Understand CAPWAP AP PMTU Discovery

Contents

Introduction

Scenario and Scope

- CAPWAP Control vs. Data (What is Negotiated)
- Facts: Maximum Sized CAPWAP Packet
- Three-Stage PMTU Checks

CAPWAP PMTU Discovery Mechanism

- IOS AP Behavior
  - AP Join Phase
  - RUN State Phase
- COS AP Behaviour
  - AP Join Phase
  - RUN State Phase

Conclusion (Algorithm Summary)

Related CDETs

Introduction

This document describes the CAPWAP Access Point Path Maximum Transmission Unit (PMTU) discovery mechanism on IOS® XE and COS, issues, and resolution.

Scenario and Scope

You typically see PMTU issues when a CAPWAP access point (AP) at a remote site registers to a wireless LAN controller (WLC) across a WAN especially when the path includes VPN, GRE, or any network segment with an MTU lower than the standard 1500 bytes.

We also examine authentication with Extensible Authentication Protocol Transport Layer Security (EAP-TLS). Because EAP-TLS exchanges large certificates, a reduced path MTU increases fragmentation risk.

All logs were captured on code version 17.9.3. Outputs are truncated to show only relevant lines.

CAPWAP Control vs. Data (What is Negotiated)

CAPWAP Control:
The control channel handles critical management messages such as join requests, configuration exchanges, and keepalive signals. These messages are secured using DTLS and are the primary focus of the Path MTU (PMTU) negotiation process to ensure reliable and efficient control plane communication.

CAPWAP Data:
This channel carries encapsulated client traffic, typically also protected by DTLS in most deployments. While PMTU negotiation occurs on the control channel, the resulting PMTU values indirectly determine the maximum packet size for data plane encapsulation, impacting client data transmission reliability and fragmentation.

Examples

- Control Packets: Join requests and responses, configuration updates, and echo/keepalive messages.
- Data Packets: Encapsulated client frames transmitted between the Access Point (AP) and Wireless LAN Controller (WLC).

Facts: Maximum Sized CAPWAP Packet

IOS AP (example)

Sent PMTU packet size: 1499 bytes = Ethernet + CAPWAP PMTU

- Ethernet = 14 Bytes
- CAPWAP PMTU = 1485 Bytes
  - Outer IP = 20 Bytes
  - UDP = 25 Bytes
  - DTLS = 1440 Bytes

AP-COS (Example)

Sent PMTU packet size: 1483 Bytes = Ethernet + CAPWAP PMTU

- Ethernet = 14 bytes
- CAPWAP PMTU = 1469 bytes
  - Outer IP = 20 bytes
  - UDP = 25 bytes
  - DTLS = 1424 bytes

Three-Stage PMTU Checks

Both platforms probe three hard-coded PMTU values: 576, 1005, and 1485. The difference is how each platform counts the Ethernet header:

- IOS APs do not include the Ethernet header in the 576/1005/1485 values.
  - Total frame = Ethernet (14) + PMTU (576/1005/1485) ⇒ 590, 1019, 1499 bytes (wire size).
- AP-COS does include the Ethernet header in the 576/1005/1485 values.
  - Total frame = PMTU (already includes Ethernet). These packets are 14 bytes smaller on the wire than IOS AP equivalents.

CAPWAP PMTU Discovery Mechanism

IOS AP Behavior

AP Join Phase

During CAPWAP join, the AP negotiates a maximum CAPWAP PMTU of 1485 bytes with the DF bit set. It waits 5 seconds for a response.

- If no response or an ICMP “Fragmentation Needed” arrives, the AP falls back to 576 bytes to

complete join quickly, then tries to raise PMTU after it reaches RUN.

Packet capture (example)

Packet number 106 You see a 1499-byte probe (DF set). No same-size response indicates the packet could not traverse the path without fragmentation. You then see ICMP “Fragmentation Needed.”

17	07:41:47.427848	0.002187	10.201.166.185	10.201.234.34	CAPWAP-Cont…	264	Set	CAPWAP-Control - Discovery Request[Malformed Packet]
88	07:42:45.435367	58.0075…	10.201.166.185	10.201.234.34	DTLSv1.0	117	Set	Client Hello
92	07:42:45.437784	0.002417	10.201.166.185	10.201.234.34	DTLSv1.0	137	Set	Client Hello
98	07:42:45.667215	0.229431	10.201.166.185	10.201.234.34	DTLSv1.0	590	Set	Certificate (Fragment)
99	07:42:45.667260	0.000045	10.201.166.185	10.201.234.34	DTLSv1.0	590	Set	Certificate (Fragment)
100	07:42:45.667293	0.000033	10.201.166.185	10.201.234.34	DTLSv1.0	178	Set	Certificate (Reassembled)
101	07:42:45.667316	0.000023	10.201.166.185	10.201.234.34	DTLSv1.0	329	Set	Client Key Exchange
102	07:42:45.667347	0.000031	10.201.166.185	10.201.234.34	DTLSv1.0	329	Set	Certificate Verify
103	07:42:45.667372	0.000025	10.201.166.185	10.201.234.34	DTLSv1.0	60	Set	Change Cipher Spec
104	07:42:45.667394	0.000022	10.201.166.185	10.201.234.34	DTLSv1.0	123	Set	Encrypted Handshake Message
106	07:42:45.674895	0.007501	10.201.166.185	10.201.234.34	DTLSv1.0	1499	Set	Application Data
107	07:42:45.675288	0.000393	10.201.166.161	10.201.166.185	ICMP	70	Not set,Set	Destination unreachable (Fragmentation needed)
112	07:42:50.671019	4.995731	10.201.166.185	10.201.234.34	DTLSv1.0	411	Set	Application Data
114	07:42:50.718532	0.047513	10.201.166.185	10.201.234.34	DTLSv1.0	539	Set	Application Data
115	07:42:50.718571	0.000039	10.201.166.185	10.201.234.34	DTLSv1.0	539	Set	Application Data

The corresponding AP level Debug ("debug capwap client path-mtu") shows that the AP tried first with 1485 bytes and waited 5 seconds for a response. If no response, it sends another join request packet with a smaller length, as this is still in the join phase and we do not have time to waste. It goes to the minimum value to get the AP to join the WLC, as indicated in the debug log:

```
*Jul 11 18:27:15.000: CAPWAP_PATHMTU: CAPWAP_DTLS_SETUP: MTU = 1485
*Jul 11 18:27:15.000: CAPWAP_PATHMTU: Setting default MTU: MTU discovery can start with 576
*Jul 11 18:27:15.235: %CAPWAP-5-DTLSREQSUCC: DTLS connection created sucessfully peer_ip: 10.201.234.34
*Jul 11 18:27:15.235: CAPWAP_PATHMTU: Sending Join Request Path MTU payload, Length 1376, MTU 576
*Jul 11 18:27:15.235: %CAPWAP-5-SENDJOIN: sending Join Request to 10.201.234.34
...
*Jul 11 18:27:20.235: %CAPWAP-5-SENDJOIN: sending Join Request to 10.201.234.34
*Jul 11 18:27:21.479: %CAPWAP-5-JOINEDCONTROLLER: AP has joined controller c9800-CL
```

And if you run **#show capwap client rcb** in this moment, you see that the CAPWAP AP MTU at 576 bytes:

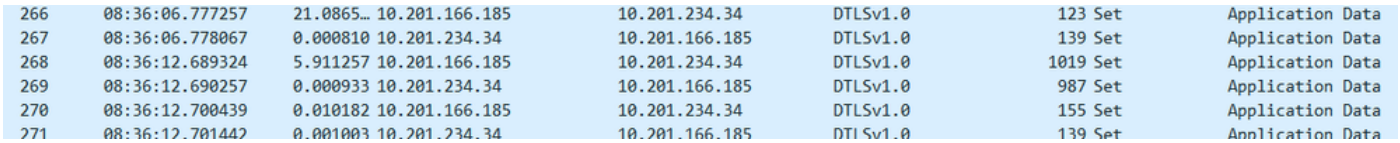
```
3702-AP#show capwap client rcb
AdminState : ADMIN_ENABLED
Primary SwVer : 17.9.3.50
..
MwarName : c9800-CL
MwarApMgrIp : 10.201.234.34
OperationState : JOIN
CAPWAP Path MTU : 576
```

RUN State Phase

After The AP successfully Join the Wireless LAN Controller. you see the PMTU Discovery Mechanism in play, where after 30 seconds you can see the AP start to negotiate a higher PMTU value by sending another CAPWAP Packet with DF bit set of that size of the next highest PMTU value.

In this example, the AP tried 1005 bytes value. Because IOS excludes the Ethernet from the PMTU field, you see 1019 bytes on the wire. If the WLC responds, the AP updates PMTU to1005 bytes. If not, it waits 30 seconds and tries again.

This screenshot displays a successful AP negotiation of 1005 PMTU (see packets #268 and #269). Notice that these packets have different sizes, which is due to the WLC having a different algorithm for PMTU calculation.



266	08:36:06.777257	21.0865…	10.201.166.185	10.201.234.34	DTLSv1.0	123 Set	Application Data
267	08:36:06.778067	0.000810	10.201.234.34	10.201.166.185	DTLSv1.0	139 Set	Application Data
268	08:36:12.689324	5.911257	10.201.166.185	10.201.234.34	DTLSv1.0	1019 Set	Application Data
269	08:36:12.690257	0.000933	10.201.234.34	10.201.166.185	DTLSv1.0	987 Set	Application Data
270	08:36:12.700439	0.010182	10.201.166.185	10.201.234.34	DTLSv1.0	155 Set	Application Data
271	08:36:12.701442	0.001003	10.201.234.34	10.201.166.185	DTLSv1.0	139 Set	Application Data

Here, the corresponding AP level Debug (debug capwap client pmtu) shows where the AP successfully negotiated the 1005 bytes PMTU and updated the AP PMTU value.

```
*Jul 11 18:28:39.911: CAPWAP_PATHMTU: PMTU Timer Expired: Trying to send higher MTU packet 576
*Jul 11 18:28:39.911: CAPWAP_PATHMTU: PMTU Timer:Sending Path MTU packet of size 1005
*Jul 11 18:28:39.911: CAPWAP_PATHMTU: MTU = 1005 for current MTU path discovery
*Jul 11 18:28:39.911: CAPWAP_PATHMTU: Ap Path MTU payload with MTU 1005 sent 888
*Jul 11 18:28:39.911: CAPWAP_PATHMTU: Stopping the message timeout timer
*Jul 11 18:28:39.911: CAPWAP_PATHMTU: Setting MTU to : 1005, it was 576
*Jul 11 18:28:39.911: CAPWAP_PATHMTU: Updating MTU to DPAA
*Jul 11 18:28:39.915: CAPWAP_PATHMTU: Sending MTU update to WLC
*Jul 11 18:28:39.915: CAPWAP_PATHMTU: MTU = 1005 for current MTU path discovery
*Jul 11 18:28:39.915: CAPWAP_PATHMTU: Ap Path MTU payload with MTU 1005 sent 21
```

And if you do (#show capwap client rcb) in this moment you find that the CAPWAP AP MTU at 1005 bytes, Here is the **show** output:

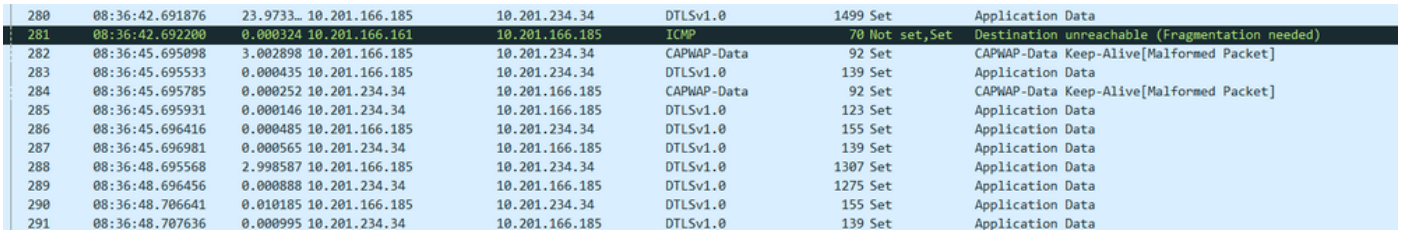
```
3702-AP#show capwap client rcb
AdminState : ADMIN_ENABLED
Primary SwVer : 17.9.3.50
Name : 3702-AP
MwarName : c9800-CL
MwarApMgrIp : 10.201.234.34
OperationState : UP
CAPWAP Path MTU : 1005
```

After 30 seconds, the AP tries again to negotiate the next higher value of 1485 bytes, yet, the AP received **ICMP unreachable** while AP status is in RUN state. The **ICMP unreachable** has a next hop value, and the AP honors this value and uses it to calculate its own PMTU as we can see in the debugs.

```
*Jul 11 18:29:45.911: CAPWAP_PATHMTU: PMTU Timer:Sending Path MTU packet of size 1485
*Jul 11 18:29:45.911: CAPWAP_PATHMTU: MTU = 1485 for current MTU path discovery
*Jul 11 18:29:45.911: CAPWAP_PATHMTU: Ap Path MTU payload with MTU 1485 sent 1368
*Jul 11 18:29:45.911: CAPWAP_PATHMTU: Received ICMP Dst unreachable
*Jul 11 18:29:45.911: CAPWAP_PATHMTU: Src port:5246 Dst Port:60542, SrcAddr:10.201.166.185 Dst Addr:10.
*Jul 11 18:29:45.911: CAPWAP_PATHMTU: Calculated MTU 1293, last_icmp_mtu 1300
*Jul 11 18:29:48.911: CAPWAP_PATHMTU: Path MTU message could not reach WLC, Removing it from the Reliab
```

The Corresponding AP level Captures

Notice the **ICMP unreachable** packet number 281 and then the AP tries to negotiate an PMTU with honoring the ICMP next hop value on 1300 bytes on packets number 288 and response on 289:



280	08:36:42.691876	23.9733…	10.201.166.185	10.201.234.34	DTLSv1.0	1499 Set	Application Data
281	08:36:42.692200	0.000324	10.201.166.161	10.201.166.185	ICMP	70 Not set,Set	Destination unreachable (Fragmentation needed)
282	08:36:45.695098	3.002898	10.201.166.185	10.201.234.34	CAPWAP-Data	92 Set	CAPWAP-Data Keep-Alive[Malformed Packet]
283	08:36:45.695533	0.000435	10.201.166.185	10.201.234.34	DTLSv1.0	139 Set	Application Data
284	08:36:45.695785	0.000252	10.201.234.34	10.201.166.185	CAPWAP-Data	92 Set	CAPWAP-Data Keep-Alive[Malformed Packet]
285	08:36:45.695931	0.000146	10.201.234.34	10.201.166.185	DTLSv1.0	123 Set	Application Data
286	08:36:45.696416	0.000485	10.201.166.185	10.201.234.34	DTLSv1.0	155 Set	Application Data
287	08:36:45.696981	0.000565	10.201.234.34	10.201.166.185	DTLSv1.0	139 Set	Application Data
288	08:36:48.695568	2.998587	10.201.166.185	10.201.234.34	DTLSv1.0	1307 Set	Application Data
289	08:36:48.696456	0.000888	10.201.234.34	10.201.166.185	DTLSv1.0	1275 Set	Application Data
290	08:36:48.706641	0.010185	10.201.166.185	10.201.234.34	DTLSv1.0	155 Set	Application Data
291	08:36:48.707636	0.000995	10.201.234.34	10.201.166.185	DTLSv1.0	139 Set	Application Data

COS AP Behaviour

There are differences in the discovery mechanism for AP-COS APs. We start at **AP join**.

AP Join Phase

At join, the AP sends a Join Request with the maximum value and waits five seconds.

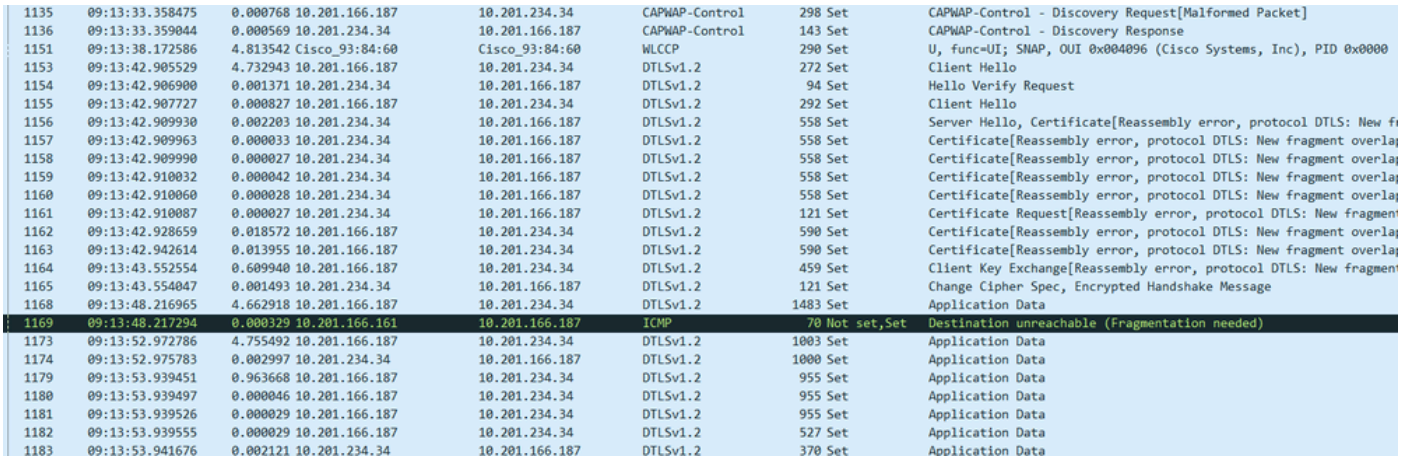
If no response, it tries again and waits another five seconds.

If still no response, it sends another Join Request with 1005 bytes. If that succeeds, it updates PMTU and proceeds (for example, image download). If the 1005-byte DF probe still cannot reach the controller, it drops to the minimum 576 and retries.

Here is the debug capwap client pmtu on the AP level:

```
Jul 11 19:06:10 kernel: [*07/11/2023 19:06:10.7065] AP_PATH_MTU_PAYLOAD_msg_enc_cb: request pmtu 1485, 
Jul 11 19:06:10 kernel: [*07/11/2023 19:06:10.7066] Sending Join request to 10.201.234.34 through port 
Jul 11 19:06:10 kernel: [*07/11/2023 19:06:10.7066] Sending Join Request Path MTU payload, Length 1376
..
Jul 11 19:06:15 kernel: [*07/11/2023 19:06:15.3235] AP_PATH_MTU_PAYLOAD_msg_enc_cb: request pmtu 1485, 
Jul 11 19:06:15 kernel: [*07/11/2023 19:06:15.3235] Sending Join request to 10.201.234.34 through port 
Jul 11 19:06:15 kernel: [*07/11/2023 19:06:15.3235] Sending Join Request Path MTU payload, Length 1376
Jul 11 19:06:15 kernel: [*07/11/2023 19:06:15.3245] chatter: chkcapwapicmpneedfrag :: CheckCapwapICMPNe
..
Jul 11 19:06:20 kernel: [*07/11/2023 19:06:20.0794] AP_PATH_MTU_PAYLOAD_msg_enc_cb: request pmtu 1005, 
Jul 11 19:06:20 kernel: [*07/11/2023 19:06:20.0794] Sending Join request to 10.201.234.34 through port 
Jul 11 19:06:20 kernel: [*07/11/2023 19:06:20.0794] Sending Join Request Path MTU payload, Length 896
Jul 11 19:06:20 kernel: [*07/11/2023 19:06:20.0831] Join Response from 10.201.234.34, packet size 917
Jul 11 19:06:20 kernel: [*07/11/2023 19:06:20.0832] AC accepted previous sent request with result code:
Jul 11 19:06:20 kernel: [*07/11/2023 19:06:20.0832] Received wlcType 0, timer 30
Jul 11 19:06:20 kernel: [*07/11/2023 19:06:20.5280] WLC confirms PMTU 1005, updating MTU now.
Jul 11 19:06:20 kernel: [*07/11/2023 19:06:20.5702] PMTU: Set capwap_init_mtu to TRUE and dcb's mtu to 
Jul 11 19:06:20 kernel: [*07/11/2023 19:06:20.5816] CAPWAP State: Image Data
Jul 11 19:06:20 kernel: [*07/11/2023 19:06:20.5822] AP image version 17.9.3.50 backup 17.6.5.22, Contro
```

Notice that the packet size is 1483 bytes which is the pmtu value without the ethernet header as expected for AP-COS. You see this on packet number 1168 here:



1135	09:13:33.358475	0.000768	10.201.166.187	10.201.234.34	CAPWAP-Control	298 Set	CAPWAP-Control - Discovery Request[Malformed Packet]
1136	09:13:33.359044	0.000569	10.201.234.34	10.201.166.187	CAPWAP-Control	143 Set	CAPWAP-Control - Discovery Response
1151	09:13:38.172586	4.813542	Cisco_93:84:60	Cisco_93:84:60	WLCCP	290 Set	U, func=UI; SNAP, OUI 0x004096 (Cisco Systems, Inc), PID 0x0000
1153	09:13:42.905529	4.732943	10.201.166.187	10.201.234.34	DTLSv1.2	272 Set	Client Hello
1154	09:13:42.906900	0.001371	10.201.234.34	10.201.166.187	DTLSv1.2	94 Set	Hello Verify Request
1155	09:13:42.907727	0.000827	10.201.166.187	10.201.234.34	DTLSv1.2	292 Set	Client Hello
1156	09:13:42.909930	0.002203	10.201.234.34	10.201.166.187	DTLSv1.2	558 Set	Server Hello, Certificate[Reassembly error, protocol DTLS: New f
1157	09:13:42.909963	0.000033	10.201.234.34	10.201.166.187	DTLSv1.2	558 Set	Certificate[Reassembly error, protocol DTLS: New fragment overla
1158	09:13:42.909990	0.000027	10.201.234.34	10.201.166.187	DTLSv1.2	558 Set	Certificate[Reassembly error, protocol DTLS: New fragment overla
1159	09:13:42.910032	0.000042	10.201.234.34	10.201.166.187	DTLSv1.2	558 Set	Certificate[Reassembly error, protocol DTLS: New fragment overla
1160	09:13:42.910060	0.000028	10.201.234.34	10.201.166.187	DTLSv1.2	558 Set	Certificate[Reassembly error, protocol DTLS: New fragment overla
1161	09:13:42.910087	0.000027	10.201.234.34	10.201.166.187	DTLSv1.2	121 Set	Certificate Request[Reassembly error, protocol DTLS: New fragmen
1162	09:13:42.928659	0.018572	10.201.166.187	10.201.234.34	DTLSv1.2	590 Set	Certificate[Reassembly error, protocol DTLS: New fragment overla
1163	09:13:42.942614	0.013955	10.201.166.187	10.201.234.34	DTLSv1.2	590 Set	Certificate[Reassembly error, protocol DTLS: New fragment overla
1164	09:13:43.552554	0.609940	10.201.166.187	10.201.234.34	DTLSv1.2	459 Set	Client Key Exchange[Reassembly error, protocol DTLS: New fragmen
1165	09:13:43.554047	0.001493	10.201.234.34	10.201.166.187	DTLSv1.2	121 Set	Change Cipher Spec, Encrypted Handshake Message
1168	09:13:48.216965	4.662918	10.201.166.187	10.201.234.34	DTLSv1.2	1483 Set	Application Data
1169	09:13:48.217294	0.000329	10.201.166.161	10.201.166.187	ICMP	70 Not set,Set	Destination unreachable (Fragmentation needed)
1173	09:13:52.972786	4.755492	10.201.166.187	10.201.234.34	DTLSv1.2	1003 Set	Application Data
1174	09:13:52.975783	0.002997	10.201.234.34	10.201.166.187	DTLSv1.2	1000 Set	Application Data
1179	09:13:53.939451	0.963668	10.201.166.187	10.201.234.34	DTLSv1.2	955 Set	Application Data
1180	09:13:53.939497	0.000046	10.201.166.187	10.201.234.34	DTLSv1.2	955 Set	Application Data
1181	09:13:53.939526	0.000029	10.201.166.187	10.201.234.34	DTLSv1.2	955 Set	Application Data
1182	09:13:53.939555	0.000029	10.201.166.187	10.201.234.34	DTLSv1.2	527 Set	Application Data
1183	09:13:53.941676	0.002121	10.201.234.34	10.201.166.187	DTLSv1.2	370 Set	Application Data

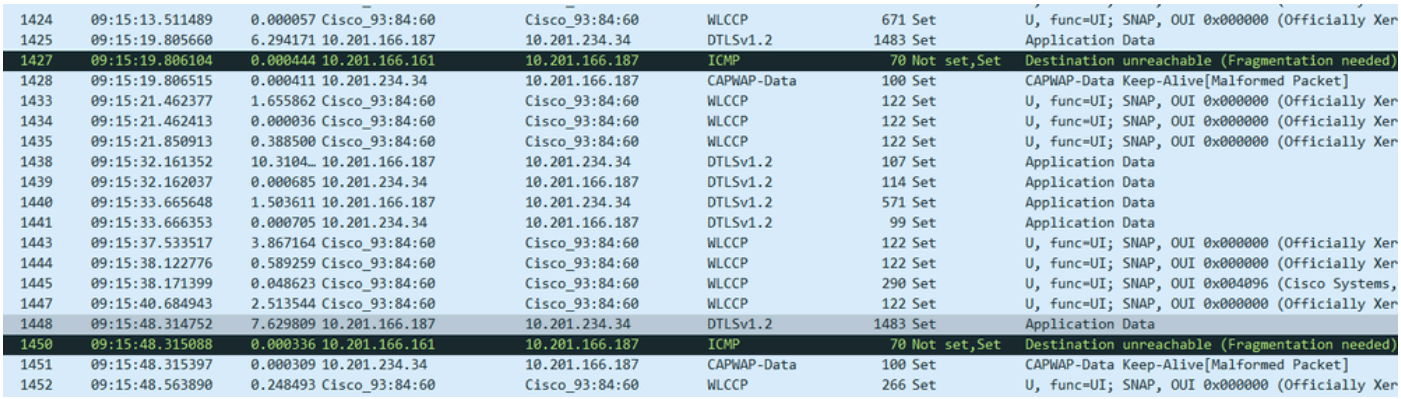
RUN State Phase

After the AP reaches RUN state. it continues attempting to improve the PMTU every 30 seconds, sending CAPWAP packets with DF set and the next hard-coded value.

Here is the AP level debug (debug capwap client pmtu)

```
Jul 11 19:08:15 kernel: [*07/11/2023 19:08:15.1341] wtpEncodePathMTUPayload: Total Packet Size:
Jul 11 19:08:15 kernel: [*07/11/2023 19:08:15.1341] wtpEncodePathMTUPayload: Capwap Size is 137
Jul 11 19:08:15 kernel: [*07/11/2023 19:08:15.1341] [ENC]AP_PATH_MTU_PAYLOAD: pmtu 1485, len 13
Jul 11 19:08:15 kernel: [*07/11/2023 19:08:15.1341] capwap_build_and_send_pmtu_packet: packet l
Jul 11 19:08:15 kernel: [*07/11/2023 19:08:15.1343] Ap Path MTU payload sent, length 1368
Jul 11 19:08:15 kernel: [*07/11/2023 19:08:15.1343] WTP Event Request: AP Path MTU payload sent
Jul 11 19:08:15 kernel: [*07/11/2023 19:08:15.1351] pmtu icmp pkt(ICMP_NEED_FRAG) from click re
Jul 11 19:08:15 kernel: [*07/11/2023 19:08:15.1351] chatter: chkcapwapicmpneedfrag :: CheckCapw
Jul 11 19:08:15 kernel: [*07/11/2023 19:08:15.1351] PMTU data: dcb->mtu 1005, pmtu_overhead:118
Jul 11 19:08:15 kernel: [*07/11/2023 19:08:15.1351] PMTU: Last try for next hop MTU failed
Jul 11 19:08:17 kernel: [*07/11/2023 19:08:17.9850] wtpCleanupPMTUPacket: PMTU: Found matching
..
Jul 11 19:08:43 kernel: [*07/11/2023 19:08:43.6435] wtpEncodePathMTUPayload: Total Packet Size:
Jul 11 19:08:43 kernel: [*07/11/2023 19:08:43.6435] wtpEncodePathMTUPayload: Capwap Size is 137
Jul 11 19:08:43 kernel: [*07/11/2023 19:08:43.6436] [ENC]AP_PATH_MTU_PAYLOAD: pmtu 1485, len 13
Jul 11 19:08:43 kernel: [*07/11/2023 19:08:43.6436] capwap_build-and-send_pmtu_packet: packet l
Jul 11 19:08:43 kernel: [*07/11/2023 19:08:43.6437] Ap Path MTU payload sent, length 1368
Jul 11 19:08:43 kernel: [*07/11/2023 19:08:43.6438] WTP Event Request: AP Path MTU payload sent
Jul 11 19:08:43 kernel: [*07/11/2023 19:08:43.6446] pmtu icmp pkt(ICMP_NEED_FRAG) from click re
Jul 11 19:08:43 kernel: [*07/11/2023 19:08:43.6446] chatter: chkcapwapicmpneedfrag :: CheckCapw
Jul 11 19:08:43 kernel: [*07/11/2023 19:08:43.6446] PMTU data: dcb->mtu 1005, pmtu_overhead:118
Jul 11 19:08:43 kernel: [*07/11/2023 19:08:43.6447] PMTU: Last try for next hop MTU failed
Jul 11 19:08:46 kernel: [*07/11/2023 19:08:46.4945] wtpCleanupPMTUPacket: PMTU: Found matching
```

Here are the corresponding AP captures. look at packet number 1427 and 1448:



1424	09:15:13.511489	0.000057	Cisco_93:84:60	Cisco_93:84:60	WLCCP	671 Set	U, func=UI; SNAP, OUI 0x000000 (Officially Xer
1425	09:15:19.805660	6.294171	10.201.166.187	10.201.234.34	DTLSv1.2	1483 Set	Application Data
1427	09:15:19.806104	0.000444	10.201.166.161	10.201.166.187	ICMP	70 Not set,Set	Destination unreachable (Fragmentation needed)
1428	09:15:19.806515	0.000411	10.201.234.34	10.201.166.187	CAPWAP-Data	100 Set	CAPWAP-Data Keep-Alive[Malformed Packet]
1433	09:15:21.462377	1.655862	Cisco_93:84:60	Cisco_93:84:60	WLCCP	122 Set	U, func=UI; SNAP, OUI 0x000000 (Officially Xer
1434	09:15:21.462413	0.000036	Cisco_93:84:60	Cisco_93:84:60	WLCCP	122 Set	U, func=UI; SNAP, OUI 0x000000 (Officially Xer
1435	09:15:21.850913	0.388500	Cisco_93:84:60	Cisco_93:84:60	WLCCP	122 Set	U, func=UI; SNAP, OUI 0x000000 (Officially Xer
1438	09:15:32.161352	10.3104…	10.201.166.187	10.201.234.34	DTLSv1.2	107 Set	Application Data
1439	09:15:32.162037	0.000685	10.201.234.34	10.201.166.187	DTLSv1.2	114 Set	Application Data
1440	09:15:33.665648	1.503611	10.201.166.187	10.201.234.34	DTLSv1.2	571 Set	Application Data
1441	09:15:33.666353	0.000705	10.201.234.34	10.201.166.187	DTLSv1.2	99 Set	Application Data
1443	09:15:37.533517	3.867164	Cisco_93:84:60	Cisco_93:84:60	WLCCP	122 Set	U, func=UI; SNAP, OUI 0x000000 (Officially Xer
1444	09:15:38.122776	0.589259	Cisco_93:84:60	Cisco_93:84:60	WLCCP	122 Set	U, func=UI; SNAP, OUI 0x000000 (Officially Xer
1445	09:15:38.171399	0.048623	Cisco_93:84:60	Cisco_93:84:60	WLCCP	290 Set	U, func=UI; SNAP, OUI 0x004096 (Cisco Systems,
1447	09:15:40.684943	2.513544	Cisco_93:84:60	Cisco_93:84:60	WLCCP	122 Set	U, func=UI; SNAP, OUI 0x000000 (Officially Xer
1448	09:15:48.314752	7.629809	10.201.166.187	10.201.234.34	DTLSv1.2	1483 Set	Application Data
1450	09:15:48.315088	0.000336	10.201.166.161	10.201.166.187	ICMP	70 Not set,Set	Destination unreachable (Fragmentation needed)
1451	09:15:48.315397	0.000309	10.201.234.34	10.201.166.187	CAPWAP-Data	100 Set	CAPWAP-Data Keep-Alive[Malformed Packet]
1452	09:15:48.563890	0.248493	Cisco_93:84:60	Cisco_93:84:60	WLCCP	266 Set	U, func=UI; SNAP, OUI 0x000000 (Officially Xer

Conclusion (Algorithm Summary)

In summary, the CAPWAP PMTUD algorithm on Access Points works like this.

Step 1. Initial CAPWAP PMTU is negotiated during the AP join phase.

Step 2. 30 seconds later, the AP attempts to improve the current CAPWAP PMTU by sending the next predefined higher value (576, 1005, 1485 bytes).

Step 3 (option 1). If the WLC responds, adjust the current CAPWAP PMTU to the new value and repeat step 2.

Step 3 (option 2). If there is no response, keep the current CAPWAP PMTU and repeat step 2.

Step 3 (option 3). If there is no response and an ICMP Unreachable (Type 3, Code 4) includes a next-hop MTU, adjust the CAPWAP PMTU to that value and repeat step 2.

NOTE: See the fixes to ensure that the right CAPWAP PMTU is used when an ICMP next-hop value is provided.

Related CDETs

Issue Number 1:

Cisco bug ID CSCwf52815

AP-COS APs not honoring the ICMP Unreachable next-hop value when higher-value probes fail.
Fixes: 8.10.190.0, 17.3.8, 17.6.6, 17.9.5, 17.12.2.
IOS APs honor the next-hop value and update PMTU.

Issue Number 2:

Cisco bug ID CSCwc05350

Asymmetric MTU (WLC→AP differs from AP→WLC) led to PMTU flapping when ICMP did not reflect the maximum bidirectional PMTU.
Fixes: 8.10.181.0, 17.3.6, 17.6.5, 17.9.2, 17.10.1.
Workaround: configure the same MTU in both directions on devices controlling MTU (router, firewall, VPN concentrator) between WLC and AP.

Related AP Side Cisco bug ID CSCwc05364: COS-AP improve PMTU mechanism to be able to identify max directional MTU size for asymmetric MTUs

Related WLC side Cisco bug ID CSCwc48316: Improve PMTU calculations for AP to be able to have two different MTUs one upstream and other (marked Closed by DE as they have no plans to address this)

Issue Number 3:

Cisco bug ID CSCwf91557

AP-COS stops PMTU discovery after reaching the maximum hard-coded value.

Fixed in 17.13.1; also via Fixed via Cisco bug ID CSCwf52815 in 17.3.8, 17.6.6, 17.9.5, 17.12.2.

Issue Number 4:

Cisco bug ID CSCwk70785

AP-COS not updating the MTU value for PMTU probe, causing disconnections.

fixed in Cisco bug ID CSCwk90660 - APSP6 17.9.5] Target 17.9.6, 17.12.5, 17.15.2, 17.16.

Issue Number 5:

Cisco bug ID CSCvv53456

9800 Static CAPWAP Path MTU configuration (parity with AireOS).

This allow the 9800 to have a static CAPWAP path MTU configured on a per AP join profile basis. Going into 17.17.