Guía de Trabajos Prácticos Nº4

Patrones de Comunicación

Patrón Request-Reply

5-

b) Cuando se detiene el servidor y luego se ejecuta el cliente, el cliente envía el mensaje y queda bloqueado esperando una respuesta. Luego, al conectar nuevamente el servidor, este recibe el mensaje y le envía la respuesta al cliente, este último la recibe y termina su ejecución.

Sí, en el protocolo HTTP por ejemplo, si el servidor no está disponible para responder a una solicitud, el cliente se quedaría esperando una respuesta de no ser que maneja los posibles fallos del servidor con errores de timeout.

Creo que esto se debe a que al enviar el mensaje a una dirección y puerto determinados, si el servidor no está en el momento para recibir la petición, queda guardado en una cola hasta que este esté disponible.

c) La diferencia principal entre listar los puertos abiertos en una máquina que actúa como cliente y en una que actúa como servidor es el tipo de conexiones que estarán abiertas. El cliente normalmente tendrá puertos abiertos para las conexiones salientes y el servidor tendrá puertos específicos abiertos y escuchando conexiones entrantes.

Patrón Publisher-Subscriber

1-

Si se apaga uno solo de los dos programas, el otro sigue funcionando normalmente.

a) Si no hay cliente, los mensajes del publisher son enviados pero nadie los recibe, por lo que se pierden.

b) Lo que me llama un poco la atención es que si el publisher se detiene y se vuelve a ejecutar mientras hay subscribers, los subscribers siguen recibiendo los mensajes del publisher normalmente. Tal vez sea por el tiempo en que se realiza la conexión, pero sólo se pierde el primer mensaje cuando se vuelve a encender el publisher.

2- Al tener más de un subscriber, ambos reciben los mensajes del publisher.

3- Al intentar ejecutar en una segunda consola otro publisher, hay un error porque la dirección 'tcp://*:3000’ ya está en uso.

4- Si se cambia el tópico en el subscriber o en publisher, el subscriber no va a recibir los mensajes del publisher.

5- Sí, un publisher puede manejar múltiples tópicos simultáneamente y un socket subscriber puede suscribirse a varios tópicos al mismo tiempo.

Patrón Push-Pull

1- Si se apaga el producer, el worker queda a la espera de mensajes, pero si se apaga el worker, a diferencia del publisher que seguía enviando mensajes que se perdían, el producer deja de enviar mensajes hasta que un worker consume el mensaje pendiente.

2- Si se ejecuta más de un worker en simultáneo, estos esperan mensajes en una cola, al recibir un mensaje el primer worker de la cola, pasa al final de la cola y el worker que seguía recibe el siguiente mensaje y así sucesivamente. En cambio, en el patrón pub-sub, todos los subscribers recibían todos los mensajes.

Probando portabilidad

Se creó un worker escrito en C usando la librería czmq, el código se encuentra en el archivo worker.c