

Assignment 3

COMP6331 – COMPUTER NETWORKS

SYED ALI HUSSAIN

U6028474

Question 1

The three channels were subscribed using matching QoS and the data capture with Wireshark for each of them are shown below. Filter was applied to show the traffic between 3310exp.hopto.org (broker) and the client.

Slow Counter q0

The command used to subscribe to the first counter, Q0, is:

```
$ ./mosquito_sub -i 3310-6028474 -L
mqtt://3310student:comp3310@3310exp.hopto.org/counter/slow/q0 -d -k 5
-C 10 -q 0
```

Acknowledge:

No.	Time	Source	Destination	Protocol	Length	Info
168	3.137890247	192.168.111.4	52.65.194.50	TCP	74	36990 → 1883 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=39977...
169	3.206646052	52.65.194.50	192.168.111.4	TCP	74	1883 → 36990 [SYN, ACK] Seq=0 Ack=1 Win=26847 Len=0 MSS=1460 SACK_PERM=1 ...
170	3.206677761	192.168.111.4	52.65.194.50	TCP	66	36990 → 1883 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=3997726329 TSecr=451...
171	3.206773287	192.168.111.4	52.65.194.50	MQTT	115	Connect Command
172	3.207710023	52.65.194.50	192.168.111.4	TCP	66	1883 → 36990 [ACK] Seq=1 Ack=50 Win=26847 Len=0 TSval=45107050 TSecr=3997...

▶ Frame 171: 115 bytes on wire (920 bits), 115 bytes captured (920 bits) on interface 0

▶ Ethernet II, Src: Dell_27:fa:db (14:b3:1f:27:fa:db), Dst: IntelCor_d9:b1:72 (00:15:17:d9:b1:72)

▶ Internet Protocol Version 4, Src: 192.168.111.4, Dst: 52.65.194.50

▶ Transmission Control Protocol, Src Port: 36990, Dst Port: 1883, Seq: 1, Ack: 1, Len: 49

▼ MQ Telemetry Transport Protocol, Connect Command

▼ Header Flags: 0x10 (Connect Command)

0001 = Message Type: Connect Command (1)

.... 0... = DUP Flag: Not set

.... 00. = QoS Level: At most once delivery (Fire and Forget) (0)

.... ...0 = Retain: Not set

Msg Len: 47

Protocol Name Length: 4

Protocol Name: MQTT

Version: 4

▼ Connect Flags: 0xc2

1... = User Name Flag: Set

.1. = Password Flag: Set

..0. = Will Retain: Not set

...0 0... = QoS Level: At most once delivery (Fire and Forget) (0)

.... 0... = Will Flag: Not set

.... .1. = Clean Session Flag: Set

.... ...0 = (Reserved): Not set

Keep Alive: 5

Client ID Length: 12

Client ID: 3310-6028474

User Name Length: 11

User Name: 3310student

Password Length: 8

Password: comp3310

We can see that the MQTT is using the TCP to communicate with the broker. First of the MQTT commands is a Connect command which is sent from the client to the broker. The first four bits of the header flag, 0001, show this in the figure above. The next four bits, 0000, show that the Duplicate (DUP) is not set, Quality of Service (QoS) is set at 00 which is Fire and Forget, and the Retain flag is not set also.

Connect Acknowledge:

No.	Time	Source	Destination	Protocol	Length	Info
171	3.296773287	192.168.111.4	52.65.194.50	MQTT	115	Connect Command
172	3.301718822	52.65.194.50	192.168.111.4	MQTT	86	1883 → 36990 [ACK] Seq=72 Ack=10 Len=4
173	3.388316822	52.65.194.50	192.168.111.4	MQTT	79	Connect Ack
174	3.388316822	192.168.111.4	52.65.194.50	MQTT	86	36990 → 1883 [ACK] Seq=0 Ack=5 Len=4
175	3.388316822	192.168.111.4	52.65.194.50	MQTT	88	Subscribe Request
Frame 173: 79 bytes on wire (632 bits), 79 bytes captured (632 bits) on interface 0 Ethernet II, Src: IntelCor_d9:b1:72 (00:15:17:d9:b1:72), Dst: Dell_27:fa:db (14:b3:1f:27:fa:db) Internet Protocol Version 4, Src: 52.65.194.50, Dst: 192.168.111.4 Transmission Control Protocol, Src Port: 1883, Dst Port: 36990, Seq: 1, Ack: 50, Len: 4 MQ Telemetry Transport Protocol, Connect Ack: Header Flags: 0x20 (Connect Ack) 0010... = Message Type: Connect Ack (2) 0... = DUP Flag: Not set 00... = QoS Level: At most once delivery (Fire and Forget) (0) 0... = Retain: Not set Msg Len: 2 Acknowledge Flags: 0x00 0000:000... = Reserved: Not set 0... = Session Present: Not set Return Code: Connection Accepted (0)						

The second of the command is sent from the broker to the client and is a Connect Acknowledge. The return code is 0 which means the broker has authorized and the username and password provided in the Connect command before were valid.

Subscribe Request:

175	3.288326229	192.168.111.4	52.65.194.50	MQTT	88	Subscribe Request
176	3.368980738	52.65.194.50	192.168.111.4	MQTT	71	Subscribe Ack
177	3.412824409	192.168.111.4	52.65.194.50	TCP	66	36990 → 1883 [ACK] Seq=72 Ack=10
178	3.532057670	52.65.194.50	192.168.111.4	MQTT	88	Publish Message
179	3.532094064	192.168.111.4	52.65.194.50	TCP	66	36990 → 1883 [ACK] Seq=72 Ack=32
Frame 175: 88 bytes on wire (704 bits), 88 bytes captured (704 bits) on interface 0 Ethernet II, Src: Dell_27:fa:db (14:b3:1f:27:fa:db), Dst: IntelCor_d9:b1:72 (00:15:17:d9:b1:72) Internet Protocol Version 4, Src: 192.168.111.4, Dst: 52.65.194.50 Transmission Control Protocol, Src Port: 36990, Dst Port: 1883, Seq: 50, Ack: 5, Len: 22 MQ Telemetry Transport Protocol, Subscribe Request Header Flags: 0x82 (Subscribe Request) 1000... = Message Type: Subscribe Request (8) 0... = DUP Flag: Not set 01... = QoS Level: At least once delivery (Acknowledged deliver) (1) 0... = Retain: Not set Msg Len: 20 Message Identifier: 1 Topic Length: 15 Topic: counter/slow/q0 Requested QoS: At most once delivery (Fire and Forget) (0)						

When the client receives the Connect Acknowledge command it sends a Subscribe Request as the third command. The QoS level is set at 01 to make sure the request is delivered at least once to the broker. The topic requested can also be seen as 'counter/slow/q0'.

Subscribe Ack:

176	3.368980738	52.65.194.50	192.168.111.4	MQTT	71 Subscribe Ack
177	3.412824409	192.168.111.4	52.65.194.50	TCP	66 36990 → 1883 [ACK] Seq=72 Ack
178	3.532057670	52.65.194.50	192.168.111.4	MQTT	88 Publish Message
179	3.532094064	192.168.111.4	52.65.194.50	TCP	66 36990 → 1883 [ACK] Seq=72 Ack

▶	Frame 176: 71 bytes on wire (568 bits), 71 bytes captured (568 bits) on interface 0
▶	Ethernet II, Src: IntelCor_d9:b1:72 (00:15:17:d9:b1:72), Dst: Dell_27:fa:db (14:b3:1f:27:fa:db)
▶	Internet Protocol Version 4, Src: 52.65.194.50, Dst: 192.168.111.4
▶	Transmission Control Protocol, Src Port: 1883, Dst Port: 36990, Seq: 5, Ack: 72, Len: 5
▼	MQ Telemetry Transport Protocol, Subscribe Ack
▼	Header Flags: 0x90 (Subscribe Ack)
	1001 = Message Type: Subscribe Ack (9)
 0... = DUP Flag: Not set
00. = QoS Level: At most once delivery (Fire and Forget) (0)
0 = Retain: Not set
	Msg Len: 3
	Message Identifier: 1
	Granted QoS: At most once delivery (Fire and Forget) (0)

The server replies with the Subscriber Acknowledge request that has a length of 71.

Publish:

270	13.186135586	52.65.194.50	192.168.111.4	MQTT	88 Publish Message
271	13.186171039	192.168.111.4	52.65.194.50	TCP	66 36990 → 1883 [ACK] Seq=76 Ack=234 Win=29312
272	13.186334943	192.168.111.4	52.65.194.50	MQTT	68 Disconnect Req
273	13.186367221	192.168.111.4	52.65.194.50	TCP	66 36990 → 1883 [FIN, ACK] Seq=78 Ack=234 Win=
274	13.271822760	52.65.194.50	192.168.111.4	TCP	66 1883 → 36990 [FIN, ACK] Seq=234 Ack=79 Win=
275	13.271867077	192.168.111.4	52.65.194.50	TCP	66 36990 → 1883 [ACK] Seq=79 Ack=235 Win=29312

▶	Frame 270: 88 bytes on wire (704 bits), 88 bytes captured (704 bits) on interface 0
▶	Ethernet II, Src: IntelCor_d9:b1:72 (00:15:17:d9:b1:72), Dst: Dell_27:fa:db (14:b3:1f:27:fa:db)
▶	Internet Protocol Version 4, Src: 52.65.194.50, Dst: 192.168.111.4
▶	Transmission Control Protocol, Src Port: 1883, Dst Port: 36990, Seq: 212, Ack: 76, Len: 22
▼	MQ Telemetry Transport Protocol, Publish Message
▼	Header Flags: 0x30 (Publish Message)
	0011 = Message Type: Publish Message (3)
 0... = DUP Flag: Not set
00. = QoS Level: At most once delivery (Fire and Forget) (0)
0 = Retain: Not set
	Msg Len: 20
	Topic Length: 15
	Topic: counter/slow/q0
	Message: 315

After the client has subscribed to the topic 'counter/slow/q0', each time the key-value pair at the broker is updated, a publish message is sent to all the subscribing client. The figure above shows one of such messages.

Disconnect:

In the end, when the client wants to disconnect with the broker, it sends a disconnect message. After this, the connection is terminated. The figure below shows the disconnect message.

272	13.186334943	192.168.111.4	52.65.194.50	MQTT	68 Disconnect Req
273	13.186367221	192.168.111.4	52.65.194.50	TCP	66 36990 → 1883 [FIN, ACK] S
274	13.271822760	52.65.194.50	192.168.111.4	TCP	66 1883 → 36990 [FIN, ACK] S
275	13.271867077	192.168.111.4	52.65.194.50	TCP	66 36990 → 1883 [ACK] Seq=79

```

▶ Frame 272: 68 bytes on wire (544 bits), 68 bytes captured (544 bits) on interface 0
▶ Ethernet II, Src: Dell_27:fa:db (14:b3:1f:27:fa:db), Dst: IntelCor_d9:b1:72 (00:15:17:d9:b1:72)
▶ Internet Protocol Version 4, Src: 192.168.111.4, Dst: 52.65.194.50
▶ Transmission Control Protocol, Src Port: 36990, Dst Port: 1883, Seq: 76, Ack: 234, Len: 2
▼ MQ Telemetry Transport Protocol, Disconnect Req
  ▼ Header Flags: 0xe0 (Disconnect Req)
    1110 .... = Message Type: Disconnect Req (14)
    .... 0... = DUP Flag: Not set
    .... 00.. = QoS Level: At most once delivery (Fire and Forget) (0)
    .... ...0 = Retain: Not set
  Msg Len: 0

```

Slow Counter q1

The slow counter uses QoS 1 and we will subscribe to it using this QoS. We will only however compare the differences in the MQTT messages send and received to save space as much of the handshake messages sent and received are similar to counter q0. The figure below shows all the MQTT and TCP messages sent and received between the client and the broker.

No.	Time	Source	Destination	Protocol	Length	Info
8	0.426831445	192.168.111.4	52.65.194.50	TCP	74	37690 → 1883 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=4011549462 ...
9	0.442574629	52.65.194.50	192.168.111.4	TCP	74	1883 → 37690 [SYN, ACK] Seq=0 Ack=1 Win=26847 Len=0 MSS=1460 SACK_PERM=1 TSval=...
10	0.442630728	192.168.111.4	52.65.194.50	TCP	66	37690 → 1883 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=4011549478 TSecr=48563762
11	0.442770640	192.168.111.4	52.65.194.50	MQTT	115	Connect Command
12	0.523510061	52.65.194.50	192.168.111.4	TCP	66	1883 → 37690 [ACK] Seq=1 Ack=50 Win=26880 Len=0 TSval=48563783 TSecr=4011549478
13	0.523944647	52.65.194.50	192.168.111.4	MQTT	70	Connect Ack
14	0.523977765	192.168.111.4	52.65.194.50	TCP	66	37690 → 1883 [ACK] Seq=50 Ack=5 Win=29312 Len=0 TSval=4011549559 TSecr=48563783
15	0.524152076	192.168.111.4	52.65.194.50	MQTT	88	Subscribe Request
16	0.685811372	52.65.194.50	192.168.111.4	TCP	66	1883 → 37690 [ACK] Seq=5 Ack=72 Win=26880 Len=0 TSval=48563823 TSecr=4011549560
17	0.771415275	52.65.194.50	192.168.111.4	MQTT	71	Subscribe Ack
18	0.811935169	192.168.111.4	52.65.194.50	TCP	66	37690 → 1883 [ACK] Seq=72 Ack=10 Win=29312 Len=0 TSval=4011549807 TSecr=48563845
21	1.178019289	52.65.194.50	192.168.111.4	MQTT	90	Publish Message
22	1.178051964	192.168.111.4	52.65.194.50	TCP	66	37690 → 1883 [ACK] Seq=72 Ack=34 Win=29312 Len=0 TSval=4011550213 TSecr=48563946
23	1.178222822	192.168.111.4	52.65.194.50	MQTT	70	Publish Ack
24	1.262565434	52.65.194.50	192.168.111.4	TCP	66	1883 → 37690 [ACK] Seq=34 Ack=76 Win=26880 Len=0 TSval=48563967 TSecr=4011550214
26	2.246117703	52.65.194.50	192.168.111.4	MQTT	90	Publish Message
27	2.246236823	192.168.111.4	52.65.194.50	MQTT	70	Publish Ack
28	2.326522138	52.65.194.50	192.168.111.4	TCP	66	1883 → 37690 [ACK] Seq=58 Ack=80 Win=26880 Len=0 TSval=48564233 TSecr=4011551282
29	3.394503921	52.65.194.50	192.168.111.4	MQTT	90	Publish Message
30	3.394663166	192.168.111.4	52.65.194.50	MQTT	70	Publish Ack
31	3.474878049	52.65.194.50	192.168.111.4	TCP	66	1883 → 37690 [ACK] Seq=82 Ack=84 Win=26880 Len=0 TSval=48564520 TSecr=4011552430
45	4.377897639	52.65.194.50	192.168.111.4	MQTT	90	Publish Message
46	4.378052435	192.168.111.4	52.65.194.50	MQTT	70	Publish Ack
47	4.378137348	192.168.111.4	52.65.194.50	MQTT	68	Disconnect Req
48	4.457927220	52.65.194.50	192.168.111.4	TCP	66	1883 → 37690 [ACK] Seq=106 Ack=88 Win=26880 Len=0 TSval=48564766 TSecr=40115534...
49	4.457960251	52.65.194.50	192.168.111.4	TCP	66	1883 → 37690 [FIN, ACK] Seq=106 Ack=91 Win=26880 Len=0 TSval=48564766 TSecr=401...
50	4.457981550	192.168.111.4	52.65.194.50	TCP	66	37690 → 1883 [ACK] Seq=91 Ack=107 Win=29312 Len=0 TSval=4011553493 TSecr=485647...

```

▶ Frame 15: 88 bytes on wire (704 bits), 88 bytes captured (704 bits) on interface 0
▶ Ethernet II, Src: Dell_27:fa:db (14:b3:1f:27:fa:db), Dst: IntelCor_d9:b1:72 (00:15:17:d9:b1:72)
▶ Internet Protocol Version 4, Src: 192.168.111.4, Dst: 52.65.194.50
▶ Transmission Control Protocol, Src Port: 37690, Dst Port: 1883, Seq: 50, Ack: 5, Len: 22
▼ MQ Telemetry Transport Protocol, Subscribe Request
  ▼ Header Flags: 0x82 (Subscribe Request)
    1000 .... = Message Type: Subscribe Request (8)
    .... 0... = DUP Flag: Not set
    .... 01.. = QoS Level: At least once delivery (Acknowledged deliver) (1)
    .... ...0 = Retain: Not set
  Msg Len: 20
  Message Identifier: 1
  Topic Length: 15
  Topic: counter/slow/q1
  Requested QoS: At least once delivery (Acknowledged deliver) (1)

```

The client sends a Connect command to the broker and is replied a Connect Ack command. After that a Subscribe Request is sent with the QoS flag set to 01 as shown in the figure above which is at least once delivery.

Publish and Publish Acknowledge:

21	1.178019289	52.65.194.50	192.168.111.4	MQTT	90 Publish Message
22	1.178051964	192.168.111.4	52.65.194.50	TCP	66 37690 → 1883 [ACK] Seq=72
23	1.178222822	192.168.111.4	52.65.194.50	MQTT	70 Publish Ack
24	1.262565434	52.65.194.50	192.168.111.4	TCP	66 1883 → 37690 [ACK] Seq=34
26	2.246117703	52.65.194.50	192.168.111.4	MQTT	90 Publish Message
27	2.246236823	192.168.111.4	52.65.194.50	MQTT	70 Publish Ack
28	2.326522138	52.65.194.50	192.168.111.4	TCP	66 1883 → 37690 [ACK] Seq=58
29	3.394503921	52.65.194.50	192.168.111.4	MQTT	90 Publish Message
30	3.394663166	192.168.111.4	52.65.194.50	MQTT	70 Publish Ack

▶ Frame 21: 90 bytes on wire (720 bits), 90 bytes captured (720 bits) on interface 0 ▶ Ethernet II, Src: IntelCor_d9:b1:72 (00:15:17:d9:b1:72), Dst: Dell_27:fa:db (14:b3:1f:27:fa:db) ▶ Internet Protocol Version 4, Src: 52.65.194.50, Dst: 192.168.111.4 ▶ Transmission Control Protocol, Src Port: 1883, Dst Port: 37690, Seq: 10, Ack: 72, Len: 24 ▼ MQ Telemetry Transport Protocol, Publish Message ▼ Header Flags: 0x32 (Publish Message) 0011 = Message Type: Publish Message (3) 0... = DUP Flag: Not set 01. = QoS Level: At least once delivery (Acknowledged deliver) (1)0 = Retain: Not set Msg Len: 22 Topic Length: 15 Topic: counter/slow/q1 Message Identifier: 1 Message: 268					
--	--	--	--	--	--

23	1.178222822	192.168.111.4	52.65.194.50	MQTT	70 Publish Ack
24	1.262565434	52.65.194.50	192.168.111.4	TCP	66 1883 → 37690 [ACK] Seq=34
26	2.246117703	52.65.194.50	192.168.111.4	MQTT	90 Publish Message
27	2.246236823	192.168.111.4	52.65.194.50	MQTT	70 Publish Ack
28	2.326522138	52.65.194.50	192.168.111.4	TCP	66 1883 → 37690 [ACK] Seq=58
29	3.394503921	52.65.194.50	192.168.111.4	MQTT	90 Publish Message
30	3.394663166	192.168.111.4	52.65.194.50	MQTT	70 Publish Ack

▶ Frame 23: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0 ▶ Ethernet II, Src: Dell_27:fa:db (14:b3:1f:27:fa:db), Dst: IntelCor_d9:b1:72 (00:15:17:d9:b1:72) ▶ Internet Protocol Version 4, Src: 192.168.111.4, Dst: 52.65.194.50 ▶ Transmission Control Protocol, Src Port: 37690, Dst Port: 1883, Seq: 72, Ack: 34, Len: 4 ▼ MQ Telemetry Transport Protocol, Publish Ack ▼ Header Flags: 0x40 (Publish Ack) 0100 = Message Type: Publish Ack (4) 0... = DUP Flag: Not set 00. = QoS Level: At most once delivery (Fire and Forget) (0)0 = Retain: Not set Msg Len: 2 Message Identifier: 1					
--	--	--	--	--	--

After the subscription is successful, a Publish message is sent by the broker to the client. We can see from the figure above that the QoS flag is set to 01 which ensures that the client receives the message at least once. For this reason, every Publish message from the broker to the client is followed by a Publish Acknowledge message. Also, each Publish message has a message identifier which is replied in the Publish Ack to tell broker that the message has been received. In the end the client and broker disconnect as q0.

Slow Counter q2

The slow counter q2 uses QoS 2 which is ‘Exactly Once’. The figure below shows the MQTT messages sent and received between the client and the broker. Again, we will discuss the differences in MQTT messages with respect to QoS.

151	4.250200477	192.168.111.4	52.65.194.50	TCP	74	37956 → 1883 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=4013987305 ...
152	4.300886525	52.65.194.50	192.168.111.4	TCP	74	1883 → 37956 [SYN, ACK] Seq=0 Ack=1 Win=26847 Len=0 MSS=1460 SACK_PERM=1 TSval=...
153	4.300946306	192.168.111.4	52.65.194.50	TCP	66	37956 → 1883 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=4013987355 TSecr=49173235
154	4.301080834	192.168.111.4	52.65.194.50	MQTT	115	Connect Command
155	4.385848780	52.65.194.50	192.168.111.4	TCP	66	1883 → 37956 [ACK] Seq=1 Ack=50 Win=26880 Len=0 TSval=49173257 TSecr=4013987356
156	4.386642825	52.65.194.50	192.168.111.4	MQTT	70	Connect Ack
157	4.386668492	192.168.111.4	52.65.194.50	TCP	66	37956 → 1883 [ACK] Seq=50 Ack=5 Win=29312 Len=0 TSval=4013987441 TSecr=49173257
158	4.386811241	192.168.111.4	52.65.194.50	MQTT	88	Subscribe Request
159	4.467029431	52.65.194.50	192.168.111.4	MQTT	71	Subscribe Ack
160	4.507498103	192.168.111.4	52.65.194.50	TCP	66	37956 → 1883 [ACK] Seq=72 Ack=10 Win=29312 Len=0 TSval=4013987522 TSecr=49173277
161	4.710650232	52.65.194.50	192.168.111.4	MQTT	90	Publish Message
162	4.710695583	192.168.111.4	52.65.194.50	TCP	66	37956 → 1883 [ACK] Seq=72 Ack=34 Win=29312 Len=0 TSval=4013987765 TSecr=49173338
163	4.710823771	192.168.111.4	52.65.194.50	MQTT	70	Publish Received
164	4.876387402	52.65.194.50	192.168.111.4	TCP	66	1883 → 37956 [ACK] Seq=34 Ack=76 Win=26880 Len=0 TSval=49173379 TSecr=4013987765
165	4.876956146	52.65.194.50	192.168.111.4	MQTT	70	Publish Release
166	4.877070578	192.168.111.4	52.65.194.50	MQTT	70	Publish Complete
167	4.957954550	52.65.194.50	192.168.111.4	TCP	66	1883 → 37956 [ACK] Seq=38 Ack=80 Win=26880 Len=0 TSval=49173400 TSecr=4013987932
178	5.693389655	52.65.194.50	192.168.111.4	MQTT	90	Publish Message
179	5.693542691	192.168.111.4	52.65.194.50	MQTT	70	Publish Received
180	5.773825798	52.65.194.50	192.168.111.4	TCP	66	1883 → 37956 [ACK] Seq=62 Ack=84 Win=26880 Len=0 TSval=49173604 TSecr=4013988748
181	5.774077722	52.65.194.50	192.168.111.4	MQTT	70	Publish Release
182	5.774209960	192.168.111.4	52.65.194.50	MQTT	70	Publish Complete
183	5.940407845	52.65.194.50	192.168.111.4	TCP	66	1883 → 37956 [ACK] Seq=66 Ack=88 Win=26880 Len=0 TSval=49173645 TSecr=4013988829
185	6.757568773	52.65.194.50	192.168.111.4	MQTT	90	Publish Message
186	6.757713198	192.168.111.4	52.65.194.50	MQTT	70	Publish Received
187	6.842151052	52.65.194.50	192.168.111.4	TCP	66	1883 → 37956 [ACK] Seq=90 Ack=92 Win=26880 Len=0 TSval=49173871 TSecr=4013989812
188	6.842391939	52.65.194.50	192.168.111.4	MQTT	70	Publish Release
189	6.842548758	192.168.111.4	52.65.194.50	MQTT	70	Publish Complete
190	7.004489643	52.65.194.50	192.168.111.4	TCP	66	1883 → 37956 [ACK] Seq=94 Ack=96 Win=26880 Len=0 TSval=49173911 TSecr=4013989897
191	7.906891377	52.65.194.50	192.168.111.4	MQTT	90	Publish Message
192	7.907050065	192.168.111.4	52.65.194.50	MQTT	70	Publish Received
193	7.987522182	52.65.194.50	192.168.111.4	TCP	66	1883 → 37956 [ACK] Seq=118 Ack=100 Win=26880 Len=0 TSval=49174157 TSecr=4013990...
195	8.073050787	52.65.194.50	192.168.111.4	MQTT	70	Publish Release
196	8.073215438	192.168.111.4	52.65.194.50	MQTT	68	Disconnect Req
197	8.073251419	192.168.111.4	52.65.194.50	TCP	66	37956 → 1883 [FIN, ACK] Seq=102 Ack=122 Win=29312 Len=0 TSval=4013991128 TSecr=...
198	8.154242911	52.65.194.50	192.168.111.4	TCP	66	1883 → 37956 [ACK] Seq=122 Ack=102 Win=26880 Len=0 TSval=49174199 TSecr=4013991...
199	8.154275597	52.65.194.50	192.168.111.4	TCP	66	1883 → 37956 [FIN, ACK] Seq=122 Ack=103 Win=26880 Len=0 TSval=49174199 TSecr=40...
200	8.154294124	192.168.111.4	52.65.194.50	TCP	66	37956 → 1883 [ACK] Seq=103 Ack=123 Win=29312 Len=0 TSval=4013991209 TSecr=49174...

▶ Frame 159: 71 bytes on wire (568 bits), 71 bytes captured (568 bits) on interface 0
▶ Ethernet II, Src: IntelCor_d9:b1:72 (00:15:17:d9:b1:72), Dst: Dell_27:fa:db (14:b3:1f:27:fa:db)
▶ Internet Protocol Version 4, Src: 52.65.194.50, Dst: 192.168.111.4
▶ Transmission Control Protocol, Src Port: 1883, Dst Port: 37956, Seq: 5, Ack: 72, Len: 5
▼ MQ Telemetry Transport Protocol, Subscribe Ack
 ▼ Header Flags: 0x90 (Subscribe Ack)
 1001 = Message Type: Subscribe Ack (9)
 0... = DUP Flag: Not set
 00. = QoS Level: At most once delivery (Fire and Forget) (0)
 0 = Retain: Not set
 Msg Len: 3
 Message Identifier: 1
 Granted QoS: Exactly once delivery (Assured Delivery) (2)

The Connect command follows with the usual Connect Acknowledge command. It can be seen that QoS 2 is granted to the client in the Subscriber Ack.

Publish:

Each of the Publish Message sent from the broker is replied with Publish Received message by the client. The broker then sends a Publish Release command and the client replies with the Publish Complete command ensuring that the message is received exactly once. Each of the messages have the Message Identifier.

Summary:

For QoS 0, the message duplication bit carries no significance. The broker pushes out the message as soon as it receives. The message from the broker received do not necessarily have to be in order therefore. It will be used where the data will not severely effect if it is lost. For example, if a temperature sensor is sending the data at a very fast rate, a few packets lost will not make a significant difference.

For QoS 1, The broker ensures that the client receives the message at least once and if the client does not receive, the message is sent again with the duplicate bit set. The message is deleted after the receiver processes it and sends an acknowledgement to the sender who then deletes it.

In QoS 2, the message is delivered exactly once, and this is the safest mode of transfer. Two pairs of transmission between the sender and receiver are used for this. In the first, sender sends the message to the receiver telling it that it has stored the message. Once the receiver replies with an acknowledgement a PUBREL is sent by the sender that tells the receiver that it can complete processing the message. When the client sends the acknowledgement of the PUBREL, sender deletes the message.

Question 2(a)

The following statistics for different QoS was collected from one run of the code:

QoS: 0

1. Rate of messages received: 3485
2. Rate of messages lost: 0
3. Rate of duplicated messages: 0
4. Rate of out-of-order messages: 0

QoS: 1

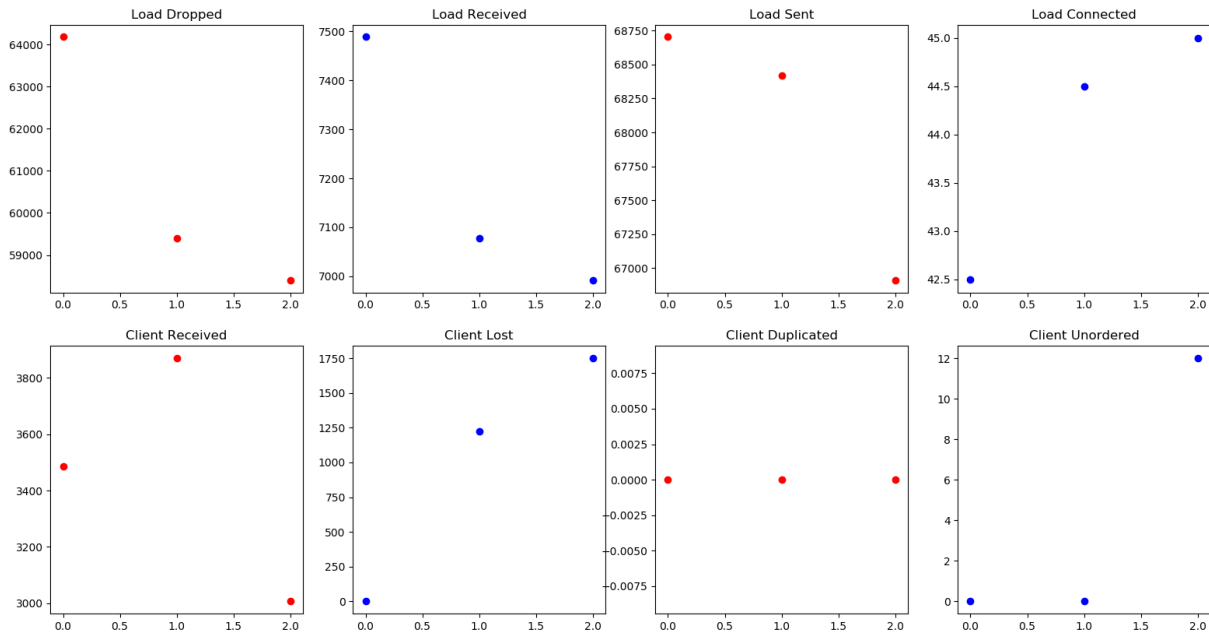
1. Rate of messages received: 3870
2. Rate of messages lost: 1226
3. Rate of duplicated messages: 0
4. Rate of out-of-order messages: 0

QoS: 2

1. Rate of messages received: 3007
2. Rate of messages lost: 1751
3. Rate of duplicated messages: 0
4. Rate of out-of-order messages: 12

Question 2(b)

The following figure shows how broker and client perform for different QoS for one session the client is connected to the broker. The first row shows the messages dropped, received, sent and number of active client on the broker in one minute from left to right. The second row shows messages received, lost, duplicated and unordered in one minute that the client experiences.



We can see that the dropped messages and the received messages on the broker are correlated. This is expected as the broker will lose more messages if the publishers want to send more data. If they send less data, the broker is less likely to lose messages. Another correlation we can expect is between number of clients connected to the broker and the rate of message loss by the client. As the number of active clients subscribed increase, we would expect the client to lose more messages. This is proven in the figure above as the number of clients increase from QoS 0 to 2, more messages are lost by the client.

Question 4

Results were compared across loss/dupe/out-of-order for the timestamp 152696xxxx for similar time comparison. The table below summarizes the values for each of the messages

Sr. No	Loss QoS (1)	Dupe	Out-of-order
1	88	24	4.15
2	47	20	1.39
3	0	12	1.77
4	59	2.38	6
5	14	24.82	1.6
6	68	15.4	8.38
7	40	2.66	1
8	35	0	10

9	39	13.64	11.59
10	36	13	1.9

There is similarity between the different rates although some extreme values can be seen as well.

No correlation in performance is seen from the different use of language.

Question 5

- a. In the broader end-to-end environment, there is a limit to which the network's performance could be extended. A CPU is one element in an MQTT network and performs processing tasks such as updating the key-value pairs when a message is received from the publishers and retrieving when subscribers need it. If processing is slow then there will be delays in subscribers receiving information and publishers would have to wait. Memory especially heap memory is another element which allows for quick update of key-value pairs of MQTT. Heap memory is dynamic and provides for quick access to key-value pairs. Amount of memory is directly proportional to different number of topics that need updating. If new topics are added, heap memory will be consumed faster. This is especially important if the messages sent to the broker have the 'retain' flag set. Lastly, type of network defines how quickly messages are transferred between the broker and subscribers and clients. A fast connection over a fiber optic would update and deliver the messages fast. If the network is slow, then it may be the case that the values received by subscribers may not be most recent. Also, fewer messages will be dropped with a more reliable mode of transfer.
- b. Different QoS can have different effects on the performance of the broader MQTT network. The 0 QoS which is 'fire and forget' will increase performance in a certain way. With this QoS, the broker does not have to ensure that the client receives the message or not and thus other clients' message will be sent more frequently. Fewer memory will be consumed as well as the broker does not have to keep an account of who has received what message as it does not send puback messages. Over a less reliable network though, clients may have a high chance that they are not receiving message and they have no way of telling the broker. QoS 1 which is 'At least once' would use more of the computing resources but it means that clients will receive message at least once. More memory would be required thus as the broker needs to keep track if the message has been received by the client. QoS 2 which is 'Exactly once' is the safest but will reduce the performance of the network the most. It involves a four-way handshake, thus using even more computing resources. Messages sent and received by the broker need to be tracked more so needs more memory resources as well. However, like other QoS it does not improve or deteriorate the effect of network type on the performance.
- c. We can see from the diagram that the number of messages received decrease as QoS level increase. This is because, lost messages are sent again until the broker is sure that we have received it. Also, as QoS levels increase, more acknowledge messages are exchanged between the client and the broker.