TB067 Redes de Comunicaciones
86.12 Comunicación de Datos

Segundo Cuatrimestre 2024

Trabajo Práctico: Nivel de Aplicaciones y de Transporte del Modelo TCP/IP

Este trabajo es para realizarse de a pares.

La fecha límite de entrega es el miércoles 9 de octubre a las 18:00.

Para la entrega se deberá adjuntar en la tarea correspondiente en el campus, un archivo .zip conteniendo el código de los programas modificados según se indica y un informe en .pdf con las respuestas y justificaciones que se piden. Tanto el .zip como el .pdf deben usar como nombre los apellidos de los estudiantes del grupo.

Todos los resultados obtenidos deben tener su argumentación que dé soporte a lo mencionado. Los tipos de argumentos pueden ser, sin limitarse a los mencionados:

- capturas Wireshark, citando ID de paquete(s)
- desglose y análisis de segmentos TCP
- secciones de código

Nota: en el informe, las capturas de pantalla de Wireshark deben estar pegadas en cada respuesta que lo requiera y deben ser lo suficientemente grandes como para que sean legibles.

Enunciado

Introducción:

MQTT (Message Queuing Telemetry Transport) es un protocolo de capa de aplicación diseñado para facilitar el intercambio de mensajes en redes de dispositivos que requieren conectividad eficiente y bajo consumo. Usaremos dicho protocolo para realizar el presente Trabajo Práctico, en grupos de dos integrantes.

MQTT se basa en un modelo donde los dispositivos pueden ser publicadores (enviador de mensajes) y/o suscriptores (recibidor de mensajes). En esta propuesta uno de los integrantes será publicador y el otro, suscriptor.

Los dispositivos son intermediados por un servidor que recibe los mensajes de los publicadores y los distribuye a los suscriptores interesados, gestionando las conexiones y el tráfico. Dicho servidor se denomina broker. En esta oportunidad

usaremos un broker de acceso gratuito, disponible en Internet. Por tal motivo, es un servicio demostrativo y no garantiza nivel de servicio.

Los mensajes en MQTT se organizan por "temas" (*topics*). Los publicadores envían mensajes a un tema específico, y los suscriptores se registran en los temas de su interés. De esta manera, un dispositivo puede recibir sólo la información que necesita. A los fines de este Trabajo Práctico, cada grupo tendrá un *topic* único, para no interferir con el resto de los grupos. El topic de los grupos deberá seguir la siguiente convención: tp1/[*apellido_1*][*apellido_2*]. Por ejemplo: si el grupo estuviera integrado por Elon Reeve Musk y William Henry Gates III, su topic será: tp1/musk_gates

MQTT no implementa fragmentación de paquetes. No obstante, si los mensajes MQTT se encapsulan sobre un protocolo de transporte como TCP, podría darse una situación de fragmentación, entre estaciones y el broker. Aunque esto sería improbable, dado que, como dijimos anteriormente, se trata de un protocolo orientado a mensajes breves.

Actividades

Leer:

1) Investigar el protocolo MQTT en fuentes públicas y escribir una reseña (versiones, niveles de servicio, encriptación, productos comerciales y de código abierto: hay más conceptos que los mencionados arriba).

Ejecutar los scripts y responder:

2) Los dos scripts Python 3 proporcionados en este trabajo práctico constan de un publicador y un suscriptor. Adicionalmente, se brinda un archivo de texto. El publicador está diseñado para transmitir el contenido del archivo, en tanto el suscriptor está diseñado para recibirlo. La transmisión se realiza mediante la fragmentación del texto en mensajes de largo variable y la recepción por lo tanto recibe fragmentos y los ensambla para escribir el contenido completo en un archivo.
   El objetivo es verificar que el publicador pueda transmitir al suscriptor el archivo completo, siendo que cada integrante lo hará desde una PC diferente, por ejemplo, cada cual en su domicilio particular. Para ello, el

suscriptor debe ejecutarse antes que el publicador. Detalles técnicos de ejecución de los scripts, en el Apéndice.

3) Sobre la aplicación, ¿de qué modo consigue implementar la fragmentación?
4) En comparación con la fragmentación que implementa el protocolo TCP, ¿hay funcionalidades o características de la fragmentación en estos scripts que sean similares a la segmentación TCP? ¿Cuáles?
5) En comparación con la segmentación que implementa el protocolo TCP, ¿hay funcionalidades o características de la segmentación TCP que en estos scripts estén ausentes? ¿Cuáles?
6) Acerca de las sesiones del publicador y del suscriptor, ¿cuáles son los extremos de las sesiones? ¿quién toma el rol de cliente y quién de servidor?
7) Sobre la extensión de las sesiones, ¿son persistentes, cómo se sostienen cuando no hay tráfico? ¿no son persistentes? ¿cómo se cierran (en qué momento y quién lo inicia)?
8) En detalle sobre las sesiones:
    a) ¿Cuál es el número de secuencia del primer segmento TCP de petición de conexión?
    b) ¿Cuáles son las opciones implementadas, si las hay?
    c) ¿Cuántos bytes tiene el buffer de recepción según se informa al inicio?
    d) ¿A qué hora se envió el primer segmento (el que contiene datos de la aplicación)? ¿A qué hora se recibió el ACK de este primer segmento que contiene datos? ¿Cuál es su RTT?
    e) ¿Cuál es la longitud (encabezado más carga útil) de cada uno de los primeros cuatro segmentosTCP que transportan datos?

Implementar:

9) HTTP indica la cantidad de datos a transmitir mediante un encabezado. Esta adecuación de MQTT para transmitir fragmentos no comunica la cantidad de datos que va a transmitir. Si algún paquete se perdiera, el suscriptor no tendría forma de detectar la falta. Modificar el código para simular la pérdida de un paquete intermedio y verificar en el suscriptor que el texto quede truncado. Luego modificar ambos scripts para que se comunique el largo de los datos a transmitir y que el suscriptor pueda validar la cantidad de datos recibidos versus los esperados.

10) De las funcionalidades o características vinculadas a segmentación, presentes en TCP pero no cubiertas por estos scripts, implementar una versión mejorada, que incorpore al menos una mejora o una funcionalidad. Se puede usar inteligencia artificial y por qué no, inteligencia natural.

Apéndice

Ejecutaremos los scripts desde la máquina virtual de la materia, creando un ambiente virtual Python:

A) Tareas a realizar única vez en cada máquina virtual:

A1) Descargar el trabajo práctico desde el Mozilla de la máquina virtual los artefactos para realizar el trabajo práctico.

A2) Abrir consola y actualizar:
apt update
apt upgrade -y

A3) Instalar ambiente virtual:
apt install python3-venv

A4) Crear directorios de trabajo:
mkdir -p tp1_{apellido1}_{apellido2}/scripts
mkdir -p tp1_{apellido1}_{apellido2}/resultados

A5) Cambiar la variable *topic* en los scripts publisher.py y subscriber.py, por ejemplo, con el editor nano, vi, etcétera.

Ejemplo:
topic = "tp1/musk_gates”

A6) Copiar los scripts y el archivo a transmitir:

cp publisher.py tp1_{apellido1}_{apellido2}/scripts/
cp subscriber.py tp1_{apellido1}_{apellido2}/scripts/
cp input.txt tp1_{apellido1}_{apellido2}/scripts/

B) Tareas a realizar cada vez que se quiera trabajar con el TP:

B1) Ingresar al directorio de trabajo
cd tp1_{apellido1}_{apellido2}/scripts

B2) Crear ambiente virtual:
python -m venv tp1

B3) Activar el ambiente virtual (debería cambiar el prompt):
source tp1/bin/activate

B4) Instalar paho:
pip install paho-mqtt
pip install typing-extensions

B5) Correr Wireshark para hacer las capturas ;-)
(existe filtro para MQTT)

B6) El participante con rol suscriptor debe ejecutar primero su script, mientras que el rol publicador lo ejecutará segundo.

(desde la casa del suscriptor)
python subscriber.py

(desde la casa del publicador)
python publisher.py

B7) Para terminar de trabajar, desactivar el ambiente virtual:
deactivate

Los resultados (scripts modificados, informe, capturas, etcétera) copiarlos a la carpeta “resultados” y comprimir todo en zip para la entrega.