Simulador Panadería

Índice

Explicación del programa

El programa consiste en un simulador de una panadería en que unos clientes realizan pedidos a un servidor, y éste a su vez gestiona los pedidos en base a un algoritmo de entre dos posibles **FIFO** (First In First Out) y **SJF** (Shortest Job First), que a su vez se gestionan mediante una *Queue* y una *PriorityQueue*.

El servidor gestiona una cantidad (fijada al inicio del programa) de hornos a los que va a asignando pedidos en función del algoritmo elegido.

A su vez, los hornos, una vez alcanzado un número máximo de ciclos de horneado (fijado también al inicio del programa) realizarán un mantenimiento, quedando inutilizados durante 5 segundos. Pasarán así al estado **MANTENIMIENTO**.

Los hornos pueden tener 3 estados diferentes: **DISPONIBLE** (puede recibir pedidos), **MANTENIMIENTO** (5 segundos en mantenimiento) u **OCUPADO** (el horno ya contiene un pedido en curso).

Los productos disponibles se establecen al inicio del programa y se pueden cambiar en el método "configurar_productos" del servidor, ya que este método únicamente retorna una lista de productos disponibles y se usa para especificar a los clientes qué productos están disponibles.

A su vez, cada cliente tras recibir la lista de productos disponibles, realiza un pedido con una cantidad aleatoria no nula (1-10).

El flujo de mensajes es el siguiente:

1. Se inicia el servidor y se mantiene a la escucha.
2. Un cliente se conecta.
2.1. El servidor crea la clave RSA y envía la clave pública.
2.2. El cliente recibe la clave pública y crea una clave AES.
2.3. El cliente envía la clave AES codificada con RSA al servidor. Y entonces envía su nombre codificado con AES.
2.4. A partir de aquí la comunicación es encriptada.
3. El servidor envía la lista de productos disponibles.
4. El cliente realiza el pedido en base a esa lista.
5. El servidor encola los pedidos en una cola para gestionarlos al recibir todos los pedidos.
6. Una vez recibidos, se asignan los pedidos a los hornos en función del algoritmo seleccionado.
7. Los hornos cocinan los pedidos hasta que no queden pedidos por asignar.
8. Tras cocinar cada pedido, los hornos notifican al servidor.
9. El servidor notifica al cliente que el pedido está listo.

Suposiciones

Una de las primeras suposiciones realizadas es el mantenimiento: ya que no se especifica nada y es algo relativamente poco importante, he optado por establecer un periodo de 5 segundos, ya que considero que es un tiempo en el que es más fácil un pedido pueda intentar asignarse a un horno en mantenimiento, y por tanto, se reciba el mensaje correspondiente sobre el estado del mismo.

Otra suposición, o más bien un cambio realizado en la lógica es la sustitución del algoritmo SRTN (Shortest Remaining Time Next) por el FIFO. Dada la naturaleza del programa y la lógica escogida, no tiene sentido un SRTN, pero sí un FIFO (se parece más a cómo funciona una panadería). Y se ha mantenido el SJF.

Por otra parte, he decidido que la configuración inicial de los productos sea estática y no forme parte de la configuración introducida por teclado en el código del servidor para no sobrecargar y ralentizar la misma, aunque se sigue pudiendo cambiar antes de iniciar el servidor, ya que está en un método a parte (configurar_productos).

Puesto que tanto Queue como PriorityQueue implementan los mecanismos necesarios para la programación multi-hilo, no se han empleado locks para el acceso a las mismas.

Posibles mejoras

Puesto que aún me falta mucho por aprender, podría decir que todo el código es mejorable, pero centrándome en la funcionalidad:

- Clientes dinámicos en lugar de clientes ficticios que realizan llamadas aleatorias.

- Uso de librerías para realizar notificaciones push a iOS y Android (ya que no he dado alcance a este punto).

- Implementar el cliente sin print() de manera que se pueda usar correctamente la consulta del estado de los pedidos.