UNIVERSIDAD POLITÉCNICA DE MADRID

Departamento de Ingeniería de Sistemas Telemáticos
Universidad Politécnica de Madrid

BROKER MQTT Y GRAFANA

Introducción

En esta práctica[1] vamos usar un broker MQTT donde publicar las constantes vitales de nuestro cliente. También utilizaremos un datasource que se suscribirá al topic donde publicamos mensajes, para poder dibujar los datos en un dashboard de Grafana.

La siguiente figura resalta las componentes que trabajaremos:

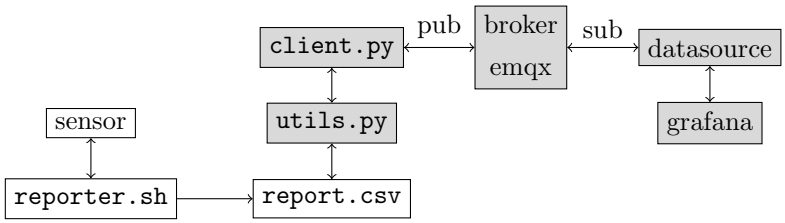

Figura 1: escenario de la práctica.

Instalación

Instalación del broker EMQX

Primero vamos a instalar el contenedor del *broker* EMQX ejecutando lo siguiente en la máquina virtual:

```
$ docker run -d --name emqx -p 1883:1883 -p 8083:8083\
  -p 8883:8883 -p 8084:8084 -p 18083:18083\
  emqx/emqx:5.0.2
```

[1] Todas las preguntas tienen el mismo valor/puntuación, a excepción del Problema 8, que tiene doble valor/puntuación.

El comando se encargará de descargar la versión 5.0.2 de EMQX y crear un contenedor llamado "emqx". Además, los puertos locales indicados con las banderas `-p` serán redireccionados al contenedor. Esto último nos permitirá acceder al dashboard de EMQX. Además, con el comando de arriba hacemos que el broker EMQX *haga disponible la suscripción a topics* en el puerto 1883.

Para acceder al dashboard del broker EMQX acceda a la siguiente URL en el navegador:

`http://localhost:18083`

y haga login usando el usuario "admin" y la contraseña "public". El dashboard mostrará paneles dedicados a las conexiones MQTT abiertas con el broker EMQX, así como los topics, suscripciones, etc.

Instalación de Grafana

Con Grafana vamos a poder visualizar las métricas reportadas en cada uno de los topics MQTT. En particular, vamos a instalar grafana y un datasource de MQTT. Un data source de grafana es un plugin que permite recoger información desde una fuente de datos. En nuestro caso, la fuente de datos serán los distintos topics MQTT.

Para instalar Grafana con el plugin del datasource MQTT ejecute el siguiente comando en la terminal:

```
$ sudo docker run -d -p 3000:3000 --name=grafana \
  -e "GF_INSTALL_PLUGINS=grafana-mqtt-datasource" \
  grafana/grafana-enterprise
```

El comando descargará la imagen correspondiente de Grafana y ejecutará el contenedor.

Para acceder al dashboard de Grafana acceda a la siguiente URL en el navegador:

`localhost:3000`

y haga login usando el usuario "admin" y contraseña "admin".

Publicación al broker EMQX

A continuación vamos a ejecutar nuestro cliente MQTT para publicar mensajes al topic[2] `Madrid/Moncloa/__Nombre__`. Para ello sírvase del programa `client.py` de la anterior sesión de laboratorio.

[2]Sustituya `__Nombre__` por su nombre.

Ejecute de nuevo el `reporter.sh` y modifique el código del `client.py` para publicar las constantes vitales al broker EMQX instalado en esta práctica. A continuación haga los siguientes pasos:

1. Inicie una captura wireshark en la interfaz `Loopback: lo`;
2. ejecute el `cliente.py` para observar la conexión y envío de, al menos, un mensaje; y
3. detenga la captura wireshark y guárdela en el archivo `publish-broker-grupoX.pcapng`.

Problema 1. Especifique el puerto destino TCP del paquete `CONNACK` (el acknowledgement de la conexión). Escriba la respuesta en el campo `ack_tcp_dst_port` del archivo `respuestas-X.json`.

El/los mensajes MQTT enviados por el cliente podrá visualizarlos en el dashboard del broker EMQX. Para ello basta con que acceda a la URL:

`http://localhost:18083`

y fíjese en el panel "Incoming messages".

Fuente de datos para Grafana

El siguiente paso es conectar Grafana con el broker EMQX para poder visualizar las constantes vitales. Para ello es necesario crear una fuente de datos de Grafana que se suscriba al topic `Madrid/Moncloa/__Nombre__` en que publica nuestro cliente.

Primero de todo, hay que buscar cuál es la dirección IP del contenedor en el que se ejecuta el broker EMQX. Para ello ejecute el siguiente comando en su terminal:

```
$ docker inspect ‘docker ps | grep 5.0.2\
  | cut -d’ ’ -f1‘ | grep "IPAddress"
```

Problema 2. Anote la dirección IP del contenedor del broker EMQX en el campo `broker_ip` de `respuestas-X.json`.

Una vez hemos identificado la IP del broker EMQX, vamos a proceder a crear la fuente de datos para Grafana. Para ello siga los siguientes pasos:

1. Acceda al panel de Grafana en la URL `localhost:3000`;
2. en el menú de la izquierda pinche en `Connections/Data Sources`;

3. pinche el botón `Add Data Source`;
4. pinche en `MQTT`;
5. rellene el campo `URI` con la IP del broker EMQX y el puerto en el que hace disponible la suscripción `IP:port` ;
6. rellene los campos `username` y `password` usando los que utiliza en su script `client.py`;
7. inicie una captura wireshark en la interfaz `docker0`;
8. vuelva al dashboard de Grafana y pinche en el botón `Save & test`; y
9. detenga la captura de wireshark y guárdela – aplicando el filtro "mqtt' – en el fichero `connect-data-source-grupoX.pcapng`.

Tras seguir los pasos se habrá iniciado una conexión MQTT entre la fuente de datos de Grafana y el broker EMQX.

Sirviéndose de la captura de wireshark, responda a las siguientes preguntas:

Problema 3. ¿Cuál es la QoS de la conexión entre el broker y la fuente de datos de Grafana? Anote su respuesta en el campo `qos_grafana` de `respuestas-X.json`.

Problema 4. ¿Cuál es el ID que utiliza el plugin de la fuente de datos de Grafana en la conexión MQTT? Anote su respuesta en el campo `id_grafana` de `respuestas-X.json`.

Creación de Dashboard de Grafana

Ahora vamos a crear un dashboard de Grafana para visualizar las constantes vitales que reporta nuestro cliente MQTT. Para ello haremos que nuestro plugin del datasource MQTT se suscriba al topic `Madrid/Moncloa/__Nombre__`.

Siga los siguientes pasos para crear el dashboard:

1. Pinche en `Dashboards` en el panel izquierdo de Grafana;
2. pinche en el botón `Create Dashboard` y luego en `Add Visualization`;
3. pinche en `grafana-mqtt-datasource`;
4. inicie una captura wireshark en la interfaz `docker0`;
5. rellene el campo `topic` usando el topic en el que su cliente publica las constantes vitales;

6. pinche con el ratón fuera del campo `topic` para que comience la visualización; y

7. detenga la captura de wireshark y guárdela – aplicando el filtro "mqtt' – en el fichero `data-source-subscribe-grupoX.pcapng`.

Sirviéndose de la visualización del dashboard y la captura de wireshark, responda a las siguientes preguntas:

Problema 5. ¿Qué columna del dataset se está dibujando? Anote su respuesta en el campo `columna_dibujada` de `respuestas-X.json`.

Problema 6. Identifique los mensajes de suscripción al topic que realiza el plugin del datasource MQTT. ¿Cuáles son los tipos de mensaje para el REQUEST y REPLY? Anote su respuesta en el campo `request_ack_types` de `respuestas-X.json`.

Reporte de constantes en JSON

A pesar de que su `client.py` está enviando distintas constantes vitales – cada una de ellas separada por comas –, Grafana solo muestra un único valor en el dashboard. Para mostrar las distintas constantes vitales en el dashboard de Grafana vamos a enviar las constantes vitales en formato JSON.

Transformar línea en JSON

Usaremos nuestro archivo `utils.py` para crear una función `line_to_json(line)` que tranforme una `line` del dataset:

```
462,462,97,18,100,36,Abnormal
```

en la siguiente cadena JSON

```
'{"Idx":"462","Time":"462","HR":"97","RESP":"18",
"SpO2":"100","TEMP":"36","OUTPUT":"Abnormal"}'
```

Para ello sírvase de la librería `json` y de la función `json.dumps(obj)`, donde `obj` es un diccionario Python.

Publicar el JSON

Una vez programada la función `line_to_json(line)`, actualice el `client.py` para publicar un JSON en lugar de una línea. Asegúrese de importar la función `line_to_json` en las primeras líneas del código de su cliente.

A continuación realice los siguientes pasos:

1. inicie una captura wireshark en la interfaz `Loopback: lo`;
2. ejecute su `client.py`;
3. deje que se publiquen diez de mensajes; y
4. detenga la captura wireshark y guárdela en el archivo `publish-json-grupoX.pcapng`.

Sirviéndose de la captura de wireshark, responda a las siguiente pregunta:

Problema 7. ¿Qué campo del mensaje MQTT contiene el JSON? Anote su respuesta en el campo `campo_json` de `respuestas-X.json`.

Problema 8. ¿Cuál es la longitud del mensaje[3] MQTT en el PUBLISH $X \mod 10$ y $X + 5 \mod 10$ de su captura? Anote su respuesta en los campos `pub_x` y `pub_xm5` de `respuestas-X.json`.

Visualización de datos con Grafana

Para concluir vamos ilustrar las series temporales de las constantes vitales. Utilizaremos el datasource que hemos configurado para obtener los datos del JSON que reporta nuestro `client.py`.

Para crear el dashboard, siga los pasos del apartado "Creación de Dashboard de Grafana", sin iniciar una captura wireshark.

Acto seguido, procederemos para acceder y transformar las distintas constantes vitales reportadas en el JSON. Para ello siga los siguientes pasos:

1. Acceda a la pestaña `Transform Data`;
2. pinche en el botón `Add Transformation`;
3. pinche en el cuadrado `Convert Field Type`;
4. pinche en el botón `+ Convert Field Type`;
5. en `Select Field` elija, por ejemplo, el campo `TEMP` y en el campo `Type` seleccione `Number`.

Observará la serie temporal de la temperatura. Puede guardar el panel pinchando en el botón `Save` que aparece arriba a la derecha.

Problema 9. Tras guardar el dashboard, acceda a `Dashboards` en el menu izquierdo, y seleccione visualizar la información de los últimos 5 min (Last 5 minutes). Haga una captura/foto a la pantalla mostrando la serie temporal y llame al fichero de la imagen `serie-temporal`.

[3] Numeramos los PUBLISH empezando en cero: $0, 1, 2, \ldots$

Entrega

Se subirá a moodle un archivo `brokerX.zip` (con `X` el número de grupo) que contenga:

1. el cliente `client.py`;
2. el archivo `utils.py`;
3. el JSON de respuestas `respuestas-X.json`;
4. las trazas de Wireskark:
   `publish-broker-grupoX.pcapng`
   `connect-data-source-grupoX.pcapng`
   `data-source-subscribe-grupoX.pcapng`
   `publish-json-grupoX.pcapng`; y
5. la captura de pantalla `serie-temporal`.

Atención I: las capturas deben contener *solamente* tráfico MQTT.
Atención II: una entrega sin los archivos especificados, o con archivos sin formato especificado tendrá un 0 en los **Problemas** correspondientes.