Informe del Servidor FTP Distribuido

Kevin Márquez Vega Jose Miguel Leyva de La Cruz

November 26, 2025

Descripción básica del proyecto

El proyecto consiste en el desarrollo de un **servidor FTP distribuido** que cumpla con las especificaciones del **RFC959**. La arquitectura distribuida busca garantizar que el sistema continúe funcionando aun cuando algunos nodos o enlaces de comunicación fallen.

El sistema se organiza en **nodos especializados**, cada uno con un rol específico, que colaboran para procesar solicitudes FTP, almacenar archivos, validar usuarios y mantener sesiones de cliente.

Requerimientos funcionales

1. **Servidor FTP:** Implementar comandos y comportamientos del RFC959.
2. **Arquitectura distribuida:** Compuesta por nodos con roles específicos que interactúan entre sí y coordinan las operaciones del sistema.
3. **Conexiones con clientes:** El sistema debe soportar múltiples conexiones con clientes FTP.

Requerimientos no funcionales

1. **Uso de Docker y Docker Swarm:** Cada nodo se desplegará como contenedor, utilizando redes overlay de Docker Swarm. No se usarán docker-compose ni services; solo contenedores, imágenes y redes.
2. **Consistencia parcial:** El sistema mantiene consistencia parcial mientras no haya particiones, con mecanismos para reconciliación eventual tras reconexión.
3. **Roles independientes:** Cada nodo mantiene un único rol, pero múltiples nodos pueden coexistir en el mismo host.
4. **Persistencia de datos:** Ningún nodo de almacenamiento debe ser un único punto de fallo; los datos no se pierden si un nodo falla.
5. **Reconexión tras partición:** Si el sistema se divide en particiones, cada subsistema debe seguir funcionando y unificarse al restablecerse la comunicación.
6. **Tolerancia a fallos 2:** El sistema debe seguir funcionando aunque fallen al menos 2 nodos cualesquiera.

Arquitectura

- **Arquitectura Física:** P2P No Estructurado : Topología de red descentralizada donde todos los nodos se comunican directamente.
- **Arquitectura Lógica:** Microservicios Especializados
  - División funcional por responsabilidades específicas.
  - Varios tipos de nodos con roles bien definidos y únicos.
  - Colaboración entre servicios para completar las operaciones.

Roles del Sistema

- **Routing Nodes:** Gestionan **conexiones** con clientes FTP y mantienen **estados de sesión**.
- **Processing Nodes: Parsean** y **procesan** comandos FTP (stateless).
- **Data Nodes:** Gestionan operaciones **CRUD** de archivos con replicación.
- **Auth Nodes:** Validan **usuarios** y **contraseñas** almacenadas en **JSON**.
- **Discovery Nodes:** Tablas para **registro** de nombres de servicios y **gestión de heartbeats**.

Distribución en Redes Docker

Entre los objetivos del sistema se encuentra lograr resistencia a particiones y tolerancia a fallos 2. Para cumplir con esto, la distribución de servicios se implementa en dos redes overlay independientes.

Cada red contiene:

- 3 réplicas de Routing Nodes
- 3 réplicas de Processing Nodes
- 3 réplicas de Data Nodes
- 3 réplicas de Auth Nodes
- 3 réplicas de Discovery Nodes

Esta configuración garantiza que, en caso de desconexión entre redes, cada una mantenga operatividad completa y tolerancia a fallos de hasta 2 nodos de cualquier tipo.

Tipos de Procesos dentro del Sistema

- **Routing Process:**
  - Servicio de conexiones FTP en puerto 21.
  - Gestor de sesiones de usuario.
- **Processing Process:**
  - Parseo de mensajes a comandos FTP.
  - Procesador de comandos FTP.
- **Data Process:** Operaciones CRUD sobre el sistema de archivos.
- **Auth Process:** Servicio de validación de credenciales.
- **Discovery Process:**
  - Servicio de registro y descubrimiento de nodos..
  - Gestor de heartbeats

Organización de Procesos

Un proceso por instancia de nodo

- Cada contenedor ejecuta exactamente un tipo de proceso.
- No hay mezcla de roles por instancia.

Cada nodo ejecuta exclusivamente los procesos correspondientes a su tipo:

- **Routing Nodes** → Routing Process
- **Processing Nodes** → Processing Process
- **Data Nodes** → Data Process
- **Auth Nodes** → Auth Process
- **Discovery Nodes** → Discovery Process

Esta separación garantiza la especialización funcional y facilita el despliegue escalable.

Patrón de Diseño de Desempeño

Modelo Híbrido Asíncrono con Hilos

- **I/O Asíncrono:** Para operaciones de red y disco
- **Pool de Hilos:** Para procesamiento de comandos y operaciones CPU-intensivas
- **Procesos Independientes:** Cada nodo como contenedor separado

- **Routing Nodes:**

  - **Proceso principal** acepta conexiones.
  - **Hilos worker** manejan sesiones de cada usuario.

- **Processing Nodes:**
  - **Pool de hilos** para parseo y procesamiento de comandos

- **Data/Auth Nodes:**
  - Servidores concurrentes con **pool de hilos**.
  - I/O asíncrono para disco.

Comunicación

Tipo de Comunicación

Protocolo Propio Basado en Mensajes JSON sobre Sockets TCP

- **Comunicación Directa Nodo a Nodo**
- **Mensajería Síncrona** para operaciones que requieren respuesta inmediata
- **Formato JSON Estructurado** con metadatos de mensaje

Estructura del Mensaje

```
{
  "message_id": "uuid",
  "source_node": "NODE_ID",
  "destination_node": "NODE_ID",
  "operation": "OPERATION_NAME",
  "payload": {...},
  "vector_clock": {...}
}
```

Ejemplo de mensaje:

```
{
  "message_id": "123e4567-e89b-12d3-a456-426614174000",
  "source_node": "routing_01",
  "destination_node": "processing_03",
  "operation": "PROCESS_FTP_COMMAND",
  "payload": {
    "command": "STOR",
    "filename": "archivo.txt",
    "data_chunk": "base64encoded..."
  },
  "vector_clock": {
    "routing_01": 5,
    "processing_03": 2
  }
}
```

Comunicación Cliente-Servidor

- **Protocolo FTP Estándar (RFC959)** sobre TCP
- Clientes se conectan a **Routing Nodes** en puerto 21
- Sesiones mantenidas por Routing Nodes

Comunicación Servidor-Servidor

- **Routing → Processing:** PROCESS_FTP_COMMAND
- **Processing → Data:** DATA_STORE, DATA_RETRIEVE, DATA_LIST
- **Processing → Auth:** AUTH_CHECK
- **Todos → Discovery:** REGISTER, HEARTBEAT, REQUEST_SERVICE

Comunicación entre Procesos

- **Memoria compartida** para datos de alta frecuencia
- **Llamadas al sistema** para sincronización y coordinación
- **Sockets TCP** para comunicación entre nodos en diferentes hosts

Coordinación

Sincronización de Acciones

- **Vector clocks** para ordenamiento causal de eventos
- **Heartbeats periódicos** para detección de nodos activos
- **Mecanismos de timeout** para operaciones bloqueantes

Acceso Exclusivo a Recursos

- **Lock distribuido** para operaciones críticas en data nodes
- **Mutex por archivo** en operaciones STOR/RETR
- **Secciones críticas** en actualización de metadatos

Toma de Decisiones Distribuidas

- **Consenso Raft** en clusters de data nodes y auth nodes
- **Elección de líder** para operaciones de escritura

Manejo de Condiciones de Carrera

- **Versiones de archivo** con vector clocks
- **Operaciones atómicas** en sistema de archivos
- **Verificación de estado** antes de commit de operaciones

Nombrado y Localización

Identificación de los Datos y Servicios

- **Servicios identificados por tipo:** routing, data, auth, processing, discovery
- **Cada servicio con múltiples instancias** numeradas (ej: data_01, data_02)
- **Datos identificados por rutas de archivo** y nombres de usuario

Ubicación de los Datos y Servicios

- **Routing Nodes:** Expuestos en puerto 21 para clientes FTP externos
- **Data Nodes:** Almacenan sistema de archivos completo en cada réplica
- **Auth Nodes:** Contienen base de datos de usuarios replicada
- **Processing Nodes:** Disponibles para procesamiento balanceado de comandos
- **Discovery Nodes:** Mantienen tablas de registro de servicios

Localización de los Datos y Servicios

Mecanismo Principal: DNS Interno de Docker

- Resolución automática de nombres en redes overlay
- Nombres predecibles por tipo de nodo (ej: routing_1, data_3)
- Balanceo de carga integrado en la resolución DNS

Mecanismo Alternativo: Discovery Nodes

- **Tablas de registro** que mapean nombres de servicio a listas de IPs
- **Descubrimiento por broadcast** en la red local mediante ajuste de máscara
- **Protocolo de mensajes** para registro y consulta

Operaciones Disponibles:

- **REGISTER:** Nodos se registran con su tipo y dirección IP
- **HEARTBEAT:** Mantenimiento de registros activos
- **REQUEST_SERVICE:** Consulta de nodos por tipo de servicio
- **UNREGISTER:** Eliminación ordenada de nodos

Descubrimiento de Discovery Nodes:

- **Broadcast limitado** mediante ajuste de máscara de red
- **Mensajes de descubrimiento** enviados a todas las IPs de la subred
- **Respuesta automática** de nodos discovery activos

Consistencia y Replicación

Distribución de los Datos

- **Sistema de archivos completo** replicado en cada data node
- **Base de datos de usuarios** replicada en cada auth node
- **Metadatos de sesión** distribuidos en routing nodes

Replicación

- **3 réplicas** de cada tipo de nodo por red overlay
- **Réplicas activas** para todos los data y auth nodes
- **Tolerancia a fallos 2** - sistema funciona con pérdida de hasta 2 nodos

Consistencia

- **Consistencia eventual** entre réplicas
- **Raft** para mantener consistencia parcial dentro de la partición activa
- **Vector clocks** para detección y resolución de conflictos

Confiabilidad de las Réplicas

- **Write-ahead logging** para operaciones de escritura
- **Confirmación por mayoría** en operaciones críticas
- **Mecanismos de recuperación** automática tras fallos

Tolerancia a Fallos

Respuesta a Errores

- **Réplicas redundantes** por cada tipo de nodo
- **Reconexión automática** tras fallos de red
- **Reelección de líder** en clusters Raft

Nivel de Tolerancia a Fallos

- **Tolerancia a fallos 2** - sistema funciona con pérdida de hasta 2 nodos de cualquier tipo
- **Múltiples réplicas** por servicio (3 por red overlay)

Manejo de Fallos Parciales

- **Nodos caídos temporalmente:** Reincorporación automática con sincronización
- **Nodos nuevos:** Descubrimiento automático y sincronización de datos
- **Particiones de red:** Cada subsistema opera independientemente

Mecanismos de Recuperación

- **Heartbeats** para detección de nodos caídos
- **Logs de operaciones** para recuperar estado consistente
- **Sincronización diferencial** para nodos reconectados

Seguridad

Seguridad en la Comunicación

- **Conexiones cifradas** entre nodos.
- **Autenticación básica** mediante tokens de servicio
- **Validación de mensajes** mediante checksums

Seguridad en el Diseño

- **Aislamiento por contenedores** mediante Docker
- **Redes overlay segregadas** para comunicación interna

Autorización y Autenticación

- **Auth Nodes** gestionan autenticación de usuarios FTP (USER/PASS)
- **Tokens de sesión** para comunicación interna entre nodos
- **Listas de control de acceso** básicas para operaciones de archivos
- **Contraseñas almacenadas cifradas** en base de datos de usuarios

Vulnerabilidades del Sistema

- **Autenticación básica** vulnerable a ataques de fuerza bruta
- **No hay control de rate limiting** para prevenir DoS
- **Tokens de sesión simples** sin mecanismos de revocación rápida