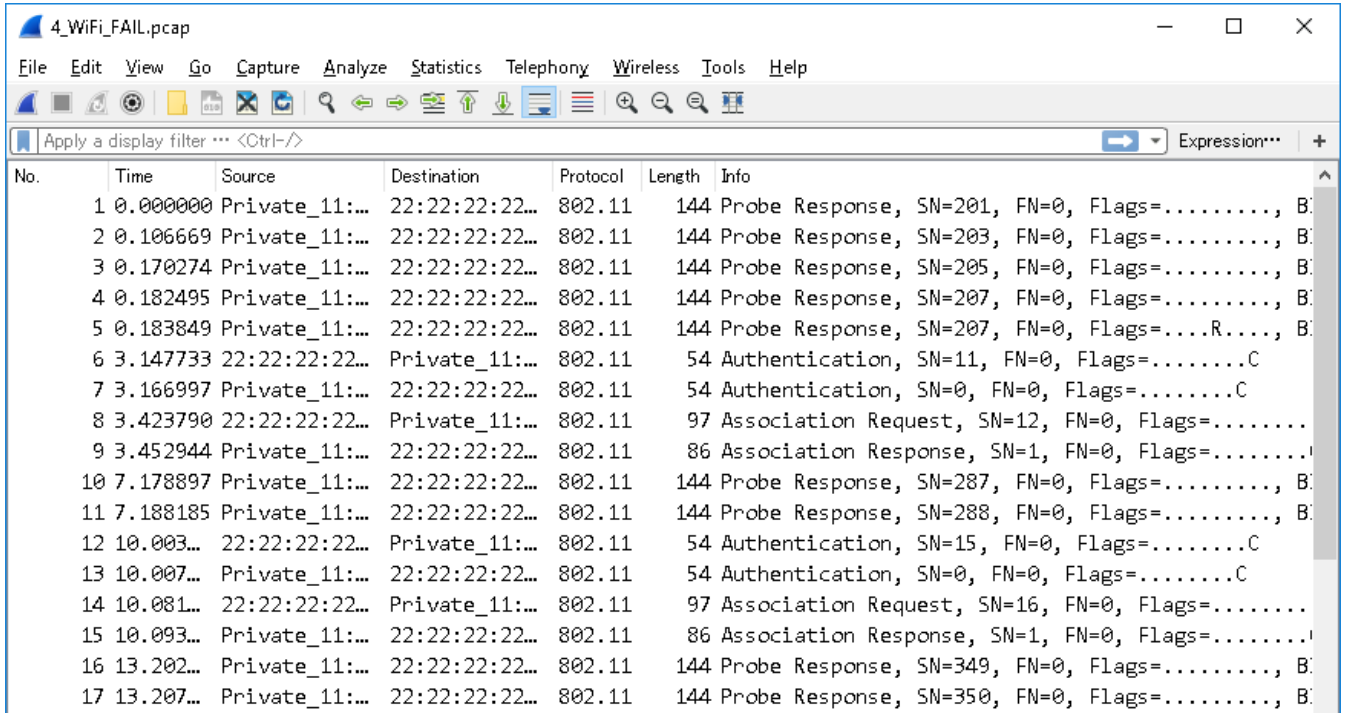
- Increasing Round Trip Time is one of the reasons.
- Sorry All HTTP/HTTPS payloads are anonymized, there is a key in reverse proxy server settings.
- Reverse Proxy SSO server never uses cache, Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0, max-age=0, private, no-transform, proxy-revalidate and Expires: -1
- We need to change reverse proxy server configuration to modify cache settings of SSO server.

CASE 4 Smartphone

- Only one smartphone fails to connect Wi-Fi, though others can access the network using WPA2-PSK.
- The trouble happens in just onely one client in the entire network, it is a hint to minimize the problem.
- Capture and filter using Smartphone
  MAC address (22-22-22-22-22-22)
  4_WiFi_Fail.pcap (anonymized trace file)

Open trace file 4_WiFi_FAIL

This time please think of layer 2 header



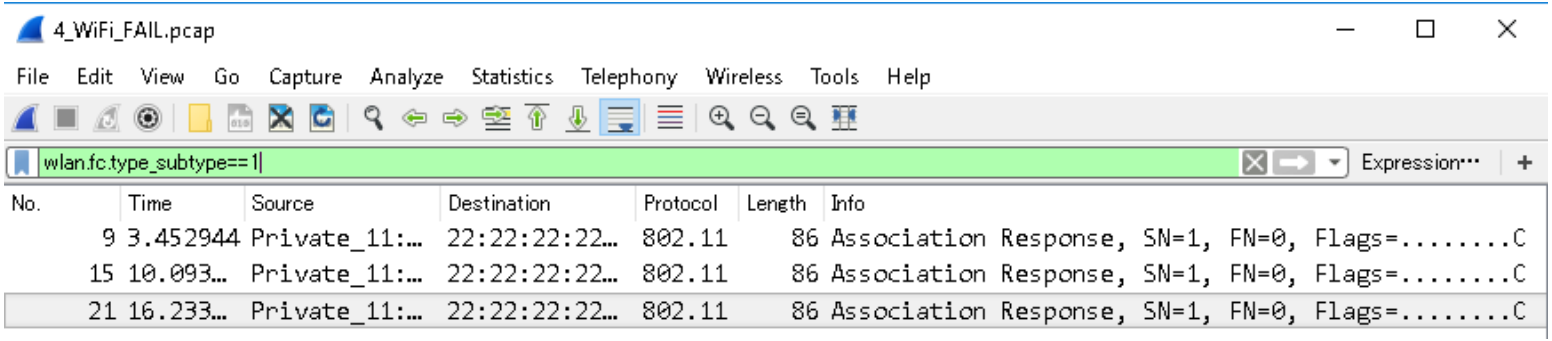
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	Private_11:…	22:22:22:22…	802.11	144	Probe Response, SN=201, FN=0, Flags=........, B
2	0.106669	Private_11:…	22:22:22:22…	802.11	144	Probe Response, SN=203, FN=0, Flags=........, B
3	0.170274	Private_11:…	22:22:22:22…	802.11	144	Probe Response, SN=205, FN=0, Flags=........, B
4	0.182495	Private_11:…	22:22:22:22…	802.11	144	Probe Response, SN=207, FN=0, Flags=........, B
5	0.183849	Private_11:…	22:22:22:22…	802.11	144	Probe Response, SN=207, FN=0, Flags=....R...., B
6	3.147733	22:22:22:22…	Private_11:…	802.11	54	Authentication, SN=11, FN=0, Flags=........C
7	3.166997	Private_11:…	22:22:22:22…	802.11	54	Authentication, SN=0, FN=0, Flags=........C
8	3.423790	22:22:22:22…	Private_11:…	802.11	97	Association Request, SN=12, FN=0, Flags=........
9	3.452944	Private_11:…	22:22:22:22…	802.11	86	Association Response, SN=1, FN=0, Flags=........
10	7.178897	Private_11:…	22:22:22:22…	802.11	144	Probe Response, SN=287, FN=0, Flags=........, B
11	7.188185	Private_11:…	22:22:22:22…	802.11	144	Probe Response, SN=288, FN=0, Flags=........, B
12	10.003…	22:22:22:22…	Private_11:…	802.11	54	Authentication, SN=15, FN=0, Flags=........C
13	10.007…	Private_11:…	22:22:22:22…	802.11	54	Authentication, SN=0, FN=0, Flags=........C
14	10.081…	22:22:22:22…	Private_11:…	802.11	97	Association Request, SN=16, FN=0, Flags=........
15	10.093…	Private_11:…	22:22:22:22…	802.11	86	Association Response, SN=1, FN=0, Flags=........
16	13.202…	Private_11:…	22:22:22:22…	802.11	144	Probe Response, SN=349, FN=0, Flags=........, B
17	13.207…	Private_11:…	22:22:22:22…	802.11	144	Probe Response, SN=350, FN=0, Flags=........, B

Association Response

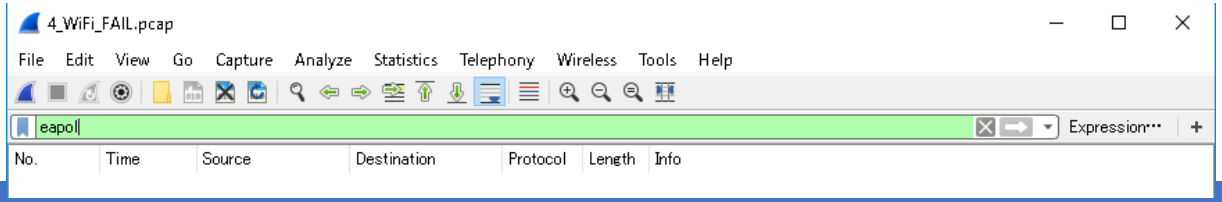
- We need to determine layer 2 or upper. Association Response (wlan.fc.type_subtype==1) is a good indicator of Wi-Fi troubleshooting.
- If you find the Association Response, Datalink procedures are almost completed, exchanging Beacon, Probe Request, Probe Response, Authentication x 2, Association Request then link up with Association Response packet
- Let’s filter packets with “wlan.fc.type_subtype==1”

EAPOL 4 way handshake

- You can find Association Response 3 times ? Why



- EAPOL 4 Way handshakes are required in a ordinal WPA-PSK process. Try to filter with “eapol”

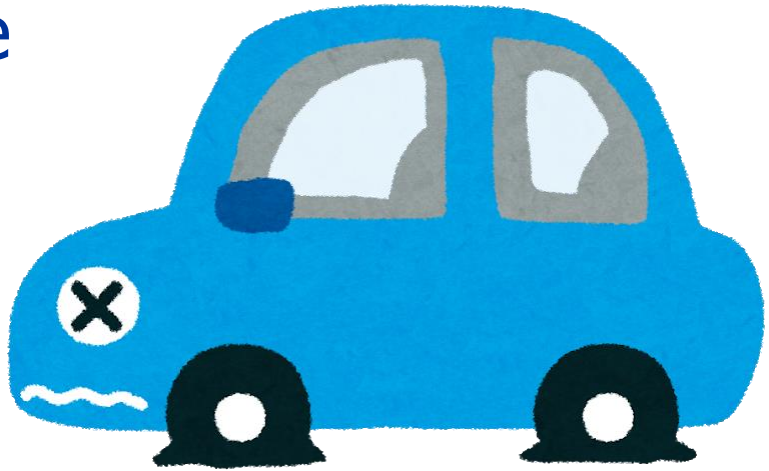


Where is the point ?

Stack point has a problem to shoot the trouble

- Finding stack point is important especially in Wireless troubleshooting, in this case, every time the processes are stopped at Association Response.
- Look into Association Response carefully to find the reason

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	Private_11:…	22:22:22:22…	802.11	144	Probe Response, SN=201, FN=0, Flags=........, B
2	0.106669	Private_11:…	22:22:22:22…	802.11	144	Probe Response, SN=203, FN=0, Flags=........, B
3	0.170274	Private_11:…	22:22:22:22…	802.11	144	Probe Response, SN=205, FN=0, Flags=........, B
4	0.182495	Private_11:…	22:22:22:22…	802.11	144	Probe Response, SN=207, FN=0, Flags=........, B
5	0.183849	Private_11:…	22:22:22:22…	802.11	144	Probe Response, SN=207, FN=0, Flags=....R...., B
6	3.147733	22:22:22:22…	Private_11:…	802.11	54	Authentication, SN=11, FN=0, Flags=........C
7	3.166997	Private_11:…	22:22:22:22…	802.11	54	Authentication, SN=0, FN=0, Flags=........C
8	3.423790	22:22:22:22…	Private_11:…	802.11	97	Association Request, SN=12, FN=0, Flags=........
9	3.452944	Private_11:…	22:22:22:22…	802.11	86	Association Response, SN=1, FN=0, Flags=........
10	7.178897	Private_11:…	22:22:22:22…	802.11	144	Probe Response, SN=287, FN=0, Flags=........, B
11	7.188185	Private_11:…	22:22:22:22…	802.11	144	Probe Response, SN=288, FN=0, Flags=........, B
12	10.003…	22:22:22:22…	Private_11:…	802.11	54	Authentication, SN=15, FN=0, Flags=........C
13	10.007…	Private_11:…	22:22:22:22…	802.11	54	Authentication, SN=0, FN=0, Flags=........C
14	10.081…	22:22:22:22…	Private_11:…	802.11	97	Association Request, SN=16, FN=0, Flags=........
15	10.083…	Private_11:…	22:22:22:22…	802.11	86	Association Response, SN=1, FN=0, Flags=........
16	13.202…	Private_11:…	22:22:22:22…	802.11	144	Probe Response, SN=349, FN=0, Flags=........, B
17	13.207…	Private_11:…	22:22:22:22…	802.11	144	Probe Response, SN=350, FN=0, Flags=........, B
18	15.994…	22:22:22:22…	Private_11:…	802.11	54	Authentication, SN=26, FN=0, Flags=........C



Invalid AKMP (Adaptive Key Management Protocol)

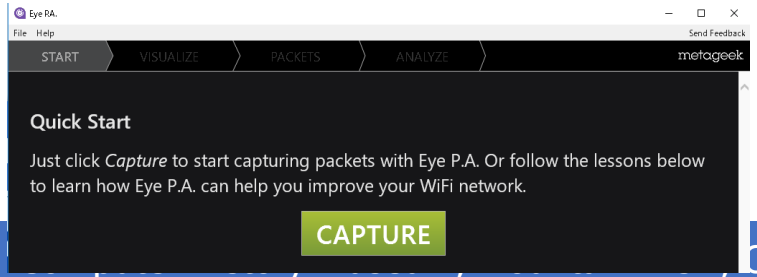
IEEE802.1x authentication method

```
> Radiotap Header v0, Length 20
> 802.11 radio information
> IEEE 802.11 Association Response, Flags: ........C
v IEEE 802.11 wireless LAN
  v Fixed parameters (6 bytes)
    > Capabilities Information: 0x0031
      Status code: Invalid AKMP (0x002b)
      ..00 0000 0000 0000 = Association ID: 0x0000
  v Tagged parameters (32 bytes)
    > Tag: Supported Rates 1(B), 2(B), 5.5(B), 11(B), [Mbit/sec]
    > Tag: Vendor Specific: Microsoft Corp.: WMM/WME: Parameter Element
```

- Invalid AKMP means key exchange settings are mismatch between Access point and Station.
- This time AP use PSK(Pre Shared Key) Passphrase, but STA use IEEE802.1x authentication server.

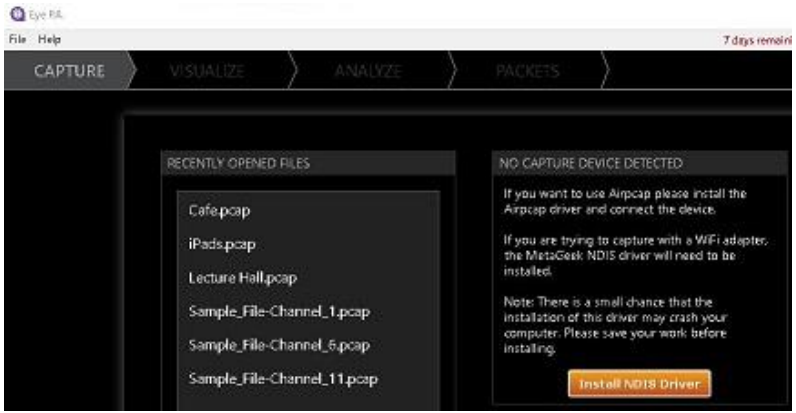
One more BREAKING NEWS

- You may know, Metageek announced Eye P.A. support capturing IEEE802.11n and then IEEE802.11ac in Windows environment !
- After the end of AirpcapNX era, wireless packet analysis on Windows is limited. (AcrylicWifi and other NDIS based driver never behaves as we wish…)



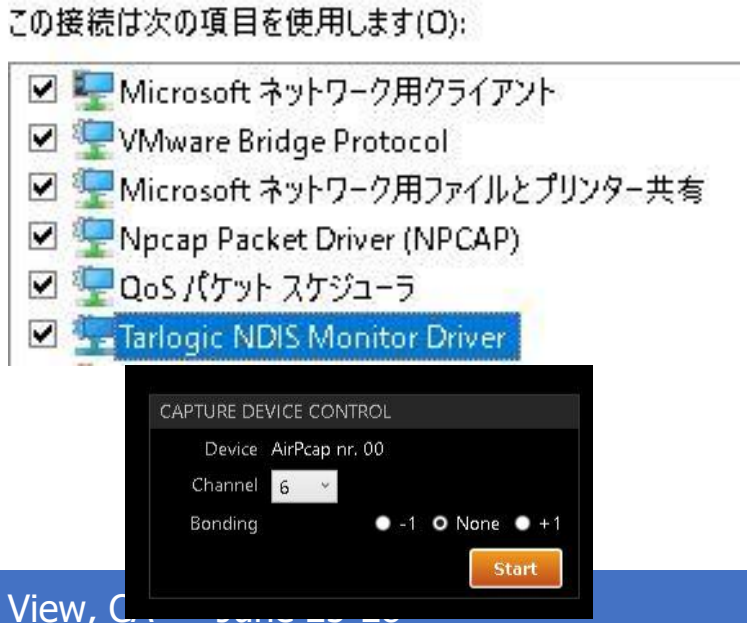
Eye P.A. (Eye Packet Analyzer)

- You need Beta version of Eye P.A.
  now EyeP.A. supports 11n with v1.12
  then 11AC support comes at last week !!
- Start up EyePA with Administrator privilege.
  Install NDIS Driver
- Then you can capture via
  CAPTURE DEVICE CONTROL
- Off course you can save and
  send trace to Wireshark !



Compatible Adapters (1)

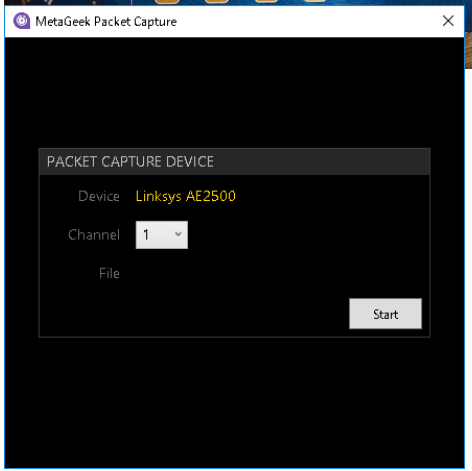
- Eye P.A. ver1.12 (May 2018) supports 11N
  Linksys AE2500, Linksys AE1200, Netgear A6200
  (NDIS based driver)
  Riverbed AirPcapNX, AirPcapTX
  (AirPcap based driver)
- Tarlogic NDIS Monitor Driver and AirPcap driver are used when we use 11n in ver1.12



Compatible Adapters (2)

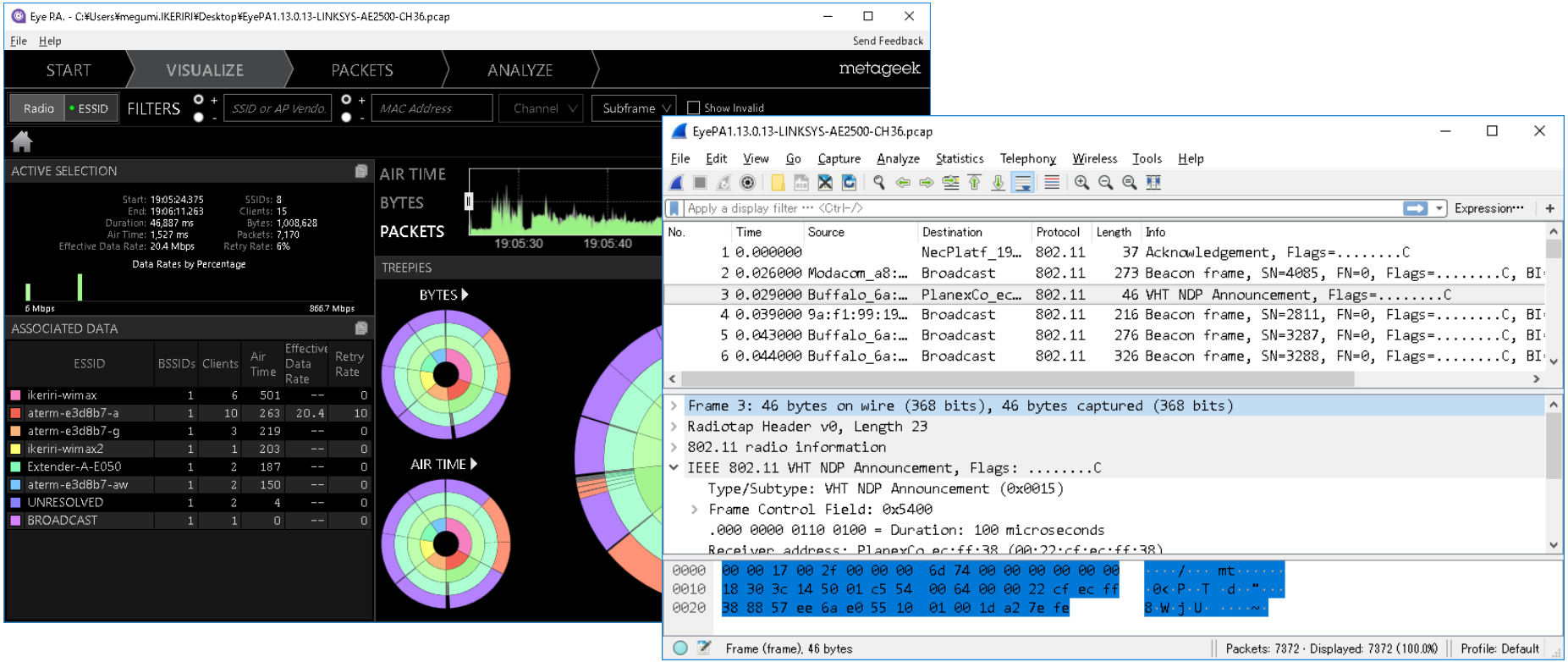
- Eye P.A. Version 1.13 (A.k.a Imperial Eye P.A.)
- Support adapter:
  ASUS USB-AC56, ASUS USB-AC68, ALFA Network AWUS1900, Linksys WUSB6300, Amped Wireless ACA1, EnGenius EUB1200AC, D-Link DWA-182 rev C1, D-Link DWA-192, TRENDnet TEW-805UB, TP-LINK Archer T4U v2, TP-LINK Archer T4UH v2, Edimax EW-7822UAC, Edimax EW-7833UAC

TEST: Linksys AE2500



- Linksys AE2500 Dual Band N USB3 2.4/5GHz USB2 BCM4323 (Broadcom chip)
- CH 1-14,CH 36-64 W52, CH 100-165 W52 W53 in Japan
- Automatically recognized in Eye P.A. version 1.13.0.13
- You can choose Packet Data is truncated or not

TEST: Linksys AE2500



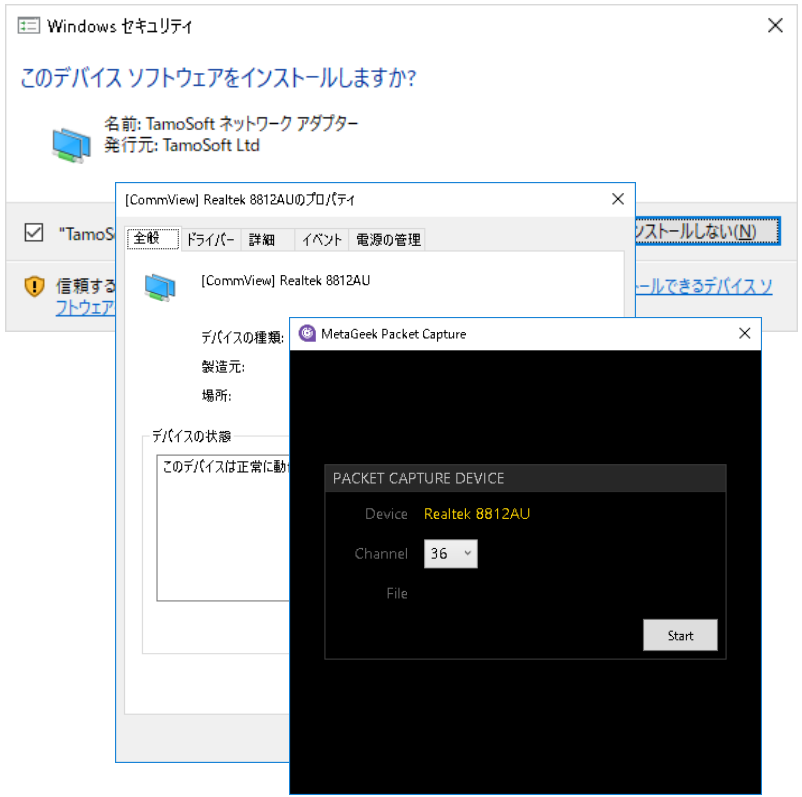
Compatible Adapters (3)

- Eye P.A. ver2 supports 11AC
- Supported adapters (IEEE802.11ac)
  Linksys WUSB6300 (recommended), ASUS USB-AC56, ASUS USB-AC68, ALFA Network AWUS1900, Amped Wireless ACA1, EnGenius EUB1200AC, D-Link DWA-182 rev C1, D-Link DWA-192, TRENDnet TEW-805UB, TP-LINK Archer T4U v2, TP-LINK Archer T4UH v2, Edimax EW-7822UAC, Edimax EW-7833UAC
- 802.11n adapters
  Linksys AE2500 (recommended), Linksys AE1200, Netgear A6200
  NOTE LinksysAE1200 and NetgearA6200 cannot be used on DFSCH
- RiverBed AirPcap NX, AirPcapTX
- Tamosoft (famous as TamoGraph) driver is used.

TEST2 : ALFA AWUS036ACH

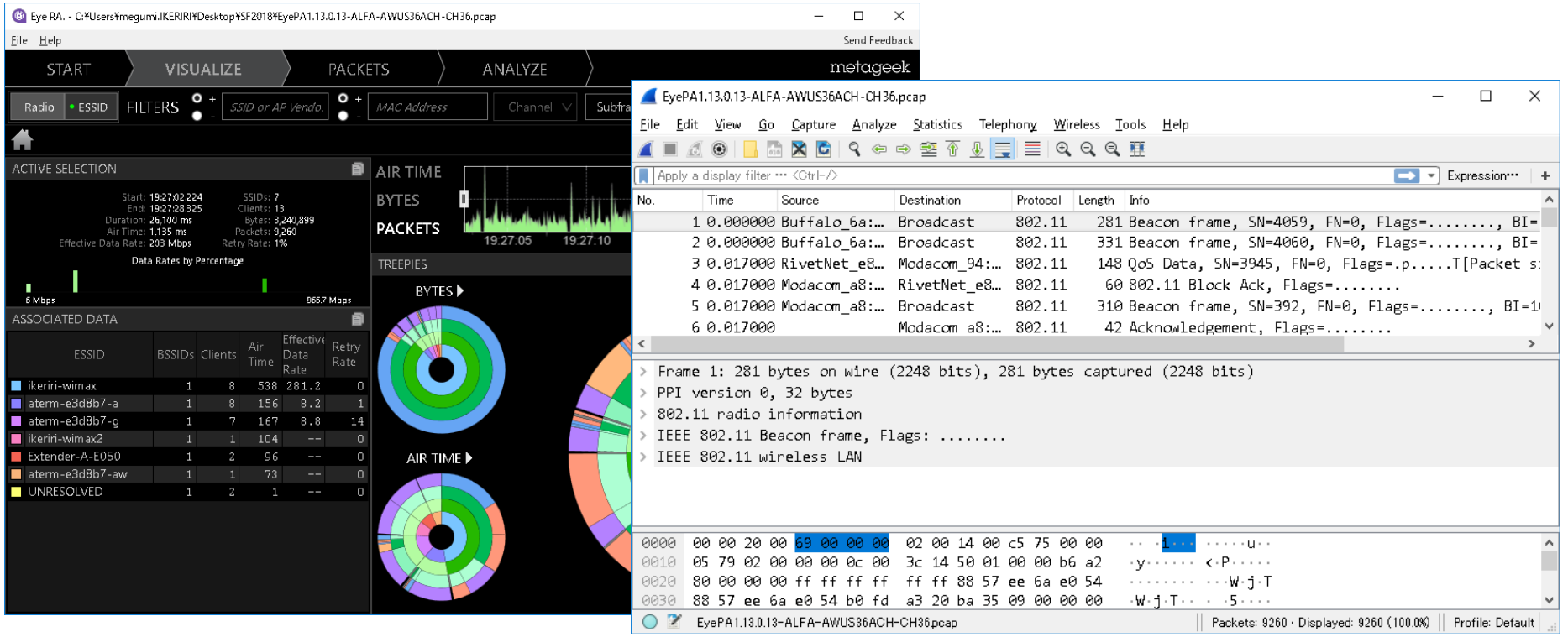
- Alfa Long-Range Dual-Band AC1200 Wireless USB 3.0 Wi-Fi Adapter w/2x 5dBi External Antennas – 2.4GHz 300Mbps/5GHz 867Mbps – 802.11ac & A, B, G, N
- RTL8812AU Realtek chipset
- CH 1-14,CH 36-64 W52, CH 100-165 W52 W53 in Japan

TEST2 : ALFA AWUS036ACH



- This ALFA model is not officially supported, but it try to install specific drivers
- Automatically recognized in Eye P.A. version 1.13.0.13
- You can choose Packet Data is truncated or not

DEMO : ALFA AWUS036ACH



Thank you for attending !

どうもありがとうございました

Q&A

いけりり★ネットワークサービス

http://www.ikeriri.ne.jp