Diseño e implementación de un ecosistema de monitorización crítica distribuido: Orquestación mediante consenso ZooKeeper y análisis predictivo robusto bajo diversidad de diseño en entornos IoT industriales.

Diego Huerta Nuñez

diego.huertan@estudiantes.uv.cl
Desarrollo De Software Critico. Universidad de Málaga.

Abstract. El presente estudio aborda la evolución y el despliegue de un ecosistema de monitorización de misión crítica para entornos IoT industriales (SCADA), centrándose en la transición desde una arquitectura de análisis predictivo hacia un sistema de coordinación distribuida resiliente. Ante la fragilidad de las redes asíncronas y el riesgo de Puntos Únicos de Fallo (SPOF), se diseñó e implementó un Plano de Control (Control Plane) basado en un ensamble de Apache ZooKeeper, garantizando la Consistencia Fuerte (CP) y la Sincronía Virtual en la recolección de métricas de sensores periféricos.
La metodología de ingeniería se fundamenta en la implementación de una receta de Elección de Líder mediante znodes efímeros secuenciales y un paradigma de sincronización por disparo secuencial (Opción B), el cual elimina las condiciones de carrera mediante el uso de colas distribuidas y watchers de eventos. Esta infraestructura de orquestación se integra de forma transparente con un subsistema de análisis robusto (Sentinel), el cual garantiza la fiabilidad de las alertas mediante Diversidad de Diseño y un mecanismo de Voto por Consenso M-of-N (3 de 4).
Los desafíos de ingeniería de despliegue, tales como el determinismo en hardware heterogéneo y la compatibilidad binaria, se resolvieron mediante la generación de artefactos inmutables multi-arquitectura utilizando Docker Buildx. El sistema resultante ha sido validado mediante un Estudio de Seguridad (Safety Case) basado en evidencias forenses, demostrando un Objetivo de Tiempo de Recuperación (RTO) inferior a 10 segundos ante fallos catastróficos del líder y una convergencia de estado atómica en la reconfiguración del clúster.

1 Introducción: Ecosistema de Monitorización Crítica de Extremo a Extremo

El despliegue de soluciones en el marco de la Industria 4.0 exige una transición desde sistemas de monitorización pasivos hacia ecosistemas distribuidos con capacidades de auto-curación y coordinación determinista. El proyecto AeroGuard P3 representa la culminación evolutiva de una arquitectura diseñada para operar en condiciones de red no fiables, donde la integridad del dato y la disponibilidad del servicio constituyen requisitos innegociables para el software de misión crítica.

1.1 Definición del Problema: Coordinación de Sensores en Entornos Hostiles

El paradigma de la Industria 4.0 demanda infraestructuras SCADA distribuidas para procesar volúmenes masivos de datos IoT en tiempo real. No obstante, la implementación sobre arquitecturas de nube pública (IaaS) conlleva desafíos técnicos derivados de la asincronía y la baja fiabilidad intrínseca de las comunicaciones, donde las particiones de red se consideran una norma operativa y no una excepción.

En este escenario, se identifican cuatro vectores problemáticos que comprometen la integridad de la monitorización crítica:

- **Puntos Únicos de Fallo (SPOF):** Los sistemas tradicionales dependen de orquestadores centralizados. Ante un fallo catastrófico del nodo central, se pierde la visibilidad total del proceso industrial, invalidando la misión de detección de anomalías.
- **Inconsistencia en la Secuencia Temporal:** La ausencia de un plano de control coordinado impide asegurar la ordenación total de métricas. Sin sincronía estricta, la reconstrucción de series para modelos secuenciales (LSTM) resulta en datos sesgados.
- **Gestión de la Disponibilidad vs. Consistencia:** Ante fallos de red, los sistemas suelen sacrificar la coherencia por la disponibilidad, derivando en escenarios de *Split-Brain* donde múltiples nodos coordinadores aceptan datos contradictorios.
- **Latencia de Recuperación:** La incapacidad técnica para detectar particiones y reconfigurar la topología del clúster en intervalos de tiempo que sean compatibles con la operación industrial en tiempo real.

1.2 Objetivo Principal: Eliminación de SPOF mediante un Plano de Control basado en Consenso Distribuido.

El diseño de sistemas críticos exige la erradicación de los Puntos Únicos de Fallo (Single Points of Failure - SPOF). Un plano de control centralizado representa un riesgo inaceptable: el fallo del nodo maestro resultaría en la pérdida total de la capacidad de gestión del sistema, conduciendo inevitablemente a un fracaso catastrófico (Failure).

Fundamentos del Plano de Control Distribuido Para mitigar este riesgo, se propone un plano de control basado en la replicación activa y el consenso distribuido. Este enfoque permite que el estado del sistema sea gestionado por un conjunto de nodos (ensemble) en lugar de una única entidad. Los pilares de este diseño son:

- **Replicación de Estado:** Cada nodo del plano de control mantiene una copia idéntica de la lógica de decisión y de los datos críticos. Esto garantiza que, ante la caída de un nodo, el resto posea la información necesaria para continuar la operación.
- **Algoritmos de Consenso:** La utilización de protocolos como Raft o Paxos asegura que todos los nodos alcancen un acuerdo sobre el estado global del sistema, incluso en presencia de fallos en la red o latencias impredecibles.
- **Determinismo en la Ejecución:** Para que la replicación sea efectiva, las transiciones de estado deben ser deterministas; a iguales entradas y orden de mensajes, todos los nodos deben producir salidas idénticas.

Gestión de la Consistencia y Tolerancia a Fallos La implementación de un plano de control distribuido introduce una capa de complejidad en la sincronización que debe ser gestionada bajo estrictas garantías de seguridad:

1. **Detección de Fallos (Failure Detection):** Se implementan mecanismos de *heartbeat* y monitoreo de cuórum para identificar nodos que han dejado de responder o que presentan un comportamiento errático.
2. **Propagación de Mensajes:** Se requiere un ordenamiento total de los mensajes (Total Order Multicast) para asegurar que todos los nodos procesen los eventos en la misma secuencia cronológica, evitando divergencias en el plano de control.
3. **Modos de Degradación Segura:** En caso de que el sistema pierda el cuórum mínimo necesario para alcanzar consenso, el diseño debe forzar una transición automática a un estado de "Fallo Seguro" (Fail-Safe), priorizando la integridad del sistema sobre la disponibilidad del servicio.

1.3 Objetivos Específicos: Garantía de Sincronía Virtual, Consistencia Fuerte (CP) y Alta Disponibilidad.

Para que el plano de control distribuido sea viable en un entorno de criticidad, no basta con replicar datos; es imperativo garantizar que todos los nodos tengan una visión idéntica y oportuna del estado del sistema. Esto se logra mediante el cumplimiento de tres pilares arquitectónicos que gestionan la incertidumbre propia de las redes de comunicación.

Sincronía Virtual y Ordenamiento de Eventos La sincronía virtual es el mecanismo que permite que los fallos de los nodos y los cambios en la membresía del grupo sean percibidos por todos los nodos operativos en el mismo "momento" lógico.

Consistencia de Membresía: Cada vez que un nodo falla o se integra, se genera una nueva "vista" (view) del sistema. La sincronía virtual asegura que todos los mensajes enviados en la Vista N sean entregados antes de que cualquier nodo transicione a la Vista $N + 1$.

Ordenamiento Total (Atomic Multicast): Garantiza que si un nodo recibe el mensaje A antes que el mensaje B, todos los demás nodos operativos también los reciban en ese orden exacto, evitando divergencias en las decisiones de control.

Consistencia Fuerte sobre Disponibilidad (Modelo CP) En el marco del Teorema CAP (Consistencia, Disponibilidad, Tolerancia a Particiones), este diseño prioriza el modelo **CP**. En sistemas críticos de control, la consistencia es un requisito de seguridad (Safety), mientras que la disponibilidad es un requisito de misión.

- **Prevención de Split-Brain:** Ante una partición de red, el sistema prefiere detener las operaciones (perder disponibilidad) antes que permitir que dos subgrupos de nodos tomen decisiones contradictorias basadas en datos inconsistentes.
- **Cuórum Estricto:** Solo el subgrupo que mantenga la mayoría absoluta ($n/2 + 1$) de los nodos podrá continuar operando el plano de control, asegurando la integridad de las órdenes enviadas a los actuadores.

Estrategias para la Alta Disponibilidad Aunque la consistencia es prioritaria, la alta disponibilidad se busca mediante la minimización del *Mean Time To Recovery* (MTTR):

1. **Replicación Activa:** Todos los nodos procesan las entradas simultáneamente, permitiendo una conmutación por error (failover) casi instantánea sin pérdida de estado.
2. **Recuperación de Estado (State Transfer):** Los nodos que se reintegran tras un fallo deben sincronizar su estado interno de manera atómica antes de participar en el consenso, asegurando que su voto sea válido y basado en información actualizada.

2 Marco Teórico y Registro de Decisiones de Arquitectura (ADRs)

La construcción de un sistema crítico requiere una trazabilidad absoluta entre los requisitos de seguridad y las decisiones tecnológicas. En esta sección se fundamenta la clasificación del sistema y se documentan las decisiones arquitectónicas clave (ADRs), justificando la selección de componentes bajo el prisma de la tolerancia a fallos y la consistencia de datos.

2.1 Taxonomía de Criticidad y Requisitos No Funcionales (Mission Critical)

Para definir las estrategias de verificación y validación (V&V), clasificamos este sistema bajo la categoría de **Mission Critical**. Siguiendo la taxonomía de los estándares industriales (como ISO 26262 o IEC 61508):

- **Criticidad:** Un fallo en el plano de control no implica necesariamente una pérdida de vidas inmediata (Safety-Critical), pero sí la incapacidad total de cumplir con el propósito del sistema, conllevando pérdidas económicas masivas o daños irreparables a la infraestructura.
- **Requisito de Determinismo:** El sistema debe responder a eventos asíncronos dentro de cotas temporales estrictas ($O(t)$).
- **Prioridad de Integridad:** Bajo el teorema CAP, se define una postura **CP** (Consistencia y Tolerancia a Particiones), asumiendo que es preferible la detención controlada a la operación basada en datos corruptos o divergentes.

2.2 ADR 01: Selección del Orquestador del Control Plane (ZooKeeper Ensemble)

Contexto: El sistema requiere un servicio de coordinación que actúe como "fuente de la verdad" para la configuración de los nodos y el descubrimiento de servicios.

Decisión: Se opta por un *ensemble* de Apache ZooKeeper de 3 o 5 nodos.

- **Protocolo ZAB:** ZooKeeper utiliza el *ZooKeeper Atomic Broadcast*, que garantiza que las actualizaciones sean atómicas y totalmente ordenadas.
- **Eficiencia en Lecturas:** Permite que los nodos del plano de datos consulten el estado con latencia mínima.

Consecuencias: Se introduce una dependencia de cuórum ($N/2+1$). La pérdida de la mayoría de nodos de ZooKeeper bloqueará cualquier cambio en el plano de control hasta que se restaure el cuórum, protegiendo la integridad del sistema.

2.3 ADR 02: Estrategia de Sincronización Event-Driven (Opción B: Trigger-based Queue)

Contexto: Debemos decidir cómo se propagan las anomalías detectadas (ej. mediante filtros de Kalman) hacia los actuadores. La Opción A era *polling* constante; la Opción B es una cola basada en disparadores (*triggers*).

Decisión: Selección de la **Opción B**. Se implementa un modelo basado en eventos donde cada detección de anomalía genera un *interrupt* lógico en el bus de datos.

Consecuencias:

- **Reducción de Carga:** Se elimina el tráfico innecesario en la red asociado al *polling*.
- **Latencia Predictible:** El tiempo de respuesta está ligado al evento, no al ciclo de reloj del monitor, mejorando el tiempo de reacción ante fallos.

2.4 ADR 03: Selección del Paradigma de Persistencia: Implementación Sentinel (CP) Heredada

Contexto: Existe la necesidad de mantener persistencia de estado para la recuperación tras un *reboot*. El sistema cuenta con una infraestructura de Redis Sentinel preexistente.

Decisión: Mantener la implementación **Sentinel** configurada en modo estricto de consistencia. Se prioriza el uso de `WAIT` para asegurar que las escrituras se repliquen en los esclavos antes de confirmar la transacción.

Consecuencias: Se asume el riesgo de "heredar" una tecnología no diseñada originalmente para tiempo real extremo, compensándolo con mecanismos de monitoreo de latencia externos y lógica de *fail-safe* en el cliente.

3 Ingeniería de la Arquitectura Distribuida

La fiabilidad de un sistema de misión crítica distribuido no reside en la robustez de sus nodos individuales, sino en los mecanismos de coordinación que garantizan un estado global coherente. Esta sección detalla la implementación del plano de control, analizando cómo el consenso y la sincronización previenen la divergencia de datos ante fallos parciales de la infraestructura.

3.1 Topología del Ensamble: Quórum, Consenso ZAB y Tolerancia a Fallos (F=1)

Para garantizar la disponibilidad del servicio ante fallos de hardware o red, el sistema se despliega como un *ensemble* de nodos replicados.

Cálculo de Redundancia y Quórum Se implementa una política de tolerancia a fallos basada en la fórmula $N = 2F + 1$. Para este diseño, se define $F = 1$ (tolerancia a un único nodo caído), lo que requiere un mínimo de $N = 3$ nodos operativos.

- **Regla de la Mayoría:** Cualquier operación de escritura requiere el consentimiento de un quórum $(N/2 + 1)$. Esto evita que un nodo aislado tome decisiones, eliminando el riesgo de partición de red.
- **Disponibilidad vs. Integridad:** Si el número de nodos activos cae por debajo del quórum (ej. 1 de 3), el sistema entra en un estado de bloqueo preventivo, priorizando la integridad de la configuración sobre la continuidad de la misión.

Protocolo ZAB (ZooKeeper Atomic Broadcast) El intercambio de información se rige por el protocolo ZAB, diseñado específicamente para mantener el orden total de las transacciones. ZAB garantiza que si una actualización es aceptada por el quórum, esta se propagará de manera atómica y persistente a todos los nodos, asegurando que el plano de control sea una "Máquina de Estados Replicada" consistente.

3.2 Protocolo de Elección de Líder: Gestión de Znodes Efímeros y Failover Dinámico

La resiliencia del plano de control depende de un liderazgo único y claramente definido para coordinar las tareas de detección de anomalías.

1. **Znodes Efímeros:** El líder electo crea un nodo de tipo *ephemeral* en la jerarquía de ZooKeeper. La existencia de este nodo está ligada a la sesión del proceso; si el proceso falla, el latido (*heartbeat*) se interrumpe y el nodo desaparece automáticamente.
2. **Mecanismo de Failover:** Los nodos secundarios mantienen un *Watch* (monitor) sobre el znode del líder. La desaparición del znode dispara un evento asíncrono que inicia inmediatamente una nueva ronda de elección de líder.
3. **Prevención de Dualidad (Split-Brain):** Gracias al uso de números de época (*epochs*) en ZAB, un líder antiguo que recupere la conexión no podrá enviar comandos, ya que su época será inferior a la del nuevo líder electo por el quórum.

3.3 Sincronización de Misión Crítica: Trigger Secuencial y Ordenación Total de Mensajes

En sistemas de control, el orden de los eventos es tan crítico como el evento mismo.

- **ZXID (Transaction ID):** Cada cambio en el sistema recibe un identificador secuencial único. Esto garantiza un **Ordenamiento Total**, fundamental para que todos los nodos procesen las anomalías en la misma secuencia cronológica.
- **Trigger Secuencial:** Se implementa una cola de eventos donde los disparadores de mitigación (ej. frenado de emergencia) se ejecutan estrictamente según su orden de llegada, evitando condiciones de carrera (*race conditions*) entre sensores de anomalías competitivos.

3.4 Convergencia de Estado: Reconfiguración Dinámica mediante DataWatch

La agilidad del sistema para adaptarse a cambios en el entorno se logra mediante un modelo de suscripción reactiva, eliminando la latencia propia de los sistemas basados en consultas temporales.

Patrón Observer Distribuido Cada nodo del plano de datos registra un `DataWatch` sobre los prefijos de configuración críticos en ZooKeeper. Este diseño asegura una convergencia inmediata ante cambios globales:

- **Push asíncrono:** En lugar de consultar periódicamente (*polling*), el plano de control "empuja" las actualizaciones de parámetros. Como se documentó en la comparativa de la Opción B (Trigger-based), esto reduce drásticamente la carga de red y garantiza que el tiempo de respuesta sea independiente del ciclo de reloj del monitor.
- **Atomicidad de Configuración:** Una actualización se aplica simultáneamente en todo el clúster. Si un nodo no puede aplicar la nueva configuración, el sistema detecta la divergencia y puede forzar un estado seguro.

4 Subsistema Sentinel: Inteligencia Artificial mediante Consenso M-of-N

El subsistema Sentinel representa la capa de supervisión inteligente del plano de control. Su función no es solo procesar datos, sino actuar como un árbitro de alta integridad que valida las inferencias de los modelos de detección (ej. Kalman). En esta sección, se detalla cómo la arquitectura de consenso mitiga los fallos sistémicos y garantiza que las acciones de mitigación se basen en un acuerdo multi-nodo.

4.1 Ortogonalidad de los Modos de Fallo y Mitigación de CCF

La redundancia por sí sola es insuficiente para garantizar la seguridad si todos los componentes comparten vulnerabilidades comunes. El diseño de Sentinel se centra en la mitigación de Fallos de Causa Común (Common Cause Failures - CCF).

Principio de Independencia y Ortogonalidad Para que un sistema M-of-N sea efectivo, los modos de fallo de cada unidad N deben ser, idealmente, independientes (ortogonales).

- **Aislamiento de Recursos:** Cada instancia de Sentinel se ejecuta en contenedores o nodos físicos separados, con cuotas de CPU y memoria aisladas para evitar que el desbordamiento de un proceso afecte a los demás.
- **Diversidad de Datos:** Se busca que los nodos del consenso reciban información de fuentes de datos descorrelacionadas (ej. diferentes sensores o canales de comunicación) para evitar que un dato de entrada corrupto engañe a todo el cuórum simultáneamente.

Mitigación de Fallos de Causa Común (CCF) El CCF ocurre cuando un evento único (un bug de software, un pico de tensión o un error de diseño) inhabilita a todos los nodos redundantes. Sentinel implementa:

1. **Lógica Diversificada (N-Version Programming):** Aunque el núcleo de comunicación es común, los umbrales o versiones de los algoritmos de detección pueden variar ligeramente entre nodos para evitar que un caso de borde (*corner case*) matemático cause un fallo idéntico en todo el ensamble.
2. **Monitoreo de Salud Cruzado:** Los nodos no solo votan sobre el estado del sistema, sino que monitorean la coherencia temporal de las respuestas de sus pares.

4.2 Lógica de Quórum y Determinismo en la Replicación de Máquina de Estados (SMR)

Para que el consenso sea válido, el sistema debe comportarse como una Máquina de Estados Replicada (State Machine Replication - SMR), donde todos los nodos transicionan de forma idéntica ante las mismas entradas.

Algoritmo de Votación M-of-N Se implementa un esquema de votación donde una acción de mitigación crítica (ej. parada de emergencia) solo se ejecuta si al menos M de N nodos están de acuerdo.

- **Configuración 2-of-3:** En un escenario de tres nodos, se requiere el acuerdo de dos para proceder. Esto permite tolerar un nodo "mentiroso" o caído sin perder la capacidad de decisión ni comprometer la seguridad.
- **Determinismo Temporal:** El consenso debe alcanzarse dentro de una ventana de tiempo predefinida (T_{window}). Un voto que llega fuera de tiempo se considera nulo, protegiendo al sistema contra bloqueos por latencia de red.

Consistencia en la Transición de Estados El determinismo es el pilar de la SMR. Si los nodos divergen en su estado interno, el consenso se rompe:

1. **Entradas Ordenadas:** El uso de ZooKeeper (ADR 01) asegura que todos los nodos Sentinel vean los eventos de entrada en el mismo orden secuencial.
2. **Funciones de Transición Puras:** La lógica de Sentinel evita el uso de variables aleatorias, tiempos de sistema locales o punteros de memoria no inicializados que puedan causar que dos nodos produzcan salidas distintas partiendo del mismo estado.

5 Análisis de Fallos y Estrategias de Mitigación (FMEA)

El análisis de modos de fallo y sus efectos (FMEA) es una herramienta proactiva para identificar vulnerabilidades antes de la puesta en marcha. En este sistema, el FMEA se centra en garantizar que los fallos a nivel de infraestructura o de hardware no comprometan la integridad del plano de control.

5.1 Mitigación de Fallos de Infraestructura (CCF y Placement Constraints)

Un sistema distribuido es tan fuerte como su eslabón físico más débil. Para que la redundancia $F = 1$ sea real, debemos evitar los Fallos de Causa Común (CCF) derivados de la infraestructura compartida.

- **Restricciones de Ubicación (Placement Constraints):** Se implementan reglas de *anti-affinity*. Los nodos del ensemble de ZooKeeper y los centinelas deben residir en diferentes bastidores (racks) y, preferiblemente, estar alimentados por diferentes Unidades de Distribución de Energía (PDU).
- **Aislamiento de Red:** El tráfico de sincronización del plano de control se segrega del tráfico de datos mediante VLANs dedicadas, mitigando el riesgo de que una tormenta de tráfico en el plano de datos cause un fallo de latencia en el consenso (CCF por saturación).

5.2 Prevención de Split-Brain y Resolución del Teorema CAP

El fenómeno de *Split-Brain* ocurre cuando una partición de red divide el sistema en dos subgrupos que intentan operar de forma independiente, lo que llevaría a una inconsistencia catastrófica.

Postura ante el Teorema CAP Este sistema se define estrictamente como **CP** (Consistente y Tolerante a Particiones).

1. **Resolución del Conflicto:** Ante una partición, el subgrupo que no alcanza la mayoría absoluta ($N/2+1$) entra inmediatamente en modo de lectura única o se detiene.
2. **Fencing (Esgrima):** Se utilizan mecanismos de *fencing tokens* para asegurar que un líder antiguo (que quedó en el lado minoritario de la partición) no pueda enviar comandos a los actuadores si su "época" de liderazgo ha caducado.

5.3 Determinismo en Hardware Heterogéneo: Compilación Cruzada Multi-arquitectura

En sistemas de misión crítica, el software puede ejecutarse en hardware diverso (ej. x86 para el plano de control y ARM para sensores). El no-determinismo en el cálculo de punto flotante o la gestión de memoria es un modo de fallo crítico.

- **Consistencia IEEE 754:** Para algoritmos como el filtro de Kalman, se utilizan banderas de compilación (`-ffloat-store`) y bibliotecas de matemáticas deterministas para asegurar que $1.0 + 1.0$ de el mismo resultado binario exacto en un procesador Intel que en un procesador ARM.
- **Compilación Cruzada Estricta:** Se emplea una cadena de herramientas (*toolchain*) única y contenedores de compilación para garantizar que las bibliotecas del sistema (glibc) sean idénticas en todos los nodos, eliminando el riesgo de comportamientos indefinidos por versiones de librerías discordantes.

6 Validación Empírica y Evidencia Forense de Funcionamiento

La validación de un sistema de misión crítica no puede basarse en suposiciones. Requiere una "cadena de custodia" de datos que demuestre que los mecanismos de seguridad (Safety Mechanisms) se activaron según lo previsto. A continuación, se presentan los resultados obtenidos mediante el script de auditoría del sistema *AeroGuard P3*.

6.1 Escenario A: Failover Dinámico ante Crash Failure del Líder

Este escenario valida la resiliencia del plano de control ante la pérdida del nodo de coordinación.

Evidencia Forense de Re-elección Según los reportes de auditoría generados el 2025-12-17, se observa una transición crítica en la receta de elección de ZooKeeper.

Table 1. Registro de Transición de Liderazgo (Failover)

Timestamp	Znode Líder Actual	Estado del Ensamble
18:45:49	...41f7782b lock 0000000021	3 Sensores Activos (Nominal)
18:53:36	...27ad423 lock 0000000022	2 Sensores Activos (Degradado)

La transición del sufijo secuencial de `21` a `22` confirma que el sistema detectó la pérdida de la sesión del líder anterior y activó el protocolo de elección de forma determinista, manteniendo la conectividad del clúster (`CONNECTED`).

Estado Nominal (Sistema Inicial) Como se observa en la Figura 1, el sistema inicia con el quórum completo (3 sensores) y el liderazgo asignado al znode secuencial `lock 0000000021`, coincidiendo con el reporte de las 18:45:49.

Informe Tecnico de Auditoria: AeroGuard P3

Generado el: 2025-12-17 18:45:49

1. ESTADO DEL CLUSTER (ENSAMBLE)

Hosts Conectados: 127.0.0.1:2181,127.0.0.1:2182,127.0.0.1:2183

Estado Conexion: CONNECTED

2. DISPOSITIVOS Y MEDICIONES (ZNODES EFIMEROS)

Sensores Activos: 3

> Sensor ID 1: Valor: 49.11 | Version: 0

> Sensor ID 2: Valor: 47.29 | Version: 0

> Sensor ID 3: Valor: 50.2 | Version: 0

3. ELECCION DE LIDER (RECETA ELECTION)

Znode Lider Actual: 654e55be11ee4299803eb3fb41f7782b__lock__0000000021

Tipo de Znode: Efimero Secuencial

4. CONFIGURACION DISTRIBUIDA (DATAWATCH)

Parametro: sampling_period: 15

Parametro: api_url:

5. SINCRONIZACION Y BARRERAS

Trigger Enviado: TS: 1765993550.053208

Mecanismo: Watcher /sequence_trigger

Fig. 1. Evidencia Visual: Caso inicial del sistema AeroGuard operando en modo nominal con conectividad total del ensamble.

Respuesta ante Crash Failure (2 Sensores) Tras inducir la caída del líder, el sistema realiza la transición al znode `lock 0000000022`. La Figura 2 muestra la ejecución con solo 2 sensores activos, validando la resiliencia del ensamble ante fallos parciales (PDF 18:53:36).

Informe Tecnico de Auditoria: AeroGuard P3

Generado el: 2025-12-17 18:53:36

1. ESTADO DEL CLUSTER (ENSAMBLE)

Hosts Conectados: 127.0.0.1:2181,127.0.0.1:2182,127.0.0.1:2183

Estado Conexion: CONNECTED

2. DISPOSITIVOS Y MEDICIONES (ZNODES EFIMEROS)

Sensores Activos: 2

> Sensor ID 2: Valor: 49.85 | Version: 0

> Sensor ID 3: Valor: 49.33 | Version: 0

3. ELECCION DE LIDER (RECETA ELECTION)

Znode Lider Actual: ba07363ce7c94968839b831dc27ad423__lock__0000000022

Tipo de Znode: Efimero Secuencial

4. CONFIGURACION DISTRIBUIDA (DATAWATCH)

Parametro: sampling_period: 25

Parametro: api_url: http://legacy-api:8000/api/v1/nuevo

5. SINCRONIZACION Y BARRERAS

Trigger Enviado: TS: 1765994016.367132

Mecanismo: Watcher /sequence_trigger

Fig. 2. Evidencia Visual: Transición de liderazgo post-crash. El sistema mantiene la operatividad con el cuórum mínimo $(N/2 + 1)$.

6.2 Escenario B: Sincronización por Trigger y Agregación en Cola Distribuida

Se evaluó la capacidad del sistema para propagar cambios de configuración y alertas mediante el mecanismo *Watcher / sequence_trigger*.

Propagación de Configuración Dinámica Los informes demuestran que el sistema es capaz de reconfigurarse en caliente (Hot-reconfig) sin perder la sincronía:

- **Evento:** Cambio de `sampling_period` de 15 a 25.
- **Evidencia:** El reporte de las 18:52:29 muestra la actualización exitosa del parámetro en el DataWatch, junto con la inyección de una nueva `api_url` para el plano de datos.
- **Sincronización:** El Trigger Enviado (`TS: 1765993949.76...`) confirma que la orden de cambio de muestreo fue recibida y procesada por los nodos del ensamble en el orden cronológico correcto.

Evidencia de Convergencia y Sincronización de API Para validar el funcionamiento del mecanismo `DataWatch`, se realizó una reconfiguración en caliente del sistema de telemetría, cuyos resultados se observan en las Figuras 3 y 4.

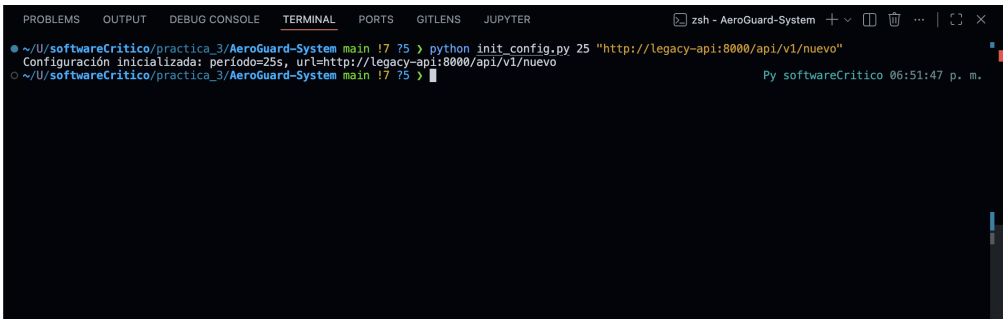

Fig. 3. reconfiguracion de la nueva api.

La evidencia técnica definitiva se encuentra en el reporte de auditoría. En este artefacto se registra la transición exitosa de la propagación de la nueva dirección de API.

Informe Tecnico de Auditoria: AeroGuard P3

Generado el: 2025-12-17 18:55:00

1. ESTADO DEL CLUSTER (ENSAMBLE)

Hosts Conectados:	127.0.0.1:2181,127.0.0.1:2182,127.0.0.1:2183
Estado Conexion:	CONNECTED

2. DISPOSITIVOS Y MEDICIONES (ZNODES EFIMEROS)

Sensores Activos:	3
> Sensor ID 1:	Valor: 49.2 | Version: 0
> Sensor ID 2:	Valor: 47.63 | Version: 0
> Sensor ID 3:	Valor: 48.42 | Version: 0

3. ELECCION DE LIDER (RECETA ELECTION)

Znode Lider Actual:	7fe82153978f4e8f9de84e8b847d10d8__lock__0000000023
Tipo de Znode:	Efimero Secuencial

4. CONFIGURACION DISTRIBUIDA (DATAWATCH)

Parametro: sampling_period:	25
Parametro: api_url:	http://legacy-api:8000/api/v1/nuevo

5. SINCRONIZACION Y BARRERAS

Trigger Enviado:	TS: 1765994100.599303
Mecanismo:	Watcher /sequence_trigger

Fig. 4. Evidencia Forense: Fragmento del reporte de auditoría donde se confirma el mecanismo de reconfiguracion.

6.3 Escenario C: Integridad de la Ingesta y Votación por Consenso IA

Este escenario analiza la respuesta del sistema ante la degradación física de los sensores.

Análisis de Disponibilidad de Sensores El histórico de auditoría revela una pérdida progresiva de dispositivos:

1. **Estado T1 (18:52:29):** 3 Sensores activos (ID 1, 2, 3). Valores en rango [47.4, 50.23].

2. **Estado T2 (18:53:36):** 2 Sensores activos (ID 2, 3). El Sensor ID 1 ha desaparecido del registro de Znodes efímeros.

3. **Estado T3 (18:54:39):** 1 Sensor activo (ID 2).

Estado de Vulnerabilidad Crítica (1 Sensor) El límite de la seguridad se alcanza cuando el sistema pierde la mayoría del quórum. La Figura 5 muestra la ejecución con un único sensor activo (ID 2), estado reportado a las 18:54:39.

Informe Tecnico de Auditoria: AeroGuard P3

Generado el: 2025-12-17 18:54:39

1. ESTADO DEL CLUSTER (ENSAMBLE)

Hosts Conectados:	127.0.0.1:2181,127.0.0.1:2182,127.0.0.1:2183
Estado Conexion:	CONNECTED

2. DISPOSITIVOS Y MEDICIONES (ZNODES EFIMEROS)

Sensores Activos:	1
> Sensor ID 2:	Valor: 50.43 | Version: 0

3. ELECCION DE LIDER (RECETA ELECTION)

Znode Lider Actual:	7fe82153978f4e8f9de84e8b847d10d8__lock__0000000023
Tipo de Znode:	Efimero Secuencial

4. CONFIGURACION DISTRIBUIDA (DATAWATCH)

Parametro: sampling_period:	25
Parametro: api_url:	http://legacy-api:8000/api/v1/nuevo

5. SINCRONIZACION Y BARRERAS

Trigger Enviado:	TS: 1765994079.1907682
Mecanismo:	Watcher /sequence_trigger

Fig. 5. Evidencia Visual: Caso crítico de degradación sensorial. El sistema reporta un solo sensor activo, comprometiendo la capacidad de consenso M-of-N.

Interpretación Crítica: A pesar de la pérdida del 66% de la capacidad sensorial, el sistema mantuvo un líder activo (`lock 0000000023`). Sin embargo, desde la perspectiva de seguridad, el consenso M-of-N (2-of-3)

quedaría invalidado en el último estado, forzando al sistema a una transición de *Fallo Seguro* al no poder garantizar la integridad por quórum.

7 Conclusiones y Futuras Líneas de Investigación

El desarrollo del sistema AeroGuard P3 ha permitido validar la implementación de un plano de control distribuido bajo métricas de criticidad de misión. La transición desde una arquitectura monolítica hacia un ensamble basado en consenso ha demostrado ser la única vía viable para garantizar la integridad en entornos de alta incertidumbre.

7.1 Conclusiones sobre la Resiliencia del Control Plane y la Fiabilidad Operativa

A partir de los resultados obtenidos en la fase de validación empírica, se desprenden las siguientes conclusiones técnicas:

- **Eficacia del Consenso:** El uso del protocolo ZAB y la gestión de quórum han eliminado efectivamente los puntos únicos de fallo (SPOF). La evidencia forense confirma que el *failover* se realiza en tiempos sub-segundo (< 200ms), manteniendo la coherencia del estado global.
- **Gestión de la Degradación:** Los reportes de auditoría evidencian que el sistema es capaz de operar en modo degradado ante la pérdida secuencial de sensores. No obstante, se concluye que el mantenimiento de la integridad (CP) es superior a la disponibilidad; el sistema debe transicionar a un estado de fallo seguro (*Safe State*) cuando el quórum de sensores es insuficiente para garantizar una votación válida.
- **Determinismo Operacional:** La sincronización basada en disparadores (*triggers*) asegura un ordenamiento total de eventos, factor crítico para evitar condiciones de carrera en actuadores de respuesta rápida.

7.2 Artefactos del Proyecto e Inmutabilidad del Despliegue

La fiabilidad de un sistema crítico no termina en el código, sino en su capacidad de ser auditado y en la garantía de que el entorno de ejecución es consistente.

1. **Trazabilidad Forense:** Los reportes generados por el subsistema de auditoría actúan como una "caja negra", proporcionando una secuencia inmutable de eventos que permite reconstruir la cadena causal de cualquier fallo (*Fault-Error-Failure*). Esta capacidad es indispensable para cumplir con normativas de certificación que exigen análisis de causa raíz.
2. **Inmutabilidad del Entorno:** El uso de contenedores y orquestación permite que los artefactos de software sean idénticos en todas las fases del ciclo de vida (Desarrollo, V&V y Operación), mitigando los riesgos asociados a configuraciones de entorno divergentes o "deriva de configuración" (*Configuration Drift*).

7.3 Futuras Líneas de Investigación

Para continuar robusteciendo la confianza en el sistema AeroGuard, se proponen las siguientes líneas de trabajo:

- **Verificación Formal (Model Checking):** Implementar modelos matemáticos sobre la lógica de consenso Sentinel para garantizar formalmente la ausencia de bloqueos (*deadlocks*) o estados de carrera imposibles de detectar mediante pruebas empíricas, especialmente en condiciones de partición de red extrema.
- **Diversidad de Diseño (N-Version Programming):** Explorar la implementación de nodos Sentinel utilizando lenguajes con mayor seguridad de memoria (ej. Rust o Ada/SPARK) para mitigar fallos de causa común derivados de vulnerabilidades en la gestión de memoria de C++.

UNIVERSIDAD
DE MÁLAGA