URJC Escuela de Ingeniería de Fuenlabrada

Examen Práctica 4: Publicador/Subscriptor

Sistema distribuidos y concurrentes

14 de Junio 2024

La tarea de este examen es ampliar las capacidades desarrolladas en la práctica 4. Es fundamental que mantengas intacta toda la funcionalidad original descrita en la práctica 4. Tu única tarea adicional es incorporar las nuevas características y funciones especificadas en este enunciado. Debes basar tu trabajo en la solución de la práctica 4 que tienes disponible en tul aula virtual

En la nueva implementación, habilitaremos la posibilidad de que cada cliente (ya sea subscriptor o publicador) pueda asignar un nivel de QoS (calidad de servicio). Este enfoque se utiliza ampliamente en servicios donde los usuarios pagan por un comportamiento premium (qos=0), lo que les garantiza un procesamiento de datos considerablemente más rápido.

Para ello, asegúrate que tu estructura de *message* es como la siguiente.

```
struct message {
   enum operations    action;
   char[100]          topic;
   // Solo utilizado en mensajes de UNREGISTER
   int                   id;
   // Solo utilizado en mensajes PUBLISH_DATA
   struct publish data;
   // Nuevo campo de QOS
   int qos;
};
```

Los valores que puede tomar el nuevo campo **QoS** es 0 y 1. La calidad de servicio que vamos a implementar solo está enfocada para los subscriptores, por lo tanto los publicadores no tendrán en cuenta ese valor de qos (aunque utilicen la estructura).

Los subscriptores tendrán la opción de registrarse en el broker con *qos=0* o *qos=1*. Este parámetro de calidad de servicio indicará al broker a cuál grupo de subscriptores debe enviar los mensajes primero. qos=0 representa la calidad de servicio más alta, por lo que los subscriptores registrados con qos=0 recibirán los mensajes antes que los subscriptores con qos=1.

Broker

Deberás realizar los cambios necesarios para poder ajustar tu práctica a los requisitos de calidad de servicio mencionados anteriormente. Además el broker debe funcionar de la siguiente manera.

- *Priorización de Mensajes*: El broker debe priorizar siempre el envío de mensajes a los subscriptores con *qos*=0 sobre aquellos con *qos*=1. Por tanto, el broker nunca debería enviar un mensaje a subscriptores con qos=1 hasta que no haya mandado los mensajes a los subscriptores con *qos=0*.
- *Modo de Envío para qos=0*: Independientemente de la configuración general del broker, los mensajes dirigidos a subscriptores con *qos*=0 siempre deben enviarse utilizando el modo 'justo'. Esta regla se aplica sin excepciones, asegurando una distribución equitativa y eficiente de los mensajes a estos subscriptores.
- *Modo de Envío para qos=1:* En el caso de los subscriptores con *qos*=1, el broker debe enviar los mensajes utilizando el modo que se haya definido mediante la línea de comando en la configuración del broker.

Subscriptor

En cuanto al subscriptor, realiza los siguientes cambios:

- Modificar la traza donde imprimes la latencia para añadir la QOS

```
[SECONDS.NANOSECONDS] Recibido mensaje topic: $topic - mensaje: $data - Generó: $time_generated_data - Recibido: $time_received_data - QoS: $QoS - Latencia: $latency.
```

- Además añade el código necesario para que tu subscriptor puede ser llamado con un nuevo parámetro de qos, que define la calidad de servicio asociada a dicho subscriptor.

```
./subscriber --ip $BROKER_IP --port $BROKER_PORT --topic $TOPIC --qos $QOS
```

Evaluación

Para la evaluación local de esta práctica, asegúrate de arrancar 900 subscriptores, 1 broker, y 1 publicador (todos ellos en local, en tu máquina). Te en cuenta lo siguiente:

- Debes lanzar 3 escenarios/experimentos distintos, que se corresponden con los 3 modos distintos que tiene el broker. Para cada uno de estos escenarios, debes recopilar y presentar las latencias de forma separada, según el QoS asignado a cada subscriptor

- Create un script que te lance los 900 subscriptores, 450 deben tener qos=0 y otros 450 deben tener qos=1. La asignación del qos debe ser aleatoria, por tanto no debes crear 450 suscriptores seguidos con el mismo qos.
- Al menos deja que tu publicador genere 10 datos nuevos, después analiza los resultados. Asegúrate que lanzas el publicador cuando sepas seguro que todos los subscriptores se han registrado.
- Debes calcular la latencia mínima, máxima, media y desviación estándar.
- Recuerda que independientemente del modo usado por defecto en el broker (secuencial, paralelo y justo), el broker **siempre** debe usar el modo justo para los subscriptores con qos=0.
- Debes crear un archivo CSV que contenga exclusivamente seis líneas. La razón de esto es que estamos trabajando con tres experimentos distintos, y para cada uno de estos experimentos, necesitamos mostrar las latencias desglosadas según el QoS (Calidad de Servicio). A continuación, encontrarás un ejemplo del formato que debe tener el fichero. Es importante recordar que todas las latencias deben expresarse en segundos, y la parte decimal debe tener una precisión de microsegundos, es decir, hasta seis cifras decimales (por ejemplo, 0.000000).

```
# Modo, N, QOS, min, max, avg, std  (no incluyas esta línea)

0,  900, 0, $MIN, $MAX, $AVG, $STD
0,  900, 1, $MIN, $MAX, $AVG, $STD

1,  900, 0, $MIN, $MAX, $AVG, $STD
1,  900, 1, $MIN, $MAX, $AVG, $STD

2,  900, 0, $MIN, $MAX, $AVG, $STD
2,  900, 1, $MIN, $MAX, $AVG, $STD
```

Entrega

Debes entregar:

- Código totalmente funcional y que compile en los laboratorios. Debe de poder ejecutar los casos de evaluación que se describen.
- Scripts que hayas utilizado.
- CSV con las 6 líneas que se pide en el apartado de evaluación.
- Un pequeño documento PDF (máximo media página), explicando detalles de implementación y los resultados que has obtenido (csv).

Consideraciones:

- Controla toda la posible concurrencia y condiciones de carrera.
- NO ESTÁ PERMITIDO espera activa.
- Sigue el estilo de código definido para esta asignatura.