Mr. J. System

Fase 3: The Queen of Distortion

Sistemes Operatius – Curs 24/25

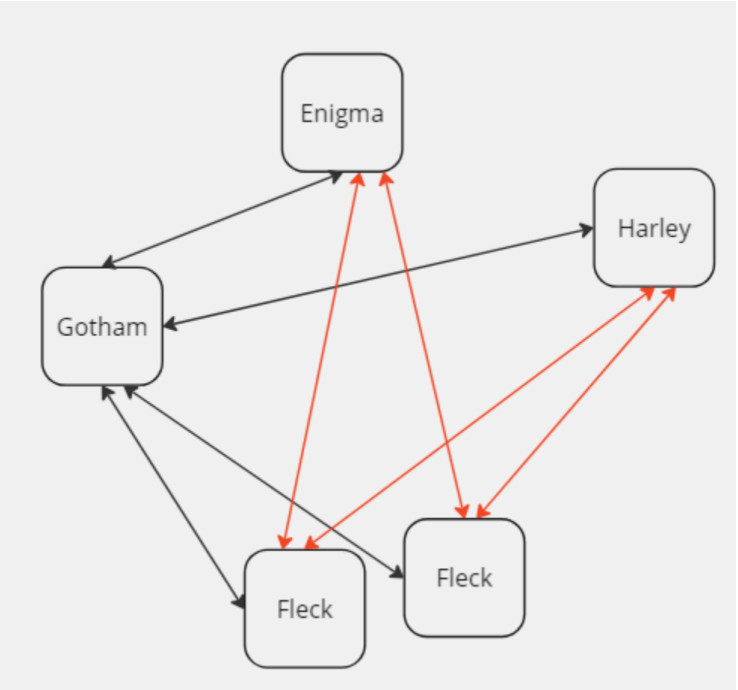
Alvaro Bello Garrido (alvaro.bello)

Carla Francos Molina (carla.francos)

15/12/2024

Índice

1. Diagrama de procesos



Se ha diseñado la Fase 3 de la siguiente manera: Dependiendo del archivo a distorsionar, Gotham le enviará los datos de un servidor Enigma o Harley a Fleck para que se conecte con él mediante sockets de nuevo y utilizando threads debido a que Fleck podrá realizar una petición de distorsión tipo Texto o una petición de distorsión tipo Media, por lo que se podrá conectar con un servidor Enigma y Harley al mismo tiempo (a su vez, el servidor Worker también administrará las peticiones de distorsión por parte de los Fleck mediante threads ya que pueden recibir peticiones de varios Flecks), y finalmente, Fleck enviará el archivo a distorsionar al Worker especificado para que este lo distorsione y se lo devuelva distorsionado.

La comunicación entre procesos Harley se implementa con memoria compartida y los ficheros se envían al worken mediante colas de mensaje.

2. Estructuras de datos

Respecto a las estructuras de datos se mantienen las estructuras creadas en la fase 1. Qué són: GothamConfig, HarleyConfig/EnigmaConfig y FleckConfig estas son las estructuras clave que permiten almacenar la configuración de cada componente. La elección de estas estructuras se han pensado así por la necesidad de almacenar datos de forma dinámica como direcciones IP, puertos o directorios, que pueden cambiar según el archivo de configuración.

También se mantienen las estructuras creadas en la fase 2 una llamada Server, esta es un elemento clave en la arquitectura de Gotham, ya que proporciona un modelo organizado para implementar servidores en cada componente, permitiendo que las conexiones sean manejadas de manera estructurada y escalable, diseñada para gestionar las conexiones de red de Gotham de manera eficiente. La otra estructura es la de TramaResult en la que se almacenan datos de la trama leída.

En esta fase se ha implementado una nueva estructura llamada WorkerFleck:

```
typedef struct {
    char* IP;   // IP de Worker
    char* Port;  // Puerto de Worker
    char* workerType;
    int socket_fd;
    int status; // Estado de la distorsión en marcha [0-100%]
} WorkerFleck;
```

Esta estructura sirve para gestionar y almacenar información sobre los workers conectados a Gotham. Se usa para supervisar múltiples conexiones simultáneas, saber el progreso de las tareas asignadas, y realizar un seguimiento de los Workers conectados a Gotham.

3. Recursos del sistema utilitzats

Primero se debe iniciar el proceso Gotham, el cual creará dos forks, uno para crear un servidor a la escucha de los Fleck y otro para crear un servidor a la escucha de los Worker.
Después se deben iniciar y conectar los procesos Harley y Enigma con Gotham (el cual guardará su información en una lista con los Workers disponibles), estas conexiones se establecerán utilizando sockets, y se utilizarán threads para manejar cada comunicación Gotham-Worker. También cabe mencionar que estas conexiones se mantendrán estables manteniendo a Worker a la espera de que se cierre Gotham y viceversa. Después de esto, Worker se mantendrá a la espera (a través de otro socket Servidor) de que le lleguen peticiones de distorsión por parte de los Fleck.
A continuación Fleck se conectará con Gotham (utilizando también sockets para establecer la conexión y threads para manejar cada conexión Gotham-Fleck desde Gotham) y realizará peticiones de distorsión. Dependiendo del archivo a distorsionar, Gotham le enviará los datos de un servidor Enigma o Harley a Fleck para que se conecte con él mediante sockets de nuevo pero sin necesidad de utilizar threads debido a que Fleck solo se podrá conectar con un servidor (sin embargo, el servidor Worker sí administrará las peticiones de distorsión por parte de los Fleck mediante threads), y después Fleck enviará el archivo a distorsionar al Worker especificado.

4. Opcionales implementados

Las tramas HEARTBEAT son mensajes periódicos enviados entre componentes de un sistema distribuido para verificar su disponibilidad. Su propósito principal es detectar desconexiones o fallos en los componentes y mantener las conexiones activas.

En este proyecto, hemos implementado el envío de tramas HEARTBEAT entre los Workers y Flecks hacia Gotham. Cada componente envía una trama HEARTBEAT a intervalos regulares, que incluye su identificador.