TECNOLÓGICA DEL ORIENTE
INSTITUCIÓN DE EDUCACIÓN SUPERIOR

SISTEMAS DISTRIBUTIVOS

TIPOS DE FALLOS EN SISTEMAS DISTRIBUIDOS

TIPOS DE FALLOS EN SISTEMAS DISTRIBUIDOS

En un sistema distribuido, la presencia de múltiples nodos, redes intermedias y dispositivos heterogéneos introduce una variedad de riesgos que pueden comprometer la operación global. A diferencia de los sistemas centralizados, donde un único punto de fallo puede ser fácilmente identificado, los sistemas distribuidos deben prever, detectar y manejar distintos tipos de fallos que pueden presentarse de manera independiente o combinada. La identificación clara de estas fallas es esencial para diseñar mecanismos de recuperación, consenso y tolerancia adecuados (Urbano López, 2015).

Este subtema analiza los principales tipos de fallos que afectan a los sistemas distribuidos, categorizándolos y explicando cómo impactan la fiabilidad, consistencia y disponibilidad del sistema.

1. Fallos de omisión.

Los fallos de omisión ocurren cuando un componente no envía o no recibe un mensaje que debía procesar. En sistemas distribuidos, estos pueden producirse tanto en los nodos como en los canales de comunicación.

Fallo de omisión del emisor.

Un proceso falla al no enviar un mensaje que se esperaba.

Fallo de omisión del receptor.

Un proceso no logra recibir un mensaje válido que sí fue enviado.

Ejemplo práctico. Un servidor de base de datos distribuidas no responde a una solicitud de lectura enviada por otro nodo, generando una inconsistencia temporal.

2. Fallos de tiempo (timing)

En entornos asincrónicos, los fallos de tiempo surgen cuando un nodo responde fuera del plazo esperado, incluso si su resultado final es correcto (Urbano López, 2015). Este tipo de fallo es especialmente relevante en algoritmos de consenso, donde la caducidad de una respuesta puede influir en la elección de un líder o en la validación de una réplica.

Ejemplo práctico. En una elección de líder, un nodo lento envía su voto después de que ya se ha elegido otro, causando incertidumbre si no hay mecanismos para ignorar respuestas tardías.

3. Fallos de estado (state failures)

Estos fallos implican que un componente pierde, corrompe o altera su estado interno, ya sea por error lógico o por problemas en el almacenamiento. En sistemas distribuidos, donde se espera que las réplicas mantengan estados coherentes, un fallo de estado puede propagarse rápidamente si no se detecta (Urbano López, 2015).

Ejemplo práctico. Una réplica de base de datos que presenta un valor desactualizado debido a una escritura mal aplicada o no replicada correctamente.

4. Fallos por parada (crash failures)

Este tipo de fallo ocurre cuando un proceso se detiene inesperadamente y deja de responder. Aunque no es malicioso ni errático, su presencia requiere que otros nodos detecten su ausencia y reconfiguren el sistema para continuar operando.

> **Ejemplo práctico.** Un nodo que actúa como coordinador principal se apaga por fallo eléctrico. Los demás nodos deben entonces ejecutar un proceso de reelección.

5. Fallos bizantinos

Considerados los más complejos, los fallos bizantinos se producen cuando un nodo se comporta de manera arbitraria o maliciosa, enviando mensajes contradictorios a diferentes partes del sistema. Estos fallos son especialmente críticos en redes abiertas o en entornos donde los nodos no son completamente confiables.

> **Ejemplo práctico.** Un nodo manipulado por un atacante envía datos falsos a unas réplicas y correctos a otras, provocando decisiones inconsistentes en el sistema.

Tabla 1. Clasificación general de fallos

Tipo de fallo	Descripción	Detectable fácilmente	Tolerancia frecuente
Omisión.	Mensaje perdido o no procesado.	Moderada.	Alta.
Tiempo.	Respuesta fuera de tiempo.	Difícil sin relojes.	Media.
Estado.	Estado incorrecto o corrompido.	Difícil sin verificación.	Media.
Parada (crash).	El proceso deja de funcionar.	Alta.	Alta.
Bizantino.	Comportamiento malicioso o errático.	Muy baja.	Baja (requiere PBFT).

Implicaciones de diseño

Conocer los tipos de fallos permite construir sistemas resilientes, mediante técnicas como:

- Heartbeats y timeouts para detectar fallos de omisión y parada.
- Protocolos de replicación fuerte para proteger el estado.
- Algoritmos tolerantes a fallos bizantinos (como PBFT) en entornos adversos.
- Logs transaccionales y versionamiento para recuperar estados corruptos.

Figura 1. Tipos de fallos