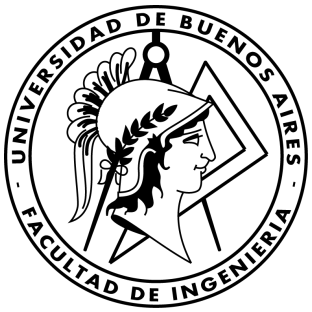


Trabajo Práctico - Agentes Autónomos de Prevención

[75.42/95.09] Taller de Programación
1C 2024

Grupo: Los Raviolines

Integrantes:

- Albornoz Tomé, Rodolfo Valentin - Padrón 107975
- Pereyra, Francisco Antonio - Padrón 105666

Correctores:

- Campodónico, Alfonso
- Masri, Noah

Índice

1. Introducción

En este trabajo práctico, nuestro objetivo era implementar un sistema de prevención de incidentes a través del manejo de drones. Para eso, se debía construir un sistema de mensajería usando algún protocolo de message broking, principalmente enfocados en el patrón de comunicación publisher/suscriber, donde un publicador realiza una publicación a un tópico, y el sistema envía esa publicación a todos los que están suscritos a ese tópico. En el caso de este TP, usamos el sistema de mensajería NATS.

2. Comparación de protocolos y elección de protocolo NATS

Realizamos una investigación de varios protocolos de message broking, principalmente consideramos 3 protocolos:

- **AMQP**: Es un protocolo que sirve para la intermediación de mensajes. Algunas de sus características mas importantes son que es un protocolo binario, por lo que hay mayor eficiencia en la comunicación. También es confiable ya que permite la confirmación y persistencia de mensajes, es flexible ya que permite varios patrones de mensajes (Aunque solos nos interesa publisher/suscriber), es seguro porque permite autenticación, autorización y encriptación, y tiene un sistema de queues (Colas) que guardan los mensajes hasta que se consuman.
  - **Ventajas**: Abarca varios puntos importantes como la seguridad y fiabilidad, incorporados en su arquitectura. Permite varios patrones de mensajes.
  - **Desventajas**: Es pesado y complejo, ideal para sistemas a gran escala o del mundo más real. Quizás se aleja del alcance de lo pedido en el enunciado.
- **MQTT**: Es un protocolo simple, que se maneja con conexiones con ancho de banda limitado. Algunas de sus características mas importantes son que es un protocolo ligero que utiliza pocos recursos y ancho de banda, que es simple y solo maneja el modelo de publisher/suscriber, que ofrece todos los niveles de Calidad de Servicios (QoS) y demás. Se maneja exclusivamente con el modelo de publisher/suscriber, con un broker en el medio que hace de intermediario entre ambos, y los tópicos que sirven como vía de comunicación entre publisher y suscriber.
  - **Ventajas**: Es fácil de implementar y usar, abarca varios puntos pedidos en la consigna.
  - **Desventajas**: Su gran simplicidad también va con la contra de que es quizás limitado para nuestro caso.
- **NATS**: Es un protocolo de mensajería con arquitectura simple que tiene su funcionalidad principal llamada NATS Core, y su propia capa de persistencia de mensajes llamada NATS JetStream, que es un agregado de NATS Core. Algunas de sus características más importantes son que es simple y eficiente, con baja latencia y alto rendimiento, que es escalable y puede manejar muchas conexiones, que ofrece clustering y soporta arquitectura distribuida para alta disponibilidad, y que no tiene persistencia incorporada, sino que puede integrarse con JetStream (Capa de persistencia). Se maneja con un sistema de cliente-servidor en el que los subjects (Tópicos) son la comunicación entre ellos, donde el cliente realiza una suscripción para recibir mensajes de un subject. Además, permite un patrón de comunicación request/response, donde un cliente puede enviar una solicitud y recibir una respuesta de otro cliente.
  - **Ventajas**: Está muy bien documentado, es muy utilizado, su arquitectura es simple y abarca varios de los puntos pedidos en la consigna.
  - **Desventajas**: Quizás es mas complejo al no ser como otros protocolos, por ejemplo incorporando por separado la capa de Jetstream que tiene su complejidad. Tiene también muchas funcionalidades que no pensábamos utilizar en nuestro trabajo.

Considerando estos 3 protocolos, decidimos elegir el de NATS por 2 motivos.

El primero, es que lo consideramos un punto intermedio entre AMQP (Que es complejo pero abarca muchos puntos) y MQTT (Que es simple, ligero y con solo un patrón de mensajes), manteniendo simpleza pero abarcando todas las características pedidas por el enunciado.

El segundo, es que está muy bien documentado, tanto NATS Core como Jetstream, ya que está la página de NATS y además están los videos de Synadia explicando ventajas y beneficios de NATS, y partes de la arquitectura de NATS con diagramas muy claros y entendibles.

3. Características del servicio de mensajeria

Nuestro trabajo práctico debía cumplir con algunos requerimientos, los cuales son:

- **Seguridad (Autenticación, autorización, encriptación)**: Se realiza agregando canales seguros de tls a las conexiones de cada cliente al servidor, verificando usuario y contraseña al conectarse al servidor, comparando con un archivo de cuentas recibido por el servidor, etc.
- **QoS (Quality of Service)**: Se implementa 'at least once' utilizando NATS Jetstream, que es la capa de persistencia de NATS.
- **Reliability**: También resuelto em parte por NATS Jetstream. Por el otro lado, si se sufre una desconexión, se hace el reintento.
- **Configuración**: Se reciben o por archivo o por consola de comando todos los parámetros necesarios, que se cargan en una configuración y se obtienen de la misma cuando es necesario.
- **Logging**: Se realiza un registro, desde el servidor, de las conexiones recibidas, los clientes conectados, las publicaciones enviadas, las sucripciones realizadas, etc.

4. Servidor

El servidor es la parte más importante del servicio de mensajería, y es el que gestiona la lógica, los hilos, las conexiones de clientes, el manejo y envío de mensajes, etc. El servidor puede tener varios hilos, los cuales manejan y se distribuyen las conexiones realizadas y los mensajes enviados, y cada hilo puede tener asociado conexiones, que son realizadas cuando un cliente se conecta al servidor. Todos los eventos significativos, tanto para hilo como conexión, se realizan desde el registrador propio de cada estructura, que se crea desde el servidor.

El sistemá está separado en varias partes que se realizan simultaneamente (Mediante threads): La registración de eventos, el manejo de nuevas conexiones, y las acciones que van ocurriendo en el n-esimo hilo.

El servidor está conformado principalmente. por la configuración, los hilos, un registrador y las cuentas (Si las hay, se cargan desde la configuracion en el servidor). Los hilos tienen id único están formados principalmente por canales para comunicarse a otros hilos, para recibir conexiones, las suscripciones, un registrador y las conexiones. Y a su vez, las conexiones tienen id único y están formados por el flujo de mensajes con el cual se comunican con el cliente, el registrador, un parser (Para procesamiento de mensajes), las cuentas del usuario, y flags para hacer seguimiento de la autenticación y conexión.

El flujo del servidor es el siguiente: Se toma la configuración ya sea por comandos o por una configuración existente, se crea el servidor junto a los elementos que irán dentro de los hilos y los canales para comunicarse entre ellos y el registrador.

En cada hilo se va a asignar un emisor para enviar instrucciones a los demas hilos. Se crea el canal para recibir nuevas conexiones desde el hilo, se crea un registrador por hilo, se asocian a los hilos, y se crea el hilo.

En el proceso del hilo, se realizan 4 acciones principales:

- 1) Recibir conexiones: Mientras se reciban conexiones, se guardan en el hilo, y el registrador informa de una conexión recibida.
- 2) Recibir instrucciones:Se recibe una instrucción desde el hilo y se procesa, y su acción se traslada a los demas hilos y conexiones.
- 3) Evaluar ciclo de conexiones: Se encarga de la lógica de cada conexión, manejando las instrucciones y mensajes, etc.
- 4) Eliminar conexiones terminadas: Se toman todas las suscripciones de las conexiones terminadas, y se desuscribe de cada una. Además, el registrador informa cada conexión terminada.

Mientras, en el servidor, se obtiene dirección y puerto de la configuración, y constantemente se escucha por nuevas conexiones. Al recibir un flujo de mensajes (Stream) válido, se crea nueva conexión, se crea el registrador para el mismo y se le asocia el hilo actual. Despues, se envía la conexión al hilo y se actualiza el proximo id de hilo. Es importante señalar que el stream se setea en modo no bloqueante para no bloquear a las demas conexiones.

5. Cliente, funcionamiento y arquitectura

El cliente es la libreria que permite la comunicación con el servidor. Las cámaras y drones por ejemplo son clientes. El cliente recibe mensajes e instrucciones

El cliente está conformado por un hilo, el cual maneja toda la lógica, donde se gestionan los mensajes, el canal por el cual se envían mensajes al servidor, y un identificador único. El hilo está formado principalmente por el flujo de mensajes por el cual se envían y reciben mensajes del servidor, los canales para recibir mensajes y de suscripciones, un parser (Para manejar la lectura de mensajes) y datos para la autenticación como el usuario y contraseña.

El flujo del cliente es el siguiente: Se crea el hilo el cual maneja los mensajes enviados por el cliente. Se reciben usuario y contraseña si son necesarios, se esperan mensajes y no se reciben instrucciones hasta que no se realice la autenticación.

Del stream se leen mensajes, y con ayuda del parser, se verifica el mensaje, se verifica si está en formato correcto (Por ejemplo un mensaje de publicar con los argumentos correctos), y se gestiona el mensaje.

Si se realizó la autenticación, se van recibiendo y gestionando instrucciones en las cuales se escribe al flujo de mensajes.

Todo este proceso se hace constantemente mientras el cliente esté conectado.

Por el lado de las suscripciones, cuando el cliente realiza una, se envía al servidor una instrucción de suscribir con la información de la misma y se le da la suscripción una forma de enviar publicaciones para comunicarse con el server.

Por el lado de las publicaciones, contienen el tópico al cual se escribe, el mensaje que se va a mandar y otra información adicional, y se envían por el canal de instrucciones del cliente

6. Sistema de Cámaras, funcionamiento y arquitectura

La característica de este cliente es que es todo un sistema con varias cámaras, no una sola, y el sistema maneja cada cámara individual. Además, se puede interactuar con el sistema para pedirle realizar acciones mediante comandos por consola.

El sistema está separado principalmente en 2 partes que se van realizando simultaneamente (Mediante threads): La interfaz, para el recibimiento de comandos, y la lógica del sistema completo y las camaras.

El sistema está formado por un conjunto de camaras, las camaras lindantes a cada cámara, y los incidentes que van a revisar las camaras. También posee una configuración que se va a recibir

como entrada o de algún archivo de configuración existente. Y por último canales para recibir comandos y para responder a ellos.

El flujo del sistema de cámaras es el siguiente: En el hilo de la interfaz, mientras el sistema esté activo, se espera el recibimiento de un comando, como conectar/desconectar una cámara, actualizar el sistema, modificar una cámara, etc, y el sistema actúa en consecuencia según el comando recibido.

Por el lado del hilo de la lógica, se crea el sistema de cámaras, se cargan las cámaras por archivo, se conectan y se calculan las lindantes a cada una. Luego, se calculan los incidentes primarios y secundarios de cada cámara.

Mientras el sistema esté activo, se utiliza la libreria del cliente para conectarse al servidor con la información de usuario, contraseña, dirección y puerto de conexión, y constantemente se actualiza al servidor del estado de las camaras y se guardan en archivo.

Entonces, se realizan suscripciones a incidentes, a comandos para recibir información de estos.

- Por el lado de la suscripción a incidentes, se esperan publicaciones de nuevos incidentes, se deserializa para obtener la información del incidente, se carga y se envía actualización al sistema. Según la cámara se añade como incidente primario o secundario.
- Si no hay publicaciones de nuevos incidentes, se pueden recibir publicaciones de incidentes terminados, deserializando el mensaje, terminando el incidente y enviando actualización al sistema.
- Si no hay publicaciones de incidentes finalizados tampoco, se pueden recibir publicaciones para actualizar los incidentes y enviar actualización al sistema.
- Por el lado de la suscripción a comandos, se esperan comandos para realizar una acción sobre el sistema o sobre una cámara en particular, y se actúa en consecuencia. Y para los comandos remotos

7. Drones

Los drones son autónomos e independientes entre sí, y cada uno maneja sus acciones y la lógica para ir a resolver un incidente o volver a la central de recarga de ser necesario.

El dron contiene la información de su posición y desplazamiento, la posición y alcance de su central, su batería (Actual, minima, tiempo de recarga, etc), los incidentes primarios (En rango) y secundarios (Al alcance máximo), la información que recibe de otros drones, el incidente que está atendiendo actualmente, y su estado actual.

El flujo del dron es el siguiente: El dron se carga ya sea pasando un id y un archivo de drones, creando el dron del archivo con ese id, o pasando por comando los valores del dron. Luego se inicia el dron, y se guarda su información en el archivo de drones.

El dron se suscribe a los incidentes creados y finalizados, recibiendo información de incidentes para atender, y también información de los demás drones y los incidentes que están atendiendo, para que a la hora de ir a un incidente, sepa si ya hay drones atendiendo un incidente que tiene en rango.

Entonces, el dron publica a los demas drones su información para dejar conocer a los demás drones, y va descargando su batería constantemente, yendo a la central a recargarse y dejando de atender incidentes de drones.

El dron constantemente actualiza los incidentes activos que tiene en rango, y también los incidentes que finalizaron y a los que ya no debe ir, asi como también actualiza la información que recibe de los demás drones y de sus cambios de estados, ubicación y demas.

Para la resolución de incidentes, si el dron no está resolviendo incidentes, busca el incidente más cercano, verifica si puede ir hasta él con su batería y si no hay ya 2 drones atendiendo el

incidente, y si se cumplen las condiciones, va al incidente, y cuando llega, espera a que otro dron lo ayude a resolver el incidente, para luego de resolverlo, volver a la central.

8. Sistema de monitoreo e Incidentes

En el sistema de monitoreo, se pueden ver visualmente todas las partes del sistema de mensajeria funcionando e interactuando entre sí: Los incidentes cargados en el mapa (Por archivo), las cámaras con su rango alcance activándose si tienen incidentes en alcance, y los drones yendo a resolver incidentes. El sistema se inicia ingresando usuario y contraseña, el puerto y dirección del servidor.

Al igual que el sistema de cámaras, el sistema de monitoreo está separado principalmente en 2 partes que se van realizando simultaneamente: La interfaz gráfica y la lógica del sistema.

El sistema de monitoreo está formado principalmente por el conjunto de cámaras, incidentes y drones con el estado de conexión del sistema y algun mensaje de error si surge en algún momento, la configuración y los canales para recibir comandos y comunicar el estado.

El sistema de monitoreo funciona de la siguiente forma: Cuando se corre, se crean canales donde la lógica del sistema recibe comandos y donde envía el estado, y la aplicación gráfica es la que le envía los comandos y la que recibe el estado de lo que le informa la lógica del sistema. Entonces, se crea el sistema, se cargan los incidentes y se inicia el sistema en bucle hasta que se desconecte.

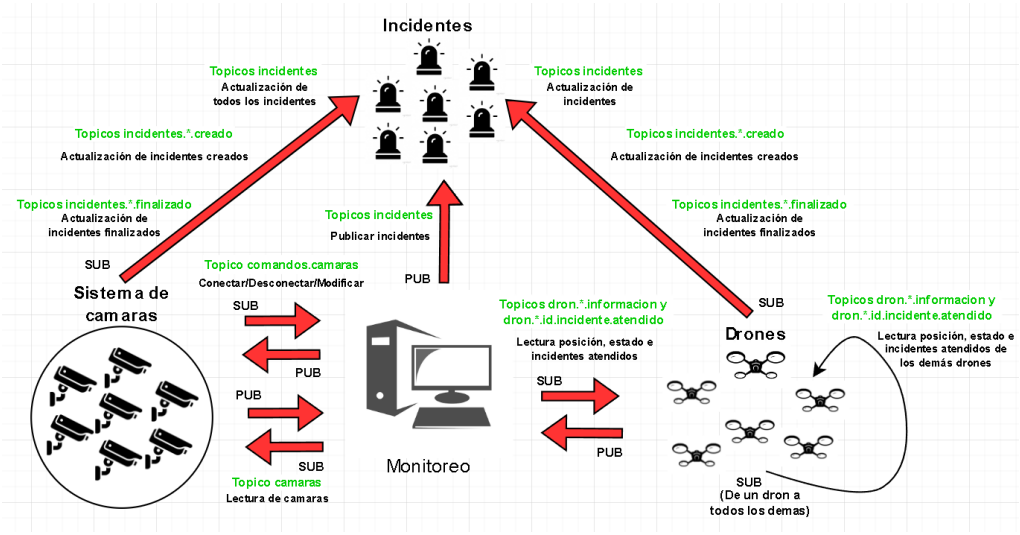
Se conecta el cliente al servidor a partir del usuario, contraseña, dirección y puerto de la configuración. El monitoreo se suscribe a los comandos del monitoreo conectado, y si se lee una publicación en cierto tiempo, se conecta al sistema. Luego, se publican y guardan todos los incidentes.

Se realiza suscripciones a las camaras, los comandos de monitoreo y los drones. Se actualiza estado de la UI enviandolo hacia la interfaz, y se solicita actualización de camaras al servidor.

- Por el lado de la suscripción a camaras, se leen publicaciones de camaras, se conectan y se cambia el estado del sistema, enviandolo hacia la interfaz.
- Por el lado de la suscripción a drones, se leen los drones y la información de los incidentes que atendieron, se conectan los drones y se procesan los incidentes finalizados por los drones
- Por el lado de la suscripción a comandos, se leen desde la interfaz y se procesan, pueden ser de incidentes (Crear, modificar, finalizar), camaras (Modificar, conectar/desconectar) o desconectar el sistema. Si de la suscripción a los comandos de monitoreo se recibe un actualizar, se guardan los incidentes en archivo y se publican los incidentes.

Por el lado de la aplicación gráfica, se muestran los iconos correspondientes, y se puede interactuar con camaras, incidentes y drones, pudiendo verlos desde un listado, modificarlos (Excepto drones), etc.

9. Arquitectura del sistema



10. Conclusiones

En este trabajo, aprendimos a realizar un proyecto de mayor escala que en otras materias, primero en un lenguaje de programación que no conocíamos (Pero que nos recordó a C, visto en Algoritmos y Programación I y II), aprendiendo sus conceptos únicos como ownership y otros, despues aplicando temas nuevos como sockets, clientes y servidores y programación asincrónica, y luego utilizando elementos que se utilizan en los proyectos reales de hoy en dia, como una herramienta de control de versiones, pruebas unitarias y de integración, pair programming, etc. Por último también fue necesaria una gran investigación por internet ya que muchos conceptos eran desconocidos por nosotros y además tuvimos muchas trabas en el camino.

Este trabajo práctico nos pareció semejante a lo que podría ser un proyecto de la vida real pero en menor escala y con menos gente, y del mismo sacamos muchas enseñanzas, mejoras y más.