Informe del Sistema Distribuido: Arquitectura RAFT con Sharding y WebSockets

Equipo RAFT

November 30, 2025

Abstract

Este informe describe el diseño, organización, y ejecución de un sistema distribuido basado en la arquitectura RAFT con sharding especializado, que incluye la replicación de datos, manejo de eventos en tiempo real, y el uso de WebSockets para notificaciones. Se detallan las principales funcionalidades del sistema, los procesos involucrados, y las decisiones tomadas para garantizar la consistencia, disponibilidad, y seguridad del sistema. El informe también aborda aspectos relacionados con la tolerancia a fallos completa y la replicación de datos en un entorno distribuido mediante el protocolo RAFT combinado con particionamiento inteligente.

Contents

1 Arquitectura del Sistema

El sistema distribuido se basa en una arquitectura **RAFT** que permite la replicación de datos entre varios nodos. La arquitectura se organiza en varios *shards*, cada uno con un líder y seguidores, siguiendo el principio de consenso de RAFT. El objetivo principal de la arquitectura es garantizar la consistencia de los datos en presencia de fallos, mediante la replicación de los logs de eventos.

El sistema también incluye un *layer* de notificaciones en tiempo real basado en el patrón **Pub/Sub** utilizando WebSockets. Cada usuario está suscrito a un canal de eventos de su interés, lo que permite una comunicación eficiente entre los usuarios del sistema.

1.1 Principios Fundamentales de la Arquitectura

1.1.1 RAFT para Datos de Agenda

- **Elección de líder** para consistencia fuerte en eventos
- **Replicación de logs** para no perder datos
- **Failover automático** cuando un nodo falla

1.1.2 PUB/SUB para Notificaciones

- **WebSockets** para notificaciones en tiempo real
- **Usuarios suscritos** a sus canales de interés
- **Desacoplado** del almacenamiento de datos

1.1.3 Tolerancia a Fallos Completa

El sistema está diseñado para proporcionar tolerancia a fallos completa mediante la combinación de múltiples patrones y tecnologías.

2 Organización del Sistema Distribuido

El sistema distribuido se organiza en varios componentes clave:

- **Cliente (Streamlit Frontend):** Interfaz de usuario con vistas de login, calendario, eventos, grupos e invitaciones
- **WebSocket Manager:** Administra las conexiones en tiempo real y las notificaciones push inmediatas
- **Coordinador Inteligente:** Decide a qué *shard* va cada operación y balancea la carga entre los *shards*

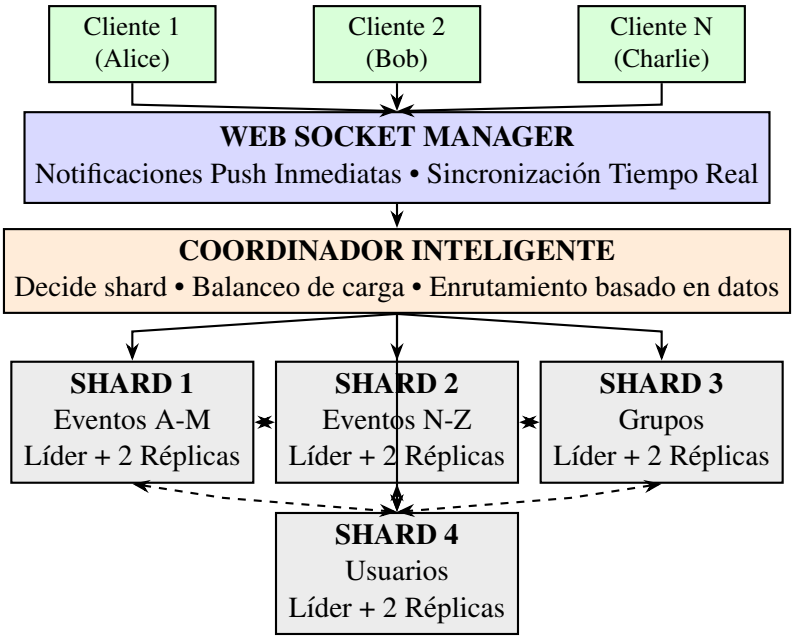

Figure 1: Arquitectura Completa del Sistema: Clientes → WebSockets → Coordinador → Shards Especializados

- **Shards Especializados:** Cada shard gestiona un subconjunto de los datos, como eventos, usuarios, y grupos. Cada shard sigue el protocolo RAFT con un líder y múltiples réplicas
- **Nodos RAFT:** Proporcionan tolerancia a fallos con almacenamiento local, réplica automática y elección de líder

2.1 Detalle de los Shards Especializados

Shard	Tipo Datos	Particionamiento	Nodos
Shard 1	Eventos	A-M	3 (Líder + 2 Réplicas)
Shard 2	Eventos	N-Z	3 (Líder + 2 Réplicas)
Shard 3	Grupos	Todos	3 (Líder + 2 Réplicas)
Shard 4	Usuarios	Todos	3 (Líder + 2 Réplicas)

Table 1: Distribución de shards especializados en el sistema

2.2 Combinación de Patrones y Tecnologías

2.2.1 Patrones Implementados

- **SHARDING (Particionado):** Data Partitioning + Horizontal Scaling

- **RAFT CONSENSUS:** Consensus Algorithm + State Machine Replication
- **SPECIALIZED LEADERSHIP:** CQRS + Service Specialization
- **PUB/SUB + WEB SOCKETS:** Publish-Subscribe + Real-Time Communication
- **INTELLIGENT ROUTING:** Router + Load Balancer

2.2.2 Esquema de Tecnología

SHARDING (Escalabilidad) + RAFT (Consistencia) →
MULTI-LEADER (Rendimiento) + SPECIALIZATION (Eficiencia) →
REAL-TIME LAYER (Experiencia) + INTELLIGENT ROUTING (Balanceo)

3 Flujo Completo de una Operación

3.1 Ejemplo: Alice crea un evento

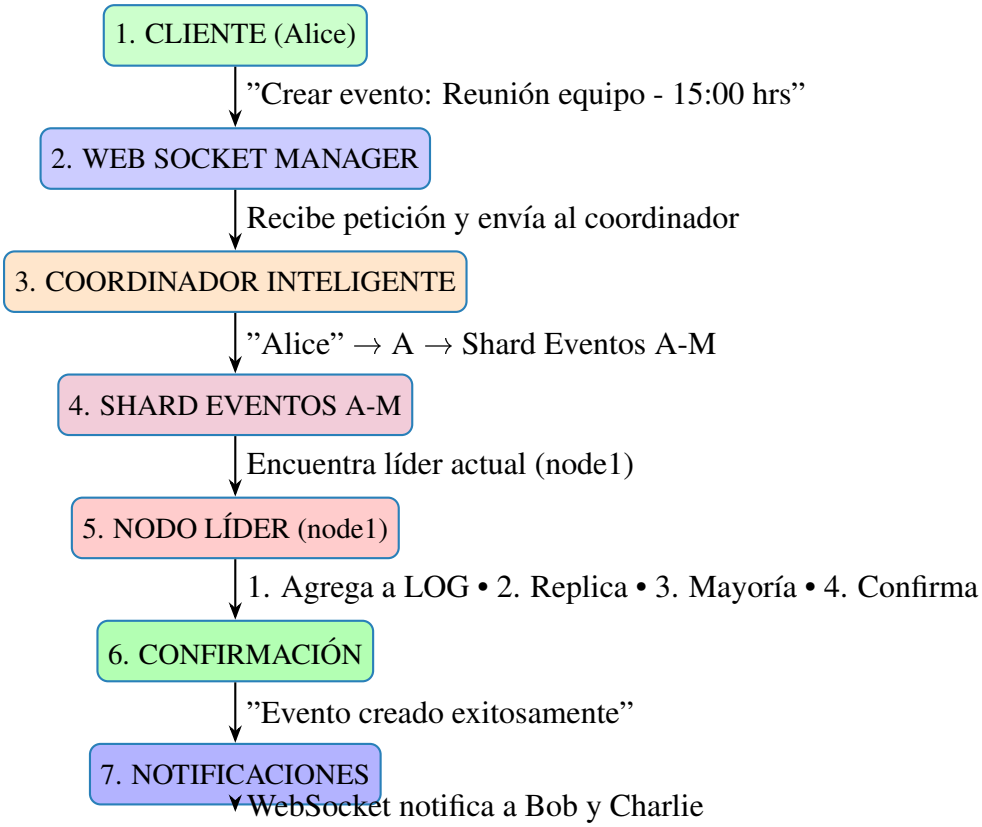

Figure 2: Flujo detallado de creación de evento por Alice

3.2 Proceso Detallado en el Nodo Líder

Cuando el nodo líder recibe una operación:

1. **Agrega operación** a su LOG local
2. **Replica** a los 2 nodos seguidores (node4, node7)
3. **Espera confirmación** de mayoría (2/3 nodos)
4. **Aplica operación** y confirma al cliente

4 Tolerancia a Fallos y Recuperación

4.1 Mecanismo de Failover Automático

4.1.1 Escenario: Fallo del Líder del Shard Eventos A-M

4.1.2 Escenario: Fallo del Líder del Shard Eventos A-M

4.1.3 Proceso de Recuperación Detallado

1. **node1 falla** (se desconecta o se apaga)
2. **Los seguidores detectan timeout** (no reciben heartbeat por 2-3 segundos)
3. **Elección automática:**
   - node4 se convierte en CANDIDATO
   - node7 se convierte en CANDIDATO
   - Piden votos entre sí
   - node4 gana la elección (mayoría)
4. **Nuevo líder:** Shard Eventos A-M: [node1: FALLIDO] [LÍDER: node4] [node7]
5. **Operaciones continúan:**
   - Las nuevas operaciones van a node4
   - node4 replica a node7
   - Cuando node1 se recupere, se sincroniza automáticamente
6. **Clientes no notan nada:**
   - El coordinador redirige automáticamente al nuevo líder
   - Las operaciones siguen funcionando normalmente

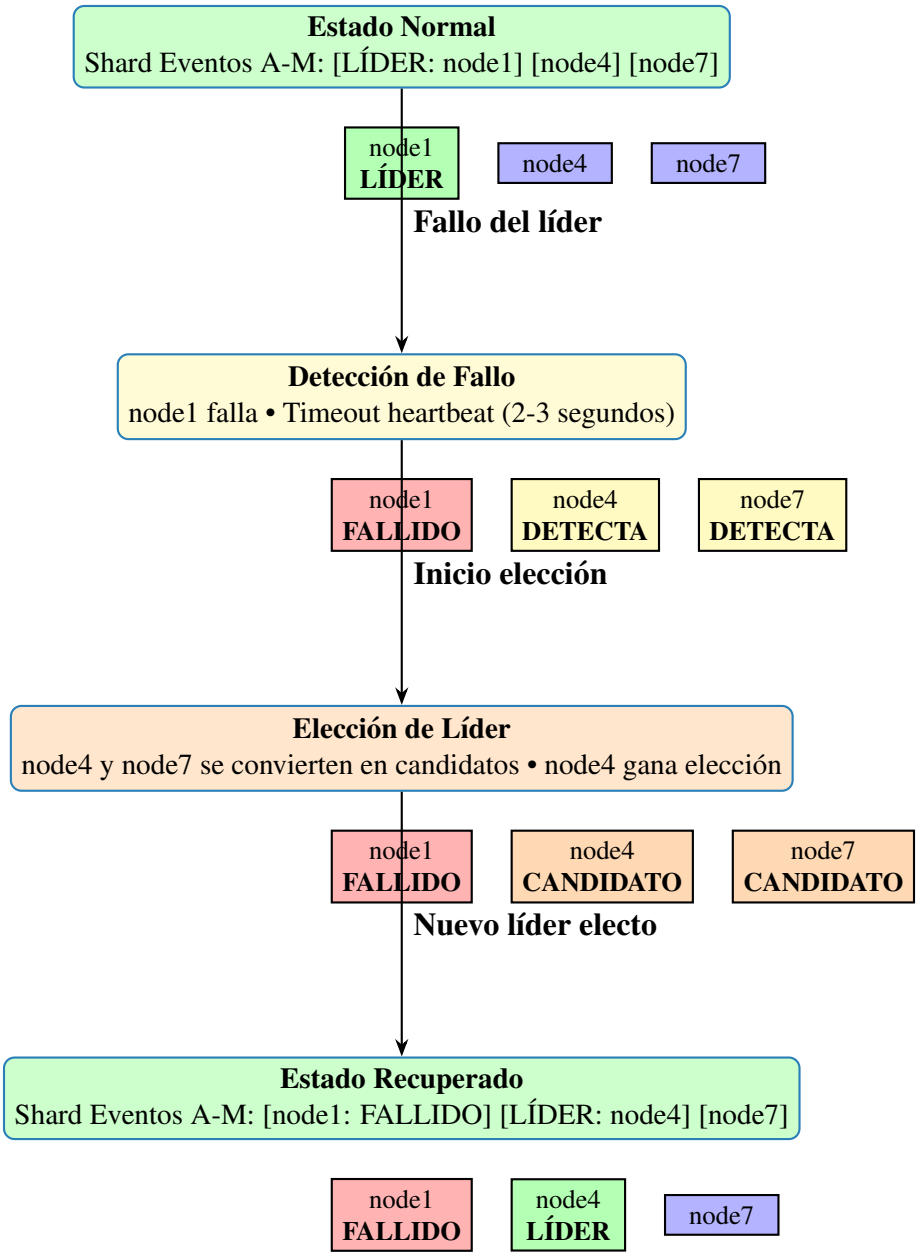

Figure 3: Proceso completo de failover automático ante fallo de líder

Usuario	Shard Destino	Ejemplo
alice	Eventos A-M	Eventos de Alice van al Shard 1
bob	Eventos N-Z	Eventos de Bob van al Shard 2
charlie	Eventos A-M	Eventos de Charlie van al Shard 1
david	Eventos N-Z	Eventos de David van al Shard 2
Grupos	Shard Grupos	Todos los grupos van al Shard 3
Usuarios	Shard Usuarios	Todos los usuarios van al Shard 4

Table 2: Distribución de datos entre shards especializados

Servidor	Nodos Alojados
Servidor 1	[Shard1-node1, Shard2-node4, Shard3-node7, Shard4-node10]
Servidor 2	[Shard1-node4, Shard2-node1, Shard3-node4, Shard4-node7]
Servidor 3	[Shard1-node7, Shard2-node7, Shard3-node1, Shard4-node4]
Servidor 4	[Shard1-node10, Shard2-node10, Shard3-node10, Shard4-node1]

Table 3: Distribución de nodos RAFT en servidores físicos para alta disponibilidad

5 Distribución de Datos y Particionamiento

5.1 Cómo se Dividen los Datos

5.2 Configuración de Nodos y Topología Física

5.2.1 Topología con 4 Servidores Físicos

6 Roles del Sistema

Cada nodo en el sistema tiene un rol específico dentro del protocolo RAFT. Los roles incluyen:

- **Líder:** El nodo que coordina la replicación de logs y garantiza la consistencia del sistema.
- **Seguidor:** Nodos que siguen al líder y replican sus logs.
- **Candidato:** Nodos que pueden ser elegidos como líderes si el líder actual falla.

7 Distribución de Servicios en Docker

El sistema distribuido está implementado en contenedores Docker. Cada grupo de RAFT se ejecuta en contenedores separados, con la comunicación entre los nodos facilitada por Docker Swarm.

7.1 Redes Docker

- **raft-network:** Red interna para comunicación entre nodos RAFT
- **api-network:** Red para comunicación frontend-backend
- **public-network:** Red para acceso externo de clientes

8 Procesos del Sistema

El sistema consta de varios procesos que trabajan en conjunto para garantizar el funcionamiento adecuado:

- **Proceso de Elección de Líder:** Los nodos RAFT eligen un líder para garantizar la consistencia de los datos.
- **Replicación de Logs:** El líder replica sus logs a los nodos seguidores.
- **Notificación en Tiempo Real:** Los clientes se suscriben a canales de eventos y reciben notificaciones a través de WebSockets.

9 Comunicación en el Sistema

El sistema utiliza varios tipos de comunicación:

- **RPC (Remote Procedure Call):** Usado en el protocolo RAFT para la replicación de logs y la elección de líder.
- **WebSockets:** Usados para la comunicación en tiempo real entre el servidor y los clientes.
- **REST API:** Usada para interactuar con la base de datos del sistema y para la creación y gestión de eventos.

9.1 Flujo de Comunicación

1. Cliente envía solicitud vía WebSocket/REST
2. Coordinador enruta al shard correspondiente
3. Líder del shard procesa y replica la operación
4. Confirmación enviada al cliente
5. Notificaciones broadcast a clientes interesados

10 Coordinación del Sistema

El sistema garantiza que todos los servicios estén sincronizados:

- **Sincronización de Acciones:** A través de RAFT, el líder asegura que las operaciones se apliquen en todos los nodos.
- **Acceso Exclusivo a Recursos:** Se asegura que solo un nodo líder pueda modificar los datos, evitando condiciones de carrera.

11 Seguridad

La seguridad del sistema se aborda en dos niveles:

- **Seguridad en la Comunicación:** Las comunicaciones entre nodos y clientes están protegidas mediante WebSockets seguros (wss://).
- **Autenticación y Autorización:** Los usuarios deben autenticar su sesión para interactuar con el sistema, y los accesos están restringidos según los roles (líder, miembro, etc.).

11.1 Medidas de Seguridad Implementadas

- Autenticación JWT para usuarios
- Encriptación TLS/SSL en todas las comunicaciones
- Validación de entrada en todos los endpoints
- Logs de auditoría para operaciones críticas