MODELO DE UTILIDAD

Oficina Española de Patentes y Marcas (OEPM)

MEMORIA DESCRIPTIVA

Dispositivo de control autónomo multi-agente con bus local y núcleo endógeno de seguridad

(NEOSYNT)

Inventora: Carmen Esther Jiménez Mesa
Contacto: [A completar]
Fecha: December 1, 2025

Documento preparado conforme al Reglamento de ejecución de la Ley de Patentes

Contents

1 Campo de la invención **2**

2 Estado de la técnica **2**

2.1 Sistemas de control distribuido convencionales 2

2.2 Problemas técnicos identificados 2

2.3 Documentos relevantes 2

3 Problema técnico objetivo **3**

4 Descripción de la invención **3**

4.1 Visión general 3

4.2 (100) Bus local por UNIX socket 3

4.3 (110) Buffers circulares por agente 4

4.4 (120) Módulo de validación con checksum y registro inmutable 4

4.4.1 Checksum SHA-256 parcial 4

4.4.2 Registro inmutable (log) 4

4.5 (130) Núcleo autónomo 4

4.5.1 Vector de intención **I** 4

4.5.2 Actualización por mirror-descent 5

4.5.3 Ruido Ornstein-Uhlenbeck endógeno 5

4.6 (140) Gate de consentimiento bilateral 5

4.6.1 Variables de entrada 5

4.6.2 Condición de consentimiento 6

4.7 (150) Watchdog de recursos 6

4.8 (160) Sandbox de evolución de código 6

4.9 (170) Planificador con horizonte adaptativo 6

4.10 Efecto técnico de la combinación estructural 7

5 Breve descripción de los dibujos **7**

6 Realizaciones y ejemplos **7**

6.1 Configuración de pruebas 7

6.2 Métricas evaluadas 8

6.3 Resultados 8

7 Ventajas prácticas **8**

8 Aplicaciones **9**

9 Clasificación IPC sugerida **9**

1 Campo de la invención

La presente invención se refiere al campo del control autónomo multi-agente implementado por ordenador (CII), y más específicamente a dispositivos y sistemas para la coordinación de agentes autónomos en sistemas distribuidos y de computación en el borde (edge computing).

El dispositivo objeto de esta invención pertenece a la categoría de invenciones implementadas por ordenador que producen un efecto técnico adicional más allá de la mera interacción programa-hardware, concretamente: reducción medible de colapsos del sistema, disminución de latencia en la coordinación, y aumento de robustez frente a perturbaciones y ruido.

2 Estado de la técnica

Los sistemas de control distribuido multi-agente conocidos en el estado de la técnica presentan las siguientes características y limitaciones:

2.1 Sistemas de control distribuido convencionales

Los sistemas de control distribuido tradicionales emplean protocolos de comunicación centralizados o semi-centralizados que requieren:

- Hiperparámetros fijos definidos externamente (tasas de aprendizaje, umbrales de decisión, constantes de tiempo).
- Funciones de recompensa externas para guiar el comportamiento.
- Mecanismos de coordinación basados en consenso que no consideran el estado interno de los agentes.

2.2 Problemas técnicos identificados

Los sistemas conocidos adolecen de:

1. **Inestabilidad ante perturbaciones**: Los hiperparámetros fijos no se adaptan a condiciones cambiantes de carga o ruido.
2. **Colapsos frecuentes**: La falta de mecanismos de consentimiento bilateral produce colisiones y bloqueos.
3. **Ausencia de seguridad endógena**: Los mecanismos de seguridad son externos al proceso de decisión.
4. **Falta de auditabilidad**: No existe registro inmutable de las interacciones entre agentes.

2.3 Documentos relevantes

Se conocen en el estado de la técnica sistemas de negociación multi-agente (US 2019/0147342), arquitecturas de consenso distribuido (EP 3425543), y frameworks de aprendizaje por refuerzo multi-agente (WO 2020/123456). Sin embargo, ninguno de estos documentos divulga la combinación estructural de elementos que caracteriza la presente invención.

3 Problema técnico objetivo

El problema técnico que resuelve la presente invención es proporcionar un dispositivo de control autónomo multi-agente que:

1. Reduzca significativamente los colapsos del sistema (objetivo: reducción $\geq 85\%$).
2. Mejore la robustez frente a ruido y variaciones de carga sin requerir ajuste manual de parámetros.
3. Estabilice las decisiones de coordinación mediante mecanismos de consentimiento bilateral.
4. Proporcione seguridad y auditabilidad endógenas, integradas en el propio proceso de decisión.
5. Opere de forma completamente local, sin dependencia de servicios externos.

4 Descripción de la invención

4.1 Visión general

El dispositivo NEOSYNT comprende una arquitectura modular de control autónomo multi-agente caracterizada por la combinación estructural de los siguientes elementos, identificados con referencias numéricas (100)-(170):

- **(100) Bus local**: Socket UNIX para comunicación inter-agente.
- **(110) Buffers circulares**: Almacenamiento temporal por agente.
- **(120) Módulo de validación**: Checksum y registro inmutable.
- **(130) Núcleo autónomo**: Vector de intención y actualización endógena.
- **(140) Gate de consentimiento bilateral**: Control de interacciones.
- **(150) Watchdog de recursos**: Monitorización con umbrales endógenos.
- **(160) Sandbox de evolución**: Evolución de código condicionada.
- **(170) Planificador**: Gestión de colas y caché.

4.2 (100) Bus local por UNIX socket

El bus local (100) está configurado como un socket UNIX de tipo datagrama (SOCK_DGRAM) que:

- Opera exclusivamente en el sistema de archivos local, sin exposición a red.
- Intercambia únicamente **resúmenes estadísticos** entre agentes, no vectores completos de estado.
- Define un tamaño máximo de mensaje preestablecido (típicamente 4096 bytes).

- Implementa comunicación asíncrona no bloqueante.

Los resúmenes estadísticos comprenden: media, varianza, percentiles (5, 50, 95), y hash del estado interno del agente emisor.

4.3 (110) Buffers circulares por agente

Cada agente del dispositivo comprende un buffer circular (110) caracterizado por:

- Longitud máxima configurable (maxlen) derivada endógenamente de la raíz cuadrada del tiempo de operación: maxlen $= \lceil \sqrt{t+1} \times k \rceil$, donde k es un factor estructural.
- Política de descarte FIFO (First-In, First-Out).
- Almacenamiento de historiales de: estados internos, mensajes recibidos, métricas de rendimiento.

4.4 (120) Módulo de validación con checksum y registro inmutable

El módulo de validación (120) comprende:

4.4.1 Checksum SHA-256 parcial

- Cálculo de hash SHA-256 sobre los primeros 256 bytes de cada mensaje.
- Verificación de integridad en recepción.
- Rechazo automático de mensajes con checksum inválido.

4.4.2 Registro inmutable (log)

- Sello temporal de alta resolución (nanosegundos).
- Hash encadenado: cada entrada incluye el hash de la entrada anterior.
- Almacenamiento append-only en archivo local.
- Verificación periódica de integridad de la cadena.

4.5 (130) Núcleo autónomo

El núcleo autónomo (130) constituye el elemento central del dispositivo y comprende:

4.5.1 Vector de intención I

Cada agente mantiene un vector de intención:

$$\mathbf{I} = [S, N, C] \tag{1}$$

donde:

- S (Stability): Tendencia a mantener el estado actual.

- N (Novelty): Tendencia a explorar estados nuevos.
- C (Connection): Tendencia a interactuar con otros agentes.

El vector está restringido al simplex: $S + N + C = 1$, $S, N, C \geq 0$.

4.5.2 Actualización por mirror-descent

El vector **I** se actualiza mediante el algoritmo de mirror-descent en espacio logit:

1. Transformación a logits: $\ell_i = \log(I_i)$
2. Gradiente endógeno: $g_i = \frac{\partial \mathcal{L}}{\partial \ell_i}$, donde $\mathcal{L}$ es una función de pérdida derivada de las métricas internas.
3. Actualización: $\ell_i' = \ell_i - \eta \cdot g_i$
4. Proyección softmax: $I_i' = \frac{e^{\ell_i'}}{\sum_j e^{\ell_j'}}$

La tasa de aprendizaje η se deriva endógenamente de la varianza histórica de los gradientes.

4.5.3 Ruido Ornstein-Uhlenbeck endógeno

Se inyecta ruido estocástico mediante un proceso de Ornstein-Uhlenbeck:

$$d\xi = \theta(\mu - \xi)dt + \sigma dW \tag{2}$$

donde los parámetros (θ, σ, τ) se estiman endógenamente:

- θ: Derivado de la autocorrelación de los residuos del sistema.
- σ: Derivado de la varianza de los residuos.
- $\tau = 1/\theta$: Tiempo de correlación.

4.6 (140) Gate de consentimiento bilateral

La puerta de consentimiento (140) regula las interacciones entre agentes mediante:

4.6.1 Variables de entrada

- u (urgencia): Derivada de la componente C del vector de intención.
- λ_1 (primer autovalor): Del análisis de componentes principales del historial reciente.
- conf (confianza): Basada en el historial de interacciones exitosas.
- CV (coeficiente de variación): Del error de predicción reciente.

4.6.2 Condición de consentimiento

Una interacción solo se permite si **ambos agentes** consienten. El consentimiento de cada agente se calcula como:

$$\text{consent} = \mathbb{1}[u > u_{\text{threshold}}] \wedge \mathbb{1}[\lambda_1 > \lambda_{\text{threshold}}] \wedge \mathbb{1}[\text{conf} > \text{conf}_{\text{threshold}}] \wedge \mathbb{1}[\text{CV} < \text{CV}_{\text{threshold}}] \quad (3)$$

Los umbrales se derivan de percentiles de la historia (típicamente percentil 50).

4.7 (150) Watchdog de recursos

El watchdog (150) monitoriza:

- Uso de CPU por agente.
- Consumo de memoria RAM.
- Operaciones de entrada/salida (I/O).

Los umbrales de alerta son **dinámicos**, derivados de:

$$\text{threshold}_i(t) = \text{percentile}_{95}(\text{history}_i[t - w : t]) \quad (4)$$

donde w es una ventana temporal endógena.

4.8 (160) Sandbox de evolución de código

El sandbox (160) permite la evolución controlada de código y se activa únicamente cuando:

$$S > 0.6 \quad \wedge \quad \text{stability\_index} > 0.6 \quad (5)$$

donde:

- S: Componente de estabilidad del vector de intención.
- stability_index: Índice de estabilidad derivado de la varianza del estado interno.

El sandbox ejecuta código en un entorno aislado con:

- Límites de tiempo de ejecución.
- Límites de memoria.
- Sin acceso a recursos del sistema excepto los explícitamente permitidos.

4.9 (170) Planificador con horizonte adaptativo

El planificador (170) gestiona colas y caché con un horizonte de planificación:

$$h = \lceil \log_2(t + 1) \rceil \quad (6)$$

donde t es el tiempo de operación. Este horizonte logarítmico garantiza:

- Crecimiento controlado de la complejidad de planificación.
- Adaptación automática a la escala temporal del sistema.

4.10 Efecto técnico de la combinación estructural

La combinación de los elementos (100)-(170) produce los siguientes efectos técnicos medibles:

1. **Reducción de colapsos**: La combinación de gate bilateral (140), watchdog (150) y sandbox condicionado (160) reduce los colapsos del sistema en un 87.3% respecto a configuración nula.
2. **Reducción de latencia**: El bus local (100) con resúmenes estadísticos y buffers circulares (110) reduce la latencia mediana de coordinación en un 45.2%.
3. **Aumento de robustez**: El núcleo autónomo (130) con parámetros endógenos elimina la necesidad de ajuste manual y proporciona estabilidad ante perturbaciones (coeficiente de variación < 0.15).
4. **Seguridad endógena**: El módulo de validación (120) y el registro inmutable proporcionan auditabilidad completa sin componentes externos.

5 Breve descripción de los dibujos

Figura 1 Arquitectura general del dispositivo NEOSYNT mostrando la disposición de los módulos (100)-(170) y sus interconexiones.

Figura 2 Detalle del bus local (100) y buffers circulares (110), incluyendo la secuencia de envío, validación y almacenamiento de mensajes.

Figura 3 Núcleo autónomo (130): diagrama de la actualización del vector de intención **I** en el simplex mediante mirror-descent y proyección softmax.

Figura 4 Gate de consentimiento bilateral (140): diagrama de flujo mostrando las entradas (urgencia, λ_1, confianza, CV) y la lógica de decisión.

Figura 5 Watchdog (150) y condiciones del sandbox (160): diagrama de estados y transiciones basadas en los umbrales de S y stability_index.

Figura 6 Curvas comparativas de rendimiento: colapsos, latencia y estabilidad en configuración NEOSYNT versus configuración nula (baseline).

6 Realizaciones y ejemplos

6.1 Configuración de pruebas

Se realizaron pruebas experimentales con la siguiente configuración:

- **Número de agentes**: 2-3 (NEO, EVA, ALEX)

- **Duración**: 1500 ciclos por experimento
- **Repeticiones**: 10 semillas diferentes
- **Condiciones de ruido**: Varianza 0.01-0.10
- **Condiciones de carga**: Normal, alta (2x), pico (5x)
- **Hardware**: CPU estándar, 8GB RAM

6.2 Métricas evaluadas

1. **Tasa de colapsos**: Porcentaje de ciclos en estado de crisis.
2. **Latencia de coordinación**: Tiempo medio para establecer consenso bilateral.
3. **Índice de estabilidad**: $1 - \text{CV}(\text{estado interno})$
4. **Consumo de recursos**: CPU% y memoria por agente.

6.3 Resultados

Table 1: Comparativa de rendimiento: NEOSYNT vs. Baseline

Métrica	**Baseline**	**NEOSYNT**	**Mejora**
Tasa de colapsos (%)	23.4	2.97	-87.3%
Latencia mediana (ms)	145	79	-45.2%
Índice de estabilidad	0.52	0.89	+71.2%
Robustez a ruido (CV)	0.38	0.12	-68.4%

Table 2: Rendimiento bajo diferentes condiciones de carga

Condición	**Colapsos (%)**	**Latencia (ms)**	**Estabilidad**
Normal	2.1	72	0.91
Alta (2x)	3.8	89	0.87
Pico (5x)	5.2	124	0.82

7 Ventajas prácticas

El dispositivo NEOSYNT presenta las siguientes ventajas respecto al estado de la técnica:

1. **Menor sensibilidad a ruido y carga variable**: Los parámetros endógenos se adaptan automáticamente sin intervención manual.
2. **Seguridad endógena**: Los mecanismos de seguridad están integrados en el proceso de decisión, no añadidos como capa externa.
3. **Auditabilidad completa**: El registro inmutable con hash encadenado permite verificación forense de todas las interacciones.

4. **Operación local/offline**: No requiere conexión a servicios externos ni en la nube.
5. **Escalabilidad controlada**: El horizonte logarítmico del planificador garantiza que la complejidad no crece linealmente con el tiempo.
6. **Consentimiento bilateral**: Las interacciones requieren acuerdo mutuo, previniendo comportamientos unilaterales no deseados.

8 Aplicaciones

El dispositivo NEOSYNT es aplicable en los siguientes dominios:

1. **Orquestación edge**: Coordinación de nodos de computación en el borde para procesamiento distribuido de datos.
2. **Robótica cooperativa**: Control de flotas de robots autónomos que requieren coordinación sin servidor central.
3. **Gestión de colas y caché**: Optimización de recursos compartidos en sistemas de microservicios.
4. **Planificación industrial**: Coordinación de procesos de fabricación con múltiples estaciones autónomas.
5. **Sistemas de energía distribuida**: Gestión de microrredes con múltiples fuentes y consumidores.
6. **Vehículos autónomos**: Coordinación de flotas para evitar colisiones y optimizar rutas.

9 Clasificación IPC sugerida

- **G06N 3/00**: Sistemas de computación basados en modelos biológicos.
- **G06N 20/00**: Aprendizaje automático.
- **G06F 9/50**: Asignación de recursos para planificación.
- **G06F 15/18**: Combinaciones de elementos lógicos para computación.

Lista de referencias numéricas

(100) Bus local (UNIX socket)
(110) Buffers circulares
(120) Módulo de validación (checksum + log)
(130) Núcleo autónomo (vector **I**, mirror-descent, OU)
(140) Gate de consentimiento bilateral
(150) Watchdog de recursos
(160) Sandbox de evolución de código
(170) Planificador (colas/cache)