MQTT for Asynchronous Messaging

1. Objective

The goal of this task was to verify that 10,000 messages were transmitted from the sender to the receiver over MQTT without any loss in transit.

---

2. Setup

- **MQTT Protocol**: Utilised for messaging between the sender and receiver. MQTT is a lightweight protocol often used for reliable, low-bandwidth communication.
- **Components**:
    - **Sender**: Publishes a sequence of 10,000 messages to a specific MQTT topic.
    - **Receiver**: Subscribes to the MQTT topic and logs each received message.

---

3. Code Overview

Sender Code (sender.py):

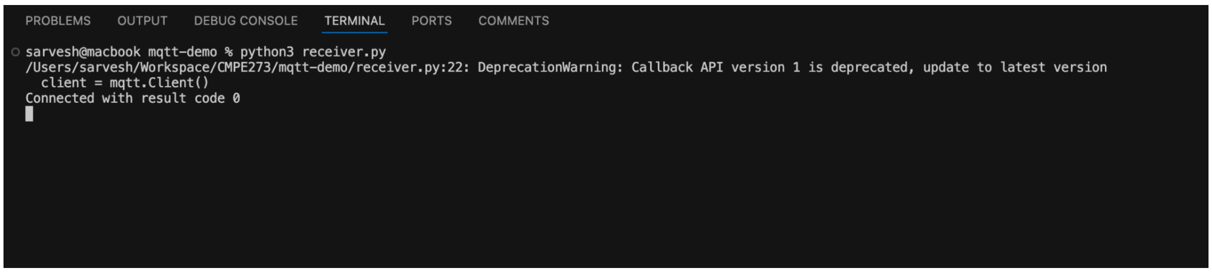
- Connects to the MQTT broker.
- Publishes messages in the format "Message X" where X is an incrementing counter from 1 to 10,000.
- Prints a confirmation message after each publication to ensure messages are sent correctly.

Receiver Code (receiver.py):

- Connects to the MQTT broker and subscribes to the specified topic.
- Logs each received message to `receiver.log`.
- Prints each received message to the terminal.

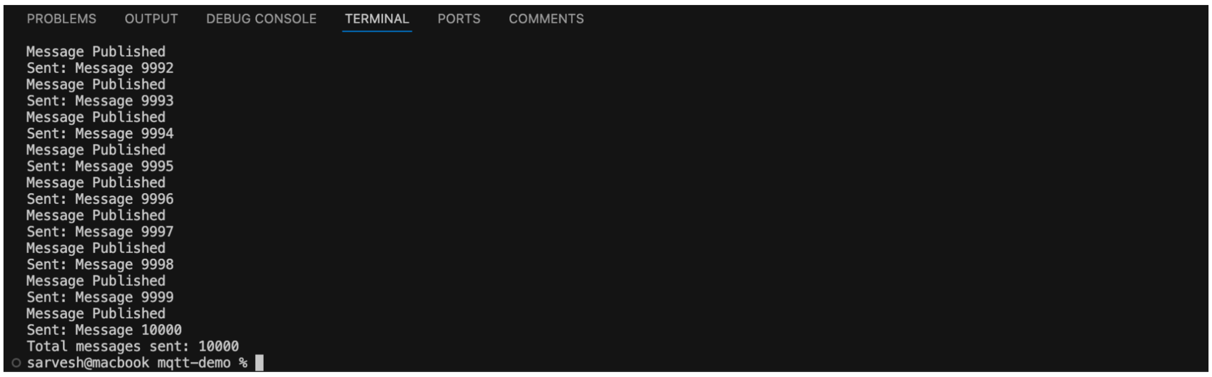
4. Results and Verification Process

Step 1: Execution and Count Verification



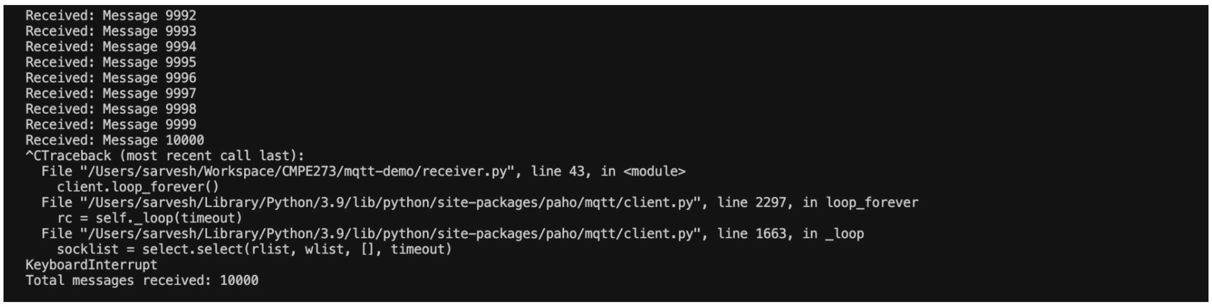
```
sarvesh@macbook mqtt-demo % python3 receiver.py
/Users/sarvesh/Workspace/CMPE273/mqtt-demo/receiver.py:22: DeprecationWarning: Callback API version 1 is deprecated, update to latest version
  client = mqtt.Client()
Connected with result code 0
```

This screenshot shows the MQTT client connection with "Connected with result code 0," indicating a successful connection to the broker.



```
Message Published
Sent: Message 9992
Message Published
Sent: Message 9993
Message Published
Sent: Message 9994
Message Published
Sent: Message 9995
Message Published
Sent: Message 9996
Message Published
Sent: Message 9997
Message Published
Sent: Message 9998
Message Published
Sent: Message 9999
Message Published
Sent: Message 10000
Total messages sent: 10000
sarvesh@macbook mqtt-demo %
```

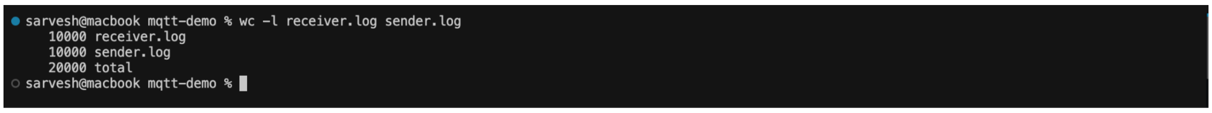
This screenshot displays the sender output, publishing messages from Message 9992 up to 10,000, confirming that all messages were sent.



```
Received: Message 9992
Received: Message 9993
Received: Message 9994
Received: Message 9995
Received: Message 9996
Received: Message 9997
Received: Message 9998
Received: Message 9999
Received: Message 10000
^CTraceback (most recent call last):
  File "/Users/sarvesh/Workspace/CMPE273/mqtt-demo/receiver.py", line 43, in <module>
    client.loop_forever()
  File "/Users/sarvesh/Library/Python/3.9/lib/python/site-packages/paho/mqtt/client.py", line 2297, in loop_forever
    rc = self._loop(timeout)
  File "/Users/sarvesh/Library/Python/3.9/lib/python/site-packages/paho/mqtt/client.py", line 1663, in _loop
    socklist = select.select(rlist, wlist, [], timeout)
KeyboardInterrupt
Total messages received: 10000
```

This screenshot shows the receiver output, successfully receiving messages up to Message 10,000, verifying that all messages sent by the sender were received.

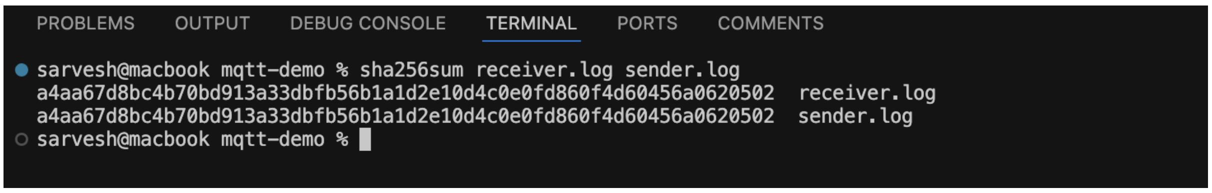
Step 2: Log File Line Count Verification



```
sarvesh@macbook mqtt-demo % wc -l receiver.log sender.log
   10000 receiver.log
   10000 sender.log
   20000 total
sarvesh@macbook mqtt-demo %
```

This screenshot shows the output of the `wc -l` command for `receiver.log` and `sender.log`, confirming that each file contains exactly 10,000 lines, which corresponds to the 10,000 messages sent and received.

Step 3: SHA-256 Checksum Verification



```
● sarvesh@macbook mqtt-demo % sha256sum receiver.log sender.log
  a4aa67d8bc4b70bd913a33dbfb56b1a1d2e10d4c0e0fd860f4d60456a0620502  receiver.log
  a4aa67d8bc4b70bd913a33dbfb56b1a1d2e10d4c0e0fd860f4d60456a0620502  sender.log
○ sarvesh@macbook mqtt-demo % █
```

This screenshot shows the SHA-256 checksums for both `receiver.log` and `sender.log`, which match exactly. This confirms that no messages were lost or altered during transmission.

---

5. Conclusion

The test was successful, with all 10,000 messages being transferred from the sender to the receiver without any loss. The verification process through message counts and checksum validation confirms that the MQTT protocol ensured reliable message delivery in this setup.