WebApp3

3I24 中川 寛之

目的

- WebSocketについて理解する
- MQTTについて理解する
- ネットワークプロトコルの違いにより、アーキテクチャ、実装方法、通信量などが異なることを理解する
- Webアプリケーション開発における、ネットワークプロトコルの選定について学ぶ

使用したツール

- docker (コンテナ)
- Node.js
- React (Frontend Web Framework: javascript/typescript)
- WebSocket
- MQTT(mosquitto)

環境構築1:

作成したシステムの概略

> Formに入力したメッセージを表示するWebアプリの構築
> *メッセージは永続化されない

動作確認の方法と結果

<chromeの画像>

<edgeの画像>

Formに入力してメッセージが表示されることが確認された。
また、メッセージは双方のWebに表示されなかった。

課題1:

動作確認の方法と結果

<HTTPストリームの内容(抜粋)>

```
OPTIONS /submit HTTP/1.1
Host: localhost:3001
Connection: keep-alive
Accept: */*
Access-Control-Request-Method: POST
Access-Control-Request-Headers: access-control-allow-credentials,access-control-
allow-origin,content-type
Origin: http://localhost:3000

HTTP/1.1 200 OK
Access-Control-Allow-Origin: http://localhost:3000
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: access-control-allow-credentials, access-control-
allow-origin, content-type
Access-Control-Allow-Methods: DELETE, GET, HEAD, OPTIONS, PATCH, POST, PUT
```

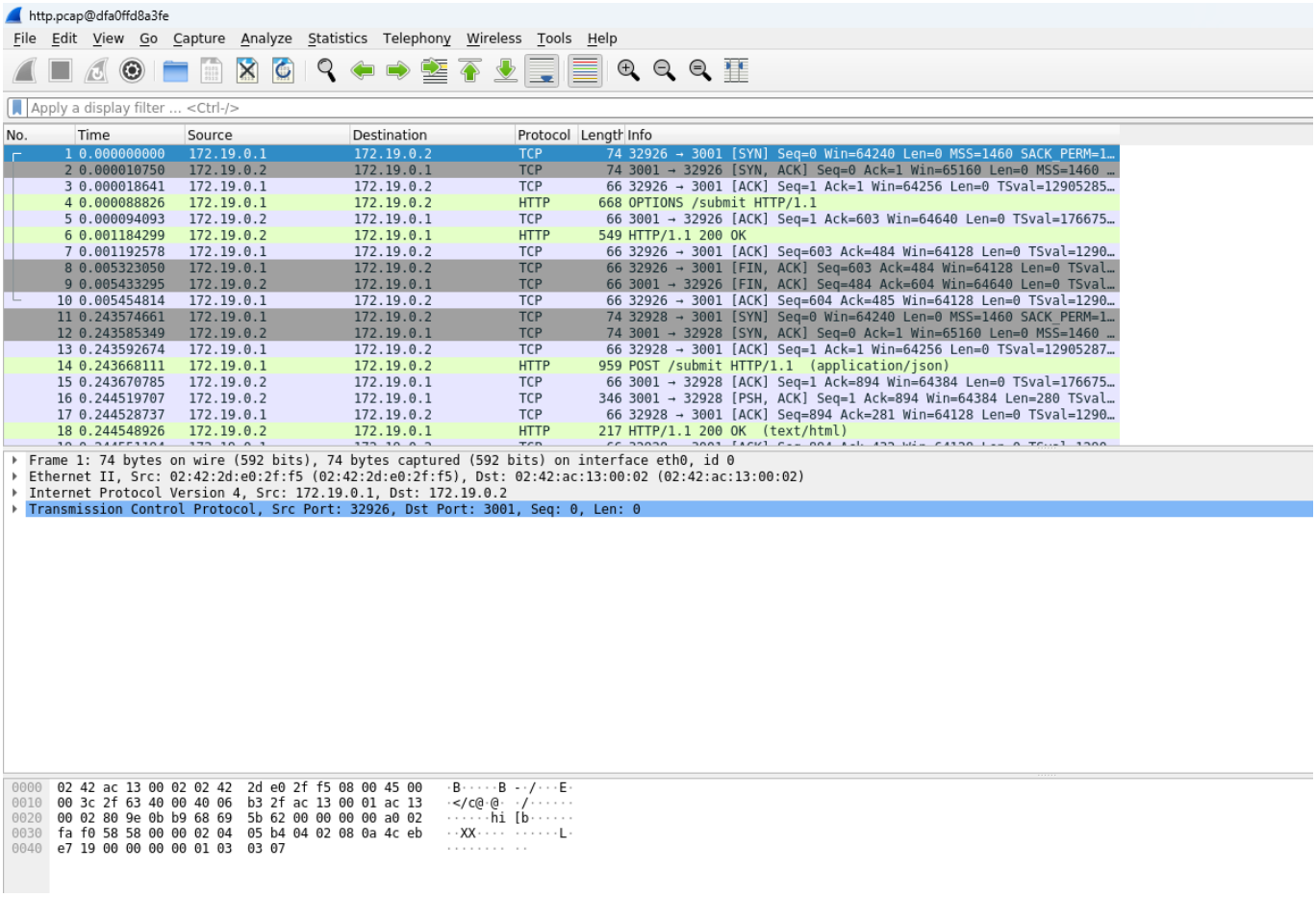
- **Access-Control-Request-Method**:送ろうとしている使用したいメソッド(POST)
  <=> **Access-Control-Allow-Methods**:許可しているメソッド

- **Access-Control-Request-Headers**:送ろうとしているヘッダー
  <=>**Access-Control-Allow-Headers**:サーバが許可してるヘッダー

- **Origin**:送ろうとしているヘッダー
  <=>**Access-Control-Allow-Origin**:許可されたオリジン(localhost:3000)

参考

<tsharkの様子>



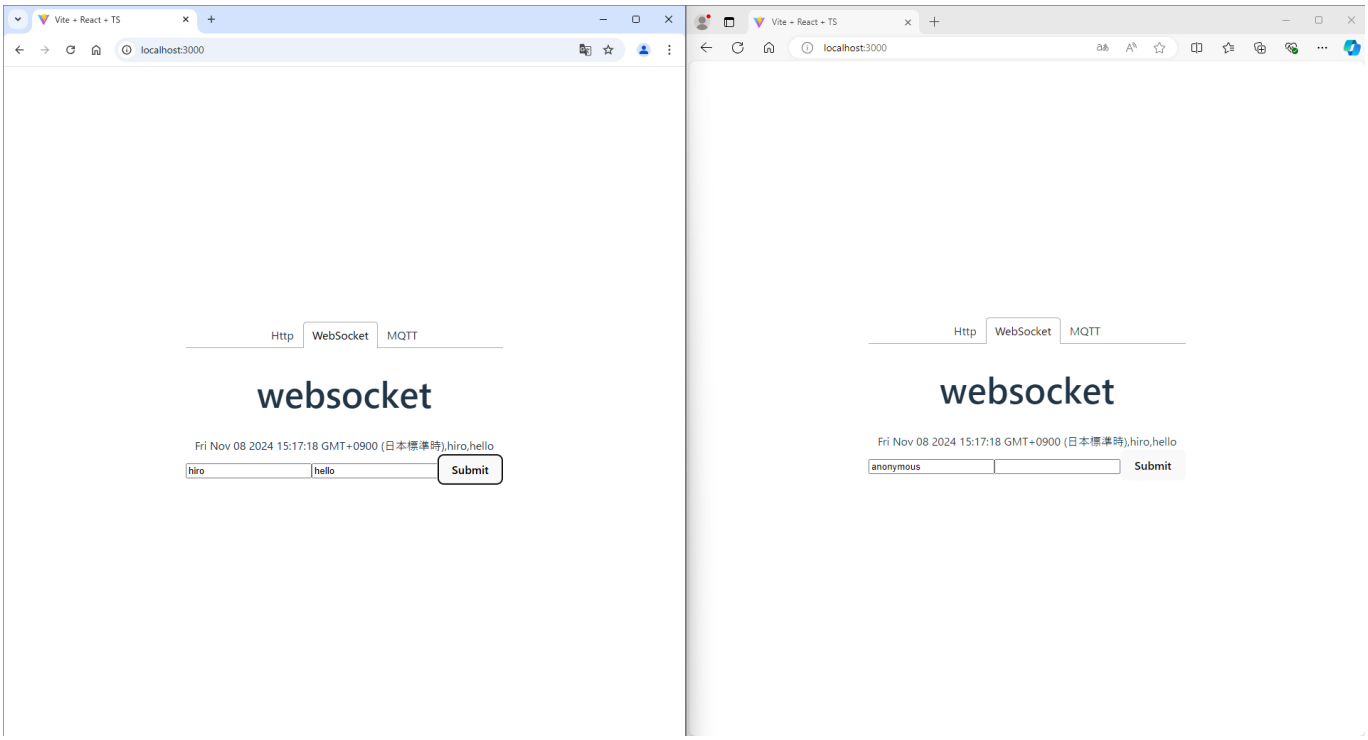
> パケット総数は1773[byte]

環境構築2:

作成したシステムの概略

> REST APIに変わり、WebSocketを用いてサーバ、クライアント間のデータ送受信を行う実装

動作確認の方法と結果



上記の画像を確認すると、chrome側で送信されたメッセージがedge側で確認できた

課題2:

動作確認の方法と結果

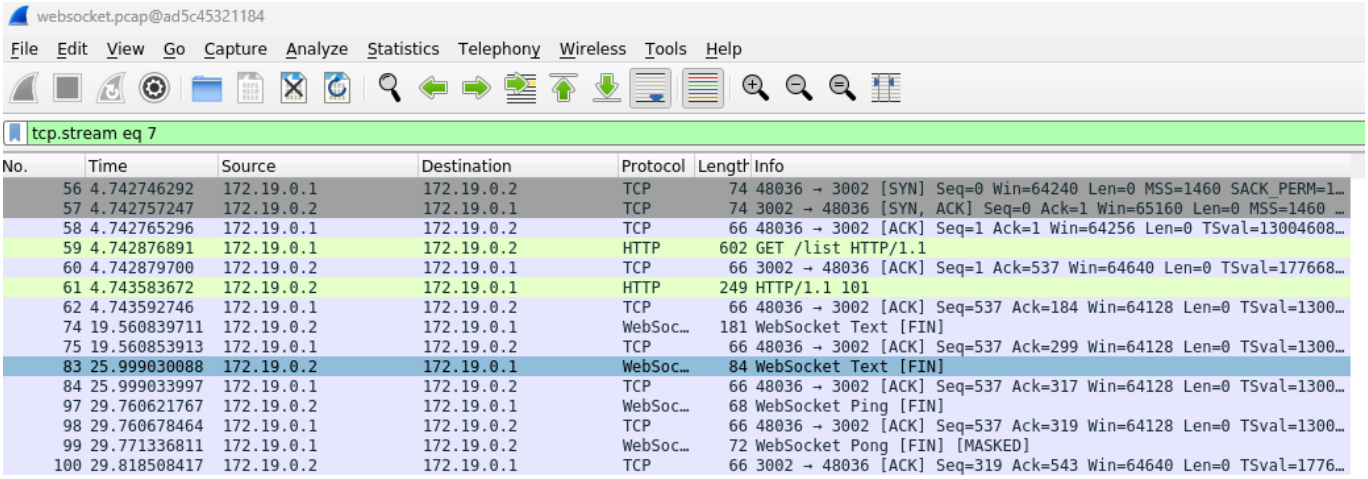
<WebSocketのストリーム内容(抜粋)>

```
GET /list HTTP/1.1
Host: localhost:3002
Connection: Upgrade
Upgrade: websocket
Origin: http://localhost:3000
Sec-WebSocket-Version: 13
Sec-WebSocket-Key: +IofNf891WAdvvEdKvBaPw==

HTTP/1.1 101
Upgrade: WebSocket
Connection: Upgrade
Sec-WebSocket-Accept: 6NlBz+lpAyFsPqh+E+kJyM61qrc=
```

- **Upgrade: websocket**：HTTPからWebSocketへのプロトコル切り替えを要求
- **Sec-WebSocket-Key**:特定のクライアントとのコネクションの確立を立証する為に使われる

<tsharkの様子>



No.	Time	Source	Destination	Protocol	Length	Info
56	4.742746292	172.19.0.1	172.19.0.2	TCP	74	48036 → 3002 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1…
57	4.742757247	172.19.0.2	172.19.0.1	TCP	74	3002 → 48036 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 …
58	4.742765296	172.19.0.1	172.19.0.2	TCP	66	48036 → 3002 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=13004608…
59	4.742876891	172.19.0.1	172.19.0.2	HTTP	602	GET /list HTTP/1.1
60	4.742879700	172.19.0.2	172.19.0.1	TCP	66	3002 → 48036 [ACK] Seq=1 Ack=537 Win=64640 Len=0 TSval=177668…
61	4.743583672	172.19.0.2	172.19.0.1	HTTP	249	HTTP/1.1 101
62	4.743592746	172.19.0.1	172.19.0.2	TCP	66	48036 → 3002 [ACK] Seq=537 Ack=184 Win=64128 Len=0 TSval=1300…
74	19.560839711	172.19.0.2	172.19.0.1	WebSoc…	181	WebSocket Text [FIN]
75	19.560853913	172.19.0.1	172.19.0.2	TCP	66	48036 → 3002 [ACK] Seq=537 Ack=299 Win=64128 Len=0 TSval=1300…
83	25.999030088	172.19.0.2	172.19.0.1	WebSoc…	84	WebSocket Text [FIN]
84	25.999033997	172.19.0.1	172.19.0.2	TCP	66	48036 → 3002 [ACK] Seq=537 Ack=317 Win=64128 Len=0 TSval=1300…
97	29.760621767	172.19.0.2	172.19.0.1	WebSoc…	68	WebSocket Ping [FIN]
98	29.760678464	172.19.0.1	172.19.0.2	TCP	66	48036 → 3002 [ACK] Seq=537 Ack=319 Win=64128 Len=0 TSval=1300…
99	29.771336811	172.19.0.1	172.19.0.2	WebSoc…	72	WebSocket Pong [FIN] [MASKED]
100	29.818508417	172.19.0.2	172.19.0.1	TCP	66	3002 → 48036 [ACK] Seq=319 Ack=543 Win=64640 Len=0 TSval=1776…

パケット総数は1866[byte]

環境構築3:

作成したシステムの概略

> mosquittoを使用し、WebSocketより、データ通信を軽くするように変更

動作確認の方法と結果

< subscribe >

```
docker compose run --rm mqtt-sub mosquitto_sub  -u appuser -P UYII8ceNiICh -h
mqtt-broker -t "#" -v
test/d1/ 1
```

< publish >

```
codespace ➜ /workspaces/webapp3-HIROSN0W/src (main) $ docker compose run --rm
mqtt-pub mosquitto_pub -u appuser -P UYII8ceNiICh -h mqtt-broker  -t test/d1/ -m
"1"
```

上記のように表示されたため、mqtt-pubからbrokerへトピック付きのメッセージが送られ、subscribe側でそのメッセージを受信することができたことが確認された。

課題3:

動作確認の方法と結果

<全デバイスの温度(temp)を取得する>

```
codespace ➜ /workspaces/webapp3-HIROSN0W/src (main) $ docker compose run --rm
mqtt-sub mosquitto_sub  -u appuser -P UYII8ceNiICh -h mqtt-broker -t
"+/+/+/+/temp" -v
snct/building1/3F/room1305/temp 28
snct/building1/3F/room1301/temp 25
snct/building1/2F/room1204/temp 24
```

<全ての3Fの部屋の湿度(humi)を取得する>

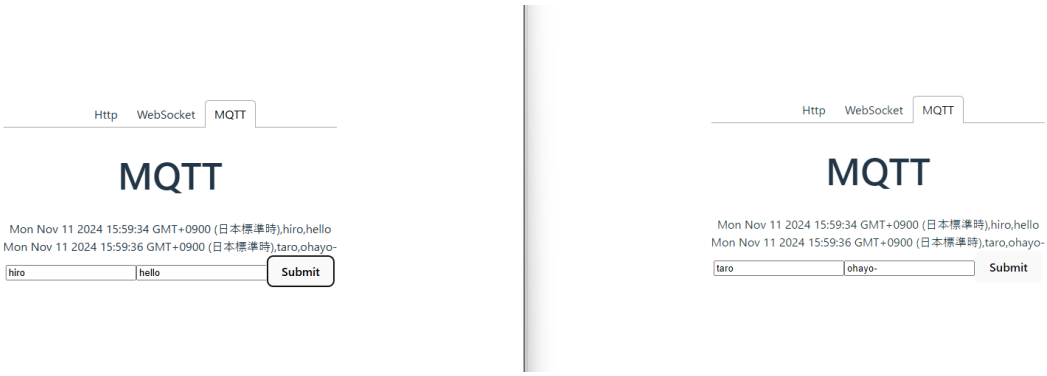
```
codespace ➜ /workspaces/webapp3-HIROSN0W/src (main) $ docker compose run --rm
mqtt-sub mosquitto_sub  -u appuser -P UYII8ceNiICh -h mqtt-broker -t
"+/+/+/+/humi" -v
snct/building1/3F/room1305/humi 56
snct/building1/3F/room1301/humi 65
snct/building1/2F/room1204/humi 70
```

環境構築4:

作成したシステムの概略

> MQTT over WebSocketを使用し、WebブラウザをMQTTクライアントとしての利用

動作確認の方法と結果



上記の画像を確認するとMQTTを使用して、双方で送ったメッセージが双方のWebページで確認された。

課題4:

動作確認の方法と結果

<mpttのストリーム内容(抜粋)>

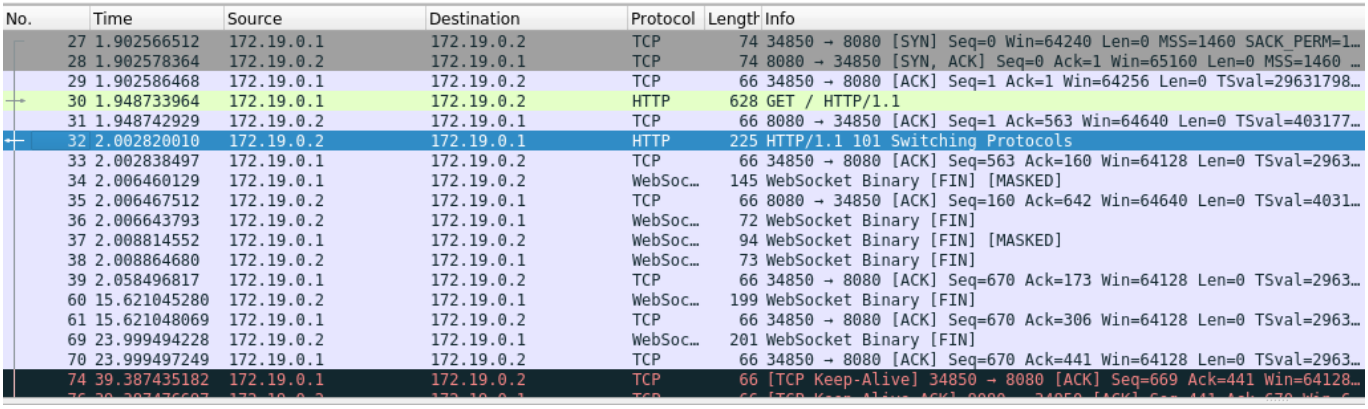
```
GET / HTTP/1.1
Host: localhost:8080
Connection: Upgrade
Pragma: no-cache
Cache-Control: no-cache
```

```
Upgrade: websocket
Origin: http://localhost:3000
Sec-WebSocket-Key: xw+cLUHKv1BRP0f01FbhDg==
Sec-WebSocket-Extensions: permessage-deflate; client_max_window_bits
Sec-WebSocket-Protocol: mqtt

HTTP/1.1 101 Switching Protocols
Upgrade: WebSocket
Connection: Upgrade
Sec-WebSocket-Accept: hLXiH03q3q4XAJCgNHG5vz0QYp4=
Sec-WebSocket-Protocol: mqtt
```

- **Upgrade: websocket** + **Connection: Upgrade**
  =>HTTPからWebSocketへのプロトコル切り替えを要求
- **Sec-WebSocket-Protocol: mqtt**
  =>クライアントがプロトコルにMQTTに指定
- **Sec-WebSocket-Protocol: mqtt**
  =>サーバーがWebSocket接続でMQTTプロトコルの使用の受け入れ

<tsharkの様子>



No.	Time	Source	Destination	Protocol	Length	Info
27	1.902566512	172.19.0.1	172.19.0.2	TCP	74	34850 → 8080 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1…
28	1.902578364	172.19.0.2	172.19.0.1	TCP	74	8080 → 34850 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 …
29	1.902586468	172.19.0.1	172.19.0.2	TCP	66	34850 → 8080 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=29631798…
30	1.948733964	172.19.0.1	172.19.0.2	HTTP	628	GET / HTTP/1.1
31	1.948742929	172.19.0.2	172.19.0.1	TCP	66	8080 → 34850 [ACK] Seq=1 Ack=563 Win=64640 Len=0 TSval=403177…
32	2.002820010	172.19.0.2	172.19.0.1	HTTP	225	HTTP/1.1 101 Switching Protocols
33	2.002838497	172.19.0.1	172.19.0.2	TCP	66	34850 → 8080 [ACK] Seq=563 Ack=160 Win=64128 Len=0 TSval=2963…
34	2.006460129	172.19.0.1	172.19.0.2	WebSoc…	145	WebSocket Binary [FIN] [MASKED]
35	2.006467512	172.19.0.2	172.19.0.1	TCP	66	8080 → 34850 [ACK] Seq=160 Ack=642 Win=64640 Len=0 TSval=4031…
36	2.006643793	172.19.0.2	172.19.0.1	WebSoc…	72	WebSocket Binary [FIN]
37	2.008814552	172.19.0.1	172.19.0.2	WebSoc…	94	WebSocket Binary [FIN] [MASKED]
38	2.008864680	172.19.0.2	172.19.0.1	WebSoc…	73	WebSocket Binary [FIN]
39	2.058496817	172.19.0.1	172.19.0.2	TCP	66	34850 → 8080 [ACK] Seq=670 Ack=173 Win=64128 Len=0 TSval=2963…
60	15.621045280	172.19.0.2	172.19.0.1	WebSoc…	199	WebSocket Binary [FIN]
61	15.621048069	172.19.0.1	172.19.0.2	TCP	66	34850 → 8080 [ACK] Seq=670 Ack=306 Win=64128 Len=0 TSval=2963…
69	23.999494228	172.19.0.2	172.19.0.1	WebSoc…	201	WebSocket Binary [FIN]
70	23.999497249	172.19.0.1	172.19.0.2	TCP	66	34850 → 8080 [ACK] Seq=670 Ack=441 Win=64128 Len=0 TSval=2963…
74	39.387435182	172.19.0.1	172.19.0.2	TCP	66	[TCP Keep-Alive] 34850 → 8080 [ACK] Seq=669 Ack=441 Win=64128…

```
パケット総数2379[byte]
```

理解したこと、理解できていないこと

WebSocketは、双方向通信を提供し、クライアントとサーバーがリアルタイムでデータを送受信するプロトコルで、MQTTは軽量で通信量が少ないプロトコルである、違いを理解することができた。 また、リアルタイム性や、低通信量など必要にあわせてプロトコルを選定することの重要性を感じた。 実際のXのような毎秒、大量のデータを受け取るようなSNSではどのようにしているか興味を持った。