Reporte de Avances: Load Balancer para ChatRoom

Equipo: Desarrollo del Reverse Proxy/Load Balancer
Estado Actual: Primer Avance

Arquitectura General

El sistema utiliza un reverse proxy desarrollado en Rust, diseñado para balancear la carga entre múltiples instancias de backend (lógico y de datos) y gestionar conexiones bidireccionales entre clientes (frontend) y servidores. La arquitectura se divide en dos modos configurables:

Modo FL (Frontend <-> Lógica): Balancea solicitudes del frontend a backends lógicos.

Modo LD (Lógica <-> Datos): Comunica componentes lógicos con servidores de datos.

Componentes Clave

Configuración Dinámica:
Definida en reverse_proxy_config.json, permite especificar direcciones IP/puertos para frontends, backends, y monitoreo de heartbeats.
Soporta múltiples backends (ej: 0.0.0.0:7000 0.0.0.0:7001).

Manejo de Conexiones:
TCP para datos: Las solicitudes se redirigen mediante conexiones TCP persistentes.
UDP para heartbeats: Monitorea la disponibilidad de backends mediante mensajes "OK" enviados cada 1 segundo.

Algoritmo de Balanceo:
Round-Robin: Distribuye solicitudes secuencialmente entre backends activos. Si un backend falla (sin heartbeat por 2 segundos), se excluye del pool.

Monitoreo de Servidores:
Los backends envían heartbeats periódicos. Si un servidor no responde, se marca como is_up = false y se deja de enviarle tráfico.

Estructura de Mensajes

Heartbeats: Mensajes UDP con contenido "OK" para confirmar disponibilidad.

Datos: Mensajes TCP en texto plano (ej: mensajes del chat), redirigidos sin procesamiento adicional.

Avances Implementados

- Configuración flexible mediante JSON.
- Sistema de heartbeats para detección de fallos.
- Balanceo round-robin con exclusión automática de nodos (backends) inactivos.
- Soporte para múltiples instancias de backend.

- Integración básica con frontend (pruebas iniciales realizadas).

Próximos Pasos

- Implementar autenticación de mensajes entre componentes.
- Mejorar el manejo de mensajes largos.
- Integrar métricas de rendimiento para ajuste dinámico de carga.

Repositorio GitHub:

Código actualizado en la rama *ref/load_balancer*, incluyendo configuración, backend de prueba (Python), y scripts de simulación (test.sh).