Verifying VoIP traffic prioritization

On a university's campus VoIP Wi-Fi clients had been deployed using a variety of different smartphone Apps. After a short trial the service was deployed to a wider group of staff but quickly found to be unreliable. Users complained of audio problems such as stuttering, periods of silence and poor quality.

The telephony team were unable to recreate the problem but confirmed their testing had taken place in quiet office environments.

The solution used SIP and RTP with a telephony server hosted by a third-party service provider. Whilst the telephony provider had a direct peering with the education and research network used for off-campus internet connectivity, there was no support for end to end QoS across this link.

When implementing VoIP over Wi-Fi on a busy network, prioritizing the latency sensitive traffic is an important part of maintaining the perceived performance and quality. Enhanced Distributed Channel Access (EDCA) is the mechanism used to implement Wi-Fi QoS, allowing clients to increase the chances of higher priority traffic being sent first when contending for access to the wireless medium.

EDCA assigns one of four Access Categories (AC) Background, Best effort, Video or Voice with the default being AC_BE (Best Effort).

Typically, as VoIP traffic arrives at an AP or Wi-Fi controller from the Distribution System (DS), Layer 3 Differentiated Services Code Point (DSCP) markings or the Layer 2 Class of Service (CoS) parameter can be mapped to the appropriate Access Category, in this case AC_VO. Conversely for received wireless traffic being forwarded on the DS, the Access Category is mapped to the appropriate DSCP or CoS. In this instance the absence of end to end QoS meant VoIP traffic arriving at the Wi-Fi controller from the DS didn't have any QoS markings.

The majority of Wi-Fi traffic at busy times was downstream from APs to clients, coinciding with increased internet bandwidth use but monitoring showed no congestion on the offsite link or the LAN. I suspected the lack of QoS markings meant VoIP traffic was not being prioritized by the APs. This possibility had previously been dismissed as an Application Level Gateway (ALG) function on the Wi-Fi controller was supposed to identify the VoIP traffic and prioritize it appropriately.

Over the air capture of VoIP traffic between a smartphone client and AP shows the ALG was not working as expected with VoIP traffic from AP to client being sent as AC_BE.

```
Flags: 0x42
     .... ..10 = DS status: Frame from DS to a STA via AP(To DS: 0 From DS: 1) (0x2)
     .... .0.. = More Fragments: This is the last fragment
     .... 0... = Retry: Frame is not being retransmitted
     ...0 .... = PWR MGT: STA will stay up
     ..0. .... = More Data: No data buffered
     .1.. .... = Protected flag: Data is protected
     0... .... = Order flag: Not strictly ordered
.000 0000 0011 0000 = Duration: 48 microseconds
Receiver address: 96:f8:fd:82:33:e8 (96:f8:fd:82:33:e8)
Transmitter address: HewlettP_66:63:10 (00:4e:35:66:63:10)
Destination address: 96:f8:fd:82:33:e8 (96:f8:fd:82:33:e8)
Source address: HewlettP_f3:69:f0 (84:34:97:f3:69:f0)
BSS Id: HewlettP_66:63:10 (00:4e:35:66:63:10)
STA address: 96:f8:fd:82:33:e8 (96:f8:fd:82:33:e8)
.... .... .... 0000 = Fragment number: 0
0101 0110 0000 .... = Sequence number: 1376
Qos Control: 0x0000
   .... .... .... 0000 = TID: 0
   [.... .... .... .000 = Priority: Best Effort (Best Effort) (0)]
   .... .... ...0 .... = EOSP: Service period
   .... .... .00. .... = Ack Policy: Normal Ack (0x0)
   .... .... 0... .... = Payload Type: MSDU
 0000 0000 .... .... = QAP PS Buffer State: 0x00
```

Figure 1- AP VoIP TX with no prioritization

On further investigation it was found the Wi-Fi controller ALG setting was not active for the role assigned to the VoIP users. After correcting this it was found the VoIP traffic from AP to client was being correctly prioritized.

```
Flags: 0x42
     .... ..10 = DS status: Frame from DS to a STA via AP(To DS: 0 From DS: 1) (0x2)
     .... .0.. = More Fragments: This is the last fragment
     .... 0... = Retry: Frame is not being retransmitted
     ...0 .... = PWR MGT: STA will stay up
     ..0. .... = More Data: No data buffered
     .1.. .... = Protected flag: Data is protected
     0... .... = Order flag: Not strictly ordered
.000 0000 0011 0000 = Duration: 48 microseconds
Receiver address: 96:f8:fd:82:33:e8 (96:f8:fd:82:33:e8)
Transmitter address: HewlettP_66:63:10 (00:4e:35:66:63:10)
Destination address: 96:f8:fd:82:33:e8 (96:f8:fd:82:33:e8)
Source address: HewlettP_f3:69:f0 (84:34:97:f3:69:f0)
BSS Id: HewlettP_66:63:10 (00:4e:35:66:63:10)
STA address: 96:f8:fd:82:33:e8 (96:f8:fd:82:33:e8)
.... .... .... 0000 = Fragment number: 0
0010 0110 1011 .... = Sequence number: 619
Qos Control: 0x0006
   .... .... .... 0110 = TID: 6
   [.... .... .... .110 = Priority: Voice (Voice) (6)]
   .... .... ...0 .... = EOSP: Service period
   .... .... .00. .... = Ack Policy: Normal Ack (0x0)
   .... .... 0... .... = Payload Type: MSDU
 0000 0000 .... .... = QAP PS Buffer State: 0x00
```

Figure 2 - AP VoIP TX with prioritization

Captures showed upstream VoIP traffic from the example iOS client was not bring prioritised.

```
Flags: 0x41
     .... ..01 = DS status: Frame from STA to DS via an AP (To DS: 1 From DS: 0) (0x1)
     .... .0.. = More Fragments: This is the last fragment
     .... 0... = Retry: Frame is not being retransmitted
     ...0 .... = PWR MGT: STA will stay up
     ..0. .... = More Data: No data buffered
     .1.. .... = Protected flag: Data is protected
     0... .... = Order flag: Not strictly ordered
.000 0000 0011 0000 = Duration: 48 microseconds
Receiver address: HewlettP_66:63:10 (00:4e:35:66:63:10)
Transmitter address: 96:f8:fd:82:33:e8 (96:f8:fd:82:33:e8)
Destination address: HewlettP_f3:69:f0 (84:34:97:f3:69:f0)
Source address: 96:f8:fd:82:33:e8 (96:f8:fd:82:33:e8)
BSS Id: HewlettP_66:63:10 (00:4e:35:66:63:10)
STA address: 96:f8:fd:82:33:e8 (96:f8:fd:82:33:e8)
.... .... .... 0000 = Fragment number: 0
0100 0101 0011 .... = Sequence number: 1107
Qos Control: 0x0000
   .... .... .... 0000 = TID: 0
   [.... .... .... .000 = Priority: Best Effort (Best Effort) (0)]
   .... .... ...0 .... = QoS bit 4: Bits 8-15 of QoS Control field are TXOP Duration Requested
   .... .... .00. .... = Ack Policy: Normal Ack (0x0)
   .... .... 0... .... = Payload Type: MSDU
   0000 0000 .... .... = TXOP Duration Requested: 0 (no TXOP requested)
```

Figure 3- iOS client VoIP TX

It was found support for QoS was available in the iOS app and by default this was not enabled. Turning this function on resulted in voice traffic from the client being placed in the video AC. Whilst using the voice AC would be preferred, this is still a higher priority for the upstream traffic.

```
Flags: 0x41
     .... ..01 = DS status: Frame from STA to DS via an AP (To DS: 1 From DS: 0) (0x1)
     .... .0.. = More Fragments: This is the last fragment
     .... 0... = Retry: Frame is not being retransmitted
     ...0 .... = PWR MGT: STA will stay up
     ..0. .... = More Data: No data buffered
     .1.. .... = Protected flag: Data is protected
     0... .... = Order flag: Not strictly ordered
.000 0000 0011 0000 = Duration: 48 microseconds
Receiver address: HewlettP_66:63:10 (00:4e:35:66:63:10)
Transmitter address: 96:f8:fd:82:33:e8 (96:f8:fd:82:33:e8)
Destination address: HewlettP_f3:69:f0 (84:34:97:f3:69:f0)
Source address: 96:f8:fd:82:33:e8 (96:f8:fd:82:33:e8)
BSS Id: HewlettP_66:63:10 (00:4e:35:66:63:10)
STA address: 96:f8:fd:82:33:e8 (96:f8:fd:82:33:e8)
.... .... .... 0000 = Fragment number: 0
1001 0101 1010 .... = Sequence number: 2394
Qos Control: 0x0005
   .... .... .... 0101 = TID: 5
   [.... .... .... .101 = Priority: Video (Video) (5)]
   .... .... ...0 .... = QoS bit 4: Bits 8-15 of QoS Control field are TXOP Duration Requested
   .... .... .00. .... = Ack Policy: Normal Ack (0x0)
   .... .... 0... .... = Payload Type: MSDU
   0000 0000 .... .... = TXOP Duration Requested: 0 (no TXOP requested)
```

Figure 4- iOS client VoIP TX with QoS enabled

Capture of an alternative client using a SIP app on an Android device showed this client was also not prioritizing the voice traffic. There was no QoS functionality in this app.

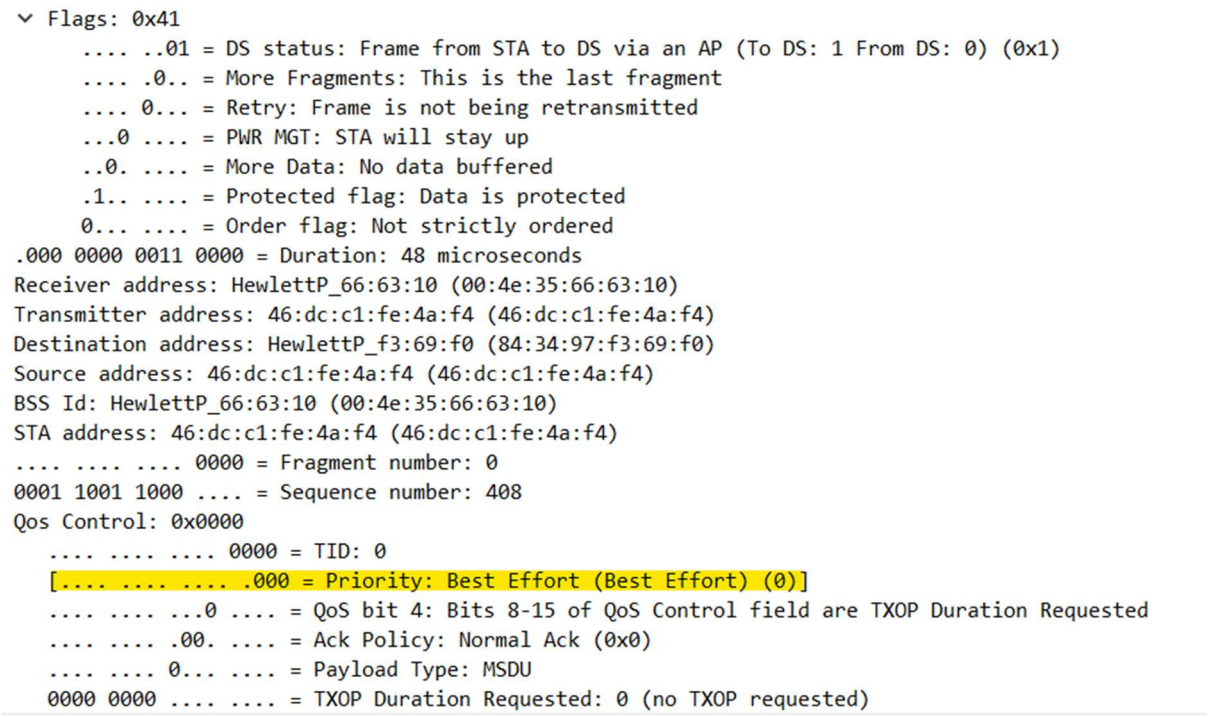

Figure 5 – Android client VoIP TX

After resolving the controller config issue and enabling QoS on the iOS SIP app, VoIP traffic sent from the DS via the AP was using AC_VO and upstream VoIP traffic from the iOS client was using AC_VI. This appeared to resolve the audio quality issues for the iOS app users.

Because the Android clients were still sending voice data using AC_BE, when channel access was heavily contended there remained a risk of latency causing quality issues. This would manifest as audio problems for the remote caller rather than at the Wi-Fi client. It was confirmed the initial audio quality problems were evident in both directions.

Based on these captures I concluded the iOS app, with QoS enabled, was the best solution. The Android app was not capable of supporting QoS and there would likely be a repeat of issues when used on a busy network.