METROPOLITANA

UNIDAD CUAJIMALPA

LICENCIATURA EN TECNOLOGÍAS Y SISTEMAS DE INFORMACIÓN.

Sistemas distribuidos

Mini Plataforma de Video Streaming P2P con Microservicios

Profesor:
Dr.Guillermo Monroy Rodríguez

Alumno:
Mateo Dorantes Andres-2233030470

Introducción

El proyecto que aquí se presenta consiste en un desarrollo para una mini plataforma de videos streaming peer to peer (P2P) esto implementado en base al lenguaje python y teniendo una pequeña simulación de publicador subscriptor (pub/sub) esto con la finalidad de simular este tipo de plataformas.

Objetivo

El objetivo con el cual se realiza este proyecto es desarrollar un sistema distribuido de pequeño tamaño para comprender estas dos arquitecturas manejadas y saber cómo sucede este intercambio de información

Objetivos Específicos

1. Dividir un video en 10 fragmentos.
2. Implementar nodos que intercambian fragmentos entre ellos (P2P).
3. Microservicio o API que gestione qué fragmentos tiene cada nodo.
4. Pub/Sub para notificar la disponibilidad de fragmentos a otros nodos.
5. Interfaz básica (consola) para pedir y recibir fragmentos.
6. Crear contenedores (Docker), Swagger

Tecnologías usadas

Dentro de las tecnologías manejadas en este proyecto son:

- **Lenguaje:** Python 3.10
- **Framework REST:** Flask
- **Comunicación P2P:** Sockets TCP
- **Mensajería Pub/Sub:** Broker TCP simulado
- **Contenerización:** Docker, Docker Compose
- **Documentación API:**Swagger

Arquitectura del sistema

El sistema se compone de los siguientes módulos:

Módulo	Descripción
video_splitter.py	Divide un video en 10 fragmentos binarios.
peer_to_peer.py	Nodo P2P que envía/recibe fragmentos por sockets TCP.
fragment_manager.py	API REST que lista los fragmentos disponibles en un nodo.
broker.py	Servidor Pub/Sub que gestiona suscripciones y publicaciones.
subscriber.py	Cliente suscriptor que recibe notificaciones de fragmentos nuevos.
notify_fragment.py	Publicador que avisa al broker sobre un nuevo fragmento.
Dockerfiles	Imagen de cada servicio.
Docker.broker	Configura el contenedor para los socket del broker.py
docker.subscriber	Configura un contenedor para los sockets del subscriber.py
docker-compose.yml	Orquesta todos los contenedores.
swagger.yaml	Construcciones para mejor visualización y documentación del proyecto.

Casa abierta al tiempo

Casa abierta al tiempo
UNIVERSIDAD AUTÓNOMA
METROPOLITANA
Unidad Cuajimalpa

Procesos de construcción

División del Video

1. Se creó video_splitter.py para leer un archivo de video y dividirlo en 10 fragmentos binarios (fragment_0.bin … fragment_9.bin).
2. Cada nodo tendrá inicialmente un subconjunto de estos fragmentos.

Comunicación P2P

1. peer_to_peer.py implementa un servidor TCP que responde con fragmentos al recibir el nombre del archivo.
2. También actúa como cliente para solicitar fragmentos a otros nodos.

Microservicio REST

1. fragment_manager.py lista los fragmentos disponibles en el nodo.
2. Se añadió integración con flasgger para generar documentación Swagger en /apidocs.

Sistema Pub/Sub

1. broker.py recibe mensajes SUB:<topic> y PUB:<topic>:<mensaje>.
2. Mantiene una lista de suscriptores por topic y reenvía mensajes publicados.
3. subscriber.py se conecta al broker y escucha un topic.
4. notify_fragment.py publica en el topic fragmentos cada vez que un nodo obtiene un nuevo fragmento.

Contenerización

1. Se crearon Dockerfile para fragment_manager.py, Dockerfile.broker y Dockerfile.subscriber.
   docker-compose.yml levanta todos los servicios juntos.

Documentación Swagger

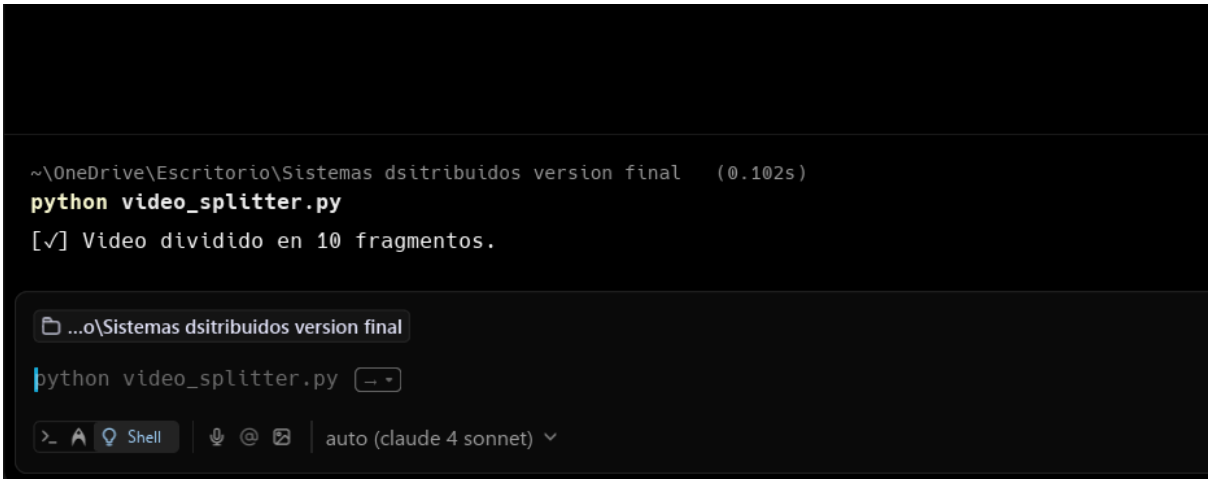
1. El archivo swagger.yaml describe el endpoint /fragments.
2. flasgger genera la interfaz gráfica de pruebas en /apidocs.

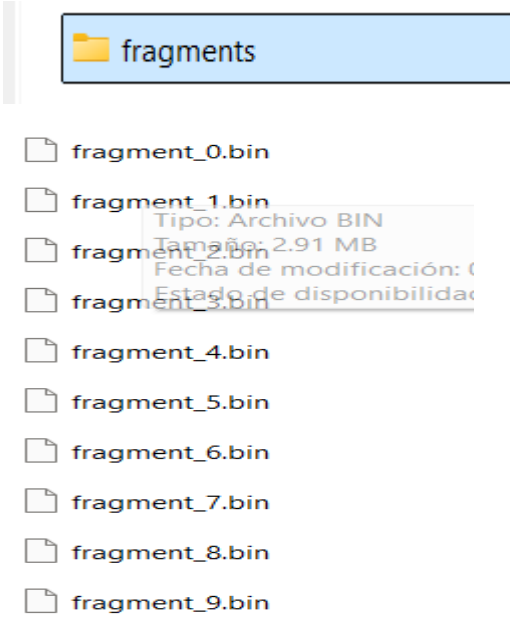
Casa abierta al tiempo

Casa abierta al tiempo
UNIVERSIDAD AUTÓNOMA
METROPOLITANA
Unidad Cuajimalpa

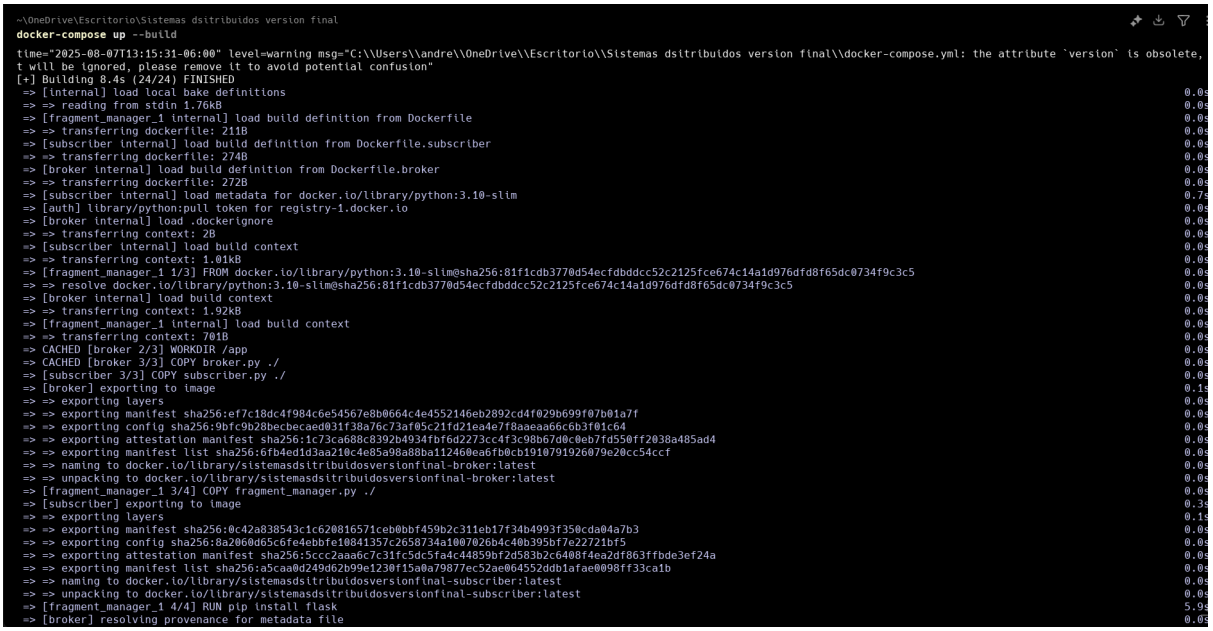
Guía de pruebas

en estas primeras imágenes se puede ver la ejecución correcta del video_splitter.py en donde se divide en diez fragmentos el archivo mp4 se crea una nueva carpeta fragments en donde estarán los archivos divididos.



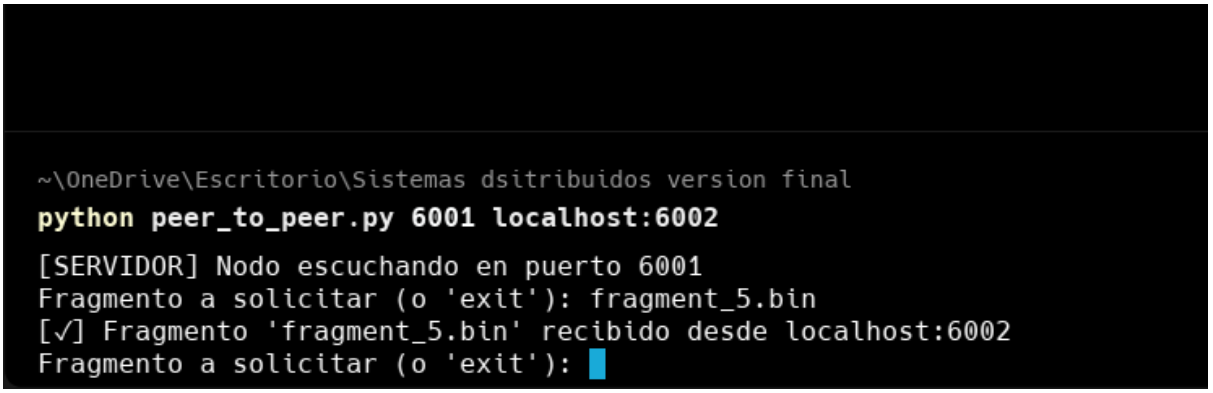


En esta imagen solo se ejecuta el comando para dockerisar el proyecto y se pueda manejar mediante contenedores.

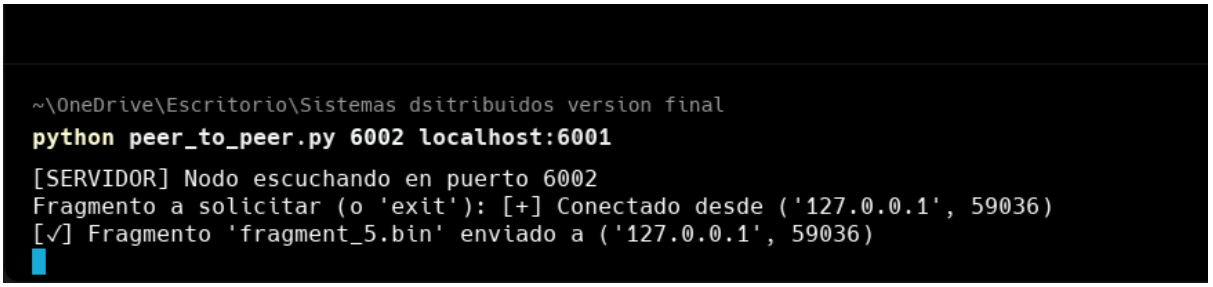


Ya en esta prueba Se ejecuta el fragment_manager para que reparta los fragmentos de videos en tre los dos nodos.

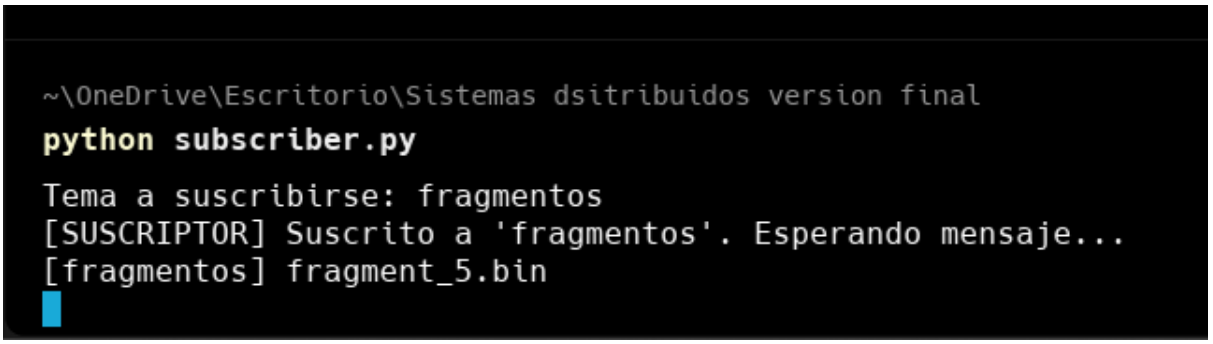
En esta prueba se ejecuta uno de los nodos poniendo el nombre del codiji su puerto de origen u puerto de destino y se pregunta cuál nodo se solicita en donde se recibe el fragmento solicitado del nodo que lo tiene



En el caso del nodo que tenía el fragmento de video se pone el comando inicial de la misma forma pero en este caso ,muestra el nodo enviado con éxitos



En esta imagen de la prueba se ejecuta el suscriptor al tópico de fragmentos donde se queda a la espera de un mensaje UNA VES EJECUTADO EL notificador se le muestra el fragmento de video disponible para ver



En esta prueba se manda la notificacion al suscriptor del fragmento de video disponible

Conclusiones

Para terminar de o ques e puede concluir con este proyecto es que se entendieron los fundamentos y funcionamiento de los sistemas distribuidos y aunque sea en poca escala se puede recrear de una manera los servicios más famosos que reconocemos además de aprender nuevas técnicas y soluciones de problemas de este tipo por que que se llevan bastantes lecciones aprendidas la única gran dificultad fue realizar la integración de schwarzenegger ya que se intentó integrar directamente el el codigo para hacer laui pero se terminó optando manejar la interfaz mediante consola como se puede ver el las pruebas por último aunque este proyecto fue un gran reto se puede lograr de buena manera aunque hubo algunos que cosas que todavía falta por aprender


Casa abierta al tiempo


Casa abierta al tiempo
UNIVERSIDAD AUTÓNOMA
METROPOLITANA
Unidad Cuajimalpa