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 al **Problema 1**..

Actividad 5. Modifique el script (`client.py`) para que, usando el diccionario obtenido en `line_to_dict(fpath, line)`, publique cada constante vital en un topic independiente (formato: `vitals/metrica`) usando QoS 1. Responda a la **Problema 2**..

Actividad 6. Haga pruebas para enviar las constantes del paciente al broker EMQX. Capture el tráfico con Wireshark sobre la interfaz lo de la máquina que emula el paciente y guarde el archivo como (sending-grupoX.pcapng), donde X corresponde al número de su grupo. JSON respuestas valor máximo de temp

Actividad 7. Modifique el script (`subs.py`) para que se suscriba exclusivamente al topic vitals/temp a través del broker EMQX. Con estas modificaciones, capture el tráfico con Wireshark sobre la interfaz lo y guarde el resultado (receiving-grupoX.pcapng). – subs-temp-grupoX.pcapng

Actividad 8. A continuación, verifique que las métricas de temperatura se reciben correctamente. Obtenga una nueva captura de tráfico y guarde el archivo con el nombre subs-temp-grupoX.pcapng. – REDUNDANTE, ES LO MISMO QUE ANTES

Actividad 9. HAY QUE METER LAS PREGUNTAS DEL P3-P6

Actividad 10. Modifique el script (`subs.py`) para que utilice un wildcard en el topic de suscripción, permitiendo así recibir todas las métricas publicadas por (`client.py`).

Actividad 11. Inicie una captura de tráfico en Wireshark sobre la interfaz lo, y ejecute (`client.py`) y (`subs.py`). Deje que se publiquen aproximadamente veinte mensajes PUBLISH y guarde la captura como (subs-wildcard-grupoX.pcapng).

Actividad 12. Modifique el script `subs.py` para que, al recibir mensajes del topic vitals/hr, analice si el valor recibido excede en más de un 15 % el valor medio esperado 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. 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).

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:

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

Problema 2. Tras hacer la **Actividad 5**. ejecute el `client.py` para que reporte las constantes al broker EMQX[1], inicie una captura en Wireshark en la interfaz `lo`. 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.

[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.

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. Broker MQTT (broker.emqx.io) (disponible en: `https://docs.datadoghq.com/es/integrations/emqx/`)
2. Published/Subscriber MQTT con EMQX: `https://www.emqx.com/en/blog/how-to-use-mqtt-in-python`

Introducción

En esta práctica[3][4] vamos a programar un suscriptor MQTT `subs.py` que recolecte las métricas publicadas por el cliente `client.py`. Para ello abriremos una suscripción con el broker y nos suscribiremos a los topics de reporte de métricas. Finalmente, haremos un detector de anomalías en el pulso para publicar alertas desde el `subs.py`. La figura mostrada a continuación resalta con fondo oscuro las componentes que modificaremos:

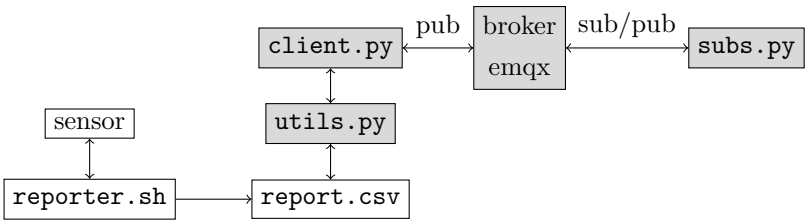

Figura 1: escenario de la práctica.

[3]Todas las preguntas tienen el mismo valor/puntuación.

[4]Este material está protegido por la licencia CC BY-NC-SA 4.0.

Arrancar broker EMQX

Antes de comenzar a programar el `subscriber.py` tenemos que arrancar el broker EMQX. Para ello siga los pasos de la práctica anterior. En caso de que el contenedor del broker EMQX ya esté disponible, ejecute el siguiente comando para arrancarlo de nuevo:

```
$ docker start ‘docker ps | grep "emqx:5.0.2"\
  | cut -d’ ’ -f1‘
```

Puede comprobar que el broker se está ejecutando correctamente accediendo al panel de EMQX desde el navegador:

`localhost:18083`

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

Publicar métricas en topics

Tras arrancar el broker EMQX vamos a publicar cada métrica en un topic distinto. En concreto, publicaremos cada columna del `report.csv` en el topic `vitals/metrica`. Por ejemplo, el ritmo cardíaco `hr` se publica en el topic `vitals/hr`.

Crear diccionario de métricas

Para ello debe modificar crear una función en `utils.py` que reciba una línea de fichero y devuelva un diccionario python. Llame a dicha función `line_to_dict(fpath,line)` y prográmela para que devuelva el siguiente diccionario al recibir la primera línea del CSV de las constantes vitales:

```
{
  "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.

Problema 1. Guarde el resultado en el campo `lectura` del JSON de respuestas.

Publicar en distintos topics

A continuación vamos a publicar cada métrica de las constantes vitales en un topic diferente. En concreto, vamos a publicar las constantes vitales `hr`, `resp`, `SpO2` y `temp`. Para ello va a modificar el código de su `client.py` para que haga lo siguiente cada vez que lee una línea de constantes vitales:

1. obtener el diccionario de la línea usando `line_to_dict(fpath,line)`; y
2. publicar cada constante vital el un topic usando QoS 1.

Una vez haya modificado en `client.py`, ejecútelo para que reporte las constantes al broker EMQX[5], inicie una captura en Wireshark en la interfaz `lo`. Deje que se publiquen varias constantes vitales y guarde[6] en `publishes-grupoX.pcapng` la captura.

Problema 2. ¿Cuál es el máximo valor de `temp` reportado en la captura? Responda en el campo `maxtemp` del JSON de respuestas.

Suscripción a topic de temperatura

A continuación va a suscribirse al topic `vitals/temp` usando el script `subs.py` que se proporciona en la práctica. Este script lo usaremos para suscribirnos al topic de temperatura a través del broker EMQX.

Una vez haya modificado el script `subs.py`, inicie una captura wireshark. Después ejecute el `client.py` y `subs.py` para comprobar que se reciben las métricas de temperatura. Detenga la captura cuando se hayan recibido un par de métricas de temperatura y guarde la captura en el fichero `subs-temp-grupoX.pcapng`.

Fíjese en la captura de wireshark y responda a las siguientes preguntas:

Problema 3. ¿Cuál es el tipo de mensaje del `Subscribe request`?

Problema 4. ¿Con qué QoS se suscribe al topic `vitals/temp`?

Problema 5. ¿Cuál es el puerto utilizado por el cliente para publicar mensajes?

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

Problema 6. ¿Cuál es el puerto utilizado por el suscriptor para recibir mensajes?

Responda a las preguntas en los siguientes campos del JSON de respuestas: `reqtype`, `subsqos`, `pubport` y `subport`; respetivamente

Suscripción a varios topics

Ahora vamos a suscribirnos a todos los topics de constantes vitales que reporte nuestro `client.py`. Para ello, use un wildcard en el topic al que se suscribe el `subs.py` y compruebe que el suscriptor recibe todas las métricas reportadas por el `client.py`.

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.

Problema 7. ¿A cuántas métricas se suscribe el `subs.py`?

Problema 8. ¿Cuál es el topic del `PUBACK` con `Message identifier` $10 + X$?

Responda a las preguntas en los siguientes campos del JSON de respuestas `nummetricas` y `pubacktopic`.

Alerta por anomalías

Para terminar vamos a generar alertas cuando la métrica `hr` exceda en un $\frac{X}{100}$ por ciento el valor medio de 89 BPM. Para ello debe modificar el script `subs.py` para que:

1. compruebe que recibe un mensaje del topic `vitals/hr`; y
2. envíe un `PUBLISH` en caso de que se cumpla la condición para generar la alerta.

Cuando publique la alerta, hágalo en el topic `alerts/hr` y envíe como mensaje cuáles son los BPM.

Una vez tenga funcionando el script `subs.py` debe iniciar una captura wireshark en la interfaz `lo`. Después inicie el `client.py` y el `subs.py`. Espere a que se publiquen un par de alertas antes de parar la captura, guárdela en el archivo `alertas-grupoX.pcapng`, y responda a las siguientes preguntas:

Problema 9. ¿Cuántas alertas se envían?

Problema 10. ¿Cuál es la QoS con la que envía las alertas?

Responda a las preguntas en los siguientes campos del JSON de respuestas: `numalerts` y `qosalerts`.

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 los **Problemas** correspondientes.