UNIVERSIDAD POLITÉCNICA DE MADRID

Departamento de Ingeniería de Sistemas Telemáticos
Universidad Politécnica de Madrid

ALERTAS MÉDICAS — mini-reto: Lectura de los datos biométricos de los pacientes y desarrollo de un sistema de generación de alertas médicas

Pregunta Esencial

¿Cómo leer, analizar, interpretar y reaccionar en tiempo real a los datos biométricos de los pacientes para detectar alteraciones en sus constantes vitales?

Descripción

Los pacientes del hospital RAID-BIO están siendo monitorizados de forma remota a través de sensores que registran de manera continua sus constantes vitales (retos anteriores). Sin embargo, para garantizar la seguridad de los pacientes, el personal sanitario necesita recibir alertas automáticas cada vez que se detecten valores anómalos que puedan indicar una posible emergencia.

En este **mini-reto** debe desarrollar un sistema de recepción y análisis de las constantes vitales utilizando el protocolo de comunicaciones MQTT. Desde el centro de datos del hospital, el dispositivo suscriptor actuará como punto de supervisión y será responsable de recibir los datos biométricos transmitidos por los pacientes, evaluarlos y generar alertas automáticas si se superan ciertos umbrales.

Para ello, el especialista debe de disponer de una herramienta donde visualiza las alertas generadas por los datos anómalos. Para ello, la herramienta está suscrita, a través de MQTT (script escrito en Python - `subs.py`) a los datos de los pacientes (escritos en el tópico correspondiente mediante el script `client.py`). La herramienta del especialista, de forma automática, identificará situaciones críticas y publicará una alerta si es necesario.

En el apartado **Recursos** del mini-reto se proporcionan más detalles sobre esta parte del escenario de laboratorio.

Objetivos

A continuación se enumeran los objetivos de este mini-reto:

- Aprender cómo se programa un **suscriptor** de MQTT que recibe y procesa datos biométricos en tiempo real.
- Aprender cómo se generan **alertas** automáticas cuando se detectan situaciones de riesgo.
- Implementar mecanismos de suscripción a diferentes **topics**, gestionando flujos de datos provenientes de distintos pacientes y asegurando su correcta identificación y procesamiento.

Actividades Guiadas

A continuación se presentan las actividades que debe realizar para conseguir resolver el mini-reto:

Actividad 1 Estudie los **Recursos** del mini-reto para entender cuáles son los componentes del escenario de laboratorio. Revise las referencias proporcionadas para el estudio del protocolo MQTT, especialmente la lógica de suscripción y recepción de mensajes.

Actividad 2 Antes de comenzar a programar el script `subs.py`, inicie el broker EMQX. Para ello, sírvase de las instrucciones de la práctica anterior. Verifique que el contenedor del bróker está activo y que el panel de administración web es accesible desde su navegador:

`localhost:18083`

usando como nombre de usuario `admin` y como contraseña `public`.

Actividad 3 Verifique que el archivo (`report.csv`) ha sido generado previamente mediante la ejecución del script (`reporter.sh`) de las anteriores sesiones.

Actividad 4 Implemente en `utils.py` la función `line_to_dict(fpath, line)` para disponer de un diccionario que almacena el valor asociado a cada medida de métrica biométrica (oxígeno, frecuencia cardiaca). Al recibir la *primera* línea del CSV de las constantes vitales el resultado de la función debería ser:

```
{
  "idx": 0,
  "time": 0,
  "hr": 94,
  "resp": 21,
  "SpO2": 97,
  "temp": 36.2,
  "output": "Normal"
}
```

Para probar que la función está bien programada ejecútela en el entorno python3:

```
$ python3
>>> import utils
>>> utils.line_to_dict('report.csv',X+2)
```

donde X es el número del grupo al que pertenece. Responda a la **Pregunta 1**.

Actividad 5 Modifique el script (`client.py`) para que, usando el diccionario obtenido en `line_to_dict(fpath, line)`, publique las constantes vitales `hr,resp,SpO2` y `temp` en topics independientes (formato: `vitals/metrica`) usando QoS 1. Responda a la **Pregunta 2**.

Actividad 6 Programe el script (`subs.py`) para que se suscriba exclusivamente al topic `vitals/temp` a través del broker EMQX. Responda a las siguientes preguntas: **Pregunta 3**, **Pregunta 4**, **Pregunta 5**, **Pregunta 6**.

Actividad 7 Programe el script (`subs.py`) para que utilice un wildcard en el topic de suscripción, permitiendo así recibir todas las métricas publicadas por (`client.py`). Responda a la **Pregunta 7** y **Pregunta 8**.

Actividad 8 Modifique el script `subs.py` para que, al recibir mensajes del topic `vitals/hr`, analice si el valor recibido excede en más de un $\frac{X}{100}$ % el valor medio de

89 BPM. Si se detecta esta anomalía, el script deberá generar una alerta publicando un nuevo mensaje en el topic `alerts/hr`, con el número de BPM como contenido del mensaje. Responda a la **Pregunta 9** y **Pregunta 10**.

Preguntas Guiadas

A continuación se presentan las preguntas guiadas asociadas a cada una de las actividades guiadas de la práctica. Respóndalas para preparar el material de entrega que servirá para evaluar cómo se ha enfrentado a este mini-reto:

Pregunta 1 Guarde el resultado de la **Actividad 4** en el campo `lectura` del JSON de respuestas.

Pregunta 2 Tras hacer la **Actividad 5** inicie una captura en Wireshark en la interfaz `lo`. Ejecute el `client.py` para que reporte las constantes al broker EMQX[1]. Deje que se publiquen varias constantes vitales y guarde[2] en `publishes-grupoX.pcapng` la captura.

¿Cuál es el máximo valor de `temp` reportado en la captura? Responda en el campo `maxtemp` del JSON de respuestas.

Pregunta 3 Tras realizar la **Actividad 6** inicie una captura Wireshark en la interfaz `lo`, después ejecture `client.py` y `sub.py`. Guarde la captura en `subs-temp-grupoX.pcapng`. ¿Cuál es el tipo de mensaje del `Subscribe request`? Responda a la pregunta en el campo `reqtype` del JSON de respuestas.

Pregunta 4 ¿Con qué QoS se suscribe al topic `vitals/temp`? Responda a la pregunta en el campo `subsqos` del JSON de respuestas.

Pregunta 5 ¿Cuál es el puerto utilizado por el cliente para publicar mensajes? Responda a la pregunta en el campo `pubport` del JSON de respuestas.

Pregunta 6 ¿Cuál es el puerto utilizado por el suscriptor para recibir mensajes? Responda a la pregunta en el campo `subport` del JSON de respuestas.

Pregunta 7 Tras realizar la **Actividad 7** inicie una captura Wireshark y ejecute el `client.py` y `subs.py`. Deje que se publiquen unos veinte `PUBLISH` y guarde en `subs-wildcard-grupoX.pcapng` la captura de Wireshark. ¿A cuántas métricas se suscribe el `subs.py`? Responda en el campo `nummetricas` del JSON de respuestas.

[1] Recuerde identificar el puerto y dirección usados por EMQX para las comunicaciones MQTT.
[2] Recuerde que todas las capturas de la práctica deben contener *solo* los paquetes MQTT.

Pregunta 8 ¿Cuál es el topic del `PUBACK` con `Message identifier` $10 + X$? Responda en el campo `pubacktopic` del JSON de respuestas.

Pregunta 9 Tras realizar la **Actividad 8** inicie una captura de tráfico en Wireshark sobre la interfaz `lo`. Ejecute `client.py` y `subs.py` y espere a que se detecten y publiquen al menos un par de alertas. Guarde la captura resultante con el nombre `alertas-grupoX.pcapng`. ¿Cuántas alertas se envían? Responda en el campo `numalerts` del JSON de respuestas.

Pregunta 10 ¿Cuál es la QoS con la que se envían las alertas? Responda en el campo `qosalerts` del JSON de respuestas.

Recursos

Documentación teórica

1. Guía básica de Python 3 (disponible en: `http://python.org/doc`).
2. Manual de uso de Wireshark para análisis de protocolos. (disponible en: `https://www.wireshark.org/docs/wsug_html_chunked/`).
3. Documentación del protocolo MQTT y funcionamiento de los niveles de QoS (disponible en: `https://www.paessler.com/es/it-explained/mqtt`).

Documentación para la implementación

1. Descripción del escenario de alertas médicas (disponible en Moodle).
2. Broker MQTT (broker.emqx.io) (disponible en: `https://docs.datadoghq.com/es/integrations/emqx/`)
3. Published/Subscriber MQTT con EMQX: `https://www.emqx.com/en/blog/how-to-use-mqtt-in-python`

Entrega

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

1. el cliente `client.py`;
2. el cliente `subs.py`;
3. el archivo `utils.py`;

4. el JSON de respuestas `respuestas-X.json`; y

5. las trazas de Wireskark `publishes-grupoX.pcapng`, `subs-temp-grupoX.pcapng`, `subs-wildcard-grupoX.pcapng`, `alertas-grupoX.pcapng`

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 las **Preguntas** correspondientes.