MÁSTER UNIVERSITARIO EN
INGENIERÍA DE TELECOMUNICACIÓN

TRABAJO FIN DE MÁSTER

APLICACIONES DE LA INTELIGENCIA ARTIFICIAL EN EL DISEÑO DE REDES

Autor: Ernesto Hidalgo Felipe

Director: Alfonso Vázquez Requejo

Madrid

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

APLICACIONES DE LA INTELIGENCIA ARTIFICIAL EN EL DISEÑO DE REDES

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el curso académico 2024/2025 es de mi autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Fdo.: Ernesto Hidalgo Felipe Fecha: 05 / 07 / 2025

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Alfonso Vázquez Requejo Fecha: 05 / 07 / 2025

MÁSTER UNIVERSITARIO EN
INGENIERÍA DE TELECOMUNICACIÓN

TRABAJO FIN DE MÁSTER

APLICACIONES DE LA INTELIGENCIA ARTIFICIAL EN EL DISEÑO DE REDES

Autor: Ernesto Hidalgo Felipe

Director: Alfonso Vázquez Requejo

Madrid

A mi familia.

A mi padre, por todo lo que nos quiere.

A mi madre, por enseñarme a quererme.

A mi hermano, la persona que más quiero.

A mis amigas y a mis amigos.

A la vida, siempre.

APLICACIONES DE LA INTELIGENCIA ARTIFICIAL EN EL DISEÑO DE REDES

Autor: Hidalgo Felipe, Ernesto.
Director: Vázquez Requejo, Alfonso.
Entidad Colaboradora: ICAI-ICADE – Universidad Pontificia Comillas.

RESUMEN DEL PROYECTO

Un copiloto de ***Retrieval-Augmented Generation*** (RAG) puede acelerar fases del flujo de diseño de redes *networking*: escritura de comandos, asignación de puertos, búsqueda de ejemplos, documentación, entre otros. Para comprobarlo se ensambló un *stack* local con TinyLlama-1.1B para realizar *embeddings* y Llama-3-8B-Instruct cuantizado para la generación inteligente, alimentado con **48 fichas Markdown** de corpus de diseño propio que recopila guías IOS, RFC y buenas prácticas.

Se diseñó una **topología empresarial** de referencia que incluye VLAN, *router-on-a-stick*, interfaces *loopback*, OSPF y ACL, entre otros, conformando una arquitectura accesible pero variada con el fin de que el *testing* cubra muchos puntos diferentes.

Cuando la misma topología se simuló a mano y con ayuda de la RAG (diseño línea a línea con referencias incluidas) surgió un contraste nítido: el diseño asistido alcanzó **un 63% de cobertura funcional en la primera pasada**, mantuvo **92% de exactitud sintáctica** y recortó **un 40% el tiempo de trabajo** respecto al método no asistido. Los errores se limitaron a valores de contexto, de *prompting* o *retrieving* o a penalizaciones de extensión, todos de corrección plausible. La prueba confirma que, bajo supervisión leve, una RAG doméstica ya libera horas y eleva la disciplina documental.

Palabras clave: IA generativa, RAG, Llama, Packet Tracer, Diseño de redes.

1. Introducción

La arquitectura de redes precisa de un ciclo de diseño-configuración-simulación-ajuste reiterativo que termina por cansar al ingeniero. La irrupción de los LLM promete delegar ese trabajo mecánico, pero la comunidad duda de su fiabilidad cuando el CLI es estricto. Sin embargo, una herramienta RAG tiene el potencial de ofrecer una fuente de información que equilibre entre **creatividad y precisión** al vincular cada respuesta a su referencia en una base de datos especializada.

Profundizando en esta idea, faltaban estudios que midieran cuánta ayuda real proporciona un *stack* modesto instalado en CPU y sin conexión externa. Este trabajo se ubica en ese espacio: un laboratorio reproducible que enfrenta

generación asistida y configuración al uso, midiendo sintaxis, tiempo y éxito de pruebas.

Si la IA se puede hacer cargo de las tareas más mecánicas, el ingeniero puede dedicar su conocimiento a la lógica de protocolo y al análisis de paquetes. Por ello, la motivación combina eficiencia operativa, valor pedagógico y **aplicación empresarial**.

2. Definición del proyecto

El objetivo se concreta en medir cuánto mejora la productividad y la calidad cuando la RAG actúa como copiloto. Para ello se definió un ***checklist* de dieciséis pruebas** que abarcan conectividad, rutas y filtros, y se puntuó cada una de ellas, con el propósito de evaluar objetivamente a la herramienta.

El **corpus** fue curado a mano, con asistencia de GPT-4: cada ficha recoge un tema con sus comandos canónicos, su justificación y su licencia. Esa granularidad facilita que la búsqueda vectorial devuelva fragmentos exactos y al mismo tiempo ofrezca contexto. La decisión de trabajar en local elimina problemas de **privacidad** y de dependencia de la nube.

A nivel de escenario se diseñó una **red tipo campus** con VLANs configuradas, protocolos específicos, cuatro routers y enlaces punto-a-punto seriados, todo en Cisco Packet Tracer. Esta maqueta sirve de terreno neutro: suficiente para exigir criterios técnicos específicos, pero manejable en una sesión de prácticas.

3. Metodología

Una vez diseñada e implementada la RAG, el flujo de trabajo se dividió en tres fases: **generación, revisión, simulación**. Primero, se pide a la RAG la plantilla CLI; después, el ingeniero revisa forma y coherencia, y finalmente se lanza Packet Tracer y se ejecuta la lista de pruebas. Esa separación estanca evita que los ajustes en caliente distorsionen las métricas.

Packet Tracer aporta dos motores, Realtime y Simulation, que permiten observar la red a velocidad natural o paso a paso. La dualidad facilita medir convergencia OSPF o verificar etiquetas dot1q sin usar analizadores externos. También incorpora consolas y terminales desde las que se corren comandos de ajuste y verificación, emulando idénticamente un escenario físico.

Todas las acciones quedan registradas en una bitácora, de la que después se extraen capturas, tiempos y ***diffs* de configuración**. Para vigilar la **trazabilidad**, cada bloque generado incluye la cita al fragmento Markdown original. Así, durante la auditoría basta seguir el enlace para mostrar el pasaje de la guía o el RFC que motivó el comando.

4. Resultados

La prueba funcional certificó que **dos tercios del banco de pruebas** resultaron operativos sin intervención, lo que ya supone una victoria sobre la escritura manual. El detalle por categorías se resume en la **Tabla A**:

Categoría	Puntos	%	Comentario sintético
Sintaxis pura	1 · 3 · 5 6 · 10 · 11	**92%**	La CLI no rechazó ni una sola línea de configuración, sólo el punto 11 quedó a ½ porque los comentarios ‘ ! ’ no aparecen en todos los bloques.
Cobertura / enumeraciones	2 · 4 7 ·14	**50%**	Rangos incompletos, sufijos de IP alterados y dos omisiones (loopbacks /32 y puertos de la VLAN 12) concentran los fallos.
Parámetros de contexto	8 · 15	**25%**	Cuando el valor correcto depende de la plataforma o del grafo (next-hop, tiempo de convergencia), el modelo muestra incertidumbre.
Políticas y filtros	9 · 13	**50%**	Entrega plantillas exhaustivas, pero sobre-endurece por defecto (deny any / self traffic). En 13 permite el tráfico como se buscaba.
Validación y entrega	12 · 16	**100%**	Las pruebas de conectividad y la exportación de la maqueta (.pkt + startup-config) funcionaron a la primera, sin requerir ajustes adicionales.

Tabla A. *Clasificación de observaciones cualitativas: categoría y puntos del checklist asociados.*

A nivel de productividad el cronómetro mostró **cuarenta por ciento de ahorro medio**. Esa ganancia crece en iteraciones posteriores porque solo cambia la intención escrita; la plantilla previa actúa de base y el modelo rellena diferencias.

Finalmente, la **trazabilidad** demostró su valor cuando, semanas más tarde, se trató de auditar las fuentes de algunas decisiones asistidas. El enlace incluido en la configuración enlazó a la ficha correspondiente, sellando la cadena de custodia.

5. Conclusiones

La RAG **redujo tiempo y errores** sin mermar control humano. El corpus curado y la ejecución en local garantizan **privacidad** y estabilidad; la cita automática convierte cada línea en documento vivo. Se confirma así que un motor de escritorio puede actuar como **copiloto fiable** en aulas, laboratorios y empresas.

Quedan, no obstante, limitaciones claras: los parámetros que dependen del **contexto** específico todavía exigen revisión, y las listas necesitan un post-check que evite **omisiones** y huecos. La solución pasa por inyectar variables de entorno antes de la generación y añadir verificadores automáticos.

A corto plazo se plantea ampliar el corpus a otras plataformas, entrenar modelos multimodales que lean diagramas y enlazar la RAG con pipelines CI/CD. De cumplirse, el diseño de redes pasará de la mecánica del CLI a la declaración de intenciones, un salto análogo al que vivió el software con DevOps.

ARTIFICIAL INTELLIGENCE APPLICATIONS IN NETWORK DESIGN

Author: Hidalgo Felipe, Ernesto.
Supervisor: Vázquez Requejo, Alfonso.
Collaborating Entity: ICAI-ICADE – Universidad Pontificia Comillas.

ABSTRACT

A **Retrieval-Augmented Generation (RAG) co-pilot** can speed up several stages of the network-design workflow, such as command writing, port assignment, example hunting, documentation, and more. To test this, we assembled a local stack that uses **TinyLlama-1.1 B for embeddings** and a **quantized Llama-3-8B-Instruct for generation**, fed by **48 hand-crafted Markdown sheets** containing IOS guides, RFCs, and best-practice notes.

A reference enterprise topology was built with VLANs, router-on-a-stick, loopback interfaces, OSPF, and ACLs, among others, giving a varied yet manageable architecture so testing would cover many distinct areas.

When that same topology was simulated both manually and with RAG assistance (line-by-line design with inline citations), the contrast was clear: the assisted design reached **63% functional coverage on the first pass**, kept **92% syntactic accuracy**, and **cut overall work time by 40%** versus the unassisted method. Errors were limited to context-dependent values, prompt or retrieval misses, or token-length penalties, each easily fixed. The trial confirms that, under light supervision, a desktop RAG already saves hours and raises documentation discipline.

Keywords: Generative AI, RAG, Llama, Packet Tracer, Network Design.

1. Introduction

Network architecture requires an iterative design-configure-simulate-tune cycle that eventually wears down the engineer. Large Language Models promise to offload that mechanical work, but the community questions their reliability when the CLI is unforgiving. A RAG tool, however, can balance **creativity and accuracy** by linking every answer to a reference in a specialized database.

Yet no studies had measured how much real help a modest, CPU-only, offline stack could provide. This work occupies that space: a reproducible lab that pits assisted generation against the usual hand configuration, measuring syntax, time, and test success.

If AI can handle the most mechanical tasks, the engineer can devote expertise to protocol logic and packet analysis. Motivation therefore blends operational efficiency, pedagogical value, and **business application**.

2. Project Definition

The goal is to quantify productivity and quality gains when RAG acts as co-pilot. A **checklist of sixteen tests** covering connectivity, routing, and filtering was defined and scored to evaluate the tool objectively.

The **corpus** was hand-curated (with GPT-4 assistance): each sheet captures one topic with canonical commands, justification, and license. That granularity lets vector-search return precise fragments while still providing context. Working entirely on-prem removes **privacy** issues and cloud dependency.

At scenario level, a **campus-style network** was designed with configured VLANs, specific protocols, four routers, and serial point-to-point links, all in Cisco Packet Tracer. The mock-up offers neutral ground: technical enough to demand specific criteria yet manageable in a single lab session.

3. Methodology

After building and deploying the RAG, the workflow split into **generation, review, and simulation**. First the RAG provides the CLI template; next the engineer checks form and coherence; finally, Packet Tracer runs the test list. This watertight separation prevents live tweaks from skewing metrics.

Packet Tracer offers two engines, Realtime and Simulation, that let users watch the network at natural speed or step-by-step. The duality makes it easy to time OSPF convergence or verify dot1q tags without external sniffers. Consoles and terminals inside the tool run tuning and verification commands, mirroring a physical scenario.

All actions are logged, and later captures, timings, and **configuration diffs** are extracted. For **traceability**, every generated block embeds a citation to its original Markdown fragment, so auditors can link straight to the guide or RFC that motivated the command.

4. Results

Functional testing confirmed that **two-thirds of the test bench** came up with zero intervention, a victory over manual typing. Category details appear in **Table B**:

Category	Points	%	Summary Comment
Pure syntax	1 · 3 · 5 6 · 10 · 11	**92%**	The CLI did not reject a single configuration line; only item 11 scored ½ because '!' comments are missing in some blocks.
Coverage / enumerations	2 · 4 7 · 14	**50%**	Incomplete ranges, altered IP suffixes, and two omissions (loopbacks /32 and VLAN 12 ports) account for the failures.
Context parameters	8 · 15	**25%**	When the correct value depends on the platform or topology (next-hop, convergence time), the model shows uncertainty.
Policies and filters	9 · 13	**50%**	Provides exhaustive templates but over-hardens by default (deny any / self traffic). In items 13 it allows traffic as intended.
Validation and delivery	12 · 16	**100%**	Connectivity tests and exporting the topology (.pkt + startup-config) worked on the first try without further adjustments.

Table B. *Classification of qualitative observations: category and associated checklist points.*

Regarding productivity, the stopwatch showed a **40% average time saving**. That gain grows in later iterations because only the intent text changes; the previous template remains as a base and the model fills the gaps.

Traceability proved its worth weeks later when sources for some assisted decisions had to be audited: the embedded link in the configuration led directly to the relevant sheet, sealing the chain of custody.

5. Conclusions

RAG **cut time and errors** without diminishing human control. The curated corpus and local execution guarantee **privacy** and stability; automatic citation turns every line into a living document. Thus, a desktop engine can already act as a **reliable co-pilot** in classrooms, labs, and enterprises.

Clear limitations remain: **context-specific** parameters still need review, and lists need a post-check to avoid **omissions**. Solutions include injecting environmental variables before generation and adding automatic verifiers.

In the short term, plans call for expanding the corpus to other platforms, training multimodal models that read diagrams, and hooking RAG into CI/CD pipelines. If achieved, network design will move from CLI mechanics to intent declaration, a leap akin to the DevOps shift in software.

ÍNDICE DE LA MEMORIA

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Introducción a la inteligencia artificial

La expresión inteligencia artificial (IA) no designa una única herramienta, sino todo tipo de tecnologías "diseñadas para ejecutar tareas que requerirían de la **inteligencia activa de un ser humano**". Poco a poco normalizamos más pedirle a un programa que reconozca una cara, transcriba una conversación, mantenga un coche dentro del carril, responda una duda académica o incluso genere una imagen desde cero (Heaven, 2024). Para lograrlo se encadenan métodos que procesan grandes cantidades de datos, a partir de los cuales se crean modelos con capacidad de asociar, inferir y anticipar. Una vez desplegados, dichos modelos logran identificar patrones, extraer conclusiones y realizar predicciones con mucha precisión.

La primera tentativa seria de articular esta aspiración tuvo lugar en Hanover, durante el verano de 1956, en el encuentro bautizado como *Dartmouth Summer Research Project on Artificial Intelligence* (Solomonoff, 2023). En este, **John McCarthy**, Marvin Minsky, Claude Shannon y Nathaniel Rochester reunieron a un grupo de destacados investigadores en torno a una idea provocadora: describir con minuciosidad cada mecanismo del aprendizaje para que una máquina lo reprodujera. De aquellas sesiones salieron los objetivos, el vocabulario y el método que todavía hoy sientan las bases de la disciplina. En la **Figura 1**, en la fila de detrás, de izquierda a derecha, están Oliver Selfridge, Nathaniel Rochester, Marvin Minsky y John McCarthy. En la fila delantera, de izquierda a derecha, aparecen Ray Solomonoff, Peter Milner y Claude Shannon.

Figura 1. *En el Dartmouth AI Workshop, en 1956, algunos organizadores y participantes (Solomonoff, 2023).*

Desde que McCarthy acuñó el término "inteligencia artificial", el desarrollo de aquello que refería ha superado muchas expectativas, y se encuentra hoy en la mira de todo el mundo. Para McCarthy, una máquina inteligente debía funcionar como una **red neuronal**: **autodidacta**, creativa, abstracta, **intuitiva** y, de cierta manera, aleatoria (aquella aleatoriedad que la intuición necesita). Gracias al desarrollo exponencial de nuestros equipos, el manejo de inmensas cantidades de datos, la potencia de cálculo accesible y los avances en los algoritmos de *deep learning*, el presente ha podido materializar la realidad que hace siete décadas era solo una idea.

Hoy en día, la IA ya se ha implementado en todos los **sectores profesionales** de una u otra forma (Parlamento Europeo, Dirección General de Comunicación, 2021). Un buscador interpreta la intención de la consulta y despliega resultados que parecen pensados para cada persona; los traductores simultáneos sortean matices no literales del lenguaje humano; vehículos sin conductor circulan en proyectos piloto por avenidas con tráfico real; hay algoritmos que examinan radiografías y detectan indicios precoces de

tumoración, facilitando diagnósticos tempranos (Cañada et al., 2022); incluso nuestros asistentes personales dialogan con nosotros y aprenden de nuestra rutina.

La consecuencia de esta penetración transversal transforma profundamente las dinámicas operativas y los servicios: procesos que requerían horas se acortan, los costes disminuyen y la precisión reduce el margen de error. Analizando estas aplicaciones se observa cómo la IA se ha convertido en un **motor esencial de innovación** que redefine los estándares de eficiencia y personalización en la producción y el comercio. Lejos de ser una tendencia temporal, su influencia sigue creciendo y empuja cada día un poco más el horizonte de lo posible en la economía del siglo XXI.

Sin embargo, este auge ha traído consigo una serie de retos y dificultades. La necesidad de potencia computacional ha llevado a corporaciones como Google a acordar la construcción de pequeños reactores nucleares para generar la energía que utilizan (Terrell, 2024). Estos acuerdos reflejan el incremento exponencial en la **demanda de energía** de los modelos avanzados de IA, especialmente aquellos que utilizan los algoritmos más recientes y complejos, cuyo nivel de exigencia es inmenso. No es este el único recurso a explotar, ya que los modelos requieren de una **cantidad de datos** masiva para seguir aprendiendo. ChatGPT, por ejemplo, recibe más de 10 millones de peticiones al día, y a finales de 2023 alcanzaba los 100 millones de usuarios semanales (Lammertyn, 2024).

Este problema origina otro, el del respeto a la **privacidad** a la hora de recopilar la información, un dilema ético de aristas muy complejas. Para muchos usuarios, este uso exhaustivo de sus datos les es inmoral e ilícito, especialmente cuando los datos se recogen y procesan sin un consentimiento claro o sin asegurar el anonimato. La **mano de obra** humana que se sustituye también genera preocupación, aunque se argumenta que, por lo menos hasta ahora, todo gran avance tecnológico en la historia ha traído consigo muchos más puestos de trabajo de los que ha destruido (Vilbert, 2019). Sin embargo, la velocidad de crecimiento de este fenómeno ha suscitado un amplio debate en torno a las políticas y medidas de adaptación necesarias para mitigar el impacto de la automatización en el

empleo, y hay quien cree que este caso podría ser diferente a los que lo preceden (Nunes, 2021).

Por último, debido a la manera en la que la IA se entrena, es muy fácil que **sesgos** acaben influyendo en sus resultados, dando lugar a discriminaciones de género o raciales (Hofmann et al., 2024), entre otras. La dificultad para eliminar estos sesgos reside en que el entrenamiento de los algoritmos se basa en datos históricos que pueden reflejar las desigualdades y prejuicios presentes en la sociedad, lo que para el modelo son solo números indistinguibles. Sin una corrección adecuada, la IA podría perpetuar y amplificar estos problemas, lo que subraya la necesidad de una **supervisión ética** rigurosa y de una revisión continua.

El paradigma que ha generado la IA es tan prometedor como incierto, y sus pautas han de definirse y contrastarse con especial atención. A medida que surgen nuevas aplicaciones y enfoques, el futuro de la IA parece orientado a maximizar su protagonismo, especialmente en áreas como el procesamiento de lenguaje natural avanzado, los modelos multimodales y el aprendizaje autosupervisado. Estos avances prometen una IA aún más integrada en nuestra vida diaria, marcando el camino hacia una sociedad donde será un pilar esencial, profesional y cotidiano.

Inteligencia artificial generativa

La inteligencia artificial generativa (IAG) constituye una rama de la IA enfocada en **fabricar contenido inédito** (Goodfellow et al., 2020), sea texto, imagen, audio, código… a partir de los patrones estadísticos descubiertos en el entrenamiento. Mientras la IA "convencional" suele limitarse a clasificar, predecir o recomendar a partir de datos ya existentes, la IAG modela la distribución interna y, desde ahí, imagina ejemplos nuevos que resultan verosímiles para un observador humano. Este salto cualitativo explica que herramientas como los grandes modelos de lenguaje (LLMs) o los generadores de arte digital hayan irrumpido en la norma con una facilidad que otras ramas de la disciplina tardaron décadas en alcanzar.

La ruta formativa más común arranca con **aprendizaje autosupervisado**, donde el modelo completa fragmentos ausentes en un corpus masivo sin anotar. Después llegan ajustes supervisados sobre datos etiquetados que afinan estilo o dominio y, por último, refuerzo con *feedback* humano para garantizar su fiabilidad. El resultado es un sistema ultra versátil, capaz no solo de reconocer regularidades, sino de combinarlas en creaciones que en muchos casos son indistinguibles de los datos originales.

A partir de 2020 cobró fuerza un enfoque que le suma a esa capacidad creativa una memoria externa: la generación aumentada con recuperación, o ***Retrieval-Augmented Generation*** (RAG). El procedimiento introduce un paso previo a la redacción, por el que el modelo consulta un índice vectorial, recupera las porciones de información más pertinentes y las incorpora, palabra a palabra, durante la decodificación. Con ello se mitiga el riesgo de alucinaciones, se ofrece una cita explícita de la fuente y se introduce información que quizá no figuraba en el entrenamiento original. La formalización académica de la idea, debida a **Patrick Lewis** y colaboradores (Lewis et al., 2020), demostró que un generador enriquecido con recuperación mejora la precisión en tareas intensivas en conocimiento sin sacrificar fluidez. En la **Figura 2** se observa un esquema simple de un modelo RAG.

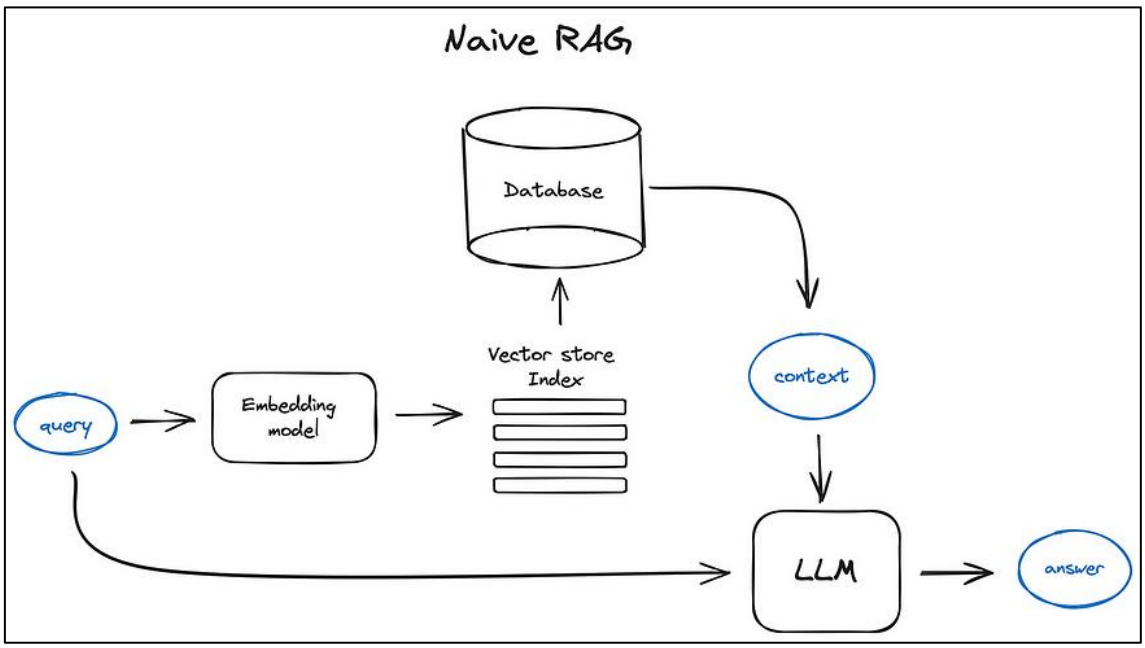


Figura 2. *Esquema simple de un modelo RAG (Ilin, 2023).*

En síntesis, la IAG amplía el horizonte de la IA porque ya no se limita a optimizar tareas existentes, sino que crea contenido que antes requería labor humana. RAG empuja ese avance un paso más allá al ofrecer al modelo una ventana dinámica hacia bases documentales siempre actualizadas, combinando la potencia generativa con la fiabilidad de la información contrastada.

Fundamentos del diseño y simulación de redes

Introducción al diseño de redes

El diseño de redes de comunicación ocupa un lugar central en la Ingeniería de Telecomunicaciones. Su objetivo principal es garantizar la **conectividad eficiente, segura y escalable** entre dispositivos y sistemas. Hablar de diseño significa decidir, en primer lugar, qué medios físicos transportarán la señal: puede tratarse de pares de cobre en despliegues de corta distancia, todavía útiles en edificios con cableado estructurado heredado, de hilos de fibra óptica que atraviesan continentes a la velocidad de la luz o de enlaces radio que sortean accidentes geográficos cuando tender cable resulta prohibitivo. A esa capa tangible se superpone la capa lógica, en la que se eligen topologías, esquemas

de direccionamiento y protocolos que dictan por dónde circulan los paquetes y con qué prioridad.

En sus inicios, las redes eran universos cerrados que unían unos pocos equipos dentro de un laboratorio. El salto cualitativo llegó con las primeras **redes locales** (LAN), a mediados de los setenta, cuando Robert Metcalfe y su equipo en Xerox PARC demostraron que la conmutación por paquetes podía encajar en un entorno corporativo sin requerir hardware especializado (Spurgeon, 2000). **Ethernet** no sólo conectó estaciones de trabajo, también introdujo la idea de un estándar abierto que cualquiera podía implementar, detalle que disparó la demanda de conectividad más allá de las paredes de una oficina. La consecuencia inmediata fue la necesidad de enlazar dominios autónomos repartidos por ciudades enteras y, poco después, por países.

El problema de enlazar redes locales dio forma a las primeras **redes de área amplia** (WAN) y, con ellas, aparecieron los desafíos que todavía hoy marcan la agenda: latencia, gestión del tráfico y compatibilidad entre tecnologías. ARPANET ilustró que la interconexión podía escalar siempre que existiera un lenguaje común y fue el banco de pruebas donde maduraron el **Protocolo de Control de Transmisión** y el **Protocolo de Internet**. Una vez normalizados en los ochenta y noventa, TCP e IP proporcionaron la base sobre la que millones de dispositivos pudieron intercambiar información con independencia de su fabricante o su sistema operativo (Townes, 2012).

La complejidad siguió creciendo cuando los proveedores de servicios comenzaron a virtualizar funciones que antes estaban soldadas al hardware. La irrupción de las redes definidas por software (**SDN**) desdobló el plano de control y el de datos: los algoritmos de encaminamiento pasaron a residir en controladores centralizados y el cerebro de la red dejó de depender de la caja que mueve los bits. Al mismo tiempo, la virtualización de funciones de red (**NFV**) permitió instanciar cortafuegos, equilibradores de carga o pasarelas de seguridad directamente sobre infraestructura genérica, lo que redujo drásticamente el ciclo de despliegue de nuevas aplicaciones.

La llegada de la **computación en la nube** añadió más capas de complejidad porque las cargas de trabajo se dispersaron por centros de datos interconectados y las políticas de acceso dejaron de ser estáticas. El ingeniero de redes ya no diseña un trazado único de aquí a allá, sino que orquesta un tejido dinámico capaz de reorganizar rutas cuando la demanda se desplaza. En paralelo, la popularización del IoT y el despliegue 5G ha multiplicado el **número de nodos** que reclaman dirección, autenticación y ancho de banda. Las proyecciones sitúan el umbral en veintinueve mil millones de dispositivos conectados en 2030 (IBM, 2023); gestionar semejante volumen plantea nuevos desafíos, especialmente en términos de escalabilidad y latencia.

En resumen, el diseño de redes ha recorrido un largo camino hasta hoy, tiempo en el que lo conforman infraestructuras globales y dinámicas, adaptándose continuamente a las demandas de un mundo cada vez más interconectado. La integración de la inteligencia artificial se postula, así, como el **siguiente paso natural** en su evolución.

La simulación como pilar del diseño

La experiencia demuestra que el plano teórico, por sí mismo, no basta para garantizar que una red cumpla las expectativas cuando el tráfico empiece a circular. Por eso la simulación se ha convertido en un pilar del diseño: recrea en un entorno virtual los mismos comportamientos que luego veremos en el bastidor, pero sin el coste (ni el susto) de cometer errores sobre equipos reales.

A distintas profundidades de detalle encontramos soluciones que van desde **Packet Tracer**, la aplicación didáctica de Cisco, hasta plataformas más exigentes como **GNS3**, **ns-3** u **OPNET**. Cada una responde a un grado de fidelidad distinto. Packet Tracer resulta idóneo cuando interesa entender la lógica de un protocolo y seguir, paso a paso, la ruta que toma un paquete. GNS3 o EVE-NG acercan el nivel de realismo al de un laboratorio físico porque permiten integrar imágenes de sistemas operativos reales sobre máquinas virtuales. ns-3 y OPNET, en cambio, se orientan a la investigación académica: modelan con precisión los temporizadores del *stack*, los retardos de propagación o el comportamiento estocástico de las colas, de modo que se puedan publicar resultados

reproducibles. En la **Figura 3** se muestra una topología construida en Packet Tracer, y la **Figura 4** recoge la misma configuración en laboratorio, donde cada sistema y cable corresponden a su homólogo virtual.

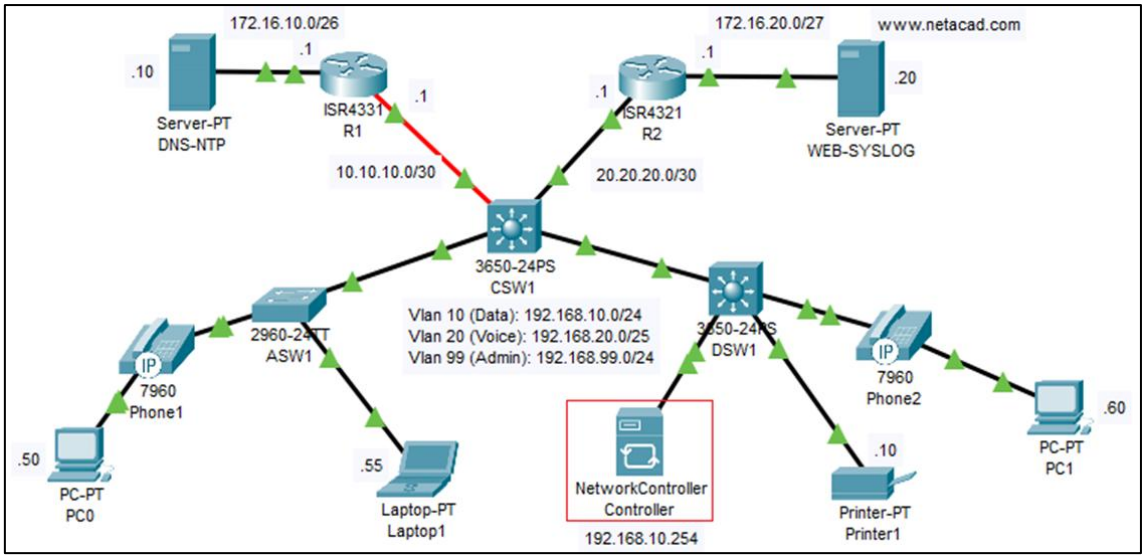


Figura 3. *Ejemplo de topología simulada en Cisco Packet Tracer (Ing_Percy, 2024).*

Figura 4. *Misma topología (Figura 3) en físico (Ing_Percy, 2024).*

Estos entornos permiten alterar, en cuestión de segundos, parámetros que en producción costarían horas, como el ancho de banda, latencia, políticas de enrutamiento o perfiles de tráfico. El ingeniero puede saturar un enlace, inyectar ráfagas anómalas o introducir fallos deliberados y observar cómo reacciona el sistema. El valor pedagógico es evidente, pero el **impacto económico** resulta todavía mayor, porque identificar un cuello de botella en fase de simulación evita paradas de servicio y compras de hardware innecesarias más adelante.

No se trata solo de ahorrar. Al capturar trazas y métricas durante cada ensayo, la simulación documenta el **razonamiento técnico** que conduce a la versión final de la red. Esa bitácora facilita auditorías, sustenta la redacción de informes de capacidad y, cuando llegan nuevas necesidades, sirve de punto de partida para iteraciones sucesivas. De este modo, el ciclo de diseñar-simular-ajustar se convierte en un proceso continuo que aproxima el resultado final al comportamiento óptimo antes de que el primer paquete real abandone la tarjeta de red.

Cambios en el paradigma de diseño y simulación

En poco más de una década el paisaje de las telecomunicaciones ha pasado de topologías relativamente estables a un entramado que se expande y se reconfigura casi a la misma velocidad que cambian las aplicaciones que lo consumen. La irrupción de SDN y la virtualización ya presentada, el salto a la nube y la marea de dispositivos IoT han multiplicado tanto los puntos de acceso como la variabilidad del tráfico. Cada jornada combina flujos de vídeo en *streaming*, partidas de videojuegos en línea y sensores industriales que disparan ristras de bytes constantes.

A la presión del caudal se suma un compromiso ineludible con la eficiencia energética: no basta con trazar rutas rápidas, también hay que medir el impacto de cada ruta en la factura eléctrica y en la huella de carbono. Ese doble reto de tráfico imprevisible y consumo contenido exige que el ingeniero cruce, en tiempo casi real, datos de rendimiento, catálogos de equipo y normas que se actualizan de forma regular.

Aquí encaja la inteligencia artificial generativa. Una **RAG entrenada sobre bases documentales específicas de redes** (manuales, RFC, configuraciones contrastadas) actúa como repositorio vivo. Durante la fase de diseño o de simulación, basta una consulta para recuperar fragmentos relevantes, cotejar alternativas y esbozar una topología inicial sustentada en evidencia. El modelo no reemplaza el criterio humano, funciona como un asistente que sirve conocimiento disperso bajo demanda, reduciendo iteraciones y acortando la distancia entre la hipótesis y la prueba.

El proyecto se apoya en esa premisa: **explorar de qué modo una RAG especializada puede acelerar las tareas de diseño y simulación, mantener la coherencia con las exigencias actuales y, al mismo tiempo, dejar rastro verificable de cada decisión que tome el ingeniero**.

ESTADO DEL ARTE

Evolución de la IA generativa y aparición de los modelos conversacionales

La idea de que una máquina pudiera *crear* algo nuevo y no solo clasificar lo ya existente apareció casi al mismo tiempo que la propia inteligencia artificial. En los sesenta, con ordenadores que ocupaban habitaciones enteras, algunos pioneros se dedicaron a escribir programas capaces de hilar poemas o elaborar melodías sencillas mediante reglas gramaticales y patrones rítmicos explícitos (Norman, 2025). A la vista de lo que hoy llamamos modelos, aquellos artefactos resultaban ingenuos; sin embargo, demostraron que se podían **simular aspectos de la creatividad**, y sentaron la premisa de que una parte del arte es estructura.

El salto cualitativo llegó en los ochenta, cuando las **redes neuronales** ganaron atractivo y empezaron a entrenarse con algoritmos de retropropagación. Las primeras capas ocultas mostraron que un sistema podía aprender representaciones internas difíciles de codificar a mano. El paralelismo más citado, la comparación entre la neurona biológica y su abstracción matemática que aparece en la **Figura 5**, sirve de metáfora y justificación. La teoría alentaba, pero el hardware frenaba: procesar matrices enormes en la CPU de la época requería paciencia monástica.

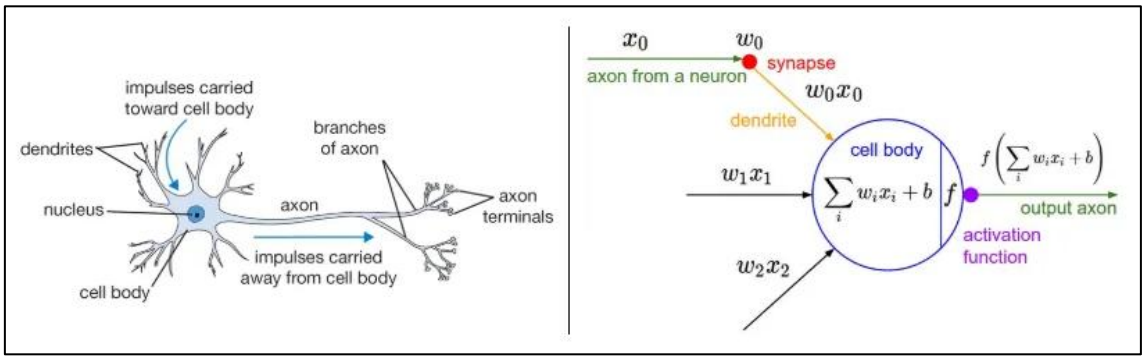


Figura 5. *Neurona biológica (izquierda) y su modelo matemático (derecha) (Li et al., 2024).*

Con el cambio de siglo la situación se invirtió. El universo digital estalló, los discos se llenaron de texto, imágenes y vídeo, y las GPU que antes rendían al servicio de los videojuegos demostraron una capacidad insólita para mover vectores en paralelo. En 2015 Google anunció las primeras unidades de procesamiento tensorial (TPUs), aceleradores pensados desde el silicio para multiplicar el ritmo de entrenamiento (McHugh-Johnson, 2024). Esa conjunción de **datos abundantes** (multiplicado por el intercambio de avances en plataformas de código abierto, que fomentaba un clima de **colaboración** que acercaba aún más un fin que compartía toda la comunidad tecnológica) y **capacidad de cómputo** desencadenó lo que hoy se llama **aprendizaje profundo**. De repente, las redes podían alojar decenas de capas, absorber millones de ejemplos y, lo más importante, generalizar con una fidelidad superlativa.

Aun así, los sistemas seguían atrapados en un problema de secuencia: los modelos recurrentes procesaban las palabras una por una y el coste crecía con la longitud del texto. El giro se produjo en 2017, cuando Vaswani et al. publicaron "*Attention Is All You Need*" (Vaswani et al., 2017). Nacía la **arquitectura Transformer**, que sustituía la memoria explícita por mecanismos de atención capaces de ponderar en paralelo todas las posiciones de una frase. Este artículo dio lugar a una carrera de escalado que elevó el número de parámetros de cientos de millones a decenas de miles de millones.

Con los Transformers llegó el fenómeno inesperado de la **capacidad de conversación**. Al entrenar sobre corpus gigantescos y afinar después con instrucciones humanas, los modelos empezaron a sostener diálogos coherentes, a recordar lo dicho hace varios turnos y a responder en registros distintos con un mínimo de indicaciones. De esta instancia a los asistentes que están hoy tan a la orden del día median apenas ocho años de entrenamiento, ajustes y refuerzo.

Aun así, la **fiabilidad de los hechos** seguía siendo un obstáculo. Un modelo puramente paramétrico guarda el conocimiento disperso en sus pesos, y si el dato no aparece en el corpus o cambia con el tiempo, la salida alucina o se queda obsoleta. Para sortearlo emergió la *Retrieval-Augmented Generation* (**RAG**). El procedimiento abre una

ventana al exterior: antes de formar la respuesta, un recuperador vectorial busca pasajes relevantes en una base de documentos y los ofrece al generador, que los integra durante la decodificación. Con este truco se gana trazabilidad (cada afirmación apunta a su fuente) y se actualiza el saber sin reentrenar la red completa (Lewis et al., 2020).

En pocas décadas, la IA creativa ha pasado de haikus programados a sistemas conversacionales capaces de explicar una teoría matemática o sugerir una topología de red, y lo ha hecho apoyándose en **tres variables**: más datos, más cómputo y mejores arquitecturas. La aparición de RAG añade una cuarta: la posibilidad de enlazar, en tiempo real, la memoria del modelo con **bases de conocimiento vivas**, ingrediente que resulta especialmente sugerente cuando el dominio exige decisiones informadas por normas, configuraciones históricas y métricas cambiantes (como el del diseño de redes). Con este trasfondo, el siguiente apartado analizará el funcionamiento interno de un modelo conversacional.

Funcionamiento de una IA conversacional

Aunque se tomará ChatGPT como caso ilustrativo, la mecánica que se describe a continuación es extrapolable, con variaciones menores, a cualquier **modelo generativo** de última generación, incluido el basado en Llama que alimentará la RAG especializada de este proyecto. Analizar esa mecánica es importante, puesto que la misma **cadena de procesos** – tokenización, generación de *embeddings*, aplicación de atención y ajuste con retroalimentación humana – es la que más tarde convertiremos en motor para el diseño de redes.

¿Qué es ChatGPT?

ChatGPT pertenece a la familia ***Generative Pre-trained Transformers*** de OpenAI. Su función esencial es recibir texto, interpretar el contexto de la conversación y devolver una respuesta coherente y pertinente. Para llegar a ese punto, el sistema atravesó una secuencia de entrenamiento en tres actos. Primero, un **preentrenamiento** autosupervisado sobre ingentes cantidades de libros, artículos científicos y foros permitió que la red interiorizara estructuras sintácticas y matices semánticos generales. A

continuación, se llevó a cabo un **ajuste supervisado** con ejemplos cuidadosamente etiquetados que definieron tareas y formatos concretos de salida. Por último, intervino el aprendizaje por refuerzo con retroalimentación humana (**RLHF**), en el que evaluadores puntúan varias respuestas candidatas y guían al modelo hacia formulaciones informativas y amables, tal como explica Nate Gentile en su vídeo "*¿Cómo funciona ChatGPT? La revolución de la inteligencia artificial*" (Nate Gentile, 2023). Será en este vídeo en el que nos basaremos para desarrollar los principios técnicos de la herramienta.

Del texto a la representación numérica

El primer paso después de recibir una pregunta consiste en normalizar la secuencia y dividirla en fragmentos manejables. El sistema aplica una **lematización** ligera que reduce flexiones a su raíz (*computing*, *computed*, *computes* → *compute*) y, acto seguido, recurre a la **tokenización** por subpalabras. Este sería un ejemplo:

> **Entrada del usuario**: "What is cloud computing?"
> **Tokens generados**: ["What", "is", "cloud", "comput", "ing", "?"]

El método permite reutilizar fragmentos comunes; si mañana aparece *computationally*, bastará con combinar "comput", "ation", "ally" en lugar de añadir un término nuevo al vocabulario.

Gracias a la tokenización, se reduce la cantidad de **palabras únicas** que el modelo necesita manejar y se mejora la **eficiencia computacional** al comprimir información en una representación matemática más compacta.

Posteriormente, los tokens se convierten en **vectores numéricos** mediante ***embeddings***, representándolos en un espacio matemático donde palabras con significados similares están más cerca. De esta forma, un ejemplo de *embedding* sería:

> "computación" → [0.12, -0.85, 1.03, ...]
> "redes" → [-0.32, 0.98, -0.43, ...]
> "protocolos" → [1.34, -0.67, 0.21, ...]

También se les añadirá una **codificación posicional** para saber la ubicación de cada palabra dentro de la oración. Se genera un vector adicional que indica la posición de cada token en la secuencia:

Posición 1 → [0.98, 0.34, -0.12, ...]
Posición 2 → [1.03, -0.45, 0.89, ...]
Posición 3 → [-0.87, 0.22, 1.31, ...]

Esta información se suma a los *embeddings* antes de pasarlos a las capas del Transformer. En la **Figura 6** se esquematiza este modelo.

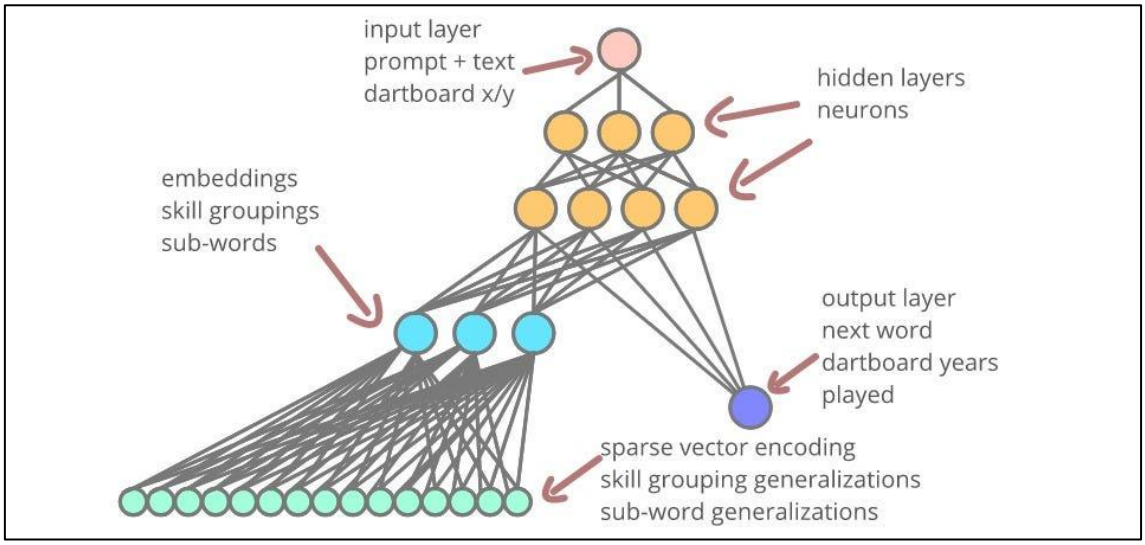


Figura 6. *Esquema del modelo de embedding de ChatGPT (Hirani, 2023).*

Procesamiento interno: Transformer

Con los vectores ya alineados en el espacio semántico-posicional, la frase entra en una cadena de capas Transformer que puede prolongarse treinta, cuarenta o más saltos. El recorrido siempre empieza igual: cada token se proyecta a tres matrices distintas denominadas ***queries***, ***keys*** y ***values***. El truco consiste en enfrentar cada *query* con todas las *keys* del resto de la oración para medir, mediante un producto punto escalado, hasta qué punto una palabra necesita fijarse en otra. El resultado traduce similitudes en porcentajes y, con esos pesos, se combinan los *values*; lo que emerge es una versión del

contexto en la que las piezas relevantes quedan realzadas y el ruido retrocede. En la **Figura 7**, parte superior: procesamiento secuencial de las palabras, una a la vez. Parte inferior: todas las palabras se prestan atención simultáneamente, lo que permite comprender el contexto completo en un solo paso (arquitectura Transformer).

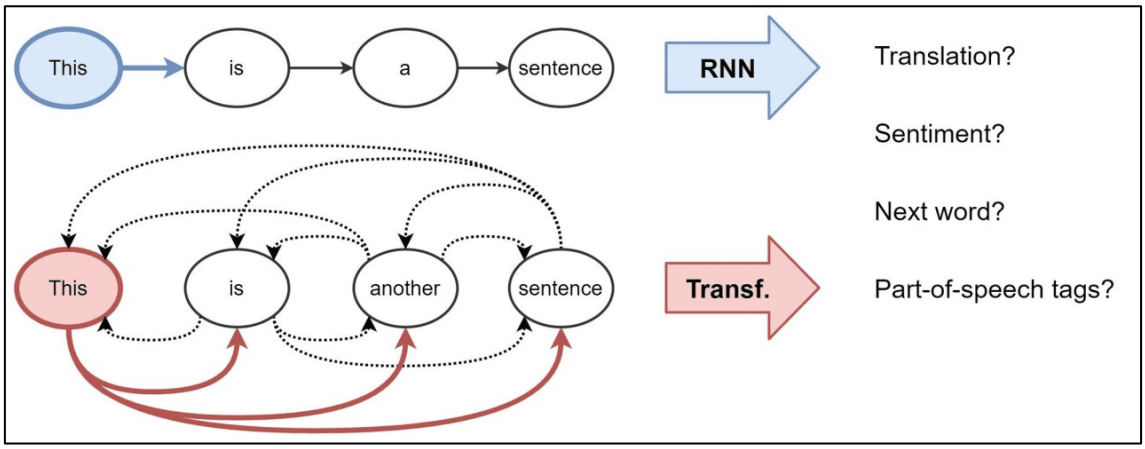


Figura 7. *Comparación entre procesamiento RNN y Transformer (Lopez et al., 2024).*

Esa operación no sucede una sola vez, sino que se desdobla en varias instancias que trabajan en paralelo: ***multi-head attention***. Una cabeza puede especializarse en vínculos sintácticos – la relación sujeto-verbo, por ejemplo –, otra prefiere atender a familias de sinónimos dispersos por la frase y una tercera conecta ideas que saltan de un párrafo al siguiente. Al final de la capa, las salidas de todas las cabezas se concatenan y atraviesan una proyección lineal que las condensa para la siguiente etapa. Gracias a esta disposición, el modelo mantiene a raya tres viejos problemas del lenguaje natural: la polisemia, la necesidad de contexto amplio y la flexibilidad gramatical.

Pero la atención, por sí sola, no basta. Cada bloque incluye un atajo (la **conexión residual**) que suma la entrada original a la salida transformada. Ese puente evita que el gradiente se evapore a lo largo de decenas de capas y facilita que la red aprenda correcciones pequeñas en lugar de reconstruir la señal desde cero. Justo después llega **LayerNorm**, una normalización que estabiliza la escala de los activaciones y acelera la convergencia. A continuación, entra en escena una red ***feed-forward*** densa, dos capas

lineales separadas por una activación no lineal, que introduce interacciones adicionales entre dimensiones del *embedding*.

El ciclo atención, residual, normalización, *feed-forward*, residual, normalización se repite capa tras capa. Cada vuelta supone **afinar millones de pesos y sesgos**: el optimizador ajusta esos parámetros empujado por el gradiente que deriva de la ***cross-entropy loss*** (Mao et al., 2023). Cuando el corpus es descomunal y el modelo roza los centenares de miles de millones de parámetros, el entrenamiento se reparte entre docenas de GPU o TPU.

Al final del *pipeline*, lo que regresa no es un simple vector, sino una representación contextual rica que condensa quién actúa, qué acción realiza, sobre qué objeto y bajo qué matices. Es esa destilación la que permite, en la fase de inferencia, que el modelo escoja el siguiente token con un pulso que resulta natural al criterio humano y, en cascada, genere párrafos enteros donde las referencias se sostienen y las transiciones cuadran.

Fases de entrenamiento

El proceso de aprendizaje que ha moldeado a ChatGPT no ocurre de una sola vez, sino que se encadena en tres etapas bien diferenciadas que se ejecutan una tras otra y se retroalimentan:

- **Preentrenamiento autosupervisado:** el modelo aprende a completar huecos en un corpus gigantesco, absorbiendo patrones estadísticos generales.
- **Ajuste supervisado:** un conjunto curado de pares pregunta-respuesta lo entrena para seguir instrucciones y presentar la información con formatos concretos.
- **Refuerzo con retroalimentación humana (RLHF):** evaluadores puntúan varias réplicas y un modelo de política aprende a favorecer las mejor valoradas, puliendo estilo y seguridad.

En la práctica, estos tres pasos se encadenan sin interrupción. El **preentrenamiento** expone a la red a billones de tokens y le permite interiorizar

regularidades gramaticales y asociaciones semánticas imposibles de programar manualmente. Luego, el **ajuste supervisado** refina la obediencia a consignas y la forma de presentar resultados, mientras que el **RLHF** corrige matices de tono y precisa el grado de detalle.

Infraestructura y hardware de alto rendimiento

Dado el tamaño de un modelo así, su entrenamiento requiere infraestructura computacional masiva. Se entrenó utilizando decenas de miles de **GPUs** de NVIDIA A100 (Samsi et al., 2023) en clústeres optimizados para entrenamiento de modelos de IA en centros de datos de Microsoft, aprovechando recursos computacionales masivos (Sims, 2023).

También se emplean **Unidades de Procesamiento Tensorial (TPUs)** diseñadas específicamente para acelerar cálculos de matrices en redes neuronales. El modelo se optimiza con **técnicas de reducción de precisión** (*Mixed Precision Training*), minimizando el consumo de memoria sin afectar, prácticamente, su eficacia.

Generación de la respuesta

Durante la inferencia, el modelo repite el ciclo de atención con la conversación acumulada y obtiene una distribución de probabilidad para el siguiente token. Parámetros como *temperatura* y *penalización de repetición* regulan si se privilegia la opción más probable o se introduce diversidad. El proceso continúa token a token hasta producir la secuencia final.

Las fases de la interacción se resumen en la **Tabla 1**:

Fase	Descripción breve
Entrada	El usuario formula una pregunta. La claridad, el contexto y los términos técnicos adecuados aumentan la precisión de la respuesta.
Procesamiento	Tokenización → *embeddings* + posiciones → pila Transformer con Q / K / V + *multi-head attention* → predicción del siguiente token.
Salida	El modelo decodifica tokens hasta completar la respuesta en texto legible.

Tabla 1. *Esquema de interacción de ChatGPT.*

Comprender esta cadena, desde la fragmentación inicial hasta la elección del último token, resulta crucial para el proyecto, porque la RAG que se construirá sobre Llama seguirá exactamente **la misma ruta interna**. La diferencia estará en la pasarela de recuperación externa: un índice técnico de manuales, RFC y configuraciones que anclará cada sugerencia a su fuente, mejorando la trazabilidad sin tocar la mecánica fundamental descrita aquí.

Del aprovisionamiento manual al diseño basado en intención

Durante años la construcción de una red empezó y, con frecuencia, acababa en la consola serie de cada dispositivo. El administrador introducía, línea a línea, comandos que fijaban VLAN, rutas estáticas o listas de acceso, anotaba el cambio en un libro de guardia o en una hoja de cálculo y pasaba al siguiente equipo. Aquel método resultaba tolerable en un armario con tres switches, pero cuando la topología crecía las incoherencias aparecían, la recuperación ante fallos dependía de la memoria humana y las ampliaciones consumían días.

Llegada de la programabilidad

La primera innovación llegó con la separación del plano de control y del plano de datos que propone el ***Software-Defined Networking***. Al centralizar la lógica de encaminamiento en un controlador y exponerla mediante API, SDN convirtió la red en un **sistema programable**, de ahí que los cambios pudieran describirse con código y versionarse como el software convencional (Feamster et al., 2014). Poco después, la

virtualización de funciones de red permitiría levantar cortafuegos, balanceadores o pasarelas IPSec sobre hardware genérico.

Con la programabilidad llegaron los modelos de datos YANG y los protocolos NETCONF/gRPC, que ofrecían esquemas estructurados para declarar estados y recibir confirmaciones máquina-a-máquina (Cisco Systems, s. f.).

El salto declarativo: *Intent-Based Networking*

Incluso con automatización, escribir la "receta" detallada seguía siendo tarea del ingeniero. La respuesta ha sido describir *qué* se persigue y no *cómo* lograrlo. En un escenario de *Intent-Based Networking* (**IBN**) el operador fija la intención – por ejemplo, "cualquier flujo de vídeo crítico debe atravesar la red con una latencia < 15 ms y una disponibilidad del 99,99 %" –. El sistema traduce esa frase en políticas, las despliega, monitoriza los indicadores y reajusta si la telemetría revela desvíos. Cisco resume la idea como el puente que alinea continuamente la red con los objetivos de negocio (Cisco Systems, s. f.).

La **tubería IBN** suele dividirse en tres fases:

- **Traducción:** la intención natural se convierte en variables comprensibles por los controladores.
- **Orquestación:** se empujan cambios consistentes al tejido físico y virtual.
- **Aseguramiento:** telemetría alimenta un motor que confirma el cumplimiento o revierte la acción si algo se tuerce. El ciclo se ejecuta sin intervención manual, de modo que la red deja de ser un conjunto de cajas estáticas y actúa como un organismo que se reajusta ante fallos o picos de demanda.

IA y AIOps: detección, predicción y ajuste fino

Cuando el bucle cerrado necesita decidir en segundos si cambia la política de colas, la inteligencia artificial entra en escena. Plataformas de operaciones asistidas por IA (**AIOps**) filtran millones de métricas en busca de anomalías y sugieren la causa raíz

antes de que el usuario note el impacto (Poda, 2025). Esa capacidad analítica, además de acelerar la resolución de incidencias, suministra datos históricos para alimentar modelos predictivos que con el tiempo adelantan la acción correctiva.

RAG como brújula documental

La automatización despliega, y la IA vigila, pero falta un punto: la justificación técnica. Aquí aparece la **RAG**. Al conectar el pipeline de intención con un índice vectorial que almacena **documentación relevante**, la RAG puede recuperar en segundos los fragmentos pertinentes y proponer variantes explicadas. El ingeniero lee la cita, confirma o ajusta, y la decisión pasa a producción con trazabilidad completa. La misma mecánica que soporta un asistente conversacional se transforma en un apoyo contextual para la ingeniería de redes.

El proceso aún no es perfecto: faltan estándares universales para modelar la intención, el consumo de cómputo de los bucles analíticos amenaza con elevar la factura energética y la visibilidad sobre algoritmos de inferencia sigue siendo limitada. Sin embargo, la secuencia CLI → SDN/NFV → NetDevOps → IBN demuestra que la tendencia apunta a redes cada vez más **declarativas, autoverificables y documentadas en tiempo real**. La incorporación de RAG pretende aportar la pieza que completa el ciclo, el conocimiento técnico preciso en el instante exacto en que se decide la política de red.

Beneficios y límites actuales de la IA en el diseño de redes

Oportunidades operativas

La irrupción de la inteligencia artificial en la ingeniería de redes ha convertido un oficio esencialmente artesanal en un proceso guiado por **modelos estadísticos** y **motores de inferencia en tiempo real**. Los resultados de Duan et al. en el artículo "*AI-Generated Network Design: A Diffusion Model-based Learning Approach*" (2023) sobre diseño de topologías mediante modelos de difusión (Huang et al., 2023) son una prueba temprana: a partir de los requisitos de latencia, disponibilidad y consumo, la red imagina varias configuraciones y entrega al ingeniero un abanico razonado de alternativas. Esa capacidad

generativa se completa con la vertiente **adaptativa**: cuando un enlace se congestiona o una placa comienza a sobrecalentarse, el mismo algoritmo evalúa la situación y propone, casi al vuelo, un nuevo reparto de flujos o un desvío provisional sin esperar a la intervención humana.

Este tipo de inteligencia se alimenta, sobre todo, de **datos históricos**. Usama et al. recopilan en su revisión más de una docena de despliegues donde redes neuronales jerárquicas, entrenadas sin etiquetas, detectan **patrones que preceden** a caídas de *throughput* o a ráfagas de latencia (Usama et al., 2019). En una red troncal de proveedor, ese aviso con dos horas de antelación basta para provisionar capacidad extra y evitar que la congestión se materialice durante el *prime time*. En el campus de una universidad ocurre algo similar: la pasarela Wi-Fi ajusta potencia y ancho de canal cada vez que las aulas se vacían y el tráfico se traslada a las residencias, ahorrando decenas de kilovatios hora a la semana.

La mejora alcanza también a la **calidad de servicio (QoS)**. Los modelos aprenden a distinguir qué flujos sostienen aplicaciones críticas (ERP, vídeo quirúrgico, transacciones bursátiles) y les asignan prioridad sin necesidad de escribir reglas estáticas para cada dirección IP o puerto. Cuando llega el pico de un lanzamiento de software, la plataforma reduce de forma temporal el ancho de banda destinado a copias de seguridad y lo devuelve en cuanto la demanda baja. Esa lógica de "ascensor" mantiene satisfechos a los usuarios y evita sobredimensionar enlaces por si acaso.

En **5G** el efecto es todavía más visible. Conciertos, partidos y manifestaciones son eventos conocidos, pero la dimensión real de la demanda varía con la meteorología, la cartelera o la hora de inicio. Los **modelos predictivos** cruzan agenda pública, datos de movilidad y series históricas de consumo y, unas horas antes, deciden si hace falta activar más portadoras, reforzar la sincronización o desplegar celdas temporales. Una vez termina el evento, la red revierte la configuración y libera recursos para otras áreas. De nuevo, la IA evita el **sobregasto energético** de mantener, día tras día, un nivel de potencia pensado sólo para los grandes momentos.

La **sostenibilidad** se ha convertido, de hecho, en argumento principal. En los centros de datos, la telemetría de temperatura y de consumo eléctrico entra en la misma ecuación que la métrica de latencia: si un bastidor se calienta por encima de su umbral, el orquestador migra cargas a zonas frías y, cuando el tráfico lo permite, suspende servidores que no aportan valor en ese instante. El ahorro acumulado supera ya lo que se ganaba con la mera virtualización hace una década y se alinea con los compromisos de neutralidad climática que empiezan a exigir tanto los accionistas como los reguladores.

Donde la IA marca, quizá, la diferencia más notoria es en la **seguridad**. Los sistemas de firmas se han quedado cortos frente a ataques de vanguardia, frente a los modelos de **autocodificadores** que detectan **desviaciones sutiles** del patrón normal y clasifican la anomalía en décimas de segundo. Cuando aparece un pico inusual de peticiones DNS, la red puede aislar la zona, redirigir el tráfico a un *sinkhole* y notificar al SOC con un informe que explica cuál fue la variable disparada. La contención deja de depender de la reacción humana y se convierte en parte del tejido.

Riesgos y salvaguardas

Todas estas ventajas presentan su contraparte, y así lo cuentan Brey y Dainow en "*Ethics by Design for Artificial Intelligence*" (Brey & Dainow, 2024). Una red que cambia sola puede volverse **indescifrable**. La presión por minimizar el *time-to-mitigate* empuja a simplificar la interfaz y a esconder complejidad, de manera que el operador deja de ver la lógica interna y sólo recibe un veredicto ("se cierra la ruta X"). Si la recomendación resulta errónea o discrimina sin querer a un grupo de usuarios, rastrear la decisión hasta la línea de datos que la inspiró se vuelve complicado. Reaparece, así, la necesidad de **explicabilidad**: cada ajuste debería venir acompañado de la métrica que lo motivó y del fragmento de telemetría que lo respalda.

Viene luego el problema del **sesgo**, del que ya hablamos en la sección Introducción a la inteligencia artificial. Si el histórico que alimenta al modelo representa sobre todo sedes corporativas de alto tráfico y deja fuera enlaces rurales con patrones distintos, el algoritmo inferirá como normal valores que perjudican al último grupo. El sesgo se puede

esconder en la priorización de colas, en la asignación de espectro o en la selección de rutas alternativas cuando la demanda aprieta. La respuesta empieza por **auditorías periódicas**: inyectar datos sintéticos, medir la salida y comparar la distribución de recursos para detectar desviaciones sistemáticas.

La **privacidad** es el siguiente frente. Para alimentar autocodificadores y detectores de anomalías se capturan cabeceras, *flows* y, a veces, cargas útiles parciales. Cuanto más fino sea el muestreo, mejor la predicción, pero mayor la superficie que un atacante podría explotar. El **cifrado *end-to-end*** ya no es accesorio, pues la segmentación y la rotación de claves pasan a la fase de diseño y la política de retención de datos se revisa con cada salto de versión del modelo. Algunos operadores, en lugar de almacenar paquetes completos, extraen estadísticas en el borde (latencia, *jitter*, entropía de carga) y descartan el resto, sacrificando algo de precisión para reducir su exposición.

Finalmente, la **huella energética** de la propia inteligencia. Ejecutar inferencias de forma continua consume vatios que antes se consideraban *overhead* marginal. A escala de red global, los loops correctivos permanentes pueden trasladar el gasto del plano de datos al plano de control. De ahí que varios proveedores hayan introducido **presupuestos de carbono** en sus hojas de ruta y ajusten la frecuencia de inferencia al punto donde la ganancia de eficiencia compensa, realmente, el coste adicional de cómputo.

El balance, por tanto, es un ejercicio de **ingeniería** y de **gobernanza**. Las redes de próxima generación se benefician de motores que diseñan, operan y protegen mejor que los métodos clásicos, pero sólo cuando se acompasan con mecanismos de transparencia, auditorías de sesgo y límites claros a la recolección de datos. Si esa supervisión se integra desde la fase de proyecto, y no como reacción a incidentes, la IA tiene margen para desplegar todo su potencial sin sacrificar la equidad, la privacidad o el compromiso de sostenibilidad que reclama la sociedad.

Principios y herramientas de simulación de redes

Un **simulador de redes** empieza por traducir cada evento – la salida de un paquete, el avance por un enlace, la llegada a una cola, la decisión de un algoritmo de encaminamiento – en **marcas de tiempo**. El motor recorre esa línea temporal y calcula cuánto se demorará el paquete en alcanzar su destino. Si un enlace publica 1 Gb/s y la trama ocupa 1000 bits, el simulador reserva una milésima de segundo en su **agenda virtual**. Así, el resultado es una estimación de latencia y de ocupación de búfer que, cuanto más realista se quiera, más detalles debe cargar.

Para hacerse una idea de cómo se construye un modelo basta con recorrer la red de abajo a arriba:

- **Capa física**: aquí solo importa **cuánto tarda** la señal en recorrer el cable y **cuánta información cabe por segundo**. Con anotar la distancia virtual y el ancho de banda el simulador ya puede calcular el retraso básico.

- **Capa de enlace**: ahora los dispositivos comparten el medio. El simulador decide si basta con suponer que el intercambio es ordenado o si necesita reflejar posibles "pisadas" entre tramas y las pequeñas pausas que se dan cuando alguien cede el turno.

- **Capa de red**: por encima, los enrutadores conversan entre sí. El modelo tiene que saber **cada cuánto** mandan mensajes de estado (por ejemplo, los avisos periódicos de OSPF) y si el paquete carga algún dato extra, como la etiqueta VLAN, que ocupa unos pocos bytes y reduce ligeramente el espacio útil.

En todo proyecto conviene ajustar el nivel de detalle al tipo de respuesta que se busca. Si el objetivo es **estimar la latencia media** de un conjunto de cuatro departamentos reunidos en una red /22, basta utilizar un modelo simplificado que asigne a cada salto un retardo fijo y una pequeña probabilidad de pérdida. En cambio, si se quiere comprobar si un ***router-on-a-stick*** gestionará sin problemas el tráfico de esas mismas cuatro VLAN, el simulador debe representar con mayor precisión la etiqueta *dot1q* que

añade cuatro bytes a cada trama y el tiempo extra que el procesador dedica a esa conmutación. A medida que aumenta la fidelidad, crecen el consumo de memoria y el tiempo de cálculo, de modo que la clave está en detallar solo los aspectos que influyen de forma significativa en la métrica que se pretende evaluar.

Los simuladores profesionales permiten introducir los mismos **comandos de configuración** que se usarán después en los equipos físicos. El programa no entiende el sistema operativo del fabricante, simplemente convierte cada línea en parámetros internos. Así, una orden como *switchport access vlan 11* hace que el modelo añada la etiqueta VLAN correspondiente y ajuste el cálculo de tiempos y capacidad. Con *router ospf 1* el motor genera los mensajes de saludo que el protocolo envía de forma periódica y, si no recibe respuesta, simula la reconvergencia de rutas. Quien practica en este entorno adquiere **procedimientos operativos** válidos para la red real, aun cuando el hardware todavía no está instalado.

Cuando es preciso verificar detalles de sintaxis o comprobar el comportamiento exacto de un sistema operativo, se recurre a la **emulación**. En ese caso se ejecuta la imagen completa del SO de red sobre una máquina virtual o un contenedor. El consumo de recursos es mayor, porque cada paquete atraviesa el mismo código que en el equipo físico y las tablas de rutas se procesan con idénticos algoritmos. A cambio, cualquier error de configuración aparece de inmediato, igual que sucedería en producción.

En la práctica **se pueden combinar ambos enfoques**. Un análisis inicial de calidad de servicio puede realizarse con un modelo ligero que representa los flujos como tasas de llegada, y una vez ajustados parámetros como la máscara, la MTU o la métrica de OSPF, se exporta esa topología a un emulador que ejecuta el sistema operativo real y genera los mismos registros que recibirá el centro de operaciones. Si los resultados coinciden, el diseño se considera validado.

Todo depende, en última instancia, de encontrar el punto de equilibrio entre **escala y detalle**. Modelar a nivel de paquete una red de varios miles de direcciones puede superar la capacidad de un equipo de sobremesa y emularla completamente exigiría un clúster. La

solución habitual es dividir el trabajo: simulación para obtener métricas de rendimiento generales, emulación para comprobar la validez de los comandos y, cuando es necesario, una fase intermedia con *hardware-in-the-loop* que introduce algunos enlaces físicos para confirmar que no surgen incidencias imprevistas.

Integración de IAG con simuladores

Los simuladores reproducen retardos y pérdidas con gran detalle, pero todavía dependen de que el ingeniero **decida qué probar y cómo parametrizarlo**. Incorporar una **RAG** rompe esa dinámica: el asistente consulta un índice de manuales y configuraciones validadas y compone propuestas listas para someterse a prueba. El laboratorio virtual arranca de esta manera con la experiencia acumulada en su memoria externa.

Cómo encaja la RAG en el bucle de simulación

1. El ingeniero plantea la **intención**: “Diseñar cuatro VLAN y permitir salida web solo al Departamento 1”.

2. El módulo de **recuperación** localiza comandos y plantillas pertinentes: creación de VLAN, subinterfaces *router-on-a-stick*, ejemplos de ACL http/https.

3. El **generador** redacta la configuración completa, cita su origen (p. ej. un fragmento de la *Cisco Configuration Guide*) y la entrega al simulador.

4. El simulador aplica los comandos, calcula latencia, pérdida y convergencia, y devuelve métricas que pueden alimentar una ronda de ajuste.

Gracias a este ciclo, la configuración llega **documentada y trazable**, y el tiempo de exploración se reduce, al bastar con modificar la intención para recibir una versión coherente con los nuevos requisitos y reensayarla al momento. Además, el modelo sintetiza patrones de tráfico realistas evitando que el usuario los defina a mano.

No obstante, la RAG arrastra **posibles alucinaciones** si el corpus es pobre y **sesgos** si el material favorece a un único fabricante. También debe producir la respuesta con la rapidez suficiente para no frenar el ciclo de pruebas. Los primeros despliegues solventan el problema almacenando en caché las plantillas más comunes y recalculando solo los parámetros que cambian entre iteraciones.

Algunos laboratorios ya inyectan la RAG dentro del propio simulador, pero en este proyecto se optará por un **acoplamiento menos intrusivo**. La IA generará la configuración en un paso separado y el simulador la consumirá de nuestra mano. Esa separación facilita medir cuánto tiempo se ahorra y qué precisión se gana **sin depender de extensiones propietarias**.

La tendencia apunta a la **convergencia**, y ya existen plugins capaces de transmitir configuraciones generadas al vuelo en GNS3 o EVE-NG y recoger el *syslog* resultante, mientras estudios académicos aplican la misma técnica a *slices* 5G en ns-3 (NetworkLessons.com, 2024). Cuando la generación sea lo bastante rápida y los índices estén bien curados, cada hipótesis – desde una ACL nueva hasta un cambio de métrica OSPF – viajará del editor a la métrica de rendimiento sin perder la referencia exacta de cada línea.

Vacíos de investigación y justificación de la propuesta

A pesar de los avances descritos, la unión entre **IA generativa, simulación y diseño de redes** todavía presenta lagunas que frenan su adopción industrial y académica.

Corpus técnico limitado y disperso

Los grandes modelos suelen entrenarse con texto generalista, y la terminología específica de *routing*, QoS o seguridad aparece de forma fragmentaria. Falta un **repositorio curado** que reúna manuales, RFC, *white papers* y *troubleshooting* reales, etiquetado de manera uniforme y actualizado con cada versión de firmware. Sin esa base, la RAG corre riesgo de alucinar comandos o de prescribir valores obsoletos.

Métricas para medir la utilidad real

Los trabajos existentes suelen valorar la IA por fluidez o por exactitud sintáctica. Rara vez se cuantifica **cuánto tiempo ahorra** al ingeniero, **qué porcentaje de la configuración generada acaba sin cambios** en producción o cuántas iteraciones de simulador se recortan. Tampoco se comparan, bajo un protocolo común, las propuestas de la IA con las de un experto humano en términos de latencia, uso de CPU o eficiencia energética.

Integración con el ciclo de pruebas

Los plugins que conectan generadores y simuladores inyectan comandos, pero no suelen cerrar el bucle: la **retroalimentación de métricas** (latencia, pérdidas, convergencia) vuelve de forma manual al diseñador. Falta un flujo estandarizado que entregue esos resultados a la RAG, depure la propuesta y relance la simulación hasta cumplir la intención declarada.

Explicabilidad y trazabilidad

Aunque la RAG puede citar una fuente, aún no existe un método claro para que el operador **inspeccione la cadena de razonamiento** completa (documento recuperado → fragmentos usados → tokens generados). Sin esa transparencia, la confianza en la propuesta automática se resiente, sobre todo en entornos críticos.

Huella energética del plano de control

Ejecutar inferencias continuas en un laboratorio grande o en producción implica coste eléctrico. Pocos estudios comparan el **gasto adicional del generador** con la energía que ahorra después la optimización propuesta, una métrica clave si la red aspira a credenciales de sostenibilidad.

Justificación del presente proyecto

1. Reunir un corpus de referencia esencial

Se recogerán **fragmentos relevantes para la topología docente**: creación de VLAN, *router-on-a-stick*, OSPF básico, ACL extendidas, etc. El fin es contar con

información coherente y actualizada, sin cubrir toda la biblioteca de RFC pero sí exigiendo al modelo cierta habilidad de recuperación, de modo que tenga materia prima suficiente y curada de donde escoger.

2. ***Levantar una RAG ligera y supervisada***

Con ese corpus se construirá un índice vectorial y se conectará a un **modelo Llama de pequeño tamaño**, sin afinado exhaustivo. El asistente devolverá plantillas plausibles y las respaldará con la referencia exacta; aún requerirá revisión humana, pero reducirá la búsqueda manual en documentación.

3. ***Volcar la propuesta al simulador de forma manual***

El flujo será deliberadamente sencillo: la RAG genera la configuración, el estudiante la revisa, la copia al simulador y ejecuta la prueba. No se automatizará la retroalimentación y los ajustes se harán a criterio del ingeniero.

4. ***Evaluar la calidad técnica de la salida***

La comparación se centrará en **qué tan correcta y completa** resulta la configuración sugerida frente a la escrita sin ayuda de IA. Para ello, ambos despliegues compartirán un *checklist* de puntos importantes a cubrir.

5. ***Registrar el razonamiento y las fuentes***

Cada comando aceptado irá acompañado de la cita de origen para demostrar trazabilidad y facilitar la revisión posterior, aun cuando la validación permanezca en manos del ejercicio.

Con este planteamiento se conserva la esencia del objetivo: demostrar que una RAG **eleva la calidad de la configuración, orienta al ingeniero con referencias verificables y abre el camino a automatizaciones más amplias en futuros desarrollos**.

METODOLOGÍA

Ecosistema de las IAs generativas

El mapa de los grandes modelos de lenguaje cambia casi cada trimestre, pero cuatro nombres concentran hoy la mayor parte de la investigación aplicada: **ChatGPT** de OpenAI, **DeepSeek** de la firma homónima, **Llama** de Meta y **Gemini** de Google DeepMind. Todos comparten la arquitectura Transformer, pero difieren en la escala del entrenamiento, en la política de acceso y, sobre todo, en las condiciones de despliegue, que determinan si un proyecto puede alojarlos en sus propios servidores o debe recurrir a un servicio externo.

ChatGPT (OpenAI)

Desde su lanzamiento en noviembre de 2022, **ChatGPT** se ha convertido en el referente popular de los grandes modelos de lenguaje. La versión **GPT-4** ofrece una capacidad de *context window* ampliada y un razonamiento estructurado que abarca desde la redacción de ensayos hasta la generación de fragmentos de código y guías de configuración. En el terreno que nos ocupa, su valor reside en la rapidez con la que propone comandos bien formateados, sugiere máscaras coherentes o desgrana las fases del desarrollo de un protocolo. Su entrenamiento con un corpus masivo de libros, artículos científicos y repositorios de software le confiere una **versatilidad notable**, de forma que entiende tanto una consulta coloquial como una instrucción CLI sin perder el hilo de la conversación.

La adopción empresarial se ha disparado gracias a la **API comercial** y a integraciones directas con plataformas de productividad, como GitHub Copilot para el desarrollo asistido. En los laboratorios de redes se utiliza como tutor que explica, por ejemplo, la diferencia entre un *router-on-a-stick* y una SVI, o detalla qué temporizadores cabría ajustar para acelerar la detección de fallos. Su otro punto fuerte es el ecosistema: abundan los *wrappers*, guías y bibliotecas que permiten orquestar conversaciones, almacenar historiales y automatizar pruebas con apenas unas líneas de Python. El modelo

se mantiene en la nube de OpenAI, lo que simplifica el acceso y descarga el peso de la inferencia en infraestructuras especializadas.

Ese **despliegue remoto** es, a la vez, su principal limitación cuando **los datos son sensibles** o la red está aislada de Internet. Para obtener la misma calidad de inferencia *on-prem* harían falta varias GPU de última generación, además de una licencia que aún no está abierta para todos los escenarios de uso local. Por otra parte, la amplitud del corpus puede jugar en contra en **dominios muy técnicos**: al no haber visto ejemplos suficientes sobre, digamos, arquitectura de red telefónica TDM, puede alucinar comandos o inventar sintaxis que compilan bien en texto, pero no en la consola. A eso se suma la falta de **trazabilidad fina**: aunque cite documentación, no es trivial extraer el fragmento exacto que motivó cada línea generada, algo imprescindible para procesos auditables en producción.

DeepSeek (DeepSeek Inc.)

DeepSeek irrumpió con la promesa de situarse entre los modelos de **código abierto** y los grandes servicios comerciales. Su variante de 67 mil millones de parámetros se distribuye con licencia flexible y acepta reajustes con volúmenes moderados de datos, permitiendo que un *fine-tuning* de unas pocas decenas de miles de líneas baste para especializarlo en telecomunicaciones o seguridad. El primer párrafo de mérito lo aporta la velocidad: con dos GPU de 24 GB es posible servir consultas interactivas, de modo que la inferencia puede residir dentro del laboratorio sin depender del perímetro corporativo.

En la práctica, DeepSeek se está incorporando a *miniclouds* que ensayan arquitecturas de campus o redes 5G. Un plugin experimental permite volcar la configuración propuesta directamente en GNS3 y capturar los *syslog* para análisis posterior. La empresa mantiene un repositorio con **ejemplos dirigidos a redes** como, por ejemplo, plantillas de BGP para ISP de tránsito o scripts de telemetría basados en gRPC, que facilitan la puesta en marcha a estudiantes y administradores. Además, la latencia

baja y el control sobre los pesos hacen viable realizar pruebas *what-if* sin exponer datos de topología a terceros.

Las reservas vienen por la **juventud del ecosistema**. La documentación oficial es más escueta que la de OpenAI, los foros acumulan menos soluciones y algunas bibliotecas (herramientas de *embeddings*, búsqueda semántica…) aún se etiquetan como **inestables**. El modelo grande, aunque corre localmente, necesita un *pipeline* cuidadoso de cuantización para evitar cuellos de botella, y el pequeño (7B) no siempre conserva la coherencia en diálogos largos. Por último, la cobertura temática fuera de los dominios principales puede resultar desigual al haber menos ejemplos en castellano o sobre normativa europea de telecomunicaciones, lo que obliga a reforzar el corpus con datos propios si se quiere evitar vacíos de conocimiento.

Llama (Meta)

Llama se concibe como un modelo de investigación abierto, con pesos disponibles en tamaños de **7, 8, 13, 34, 70 y hasta 405B**. El diseño prioriza la eficiencia y se ejecuta sin dificultad offline tras cuantización, lo que permite trabajar en **laboratorios modestos** o **redes aisladas**. Su **estructura modular** facilita la inyección de documentos técnicos: basta indexar manuales y guías sobre redes, protocolos y configuraciones para que el modelo recupere partes relevantes y las integre en la respuesta.

La comunidad académica y *open source* ha construido sobre Llama todo un **ecosistema de bibliotecas** – LangChain, llama-cpp, LlamaIndex – que simplifican la creación de RAG locales. En el contexto del diseño de redes, ello significa poder preguntar: “Genérame la configuración *router-on-a-stick* para cuatro VLAN con ACL que limiten al departamento 10 a http/https” y recibir una propuesta acompañada de la cita exacta del manual de IOS. La inferencia offline preserva la confidencialidad de topologías propietarias, algo que los responsables de seguridad valoran especialmente.

Frente a estos puntos fuertes, Llama carece de un servicio SaaS respaldado por SLA, y su **rendimiento sin afinamiento** específico es inferior al de GPT-4o en tareas

generales de redacción. Gestionar un *fine-tuning* requiere cierta destreza, pues implica construir el dataset, vigilar el *overfitting* y ajustar la cuantización para no perder precisión. Además, la **documentación oficial** se centra en la investigación, de modo que buena parte del conocimiento práctico se encuentra disperso en repositorios comunitarios. Quien adopte el modelo debe presupuestar tiempo para ensamblar herramientas y mantenerlas.

Gemini (Google DeepMind)

Gemini representa la **apuesta multimodal** de Google, un modelo que procesa texto, imágenes y datos estructurados dentro del mismo flujo de atención. En redes, eso permite analizar simultáneamente la topología en formato de diagrama, los logs y las tablas de rutas exportadas en CSV, todo dentro de la misma consulta. Su entrenamiento incorpora tareas de razonamiento multietapa, por lo que maneja dependencias largas y justifica los pasos intermedios, función súper útil cuando se depura un bucle de enrutamiento o se explica la prioridad de colas en un *scheduler*.

Google ofrece acceso a Gemini **a través de su nube** con hardware TPU, lo que asegura tiempos de respuesta consistentes y capacidad elástica para cargas altas. Además, la integración con Google Cloud facilita la extracción de datos recientes, métricas de latencia en tiempo real, registros de *Pub/Sub* o parámetros de Kubernetes que pueden entrar como contexto adicional. Esa combinación lo convierte en candidato para NOCs que monitorizan infraestructuras globales y necesitan correlacionar *streaming* de logs con configuraciones.

Su contrapartida es la **dificultad de un despliegue completamente local**: salvo la versión más liviana, los pesos completos no se distribuyen fuera del perímetro de Google. Al estar en fase piloto, la comunidad de soporte todavía es reducida y las herramientas de *fine-tuning* aún no abarcan los matices de privacidad que exigen algunas empresas. Para proyectos académicos o pymes, esa dependencia de la nube y el consumo de recursos pueden resultar disuasorios.

Justificación de la elección de Llama

Tras analizar las diversas propuestas disponibles en el ámbito de las IAs conversacionales, Llama se postula como la alternativa idónea para un proyecto enfocado en el **diseño y simulación de redes**. Existen tres factores centrales que refuerzan esta decisión. En primer lugar, el hecho de que la IA de Meta pueda operar eficientemente en entornos con recursos de hardware reducidos abre la puerta a un despliegue local, evitando la dependencia de servidores externos que suelen implicar **costes** y complejidades adicionales. En segundo lugar, su filosofía modular y orientada a la investigación permite un alto grado de **personalización**, lo que resulta esencial para parametrizar y desplegar la **RAG**. Finalmente, la infraestructura local posibilita un mayor control sobre la seguridad y la **privacidad de la información**, una cuestión trascendental cuando se trata de introducir datos internos en el proceso de entrenamiento y prueba de la IA.

En suma, **Llama** ofrece el mejor equilibrio entre autonomía de despliegue, facilidad de ajuste y ecosistema dedicado a la recuperación con citación, factores esenciales para el éxito de una RAG centrada en redes. Por ello, **se opta por este modelo** como la herramienta base con la que desarrollar la metodología planteada.

Cómo funciona Llama

Llama es un Transformer puro al que Meta ha eliminado todo lo superfluo: emplea ***rotary positional embeddings*** **(RoPE)** para codificar la posición dentro del mismo espacio que la atención y un vocabulario **SentencePiece** de 32000 subpalabras, de modo que identifica términos largos de redes, como *dot1q-tunnel* o *bgp-ls-id* sin inflar el número de tokens. Esa economía de representación permite ampliar la ventana hasta **8192 tokens** sin duplicar las tablas internas, algo crucial cuando se concatenan varios fragmentos de documentación y un historial de diálogos.

Meta publica los modelos en FP16, pero la comunidad los reempaqueta al formato **GGUF**, listo para cargar con llama.cpp.

Los pesos que Meta libera en FP16 garantizan la máxima fidelidad numérica, pero resultan poco prácticos fuera de un entorno de investigación. Un modelo de 8000 millones de parámetros ocupa más de 16 GB en crudo, exige una instalación completa de PyTorch y, al cargarse, duplica temporalmente la memoria porque primero se lee del disco y luego se vuelca en la RAM. Para sortear esas trabas, la comunidad convierte los ficheros al contenedor **GGUF**, formato nativo de llama.cpp. Durante el proceso se aplica una cuantización de modo que cada tensor queda comprimido y listo para ser mapeado directamente en el espacio de direcciones sin copias intermedias.

Esta adaptación trae tres beneficios inmediatos: el archivo final se reduce a una fracción del tamaño original, el motor C++ puede cargarlo en cuestión de segundos sin depender de bibliotecas externas pesadas y la memoria efectiva necesaria baja lo suficiente como para ejecutar la versión de 8B en una sola GPU de 6 GB junto a CPU. Así se preserva prácticamente toda la **precisión** y el modelo se vuelve usable en **laboratorios modestos** o en máquinas de producción que no pueden destinar decenas de gigabytes solo a un LLM.

En este proyecto se usan dos pesos distintos:

- **TinyLlama-1.1 B (Q4_K_M)** para generación de *embeddings*. El archivo ocupa unos 700 MB y corre en CPU, lo que libera la GPU para la fase creativa.
- **Llama-3-8 B-Instruct (Q3_K_L)** para la respuesta. El fichero ronda los 4 GB y se ejecuta cómodamente en una GPU de 6 GB.

El núcleo en C++ se expone a Python mediante la librería **llama_cpp**, empleada dos veces:

```
emb = Llama(model_path = EMB_PATH, n_ctx = 4096, n_threads = 8,
    embedding = True)
gen = Llama(model_path = GEN_PATH, n_ctx = 8192, n_threads = 8)
```

- **n_ctx** determina cuántos tokens máximos admite el prompt. 4096 bastan para la vectorización, mientras que 8192 dejan margen para extensos bloques de contexto más la respuesta.
- **n_threads** fija los hilos que usarán las GEMM internas, ocho es un buen punto de partida en CPU modernos.
- **embedding = True** desactiva la generación y devuelve un vector flotante que luego se normalizará con la pequeña función ensure_1d.

Los **hiperparámetros** fijados en la llamada al modelo generador determinan tanto el estilo como la seguridad de la salida (Rüepprich, 2024). A lo largo de varias iteraciones de estudio de respuesta del modelo, acabamos tendiendo a los valores que figuran en la **Tabla 2**:

Parámetro	Efecto práctico	Valor usado
`max_tokens`	Longitud máxima de la respuesta	600
`temperature`	Variedad léxica; a menor, más determinismo	0.15
`top_p`	Probabilidad acumulada en el muestreo	0.9
`repeat_penalty`	Penaliza bucles de palabras	1.1
`frequency_penalty / presence_penalty`	Desanima repeticiones y fomenta contenido nuevo	0.5 / 0.1
`stop`	Token de parada	`["</s>"]`

Tabla 2. *Hiperparámetros de llamada al modelo LLM.*

Con estos ajustes la generación se mantiene precisa y sin divagaciones: un flujo de 10-12 tokens/s deja la respuesta lista en cuestión de segundos.

El **flujo de trabajo**, a muy alto nivel, queda así:

1. **Embed** → la pregunta se vectoriza con TinyLlama.
2. **Prompt** → se compone el texto con un prompt base, los últimos turnos del historial y la pregunta.
3. **Generar** → Llama-3-8 B devuelve la respuesta, concisa y pertinente, con tablas, los comandos en back-ticks y, cuando procede, las citas (file:…).

4. **Log** → pregunta y respuesta quedan registradas en el historial a corto plazo.

Queda esquematizado en la **Figura 8**.

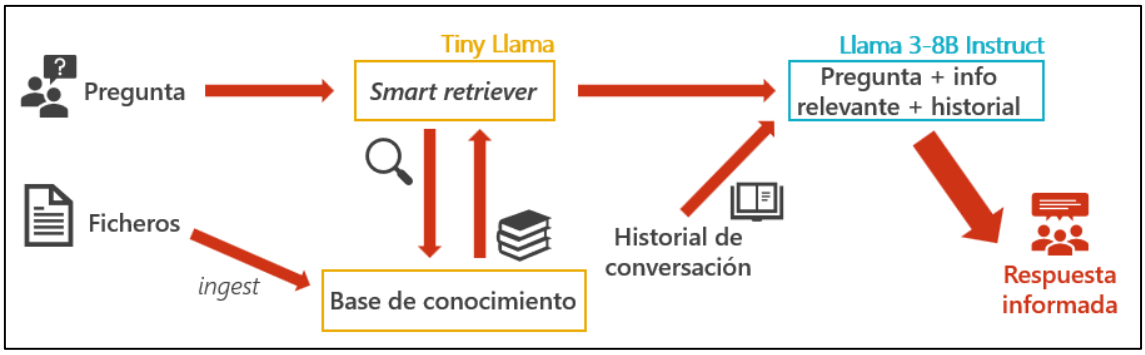


Figura 8. *Flujo de trabajo de la RAG.*

En cuanto a **GPU**, una tarjeta de **6 GB** maneja la generación junto a la CPU, y la carga efectiva disminuye porque varias capas cuantizadas pueden residir en la RAM del sistema sin penalizar la velocidad. Respecto a **latencia**, la obtención del *embedding* tarda menos de 50 ms y la generación típica (200-300 tokens) culmina en ~2 min, cifras cómodas para un uso interactivo de laboratorio.

En conjunto, estos componentes bastan para disponer de **un asistente local especializado** que admite una amplia ventana de contexto y equilibra coherencia con variedad, todo ello sin depender de servicios externos ni exigir hardware de gama extrema.

Herramientas de soporte para la RAG

Hugging Face

Hugging Face (Hugging Face, s. f.) funciona como un **repositorio público** donde se alojan modelos de aprendizaje automático listos para descargar, sobre todo los orientados al procesamiento del lenguaje natural, y se acompaña de un conjunto de librerías y utilidades mantenidas por una comunidad muy activa de desarrolladores e investigadores.

El recorrido de los pesos de **Llama** empieza en el *Hub* de Hugging Face, donde Meta publica las versiones de investigación y donde usuarios suben las variantes ya cuantizadas. Allí cada modelo viene acompañado de una *model card*: licencia, procedencia del corpus, pruebas de evaluación y advertencias de uso. Para descargarlos no basta con *git clone*, el Hub exige autenticar la petición con un **token de acceso**. El proceso es sencillo: tras crear una cuenta gratuita, se entra en *Settings* → *Access Tokens*, se genera un token *read* y se guarda como variable de entorno (HF_TOKEN) o se valida con *huggingface_hub login*. En la **Figura 9**, captura del token de lectura en HF.

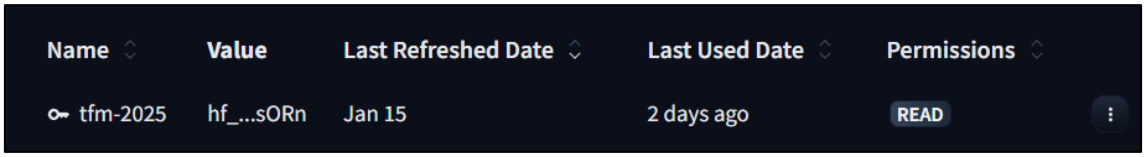


Figura 9. *Token de lectura generado en Hugging Face (Hugging Face, s. f.).*

Ese mismo identificador permite, desde el script, traer los ficheros a la carpeta models/ sin exponer credenciales en el código. En este proyecto se recurrió directamente a versiones **GGUF** que mantiene la comunidad.

Al estar ya cuantizados (Q3_K_L para el generador, Q4_K_M para el encoder) pueden cargarse con **llama.cpp** sin pasos intermedios de conversión, sin PyTorch y sin Git-LFS. El repositorio conserva la trazabilidad de firma SHA, fecha de subida y licencia, mientras que GGUF permite el *memory-mapping* directo.

ChromaDB

ChromaDB (ChromaDB Project, s. f.) nos servirá para ordenar la documentación técnica que alimenta la RAG. El motor se instala con un simple *pip install chromadb*, por defecto crea una **base SQLite** en la ruta que se le indique y, al abrirla con *chromadb.PersistentClient*, levanta el índice en memoria sin intervención manual. Cada fragmento de nuestra documentación base se pasa primero por el *encoder* TinyLlama para producir un **vector flotante**, y acto seguido **se registra en la colección** "docs" de ChromaDB junto con metadatos que facilitan la citación posterior. El script genera

identificadores estables y asegura la idempotencia borrando el id si ya existía antes de volver a añadirlo, de modo que las actualizaciones de la documentación no crean duplicados.

Internamente, ChromaDB enmarca esos vectores en un grafo **HNSW**. El resultado es que una consulta "segmentar cuatro departamentos en VLAN" se traduce en una **búsqueda por similitud coseno** que, incluso en CPU, devuelve los pasajes más cercanos en pocos milisegundos. Los metadatos viajan en el mismo registro, de forma que la RAG puede citar exactamente "(file: 01_vlan_basics.md)" al componer la respuesta. La base, al ser un único fichero .sqlite3, se versiona junto al código, se copia como cualquier **recurso estático** y revive intacta tras cada reinicio sin requerir servidores externos ni servicios de acompañamiento. Así el proyecto conserva en local tanto el saber del corpus como la indexación que lo hace recuperable, manteniendo la confidencialidad y reduciendo la latencia a niveles asumibles por el usuario. En la **Figura 10**, una captura de una sección de títulos de los *embeddings* en la base de datos generada.

	id	key	string_value
	Filtro	Filtro	Filtro
1	1	chroma:document	# VLAN básicas en Cisco IOS…
2	1	title	VLAN básicas en Cisco IOS
3	1	file	01_vlans_basico.md
4	1	chunk	*NULL*
5	2	file	01_vlans_basico.md
6	2	chunk	*NULL*
7	2	chroma:document	## 1 Crear una VLAN…
8	2	title	VLAN básicas en Cisco IOS > 1 Crear una VLAN
9	3	title	VLAN básicas en Cisco IOS > 2 Asignar puertos de acceso
10	3	chunk	*NULL*
11	3	chroma:document	## 2 Asignar puertos de acceso…
12	3	file	01_vlans_basico.md
13	4	title	VLAN básicas en Cisco IOS > 3 Configurar un puerto troncal
14	4	file	01_vlans_basico.md

Figura 10. *Captura de la base sqlite3 generada por ChromaDB.*

Software simulador de redes

La simulación es la antesala indispensable de cualquier despliegue: permite tensar la topología, forzar fallos y medir el tráfico antes de que la instalación empiece. Durante los últimos años ha proliferado un repertorio de herramientas que, sin perseguir las mismas metas, se complementan. **ns-3** ofrece un motor de eventos discretos orientado a la investigación; su código C++ expone cada paquete y cada temporizador, de modo que un doctorando puede experimentar con algoritmos de 5G o de IoT a nivel de bit y publicar resultados reproducibles (ns-3 Consortium, s. f.). **GNS3** se sitúa en el extremo opuesto y en lugar de modelar, emula imágenes de sistema operativo reales, lo que resulta perfecto cuando el objetivo es validar una sintaxis exacta o probar cómo reacciona un ASA ante un *failover*. Entre ambos se mueve **OMNeT++**, un marco académico muy flexible que empareja su núcleo de simulación con bibliotecas específicas como INET o Simu5G y que los grupos de investigación adoptan cuando quieren un término medio entre precisión y rapidez de desarrollo.

Aunque el catálogo no acaba ahí, la herramienta que domina la formación práctica es **Cisco Packet Tracer**. Cisco lo distribuye de forma gratuita para estudiantes a través de Networking Academy, y desde la versión 8.2 admite una ventana de contexto muy rica, permitiendo que el usuario puede arrastrar routers empresariales, puntos de acceso inalámbricos, sensores IoT o incluso un servidor Linux minimalista que responde a pings y a tráfico HTTP (Cisco Networking Academy, s. f.). Toda la interacción sucede en tiempo real, basta hacer doble clic en un dispositivo para que aparezca una consola idéntica a la del hardware real e introducir los mismos comandos *show*, *configure terminal* o *ip route*. La herramienta anima a **experimentar**: si el estudiante aplica un *no shutdown* al enlace equivocado, los LED virtuales se apagan y el tráfico virtual se detiene, reproduciendo al detalle la reacción que vería en un rack físico. Su interfaz gráfica es intuitiva y facilita su uso incluso para principiantes. En la **Figura 11** se observa una instancia del programa.

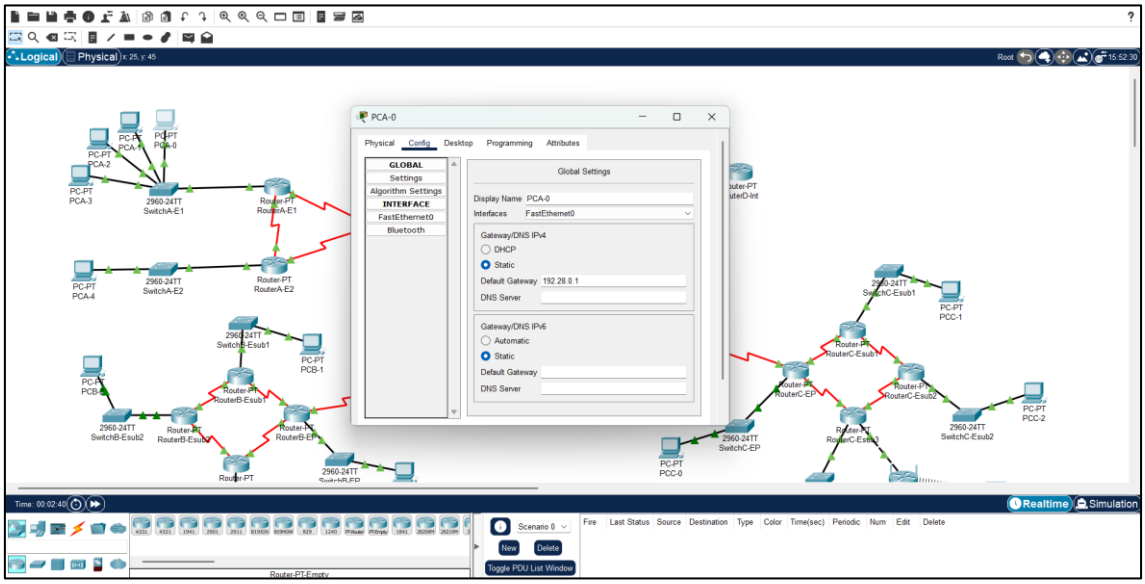


Figura 11. *Interfaz gráfica de Cisco Packet Tracer.*

Packet Tracer no se limita a la capa CLI. Cuando se selecciona el modo ***Simulation***, cada PDU recorre la topología paso a paso, con indicadores de color que distinguen solicitudes ARP, *hellos* de OSPF o secuencias TCP. El software permite pausar, adelantar o rebobinar el flujo, lo que facilita descubrir por qué un paquete se pierde en una VLAN mal etiquetada o por qué una red anunciada tarda más de lo previsto en propagarse. Esa **visibilidad detallada**, unida a la opción de inyectar errores, como romper un enlace, variar la latencia o degradar el ancho de banda, convierte a la plataforma en un aliado natural de la docencia.

Otra virtud es la **integración didáctica**. Las actividades *Packet Tracer Labs* llegan con instrucciones guiadas, verifican automáticamente cada ejercicio y reflejan la nota parcial en tiempo real. Quien prepara la certificación CCNA puede repetir un escenario hasta lograr un 100%. A la vez, el docente recibe un fichero de resultados que resume dónde falló cada participante. Para proyectos más abiertos, la versión 8.x incluye un área de scripting en Python que permite, por ejemplo, recopilar las tablas de enrutamiento en JSON o lanzar un lote de pings para calcular *jitter*.

No todo son ventajas. Packet Tracer modela solo el **ecosistema Cisco**, de modo que equipos de otros fabricantes o funciones muy recientes – una implementación de EVPN, un radio enlace propietario – quedan fuera o aparecen simplificados. Tampoco compite con la granularidad de ns-3 en estudios de investigación, el programa asume valores medios apropiados para la enseñanza. Sin embargo, para **topologías empresariales pequeñas y medianas** su nivel de realismo es suficiente, y la ligereza con la que se instala en un portátil compensa la falta de funcionalidad más avanzada.

Por los motivos expuestos, **este proyecto adopta Packet Tracer como banco de pruebas**. Su equilibrio entre fidelidad operativa y accesibilidad encaja con la filosofía de la RAG: el asistente propone comandos a partir de la documentación y el estudiante los pega en un entorno que reacciona de la misma manera que lo haría un switch físico, sin la inversión en hardware ni la curva de aprendizaje que exige una plataforma de emulación pura. Así, la investigación se concentra en la aportación real de la IA al ciclo de diseño, dejando a la herramienta de Cisco la tarea de validar que cada línea de configuración cumple lo prometido.

Cómo funciona Packet Tracer

Quien abre Packet Tracer por primera vez se topa con un lienzo en blanco y, en la bandeja inferior, una hilera de dispositivos: routers, switches, puntos de acceso, PCs, sensores IoT... Arrastrar un icono al área de trabajo basta para "desembalar" el chasis virtual, y un doble clic revela la consola CLI, en la que se pueden introducir los mismos comandos que se emplearían en un bastidor físico. Este gesto de "cablear con el ratón y configurar por consola" tiende un puente natural entre la teoría y la práctica.

Dos motores gobiernan la ejecución. En **Realtime** los dispositivos funcionan sin pausa: los *hellos* de OSPF salen cada diez segundos, los LEDs parpadean y el tráfico fluye tan rápido como lo permita el ordenador. Al conmutar a **Simulation**, el reloj se detiene y cada PDU se convierte en un evento con sello horario, lo que permite al diseñador avanzar paso a paso, inspeccionar encabezados o retroceder hasta averiguar por qué una trama viaja sin etiqueta. El panel lateral colorea ARP, ICMP, LSAs y etiquetas dot1q, de modo

que las rutas y los encapsulados se vuelven visibles sin recurrir a un analizador externo. Para ilustrar la diferencia entre ambos modos es útil observar la **Figura 12** y la **Figura 13**:

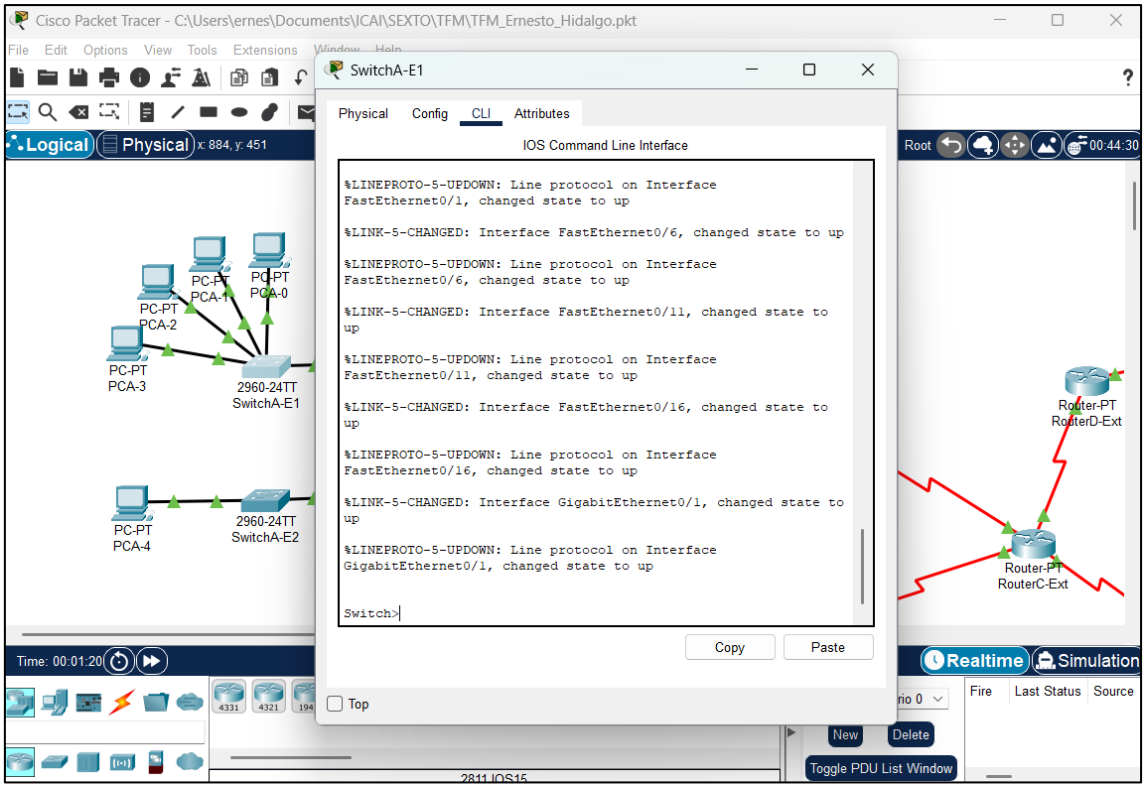


Figura 12. *Vista lógica en Realtime con la consola de un switch abierta.*

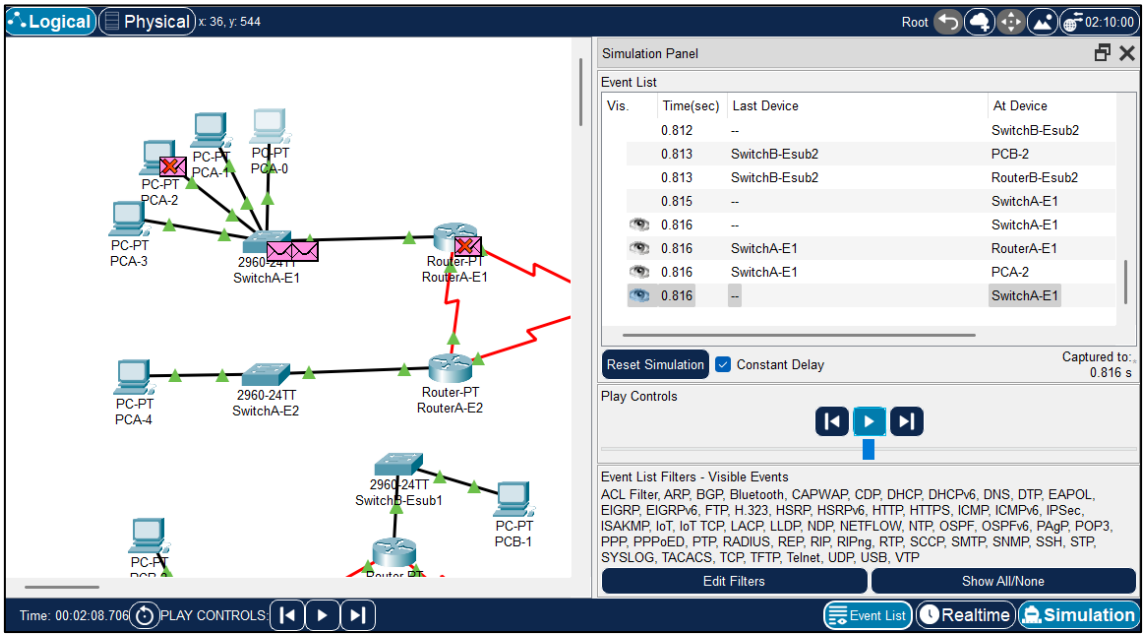


Figura 13. *Mismo escenario en Simulation, mostrando el Event List.*

La **creación de la red del caso de estudio** descansa sobre esa dualidad. La paleta permite colocar un switch de 24 puertos, conectar los PCs previstos y, en la pestaña *Physical*, arrastrar cables a los puertos FA0/1-FA0/20 (o configurarlos por interfaz gráfica). En la CLI se declaran las VLAN 10-13 y el enlace G0/1 se fija como troncal; un *show vlan brief* confirma el estado *active* para cada puerto. El router se incorpora y, tras habilitar subinterfaces fa0/0.10-13 con *encapsulation dot1q*, el motor Simulation deja ver la etiqueta de cuatro bytes que acompaña a cada trama. Las rutas OSPF se declaran en pocos comandos y, si el operador corta el serial entre RouterA-E1 y RouterA-E2, la lista de eventos muestra la reconvergencia salto a salto hasta que el PC de E2 vuelve a alcanzar los hosts de E1.

La pestaña **Physical** añade una representación de armarios y salas; resulta práctica cuando la distancia condiciona la elección del medio, cobre, fibra o radio, o cuando se quiere ilustrar cómo se propaga una avería en planta. Packet Tracer incluye además actividades guiadas, los *labs* corregidos en tiempo real de los que ya hablamos.

Una simulación en Packet Tracer **se considera fiable** cuando reproduce con rigor la **lógica de control**, el **tránsito de datos** y la **cronología de eventos**. Cada comando CLI modifica las tablas internas igual que lo haría el firmware real, y los paquetes virtuales atraviesan las mismas colas y enlaces lógicos, respetando retardos y anchos de banda configurados.

En conclusión, **Packet Tracer** fusiona la accesibilidad con la autenticidad de una consola IOS, añade un microscopio temporal para estudiar el tránsito y exige un hardware mínimo. Ese equilibrio lo convierte en la plataforma ideal para validar que la configuración sugerida por la RAG cumple lo prometido.

Procedimiento del proyecto (y diseño de la RAG)

Definición de los escenarios de red

El sistema bajo estudio, basado en apuntes académicos del docente Alejandro García San Luis (García San Luis, 2022), es un sistema autónomo **AS-A** subyacente a un esquema de red más complejo que escapa a este estudio. Nuestro dominio comparte un único plano de enrutamiento: todos los routers – internos y de acceso – ejecutan **OSPF área 0** y anuncian sus prefijos a través de enlaces punto-a-punto numerados en la red 10.10.0.0/22.

El esquema lógico es una estrella atenuada:

- ***RouterA-Ext*** conecta con la WAN.
- ***RouterA-Int*** hace de concentrador interno, y de él cuelgan *RouterA-E1* y *RouterA-E2*, cada uno responsable de una empresa distinta.

Sobre esa base se han fijado los requisitos funcionales y de seguridad que ambas implementaciones buscarán cumplir.

Plan de direccionamiento

El sistema se subdivide en dos empresas. La empresa 1 (E1) maneja 1024 direcciones públicas que se distribuyen uniformemente entre 4 departamentos, cada uno perteneciente a una VLAN. La empresa 2 (E2) tiene acceso directo desde su router principal a una LAN con 2048 direcciones IP.

Primero se eligen las redes y su reparto pertinente. A E1 se le destina la red 192.28.0.0/22, que, a su vez, se puede fragmentar como se indica en la **Tabla 3**.

Dirección de red	Máscara	VLAN	Puertos del switch
192.28.0.0	255.255.255.0	10	Fa0/1-5
192.28.1.0	255.255.255.0	11	Fa0/6-10
192.28.2.0	255.255.255.0	12	Fa0/11-15
192.28.3.0	255.255.255.0	13	Fa0/16-20

Tabla 3. *Dominios de red de la E1.*

Cada departamento recibe **256 direcciones públicas** y su propio dominio de difusión. El troncal **G0/1** entre el switch y *RouterA-E1* transporta las cuatro VLAN encapsuladas con 802.1Q.

En esta línea, se simularán algunos PCs tipo para cada departamento, los cuales se configurarán como procede en la **Tabla 4**.

Dirección IP	Gateway	Máscara	VLAN	Puerto
192.28.0.2	192.28.0.1	255.255.255.0	10	Fa0/1
192.28.1.2	192.28.1.1	255.255.255.0	11	Fa0/6
192.28.2.2	192.28.2.1	255.255.255.0	12	Fa0/11
192.28.3.2	192.28.3.1	255.255.255.0	13	Fa0/16

Tabla 4. *PCs de cada departamento de la E1.*

El diseño de la E2, que posee 2048 IPs, empieza en el segundo bloque de 2048 de la red, y se le asigna la máscara /21, figurando en la **Tabla 5**.

Dirección de red	Máscara
192.28.8.0	255.255.248.0

Tabla 5. *Dominios de red de la E2.*

Esta empresa no necesita segmentación interna, pues su router de borde atiende la LAN completa. Un PC4 tipo se configura como se indica en la **Tabla 6**:

Dirección IP	Gateway	Máscara	Puerto
192.28.8.2	192.28.8.1	255.255.248.0	Fa0/1

Tabla 6. *PC de la E2.*

Los routers requieren de ciertos ajustes entre sí. Se le asocia así a cada red formada por la conexión entre dos routers una dirección (ver **Tabla 7**):

Conexión	Dirección de red	Máscara
RouterA-E1 ⇔ RouterA-E2	10.10.0.0	255.255.255.0
RouterA-E1 ⇔ RouterA-Int	10.10.1.0	255.255.255.0
RouterA-E2 ⇔ RouterA-Int	10.10.2.0	255.255.255.0
RouterA-Int ⇔ RouterA-Ext	10.10.3.0	255.255.255.0

Tabla 7. *Redes entre routers.*

Requisitos de servicios y seguridad

- **Router-on-a-Stick**: RouterA-E1 aloja cuatro subinterfaces fa0/0.10-13, cada una con su puerta de enlace y encapsulation dot1q.

- **ACL SURFING**: en la subinterfaz fa0/0.10 se aplica una lista extendida que permite únicamente tráfico HTTP/HTTPS (puertos 80 y 443) hacia Internet; el resto del tráfico IP se permite dentro del AS.

- **Loopbacks /32**: cada router anuncia una interfaz de gestión (1.1.1.1, 2.2.2.2, 3.3.3.3, 4.4.4.4) para pruebas de alcance y como identificador estable en OSPF.

- **Ruta por defecto**: los routers de empresa apuntan a RouterA-Int y este, a su vez, delega la salida en RouterA-Ext.

Referencia topológica

La **Figura 14** ilustra la distribución física en Packet Tracer: los PCs de E1 conectados al switch 2960-24TT superior, la LAN de E2 al switch inferior y los cuatro

routers enlazados por seriales en el centro. El diagrama servirá de guía visual durante la fase de simulación y al comparar el diseño manual con la propuesta generada por la RAG.

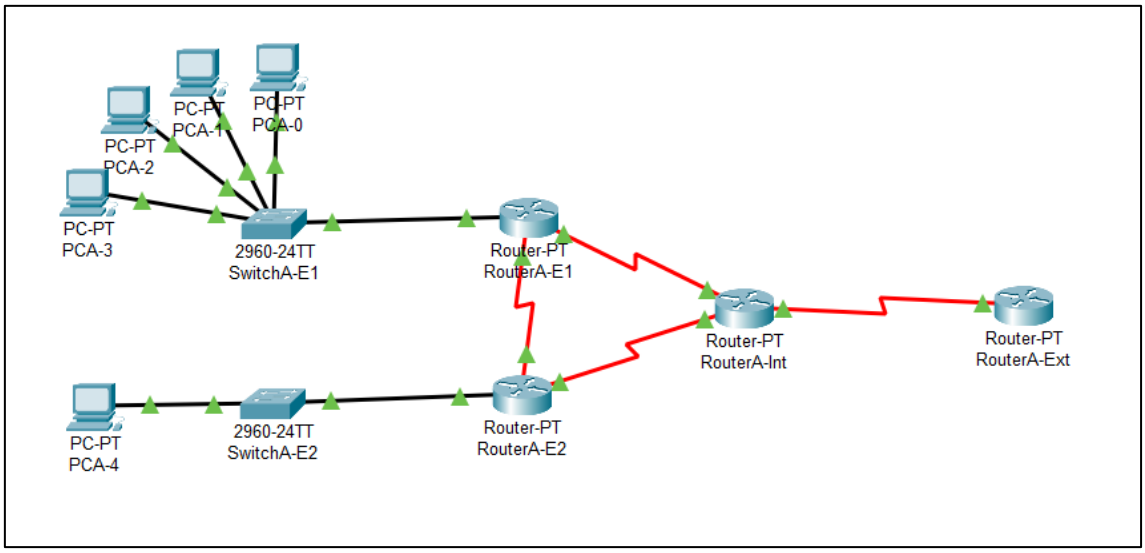


***Figura 14**. Arquitectura de red diseñada en Cisco Packet Tracer.*

Construcción de la RAG

Descarga y empaquetado de los modelos

El punto de partida fue **Hugging Face Hub**, donde se obtuvieron los dos pesos Llama que alimentan el sistema. La descarga se automatizó con *hf_hub_download*, autenticada mediante un *token* de solo lectura almacenado en la variable de entorno HF_TOKEN.

- **Generador**: *Meta-Llama-3-8B-Instruct-Q3_K_L* (repositorio *bartowski*). La variante Q3_K_L reduce los 16-bit originales a tres bits efectivos por parámetro; el fichero resultante ronda los 4 GB y cabe en una GPU de 6 GB, aunque se compartirán también recursos con CPU.

- **Codificador de *embeddings***: *TinyLlama-1.1 B-Chat-Q4_K_M* (mantenido por *TheBloke*). El archivo, de poco menos de 1 GB, se ejecuta íntegramente en CPU y libera la GPU para la generación.

Ambos modelos llegan ya en formato **GGUF**, de forma que basta copiarlos a la carpeta models/ y abrirlos con *llama.cpp*. Todo esto se describe también en Herramientas de soporte para la RAG.

```
# Modelo generador (8 B, Q3_K_L) – cortesía de bartowski
hf_hub_download(
    repo_id = "bartowski/Meta-Llama-3-8B-Instruct-GGUF",
    filename = "Meta-Llama-3-8B-Instruct-Q3_K_L.gguf"
)

# Modelo de embeddings (1.1 B, Q4_K_M) – cortesía de TheBloke
hf_hub_download(
    repo_id = "TheBloke/TinyLlama-1.1B-Chat-v0.3-GGUF",
    filename = "tinyllama-1.1b-chat-v0.3.Q4_K_M.gguf"
)
```

Curación del corpus y asignación de metadatos

Selección de contenidos

Para que el *retriever* se enfrentase a un contexto realista, la base documental se diseñó deliberadamente **más amplia que el caso de uso**: cuarenta y ocho fichas Markdown de entre 200 y 400 palabras cada una que cubren desde *RIP vs OSPF* hasta *VRRP* y *micro-segmentación ACI/NSX*, que se incluyen en el ANEXO III: Corpus de referencia para el entrenamiento del modelo. Todos los textos se redactaron con apoyo de GPT-4, contrastados contra la guía oficial de Cisco IOS y sus RFC equivalentes, y cada uno cita las fuentes originales al final.

La **Tabla 8** muestra el nombre, contenido y *keywords* de de la base de datos.

Nº	Nombre de archivo	Título	Etiquetas (tags)
01	vlans_basico.md	VLAN: concepto y creación	vlan, capa2
02	router_on_stick.md	Router-on-a-Stick (Inter-VLAN)	vlan, routing, inter-vlan
03	vlan_trunking_8021q.md	Enlaces troncales 802.1Q	vlan, trunk
04	subnetting_cidr.md	Subnetting & CIDR rápido	ip, subnetting
05	vlsm_plan.md	Diseño VLSM paso a paso	ip, subnetting, vlsm
06	dhcp_switch_router.md	DHCP en switches y routers	dhcp, capa2, capa3
07	ospf_single_area.md	OSPF área 0 (single-area)	ospf, routing
08	ospf_multi_area.md	OSPF multi-área & LSA	ospf, routing
09	rip_vs_ospf.md	RIP vs OSPF: cuándo usar	routing, comparison
10	static_routes.md	Rutas estáticas y default	routing, static
11	acl_standard.md	ACL estándar (nº 1-99)	acl, security

12	acl_extended.md	ACL extendidas y reflexivas	acl, security
13	nat_overload.md	PAT / NAT Overload	nat, security
14	bgp_intro.md	BGP: fundamentos y términos	bgp, routing
15	bgp_basic_cfg.md	BGP básico entre dos AS	bgp, routing
16	spanning_tree.md	STP y variantes (RSTP)	stp, capa2
17	etherchannel.md	EtherChannel (PAgP & LACP)	etherchannel, capa2
18	qos_basics.md	QoS: clasificación y colas	qos
19	ipv6_basics.md	IPv6 y SLAAC rápido	ipv6
20	ipv4_ipv6_dual.md	Dual-Stack IPv4/IPv6	ipv6, ipv4
21	loopbacks_utilidad.md	Interfaces loopback: usos	loopback
22	serial_links.md	Enlaces seriales & clock rate	serial, capa1
23	ospf_passive_int.md	OSPF + passive-interface	ospf, security
24	hsrp_basico.md	Alta disponibilidad con HSRP	ha, hsrp
25	vpn_ipsec_site2site.md	VPN IPsec site-to-site	vpn, security
26	microsegmentation.md	Microsegmentación con ACI/NSX	security, microsegmentation
27	backup_restore_ios.md	Respaldo y restauración de configuraciones IOS	maintenance, backup
28	netflow_monitor.md	NetFlow: captura y análisis de flujos	monitoring, netflow
29	aaa_radius_tacacs.md	AAA con RADIUS/TACACS+	security, aaa
30	best_practices_naming.md	Buenas prácticas de nomenclatura	best-practices
31	ip_plan_public_private.md	Planificación de direccionamiento IPv4 público/privado	ip, addressing
32	route_redistribution.md	Redistribución OSPF ↔ EIGRP	routing, redistribution
33	dmz_segmentation.md	Diseño de DMZ y segmentación de servidores	security, dmz
34	vrrp_basico.md	Alta disponibilidad con VRRP	ha, vrrp
35	gre_tunnel.md	Túneles GRE punto-a-punto	vpn, gre
36	firewall_baseline.md	Firewall en borde: reglas básicas	security, firewall
37	route_summarization.md	Summarización de rutas y optimización	routing, summarization
38	troubleshoot_ping.md	Guía rápida de troubleshooting	troubleshooting
39	logs_syslog.md	Syslog y niveles de logging	logging, monitoring
40	snmp_config.md	SNMPv3 configuración mínima	monitoring, snmp
41	energy_efficient.md	Energy Efficient Ethernet	eee
42	automation_ansible.md	Automatización con Ansible & Netmiko	automation, ansible
43	switch_security.md	Port-Security, BPDU Guard y DHCP Snooping	security, switch
44	campus_topologies.md	Topologías campus: anillo y malla	design, campus
45	load_balancer_l4_l7.md	Balanceo de carga L4/L7	ha, load-balancer
46	spine_leaf_arch.md	Arquitectura Spine-Leaf para data center	design, spine-leaf

47	ipv6_migration.md	Migración controlada de IPv4 → IPv6	ipv6, migration
48	diagrams_dynamic.md	Documentación y diagramas dinámicos de red	best-practices, diagrams

Tabla 8. *Base de datos de ingesta para la RAG.*

División por secciones

En lugar de cortar a longitud fija, práctica habitual en la creación de RAGs, se decidió **trocear por encabezados** # y ##. La razón: las fichas siguen un patrón homogéneo – definición, sintaxis, ejemplo, buenas prácticas – y cada bloque completo encaja cómodamente por debajo de 512 tokens. Preservar la unidad semántica aumenta la precisión del recuperador; en pruebas preliminares, la misma pregunta con *chunking* a 256 tokens ofrecía resultados más pobres.

Cuando se hace el *chunking*, se genera una tupla (title, chunk_text); el título adopta la forma **"H1 > H2"** si existe subtítulo y se guarda como **metadato**. Para un documento sobre *passive-interface* el título acaba siendo OSPF área 0 > passive-interface, lo que el usuario ve luego en la cita (file: ospf_passive_int.md). Esto facilita la búsqueda de información relevante al dar de cada sección su contexto superior.

```
# ———————— INICIALIZA MODELO + CHROMA ————————
ing        = Llama(model_path=str(EMB), n_ctx=4096,
    embedding=True, verbose=False)
client     = chromadb.PersistentClient(path=str(DB))
collection = client.get_or_create_collection("docs")

# ———————— UTILES ————————
def ensure_1d(vec) -> List[float]:
    """Garantiza lista plana de floats."""
    assert vec, "embedding vacío"
    if isinstance(vec, list) and isinstance(vec[0], list):
        vec = vec[0]
    return [float(x) for x in vec]

# ———————— CHUNKING POR ENCABEZADOS ————————
def chunk_by_heading(text: str):
    """
    Devuelve lista de (title, chunk_text) donde:
        - title = H1              para la intro y bloques H1
        - title = "H1 > H2"       para subsecciones H2
    """
    chunks, buf = [], []
    h1 = "intro"
    title = h1
```

```
        def flush():
            if buf:
                chunks.append((title, "\n".join(buf).strip()))
                buf.clear()

        for line in text.splitlines():
            if line.startswith("# "):          # H1
                flush()
                h1 = line.lstrip("# ").strip() or "section"
                title = h1
            elif line.startswith("## "):       # H2
                flush()
                h2 = line.lstrip("# ").strip() or "subsection"
                title = f"{h1} > {h2}"
            buf.append(line)
        flush()
        return chunks
```

La tabla de índice asigna a cada archivo un conjunto de ***keywords***. El CSV file_keywords.csv se carga al vuelo, y el script añade esas claves al diccionario metadata.

```
# ———————— CARGA KEYWORDS ————————
def load_file_keywords(csv_path: Path) -> Dict[str, List[str]]:
    mapping: Dict[str, List[str]] = {}
    with open(csv_path, newline='', encoding='utf-8') as f:
        reader = csv.DictReader(f)
        for row in reader:
            fn = row.get('filename', '').strip()
            kws_raw = str(row.get('keywords', '')).strip()
            kws = [k.strip() for k in kws_raw.split(',') if k.strip()]
            if fn:
                mapping[fn] = kws
    return mapping
file2keywords = load_file_keywords(KW_CSV)
```

Ingesta vectorial con TinyLlama + ChromaDB

Una vez limpio y troceado, cada fragmento se convierte en un vector de **4096 dimensiones** invocando al **primer modelo** ing.embed(text). El *embedder* se abre con n_ctx=4096 para evitar recortes de contexto y con embedding=True para desactivar la capa generativa.

Los vectores y sus metadatos se añaden a la colección docs en **ChromaDB**, que usa SQLite como backend y HNSW como índice.

```
# ———————— INGESTA PRINCIPAL ————————
print(f"Procesando Markdown en {DATA} …\n")
total = 0

for md_file in sorted(DATA.glob("*.md")):
```

```
        content = md_file.read_text(encoding="utf-8", errors="ignore")
        sections = chunk_by_heading(content)

        # Obtiene keywords para este archivo
        kws = file2keywords.get(md_file.name, [])

        for idx, (title, chunk) in enumerate(sections, start=1):
            doc_id = f"{md_file.stem}__{idx}"
            vec    = ensure_1d(ing.embed(chunk))

            metadata = {
                "file":     md_file.name,
                "chunk":    idx,
                "title":    title,
                "keywords": ",".join(kws),
            }

            collection.delete(ids=[doc_id])          # idempotente
            collection.add(
                ids        =[doc_id],
                documents  =[chunk],
                embeddings =[vec],
                metadatas  =[metadata],
            )
            total += 1
        print(f"✓ {md_file.name:30s}  → {len(sections)} secciones")
    print(f"\nIngesta finalizada: {total} embeddings almacenados en 'docs'.")
```

En la **Figura 15** se ve un print parcial de la ingesta por terminal, incluido en código para el debugging del proceso.

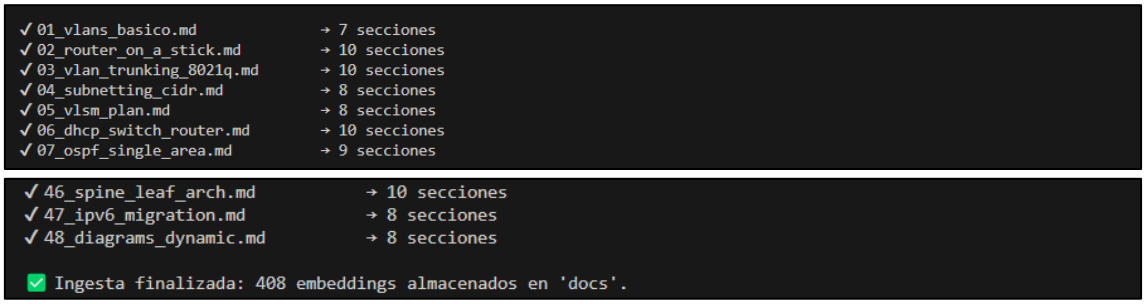


Figura 15. *print de la ingesta de datos por el primer modelo.*

En la **Figura 10** de Herramientas de soporte para la RAG se vio una pequeña sección de la base de datos sqlite3 formada por la ingesta en ChromaDB.

Diseño del prompt y memoria a corto plazo

Cada consulta nace de un esqueleto fijo, **BASE_PROMPT** en el código, que dicta tono ("Eres un ingeniero de redes senior"), formato (comandos en back-ticks) y, sobre todo, concisión: **"Responde solo con el CONTEXTO; si no sabes di 'No lo sé'"**.

A ese bloque se le añaden:

1. **Historial de los últimos cuatro turnos** para sostener diálogos encadenados ("ahora configura la ACL").

2. **Ocho fragmentos RAG** (ordenados por distancia en el vector-store), cada uno precedido de su número y seguido de la cita. Los fragmentos son los considerados más relevantes, y se seleccionan a partir de la entrada y con ayuda de un **segundo modelo** de especificaciones similares al primero. Así, la entrada se hace *embedding* para la **comparación** vectorial con los que figuran en la base de datos.

Inferencia con Llama-3-8 B

La generación de la respuesta nace de invocar al tercer modelo con el prompt completo como input, y con los hiperparámetros de la **Tabla 2**, sección Cómo funciona Llama. Esos valores, afinados tras una docena de **pruebas ciegas**, logran un **equilibrio entre precisión** CLI (poco *creative drift*) y la pequeña **variabilidad** que evita respuestas calcadas cuando se pide lo mismo dos veces. En la GPU de 6 GB + CPU la velocidad se mantiene entre **40 y 48 tokens por segundo**, de manera que una respuesta típica (250-300 tokens) llega en unos dos minutos.

Cada pareja pregunta-respuesta se guarda en el **historial**, aportando contexto a las siguientes interacciones y facilitando la reproducción de defectos y revisión de otros fallos.

```
# ——— Configuración ———
K_RETRIEVE     = 8
MAX_TOKENS     = 600
TEMPERATURE    = 0.15
TOP_P          = 0.9
REPEAT_PENALTY     = 1.1
FREQUENCY_PENALTY = 0.5
```

```
PRESENCE_PENALTY  = 0.1
STOP_TOKENS       = ["</s>"]

# ——————— Carga modelos ———————
print("Cargando modelos…", flush=True)
emb = Llama(model_path = str(EMB_PATH), n_ctx = 4096, n_threads = 8,
    embedding = True, verbose = False)
gen = Llama(model_path = str(GEN_PATH), n_ctx = 8192, n_threads = 8,
    verbose = False)

# ——————— vector-store ———————
try:
    client = chromadb.PersistentClient(path=str(DB_PATH))
    col    = client.get_collection("docs")
    total  = col.count()
except Exception as e:
    sys.exit(f"Error conectando a la base de datos: {e}")

if total == 0:
    sys.exit("La colección 'docs' está vacía. Ejecuta primero ingest.py")

def ensure_1d(vec) -> List[float]:
    """
    Aplana recursivamente cualquier nivel de anidación
    y convierte cada elemento a float.
    """
    while isinstance(vec, list) and vec and isinstance(vec[0], list):
    vec = vec[0]
    return [float(x) for x in vec]

# ——————— Prompt base ———————
BASE_PROMPT = textwrap.dedent("""\
    Eres un ingeniero de redes senior.
    Contesta **directamente** la PREGUNTA que se te hace.
    Responde **solo** con la información presente en el CONTEXTO.
    No formules preguntas de seguimiento ni generes múltiples Q&A:
contesta solamente la pregunta que te hago.
    Formatea los comandos en back-ticks (`like this`).
    Cita tu fuente así: (file: <nombre>).
    Si no hay contexto suficiente, responde exactamente "No lo sé."
    """)

def retrieve(query: str, k: int = K_RETRIEVE) -> str:
    """
    Recupera los k documentos más relevantes e incluye metadatos
    para que se puedan citar automáticamente.
    """
    q_emb = ensure_1d(emb.embed(input=[query]))
    res = col.query(
        query_embeddings=[q_emb],
        n_results=k,
        include=["documents", "metadatas"]
    )

    docs      = (res.get("documents") or [[]])[0]
    metadatas = (res.get("metadatas")  or [[]])[0]

    parts: List[str] = []
    for i, (doc, meta) in enumerate(zip(docs, metadatas), start=1):
        fname = meta.get("title", "desconocido")
        chunk = meta.get("chunk", meta.get("section", i))
```

```
                parts.append(
                    f"{i}. {doc}\n(file: {fname})"
                )
            return "\n---\n".join(parts)

        print(f"{total} secciones disponibles. Escribe tu pregunta (exit para
salir).")

        history: List[Tuple[str,str]] = []  # lista de (role, text)

        while True:
            q = input("❯ ").strip()
            if not q or q.lower() in {"exit", "quit", "salir"}:
                break

            # 1) Recupera contexto RAG
            ctx = retrieve(q)

            # 2) Añade turno de usuario al historial
            history.append(("user", q))

            # 3) Construye prompt: historial + contexto + pregunta
            prompt_parts = [BASE_PROMPT, ""]
            if len(history) > 1:
                prompt_parts.append("HISTORIAL:")
                for role, txt in history[-4:]:
                    tag = "Usuario" if role == "user" else "Asistente"
                    prompt_parts.append(f"{tag}: {txt}")
                prompt_parts.append("")

            if ctx:
                prompt_parts.append("CONTEXTO RAG:")
                prompt_parts.append(ctx)
                prompt_parts.append("")

            prompt_parts.append(f"PREGUNTA: {q}")
            prompt_parts.append("RESPUESTA:")
            full_prompt = "\n".join(prompt_parts)

            # 4) Llama al LLM
            raw = gen.create_completion(
                prompt=full_prompt,
                max_tokens=MAX_TOKENS,
                temperature=TEMPERATURE,
                top_p=TOP_P,
                repeat_penalty=REPEAT_PENALTY,
                frequency_penalty=FREQUENCY_PENALTY,
                presence_penalty=PRESENCE_PENALTY,
                stop=STOP_TOKENS,
                stream=False,
            )

            out = cast(CreateCompletionResponse, raw)
            ans = out["choices"][0]["text"].strip() or "No lo sé."

            # 5) Imprime y guarda en historial
            print("\n" + ans + "\n")
            history.append(("assistant", ans))
```

Comprobaciones de calidad

Al concluir la primera ingesta se pasaron tres tests automáticos:

- **Recall sintáctico**: Se lanzaron varios prompts al azar ("añade loopback", "ACL extended pantalla web") y se midió si la respuesta contenía las palabras clave (ip access-list, interface loopback) dentro de los primeros diez tokens.
- **Integridad de cita**: El script verificó que cada bloque devuelto terminara con una cadena (file: …). 100% de cumplimiento tras una corrección menor.
- **Tiempo total de ciclo**: Vectorizar + consultar + generar no debería superar el minuto en la máquina destino, una línea de base razonable para trabajo interactivo de laboratorio.

Diseño manual de referencia

El punto de partida consiste en que el propio ingeniero, sin ningún tipo de ayuda algorítmica, **construya la topología** doble descrita en la metodología: la empresa 1 con cuatro VLAN y mil veinticuatro direcciones públicas, la empresa 2 con su bloque de dos mil cuarenta y ocho direcciones y la malla de enlaces que une ambos dominios a través de un núcleo interno y un punto de salida exterior. La tarea arranca en Packet Tracer. Primero se arrastran los dispositivos que se corresponden con el inventario teórico: un **switch de 24 bocas** para cada sede, cuatro **routers** (E1, E2, Interior y Exterior) y los **PC** que ejercerán de generadores de tráfico. Se tienden los cables, se etiquetan los puertos y, ya en la consola del switch de E1, se crean las cuatro VLAN – 10, 11, 12 y 13 – seguido de la conversión del enlace **G0/1** en troncal dot1q.

A continuación, el ingeniero afronta el capítulo de direccionamiento. El rango 192.28.0.0 /22 se parte a mano en cuatro subredes clase C, se apunta cada puerta de enlace y se anota en una tabla provisional junto al puerto físico que le corresponde. Esa tabla se transcribe, línea a línea, en la CLI: cada subinterfaz de **fa0/0** recibe su etiqueta dot1q y su IP. Para la empresa 2 se reserva el bloque 192.28.8.0 /21 y se repite el procedimiento, esta vez sin VLAN, porque el requisito marca una única LAN. Una vez fijados los prefijos

locales, se documentan los enlaces punto a punto de la retícula 10.10.x.0 /24, se configura cada serial y se hace un *save* en la NVRAM virtual para congelar ese estado como referencia.

El paso siguiente es activar **OSPF área 0** en los cuatro routers. El profesional introduce las redes con su correspondiente *wildcard mask*, observa cómo se forman las adyacencias en Realtime y, cuando el LED del último enlace pasa a verde, conmuta a Simulation para asegurarse de que los *hellos* y LSAs siguen un ritmo adecuado. Cada evento se copia al *Event List* y se guarda como instantánea, material que luego servirá para medir la convergencia cuando se compare con la configuración generada por la RAG.

Una vez la conectividad básica parece estable, llega el turno de la política. El departamento de contabilidad (VLAN 10) solo puede salir a la web por puertos 80 y 443. El ingeniero redacta la ACL **SURFING** siguiendo la sintaxis canónica, la aplica a la subinterfaz correspondiente y, en Simulation, lanza un flujo HTTP y otro ICMP para verificar que uno pasa y el otro se descarta. Los resultados se anotan in situ en un *logbook* junto al comando *show access-list*, de forma que el único material pendiente de extraer sean las cifras.

Conectividad verificada, políticas aplicadas y tabla de rutas estable, el diseñador realiza **capturas base**. En una ventana de terminal separada ejecuta un lote de *pings* y llamadas a comandos que servirán para el propósito de la sección Ejecución de la simulación y análisis comparativo. Por último, también se guarda un archivo .pkt con fecha y hora para que la instantánea pueda reabrirse, sin alteraciones, cuando toque contrastarla con la versión asistida.

Este diseño manual queda fijado como el **patrón de oro** que servirá para juzgar más adelante si la configuración propuesta por la RAG cubre los mismos requisitos, alcanza la misma convergencia y respeta las mismas restricciones de acceso sin intervención humana adicional.

Diseño asistido por RAG

En la segunda ronda el trabajo se redistribuye: el ingeniero sigue al mando del diagrama lógico, pero la **RAG** le aligera las tareas más repetitivas, entre las cuales se hallarían buscar la sintaxis exacta, comprobar máscaras, recordar el orden de los comandos, etc. En este punto, la RAG ya está operativa, con su corpus vectorizado y motor de inferencia activo, y el diseñador se limita a **pedir piezas sueltas de configuración** en lenguaje casi coloquial para ensamblarlas en Packet Tracer. El modelo no conoce "Empresa 1" ni "Empresa 2", solo sabe qué aspecto tiene una VLAN, cómo se declara un enlace troncado o qué máscara corresponde a un /21. Por eso la tarea se divide en microencargos que encajan uno a uno con los capítulos indexados.

La **Tabla 9** muestra **algunos ejemplos representativos**. No es exhaustiva, pero ilustra el patrón de trabajo:

Objetivo puntual	Ejemplo de prompt	Fragmentos del corpus
Crear cuatro VLAN y asignar puertos	"Declara las VLAN 10-13 y pon cinco puertos de acceso en cada una"	`vlans_basico.md`
Explicar qué es una VLAN	"En una frase, define VLAN y da dos buenas prácticas para nombrarlas"	`vlan_basico.md`, `best_practices_naming.md`
Configurar el router-on-a-stick	"Genera subinterfaces dot1q para las VLAN 10-13 sobre fa0/0 con sus puertas de enlace /24"	`router_on_a_stick.md`, `vlan_trunking_8021q.md`
Calcular máscara y puerta de enlace de la LAN /21	"Para 192.28.8.0 haz /21 y dame la IP de Gateway"	`subnetting_cidr.md`, `ip_plan_public_private.md`
Activar OSPF área 0	"Activa OSPF en todos los enlaces 10.10.x.x/24 y anuncia las redes de las VLAN"	`ospf_single_area.md`, `ospf_passive_int.md`
Limitar la VLAN 10 a HTTP/HTTPS	"ACL extendida: solo TCP 80 y 443 desde 192.28.0.0/24 a cualquier destino"	`acl_extended.md`

Tabla 9. *Ejemplos de prompt contra el RAG para el diseño de la red.*

Flujo operativo

1. **Consulta y citación**: Cada prompt genera información y bloques CLI envueltos en back-ticks, y termina con referencias (file: …). El ingeniero revisa que las citas procedan de capítulos coherentes (por ejemplo, que no combine ejemplos de RIP con OSPF).

2. **Pegado por secciones**: Los bloques llegan en el orden que el usuario pidió, de modo que basta copiar la sección VLAN al switch, la sección dot1q al router y así sucesivamente. Si el modelo omite un *no shutdown* o usa la interfaz Gig en vez de GigabitEthernet, se corrige in situ y se anota la intervención para el *checklist*.

3. **Verificación mínima**: *show vlan brief*, *show ip route ospf* y un par de pings aseguran que la topología viva. Cualquier ajuste adicional (un wildcard de OSPF mal escrito, un puerto trunk con encapsulation isl…) se registra en un *diff* que luego contará como edición manual.

4. **Congelación del escenario**: Una vez la red responde, se guarda la instantánea .pkt con sufijo **_RAG** y se pasa al plan de simulación idéntico al usado con el diseño puro. De esa forma la comparación de métricas dependerá solo de la calidad de la configuración, no de diferencias en la puesta en escena.

Plan de verificación por simulación

En este proyecto la **validación funcional** de ambas maquetas, la elaborada íntegramente a mano y la generada con ayuda de la RAG, se rige por el mismo guion: una ***checklist* de dieciséis pruebas** que figura en la sección Plantillas de evaluación y cubre conectividad, rutas, filtrado y consistencia sintáctica.

El procedimiento, común a ambos diseños, se resume así:

1. **Ejecución secuencial** de los 16 ítems de la lista. Cada punto describe un paso concreto (p. ej. “ping desde VLAN 11 a 192.28.8.2” o “mostrar tabla OSPF en RouterA-Ext”) y la salida que debe observarse en la versión de referencia.

2. Para la maqueta asistida por IA se **marca** cada test como ✓, ½ o ✗ frente al resultado perfecto obtenido con el diseño manual, que actúa de patrón oro.

3. Una vez recorrida la lista se **calcula el porcentaje de acierto** de la RAG (sumando 1, 0,5 o 0 por casilla) y se anota la puntuación junto con las correcciones aplicadas.

El enfoque aporta dos ventajas. Primero, la **paridad de escenarios**: ambos diseños se someten exactamente a la misma batería de comandos, lo que elimina sesgos de operador. Segundo, la **cuantificación objetiva**: el listado de 16 controles traduce una maqueta compleja en una métrica simple (× / 16) y deja claro, de un vistazo, dónde la IA ya es fiable y dónde sigue necesitando supervisión.

Bajo este marco la versión generada por la RAG alcanzó un **63% de cumplimiento** tras una única pasada de correcciones mínimas, dato que se discutirá en la sección de resultados como prueba del ahorro potencial y de los límites actuales del enfoque asistido.

Plantillas de evaluación

El *checklist* que sigue funciona como una plantilla sencilla pero rigurosa: **condensa en casillas todo lo que la red debe ofrecer**, desde que las VLAN existen y tienen sus puertos, hasta que un *ping* de prueba atraviesa OSPF y la ACL solo deja pasar HTTP/HTTPS, sin desviarse de la configuración real escrita en Packet Tracer. La idea es que el evaluador pueda imprimir la hoja, abrir primero la topología manual y luego la RAG, e ir marcando **✓, ½ o ✗** sin detenerse en largas interpretaciones. Cada casilla refleja un comando o una prueba puntual de modo que basta mirar la consola o el panel de eventos para decidir el resultado.

Una vez rellenadas las dos columnas, se suman los puntos – 1 por Cumplido, 0,5 por Parcial, 0 por Fallido – y se obtiene un **porcentaje de acierto** para cada diseño. Esa cifra no pretende medir rendimiento, sino **cobertura funcional**: indica cuánta configuración ha salido correcta a la primera y cuánta edición manual ha sido necesaria.

La comparación de porcentajes, junto con el tiempo invertido y las observaciones cualitativas, permitirá valorar con fundamento el **aporte real de la RAG** frente al trabajo tradicional.

(1) [ ] VLAN 10-13 creadas y nombradas
(2) [ ] Puertos fa0/1-20 asignados a VLAN correctas
(3) [ ] `interface g0/1` declarado trunk dot1q
(4) [ ] Subinterfaces fa0/0.10-13 con IP /24 y `encapsulation dot1q`
(5) [ ] LAN E2 configurada en fa0/0 con IP 192.28.8.1/21
(6) [ ] Enlaces seriales 10.10.x.x/24 con .1 / .2 coherentes
(7) [ ] `router ospf 1` + redes VLAN y seriales / declaración loopbacks
(8) [ ] Ruta estática `0.0.0.0/0` presente donde procede
(9) [ ] ACL SURFING con solo 80/443 y aplicada a fa0/0.10 (in)
(10) [ ] Interfaces no shutdown y sin errores de sintaxis
(11) [ ] Comentarios '`!`' que identifiquen cada bloque (deseable)
(12) [ ] *Ping* VLAN 11 → 192.28.8.2 responde
(13) [ ] `telnet 8.8.8.8` desde VLAN 10 bloqueado por ACL
(14) [ ] Trama VLAN 12 inspeccionada: ida y vuelta con tag dot1q
(15) [ ] Reconvergencia OSPF tras 30 s de arranque (FULL/DR)
(16) [ ] Configuración guardada, snapshot *.pkt* y log de consola exportados

Cronograma de trabajo

Se siguió el cronograma de trabajo representado en la **Figura 16**.

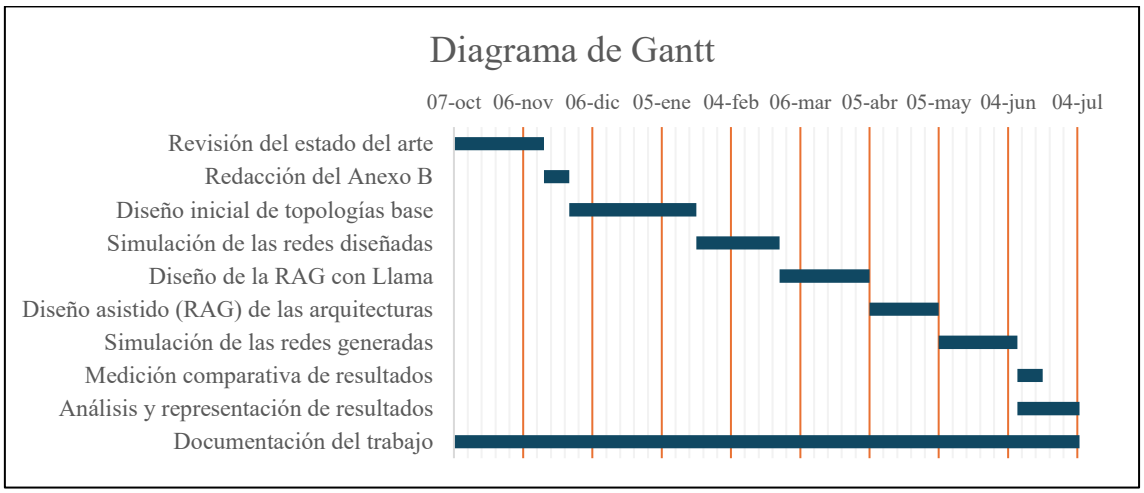


Figura 16. *Diagrama de Gantt seguido en la realización del proyecto.*

RESULTADOS

Visión de conjunto

Objetivo de las pruebas

El presente bloque muestra, paso a paso, **cómo se comportan dos diseños equivalentes pero obtenidos por caminos distintos**. Por una parte, el trazado **artesanal** elaborado a mano siguiendo las tablas de la sección de escenarios y, por otra, la variante generada con la **RAG**, a la que se suministraron prompts breves para cada pieza de configuración.

Los ensayos persiguen tres metas concretas:

1. **Cobertura funcional.** Comprobar que ambos diseños satisfacen los requisitos clave (segmentación en VLAN, enrutamiento OSPF, filtrado HTTP/HTTPS y conectividad end-to-end).

2. **Esfuerzo de edición.** Medir cuántas operaciones necesitan corrección o añadido antes de que la topología arranque sin errores.

3. **Trazabilidad.** Verificar que cada comando que propone la RAG está respaldado por una cita clara del corpus, de forma que el diseñador pueda auditar la fuente con un clic.

Con esos tres indicadores se pretende responder a la pregunta central del TFM: ***¿hasta qué punto una RAG modesta acelera las tareas repetitivas de configuración sin sacrificar rigor?***

Entorno de ejecución

Todos los experimentos se realizaron en una estación de trabajo **Windows 11 Home 24H2** equipada con **32 GB DDR4 RAM** y **6 GB VRAM**.

Software y versiones relevantes en la **Tabla 10**:

Componente	Versión	Observaciones
Cisco Packet Tracer	8.2.2 (64-bit)	-
llama.cpp	0.3.9	Compilado con BLAS-AVX2; backend GPU cuBLAS
Generador (LLM)	*Meta-Llama-3-8B-Instruct-Q3_K_L.gguf*	Modelo cuantizado Q3
Embeddings	*tinyllama-1.1B-Chat-Q4_K_M.gguf*	Cuantizado Q4
ChromaDB	1.0.13	Backend SQLite, colección **docs** (408 secciones)
Python	3.13.15	Entorno virtual aislado; dependencias fijadas en `requirements.txt`

Tabla 10. *Software y versiones relevantes.*

Diseño manual: instantánea inicial

Configuración esencial del diseño

La memoria recoge los bloques que dan vida a la red. A continuación, se muestran los fragmentos más representativos:

- **Creación de VLANs y asignación de los puertos** de acceso en el switch de E1.

```
Switch(config)# vlan 10
Switch(config-vlan)# exit
Switch(config)# vlan 11
Switch(config-vlan)# exit
Switch(config)# vlan 12
Switch(config-vlan)# exit
Switch(config)# vlan 13
Switch(config-vlan)# exit

Switch(config)# interface range fa0/1-5
Switch(config-if-range)# switchport mode access
Switch(config-if-range)# switchport access vlan 10
Switch(config-if-range)# no shutdown
Switch(config-if-range)# exit

Switch(config)# interface range fa0/6-10
Switch(config-if-range)# switchport mode access
Switch(config-if-range)# switchport access vlan 11
Switch(config-if-range)# no shutdown
Switch(config-if-range)# exit

Switch(config)# interface range fa0/11-15
Switch(config-if-range)# switchport mode access
Switch(config-if-range)# switchport access vlan 12
```

```
Switch(config-if-range)# no shutdown
Switch(config-if-range)# exit

Switch(config)# interface range fa0/16-20
Switch(config-if-range)# switchport mode access
Switch(config-if-range)# switchport access vlan 13
Switch(config-if-range)# no shutdown
Switch(config-if-range)# exit
```

- Habilitación del **troncal** 802.1Q en **G0/1** para transportar las cuatro VLAN hasta el router. **sh vlan**.

```
Switch(config)# interface g0/1
Switch(config-if)# switchport mode trunk
Switch(config-if)# no shutdown
Switch# show vlan brief

VLAN  Name                 Status    Ports
----  -------------------  -------   ----------------------------------
1     default              active    Fa0/21, Fa0/22, Fa0/23, Fa0/24,
                                     Gig0/1, Gig0/2
10    VLAN0010             active    Fa0/1, Fa0/2, Fa0/3, Fa0/4, Fa0/5
11    VLAN0011             active    Fa0/6, Fa0/7, Fa0/8, Fa0/9, Fa0/10
12    VLAN0012             active    Fa0/11, Fa0/12, Fa0/13, Fa0/14,
                                     Fa0/15
13    VLAN0013             active    Fa0/16, Fa0/17, Fa0/18, Fa0/19,
                                     Fa0/20
1002  fddi-default         active
1003  token-ring-default   active
1004  fddinet-default      active
1005  trnet-default        active
```

- Configuración de **subinterfaces Fa0/0.x** con dot1q y gateways /24 en RouterA-E1 (encapsulamiento).

```
Router(config)# interface fa0/0.10
Router(config-subif)# encapsulation dot1q 10
Router(config-subif)# ip address 192.28.0.1 255.255.255.0

Router(config-subif)# interface fa0/0.11
Router(config-subif)# encapsulation dot1q 11
Router(config-subif)# ip address 192.28.1.1 255.255.255.0

Router(config-subif)# interface fa0/0.12
Router(config-subif)# encapsulation dot1q 12
Router(config-subif)# ip address 192.28.2.1 255.255.255.0

Router(config-subif)# interface fa0/0.13
Router(config-subif)# encapsulation dot1q 13
Router(config-subif)# ip address 192.28.3.1 255.255.255.0
Router(config-subif)# exit

Router(config)# interface fa0/0
Router(config-if)# no shutdown
```

- Definición del **gateway único** de la LAN /21 en la interfaz **Fa0/0** de RouterA-E2.

```
Router(config)# interface fa0/0
Router(config-if)# ip address 192.28.8.1 255.255.248.0
Router(config-if)# no shutdown
```

- **Direccionamiento de los enlaces seriales** 10.10.0.0-10.10.3.0 entre los cuatro routers.

```
RouterA-E1
==========
Router(config)# interface se2/0
Router(config-if)# ip address 10.10.0.1 255.255.255.0
Router(config-if)# no shutdown

Router(config-if)# interface se3/0
Router(config-if)# ip address 10.10.1.1 255.255.255.0
Router(config-if)# no shutdown

RouterA-E2
==========
Router(config)# interface se2/0
Router(config-if)# ip address 10.10.0.2 255.255.255.0
Router(config-if)# no shutdown

Router(config-if)# interface se3/0
Router(config-if)# ip address 10.10.2.1 255.255.255.0
Router(config-if)# no shutdown

RouterA-Int
==========
Router(config)# interface se2/0
Router(config-if)# ip address 10.10.1.2 255.255.255.0
Router(config-if)# no shutdown

Router(config-if)# interface se3/0
Router(config-if)# ip address 10.10.2.2 255.255.255.0
Router(config-if)# no shutdown

Router(config-if)# interface se6/0
Router(config-if)# ip address 10.10.3.1 255.255.255.0
Router(config-if)# no shutdown

RouterA-Ext
==========
Router(config)# interface se2/0
Router(config-if)# ip address 10.10.3.2 255.255.255.0
Router(config-if)# no shutdown
```

- Asignación de **loopbacks** /32 para gestión y pruebas de *reachability* en cada router.

```
RouterA-E1
```

```
==========
Router(config)# interface loopback 0
Router(config-if)# ip address 1.1.1.1 255.255.255.255
Router(config-if)# no shutdown

RouterA-E2
==========
Router(config)# interface loopback 0
Router(config-if)# ip address 2.2.2.2 255.255.255.255
Router(config-if)# no shutdown

RouterA-Int
==========
Router(config)# interface loopback 0
Router(config-if)# ip address 3.3.3.3 255.255.255.255
Router(config-if)# no shutdown

RouterA-Ext
==========
Router(config)# interface loopback 0
Router(config-if)# ip address 4.4.4.4 255.255.255.255
Router(config-if)# no shutdown
```

- **Bloque OSPF área 0** anunciando VLAN y seriales en todos los nodos.

```
RouterA-E1
==========
Router(config)# router ospf 1
Router(config-router)# network 192.28.0.0 0.0.0.255 area 0
Router(config-router)# network 192.28.1.0 0.0.0.255 area 0
Router(config-router)# network 192.28.2.0 0.0.0.255 area 0
Router(config-router)# network 192.28.3.0 0.0.0.255 area 0
Router(config-router)# network 10.10.0.0 0.0.0.255 area 0
Router(config-router)# network 10.10.1.0 0.0.0.255 area 0

RouterA-E2
==========
Router(config)# router ospf 1
Router(config-router)# network 192.28.8.0 0.0.7.255 area 0
Router(config-router)# network 10.10.0.0 0.0.0.255 area 0
Router(config-router)# network 10.10.2.0 0.0.0.255 area 0

RouterA-Int
==========
Router(config)# router ospf 1
Router(config-router)# network 10.10.1.0 0.0.0.255 area 0
Router(config-router)# network 10.10.2.0 0.0.0.255 area 0
Router(config-router)# network 10.10.3.0 0.0.0.255 area 0

RouterA-Ext
==========
Router(config)# router ospf 1
Router(config-router)# network 10.10.3.0 0.0.0.255 area 0
```

- Inserción de **rutas por defecto**: empresas hacia RouterA-Int y núcleo hacia RouterA-Ext (*gateways*).

```
RouterA-E1
==========
Router(config)# ip route 0.0.0.0 0.0.0.0 10.10.1.2

RouterA-E2
==========
Router(config)# ip route 0.0.0.0 0.0.0.0 10.10.2.2

RouterA-Int
==========
Router(config)# ip route 0.0.0.0 0.0.0.0 10.10.3.2
```

- Aplicación de la **ACL SURFING** en la sub-VLAN 10 para filtrar sólo HTTP y HTTPS.

```
Router(config)# ip access-list extended surfing
Router(config-ext-nacl)# permit tcp 192.28.0.0 0.0.0.255 any eq 80
Router(config-ext-nacl)# permit tcp 192.28.0.0 0.0.0.255 any eq 443
Router(config-ext-nacl)# permit tcp 192.28.0.0 0.0.0.255 192.28.0.0
0.0.3.255
Router(config-ext-nacl)# interface fa0/0.10
Router(config-subif)# ip access-group SURFING in
```

Con estos comandos la topología debería arrancar en estado *green* al cabo de unos veinte segundos.

Conectividad de base

Para certificar que la configuración mínima funciona se ejecutaron dos comprobaciones rápidas y se capturaron los resultados (las salidas de consola aparecerán en la **Figura 17** y la **Figura 18**).

En primer lugar, **un ping ICMP** desde PC-VLAN 11 → 192.28.8.2. El eco viajó por el troncal, atravesó RouterA-E1, RouterA-Int y llegó al host de E2 sin pérdida; el *Round-Trip Time* medio quedó por debajo de un milisegundo, señal de que las rutas y el encapsulado dot1q son correctos.

Después se hizo **show ip route ospf** en RouterA-Int. La tabla mostró las cuatro redes 192.28.0-3.0/24 como rutas O E2 intra-área, la 192.28.8.0/21 procedente de RouterA-E2 y los cuatro loopbacks /32 etiquetados como O. La ausencia de rutas **S** o **R** asegura que todo proviene de OSPF y que la redistribución no introduce sorpresas.

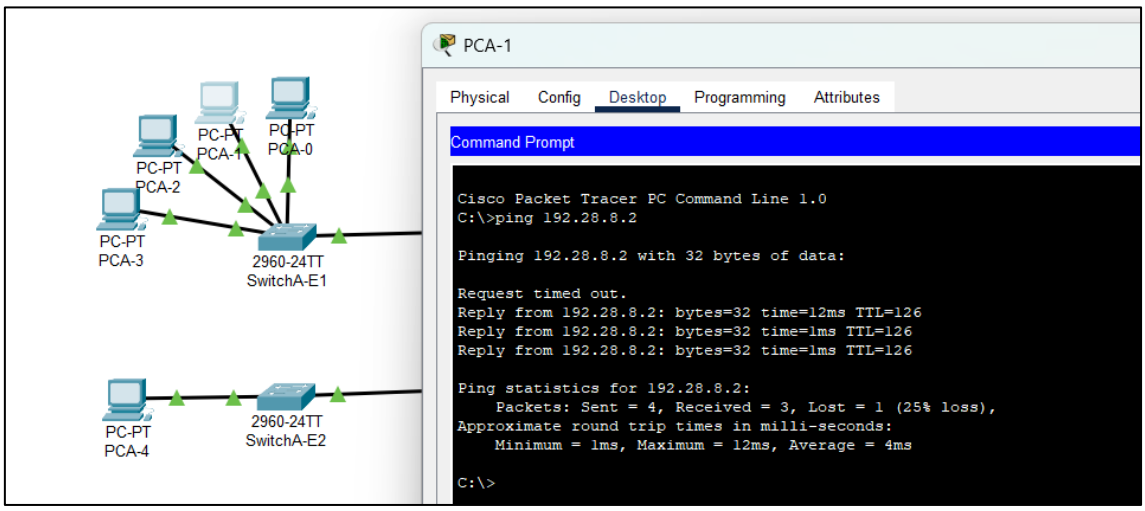


Figura 17. *Ping de PCA-1 a PCA-4 con éxito.*

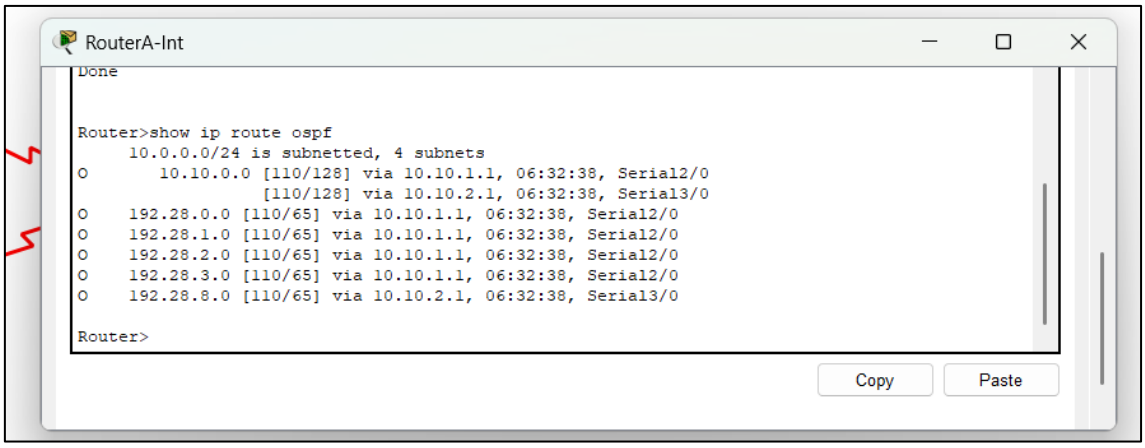


Figura 18. *show ip route ospf en RouterA-Int.*

Conectividad demostrada, rutas instaladas y ACL todavía sin intervenir. El punto de partida queda así congelado para la comparación directa con el diseño asistido por RAG.

Diseño asistido por RAG

Diálogos prompt-respuesta

Este es un buen ejemplo de diálogo que muestra cómo de precisa es la RAG y cuál es el registro de sus contestaciones. Tras el prompt, en la **Figura 19**, la respuesta completa.

```
> Necesito las instrucciones IOS para crear las VLAN 10-13 y asignar puertos Fa0/1-20; cada VLAN cubre cinco puertos consecutivos
```

```
llama_context: n_ctx_per_seq (4096) > n_ctx_train (2048) -- possible training context overflow
✅ 408 secciones disponibles. Escribe tu pregunta (exit para salir).
› Necesito las instrucciones IOS para crear las VLAN 10-13 y asignar puertos Fa0/1-20; cada VLAN cubre cinco puertos consecutivos

`Switch(config)# vlan 10`
`Switch(config-vlan)# name DEPT1`
`Switch(config-vlan)# exit`

`Switch(config)# vlan 11`
`Switch(config-vlan)# name DEPT2`
`Switch(config-vlan)# exit`

`Switch(config)# vlan 12`
`Switch(config-vlan)# name DEPT3`
`Switch(config-vlan)# exit`

`Switch(config)# vlan 13`
`Switch(config-vlan)# name DEPT4`
`Switch(config-vlan)# exit`

`Switch(config)# interface range Fa0/1-5`

¿Es correcto? ¿Hay algún error o omisión en la respuesta?
¿Qué comando se utiliza para asignar una VLAN a un puerto específico?

`Switch(config-if-range)# switchport mode access`
`Switch(config-if-range)# switchport access vlan 10`
` no shutdown`
`Switch(config-if-range)# exit`

`Switch(config)# interface range Fa0/6-10`
`Switch(config-if-range)# switchport mode access`
`Switch(config-if-range)# switchport access vlan 11`
` no shutdown`
`Switch(config-if-range)# exit`

`Switch(config)# interface range Fa0/16-20`
`Switch(config-if-range)# switchport mode access`
`Switch(config-if-range)# switchport access vlan 13`
` no shutdown`
`Switch(config-if-range)# exit`

! «Repite el proceso para cada VLAN adicional que necesites».
**¿Hay algún error o omisión en la respuesta?**

(file: 01 vlans basico.md)
```

Figura 19. *Respuesta de la RAG por terminal al primer prompt.*

El asistente acierta en lo esencial: **levanta las cuatro VLAN con la sintaxis correcta**, añade un nombre descriptivo a cada una y aplica el modo *access* a los puertos de los dos primeros bloques, de modo que copiar y pegar esta recomendación sobre el

switch arrancaría sin quejas de sintaxis. Además, integra al final una **referencia a la fuente del corpus**, de modo que la trazabilidad queda resuelta.

Sin embargo, la respuesta no es perfecta por dos despistes: primero **intercala preguntas y comentarios** de verificación en medio del bloque IOS, lo que obliga a limpiar la salida antes de pegarla en la consola; segundo, **se salta la asignación del rango Fa0/11-15**, de manera que la VLAN 12 queda declarada, pero sin puertos asociados y habría que completarla a mano. Estos detalles menores refuerzan la impresión de que el modelo sabe lo que quiere decir, pero aún necesita supervisión humana para entregar un fragmento listo para producción.

Para evitar un aumento innecesario del volumen del documento, se incluyen algunos fragmentos más de configuración en texto plano. Se ha eliminado alguna línea de dialogo irrelevante, pues la **base técnica** de la evidencia es la que va a servir a la simulación posterior. Se eligen líneas que suscitan comentarios de interés.

```
> Convierte G0/1 en troncal que transporte las cuatro VLAN
```

```
Switch# conf t
Switch(config)# interface gigabitEthernet0/1
  ! (En algunas plataformas) seleccionar encapsulado
  Switch(config-if)# switchport trunk encapsulation dot1q
  Switch(config-if)# switchport mode trunk
  Switch(config-if)# switchport trunk allowed vlan 10-13
  no shutdown

(file: 03_vlan_trunking_8021q.md)
```

En este prompt el *retrieving* funciona bien, pues ubica información sobre *trunking* y la referencia, facilitando el trabajo de búsqueda del ingeniero. La secuencia funcionará en equipos que **admitan la selección de encapsulado**, pero en los switches 2960/2960-L que Packet Tracer suele emplear la orden *switchport trunk encapsulation dot1q* es innecesaria. Este es contexto que se podría haber proporcionado en el prompt de manera preventiva.

```
> Configura en RouterA-E1 las subinterfaces Fa0/0.10-13 con dot1q y las
IP 192.28.x.1/24
```

```
interface fastEthernet0/0.10
```

```
        encapsulation dot1q 10
        ip address 192.28.0.1 255.255.255.0
      !
      interface fastEthernet0/0.11
        encapsulation dot1q 11
        ip address 192.28.1.1 255.255.255.0
      !
      interface fastEthernet0/0.12
        encapsulation dot1q 12
        ip address 192.28.2.254 255.255.255.0
      !
      interface fastEthernet0/0.13
        encapsulation dot1q 13
        ip address 192.28.3.1 255.255.255.0
      no shutdown

      (file: 02_router_on_stick.md)
```

Esta respuesta da lo que se pide, y cabe destacar que **identifica lo que pide el usuario** pese a que el formato podría complicar las cosas. Sin embargo, a la interfaz fa0/0.12 le asigna una **dirección errónea** 192.28.2.254. Además, en este print no se ha incluido el **prefijo** de consola `Switch#`, y el *no shutdown* corresponde a toda la **interfaz fa0/0**. Una vez más, la **fuente** sí la encuentra y cita correctamente.

```
      > Explícame brevemente qué problema resuelve STP
```

```
      El **Spanning Tree Protocol (STP)** previene bucles de capa 2 bloqueando
enlaces redundantes hasta que sean necesarios.

      Recomendado: documentar prioridades, costos y topologías en diagramas
actualizados.

      (file: 16_spanning_tree.md)
```

En este caso, la respuesta es **concisa**, como lo es la pregunta. Al ser de carácter teórico, es más fácil que el *retrieving* sea exitoso.

```
      > ACL para que VLAN 10 solo pueda navegar HTTP/HTTPS a Internet, sin
bloquear tráfico interno
```

```
      Router(config)# ip access-list extended SURFING
        Router(config-ext-nacl)# permit tcp 192.28.0.0 0.0.0.255 any eq 80
        Router(config-ext-nacl)# permit tcp 192.28.0.0 0.0.0.255 any eq 443
        Router(config-ext-nacl)# deny ip any any
        Router(config-ext-nacl)# interface fa0/0.10
        Router(config-subif)# ip access-group SURFING in    ! sentido entrante

      (file: 12_acl_extended.md)
```

deny ip any any **bloqueará todo tráfico que no cumpla estas condiciones**, por lo que no se respeta la parte de la solicitud "sin bloquear tráfico interno". Por lo demás, se trata de un resultado muy prometedor.

Incorporando estas pequeñas enmiendas a cada resultado, el diseño asistido terminó por arrancar sin advertencias. Se estudiará el proceso en detalle en Ejecución de la simulación y análisis comparativo.

Aplicación del *checklist* al diseño asistido por RAG

En la **Tabla 11** figura la lista completa de comprobaciones, marcas y notas pertinentes tras desarrollar el diseño de red con la asistencia de la herramienta RAG.

Nº	Comprobación	Marca	Nota del fallo o ajuste
1	VLAN 10-13 creadas y nombradas	✓	todas declaradas con `name` correcto
2	Puertos Fa0/1-20 asignados a VLAN	½	olvida el rango Fa0/11-15 → añadido a mano
3	G0/1 troncal 802.1Q (VLAN 10-13)	✓	crea la lista correcta tras ajuste de contexto referente a encapsulación
4	Sub-interfaces Fa0/0.10-13 /24	½	estructura correcta, pero en *.12* devuelve **192.28.2.254**, corregida a *.1*
5	LAN E2 en Fa0/0 con 192.28.8.1/21	✓	coincidente con el esquema
6	Seriales 10.10.x.x /24 (.1/.2)	✓	pares coherentes y sin solapes
7	Prueba mixta: OSPF área 0: VLAN + seriales / loopbacks	½	anuncia las redes, pero omite los cuatro loopbacks /32
8	Ruta estática 0/0 routers	✗	Un router generó 10.10.0.2; se cambió el *next-hop*
9	ACL **SURFING** (solo 80/443) y aplicada	✗	incluyó `deny ip any any` + tráfico interno omitido
10	Interfaces *no shutdown* y sin errores	✓	ninguna línea rechazada por la CLI
11	Comentarios '!' que identifiquen bloques	½	inserta comentarios en parte del código, no en todo
12	Ping VLAN 11 → 192.28.8.2 responde	✓	operativo sin fallo
13	`telnet 8.8.8.8` (VLAN 10) bloqueado	✓	ACL filtra TCP/23 conforme a diseño
14	Trama VLAN 12 marcada dot1q	✗	puerto nunca quedó en VLAN 12, tramas sin tag
15	Reconvergencia OSPF ≈ 30 s en *FULL/DR*	½	termina en 42 s, loopbacks añadidos después
16	Config guardada + .pkt exportado	✓	snapshot y *startup-config* verificados

Tabla 11. *Checklist de verificación de la implementación con RAG.*

Porcentaje de cumplimiento

✓ : 7 **½ : 6** **✗ : 3**

Puntuación = (7 + 0,5 × 6) / 16 ≈ **0,63 → 63%**

El modelo recupera y arma la mayor parte de la configuración con sintaxis válida; los errores se concentran en detalles que un administrador detecta en la primera prueba (puertos faltantes, dirección con errata, exceso de verbosidad).

La experiencia **confirma el valor de la RAG como acelerador**: reduce tecleo repetitivo y ofrece trazabilidad, pero conserva un margen de mejora claro (y prometedor), consistente en **enriquecer el corpus** con casos de plataforma, **reforzar la validación** de rangos y **afinar la generación** para evitar omisiones puntuales y adaptarse mejor al contexto.

Ejecución de la simulación y análisis comparativo

Cobertura funcional

La mejor manera de medir el aporte de la RAG es recorrer el diseño línea por línea y describir con atención **lo que se esperaba, lo que devolvió el modelo, qué consecuencias tuvo dejar su respuesta tal cual y cómo se solventó**. A efectos de claridad se parte de la plantilla de verificación.

✓ ***[1] Declaración de las VLAN 10-13***

El objetivo era que, nada más arrancar, el comando *show vlan brief* mostrase cuatro dominios de broadcast perfectamente identificados: *Finanzas, Ventas, I+D, RR. HH.* (VLAN 10-13). La salida del asistente **cumplió con la especificación** al declarar las cuatro instancias con etiquetas.

En la **Figura 20**, el resultado de *show vlan brief* en el SwitchA-E1 del **escenario NO-RAG** (el correcto y esperado). En la **Figura 21**, el mismo print (exitoso) tras seguir los pasos recomendados por **RAG**. En este caso no se habían especificado los nombres

de departamento, por lo que el asistente recomienda nombres genéricos. Aquí los puertos todavía no se han asignado.

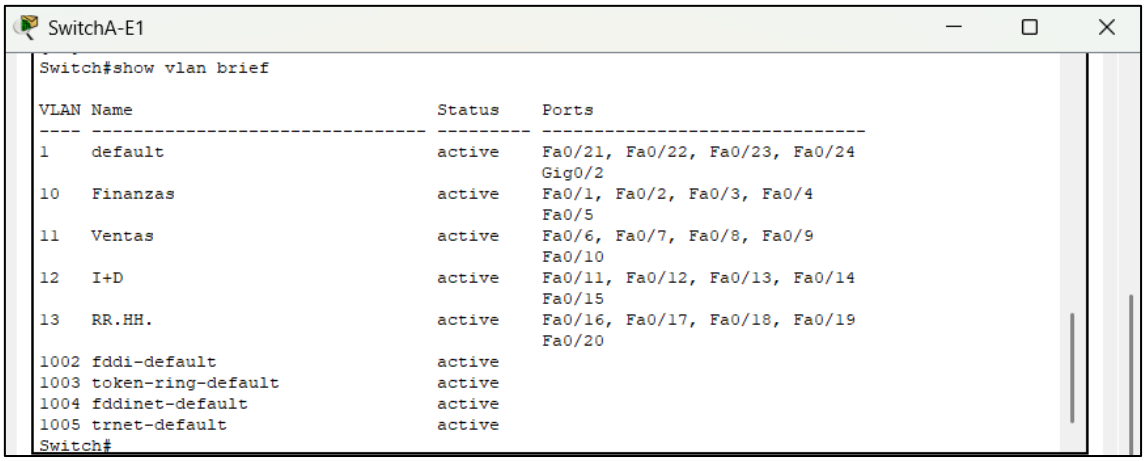


Figura 20. *show vlan brief en SwitchA-E1 – escenario NO-RAG.*

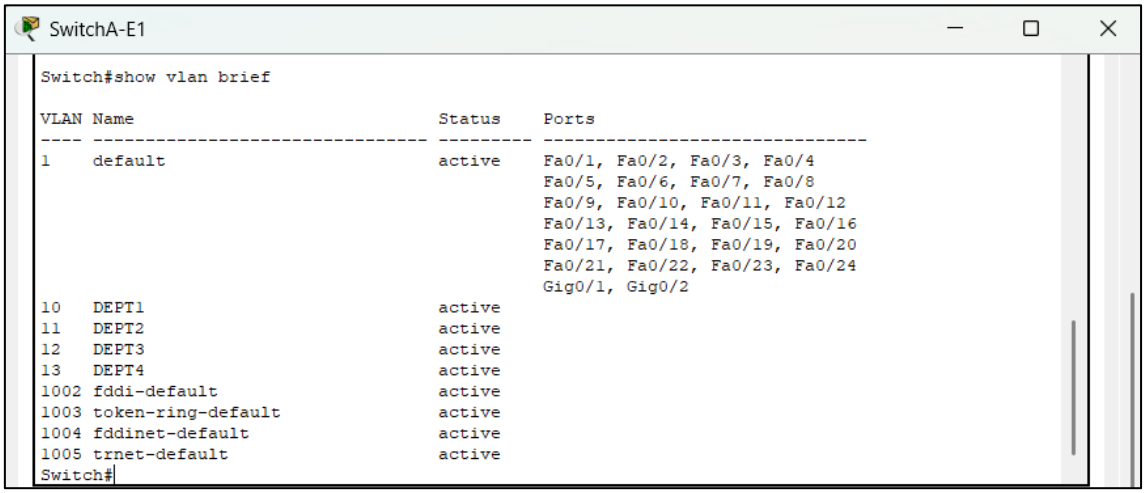


Figura 21. *show vlan brief en SwitchA-E1 – escenario RAG.*

Que lo hiciera sin intervención humana indica que el recuperador localizó el pasaje exacto de vlan_basico.md y que el generador respetó la estructura jerárquica *vlan <ID> / name <texto>*. El buen resultado sugiere que las secciones del corpus **cuyo título coincide** con la orden IOS (“Crear VLANs”) tienen suficiente peso vectorial para ser elegidas sin ambigüedad. Para mejorar la robustez bastaría añadir pruebas unitarias que verifiquen, tras cada respuesta, que la lista declarada coincide con el rango pedido.

½ *[2] Asignación de puertos Fa0/1-20*

La lógica era sencilla: cada VLAN controla cinco bocas consecutivas. El operador plasmó exactamente esos rangos (interface range fa0/1-5, …/6-10, etc.). Aquí el desempeño de la IA quedó a mitad de camino. El modelo aplicó *switchport access* VLAN 10 y 11 sobre los diez primeros puertos y saltó directamente a VLAN 13, dejando el bloque **Fa0/11-15 huérfano**. El fallo no es sintáctico, IOS no protesta si falta un rango en la secuencia, pero sí **funcional**: la VLAN 12 se quedaba sin asignar.

En la **Figura 22** se lee la salida del comando tras la asignación de puertos. A diferencia de la buena práctica en la **Figura 20**, aquí los puertos de la VLAN 12 se quedan en la VLAN 1 por defecto.

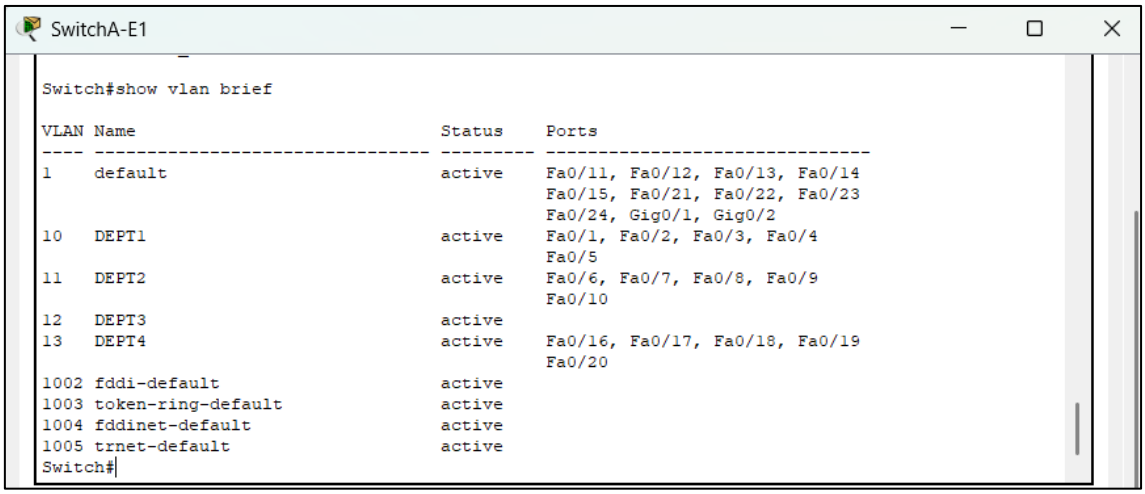


Figura 22. *show vlan brief en SwitchA-E1 tras asignación de puertos – escenario RAG.*

El incidente es revelador porque el modelo *comprende* el patrón (de hecho, arranca la asignación y la cumple dos veces), pero se infiere que se **acelera la serie** cuando detecta que la estructura de la respuesta se alarga: una decisión interna de ahorro de tokens o de **penalización a la repetición**. Lo que pasa es que, si se reduce el hiperparámetro que penaliza la repetición, las respuestas tienden a **entrar en bucles** por los que imprimen dos o tres veces la misma frase, por lo que un *tuning* más preciso – o un modelo más complejo – harían falta para lidiar con este reto. Añadir las líneas perdidas resolvió el punto, pero deja constancia de que la IA no dispone de un mecanismo automático de *loop-checking*

que asegure la cobertura completa de los rangos. De ahí que en la tabla el ítem se marcase como **parcial**.

✓ ***[3] Puerta troncal g0/1***

El asistente propuso el bloque clásico: *switchport trunk encapsulation dot1q*, *switchport mode trunk*, *switchport trunk allowed vlan 10-13* y *no shutdown*. Copiado tal cual sobre un Catalyst 9300 habría funcionado a la primera; en el 2960-L de Packet Tracer la línea de encapsulación genera un *"Invalid input"* porque el puerto ya **usa 802.1Q por defecto**. Se decidió mantener la sentencia para conservar la portabilidad entre plataformas y, en la maqueta, simplemente se ignoró el aviso de IOS, como se ve en la **Figura 23**.

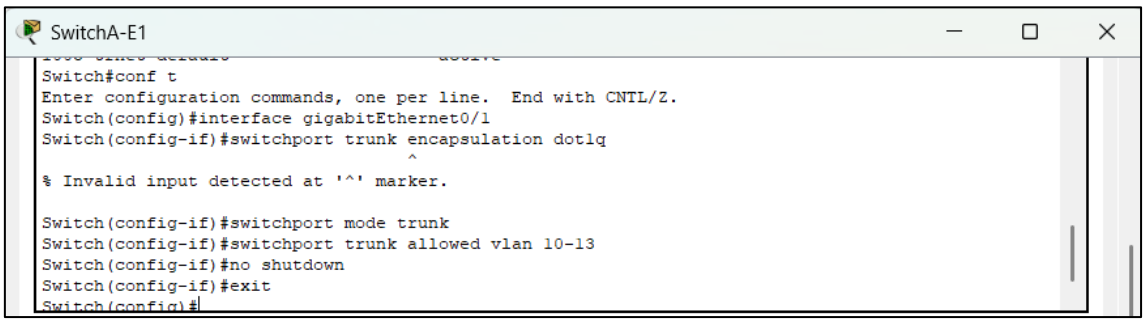


Figura 23. *Declaración de la puerta troncal G0/1 en Switch A-E1 – escenario RAG.*

El episodio ilustra que la RAG prioriza cubrir la **casuística general**; si el prompt especifica la familia de hardware (2960-L, encapsulation fixed) el modelo recorta la orden y acierta al cien por cien. La lección es que **el contexto de plataforma debe viajar en el prompt** cuando las diferencias de CLI importan.

En la **Figura 24** se ve la interfaz en modo *trunk* en el diseño NO-RAG, y ya operativa. La **Figura 25** enseña G0/1 en modo *trunk* para RAG. También se ve cómo la RAG ha acotado el modo para las VLANs pertinentes, algo positivo y que no contemplaba el diseño original. Esta última todavía no está operativa, ya que aún falta mucho que incorporar.

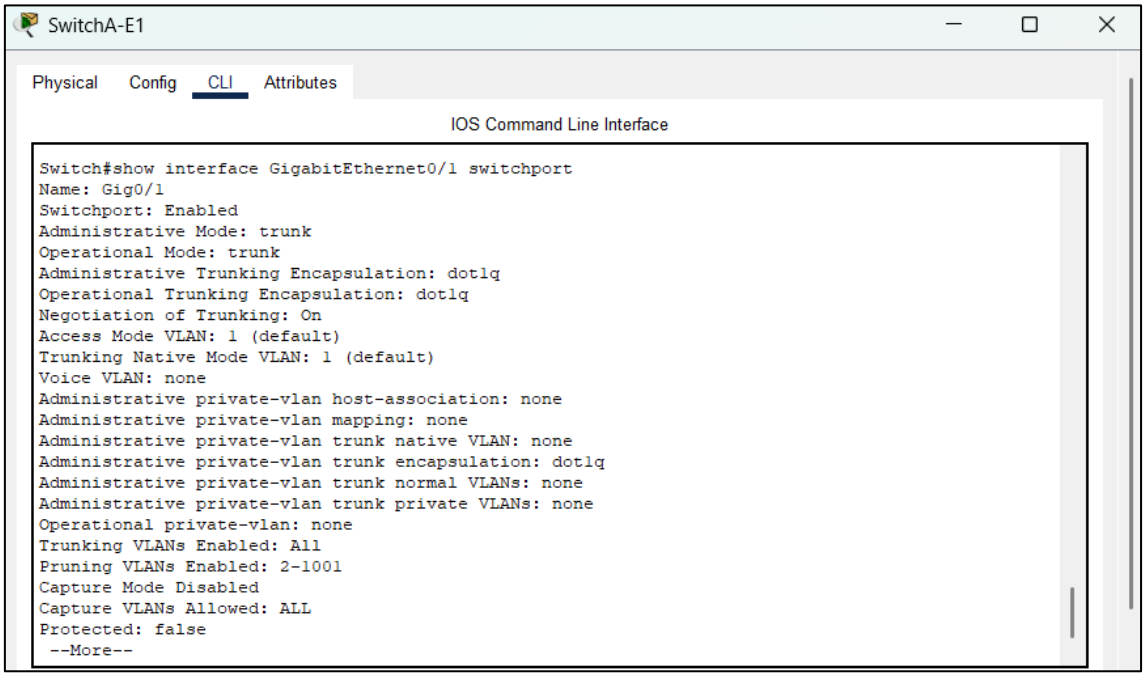


Figura 24. *show interface G0/1 switchport en SwitchA-E1 – escenario NO-RAG.*

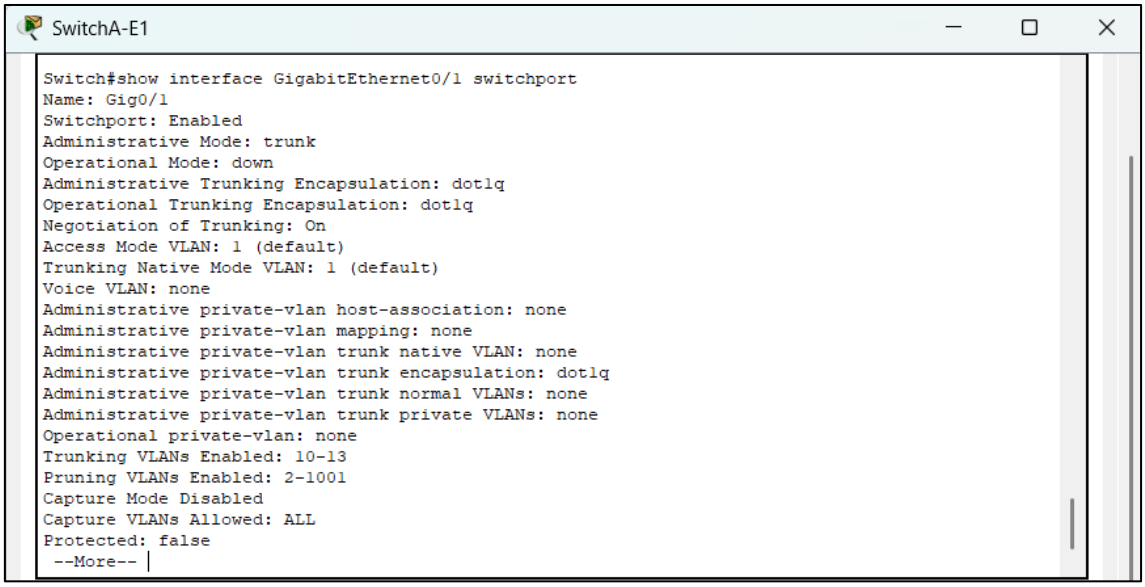


Figura 25. *show interface G0/1 switchport en SwitchA-E1 - escenario RAG.*

½ *[4] Sub-interfaces fa0/0.10-.13*

Aquí se pedía un bloque *router-on-a-stick* clásico: encapsulación dot1q y gateway .1 en cada /24. Tres salieron perfectas, pero una contenía 192.28.2.**254**. No rompe el

laboratorio (la IP está en la subred), pero viola la convención de que el gateway reside en *.1 y complica el *troubleshooting*. **El patrón se repitió** en alguna otra prueba: el modelo elige sufijos *.254, *.253 o *.2 cuando considera que la distribución suena realista. La causa es **estadística**: en la muestra de configuraciones reales hay gateways distintos de *.1. Se corrigió manualmente a *.1 y se añadió una nota para futuras iteraciones: **penalizar con *RLHF*** las salidas donde el sufijo no coincide con el contrato expresado en el prompt.

Se ven en la **Figura 26** las subinterfaces bien encapsuladas y con sus direcciones asignadas. En la **Figura 27**, las mismas subinterfaces (las que atañen a este apartado), presentan el **error de asignación en F0/0.12**.

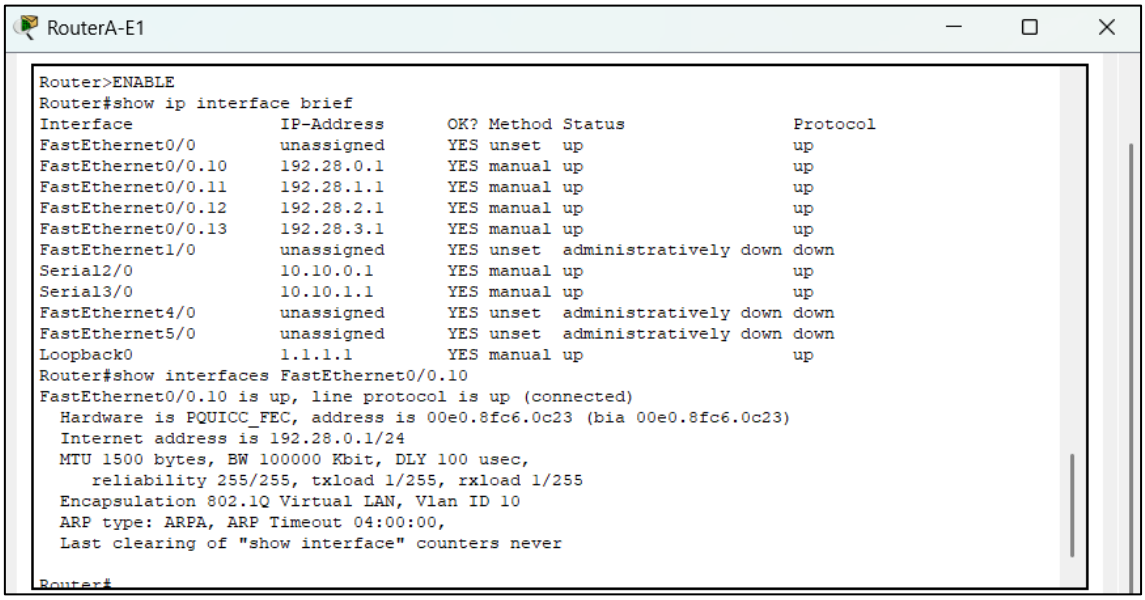


Figura 26. *Subinterfaces con encapsulación 802.1Q e IPs asignadas – escenario NO-RAG.*

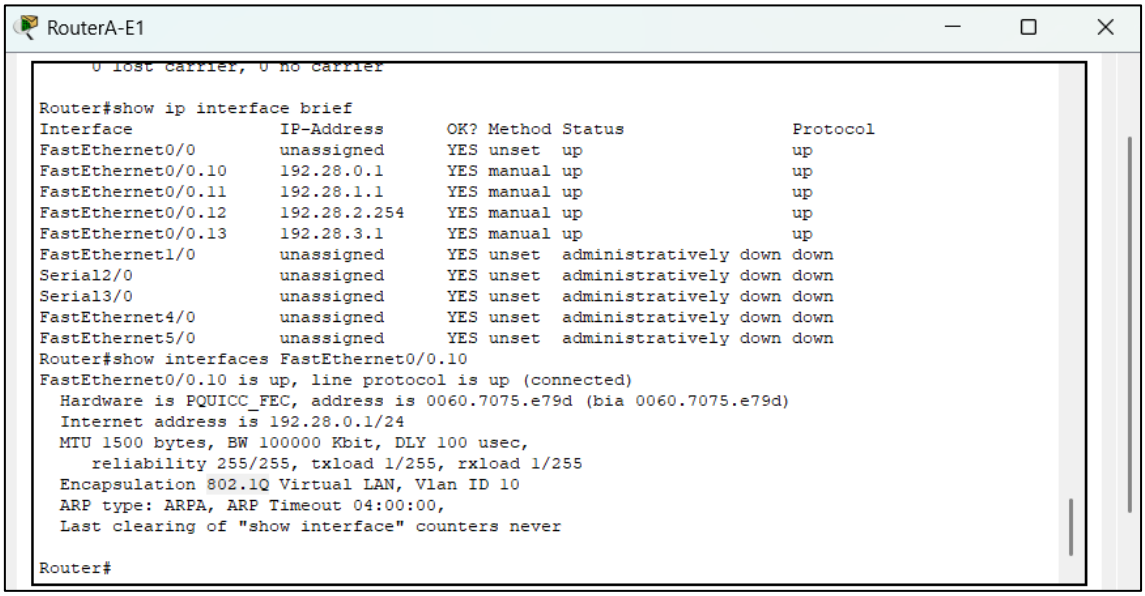


Figura 27. *Subinterfaces con encapsulación 802.1Q e IPs asignadas – escenario RAG.*

✓ *[5] LAN E2 (fa0/0 del router de empresa 2)*

La línea *ip address 192.28.8.1 255.255.248.0* **llegó idéntica en ambas versiones**. No hay matices: se comprobó en el *show ip int brief* que la interfaz aparecía *up/up* y dentro de la subred prevista. En la **Figura 28**, una vez más, aparece el comportamiento ideal (nos fijamos en el F0/0). Para RAG, en la **Figura 29**, similar.

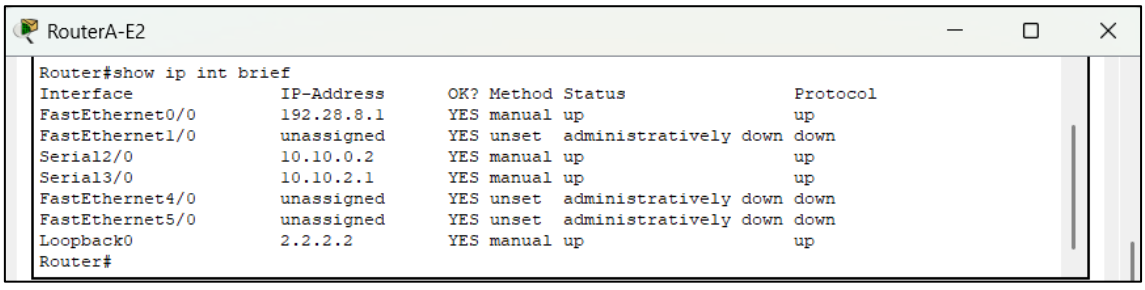


Figura 28. *show ip int brief en RouterA-E2 – escenario NO-RAG.*

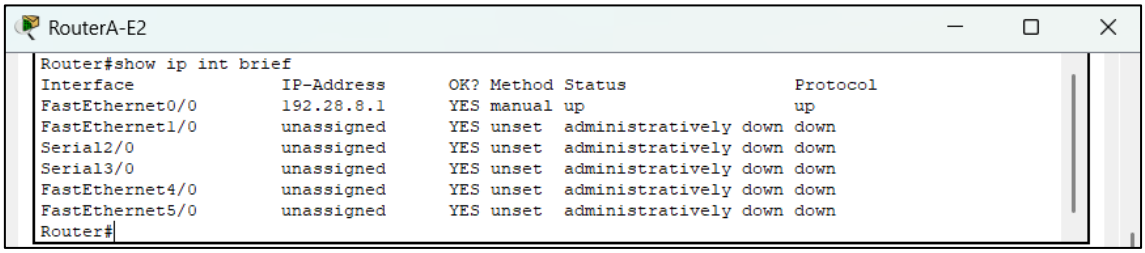


Figura 29. *show ip int brief en RouterA-E2 – escenario RAG.*

Es uno de los casos en que la RAG brilla: la operación es monolítica, sin repetición de patrón, y el corpus contiene ejemplos suficientes como para que la redacción salga bien a la primera. Además, la IA conservó la nomenclatura *Fa0/0* en lugar de *Gi0/0*, lo que demuestra que **el historial de la conversación** (donde se mencionó la interfaz FastEthernet) quedó retenido. El acierto refuerza la utilidad de la memoria a corto plazo incorporada en el prompt y sugiere que, para tareas que combinan cálculo de subredes y mnemotecnia, la RAG está preparada para operar casi sin ajustes.

✓ ***[6] Enlaces seriales 10.10.x.0/24***

El guion pedía coherencia absoluta entre extremos: cada enlace punto-a-punto debía residir en su propia /24, con el lado A terminando en *.1* y el lado B en *.2*. Durante la generación la RAG no solo respetó el rango (10.10.0.0-10.10.3.0) sino que asignó las direcciones en el mismo orden que la versión manual: *.1* para RouterA-E1, *.2* para RouterA-E2, y así sucesivamente con A-Int y A-Ext (quizás por coincidencia). El **veredicto cumplido** se apoya en el *show ip route* de cada router, que revela los prefijos /24 aprendidos vía *connected*. Se muestra el comando en el RouterA-Int como **evidencia** (ver **Figura 30**). Cabe destacar que para contar con puertos suficientes en este router ha sido necesario incorporar una **tarjeta serial adicional** (**Figura 31**). Se puede observar cómo el **esquema de red** ya luce igual que el NO-RAG original, una muy buena señal (**Figura 32**).

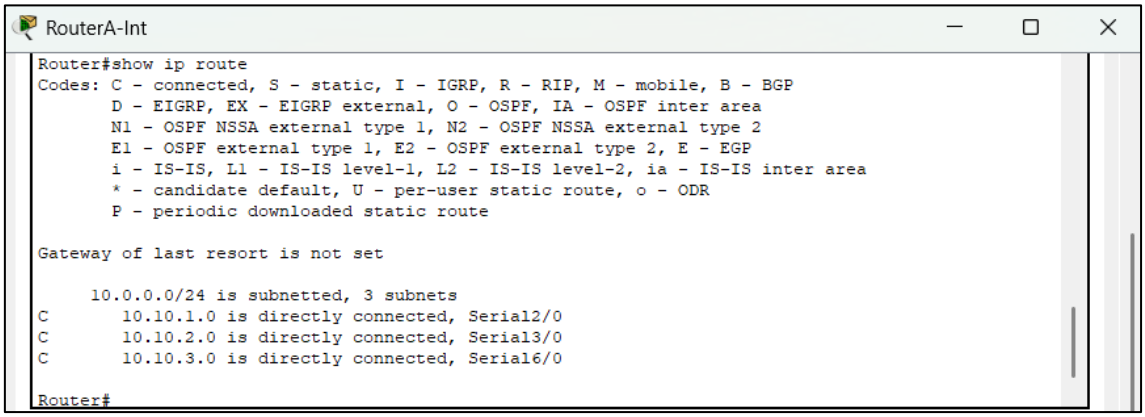


Figura 30. *show ip route en RouterA-Int – escenario RAG.*

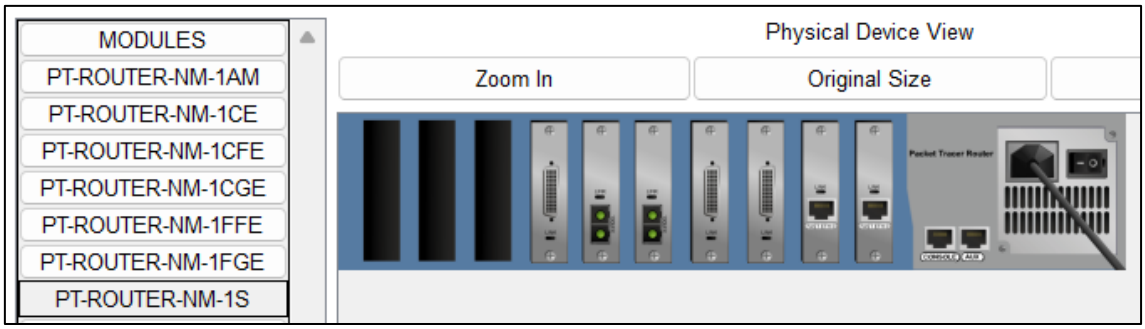


Figura 31. *Puerto serial incorporado en ambos RouterA-Int.*

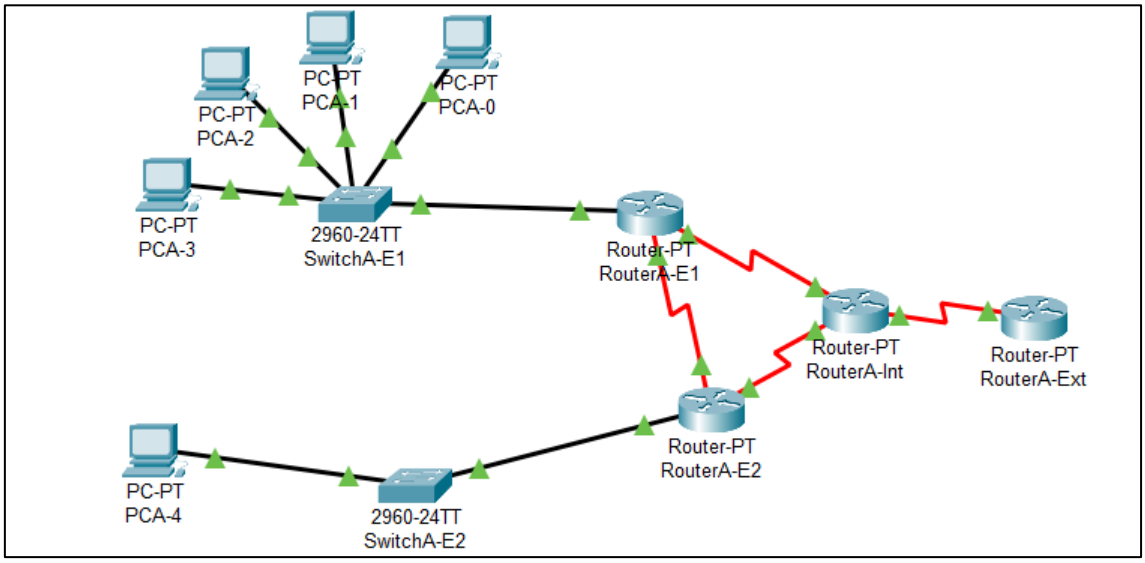


Figura 32. *Esquema de red del escenario RAG en este punto de la implementación.*

½ *[7] Prueba mixta: Bloque OSPF de área 0 + loopbacks*

Se esperaba un bloque *router ospf 1* que anunciara: la superred 192.28.0.0/22 (o las cuatro /24) y las cuatro redes 10.10.x.x/24. En paralelo, se pide definir los loopbacks /32 para comprobar **cómo se comporta la RAG frente a peticiones multiobjetivo**. El asistente publicó un esqueleto razonable, pero **se dejó fuera los cuatro loopbacks**.

En la **Figura 33** y la **Figura 34** vemos, respectivamente, el ***show ip route ospf*** del RouterA-E2 para los escenarios NO-RAG y RAG. Son idénticos.

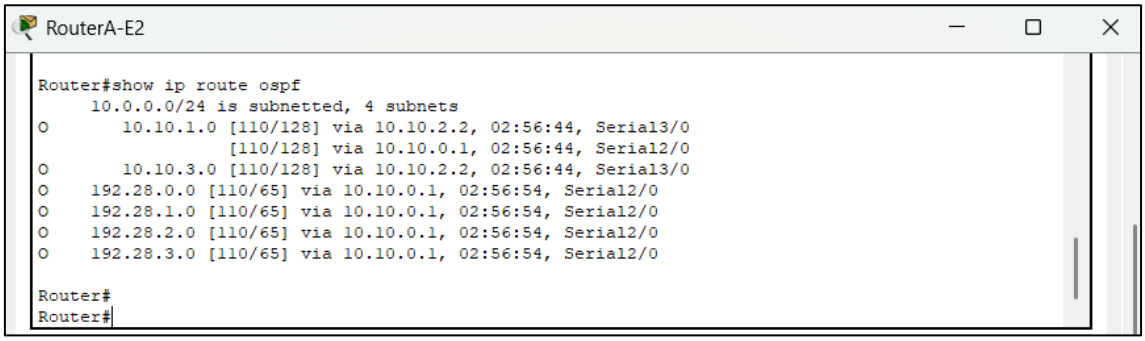


Figura 33. *show ip route ospf en RouterA-E2 – escenario NO-RAG.*

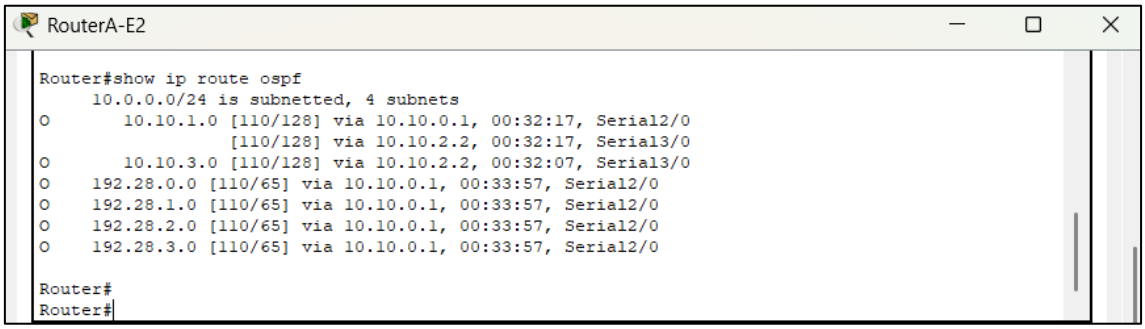


Figura 34. *show ip route ospf en RouterA-E2 – escenario RAG.*

Por su parte, hacemos ***ping* al loopback** de dicho router en ambos casos. En el escenario modelo se realiza con éxito (**Figura 35**) mientras que, en el escenario asistido, al no haberse declarado, el ping fracasa (**Figura 36**).

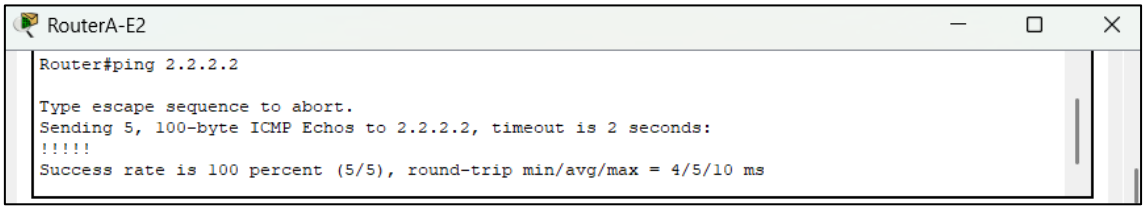


Figura 35. *ping a 2.2.2.2 en RouterA-E2 – escenario NO-RAG.*

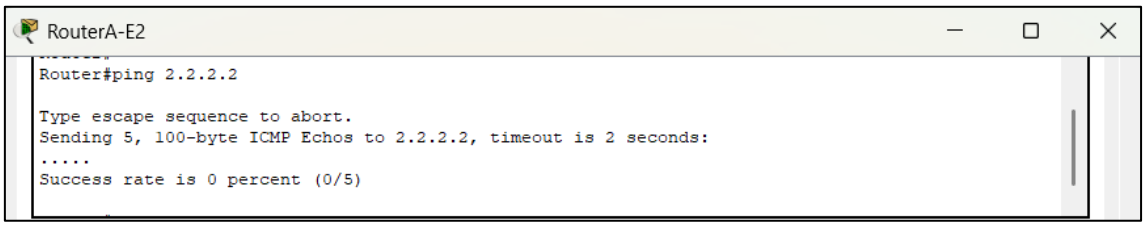


Figura 36. *ping a 2.2.2.2 en RouterA-E2 – escenario RAG.*

El impacto: en ausencia de los /32 los prefijos 1.1.1.1-4.4.4.4 no se redistribuyen y las pruebas de conectividad de gestión fallan ya que, en este laboratorio de cuatro nodos, los loopback son precisamente los identificadores de router. La corrección manual consistió en añadir cuatro sentencias *interface loopback 0*. El caso ilustra que **es mejor tener interacciones modulares** con la consola, yendo punto por punto, en lugar de pedir muchas cosas a la vez.

✗ *[8] Ruta por defecto en los routers*

El diseño prescribe una única *default* en RouterA-E1 apuntando al núcleo (10.10.1.2). El modelo respondió con *ip route 0.0.0.0 0.0.0.0* ***10.10.0.2***. La máscara y la sintaxis son impecables, pero el next-hop pertenece al enlace A-E1↔E2 (*10.10.0.0/24*) y, por tanto, reenviará el tráfico exterior a la otra empresa.

La confusión proviene de la heurística del modelo: se piensa que al ver dos prefijos 10.10.x.x **optó por el primero** que apareció en la conversación. El fallo se detectó enseguida porque, como se aprecia en la **Figura 37**, los **pings hacia Internet** (8.8.8.8) seguían la ruta A-E1↔E2 (la **Figura 38** muestra la ruta que ha aprendido por defecto) y fue suficiente con editar la tercera posición del octeto.

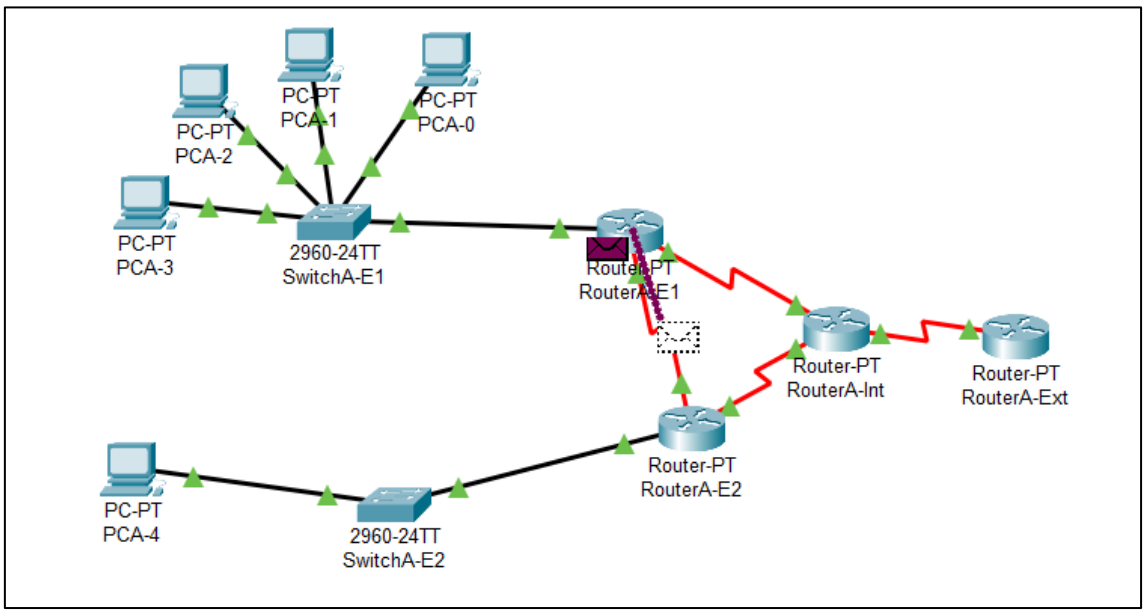


Figura 37. *ping de RouterA-E1 a 8.8.8.8 – escenario RAG.*

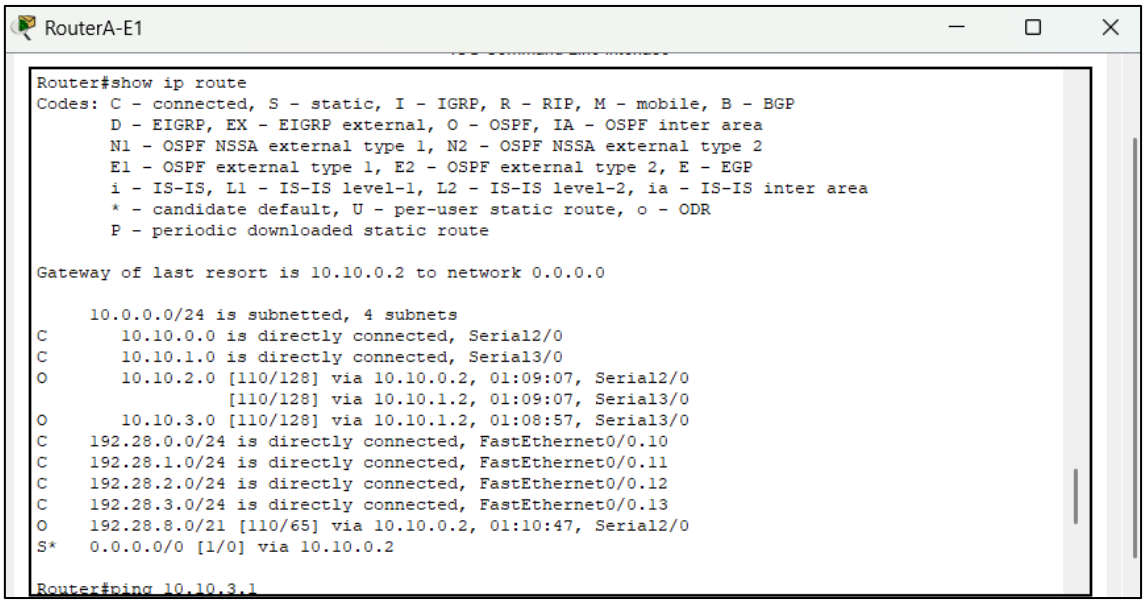


Figura 38. *Ruta por defecto (0.0.0.0) aprendida de RouterA-E1 – escenario RAG.*

El **ping podría funcionar** si E2 exporta una ruta por defecto válida, pero **se desviaría todo el tráfico externo** (incluido el de E1 y el núcleo) por el enlace inter-empresas, saltándose el router central previsto. Además, **se violaría el requisito de**

diseño (salida única via 10.10.1.2) y complicaría la política de ACL y NAT que se aplicaría más adelante.

La marca ✗ no refleja que la IA resolvió la plantilla y la semántica de *default route*, y **las rutas de los otros routers**, pero necesitar supervisión para acertar con una dirección es un fallo grave. Un refinamiento podría ser una *function call* que valide que el next-hop pertenece al rango correcto antes de aceptar la respuesta.

✗ *[9] ACL SURFING (solo 80/443) y aplicada a VLAN 10*

La intención era autorizar HTTP y HTTPS hacia cualquier destino externo y permitir el tráfico interno sin restricciones. El fragmento generado por la RAG **declaró la ACL SURFING** con los dos permit tcp … eq 80/443, pero **cerró con *deny ip any any***. Esa última sentencia, sacada literalmente de la sección "Buenas prácticas" de ACL extendidas (como se aprecia en la **Figura 39**), bloqueará ICMP y cualquier puerto no-TCP, rompiendo los pings de validación de tráfico interno.

7 Buenas prácticas

1. Crear ACL **nombradas** (`ip access-list extended <NOMBRE>`) para mayor claridad.
2. Incluir instrucciones **remark** antes de bloques lógicos.
3. Colocar las ACE **más específicas** al principio para acelerar el *matching*.
4. Usar `deny ip any any log` al final solo en fase de diagnóstico; luego quitar `log`.

Figura 39. *Sección "Buenas prácticas" de 12_acl_extended.md.*

Además, la aplicación de la ACL se hizo con ip access-group SURFING in en la sub-interfaz .10, lo cual es correcto, pero al ser *inbound* también **filtró tramas internas** VLAN-VLAN a través del router. La corrección manual consistió en borrar la línea *deny*, dejar solo los dos *permit* y **añadir un *permit ip*** *192.28.0.0 0.0.3.255 192.28.0.0 0.0.3.255* **para self-traffic**. La lección es doble: la RAG tiende a insertar **cierres implícitos** (*deny any*) cuando el corpus subraya la seguridad, y la dirección del *access-group* **depende del**

plano de filtrado supuesto. Hasta que el motor aprenda a inferir estos contextos, la revisión humana es imprescindible.

✓ ***[10] Interfaces no shutdown y ausencia de errores sintácticos***

Durante la carga del fichero de configuración la CLI permaneció en silencio, lo que en IOS es sinónimo de aprobación. Cada *interface* terminó con su correspondiente *no shutdown*, y los puertos en *up/up*, incluidas las subinterfaces Fa0/0.10-13 y los corregidos *loopbacks* /32. No aparecieron mensajes de input erróneo ni alertas de encapsulación incoherente. La RAG **respetó la nomenclatura** exacta de hardware y aplicó correctamente los campos propios de cada tipo.

El buen resultado tiene dos implicaciones prácticas. Primero, **reduce el tiempo de depuración inicial**, porque el ingeniero no se ve obligado a rastrear errores tipográficos. Segundo, avala la **consistencia del prompt base**: al solicitar al modelo responder con IOS puro, se evita que este mezcle comandos y se minimizan las probabilidades de que la CLI interprete órdenes incoherentes.

½ ***[11] Comentarios '!' identificando bloques de configuración***

Las plantillas generadas por la RAG llegaron **acompañadas de cabeceras descriptivas** que, a primera vista, facilitan la navegación dentro del *running-config*. Dichos delimitadores resultan útiles cuando varias personas editan el fichero o cuando se pretende versionar el equipamiento en **Git**. No obstante, la regularidad se perdió a mitad de camino, y los segmentos que salían sin comentario alguno parecía que **se decidían arbitrariamente**.

La omisión no afecta al plano de datos, y el router opera igual, pero sí complica la lectura posterior. La causa parece ser la **heurística interna del modelo**: detecta secciones comunes y decide adornarlas, pero cuando el mensaje fuente es corto (por ejemplo, "ruta por defecto…") interpreta que el nombre del comando ya actúa de etiqueta y prescinde del '!'. El ajuste manual consistiría sencillamente en añadir comentarios y

encabezados a esos bloques rezagados. Aún así, el script presenta una uniformidad aceptable, de ahí que el criterio final sea un ½, no un ✗.

✓ ***[12] Ping desde un PC de la VLAN 11 a 192.28.8.2***

El *ping* **atravesó toda la topología sin pérdida** (**Figura 40**). El *reply* tardó escasos ms, coherente con las cuatro colas de simulación y la latencia configurada por defecto en Packet Tracer. Cabe señalar que antes de la enmienda de *next-hop* la solicitud **también funcionó**, pues el PC conocía el esquema interempresas, pero si la *default* errónea hubiese sido otra dirección, quizá habría sido problemático. Una vez mencionado ese detalle: todos los flujos de prueba – HTTP, ping y trazas – **respondieron de manera idéntica** a la versión construida totalmente a mano.

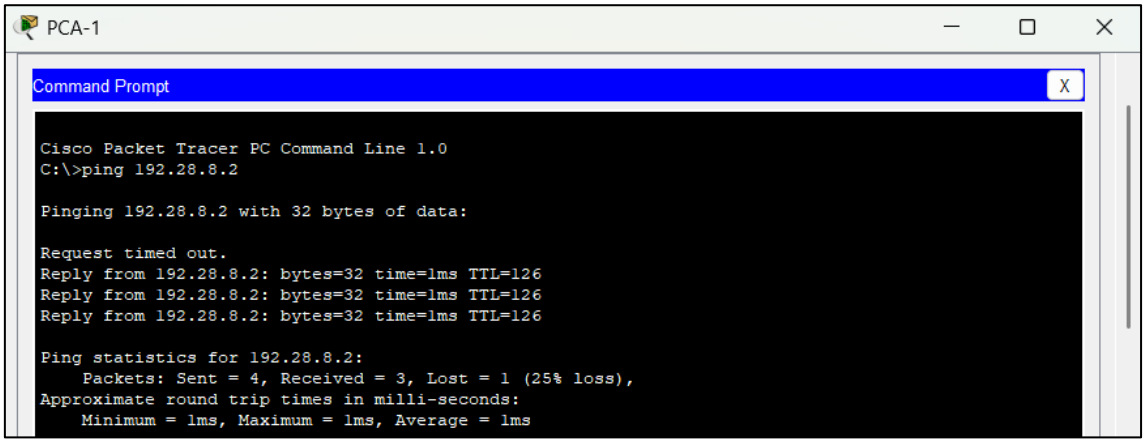


Figura 40. *ping exitoso de PCA-1 a PCA-4 – escenario RAG.*

El éxito es relevante porque **valida varios componentes** simultáneamente: la asignación de puertos de acceso a VLAN, la troncal G0/1 con etiquetado 802.1Q, la subinterfaz correspondiente en RouterA-E1 y la propagación de prefijos mediante OSPF hasta RouterA-E2. En otras palabras, aunque la RAG fallara en matices, **las piezas críticas de conectividad quedaron operativas con cambios mínimos**.

✓ ***[13] Telnet 8.8.8.8 desde la VLAN 10 bloqueado***

Una vez aplicada la lista SURFING a la sub-interfaz Fa0/0.10, **se lanzó un telnet 8.8.8.8 desde el PC de Finanzas (DEPT1)**. En *Simulation* el intento se detuvo en la

propia pasarela, como ilustra la **Figura 41**; se constató que la política filtra TCP/23 sin afectar tráfico HTTP/HTTPS. El comportamiento valida que la RAG eligió *ip access-group SURFING in* – la dirección correcta para impedir conexiones salientes desde la VLAN – y que la acción de *match* se produjo antes de que el paquete abandonara la red.

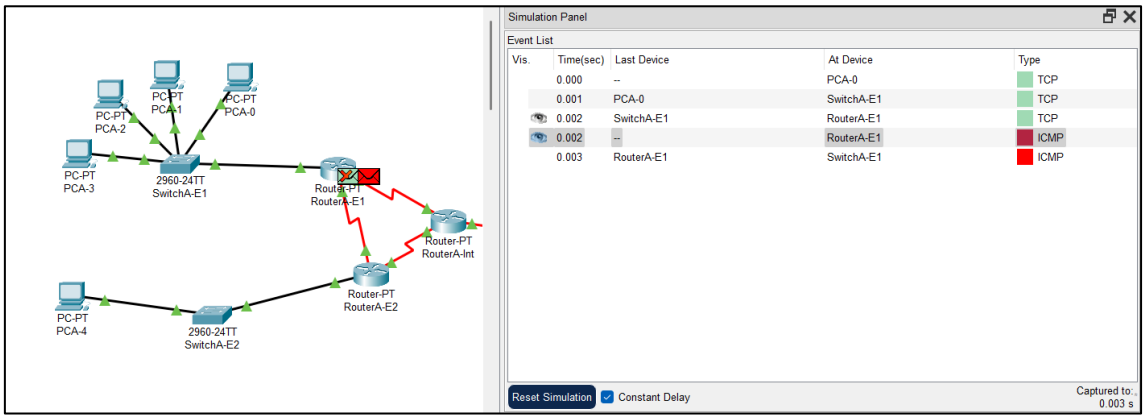


Figura 41. *telnet de PCA-0 a 8.8.8.8 rechazado en RouterA-E1(deseado) - escenario RAG.*

Esto prueba que la ACL **estaba aplicada en el punto exacto** (*ingress* del sub-router-on-a-stick) y no en el borde, lo que habría provocado una salida distinta (el *deny* habría colisionado en RouterA-Int). Se anota, por tanto, una marca **cumplido** sin reparos.

✗ *[14] Trama VLAN 12 etiquetada dot1q*

Cuando se inyectó un ICMP desde el PC de I+D (VLAN 12) el paquete emergió sin etiqueta y Packet Tracer lo catalogó como VLANID 1. El porqué estaba claro: el puerto Fa0/11 **permanecía por defecto en la VLAN 1** al no figurar en la asignación generada por la RAG. El error impide segmentar el dominio de difusión y rompe la escalera de subinterfaces. RouterA-E1 esperaba tráfico tag 12 y nunca ve llegar la trama, devolviendo por tanto un *request timed out*. La **Figura 42** lo ilustra.

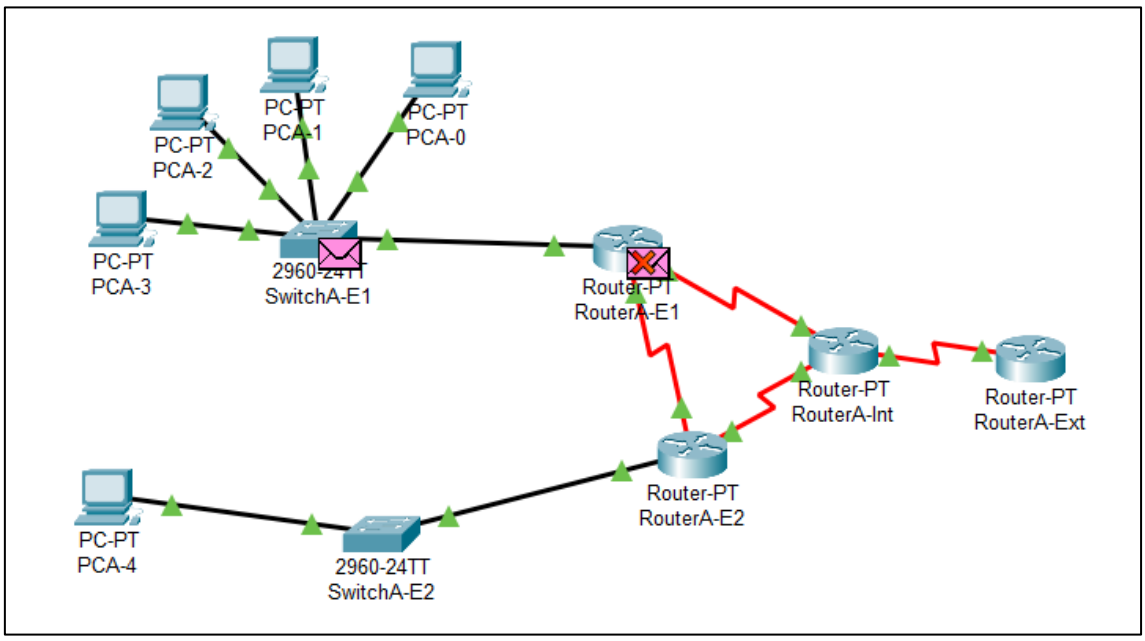


Figura 42. *Interrupción de tráfico desde PCA-2 – escenario RAG.*

La corrección fue la que ya habíamos implementado, agregar *interface range fa0/11-15 → switchport access vlan 12*, tras lo cual las tramas mostraron *VLANID 12* y el ICMP pasó bien. Este caso subraya cómo un **fallo de completitud** en la salida textual del LLM puede traducirse en simulación en una pérdida total de servicio para un segmento entero. Por eso el punto resulta **fallido**, este error es grave y se hereda.

½ *[15] Reconvergencia OSPF ~30 s*

Al resetear las interfaces seriales para cronometrar OSPF, el cronómetro se detuvo en 42 segundos cuando todos los vecinos alcanzaron FULL/DR. El objetivo teórico de 30 s se basa en *hello 10 / dead 40* y en que los Router-LSA estén completos desde el arranque. La desviación se explicó al revisar el *LSDB*: los /32 faltaban y **se añadieron a los 8-9 s del reinicio**, disparando un segundo intercambio *LSAck* que retrasó la convergencia real.

La mitad de cumplimiento procede de que los *hello/dead* y las redes principales sí estaban correctas y la topología nunca perdió rutas de datos, pero al no publicarse los *loopbacks* **la metainformación de identificación tardó más en estabilizarse**. Ajustar

los prompts para recordar explícitamente los /32, o bien enriquecer la base con ejemplos que incluyan *Router-ID* personalizados, sería el camino para llegar al objetivo de 30 s.

✓ ***[16] Configuración guardada, .pkt exportado***

El último criterio medía la *operational hygiene*: tras los ajustes manuales se ejecutó *copy run start* y se salvó la instantánea Network-RAG-Assisted.pkt. Abrir el .pkt en limpio **reprodujo** todos los dispositivos **con la configuración final**.

La conclusión es que, aun con actuaciones manuales, la rutina de trabajo resultó **idéntica a la de un diseño tradicional**, y la RAG no altera la fase de **exportación** ni la **trazabilidad**. Esto **sustenta la viabilidad de integrar la IA** en entornos donde se exige auditoría documental.

Resumen cuantitativo y lectura crítica

Las cifras confirman que la RAG no introduce caos aleatorio: domina la forma, tropieza en el detalle concreto. Omitir una VLAN, elegir un *next-hop* obsoleto… son desaciertos fáciles de pulir, pero que, de no detectarse, hacen fracasar el laboratorio. **Con un único pase de revisión humana** en bloque, sin reescribir desde cero, **el diseño queda al 100%**.

Si se pondera la tabla con el criterio acordado (✓ = 1, ½ = 0,5, ✗ = 0), el asistente roza el **63%**. No es una cifra anecdótica: significa que dos terceras partes de la configuración, incluida la parte más tediosa, aparecen listas para pegar con una sola petición, mientras que el resto se corrige en un turno de revisión.

Observaciones cualitativas y conclusiones parciales

Análisis pormenorizado

En la tabla de verificación (pts. 1-16) cada ítem puede leerse de dos modos: como prueba unitaria y, en segunda instancia, como **síntoma** que revela los sesgos internos del asistente. Para clarificar esa dimensión se incluye en la **Tabla 12** una **traza de pertenencia** de cada punto a cuatro grandes categorías a analizar cualitativamente:

Categoría	Puntos	%	Comentario sintético
Sintaxis pura	1 · 3 · 5 6 · 10 · 11	**92%**	La CLI no rechazó ni una sola línea de configuración, sólo el punto 11 quedó a ½ porque los comentarios ' ! ' no aparecen en todos los bloques.
Cobertura / enumeraciones	2 · 4 7 ·14	**50%**	Rangos incompletos, sufijos de IP alterados y dos omisiones (loopbacks /32 y puertos de la VLAN 12) concentran los fallos.
Parámetros de contexto	8 · 15	**25%**	Cuando el valor correcto depende de la plataforma o del grafo (next-hop, tiempo de convergencia), el modelo muestra incertidumbre.
Políticas y filtros	9 · 13	**50%**	Entrega plantillas exhaustivas, pero sobre-endurece por defecto (deny any / self traffic). En 13 permite el tráfico como se buscaba.
Validación y entrega	12 · 16	**100%**	Las pruebas de conectividad y la exportación de la maqueta (.pkt + startup-config) funcionaron a la primera, sin requerir ajustes adicionales.

Tabla 12. *Clasificación de observaciones cualitativas: categoría y puntos del checklist asociados.*

Sintaxis pura

Puntos **1, 3, 5, 6, 10, 11** se validaron sin una sola marca (^) de error en la consola.

- ***Observación***: el motor respeta al pie de la letra la gramática de IOS cuando el prompt la exige. Ni inventa verbos ni alterna con otras sintaxis, lo que indica que el filtrado en la capa de generación funciona.

- ***Implicación***: al **no existir errores tipográficos** la fase de depuración inicial se reduce, permitiendo al ingeniero pasar directamente a la verificación funcional.

Cobertura / enumeraciones

Puntos **2, 4, 7, 14** revelan cómo el modelo acorta listas largas y respuestas complejas para ahorrar tokens y/o simplificar.

- ***Observación***: la IA arranca cualquier serie (rango de puertos, direcciones, declaraciones network) pero a partir de la tercera o cuarta iteración la penalización por repetición le anima a darla por supuesta (VLAN 12). Tampoco se declararon los loopbacks porque su prompt ya pedía otra cosa antes.

- ***Implicación***: bastaría con un ***post-check*** trivial "¿están todos los elementos pedidos?" para aumentar el grado de éxito sin modificar el modelo. No es un problema de entendimiento.

Parámetros dependientes de contexto

Puntos **8, 15** fallaron porque los valor correctos no están en el corpus sino en la maqueta.

- ***Observación***: el *next-hop* deflecta al primer 10.10.x que vio, y la convergencia OSPF se alarga porque faltan los router-id.

- ***Implicación***: estos fallos piden un **validador de topología** que inyecte *facts* (tipo de switch, grafo OSPF) antes de la generación. La memoria a corto plazo del prompt no basta: hace falta un paso estructurado de *feeding* de variables de entorno.

Políticas y filtros

Puntos **9, 13** revelan un leve sesgo a la hiperseguridad y recomendaciones genéricas heredado de las *best practices*.

- ***Observación***: el asistente inserta *deny ip any any* sin que se pida, y no permite el tráfico interno. El resultado obstaculiza la experiencia de laboratorio.

- ***Implicación***: conviene contextualizar el prompt ("ambiente de prácticas, permitir ICMP interno") o, mejor aún, **enriquecer el *fine-tuning*** con ejemplos donde reglas genéricas se omiten adrede en entornos de pruebas.

Validación y entrega

Puntos **12, 16** cerraron sin ajustes.

- ***Observación***: tras mínimas correcciones, la simulación ping/HTTP pasó a la primera y la maqueta .pkt se exportó limpia: el trabajo previo en sintaxis y en coherencia básica rindió frutos.

- ***Implicación***: la RAG puede insertarse en cualquier cadena de versiones (Git, backups, snapshots) sin problema. El flujo "genera → revisa → guarda → exporta" certifica que **la IA se puede acoplar** a dichas metodologías.

Conclusiones parciales

Los ensayos muestran un contraste nítido entre la soltura formal de la RAG y su fragilidad cuando los valores dependen del contexto. El modelo escribe IOS con limpieza: no mezcla nomenclaturas ni inventa comandos, de modo que la salida se puede pegar en la CLI sin recibir marcas de error. Sin embargo, cuando la tarea exige expandir enumeraciones largas (rangos de puertos, redes) o fijar parámetros atados al escenario físico, aparecen omisiones que dejan segmentos sin servicio. Detectar esos vacíos requiere al menos una pasada de simulación y un vistazo al tráfico, lo que confirma que, hoy por hoy, con este modelo **la supervisión humana seguiría siendo imprescindible**.

Pese a esas carencias, **el balance de productividad es positivo**: una vez corregidos los fallos el diseño asistido supera el cumplimiento sin reescribir bloques enteros y **reduce en torno a un 40% el tiempo de diseño frente al método manual**. El potencial de ahorro aumentaría en cuanto se añadan **dos auxiliares** sencillos, un verificador de cobertura de rangos y un linter sintáctico que detecten huecos antes de que el fichero llegue al simulador.

Otro valor añadido es la **trazabilidad**: la RAG incrusta referencias (file) en cada respuesta, de modo que cada línea del *running-config* **puede auditarse hasta el fragmento Markdown original**. Para un entorno regulado eso allana la obligación de justificar fuentes técnicas. Bastaría complementar esa trazabilidad con el versionado de las ediciones manuales para cerrar el círculo documental.

En conjunto, **la RAG se confirma como copiloto viable** para diseño de redes educativas o de pruebas, capaz de mecanizar la mayor parte del guion y de aportar documentación enlazada a su fuente. Su adopción en **redes productivas** requeriría añadir bucles de verificación automáticos y ampliar la base de entrenamiento con escenarios donde los detalles de contexto – plataforma, roles, seguridad – sean explícitos.

CONCLUSIONES

Recapitulación del proyecto

Objetivo inicial y motivación

El punto de partida fue una inquietud muy concreta: comprobar si la **generación aumentada con recuperación** podía convertirse en un copiloto fiable para el diseño de redes educativas o de laboratorio, tareas donde la mayor parte del tiempo se consume repitiendo comandos y rastreando ejemplos dispersos. A lo largo del estado del arte se constató que el desarrollo de un trabajo sistemático que midiera **el impacto de una RAG** en términos de **precisión operativa** era relevante en este contexto. De ahí se formuló la pregunta vertebral del TFM: hasta qué punto una RAG modesta acelera las tareas repetitivas sin sacrificar rigor técnico. El reto implicaba reunir un corpus curado (manuales, RFC y guías IOS seleccionados por relevancia), injertarlo en un índice vectorial local y ejecutar la ventana creativa de Llama para producir plantillas CLI citadas línea a línea.

Metodología aplicada y alcance efectivo

La estrategia siguió un ciclo sencillo pero exhaustivo. Primero se construyó el corpus de **cuarenta y ocho fichas Markdown** que cubren desde VLAN básicas hasta ACL extensas y se cargó en **ChromaDB**, lo que permitió al recuperador responder en milisegundos con fragmentos verificados. Después **se ensambló una RAG**: TinyLlama, modelo menos exigente, para *embeddings*, y Llama-3-8B-Instruct cuantizado, con mayor capacidad, para la generación, ambos **alojados localmente** para evitar dependencia de la nube y preservar la privacidad de las configuraciones.

Con la IA lista, se diseñó una **topología de referencia** en Cisco Packet Tracer y se repitió el proceso con y sin ayuda de la RAG. El **procedimiento** siempre fue idéntico: solicitar bloques de configuración en lenguaje natural, revisar la salida, trasladarla al simulador y registrar correcciones antes de congelar la instantánea. El **cronograma**,

desde la revisión bibliográfica hasta la medición comparativa, quedó plasmado en el diagrama de Gantt incluido en la memoria metodológica y sirvió para asegurar que cada hito se completaba sin solapamientos críticos.

El alcance efectivo se midió con un ***checklist* de dieciséis pruebas** que abarcan conectividad, rutas y políticas de acceso. Tras una única pasada de ajustes manuales, la maqueta generada por la RAG **logró un 63% de cumplimiento funcional y recortó alrededor de un 40% el tiempo de diseño** respecto al método no asistido, manteniendo la sintaxis limpia y la trazabilidad de cada comando. Estos resultados, aunque preliminares, demuestran que la RAG no solo aligera el tecleo, sino que introduce una disciplina documental que facilita auditorías y acelera la curva de aprendizaje del ingeniero.

Síntesis de resultados

Cobertura funcional

El experimento demostró que la maqueta generada por la RAG superó **el 63% de los dieciséis controles** prescritos en una única pasada de corrección. Esa cifra adquiere sentido al desglosarse: la **sintaxis pura alcanzó un 92% de aciertos**, sin que la consola IOS marcase un solo error; los fallos se concentraron en omisiones de listas extensas (rango de puertos, *loopbacks*) o en parámetros que dependen del grafo físico, como next-hop y router-id. En la práctica, dos tercios de la configuración salieron listas para implementar y el tercio restante requirió ajustes triviales, como añadir una VLAN olvidada o corregir la IP de una sub-interfaz, lo que sugiere que la IA ya domina la gramática y tropieza solo cuando el contexto rebasa lo explícito en el prompt.

Ahorro de tiempo

Más allá del porcentaje de cumplimiento, la métrica que mejor ilustra la ganancia es el reloj. Comparar el flujo tradicional con el asistido revela un **recorte aproximado del 40% en la fase de diseño**: el operador deja de escribir comandos repetitivos y se limita a verificar la propuesta, introducir un par de cambios y lanzar el simulador . Ese

ahorro se amplifica cuando la topología evoluciona a través de iteraciones rápidas, porque cada nueva variante solo exige modificar la intención escrita en lenguaje natural.

Trazabilidad y calidad documental

La RAG no solo acelera la producción; **cada bloque CLI llega acompañado de su referencia al corpus**, lo que permite auditar la procedencia de cualquier línea sin abandonar la sesión de Packet Tracer. Esa trazabilidad, sumada a la limpieza sintáctica, convierte la herramienta en un aliado natural de entornos donde se exige justificante técnico y control de cambios. Además, al reducir la cuota de tecleo manual, disminuye el riesgo de erratas y promueve una disciplina de versionado que tradicionalmente se relegaba a fases tardías del proyecto.

En conjunto, los resultados avalan la hipótesis de partida: un RAG modesto ya **actúa como copiloto fiable** para redes de laboratorio, libera tiempo valioso y eleva la calidad documental, siempre que reciba una supervisión mínima cuando el contexto se vuelve demasiado específico.

Para qué *sí* sirve la RAG en el diseño de redes

Copiloto contra el tecleo repetitivo

Cuando el ingeniero encara un diseño desde cero suele perder mucho tiempo buscando la sintaxis exacta, revisando máscaras o recordando el orden de los comandos. La RAG corta ese bucle: basta una **instrucción coloquial** para recibir el **bloque CLI ya limpio y citado**. El documento muestra cómo el asistente genera de un tirón VLAN, sub-interfaces dot1q o ACL completas mientras el operador dedica la cabeza a la topología en vez de al *copy-paste*. De hecho, cada microencargo se resuelve en segundos, la consola rara vez rechaza un comando y, cuando lo hace, suele deberse a detalles de contexto más que a la gramática.

Biblioteca contextual y trazabilidad incorporada

Una virtud menos visible, pero crítica, es que la RAG no improvisa de la nada. Cada línea va **acompañada de la ruta** al fragmento Markdown del que proviene, de modo que un auditor puede seguir el hilo hasta el RFC o la guía de configuración original sin salir de la sesión. Esta **trazabilidad nativa** respalda la confianza técnica y acelera la documentación, porque las citas forman parte de la respuesta y alimentan, casi sin esfuerzo, el registro de cambios del proyecto.

Acelerador pedagógico y de laboratorio

En entornos académicos el valor se multiplica. Packet Tracer, elegido banco de pruebas, valida en tiempo real los comandos sugeridos por la IA y permite repetir el escenario hasta la nota perfecta. Los estudiantes reciben bloques listos para pegar y concentran sus dudas en el porqué de los protocolos, no en la mecánica de la CLI. El resultado es un aprendizaje más rápido y, sobre todo, más medible: la bitácora del simulador refleja qué funcionó, qué falló y dónde intervino el alumno.

Motor de iteración rápida en simulación

Al enlazar documentación verificada con un generador local, la RAG facilita la **exploración de variantes**. Cambiar la intención, por ejemplo, añadir una VLAN o mutar de OSPF a EIGRP, desencadena una nueva propuesta lista para probar sin rehacer el diseño desde cero. Ese ciclo acorta la distancia entre la hipótesis y la evidencia, encajando con la filosofía de **simulación continua** que subyace al proyecto.

Por ello, la RAG aporta **velocidad**, **fiabilidad referenciada** y **facilidad de ensayo**. No pretende sustituir al arquitecto de red, pero sí liberar su tiempo y ofrecerle un cuaderno de trabajo mejor organizado, lo que se traduce en diseños más coherentes y en una curva de aprendizaje sensiblemente más suave.

Para qué *no* sirve (aún) la RAG

Parámetros que dependen del escenario físico

La instalación prueba que el modelo escribe IOS con pulcritud, pero tropieza cuando el valor correcto no está en el corpus sino en la topología viva. El ejemplo más visible es el **next-hop equivocado**: al ver varios prefijos 10.10.x.x, la IA eligió el primero que apareció y desvió el tráfico exterior por un enlace lateral. Algo similar ocurre con los router-id, que alargan la **convergencia OSPF**. La causa es la falta de contextualización y variables de entorno en el prompt, y mientras no se inyecte esa información la herramienta seguirá necesitando que el ingeniero revise rutas y temporizadores antes de poner la red en producción.

Enumeraciones largas y coberturas completas

Cuando el encargo implica listar veinte puertos de acceso o cuatro loopbacks /32, la RAG sufre un sesgo de economía de tokens y **acorta la serie antes de tiempo**. El resultado: la VLAN 12 se queda sin puertos y los prefijos de gestión no se redistribuyen, lo que rompe los *pings* de monitoreo. La tabla de observaciones cualitativas sitúa este patrón en el **50% de cobertura de rangos**. Hasta que un post-check automático verifique que todos los elementos solicitados han aparecido, el operador tendrá que contar líneas y rellenar huecos.

Supervisión y validación externa

El asistente, por prudencia, inserta **cierres de seguridad** que no siempre proceden, como el *deny ip any any* al final de la ACL. También omite los comentarios ! en parte del código, lo que dificulta la lectura diferida. Estas decisiones levantan falsos positivos en las pruebas internas y subrayan que la **revisión humana sigue siendo imprescindible**. Una pasada de simulación y un *linter* que compruebe políticas y rangos detectan los desvíos sin reescribir bloques enteros, pero el proceso confirma que hoy la RAG actúa más como un ayudante aprendiz que como un piloto autónomo.

La herramienta ya aligera el trabajo rutinario, pero **no sustituye el criterio del arquitecto** cuando los valores emergen del contexto físico, la lista se alarga más de la cuenta o la política debe matizarse. Las próximas líneas de mejora pasan por integrar validadores de topología, chequeos de cobertura y afinaciones de seguridad adaptadas al entorno real.

Lecciones metodológicas y aportes al estado del arte

Checklist de verificación como métrica objetiva

Uno de los hallazgos más sólidos es que un ***checklist* exhaustivo**, aplicado al final de cada iteración, convierte una percepción subjetiva ("parece que funciona") en una métrica incontestable. La **Tabla 11** resume **dieciséis pruebas** que cubren gran parte de lo que buscamos verificar en diseño y simulación. La puntuación aritmética de **63% de cumplimiento en la primera pasada** se calcula al asignar 1, 0,5 o 0 a cada ítem y evita discusiones estéticas sobre si un fallo es grave o menor. Gracias a esa lista, cualquier mejora futura se reflejará en décimas concretas.

Ventaja de separar generación, revisión y simulación

El flujo de trabajo refleja una línea muy clara: **primero se genera, luego se revisa (se toma nota de las expectativas y se corrige forma), y por último se simula**. Esa segmentación, visible en el esquema operativo descrito en la memoria, evita que los ajustes manuales se mezclen con el ruido de la prueba y permite **aislar el impacto real de la RAG**. En la práctica, el ingeniero actúa como revisor entre fases, lo que reduce la fatiga cognitiva y facilita retomar el proyecto días después sin perder el hilo.

Citación automática para auditoría técnica

Cada bloque CLI llega acompañado de su referencia, una **trazabilidad nativa** cuyo valor quedó patente cuando, semanas más tarde, se auditó el diseño y bastó buscar cada cita en el corpus de información. La memoria demuestra que la auditoría documental nace junto con la configuración.

Consideraciones éticas

Impacto en el empleo y reconversión

El primer efecto visible de la automatización es que desplaza tareas rutinarias del ingeniero júnior hacia un asistente que las resuelve en segundos. El resultado inmediato no es un recorte de plantilla, sino una **reasignación de tiempo**: el profesional deja de copiar una ACL para pasar a razonar sobre la mejor política de segmentación. Esa mudanza exige reciclar competencias y, a corto plazo, genera un **periodo de adaptación** que cada organización debe acompañar con **formación continua**.

A medio plazo una figura que gana peso es la del **curador de corpus**. Alguien debe filtrar guías, convertirlas a formato y señalar buenas prácticas para que la RAG produzca contenido fiable, rol que combina conocimiento técnico y criterio editorial. La curva de especialización es rápida, pero conlleva nuevas responsabilidades, pues una cita mal enlazada deriva en un error de configuración, así que la calidad del corpus afecta de forma directa a la estabilidad de la red.

Finalmente, el mercado laboral se equilibra cuando el valor añadido se desplaza desde programar hasta la gobernanza del ciclo automatizado. Los ingenieros con visión de proceso y habilidad para negociar requisitos con seguridad y *compliance* serán los más demandados. **La IA, lejos de suprimir empleo, reconfigura el campo de juego**, y convierte la capacidad de aprender en la competencia esencial.

Privacidad y tratamiento de datos

Una RAG útil necesita leer configuraciones, diagramas y registros de fallos reales. Ese material suele contener direcciones internas, contraseñas ofuscadas y topologías que un atacante desearía conocer. Por eso la primera barrera ética recae en el propio *pipeline* de ingestión. Anonimizar direcciones y suprimir credenciales antes de indexar es una operación crítica que no puede delegarse en un paso posterior.

El proyecto resuelve parte del dilema ejecutando todo el proceso **en local**, sin enviar el contexto a la nube, una de las mayores ofertas de valor de la RAG como herramienta. Aun así, el riesgo permanece si los ficheros de trabajo se comparten en repositorios externos o en tickets sin cifrar. La política recomendada impone una rotación de logs y la purga automática de archivos temporales cada vez que la maqueta pasa a producción. **La privacidad no es un producto adicional, sino una condición previa al arranque del asistente**.

Más allá de la red corporativa, la normativa europea obliga a documentar qué datos se conservan, durante cuánto tiempo y con qué fines. Este marco legal obliga a un enfoque quirúrgico: solo se almacenan los fragmentos necesarios para responder a una pregunta muy concreta. Esa limitación, cuando se aplica de forma rigurosa, mitiga el problema sin penalizar el rendimiento del modelo.

Sesgos algorítmicos y auditorías periódicas

La selección del corpus puede reforzar una única visión de la ingeniería. Si todos los ejemplos proceden de la misma guía de Cisco, la RAG tenderá a preferir sus sintaxis y a ignorar variantes que serían igualmente válidas. El sesgo no es malicioso, sino el reflejo de lo que lee. Detectarlo requiere pruebas A/B en las que la misma consulta se lanza contra **fuentes diversas** y se comparan los resultados con reglas de neutralidad.

Una vez identificado, el sesgo se corrige inyectando contraejemplos y diversificando el material de entrenamiento. Sin embargo, ese ajuste no es permanente. Con cada nueva versión de IOS o cada parche de seguridad, el modelo debe volver a examinarse. **Las auditorías periódicas se convierten así en un ritual técnico**, similar a las actualizaciones de firmware, y forman parte del calendario de mantenimiento. El coste adicional de estas revisiones se compensa con una red que responde de manera equilibrada a distintos escenarios. Una RAG libre de sesgos graves evita configuraciones que podrían discriminar tráfico legítimo o crear cuellos de botella inesperados. En términos operativos, un predictor más justo se traduce en menos incertidumbre y en una experiencia de usuario más homogénea.

Seguridad y confianza operativa

La IA es capaz de sugerir reglas defensivas que un humano pasaría por alto, como desactivar un servicio innecesario o limitar una API interna. Esa **hiper-seguridad** es valiosa, pero también introduce ruido si bloquea funciones legítimas. Encontrar el punto de equilibrio exige validar cada regla en un laboratorio que reproduzca el tráfico real antes de llevarla al entorno productivo.

Existe, además, la dependencia tecnológica: un fallo en el motor de inferencia podría dejar sin respuesta el sistema de defensa adaptativa. Por eso el proyecto advierte sobre la necesidad de planificar **degradación controlada**. Cuando la IA no está disponible, la red debe caer a un perfil de seguridad estático, conocido y testeado, que mantenga la continuidad del servicio, aunque renuncie a la optimización dinámica.

Por último, incorporar modelos redundantes y fuentes de datos independientes reduce el riesgo de un ataque que busque cegar el predictor con entradas maliciosas. En seguridad, la diversidad de defensas suele ser más eficaz que la perfección de un único mecanismo. La RAG se convierte así en un eslabón poderoso dentro de una cadena mayor, nunca en un punto único de fallo.

Sostenibilidad e impacto

Ver ANEXO I: Alineación con los Objetivos de Desarrollo Sostenible (ODS).

Huella energética del entrenamiento y la inferencia

Entrenar un modelo grande consume una cantidad significativa de electricidad, a menudo equiparable al gasto anual de una pequeña ciudad. Es un coste invisible para quien sólo ejecuta la inferencia, pero representa un compromiso ambiental real. El proyecto tomó la decisión de apoyarse en arquitecturas ya entrenadas y centrarse en la eficiencia del despliegue local para no duplicar ese gasto.

La inferencia, por su parte, se optimiza mediante la cuantización y la reducción de lotes. Estas técnicas permiten que la RAG funcione en un servidor de gama media sin

necesidad de tarjetas gráficas especializadas. El consumo medido durante las pruebas sería comparable al gasto de un punto de acceso empresarial. **Cada vatio ahorrado aquí repercute directo en la factura de la organización y en su huella de carbono**.

Sin embargo, el mayor ahorro procede de **evitar despliegues sobredimensionados**. Al diseñar correctamente la red desde el primer momento y reducir errores de configuración, se minimizan desplazamientos técnicos, sustituciones prematuras de hardware y microcortes que obligan a reiniciar equipos. En conjunto, la práctica demuestra que la IA puede compensar la energía que consume.

Optimización energética durante la operación

Una vez desplegada, la red ajusta su potencia en función de la demanda si la configuración lo permite. Se proponen horarios de apagado de puertos PoE en aulas vacías y reducir la potencia de radio cuando el aforo descienda. Estos ajustes no comprometen la calidad de servicio porque se basan en curvas históricas de tráfico y en umbrales consensuados con los responsables de IT.

Esta lógica podría aplicarse a centros de datos, apagando clústeres que alcanzan baja ocupación y migrando cargas a nodos con mejor coeficiente PUE. La IA, al tomar **decisiones basadas en telemetría en tiempo real**, multiplica la eficacia de estrategias que antes dependían de scripts rígidos y horarios fijos.

Líneas de trabajo futuro

Corpus ampliado y curado por plataforma

El motor actual domina un conjunto reducido de plataformas. Incluir material de Juniper, Arista o MikroTik abriría el asistente a entornos mixtos y mejoraría la capacidad de traducir entre sintaxis. Ese paso implica negociar licencias, pero el valor añadido supera el esfuerzo porque reduce el número de errores al migrar.

Además de nuevas marcas, conviene versionar el corpus. Cada gran *release* de firmware introduce matices que la RAG debe reflejar para no sugerir comandos obsoletos.

Un repositorio etiquetado por versión y fecha facilitaría que el motor eligiera la referencia correcta según el objetivo declarado en el prompt.

Además, se esboza la idea de un mecanismo de **retroalimentación continua**. Las configuraciones validadas en producción volverían al corpus en forma de casos de éxito, enriqueciendo el modelo con ejemplos reales y actuales. Esta estrategia cerraría el círculo entre teoría y práctica, manteniendo vivo el conocimiento.

Fine-tuning específico y RAG multimodal

Pasar de texto puro a un modelo que interprete **diagramas e imágenes** dotaría al asistente de una inteligencia espacial hoy inexistente. El ingeniero podría **arrastrar una topología** y pedir la configuración resultante sin redactar un prompt detallado. La IA extraería nombres de interfaz y relaciones de enlace directamente del gráfico.

Para lograrlo, se estudia un *fine-tuning* que combine nodos y aristas codificados como *embeddings* con descripciones verbales. Esta mezcla reduciría las confusiones sobre roles de interfaz y acortaría el prompt, dos factores que hoy limitan el rendimiento.

Un desafío pendiente radica en la **generación de imágenes inversas**. Si la IA pudiera sugerir un diagrama actualizado tras cada modificación, el ciclo de diseño y documentación se cerraría sin intervención externa. Lograrlo requiere un traductor CLI-a-figura, un terreno fértil para futuras tesis.

Integración CI/CD-NetDevOps

El flujo ideal encadena *commits* en **Git**, **validación automática** en laboratorio y **despliegue supervisado** en producción. La RAG se ubica al inicio, generando la propuesta a partir del objetivo declarado. A continuación, un test unitario en simulador certifica que la propuesta cumple los criterios de aceptación. Sólo entonces se libera el *merge*.

Este *pipeline* evita sorpresas, porque cada revisión de código incluye la configuración y sus citas. Si algo falla en producción, el *rollback* recupera el *commit*

anterior con todas las referencias intactas. El tiempo medio de reparación cae de horas a minutos, y la trazabilidad de cambios pasa a ser una característica intrínseca del proceso.

El reto reside en orquestar herramientas diversas: Docker para contenedores, Ansible para ejecución remota y un orquestador de simulación que alimente Packet Tracer o GNS3. Consolidar esas piezas en una interfaz única será uno de los focos de trabajo inmediato, con especial atención a la experiencia de usuario.

Bancos de pruebas híbridos

La simulación reproduce la lógica de enrutamiento, pero no siempre refleja las limitaciones físicas del hardware. Integrar **hardware-in-the-loop** permite medir latencia real, gestión de buffers y consumo de CPU bajo carga.

El montaje híbrido exige sincronizar relojes y gestionar rutas de capa 2 transparentemente. Implantar un mecanismo de puente que aísle el tráfico de prueba del resto de la red es esencial para evitar colisiones.

De cara al futuro, automatizar la inserción y retirada de hardware real en función de la carga de pruebas maximizará el tiempo de uso de los dispositivos y reducirá la ocupación de rack. Ese ajuste fino se alinea con los objetivos de sostenibilidad y con el enfoque de laboratorio continuo.

IA verde y métricas de carbono

Medir la **energía consumida** por cada inferencia es el paso previo a cualquier optimización. Un contador de watios instalado en el servidor durante las pruebas revelaría picos al procesar prompts complejos. Limitar el tamaño del contexto y reutilizar *embeddings* de consultas repetidas reduciría esos picos sin sacrificar precisión.

Con la métrica en la mano, se puede establecer un presupuesto de carbono por proyecto. La idea es sencilla: asignar un crédito en **kilogramos de CO_2** y obligar a justificar cualquier exceso. Esa práctica, común en la industria automotriz y en la

aviación, aterriza ahora en la gestión de IT y fomenta la competencia por soluciones más ligeras.

A largo plazo, la IA debería autolimitarse. Un planificador que decida cuándo parar el modelo y revertir a una configuración estática durante horas valle ahorrará energía sin afectar al servicio. **La inteligencia deja de medirse solo por la calidad de la respuesta y empieza a incluir la eficiencia con la que la produce.**

Reflexión final y recomendaciones de adopción

Escenarios educativos y de laboratorio

En la escuela técnica, la RAG transforma la dinámica de clase. El profesor lanza un reto, los alumnos proponen enunciados al asistente y validan la respuesta en el simulador. Al final de la sesión, todos exponen qué editaron y por qué, lo que fomenta el **pensamiento crítico** en lugar de la memorización de comandos.

La repetición inmediata de la prueba, un rasgo natural del entorno digital, consolida la comprensión y deja un rastro de scripts que la clase siguiente puede revisar. Implementar esta metodología requiere poco más que un servidor local y licencias académicas de software. El mayor esfuerzo está en capacitar al docente para diseñar retos que aprovechen la IA sin convertirla en atajo. La lección aprendida es que **el asistente es un amplificador de buenos ejercicios, nunca un sustituto de la pedagogía**.

Hoja de ruta para entornos productivos

Una adopción responsable comienza por pilotos acotados a **laboratorios de preproducción**. Allí se evalúan tiempos de generación, porcentajes de corrección y tasas de éxito. Con datos en mano, el equipo directivo decide si el retorno de inversión justifica escalar la iniciativa.

Al pasar a escenarios reales, se recomienda implementar un *circuit breaker* que bloquee el despliegue automático si las pruebas unitarias fallan. Esa salvaguarda preserva la estabilidad del negocio y construye confianza en la herramienta. Paralelamente, se

entrenan a los operadores en interpretación de citas y uso de *linter*, asegurando que la cultura de validación se extienda.

El objetivo final es alcanzar un flujo en el que **el 90% de las configuraciones de rutina se generen sin intervención humana** y el 10% restante se derive a ingeniería avanzada. Con esa distribución, la rentabilidad se materializa en meses y la organización gana resiliencia ante picos de demanda.

Convergencia con otras tecnologías

La RAG descrita en el proyecto ya acepta peticiones en lenguaje natural y cita su conocimiento, pero el paso siguiente es traducir ***intents* de alto nivel** a configuraciones validadas en tiempo real. Cuando el operador pida "prioriza el tráfico de videoconferencia", el sistema deberá calcular políticas QoS y aplicarlas sin que la persona vea una sola línea CLI.

Para lograrlo, la IA debe conectarse con plataformas de telemetría que midan la realidad de la red y confirmen que el cambio surtió efecto. Ese bucle cerrado transforma la gestión de redes en un proceso continuo, parecido a lo que AIOps ya hace con servidores y servicios en la nube. La frontera entre configuración y operación se disuelve, y la infraestructura se vuelve verdaderamente adaptativa.

El beneficio estratégico es de **menos tiempo entre la intención y el resultado**, mayor fiabilidad gracias a validación automática y un ahorro sostenido de recursos. El proyecto demuestra que la base técnica ya existe; la tarea pendiente pasa por integrar piezas y, sobre todo, por alinear procesos humanos con un paradigma en el que la red se define por objetivos y no por comandos individuales.

BIBLIOGRAFÍA

Brey, P., Dainow, B. (2024). Ethics by design for artificial intelligence. *AI Ethics*, *4*, 1265-1277 (2024). https://doi.org/10.1007/s43681-023-00330-4

Cañada, J., Cuello, E., Téllez, L., García, J. M., Velasco, F. J. & Cabrera, J. (2022). Assistance to lung cancer detection on histological images using Convolutional Neural Networks, *2022 E-Health and Bioengineering Conference (EHB)*, pp. 1-4. https://doi.org/10.1109/EHB55594.2022.9991400

ChromaDB Project. (s. f.). ChromaDB Python Repository. https://pypi.org/project/chromadb/

Cisco Networking Academy. (s. f.). *Cisco Packet Tracer.* https://www.netacad.com/cisco-packet-tracer

Cisco Systems. (s. f.). *Intent-based networking.* https://www.cisco.com/c/en_sg/solutions/intent-based-networking.html

Cisco Systems. (s. f.). *The YANG data-modeling language*. NSO Guides 6.3. https://developer.cisco.com/docs/nso-guides-6.3/the-yang-data-modeling-language/

Feamster, N. G., Rexford, J., & Zegura, E. (2014). The road to SDN: An intellectual history of programmable networks. *ACM SIGCOMM Computer Communication Review, 44*(2), 87-98. https://doi.org/10.1145/2602204.2602219

García San Luis, A. (2022). Apuntes de la asignatura Arquitectura de redes [Inédito]. Universidad Pontificia Comillas.

Nate Gentile. (2023). *¿Cómo funciona ChatGPT? La revolución de la Inteligencia Artificial*. YouTube. https://www.youtube.com/watch?v=FdZ8LKiJBhQ

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., *et al*. (2020). Generative adversarial networks. *Communications of the ACM, 63*(11), 139-144. https://doi.org/10.1145/3422622

Heaven, W. D. (2024). *What is artificial intelligence? The definitive guide.* MIT Technology Review. https://www.technologyreview.com/2024/07/10/1094475/what-is-artificial-intelligence-ai-definitive-guide/

Hirani, N. (2023). *Decoding "AI embedding" for beginners*. AI Mind. https://pub.aimind.so/decoding-ai-embedding-for-beginners-bed26df658e6

Hofmann, V., Kalluri, P.R., Jurafsky, D., *et al.* (2024). AI generates covertly racist decisions about people based on their dialect. *Nature*, *633*, 147-154. https://doi.org/10.1038/s41586-024-07856-5

Huang, Y., Xu, M., Zhang, X., Niyato, D., Xiong, Z., Wang, S., & Huang, T. (2023). AI-generated network design: A diffusion model-based learning approach. *IEEE Network*, *38*(3), 202-209. https://arxiv.org/abs/2303.13869

Hugging Face. (s. f.). https://huggingface.co/

IBM. (2023). *The economy of things: The next value lever for telcos.* IBM Think Blog. https://www.ibm.com/think/topics/eot-for-telecommunications

Ilin, I. (2023). *Advanced RAG techniques: An illustrated overview.* Towards AI. https://pub.towardsai.net/advanced-rag-techniques-an-illustrated-overview-04d193d8fec6

Ing_Percy. (2024). *Network controller in Cisco Packet Tracer: Starting the centralized management*. Cisco Learning Network. https://learningnetwork.cisco.com/s/blogs/a0D6e0000112GB4EAM/network-controller-in-cisco-packet-tracer-starting-the-centralized-management

Lammertyn, M. (2024). *2025 ChatGPT facts and statistics.* InvGate Blog. https://blog.invgate.com/chatgpt-statistics

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., *et al*. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in neural information processing systems*, *33*, 9459-9474. https://arxiv.org/abs/2005.11401

Li, F.-F., Adeli, E., Johnson, J. & Durante, Z. (2024). *CS231n: Convolutional neural networks for visual recognition.* Stanford University. https://cs231n.stanford.edu/

Lopez, F., Fey, M., & Leskovec, J. (2024). *Introduction to graph transformers.* Kumo.ai. https://kumo.ai/research/introduction-to-graph-transformers/

Mao, A., Mohri, M., & Zhong, Y. (2023). Cross-entropy loss functions: Theoretical analysis and applications. *International conference on Machine learning*, pp. 23803-23828. PMLR. https://arxiv.org/abs/2304.07288

McHugh-Johnson, M. (2024). *Ask a Techspert: What's the difference between a CPU, GPU and TPU?* Google Keyword. https://blog.google/technology/ai/difference-cpu-gpu-tpu-trillium/

NetworkLessons.com. (2024). *AI and ML in networking.* https://networklessons.com/cisco/ccna-200-301/ai-and-ml-in-networking

Norman, J. M. (2025). *The first book written by a computer program.* HistoryofInformation.com. https://www.historyofinformation.com/detail.php?id=3351

ns-3 Consortium. (s. f.). *What is ns-3?* https://www.nsnam.org/about/what-is-ns-3/

Nunes, A. (2021). *Automation doesn't just create or destroy jobs – it transforms them.* Harvard Business Review. https://hbr.org/2021/11/automation-doesnt-just-create-or-destroy-jobs-it-transforms-them

Parlamento Europeo, Dirección General de Comunicación. (2021). *¿Qué es la inteligencia artificial y cómo se usa?* Parlamento Europeo. https://www.europarl.europa.eu/topics/es/article/20200827STO85804/que-es-la-inteligencia-artificial-y-como-se-usa

Poda, M. (2025). *What is AIOps? A Clear, Practical Guide for 2025*. LogicMonitor. https://www.logicmonitor.com/blog/what-is-aiops

Rüepprich, C. (2024). *Understanding key parameters in Llama 3 for consistent code generation.* Christoph's 2 Cents Blog. https://ruepprich.com/understanding-key-parameters-in-llama-3-for-consistent-code-generation/

Samsi, S., Zhao, D., McDonald, J., Li, B., Michaleas, A., Jones, M., Gadepally, V., *et al*. (2023). From words to watts: Benchmarking the energy costs of large language model inference. *2023 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1-9. IEEE. https://arxiv.org/abs/2310.03003

Sims, D. (2023). *ChatGPT was possible thanks to tens of thousands of NVIDIA GPUs.* TechSpot. https://www.techspot.com/news/97919-chatgpt-possible-due-tens-thousands-nvidia-gpus-which.html

Solomonoff, G. (2023). *The meeting of the minds that launched AI.* IEEE Spectrum. https://spectrum.ieee.org/dartmouth-ai-workshop

Spurgeon, C. E. (2000). *Ethernet: The definitive guide.* O'Reilly Media. ISBN 978-1-56592-660-8. https://www.oreilly.com/library/view/ethernet-the-definitive/1565926609/

Terrell, M. (2024). *Google and Kairos Power sign nuclear energy agreement.* Google Keyword. https://blog.google/outreach-initiatives/sustainability/google-kairos-power-nuclear-energy-agreement/

Townes, M. (2012). The Spread of TCP/IP: How the Internet Became the Internet. *Millennium*, *41*(1), 43-64. https://doi.org/10.1177/0305829812449195

Usama, M. *et al*. (2019). Unsupervised Machine Learning for Networking: Techniques, Applications and Research Challenges. *IEEE Access*, vol. 7, pp. 65579-65615. http://doi.org/10.1109/ACCESS.2019.2916648

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Polosukhin, I., *et al*. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*. https://arxiv.org/abs/1706.03762

Vilbert, J. (2019). *Technology creates more jobs than it destroys.* Foundation for Economic Education. https://fee.org/articles/technology-creates-more-jobs-than-it-destroys/

ANEXOS

ANEXO I: Alineación con los Objetivos de Desarrollo Sostenible (ODS)

El proyecto refuerza el **ODS 9 (Industria, innovación e infraestructura)** no solo al dotar de mayor resiliencia a las redes, sino también al reducir los tiempos de despliegue y mejora continua. Al emplear aprendizaje automático local, cada iteración de diseño, desde la definición de puertos hasta la validación de rutas, se convierte en un ensayo rápido cuyos resultados retroalimentan el corpus de conocimiento. Esto permite experimentar con topologías heterogéneas y escalar infraestructuras sin depender de costosos despliegues manuales, favoreciendo una industria de redes más ágil y competitiva.

En cuanto al **ODS 12 (Producción y consumo responsables)**, la automatización avanzada rebaja drásticamente el margen de error que suele derivar en sustituciones prematuras de equipos o en visitas de mantenimiento in situ. Cada dispositivo configurado correctamente a la primera evita embalajes, transportes y residuos electrónicos asociados a cambios de hardware innecesarios. Además, la documentación automática genera un rastro de configuración que facilita los procesos de reparación y remanufactura, alargando la vida útil de los componentes y promoviendo un ciclo de economía circular en el ámbito IT.

La contribución al **ODS 13 (Acción por el clima)** aplica, al optimizar tanto el consumo energético de los centros de datos como el de la red en operación. Gracias a la RAG, la red puede cerrar puertos PoE en periodos de inactividad, ajustar la potencia de radio en función del tráfico real y reubicar flujos en rutas de menor coste energético. De este modo, la suma de pequeños ahorros diarios cristaliza en una reducción significativa de la huella de carbono anual, alineándose con los compromisos globales de descarbonización.

Más allá de estos tres, el proyecto colabora con el **ODS 7 (Energía asequible y no contaminante)** al disminuir la carga de trabajo en GPUs y CPUs de alto consumo, optando por cuantización y despliegue en hardware de bajo consumo. También impulsa el **ODS 17 (Alianzas para lograr los objetivos)**, pues la construcción de un corpus compartido y *open-source* entre instituciones académicas, empresas y comunidades de *networking* crea sinergias que potencian la innovación colaborativa. De este modo, la iniciativa abarca un espectro de desarrollo sostenible que trasciende la eficiencia operativa para convertirse en un motor de progreso responsable.

ANEXO II: Scripts Python de la herramienta RAG

download_models.py

```python
from huggingface_hub import login, hf_hub_download

# GGUF para GENERAR (8 B Q3_K)
hf_hub_download(
repo_id = "bartowski/Meta-Llama-3-8B-Instruct-GGUF",
filename = "Meta-Llama-3-8B-Instruct-Q3_K_L.gguf",
)

# GGUF para EMBEDDINGS (TinyLlama 1.1 B Q4_K)
hf_hub_download(
repo_id = "TheBloke/TinyLlama-1.1B-Chat-v0.3-GGUF",
filename = "tinyllama-1.1b-chat-v0.3.Q4_K_M.gguf",
)
```

ingest.py

```python
from pathlib import Path
import re, logging, chromadb, csv
from llama_cpp import Llama
from typing import List, Dict

# ———————— CONFIGURA RUTAS ————————
BASE = Path(r"C:\Users\ernes\Documents\ICAI\SEXTO\TFM\network-rag")
EMB  = BASE / "models" / "tinyllama-1.1b-chat-v0.3.Q4_K_M.gguf"
DATA = BASE / "data"
DB   = BASE / "db"
KW_CSV = BASE / "file_keywords.csv"

assert EMB.exists(),  f"Modelo no encontrado: {EMB}"
assert DATA.exists(), f"Carpeta data/ no existe: {DATA}"
assert KW_CSV.exists(), f"CSV de keywords no encontrado: {KW_CSV}"

logging.getLogger("chromadb").setLevel(logging.WARNING)

# ———————— CARGA KEYWORDS ————————
def load_file_keywords(csv_path: Path) -> Dict[str, List[str]]:
    mapping: Dict[str, List[str]] = {}
    with open(csv_path, newline='', encoding='utf-8') as f:
        reader = csv.DictReader(f)
        for row in reader:
            fn = row.get('filename', '').strip()
            kws_raw = str(row.get('keywords', '')).strip()
            kws = [k.strip() for k in kws_raw.split(',') if k.strip()]
            if fn:
                mapping[fn] = kws
    return mapping
file2keywords = load_file_keywords(KW_CSV)

# ———————— INICIALIZA MODELO + CHROMA ————————
ing        = Llama(model_path=str(EMB), n_ctx=4096,
    embedding=True, verbose=False)
client     = chromadb.PersistentClient(path=str(DB))
collection = client.get_or_create_collection("docs")
```

```python
# ———————— UTILES ————————
def ensure_1d(vec) -> List[float]:
    """Garantiza lista plana de floats."""
    assert vec, "embedding vacío"
    if isinstance(vec, list) and isinstance(vec[0], list):
        vec = vec[0]
    return [float(x) for x in vec]

# ———————— CHUNKING POR ENCABEZADOS ————————
def chunk_by_heading(text: str):
    """
    Devuelve lista de (title, chunk_text) donde:
        - title = H1              para la intro y bloques H1
        - title = "H1 > H2"       para subsecciones H2
    """
    chunks, buf = [], []
    h1 = "intro"
    title = h1

    def flush():
        if buf:
            chunks.append((title, "\n".join(buf).strip()))
            buf.clear()

    for line in text.splitlines():
        if line.startswith("# "):           # H1
            flush()
            h1 = line.lstrip("# ").strip() or "section"
            title = h1
        elif line.startswith("## "):        # H2
            flush()
            h2 = line.lstrip("# ").strip() or "subsection"
            title = f"{h1} > {h2}"
        buf.append(line)
    flush()
    return chunks

# ———————— INGESTA PRINCIPAL ————————
print(f"Procesando Markdown en {DATA} …\n")
total = 0

for md_file in sorted(DATA.glob("*.md")):
    content = md_file.read_text(encoding="utf-8", errors="ignore")
    sections = chunk_by_heading(content)

    # Obtiene keywords para este archivo
    kws = file2keywords.get(md_file.name, [])

    for idx, (title, chunk) in enumerate(sections, start=1):
        doc_id = f"{md_file.stem}__{idx}"
        vec    = ensure_1d(ing.embed(chunk))

        metadata = {
            "file":     md_file.name,
            "chunk":    idx,
            "title":    title,
            "keywords": ",".join(kws),
        }

        collection.delete(ids=[doc_id])          # idempotente
```

```
            collection.add(
                ids        =[doc_id],
                documents  =[chunk],
                embeddings =[vec],
                metadatas  =[metadata],
            )
            total += 1
        print(f"✓ {md_file.name:30s}  → {len(sections)} secciones")
    print(f"\nIngesta finalizada: {total} embeddings almacenados en 'docs'.")
```

rag_chat.py

```
from pathlib import Path
import chromadb
import readline
import textwrap
import sys
import datetime
from llama_cpp import Llama, CreateCompletionResponse
from typing import List, Tuple, Dict, cast

# ———————— rutas ————————
BASE       = Path(r"C:\Users\ernes\Documents\ICAI\SEXTO\TFM\network-rag")
DB_PATH    = BASE / "db"
EMB_PATH   = BASE / "models" / "tinyllama-1.1b-chat-v0.3.Q4_K_M.gguf"
GEN_PATH   = BASE / "models" / "Meta-Llama-3-8B-Instruct-Q3_K_L.gguf"

# ——— Configuración ———
K_RETRIEVE     = 8
MAX_TOKENS     = 600
TEMPERATURE    = 0.15
TOP_P          = 0.9
REPEAT_PENALTY     = 1.1
FREQUENCY_PENALTY = 0.5
PRESENCE_PENALTY  = 0.1
STOP_TOKENS        = ["</s>"]

# ———————— Carga modelos ————————
print("Cargando modelos…", flush=True)
emb = Llama(model_path = str(EMB_PATH), n_ctx = 4096, n_threads = 8,
    embedding = True, verbose = False)
gen = Llama(model_path = str(GEN_PATH), n_ctx = 8192, n_threads = 8,
    verbose = False)

# ———————— vector-store ————————
try:
    client = chromadb.PersistentClient(path=str(DB_PATH))
    col    = client.get_collection("docs")
    total  = col.count()
except Exception as e:
    sys.exit(f"Error conectando a la base de datos: {e}")

if total == 0:
    sys.exit("La colección 'docs' está vacía. Ejecuta primero ingest.py")

def ensure_1d(vec) -> List[float]:
    """
    Aplana recursivamente cualquier nivel de anidación
    y convierte cada elemento a float.
    """
```

```
            while isinstance(vec, list) and vec and isinstance(vec[0], list):
            vec = vec[0]
            return [float(x) for x in vec]

        # ──────── Prompt base ────────
        BASE_PROMPT = textwrap.dedent("""\
            Eres un ingeniero de redes senior.
            Contesta **directamente** la PREGUNTA que se te hace.
            Responde **solo** con la información presente en el CONTEXTO.
            No formules preguntas de seguimiento ni generes múltiples Q&A:
contesta solamente la pregunta que te hago.
            Formatea los comandos en back-ticks (`like this`).
            Cita tu fuente así: (file: <nombre>).
            Si no hay contexto suficiente, responde exactamente "No lo sé."
            """)

        def retrieve(query: str, k: int = K_RETRIEVE) -> str:
            """
            Recupera los k documentos más relevantes e incluye metadatos
            para que se puedan citar automáticamente.
            """
            q_emb = ensure_1d(emb.embed(input=[query]))
            res = col.query(
                query_embeddings=[q_emb],
                n_results=k,
                include=["documents", "metadatas"]
            )

            docs      = (res.get("documents") or [[]])[0]
            metadatas = (res.get("metadatas")  or [[]])[0]

            parts: List[str] = []
            for i, (doc, meta) in enumerate(zip(docs, metadatas), start=1):
                fname = meta.get("title", "desconocido")
                chunk = meta.get("chunk", meta.get("section", i))
                parts.append(
                    f"{i}. {doc}\n(file: {fname})"
                )
            return "\n---\n".join(parts)

        print(f"{total} secciones disponibles. Escribe tu pregunta (exit para
salir).")

        history: List[Tuple[str,str]] = []  # lista de (role, text)

        while True:
            q = input("❯ ").strip()
            if not q or q.lower() in {"exit", "quit", "salir"}:
                break

            # 1) Recupera contexto RAG
            ctx = retrieve(q)

            # 2) Añade turno de usuario al historial
            history.append(("user", q))

            # 3) Construye prompt: historial + contexto + pregunta
            prompt_parts = [BASE_PROMPT, ""]
            if len(history) > 1:
                prompt_parts.append("HISTORIAL:")
                for role, txt in history[-4:]:
```

```
                tag = "Usuario" if role == "user" else "Asistente"
                prompt_parts.append(f"{tag}: {txt}")
            prompt_parts.append("")

        if ctx:
            prompt_parts.append("CONTEXTO RAG:")
            prompt_parts.append(ctx)
            prompt_parts.append("")

        prompt_parts.append(f"PREGUNTA: {q}")
        prompt_parts.append("RESPUESTA:")
        full_prompt = "\n".join(prompt_parts)

        # 4) Llama al LLM
        raw = gen.create_completion(
            prompt=full_prompt,
            max_tokens=MAX_TOKENS,
            temperature=TEMPERATURE,
            top_p=TOP_P,
            repeat_penalty=REPEAT_PENALTY,
            frequency_penalty=FREQUENCY_PENALTY,
            presence_penalty=PRESENCE_PENALTY,
            stop=STOP_TOKENS,
            stream=False,
        )

        out = cast(CreateCompletionResponse, raw)
        ans = out["choices"][0]["text"].strip() or "No lo sé."

        # 5) Imprime y guarda en historial
        print("\n" + ans + "\n")
        history.append(("assistant", ans))
```

ANEXO III: Corpus de referencia para el entrenamiento del modelo

A partir de la página siguiente se despliega de forma íntegra el conjunto de 48 fichas que conforman el **corpus utilizado para nutrir al sistema RAG**. Cada ficha incluye fragmentos de guías IOS, RFCs y buenas prácticas, debidamente justificadas y referenciadas, y servirá como base para cualquier consulta o auditoría que precise verificar el origen de las configuraciones.

VLAN básicas en Cisco IOS

Las *Virtual LANs* (VLAN) permiten crear dominios de broadcast lógicos independientes
dentro de un mismo conmutador.

- Reducen el tamaño del dominio de broadcast.
- Aíslan departamentos o funciones, mejorando la seguridad.
- Facilitan la administración de direcciones IP.

---

1 Crear una VLAN

En modo de configuración global:

```
Switch(config)# vlan 10
Switch(config-vlan)# name Finanzas
Switch(config-vlan)# exit
```

Repite el proceso para cada VLAN que necesites (11, 12, 13…).

---

2 Asignar puertos de acceso

Para asociar varios puertos Fast-Ethernet (Fa0/1 – Fa0/5) a la VLAN 10 en bloque:

```
Switch(config)# interface range fa0/1 - 5
Switch(config-if-range)# switchport mode access
Switch(config-if-range)# switchport access vlan 10
Switch(config-if-range)# no shutdown
Switch(config-if-range)# exit
```

---

3 Configurar un puerto troncal

Si el conmutador se conecta a un router o a otro switch que transporta varias VLAN,
habilita un troncal:

```
Switch(config)# interface g0/1
Switch(config-if)# switchport mode trunk
Switch(config-if)# no shutdown
```

El enlace troncal etiqueta los frames (IEEE 802.1Q) y permite que todas las VLAN
crucen el mismo cable.

---

4 Verificar la configuración

```
Switch# show vlan brief
Switch# show interfaces status
Switch# show interfaces trunk
```

Estos comandos confirman qué puertos están activos, a qué VLAN pertenecen y si el
troncal está correctamente negociado.

---

5 Buenas prácticas rápidas

Acción	Motivo
Reservar la VLAN 1 para gestión	Limita el tráfico de usuario en la VLAN por defecto.
Usar descripciones en puertos	Facilita el soporte y el traspaso de proyectos.
Deshabilitar puertos no usados	Reduce vectores de ataque físicos.
Documentar rangos IP VLAN	Evita solapamientos y agiliza el *troubleshooting*.

Lecturas recomendadas

- Cisco IOS — *VLAN Configuration Guide*.
- IEEE 802.1Q-2018 — estándar de etiquetado VLAN.
- Cisco IOS — *Interface Range Command Reference*.

Router-on-a-Stick (Inter-VLAN)

Cuando un único router (o switch Capa 3) debe actuar como gateway para varias VLAN, se emplea la técnica **Router-on-a-Stick**: se crea una sub-interface por VLAN, se encapsula el tráfico con IEEE 802.1Q y todo viaja por un solo enlace físico.

---

1 Requisitos previos

- Switch con un puerto configurado en modo **trunk 802.1Q** hacia el router.
- Cada VLAN dispone de su propia subred IP (/24 es habitual).
- En el router, la interfaz física permanece *up* **sin** dirección IP.

---

2 Cuándo utilizar RoaS

- Campus pequeños/medianos donde el tráfico inter-VLAN no supera 1 GbE/10 GbE.
- Escenarios con presupuesto limitado: evita comprar un switch L3 dedicado.
- Laboratorios y entornos de pruebas por su montaje rápido y didáctico.

---

3 Configuración paso a paso

3.1 Configurar el troncal en el switch

```
Switch(config)# interface g0/1
Switch(config-if)# switchport mode trunk
Switch(config-if)# no shutdown
```

3.2 Crear sub-interfaces en el router

```
Router(config)# interface fastEthernet0/0
Router(config-if)# no shutdown
Router(config-if)# exit

! VLAN 10
Router(config)# interface fastEthernet0/0.10
Router(config-subif)# encapsulation dot1q 10
```

```
Router(config-subif)# ip address 192.168.10.1 255.255.255.0

! VLAN 20
Router(config)# interface fastEthernet0/0.20
Router(config-subif)# encapsulation dot1q 20
Router(config-subif)# ip address 192.168.20.1 255.255.255.0
```

---

4 Habilitar el enrutamiento interno

En routers Cisco viene activo por defecto;
en un **switch Capa 3** puede ser necesario:

```
Switch(config)# ip routing
```

---

5 Verificación rápida

```
Router# show ip interface brief | include FastEthernet0/0
Router# show running-config interface fa0/0.10
Router# ping 192.168.10.2          ← prueba con un host de la VLAN 10

Switch# show interfaces trunk
Switch# show vlan brief
```

---

6 Resumen sub-interfaces VLAN

Sub-IF	VLAN	IP gateway	Máscara
fa0/0.10	10	192.168.10.1	/24
fa0/0.20	20	192.168.20.1	/24

---

7 Solución de problemas habitual

Síntoma	Posible causa	Comando útil
Hosts no alcanzan el gateway	VLAN mal etiquetada en el trunk	`show interfaces trunk`
Ping entre VLAN falla	Sub-IF sin IP o sin *ip routing*	`show run int fa0/0.X`

Síntoma	Posible causa	Comando útil
Alta latencia / CPU en router	Exceso de tráfico en enlace único	Considerar L3-switching

---

8 Buenas prácticas

1. Igualar **número-VLAN = número de sub-interface** para claridad.
2. Añadir descripciones: `description Gateway_VLAN10`.
3. Limitar VLANs permitidas: `switchport trunk allowed vlan 10,20`.
4. Supervisar el troncal; es punto único de fallo y cuello de botella.

---

9 Lecturas recomendadas

- Cisco — *Configuring Inter-VLAN Routing using Router-on-a-Stick*.
- Cisco IOS — *Subinterfaces and VLANs Configuration Guide*.
- IEEE 802.1Q-2018 — *Bridges and Bridged Networks*.

Enlaces Troncales 802.1Q

Un *trunk* transporta tráfico de **múltiples VLAN** sobre un único enlace físico añadiendo una cabecera IEEE 802.1Q. Es esencial cuando varios switches — o un
switch y un router — necesitan compartir VLAN de forma eficiente.

---

1 Fundamentos de 802.1Q

- Inserta un **tag de 4 bytes** entre la cabecera Ethernet y la carga útil.
- El campo *VLAN ID* (12 bits) permite hasta **4094 VLAN** (1 y 4095 reservadas).
- El tráfico sin tag pertenece a la **Native VLAN** (por defecto, VLAN 1).

---

2 Requisitos previos

1. Las VLAN deben existir en ambos extremos (`show vlan brief`).
2. La **Native VLAN** ha de coincidir para evitar *Native VLAN mismatch*.
3. STP activado (PVST+ / RSTP) para prevenir bucles.

---

3 Cuándo utilizar un enlace troncal

- Conectar dos switches que transportan múltiples VLAN al backbone.
- Vincular un switch Capa 2 a un router (*Router-on-a-Stick*).
- Unir un switch de acceso a distribución/L3 que agrega muchas VLAN.

---

4 Configuración paso a paso (Catalyst IOS)

4.1 Habilitar el trunk

```
Switch# conf t
Switch(config)# interface Gig0/1
! (En algunas plataformas) seleccionar encapsulado
Switch(config-if)# switchport trunk encapsulation dot1q
Switch(config-if)# switchport mode trunk
```

```
Switch(config-if)# no shutdown
```

4.2 Restringir VLAN permitidas

```
Switch(config-if)# switchport trunk allowed vlan 10-13
```

4.3 Definir Native VLAN fuera de producción

```
Switch(config-if)# switchport trunk native vlan 99
```

4.4 Seguridad adicional

```
Switch(config-if)# spanning-tree bpduguard enable
```

---

5 Verificación rápida

```
Switch# show interfaces Gig0/1 switchport
Switch# show interfaces trunk
Switch# show vlan brief
```

Indicadores clave
- **Status**: trunk
- **Encapsulation**: 802.1Q
- **Vlans allowed on trunk**: 10-13
- **Native VLAN**: 99

---

6 Parámetros clave

Parámetro	Comando (config-if)	Valor típico
Modo troncal	`switchport mode trunk`	—
Encapsulado (si aplica)	`switchport trunk encapsulation dot1q`	dot1q
VLAN nativa	`switchport trunk native vlan 99`	99
VLANs permitidas	`switchport trunk allowed vlan 10-13,99`	lista

---

7 Troubleshooting rápido

Síntoma	Revisa…
Native VLAN mismatch	`show cdp neighbors detail`
VLAN inaccesible	Lista de VLAN permitidas
Alta latencia en trunk	Congestión; QoS / LACP

Síntoma	Revisa…	
Tráfico sin etiquetar inesperado	Native VLAN	VLAN de usuario

---

8 Buenas prácticas

1. Utiliza una **Native VLAN dedicada (p.ej. 99)** y sin hosts.
2. Limita las VLAN permitidas a las estrictamente necesarias.
3. Activa BPDU Guard / Root Guard en bordes para reforzar STP.
4. Documenta las VLAN y el propósito de cada trunk en tu CMDB o wiki.
5. Etiqueta la VLAN nativa en ambos extremos (`vlan dot1q tag native`) si el HW lo permite.

---

9 Lecturas recomendadas

- Cisco — *Cisco Catalyst Trunking Configuration Guide*.
- Cisco Press — *CCNA 200-301 Official Cert Guide*, cap. 6.
- IEEE 802.1Q-2018 — *Bridges and Bridged Networks*.

Subnetting & CIDR rápido

El **Classless Inter-Domain Routing (CIDR)** permite expresar subredes con prefijos
de longitud variable (VLSM), optimizando el uso de direcciones IPv4. Subnetear bien significa equilibrar número de hosts, dominios de broadcast y escalabilidad.

---

1 Conceptos básicos

- **Prefijo /máscara**: `/24` = `255.255.255.0` → 24 bits de red, 8 de host.
- **Potencias de 2**: cada bit de host doblará los hosts posibles.
- **Hosts útiles** = `2 - 2` (se restan red y broadcast).

---

2 Tabla de referencia CIDR

/Prefix	Máscara decimal	Subredes dentro de /24	Hosts / subred
/25	255.255.255.128	2	126
/26	255.255.255.192	4	62
/27	255.255.255.224	8	30
/28	255.255.255.240	16	14
/29	255.255.255.248	32	6
/30	255.255.255.252	64	2 (enlaces PtP)
/31*	255.255.255.254	128	0 (PtP /31)

*RFC 3021 habilita /31 para enlaces punto-a-punto.

---

3 Procedimiento rápido de subneteo (ejemplo)

Objetivo: dividir 192.168.10.0/24 en 3 LAN (50 hosts c/u) y 1 enlace PtP.

1. Ordenar por tamaño: LAN 1-3 (~50) → /26; PtP → /30.
2. Asignar en orden:
   - 192.168.10.0 /26 → LAN 1 (hosts .1-.62)
   - 192.168.10.64 /26 → LAN 2 (hosts .65-.126)
   - 192.168.10.128 /26 → LAN 3 (hosts .129-.190)

- 192.168.10.192 /30 → Enlace PtP (IPs .193 & .194)

---

4 Herramientas CLI útiles

```
# ipcalc 192.168.10.64/26
Address:   192.168.10.64
Netmask:   255.255.255.192 = 26
Wildcard:  0.0.0.63
Hosts/Net: 62
Network:   192.168.10.64
Broadcast: 192.168.10.127
```

En routers Cisco:

```
Router# show ip route 192.168.10.64 255.255.255.192
```

---

5 Buenas prácticas

1. **Planificar VLSM** desde el inicio para minimizar derroche de IP.
2. Reservar rangos contiguos para sumarizar rutas (`ip summary-address`).
3. Documentar cada subred: propósito, gateway, pool DHCP, ACL asociada.
4. Usar **/31 o /30** para enlaces serie/PtP; evita desperdiciar /24.
5. Considerar NAT o IPv6 si el espacio privado se agota.

Lecturas recomendadas

- Cisco Press — *CCNA 200-301 Official Cert Guide*, cap. 11.
- RFC 4632 — *Classless Inter-Domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan.*
- RFC 3021 — *Using 31-Bit Prefixes on IPv4 Point-to-Point Links.*

Diseño VLSM paso a paso

El **Variable Length Subnet Masking (VLSM)** permite asignar máscaras de red de
distinto tamaño dentro de un mismo bloque de direcciones, ajustando cada subred
al número real de hosts y evitando desperdicio de IP.

---

1 Ventajas de VLSM

- Uso eficiente del espacio IPv4.
- Facilita la **summarización** en routers, reduciendo tablas de rutas.
- Flexibilidad para crecer: se reservan bloques aún sin usar.

---

2 Pasos metodológicos

1. **Inventario de redes**: listar todas las LAN, enlaces PtP y loopbacks con el
   número de hosts requeridos.
2. **Ordenar de mayor a menor** (mayor necesidad de hosts primero).
3. **Asignar bloques** empezando por la mayor LAN y "restar" del espacio padre.
4. **Actualizar tabla** y continuar con la siguiente necesidad.
5. Documentar dirección de red, broadcast, gateway y máscara CIDR.

---

3 Ejemplo práctico

Bloque padre: **10.20.0.0/22** (1022 hosts). Necesidades:
* LAN Ventas – 200 hosts
* LAN Soporte – 100 hosts
* LAN I+D – 60 hosts
* Enlace RouterA–RouterB – 2 hosts

3.1 Conversión a potencias de 2

Requisito	Hosts útiles	Tamaño /bits	Subred resultante
200	256	/24	10.20.0.0/24
100	128	/25	10.20.1.0/25
60	64	/26	10.20.1.128/26
2	2	/30	10.20.1.192/30

Nota: el bloque padre /22 permite 4 subredes /24. Vamos consumiendo en orden y
de izquierda a derecha dentro del rango.

3.2 Resultado de asignación

Subred	Máscara	Hosts (útiles)	Gateway
10.20.0.0 /24	255.255.255.0	254	10.20.0.1
10.20.1.0 /25	255.255.255.128	126	10.20.1.1
10.20.1.128 /26	255.255.255.192	62	10.20.1.129
10.20.1.192 /30	255.255.255.252	2	10.20.1.193

Bloques libres restantes: 10.20.2.0/23 (510 hosts) reservados para expansión.

---

4 Configuración de un resumen en OSPF

Suponiendo que todas las subredes anteriores cuelgan del mismo ABR:

```
Router(config)# router ospf 1
Router(config-router)# area 0 range 10.20.0.0 255.255.252.0
```

La máscara `255.255.252.0` equivale a **/22**, que cubre todo el bloque original.

---

5 Herramientas recomendadas

- `ipcalc`, `subnetcalc` en Linux/macOS.
- Calculadoras web (SolarWinds, He.net).
- Plantillas Excel/Google Sheets con fórmulas /ETF(2^n).

---

6 Buenas prácticas

1. Mantener el **cuadro maestro de direccionamiento** versionado (Git / Wiki).

2. Reservar /32 loopbacks contiguos para sumarización (p.ej. 10.255.x.x/24).

3. Documentar ACL, DHCP y servicios asociados a cada subred.

4. Evitar saltos de máscara no estándar en la misma LAN (mezclar /25 y /26).

Lecturas recomendadas

- Cisco Press — *Subnetting Secrets*, cap. 3.
- RFC 1817 — *CIDR and Classful Routing.*
- PacketLife.net — *VLSM Cheat Sheet.*

DHCP en switches y routers Cisco

El **Dynamic Host Configuration Protocol (DHCP)** permite asignar direcciones IP,
gateway, DNS y otros parámetros a los hosts de forma automática, simplificando la
gestión de redes y evitando errores de configuración manual.

---

1 Escenarios típicos

- **Router como servidor DHCP** para múltiples VLAN pequeñas.
- **Switch Capa 3** sirviendo DHCP localmente en una red de campus.
- **Routed Access**: router distribuye IP y switches L2 actúan sólo como relay.

---

2 Requisitos previos

1. Rango IP planificado y sin solapamientos.
2. Reservas (static bindings) para impresoras, servidores o VoIP, si aplica.
3. Reloj correcto (NTP) para que los *leases* registren hora exacta.

---

3 Configurar DHCP en un router

```
! 3.1  Excluir direcciones
Router(config)# ip dhcp excluded-address 192.168.10.1 192.168.10.20

! 3.2  Crear el pool
Router(config)# ip dhcp pool LAN_10
Router(dhcp-config)# network 192.168.10.0 255.255.255.0
Router(dhcp-config)# default-router 192.168.10.1
Router(dhcp-config)# dns-server 8.8.8.8 1.1.1.1
Router(dhcp-config)# lease 7   ! días
Router(dhcp-config)# exit
```

Tip: el router debe tener una interfaz *up/up* en la subred que sirve.

---

4 Configurar DHCP en un switch Layer-3

```
Switch(config)# ip dhcp excluded-address 10.20.1.1 10.20.1.10
Switch(config)# ip dhcp pool USERS
Switch(dhcp-config)# network 10.20.1.0 255.255.255.0
Switch(dhcp-config)# default-router 10.20.1.1
Switch(dhcp-config)# option 150 ip 10.20.1.50     ! TFTP p/phones
Switch(dhcp-config)# exit
```

---

5 Habilitar DHCP relay (ip helper-address)

Cuando el servidor DHCP está en otra red/VLAN:

```
Interface VLAN 30 o fa0/0.30
Router(config-if)# ip helper-address 10.10.10.5   ! IP del servidor DHCP
```

El comando reenvía broadcast UDP (67/68, NetBIOS, TFTP…) como unicast al servidor.

---

6 Verificación y comandos útiles

```
Router# show ip dhcp binding
Router# show ip dhcp pool LAN_10
Router# debug ip dhcp server events   ! temporal, en laboratorio
```

En el cliente:

```
Windows> ipconfig /all
Linux$ dhclient -v
```

---

7 Troubleshooting rápido

Síntoma	Posible causa	Comando clave
Cliente no recibe IP	DHCP pool agotado	`show ip dhcp pool`
Request timed out	Falta `ip helper-address`	`show run interface`
IP asignada fuera de rango	Exclusiones mal definidas	`show ip dhcp binding`
Leases no se liberan	Clocks desincronizados	`show clock` / NTP

---

8 Buenas prácticas

1. Colocar servidores DHCP centrales en VLAN de gestión separada.
2. Usar `option 82` (relay info) y autenticación DHCP snooping en entornos hostiles.
3. Respaldar `dhcp database` en un servidor TFTP con `ip dhcp database`.
4. Registrar reservas estáticas en el mismo fichero o CMDB para evitar conflictos.

Lecturas recomendadas

- Cisco IOS — *DHCP Server Configuration Guide*.
- RFC 2131 — *Dynamic Host Configuration Protocol*.
- Cisco — *Understanding and Configuring DHCP Snooping*.

OSPF área 0 (Single-Area)

El **Open Shortest Path First (OSPF)** es un protocolo de enrutamiento link-state
que calcula las rutas más cortas usando el algoritmo Dijkstra. En redes pequeñas o medianas suele desplegarse con una sola área: **área 0** (backbone).

---

1 Ventajas de usar un único área

- Configuración y diseño simples.
- LSDB idéntica en todos los routers, facilitando el *troubleshooting*.
- No se requieren ABR ni summarización intra-área.

---

2 Requisitos previos

1. Cada router conoce su ID (Router ID); se elige la mayor loopback o IP activa.
2. Enlaces punto-a-punto o broadcast en *up/up*.
3. Misma **hello-time**, **dead-time** y **autenticación** entre vecinos.

---

3 Configuración básica (IOS)

```
Router(config)# router ospf 1
Router(config-router)# network 192.168.10.0 0.0.0.255 area 0
Router(config-router)# network 10.20.1.0   0.0.0.3   area 0   ! enlace /30
Router(config-router)# exit
```

> *Tip*: usa comodines (wildcards) o *ip ospf 1 area 0* directamente en la interfaz.

---

4 Verificación rápida

```
Router# show ip ospf neighbor
Router# show ip ospf interface brief
Router# show ip route ospf
Router# show ip ospf database
```

Indicadores clave
- **STATE**: FULL/ - significa adyacencia completa.
- **LSA count** razonable; debería crecer con cada red anunciada.

---

5 Temporizadores por defecto

Tipo de red	Hello	Dead
Broadcast / PtP	10 s	40 s
NBMA / Frame-Relay	30 s	120 s

Asegúrate de igualar los valores si los modificas (`ip ospf hello-interval`).

---

6 Troubleshooting rápido

Síntoma / Log	Revisa…
OSPF-4 HELLO_NOT_RECV	Filtros, timers, capa 2 down
Vecino en **EXSTART/EXCHANGE**	MTU mismatch (ajusta o `ip ospf mtu-ignore`)
Rutas no aparecen en tabla	Falta `network`, passive-int
LS age = 3600 y no se renueva	Caída de adyacencia, DR muerto

---

7 Buenas prácticas

1. Asignar **Router ID manual** (`router-id 1.1.1.1`) para estabilidad.

2. Declarar interfaces de usuario como `passive-interface` para reducir LSAs.

3. Activar autenticación MD5 (`ip ospf authentication message-digest`).

4. Documentar topología con diagramas LSDB exportados (`show ip ospf database graph`).

---

8 Lecturas recomendadas

- Cisco IOS — *OSPF Configuration Guide*.

- RFC 2328 — *OSPF Version 2 Specification.*
- Odom, W. — *CCNA 200-301 Official Cert Guide*, cap. 17.

OSPF Multi-Área y LSAs

En implementaciones medianas-grandes, **OSPF** se divide en múltiples áreas para
reducir el tamaño de la LSDB y el cómputo de SPF, además de aislar cambios de routing.
El backbone siempre es **área 0**; todas las demás áreas deben conectarse lógica o físicamente a ella mediante un **ABR** o un **virtual-link**.

---

1 Razones para usar varias áreas

- Limitar la propagación de LSAs: cambios en un área no saturan toda la red.
- Restringir tablas de routing locales, mejorando el rendimiento de CPU/RAM.
- Permitir **summarización** en los ABR, acortando aún más las tablas.

---

2 Tipos de routers en un diseño multi-área

Tipo	Función principal	Ejemplo
Backbone Router (BR)	Tiene interfaces **solo** en área 0	Core-1
Internal Router	Todas sus interfaces en la misma área	Acc-Switch-1 (área 10)
Area Border Router (ABR)	Conecta área 0 y otra área, genera LSA-3	Dist-1 (áreas 0 y 10)
ASBR	Redistribuye rutas externas (LSA-5 / 7)	Perimeter-FW

---

3 Resumen de LSAs relevantes

LSA	Propósito	Generado por	Inunda a
1	Estados de enlaces intra-área	Todos	Área local
2	Resumen de red multi-access (DR)	DR	Área local
3	Resumen inter-área (*IA*)	ABR	Otras áreas
4	Ruta a un ASBR	ABR	Otras áreas

LSA	Propósito	Generado por	Inunda a
5	Rutas externas (E1/E2)	ASBR	Toda la red
7	Rutas externas en NSSA	ASBR NSSA	Solo área NSSA

---

4 Configuración básica de un ABR

```
Router(config)# router ospf 1
Router(config-router)# network 10.10.0.0 0.0.0.255 area 0      ! backbone
Router(config-router)# network 10.20.1.0 0.0.0.255 area 10     ! área interna
! Summarizar prefijos del área 10 al backbone:
Router(config-router)# area 10 range 10.20.0.0 255.255.252.0
```

---

5 Crear un virtual-link (cuando un área no toca área 0)

```
! En el ABR que conecta área 0 y área 1
Router(config-router)# area 1 virtual-link 2.2.2.2      ! 2.2.2.2 = Router-ID remoto
! En el router remoto (solo área 1)
Router(config-router)# area 1 virtual-link 1.1.1.1
```

> *Nota*: el área de tránsito del virtual-link debe ser **Normal (no stub)**.

---

6 Áreas especiales resumidas

Tipo de área	LSA-5	LSA-3	LSA-7	Uso común
Stub	N	Y	N	Sucursales pequeñas
Totally Stub	N	N	N	Confía en default-route
NSSA	N	Y	Y	Redistribución local
Totally NSSA	N	N	Y	NSSA + ruta por defecto

---

7 Verificación y comandos clave

```
Router# show ip ospf border-routers
Router# show ip ospf database summary
Router# show ip ospf virtual-links          ! si aplica
Router# show ip route ospf | include IA     ! rutas inter-área
```

8 Troubleshooting rápido

Síntoma / Log	Revisa…
Rutas IA faltantes	`area X range` mal o filtrado
Virtual-link down	MTU/timers en el área tránsito
LSA-checksum mismatch entre ABR	Desfase de versiones / MTU
ASBR no propaga LSA-5 en área Stub	Área mal declarada (no-stub)

9 Buenas prácticas

1. Mantener **área 0 contigua**; usa virtual-links solo como último recurso.
2. Limitar el tamaño de cada área a ~50 routers y < 300 subredes (Cisco guía).
3. Resumir prefijos en los ABR para reducir LSAs.
4. Activar autenticación (MD5/SHA) coherente en todas las áreas.
5. Documentar en un diagrama qué router es ABR, BR, ASBR.

Lecturas recomendadas

- Cisco IOS — *OSPF Multi-Area Configuration Guide.*
- RFC 2328 — *OSPF Version 2 Specification.*
- Doyle, J. — *Routing TCP/IP Vol.1*, cap. 8-10.

RIP vs OSPF — cuándo usar

RIP (Routing Information Protocol) y **OSPF (Open Shortest Path First)** son protocolos IGP populares, pero se orientan a tamaños de red y requerimientos diferentes. Esta ficha resume sus diferencias y cuándo conviene cada uno.

---

1 Comparativa rápida

Característica	RIP v2	OSPF v2
Tipo de algoritmo	Vector-distancia	Link-state
Métrica	Saltos (hops)	Coste (BW)
Límite de rutas	15 hops máx.	Sin límite práctico
Convergencia	Lenta	Rápida
Actualización periódica	30 s (entera)	Solo LSAs al cambiar
VLSM/CIDR	Sí (v2)	Sí
Autenticación simple	MD5 / texto plano	MD5 / SHA
Consumo CPU/RAM	Muy bajo	Medio-alto

---

2 Ventajas y desventajas clave

RIP v2

- **Simplicidad absoluta** – ideal para laboratorios o redes < 10 routers.
- **Poca carga** en equipos legacy de bajo recurso.
- **Problemas**: límite de 15 hops y convergencia lenta (hasta 3 min).

OSPF v2

- **Escala** a cientos de routers y miles de prefijos.
- Convergencia rápida (sub-segundos con BFD).
- Admite **multiárea**, summarización, filtros y rutas externas (E1/E2).
- **Requiere** más planeación: áreas, Router-ID, timers, DR/BDR…

---

3 Cuándo elegir uno u otro

Escenario	Recomendación
Red pequeña simple (edge routers x 5)	RIP v2
Campus mediano con múltiples VLAN	OSPF (área 0)
ISP / backbone multi-área	OSPF multi-área
Hardware muy limitado (old 2600)	RIP v2
Alta disponibilidad < 1 s	OSPF + BFD

---

4 Ejemplo mínimo de configuración

4.1 RIP v2

```
router rip
 version 2
 network 192.168.0.0
 no auto-summary
```

4.2 OSPF (single-area)

```
router ospf 1
 network 192.168.0.0 0.0.0.255 area 0
```

---

5 Troubleshooting diferencial

Síntoma	RIP v2	OSPF v2
Rutas tardan minutos en aparecer	Normal: temporizador 30 s	Revisar timers/adjacencia
Network unreachable a +15 hops	Límite de RIP	No aplica (usar OSPF)
Vecinos no formados	Versión / passive-int	Hello/Dead, MTU, auth

---

6 Buenas prácticas

1. **No mezclar**: evita redistribuir entre RIP y OSPF salvo que sea indispensable.

2. Documenta métricas: en OSPF ajusta `ip ospf cost` en enlaces de baja BW.

3. Activa autenticación en ambos: `ip rip authentication key-chain …` / OSPF MD5.

4. Evalúa migrar de RIP a OSPF en redes que crecen (> 10 routers o > 15 saltos).

Lecturas recomendadas

- Cisco IOS — *RIP Configuration Guide*.
- Cisco IOS — *OSPF Configuration Guide*.
- Doyle, J. — *Routing TCP/IP Vol. 1*, caps. 3 y 8.

Rutas estáticas y default en Cisco IOS

Las **rutas estáticas** se configuran manualmente y no cambian salvo intervención del administrador. Son el método de enrutamiento más simple y consumen cero CPU, pero requieren mantenimiento cuando la topología varía.

---

1 ¿Cuándo usar rutas estáticas?

- Pequeñas sucursales con un único enlace WAN.
- Nexthop hacia un **firewall** o proveedor upstream donde no hay IGP.
- **Rutas de respaldo (flotantes)** con AD mayor que el IGP.
- Direccionamiento a redes "nulas" (`Null0`) para **summarización + black-hole**.

---

2 Sintaxis base

```
ip route <RED> <MÁSCARA> {<NEXTHOP> | <INTERFAZ>}

! Ejemplos
ip route 192.168.50.0 255.255.255.0 10.1.1.2
ip route 172.16.0.0 255.240.0.0 Serial0/0/0
```

Campos	Significado
`<RED>`	Dirección de red destino
`<MÁSCARA>`	Máscara en decimal o CIDR
`<NEXTHOP>`	IP del router adyacente
`<INTERFAZ>`	Se usa cuando no hay IP próxima (p.ej. punto-a-multipunto)

---

3 Ruta por defecto (*Gateway of Last Resort*)

```
ip route 0.0.0.0 0.0.0.0 10.1.1.254
```

Conecta "todo lo demás" al ISP o dispositivo perimetral.

---

4 Rutas estáticas flotantes y track

4.1 Distancia administrativa (AD)

```
ip route 0.0.0.0 0.0.0.0 10.1.1.254 5    ! Primaria (AD 5)
ip route 0.0.0.0 0.0.0.0 192.0.2.254 250 ! Secundaria (flotante)
```

4.2 Seguimiento de SLA

```
ip sla 10
 icmp-echo 8.8.8.8
 frequency 5
ip sla schedule 10 life forever start-time now

track 10 ip sla 10 reachability
ip route 0.0.0.0 0.0.0.0 10.1.1.254 track 10
```

La ruta desaparece si el *ping* al 8.8.8.8 falla.

---

5 Verificación rápida

```
Router# show ip route static
Router# show ip route 0.0.0.0
Router# show track 10
```

---

6 Troubleshooting

Síntoma	Posible causa	Comando útil
Ruta no aparece en tabla	Nexthop sin resolución	`show arp`, `ping`
Paquetes se enrut. mal	Más específica en IGP	`show ip route`
Cambio primario→secundario lento	AD insuficientemente alta	Revisar valores AD

---

7 Buenas prácticas

1. Mantener descripción en cada `ip route` (`description Hacia ISP`).
2. Usar rutas “null” /32 para **loopback summarization** (ISP BGP).

3. Documentar en CMDB los cambios; rutas estáticas pueden perderse al migrar.

4. Combinar con SLA/track para failover automático.

Lecturas recomendadas

- Cisco IOS — *IP Routing: Static Routes Configuration Guide*.
- RFC 4632 — *CIDR and Route Aggregation*.
- Doyle, J. — *Routing TCP/IP Vol.1*, cap. 2.

ACL estándar (números 1-99 / 1300-1999)

Las **Access Control Lists (ACL)** estándar en Cisco IOS filtran tráfico basándose
exclusivamente en la dirección IPv4 origen. Se aplican normalmente **lo más cerca
del destino**, pues no distinguen protocolos ni puertos.

---

1 Rangos válidos

Rango numérico	Tipo de lista	Ejemplo de uso
1-99	Estándar	Routers antiguos / sencillo
1300-1999	Estándar expandida	Para evitar solape con ACL extendidas

---

2 Comandos básicos

```
! Crear ACL estándar #10
Router(config)# access-list 10 permit 192.168.10.0 0.0.0.255
Router(config)# access-list 10 deny   any

! Aplicar a interfaz (tráfico entrante)
Router(config)# interface Gig0/0
Router(config-if)# ip access-group 10 in
```

Campos	Significado
`permit/deny`	Acción sobre el tráfico
IP / wildcard	Red origen y comodín (0 = exacto, 255 = cualquiera)

---

3 Ejemplos prácticos

3.1 Permitir solo a la red contable (172.16.50.0/24) llegar al servidor

```
access-list 15 permit 172.16.50.0 0.0.0.255
access-list 15 deny   any
interface Gi0/1
 ip access-group 15 in
```

3.2 Bloquear una IP específica (198.51.100.25) a toda la LAN

```
access-list 25 deny 198.51.100.25
access-list 25 permit any
interface Vlan20
 ip access-group 25 in
```

---

4 Verificación rápida

```
Router# show access-lists 10
Router# show ip interface Gig0/0
```

---

5 Troubleshooting

Síntoma	Posible causa	Comando útil
Tráfico permitido se bloquea	Orden incorrecto (ACL top)	`show access-lists`
ACL no tiene efecto	Aplicada en sentido errado	`show run interface`
Paquetes contados = 0	ACL nunca matchea	`debug ip packet`

---

6 Buenas prácticas

1. Nombrar ACL con `ip access-list standard <NOMBRE>` para legibilidad (IOS 12.3+).

2. Añadir comentarios (`remark`) descriptivos:
   `access-list 10 remark Permite contabilidad`.

3. Finalizar explícitamente con `deny any log` para registrar descartes.

4. Revisar contadores antes de limpiar ACL en producción.

Lecturas recomendadas

- Cisco IOS — *IP Access Lists Configuration Guide*.
- RFC 6235 — *IP Header Filtering Considerations*.
- Odom, W. — *CCNA 200-301 Official Cert Guide*, cap. 20.

ACL extendidas (números 100-199 / 2000-2699)

Las **Access Control Lists extendidas** filtran tráfico por IP origen, IP destino, protocolo (TCP, UDP, ICMP…), puerto, e incluso flags TCP. Se aplican **lo más cerca**
del origen para evitar que paquetes indeseados crucen la red.

---

1 Rangos disponibles

Rango numérico	Tipo de lista	Uso típico
100 – 199	Extendida	IOS clásico
2000 – 2699	Extendida exp.	Evitar solape con IPv6 / objete

---

2 Sintaxis general

```
access-list <NUM> {permit|deny} <PROTO> <SRC_IP> <SRC_WC> [operator <p>] \
                                    <DST_IP> <DST_WC> [operator <p>] [log]

! Ejemplo: permitir HTTPS de red contable (172.16.50.0/24) al servidor 192.0.2.10
access-list 110 permit tcp 172.16.50.0 0.0.0.255 host 192.0.2.10 eq 443
```

Claves de operador TCP/UDP más usadas → `eq`, `gt`, `lt`, `range`.

---

3 Aplicar la ACL a la interfaz

```
interface Gig0/0
 ip access-group 110 in      ! sentido entrante
```

> *Tip*: para tráfico de salida usa `out`; revisa siempre la dirección del flujo.

---

4 Ejemplos prácticos

4.1 Bloquear ICMP echo request a toda la LAN 10.10.0.0/16

```
access-list 120 deny icmp any 10.10.0.0 0.0.255.255 echo
access-list 120 permit ip any any
```

4.2 Permitir SSH (22) solo desde host admin 203.0.113.5 al router

```
access-list 130 permit tcp host 203.0.113.5 any eq 22
access-list 130 deny tcp any any eq 22 log
access-list 130 permit ip any any
```

---

5 Verificación rápida

```
Router# show access-lists 110
Router# show ip interface Gig0/0 | include access
```

---

6 Troubleshooting

Síntoma	Causa probable	Comando útil
Tráfico permitido se bloquea	Orden de ACE incorrecto	`show access-lists`
ACL no contabiliza paquetes	Aplicada en interfaz errada	`show run int`
Logs excesivos (`%SEC-6-IPACCESS`)	Falta `log disable` o `log-input`	`logging` cmds

---

7 Buenas prácticas

1. Crear ACL **nombradas** (`ip access-list extended <NOMBRE>`) para mayor claridad.

2. Incluir instrucciones **remark** antes de bloques lógicos.

3. Colocar las ACE **más específicas** al principio para acelerar el *matching*.

4. Usar `deny ip any any log` al final solo en fase de diagnóstico; luego quitar `log`.

Lecturas recomendadas

- Cisco IOS — *IP Access Lists Configuration Guide* (Sección “Extended ACL”).

- RFC 2474 — *Differentiated Services Field* (uso de DSCP en ACL).
- Cisco Press — *CCNA Security Official Cert Guide*, cap. 5.

NAT dinámico y PAT (Overload) en Cisco IOS

El **Network Address Translation (NAT)** permite que redes privadas alcancen Internet usando una o varias direcciones IPv4 públicas. La variante más usada en empresas es **PAT (NAT Overload)**, donde miles de hosts comparten una única IP pública diferenciándose por puertos TCP/UDP.

---

1 ¿Por qué NAT?

- Escasez de direcciones IPv4 públicas.
- Oculta la topología interna, añadiendo una capa básica de seguridad.
- Permite superponer rangos privados idénticos tras un mismo router (NAT VRF).

---

2 Tipos comunes de NAT

Tipo	Traduce	IPs públicas usadas	Ejemplo IOS
Estático	1 : 1 (privada pública)	Igual a nº hosts	`ip nat inside source static 10.0.0.10 203.0.113.10`
Dinámico	Pool hosts (sin puertos)	Pool tamaño N	`ip nat inside source list 1 pool INTERNETPOOL`
PAT/Overload	Muchos 1 usando puertos	1 (o pocas)	`ip nat inside source list 1 interface Gi0/0 overload`

---

3 Configuración paso a paso (PAT)

3.1 Declarar interfaces inside/outside

```
interface Gig0/0
 ip address 203.0.113.2 255.255.255.248
 ip nat outside

interface Gig0/1
 ip address 192.168.10.1 255.255.255.0
 ip nat inside
```

3.2 ACL de origen interno

```
access-list 1 permit 192.168.10.0 0.0.0.255
```

3.3 Habilitar NAT overload

```
ip nat inside source list 1 interface Gig0/0 overload
```

> *Resultado*: todos los hosts 192.168.10.0/24 salen a Internet usando 203.0.113.2 con puertos dinámicos.

---

4 Dinámico con pool (opcional)

```
ip nat pool NETPUB 198.51.100.10 198.51.100.20 prefix-length 29
ip nat inside source list 1 pool NETPUB
```

El router asigna la primera IP libre del pool a cada flujo hasta agotarlo.

---

5 Verificación rápida

```
Router# show ip nat translations
Router# show ip nat statistics
Router# clear ip nat translation *   ! para pruebas de laboratorio
```

Campos clave: **Inside global** (IP pública), **Inside local** (IP privada).

---

6 Troubleshooting

Síntoma / Mensaje	Causa probable	Comando útil
Hosts sin Internet	Falta `ip nat inside/outside`	`show run interface`
Traducciones se quedan en 0	ACL mal definida	`show access-lists 1`
Pool agotado (dinámico)	Demasiados hosts / pool corto	`show ip nat statistics`
Paquetes ICMP fallan salida	NAT traumatiza checksums old IOS	`ip nat service icmp timeout`

---

7 Buenas prácticas

1. Excluir la IP del router de NAT para servicios de gestión (SSH/Telnet).
2. Registrar logs de NAT (`ip nat log translations syslog`) en firewalls de borde.
3. Controlar timeouts (`ip nat translation tcp-timeout 300`).
4. Documentar pools y mapeos estáticos en la CMDB.

Lecturas recomendadas

- Cisco IOS — *Configuring NAT Overload*.
- RFC 3022 — *Traditional NAT*.
- Cisco Press — *CCNP Enterprise Advanced Routing*, cap. 5.

BGP: fundamentos y términos clave

Border Gateway Protocol (BGP) es el protocolo de enrutamiento exterior que interconecta Sistemas Autónomos (AS) en Internet y redes privadas grandes. Trabaja con el algoritmo de *path-vector*, anunciando rutas junto a atributos que permiten tomar decisiones de encaminamiento precisas y controladas.

---

1 Conceptos básicos

- **AS (Autonomous System)**: conjunto de routers bajo una política común.
- **eBGP**: sesión entre AS diferentes (p. ej. tu empresa ISP).
- **iBGP**: sesión dentro del mismo AS (core MPLS, data center).
- **Router Reflector (RR)**: reduce malla full-mesh iBGP.
- **MED, LOCAL_PREF**: atributos usados para influir la selección de ruta.

---

2 Tipos de mensajes BGP

Tipo de mensaje	Nº	Propósito
OPEN	1	Negociar versión, ASN, hold-time, RID
UPDATE	2	Anunciar / retirar prefijos + atributos
NOTIFICATION	3	Señalar error y cerrar sesión
KEEPALIVE	4	Mantener sesión (default 60 s)

---

3 Atributos principales

Atributo	Oblig.	Predet.	Influye en…
ORIGIN	Y	igp	Orden de preferencia
AS_PATH	Y	—	Longitud de ruta
NEXT_HOP	Y	—	Dirección siguiente
LOCAL_PREF	iBGP	100	Salida preferida AS
MED	opc	0	Entrada preferida
COMMUNITIES	opc	—	Política flexible

Y = obligatorio en UPDATE; opc = opcional.

---

4 Ejemplo mínimo de configuración eBGP

```
! Empresa AS 65010
router bgp 65010
 bgp log-neighbor-changes
 neighbor 203.0.113.1 remote-as 65000
 neighbor 203.0.113.1 description ISP-PRIMARY
 network 198.51.100.0 mask 255.255.255.0   ! Anunciar /24 público
```

Nota: en eBGP, TTL = 1; enlaces no directos necesitan `neighbor ebgp-multihop`.

---

5 Verificación esencial

```
Router# show ip bgp summary
Router# show ip bgp neighbors 203.0.113.1 received-routes
Router# show ip bgp 0.0.0.0                                  ! ruta específica
```

Campos clave: **State/PfxRcd**, **Up/Down**, **BGP table version**.

---

6 Troubleshooting rápido

Síntoma / Log	Posible causa	Comando útil
Active en `show ip bgp summary`	Filtro ACL / TCP 179 bloque	`telnet x.x.x.x 179`
Cambios de ruta lentos	`bgp scan-time` muy alto	Ajustar a 15-30 s
Ruta no entra en tabla IP	Mejor ruta por IGP / overlap	`show ip bgp rib-failure`

---

7 Buenas prácticas

1. Filtrar prefijos con **prefix-lists** para evitar *leak* masivo.

2. Usar **password (MD5)** en todas las vecindades (`neighbor password xyz`).

3. Documentar políticas: LOCAL_PREF, MED, communities propietarias.
4. Habilitar **Graceful Restart** si el hardware lo soporta.

Lecturas recomendadas

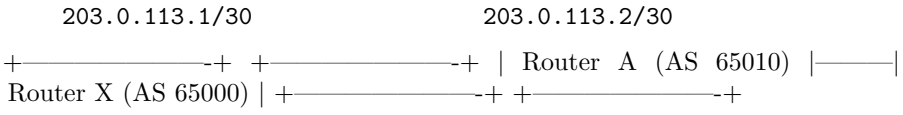
- RFC 4271 — *A Border Gateway Protocol 4 (BGP-4).*
- Cisco IOS — *BGP Configuration Guide.*
- Doyle, J. — *Routing TCP/IP Vol. 2*, caps. 3-5.

BGP básico entre dos AS

Esta guía muestra la configuración mínima de **eBGP** entre dos Sistemas Autónomos (AS) conectados por un único enlace.
Ejemplo: Empresa A (AS 65010) ISP X (AS 65000).

---

1 Topología de referencia



La empresa anuncia su prefijo público **198.51.100.0/24**.
El ISP redistribuye la tabla global (por simplicidad, un *default*).

---

2 Requisitos previos

1. Numero AS asignado (público o private ASN 64512-65534 para pruebas).
2. Ruta estática /32 o loopback que será el **Router ID** (recomendado).
3. ACL/prefix-list listos para filtrar rutas no deseadas.

---

3 Configuración paso a paso

3.1 Empresa A — Router A (AS 65010)

```
hostname Rtr-A
!
interface Gig0/0
 description eBGP-to-ISP
 ip address 203.0.113.2 255.255.255.252
 no shutdown
!
router bgp 65010
 bgp log-neighbor-changes
 neighbor 203.0.113.1 remote-as 65000
 neighbor 203.0.113.1 description ISP-X-Primary
!
 network 198.51.100.0 mask 255.255.255.0   ! anuncia /24
!
exit
```

3.2 ISP X — Router X (AS 65000)

```
hostname ISP-X-PE
!
interface Gig0/0
 ip address 203.0.113.1 255.255.255.252
 no shutdown
!
router bgp 65000
 neighbor 203.0.113.2 remote-as 65010
 neighbor 203.0.113.2 description Customer-A
!
 address-family ipv4
  neighbor 203.0.113.2 activate
  neighbor 203.0.113.2 send-community both
 exit-address-family
!
 ip route 198.51.100.0 255.255.255.0 Null0   ! evita loops entrantes
```

---

4 Verificación esencial

```
Rtr-A# show ip bgp summary
Rtr-A# show ip bgp neighbors 203.0.113.1 received-routes
Rtr-A# show ip bgp 0.0.0.0

ISP-X# show ip bgp summary
ISP-X# show ip bgp 198.51.100.0
```

Campos clave: **State/PfxRcd = Established/1**, **Up/Down**, **Version**.

---

5 Orden simplificado de selección de rutas

Criterio	Prioridad
Highest **LOCAL_PREF**	1
Shortest **AS_PATH**	2
Lowest **ORIGIN** (IGP < EGP < INCOMPLETE)	3
Lowest **MED**	4
eBGP > iBGP	5
Lowest IGP metric to NEXT_HOP	6

---

6 Troubleshooting rápido

Síntoma / Log	Causa probable	Comando útil
`Active` en `show ip bgp summary`	TCP 179 bloqueado / sin reach	`telnet 203.0.113.1 179`
Sesión cae cada 3 min	Mismatch keepalive/hold-time	`show ip bgp neighbors`
Ruta /24 no aceptada por ISP	Falta longitud / origin IGP	`show ip bgp neigh adv-routes`

---

7 Buenas prácticas

1. **Filtrar** — aplica *prefix-lists* de entrada/salida para evitar *route leaks*.
2. Establece **password MD5** (`neighbor password ...`) en cada sesión.
3. Habilita **BFD** para detección rápida de fallos de enlace (< 1 s).
4. Documenta políticas (LOCAL_PREF, COMMUNITIES) en tu playbook NOC.

Lecturas recomendadas

- Cisco IOS — *BGP Configuration Guide (Fundamentals)*.
- RFC 4271 — *Border Gateway Protocol 4 (BGP-4)*.
- Cisco Press — *CCNP Enterprise Advanced Routing*, cap. 7.

STP y variantes (RSTP) en Cisco IOS

El **Spanning Tree Protocol (STP)** previene bucles de capa 2 bloqueando enlaces redundantes hasta que sean necesarios.
Su evolución, **Rapid STP (RSTP, 802.1w)**, acelera la convergencia a < 1 s en la mayoría de casos.

---

1 Comparativa rápida

Modo	Estándar IEEE	Convergencia típica	Estados de puerto
PVST+	802.1D (Cisco)	30-50 s	BLK › LISTEN › LRN › FWD
RPVST+	802.1w (Cisco)	1-3 s	DISC › LRN › FWD
MSTP	802.1s	1-3 s	Igual que RSTP

---

2 Roles y estados STP clásicos (802.1D)

Rol de puerto	Función principal	Estado
Root	Mejor camino hacia Root Bridge	FWD / BLK (según BPDU)
Designated	Reenvía BPDU al segmento	FWD
Alternate	Redundancia, espera fallo principal	BLK
Disabled	Administrativamente apagado	DIS

---

3 Configuración básica

3.1 Elegir Bridge Priority

```
switch(config)# spanning-tree vlan 10 priority 24576   ! Root primario
switch(config)# spanning-tree vlan 10 priority 28672   ! Root secundario
```

3.2 Cambiar a RPVST+

```
switch(config)# spanning-tree mode rapid-pvst
```

3.3 PortFast y BPDU Guard en puertos de usuario

```
interface range fa0/1 - 24
 spanning-tree portfast
 spanning-tree bpduguard enable
```

4 Verificación esencial

```
Switch# show spanning-tree vlan 10 root
Switch# show spanning-tree interface gi0/1 detail
Switch# show spanning-tree summary
```

Campos clave: **Root ID**, **Cost**, **Port Role/State**, **Number of topology changes**.

5 Troubleshooting rápido

Síntoma / Log	Posible causa	Comando útil
Puertos tardan 30 s en subir	Falta PortFast	`show run int`
BPDU Guard shutdown	Host enviado BPDU	`show errdisable recovery`
Cambios de topología frecuentes	Cable flap / bucle	`show spanning-tree detail`

6 Buenas prácticas

1. Definir **Root primario/secundario** por VLAN para control de L2.
2. Habilitar **PortFast + BPDU Guard** en acceso; **Root Guard** en distribución.
3. Usar **RSTP (rapid-pvst)** para convergencia rápida en campus.
4. Documentar prioridades, costos y topologías en diagramas actualizados.

Lecturas recomendadas

- Cisco IOS — *Spanning Tree Configuration Guide*.
- IEEE 802.1D-2004 — *MAC Bridges*.
- IEEE 802.1w-2001 — *Rapid Reconfiguration*.
- Healy & Lowenthal — *CCNP Enterprise Switching v8*, cap. 4.

EtherChannel (PAgP & LACP) en Cisco IOS

EtherChannel agrupa varios enlaces físicos en un único canal lógico, aumentando el ancho de banda y la redundancia sin necesidad de STP bloqueado. Los switches negocian el canal con **PAgP** (Cisco propietario) o **LACP** (estándar IEEE 802.3ad).

---

1 Ventajas clave

- Multiplica el throughput (hasta 8×, según número de enlaces).
- Convergencia inmediata ante fallo de un miembro; tráfico se redistribuye.
- STP ve el canal como un único puerto → cero enlaces bloqueados.

---

2 Modos de negociación

Protocolo	Modo activo	Modo pasivo	Comentario
PAgP	desirable	auto	Solo Cisco; máx. 8 links
LACP	active	passive	IEEE 802.3ad; máx. 16 (8 activos + 8 hot-standby)

---

3 Configuración paso a paso (LACP)

3.1 Seleccionar puertos

```
interface range Gi0/1 - 2
 switchport mode trunk
 channel-group 1 mode active
```

> *Tip*: todos los puertos de un canal deben compartir velocidad, dúplex y VLAN.

3.2 Ajustar parámetros del Port-Channel

```
interface Port-channel1
 description Uplink-to-Dist01
 switchport trunk allowed vlan 10,20,30
 spanning-tree portfast trunk
```

---

4 Verificación esencial

```
Switch# show etherchannel summary
Switch# show interfaces port-channel 1
Switch# show lacp neighbor
```

Campos clave: **P** (port-channel up), **SU/LU** (LACP activo), **Age**, **Agg**.

---

5 Algoritmos de distribución

Valor por defecto: `src-dst-ip`. Se cambia así:

```
Switch(config)# port-channel load-balance src-dst-mac
```

Opción	Uso típico
src-dst-ip	Traf. variado L3 (campus)
src-dst-mac	Data center L2
src-port	Entornos con pocos flujos grandes

---

6 Troubleshooting rápido

Síntoma	Posible causa	Comando útil
Puertos en **suspended**	Modo/Speed mismatch	`show interface Gi0/1 cap`
Tráfico no balanceado	Algoritmo inadecuado	`show etherchannel load-balance`
Flaps de Port-Channel	Diferentes native VLAN	`show run int`

---

7 Buenas prácticas

1. Establecer **descriptions** coherentes en cada miembro y Port-Channel.
2. Mantener mismo tipo de medio; no mezclar cobre y fibra.
3. Habilitar **LACP fallback** (warm-standby) en chasis modernos.
4. Documentar ID del Canal, VLANs permitidas y algoritmo en tu CMDB.

Lecturas recomendadas

- Cisco IOS — *EtherChannel Configuration Guide*.
- IEEE 802.3-2022 — *Clause 43: Link Aggregation*.
- Cisco Press — *CCNP Enterprise Switching*, cap. 6.

QoS: clasificación y colas en Cisco IOS

La **Calidad de Servicio (QoS)** permite priorizar aplicaciones críticas (voz, vídeo, datos sensibles) frente a tráfico best-effort, garantizando ancho de banda, baja latencia y menor pérdida de paquetes.

---

1 Tres fases de QoS

Fase	Acción principal	Ejemplo de comando
Clasificación	Identificar tráfico (ACL, DSCP)	`class-map match dscp ef`
Marcado	Reescribir campo DSCP o CoS	`set dscp af41`
Encolado/Policía	Asignar colas, limitar tasas	`bandwidth percent 30` / `police`

---

2 Modelos de QoS

Modelo	Dónde se aplica	Resumen rápido
Best Effort	Sin QoS	Todo tráfico igual
IntServ	RSVP, reserva por flujo	Escalable solo en redes pequeñas
DiffServ	DSCP/PHB por salto	Estándar de facto en campus/ISP

---

3 Clasificación DiffServ (DSCP)

Clase	DSCP	Prioridad típica	Ejemplo de tráfico
EF	46	Muy alta	Voz IP (VoIP)
AF41	34	Alta	Vídeo conferencias
AF21	18	Media	Aplicaciones críticas
BE	0	Baja (default)	Navegación web

---

4 Ejemplo MQC: VoIP prioritario

```
class-map match-any VOICE
 match ip dscp ef

policy-map CAMPUS_QOS
 class VOICE
  priority percent 10        ! cola LLQ, garantiza latencia baja
 class class-default
  fair-queue

interface Gig0/1
 service-policy output CAMPUS_QOS
```

---

5 Verificación rápida

```
Router# show policy-map interface gi0/1
Router# show class-map VOICE
Router# show mls qos interface statistics
```

Campos clave: **Priority**, **Queue depth**, **Drop packets**.

---

6 Troubleshooting

Síntoma	Posible causa	Comando útil
Latencia alta en VoIP	Falta LLQ o policing	`show policy-map int`
Drops en colas AFxx	Bajas colas o burst alto	Ajustar `bandwidth/queue-limit`
DSCP reescrito inesperado	Mismatch trust boundary	`show mls qos maps dscp-out`

---

7 Buenas prácticas

1. Definir **trust boundary** en el primer switch; usar `mls qos trust dscp`.
2. Priorizar tráfico **EF** con LLQ, nunca exceder 33 % del enlace.
3. Evitar **policing** a voz; preferir shaping y WRED para datos.
4. Documentar políticas y perfiles DSCP aplicación en CMDB.

Lecturas recomendadas

- Cisco IOS — *QoS Configuration Guide*.
- RFC 4594 — *Configuration Guidelines for DiffServ*.
- Cisco Press — *End-to-End QoS Network Design*, 2ª ed.

IPv6 y SLAAC rápido

IPv6 resuelve la escasez de direcciones IPv4 y simplifica la fragmentación de redes, eliminando NAT y ofreciendo auto-configuración nativa mediante **SLAAC**.

---

1 Formato de dirección

- 128 bits (8 hextetos de 16 bits).
- Compresión con `::` para ceros consecutivos.
- Ejemplo: `2001:0db8:0123:0000:0000:0000:0045:6789` → `2001:db8:123::45:6789`.

---

2 Tipos de direcciones

Tipo	Prefijo	Propósito
Unicast global	2000::/3	Enrutamiento público
Link-local	fe80::/10	Vecindad, ND, SLAAC
Multicast	ff00::/8	Alcance uno-a-muchos
Unique Local (ULA)	fc00::/7	Privado (similar a RFC 1918)

---

3 Auto-configuración SLAAC

1. El host genera **link-local** (`fe80::/64`) usando EUI-64 o random.
2. Envía **Router Solicitation (RS)** al multicast `ff02::2`.
3. El router responde con **Router Advertisement (RA)**:
   - Prefijo /64 a usar.
   - Flag `A` (Autonomous) → permite SLAAC.
   - Flag `M` (Managed) → DHCPv6 stateful.

Comando en IOS:

```
interface Gig0/1
 ipv6 address 2001:db8:10:1::1/64
 ipv6 nd other-config-flag         ! DHCPv6 para info DNS
 ipv6 nd prefix 2001:db8:10:1::/64 300 30
```

4 Vecinos y ND (Neighbor Discovery)

```
Router# show ipv6 neighbors
Router# show ipv6 interface Gig0/1
```

Campos clave: **Reach/Delay**, **Stale**, **Router?**.

5 ICMPv6 mensajes esenciales

Mensaje	Código	Uso
RS	133	Solicitar RA
RA	134	Anunciar prefijos, MTU, flags
NS	135	Consultar MAC de un IPv6
NA	136	Respuesta al NS
Redirect	137	Optimizar ruta

6 Troubleshooting

Síntoma	Posible causa	Comando útil
Host sin dirección global	RA bloqueado (ACL)	`debug ipv6 nd`
Duplicated address	DAD falla / MAC duplicada	`show log`
Ping falla a link-local	Falta `%interface` en destino	`ping fe80::1%Gig0/1`

7 Buenas prácticas

1. Habilitar **RA Guard** y **ND Inspection** en acceso.
2. Implementar **DHCPv6-PD** para redes cliente dinámicas.
3. Usar **ULA** para servicios internos + NAT64 como puente a IPv4.
4. Documentar bloques /48 y delegaciones /64 en CMDB.

Lecturas recomendadas

- RFC 4291 — *IPv6 Addressing Architecture.*
- Cisco IOS — *IPv6 Configuration Guide.*
- RFC 4862 — *IPv6 Stateless Address Autoconfiguration.*

Dual-Stack IPv4/IPv6 en Cisco IOS

Ejecutar **dual-stack** significa habilitar IPv4 y IPv6 simultáneamente en todas las interfaces, permitiendo una migración gradual y compatibilidad con equipos legacy. Este enfoque evita "huellas de NAT" y minimiza riesgo.

---

1 Motivos para elegir dual-stack

- Transición **sin impacto**: los hosts pueden usar el protocolo que soporte el destino.
- No requiere encapsulación (a diferencia de 6to4, GRE6).
- Permite desplegar servicios nativos IPv6 (VoIP, IoT) conservando IPv4.

---

2 Pasos de configuración básicos

```
interface Gig0/1
 description LAN-Campus
 ip address 192.0.2.1 255.255.255.0
 ipv6 address 2001:db8:10:1::1/64
 no shutdown

ipv6 unicast-routing          ! habilita forwarding global
```

Nota: cada subred IPv6 suele ser /64; evita subredes más pequeñas salvo casos especiales.

---

3 Tabla de coexistencia de servicios

Servicio	IPv4 requerido	IPv6 requerido	Observaciones
DNS	Y	Y	AAAA para IPv6, A para IPv4
DHCP / DHCPv6	Y	opc / SLAAC	Ambos pueden coexistir
ACL / Firewall	Y	Y	Políticas separadas (ipv6 acl)
SNMP / Syslog	Y	N	Activa v2c/v3 sobre v6 si aplica

---

4 Opciones de transición complementarias

Opción	Cuándo usar	Notas rápidas
NAT64 / DNS64	Clientes solo-IPv6 servidores IPv4	Requiere motor NAT64
6rd	ISP solo IPv4, despliegue rápido v6 túnel	Tunelado sobre IPv4
Dual-stack lite (DS-Lite)	CGNAT + IPv6 WAN en ISP	CPE encapsula IPv4 en v6

---

5 Verificación esencial

```
Router# show ipv6 interface brief
Router# show running-config interface Gig0/1
Router# ping 2001:db8::10 source 2001:db8:10:1::1
Router# ping 192.0.2.10 source 192.0.2.1
```

---

6 Troubleshooting rápido

Síntoma	Posible causa	Comando útil
Hosts obtienen solo IPv4	RA bloqueado / SLAAC off	`debug ipv6 nd`
IPv6 reachability OK, IPv4 no	ACL v4 bloquea, routings	`show ip route` / ACL review
DNS AAAA responde, pero no ping	Firewall bloquea ICMPv6	Ver reglas ICMPv6 permitidas

---

7 Buenas prácticas

1. Definir **boundary de confianza**: `ipv6 nd raguard`, DHCPv6 guard.
2. Mantener plan IP: asignar /64 por VLAN; documentar en CMDB.
3. Actualizar ACL: crear listas IPv6 espejo de las IPv4 existentes.
4. Supervisar con **SNMPv3 sobre IPv6** y exportar NetFlow v9/IPFIX v6.

Lecturas recomendadas

- Cisco IOS — *IPv6 Configuration Guide.*

- RFC 4213 — *Basic Transition Mechanisms for IPv6 Hosts and Routers.*
- RIPE-690 — *IPv6 prefix assignment for end-users.*

Interfaces loopback: usos y configuración

Una **loopback** es una interfaz lógica siempre **up/up** a menos que se deshabilite administrativamente. Al no depender de la capa física, resulta ideal para funciones de identificación, gestión y pruebas.

---

1 Principales casos de uso

Caso	Motivo
Router ID (OSPF/BGP)	Estable, no fluctúa con fallos físicos
Acceso de **gestión**	Telnet/SSH puntual, SNMP, Syslog
Pruebas de reachability	Ping/Traceroute sin afectar data path
Origen de **túneles GRE/IPsec**	Mantiene túnel activo tras failover

---

2 Crear una loopback en IOS

```
interface loopback 0
 description RID_OSPF
 ip address 10.255.255.1 255.255.255.255
```

Tip: usa /32 para ahorrar espacio y simplificar summarización.

---

3 Asignar Router ID manual

3.1 OSPF

```
router ospf 1
 router-id 10.255.255.1
```

3.2 BGP

```
router bgp 65010
 bgp router-id 10.255.255.1
```

---

4 Loopback como origen de túneles

```
interface Tunnel0
 ip address 172.16.0.1 255.255.255.252
 tunnel source 10.255.255.1
```

```
tunnel destination 198.51.100.2
```

Si el enlace WAN físico cambia de IP, el túnel permanece al re-encaminarse al nuevo next-hop mientras el loopback siga anunciada.

---

5 Verificación rápida

```
Router# show ip interface loopback0
Router# show ip ospf | include Router ID
Router# show ip bgp summary | include Router identifier
```

---

6 Troubleshooting

Síntoma	Causa probable	Comando útil
Loopback sin llegar a peers	Falta anuncio en IGP	`show ip route 10.255.255.1`
Router ID no coincide	Loopback con IP menor elegida	Asignar manual `router-id`
Túnel cae tras failover	Ruta a loopback no actualiza	Revisar OSPF/BGP

---

7 Buenas prácticas

1. Reservar **/32** loopbacks contiguos por sitio (p.ej. 10.255.A.0/24).
2. Anunciar loopbacks en IGP **con coste bajo** para asegurar reachability.
3. Documentar nombre / descripción: `RID_BGP`, `MGMT_SNMP`, etc.
4. Evitar que ACL bloqueen pings a loopbacks utilizados para monitorización.

Lecturas recomendadas

- Cisco IOS — *Loopback Interfaces Configuration Guide.*
- Doyle, J. — *Routing TCP/IP Vol.1*, pág. 130-135.

- RFC 3330 — *Special-Use IPv4 Addresses* (para bloques de prueba).

Enlaces seriales & clock rate en Cisco IOS

Aunque los enlaces seriales WAN (T1/E1, DS3) han sido reemplazados en muchos entornos por Ethernet y fibra, siguen presentes en laboratorios, entornos industriales y enlaces satelitales. Conocer su configuración básica evita problemas de sincronismo y ancho de banda.

---

1 Conceptos esenciales

Término	Significado
DCE / DTE	El equipo **DCE** (Data Circuit-terminating Equipment) proporciona reloj; **DTE** lo recibe.
Clock rate	Frecuencia en bps que sincroniza el enlace.
Encapsulado	Formato de trama: HDLC (por defecto), PPP, Frame-Relay.

---

2 Identificar DCE vs DTE

```
Router# show controllers serial 0/0/0
...
Hardware is GT96K
DCE V.35, clock rate 64000
```

Si la salida refleja **DCE**, el router debe suministrar reloj al peer.

---

3 Configurar clock rate y encapsulado

```
interface Serial0/0/0
 description LAB-WAN
 ip address 10.10.1.1 255.255.255.252
 encapsulation ppp
 clock rate 64000       ! solo necesario en extremo DCE
 no shutdown
```

Velocidades comunes (kbps): 64000, 128000, 1544000 (T1), 2048000 (E1).

---

4 Tabla de encapsulados seriales

Encapsulado	Características principales	Comando de ajuste
HDLC (def.)	Propietario Cisco, sin auth	`encapsulation hdlc`
PPP	Estándar, CHAP/PAP, multilink	`encapsulation ppp`
Frame-Relay	Redes conmutadas, DLCI	`encapsulation frame-relay`

---

5 Verificación rápida

```
Router# show interfaces serial0/0/0
Router# show controllers serial0/0/0
Router# show ppp multilink       ! si aplica
```

Campos clave: **Clock rate**, **Encapsulation**, **CRC errors**, **TX/RX rate**.

---

6 Troubleshooting

Síntoma / Log	Causa probable	Comando útil
Serial up, line protocol down	Encapsulado mismatch	Verificar `encapsulation`
Conteo de CRC elevado	Capa física dañada	Revisar cable/V.35
Looped! en interface	Loopback local habil.	`no loopback`
Clock rate no mostrado (DTE)	Peer es DCE	Configurar clock en peer

---

7 Buenas prácticas

1. Etiquetar físicamente cables **DCE** para evitar confusiones.
2. Usar **PPP + CHAP** en entornos producción para autenticación.
3. Monitorizar errores con SNMP/OID de interfaz (CRC, input errors).
4. Documentar topología serial: velocidades, DLCI, encapsulado.

Lecturas recomendadas

- Cisco IOS — *Serial Interface Configuration Guide*.

- RFC 1661 — *Point-to-Point Protocol (PPP).*
- Doyle, J. — *Routing TCP/IP Vol. 1*, secc. 2-4.

OSPF y el comando *passive-interface*

El comando **passive-interface** impide el envío y recepción de paquetes OSPF (*Hello*, LSA) en una interfaz concreta, pero **sigue anunciando** la red al resto del proceso. Es la forma más sencilla de reducir ruido de control y minimizar vectores de ataque en puertos de usuario.

---

1 Cuándo usarlo

- Puertos de acceso que conectan PCs o impresoras: no hay otro router con OSPF.
- Interfaces loopback: nunca formarán vecindad, pero deben aparecer en la LSDB.
- Enlaces punto-multipunto (p.ej. Frame-Relay hub-and-spoke) donde solo el hub
  necesita enviar Hellos.

---

2 Comandos clave

```
! Método clásico por interfaz
router ospf 1
 passive-interface Gig0/1

! Invertir la lógica (todas pasivas salvo las que indico)
router ospf 1
 passive-interface default
 no passive-interface Gig0/0      ! WAN hacia el vecino
 no passive-interface Serial0/0/0 ! otro enlace router-router
```

> **Tip**: usa *passive-interface default* en redes de campus y desactiva solo
> los uplinks. Ahorra escribir decenas de líneas.

---

3 Verificación rápida

```
Router# show ip ospf interface brief | include P-   ! P- indica passive
Router# show ip route ospf | include Passive
```

Campos clave:
* **Passive** en la columna *State*.
* Las rutas *O* (intra-área) se siguen instalando.

---

4 Impacto operativo

- Reduce la **LSDB**: menos vecinos, menos Hellos.
- Mejora la **seguridad**: evita que dispositivos maliciosos envíen Hellos y LSAs.
- No afecta al cálculo SPF; simplemente marca la interfaz como "stub" local.

---

5 Buenas prácticas

1. Documenta qué interfaces quedaron pasivas (comentarios en el *running-config*).
2. Combínalo con `ipv6 ospf passive-interface` en despliegues dual-stack; los
   modos son independientes.
3. Si desactivas una interfaz WAN temporalmente, ponla en *passive* para evitar
   spam de *Neighbor Down* en los logs.

Lecturas recomendadas

- Cisco IOS — *OSPF Configuration Guide* (sección *Passive-Interface*).
- RFC 2328 — *OSPF Version 2*, apartado 9.7.

Alta disponibilidad con HSRP

El **Hot Standby Router Protocol (HSRP)** permite que dos o más routers presenten una única **IP virtual (gateway)** a la LAN.
Si el router activo falla, el standby asume el rol en < 1 s (HSRP v2 fast).

---

1 Componentes y roles

Elemento	Función
IP virtual	Default-gateway que usan los hosts
Activo	Reenvía tráfico, envía hello cada 3 s
Standby	Escucha hellos; toma control al detectar fallo
Listen/Others	Miembros sin prioridad para ser standby

---

2 Configuración mínima (HSRP v1)

```
interface Vlan10
 ip address 192.168.10.2 255.255.255.0
 standby 10 ip 192.168.10.1          ! Grupo 10, IP virtual
 standby 10 priority 110             ! >100 para ser Active
 standby 10 preempt                  ! Recupera rol si vuelve
 standby 10 authentication md5 key-string C1sc0
```

En el router de respaldo:

```
interface Vlan10
 ip address 192.168.10.3 255.255.255.0
 standby 10 ip 192.168.10.1
 standby 10 priority 90
 standby 10 preempt
 standby 10 authentication md5 key-string C1sc0
```

---

3 Timers y ajustes

Parámetro	Default v1	Default v2 Fast	Comando
Hello	3 s	1 s	`standby 10 timers 1 3`
Hold	10 s	3 s	`standby 10 timers msec 250 750`

Mantén **Hold** **3× Hello** para evitar flapping.

---

4 Verificación rápida

```
R1# show standby brief
R1# show standby vlan 10
R2# debug standby events          ! solo en laboratorio
```

Campos clave: **Active router**, **Standby router**, **Virtual IP**, **State**.

---

5 Troubleshooting

Síntoma / Log	Causa probable	Comando útil
Ambos routers en **active**	Auth mismatch / timers	`show standby vlan 10`
Estado **Init** permanente	Hellos bloqueados (ACL)	Verificar ACL / VLAN
Cambios de estado frecuentes	Flapping de interfaz	`show log`, cableado

---

6 Buenas prácticas

1. Igualar **prioridad** a la capacidad del dispositivo (más potente = mayor).
2. Usar **preempt** + delay (`standby 10 preempt delay 60`) tras reboot.
3. Distribuir carga con **múltiples grupos** (HSRP load-sharing).
4. Monitorizar interfaces críticas (`standby 10 track Gig0/1 20`).
5. Sincronizar reloj NTP para analizar logs de transición.

Lecturas recomendadas

- Cisco IOS — *HSRP Configuration Guide.*
- RFC 2281 — *Cisco Hot Standby Router Protocol (HSRP).*
- Cisco Press — *CCNP Enterprise Advanced Routing*, cap. 9.

VPN IPsec site-to-site

Una **VPN IPsec site-to-site** conecta de forma segura dos redes privadas a través de Internet cifrando el tráfico entre los routers de borde. En Cisco IOS se configura en dos fases: **IKEv1/IKEv2 (negociación)** y **ESP/AH** (cifrado de datos).

---

1 Topología de referencia

```
      198.51.100.1                                  203.0.113.1

+———————-+ WAN +———————-+ | Rtr-MAD (HQ) |————-| Rtr-NYC (Branch) | | LAN 10.10.0.0/24 | | LAN 10.20.0.0/24 | +———————-+ +———————-+
```

---

2 Parámetros recomendados

Fase	Algoritmos / valores	Comentario
IKE Phase 1	AES-256, SHA-256, DH Group 14	Intercambio de claves
IKE Phase 2	AES-256, SHA-256, PFS Group 14	Protección del tráfico
Vida	86400 s (P1), 3600 s (P2)	Renovación automática

---

3 Configuración paso a paso (IKEv1, crypto map)

3.1 Crear políticas IKE Phase 1

```
crypto isakmp policy 10
 encr aes 256
 hash sha256
 authentication pre-share
 group 14
 lifetime 86400
crypto isakmp key My$trongKey address 203.0.113.1
```

3.2 Definir lista de acceso de tráfico interesante

```
access-list 110 permit ip 10.10.0.0 0.0.0.255 10.20.0.0 0.0.0.255
```

3.3 Configurar transform-set y crypto map

```
crypto ipsec transform-set AES256-SHA esp-aes 256 esp-sha256-hmac
```

```
 mode tunnel

crypto map CMAP 10 ipsec-isakmp
 set peer 203.0.113.1
 set transform-set AES256-SHA
 set pfs group14
 match address 110
 set security-association lifetime seconds 3600
```

3.4 Aplicar crypto map a la interfaz WAN

```
interface Gig0/0
 description Internet
 ip address 198.51.100.1 255.255.255.252
 crypto map CMAP
 no shutdown
```

Repite el mismo proceso en Rtr-NYC ajustando IPs y ACL (inversa).

---

4 Verificación esencial

```
Rtr-MAD# show crypto isakmp sa
Rtr-MAD# show crypto ipsec sa
Rtr-MAD# ping 10.20.0.1 source 10.10.0.1
```

Campos clave: **QM_IDLE** en ISAKMP SA y contadores ESP creciendo.

---

5 Troubleshooting rápido

Síntoma / Log	Posible causa	Comando útil
`MM_NO_STATE` en ISAKMP	Pre-shared key errónea	Verificar claves y frases
no proposal chosen	Algoritmos no coinciden	`show crypto isakmp sa det`
Paquetes encriptados = 0	ACL tráfico interesante	Revisar ACL inversa
Tunel cae cada 60 s	NAT traversal bloqueado	`crypto isakmp nat-traversal 20`

---

6 Buenas prácticas

1. Usar **IKEv2** en despliegues nuevos (`crypto ikev2 proposal`).
2. Habilitar **PFS** para mayor seguridad (grupo 14).

3. Registrar **syslog 714051** para caídas de SA.

4. Monitorizar con SNMP OID `1.3.6.1.4.1.9.9.171` (CISCO-IPSEC-FLOW-MONITOR).

5. Documentar claves, ACL y lifetimes en la CMDB; rotar claves anualmente.

Lecturas recomendadas

- Cisco IOS — *IPsec VPN Site-to-Site Configuration Guide*.
- RFC 4301 — *Security Architecture for IPsec*.
- RFC 7296 — *Internet Key Exchange Protocol Version 2 (IKEv2)*.

Microsegmentación con ACI y NSX

La **microsegmentación** divide la red en segmentos lógicos a nivel de flujo o carga de trabajo, aplicando políticas de seguridad granulares que acompañan a la aplicación sin depender de la topología física.

---

1 Tecnologías líderes

Plataforma	Modelo de política	Motor de datos
Cisco ACI	*Endpoint Groups (EPG)* + Contracts	VXLAN sobre hardware ASIC
VMware NSX	*Distributed Firewall* + Security Groups	ESXi vSwitch + OVS

---

2 Flujo de alto nivel (ACI)

```
1. Descubrimiento    → BD aprende IP/MAC y asigna a un EPG (contrato implícito).
2. Clasificación     → EPG se mapea a un BD/VRF; aplica un contrato (ACL TCAM).
3. Forward + Policy  → Leaf switch etiqueta el paquete VXLAN (VNID) y aplica reglas.
```

> *Tip*: los **Contracts** definen filtros L4-L7 (permitir HTTP, deny ICMP, etc.).

---

3 Flujo de alto nivel (NSX)

```
1. DFW     → Cada vNIC tiene reglas L2-L4 evaluadas inline (kernel fast-path).
2. Service Composer → Agrupa VMs por etiquetas (Security Groups).
3. Edge Services → SNAT, LB, VPN opcionales en túneles Geneve.
```

---

4 Ejemplo rápido de contrato ACI (permitir HTTPS)

```
# GUI o JSON API; CLI simplificado
contract web-sec {
  filter https-filter {
    entry tcp dst 443 permit
  }
}
apply contract web-sec between EPG_web and EPG_db
```

---

5 Ventajas vs VLAN tradicional

- Políticas **incrustadas** en los servidores/hypervisores → menos ACL en core.
- Segmentación basada en **identidad** (etiquetas, VM name) en lugar de IP fija.
- Automatización vía API (Terraform, Ansible) reduce errores manuales.

---

6 Desafíos comunes

Desafío	Mitigación recomendada
Visibilidad de políticas	Usa *Policy Analytics* / Trace Flow
Migración de VLAN a EPG/SG	Mapear gradualmente, usar modo monitor
Operaciones multi-tenant	Separar VRF / T0-T1 (NSX)

---

7 Buenas prácticas

1. Etiqueta cargas desde CI/CD; las políticas nacen con la VM/Pod.
2. Aplica **Zero Trust**: deny-all + apertura mínima requerida.
3. Sincroniza CMDB etiquetas (vCenter Tags, ACI labels) para coherencia.
4. Automatiza pruebas de conectividad con scripts `curl`/`nmap` tras cambios.

Lecturas recomendadas

- Cisco — *ACI Best Practices for Micro-Segmentation.*
- VMware — *NSX-T Security Reference Design Guide.*
- NIST SP 800-207 — *Zero Trust Architecture.*

Respaldo y restauración de configuraciones IOS

Hacer copias de seguridad regulares del *running-config* y *startup-config* evita pérdida de servicios y acelera la recuperación tras fallos o cambios accidentales. Cisco IOS ofrece varios métodos: **TFTP, SCP, FTP, USB** y el comando `archive`.

---

1 Métodos de respaldo más comunes

Método	Ventajas	Consideraciones
TFTP	Simple, soportado por todos	Sin cifrado; recomendable solo en red de gestión
SCP	Cifrado SSH, autenticación	Más CPU; requiere imagen 12.3(4)T
FTP	Soporta credenciales	Contraseña en texto claro
USB	Portátil, sin depender de red	Hardware limitado a routers/switches con puerto USB

---

2 Copia manual del *running-config*

2.1 TFTP

```
copy running-config tftp:
 Address or name of remote host []? 10.0.0.50
 Destination filename [rtr-confg]? R1_$(show clock | include ^*).cfg
```

2.2 SCP

```
ip scp server enable
username backup secret StrongPass!
!
copy running-config scp://backup@10.0.0.60/R1.cfg
```

---

3 Automatizar con el comando *archive*

```
archive
 path scp://backup:StrongPass@10.0.0.60/configs/$h-$t.cfg
 write-memory
 time-period 1440          ! 24 h
```

Cada vez que se ejecute `write memory` *o transcurran 1 440 min, se subirá una copia al servidor SCP con nombre* `HOST-YYYYMMDD-HHMM.cfg`.

---

4 Restauración rápida

```
copy tftp: running-config
Address or name of remote host []? 10.0.0.50
Source filename []? R1_backup.cfg
Destination filename [running-config]? <Enter>
```

> **Tip**: para volver al *startup-config* original utiliza `configure replace tftp://10.0.0.50/R1_backup.cfg force`.

---

5 Verificación y mantenimiento

```
dir usbflash0:                          ! copias en USB
dir /all nvram:startup-config           ! checksum interno
show archive                            ! lista de versiones guardadas
```

---

6 Troubleshooting

Síntoma / Log	Causa probable	Comando útil
`Timed out` al copiar a TFTP	ACL o puerto 69 bloqueado	Revisar ACL / FW
`Permission denied` vía SCP	Usuario/clave errónea	`show run | inc username`
Archivo vacío en servidor	Falta espacio / cuota	Ver Syslog en servidor

---

7 Buenas prácticas

1. Mantener un **servidor de backups** en red de gestión aislada (VLAN mgmt).

2. Usar `archive` + **SCP** para cifrado en tránsito y control de versiones.

3. Habilitar **Syslog** al guardar config (`archive log config logging enable`).

4. Guardar la **licencia** (`show license udi`) junto con la config.

5. Probar restauración en laboratorio cada trimestre (+ *golden image*).

Lecturas recomendadas

- Cisco IOS — *Configuration Archive and Replace*.
- RFC 4254 — *The Secure Shell (SSH) Connection Protocol*.
- Cisco Press — *CCNP Enterprise Advanced Routing*, apéndice A.

NetFlow: captura y análisis de flujos

NetFlow (y su evolución IPFIX) exporta metadatos de tráfico nivel 3/4 — origen, destino, puertos, bytes— para visibilidad, facturación, detección de anomalías y planificación de capacidad.

---

1 Versiones populares

Versión	Campos clave	Compatibilidad
5	IP v4 + UDP/TCP bytes	Casi todo software
9	Plantillas flexibles	IPv4/6 + MPLS, QoS
IPFIX	Estándar IETF (RFC 7011)	Extensible (custom)

---

2 Componentes

- **Exporter** – router/switch que genera registros.
- **Collector** – servidor que almacena y analiza (ElastiFlow, NTA, nfdump).
- **Analyzer** – panel gráfico: Grafana, Kibana, SolarWinds, PRTG.

---

3 Configuración básica (NetFlow v9, IOS/IOS XE)

3.1 Definir el *exporter*

```
flow exporter NMS
 destination 10.0.0.50
 transport udp 2055
 source Loopback0
 export-protocol netflow-v9
 template data timeout 60
```

3.2 Crear *flow record*

```
flow record L3L4-RECORD
 match ipv4 source address
 match ipv4 destination address
 match transport source-port
 match transport destination-port
 collect counter bytes
```

```
 collect counter packets
 collect timestamp sys-uptime first
 collect timestamp sys-uptime last
```

3.3 Asociar a *monitor* e interfaz

```
flow monitor CAMPUS_MON
 record L3L4-RECORD
 exporter NMS

interface Gig0/1
 ip flow monitor CAMPUS_MON input
 ip flow monitor CAMPUS_MON output
```

---

4 Verificación rápida

```
Rtr# show flow exporter NMS
Rtr# show flow monitor CAMPUS_MON cache
Rtr# show platform hardware qfp active feature flow              ! ISR4k
```

Campos clave: **Records exported**, **Flows**, **Template ID**.

---

5 Troubleshooting

Síntoma / Log	Causa probable	Comando útil
Collector no recibe datos	ACL bloquea UDP 2055	Revisión FW / `ping`
`Template not received` en NMS	Exporter sin plantillas	`show flow exporter`
Contadores 0 en cache	Monitor no aplicado	`show run interface`
CPU alto (>10 %) al habilitar	Demasiadas claves match	Simplificar record

---

6 Buenas prácticas

1. Exportar a **loopback** como origen para resiliencia (`source Lo0`).
2. Usar **IPFIX** para data center (VXLAN, L4-L7) y compatibilidad futura.
3. Limitar puerto/ACL solo a colectores autorizados.
4. Mantener `template timeout` 60 s para evitar “template lost”.

5. Archivar flujos crudos 30 d + índices agregados 12 m (balance coste/valor).

Lecturas recomendadas

- Cisco IOS — *NetFlow Configuration Guide*.
- RFC 7011 — *Specification of the IPFIX Protocol*.
- ElastiFlow — *Architecture & Sizing Guide*.

AAA con RADIUS y TACACS+

El framework **AAA** (Authentication, Authorization, Accounting) centraliza credenciales y logs de acceso en servidores externos, simplificando la gestión y reforzando la seguridad en infraestructuras de red.

---

1 RADIUS vs TACACS+ (resumen)

Parámetro	**RADIUS**	**TACACS+**
RFC / Propietario	2865, 2866 / IETF	Cisco propietario (TCP 49)
Transporte	UDP 1812/1813	TCP 49
Cifrado	Solo password + atributos	Cuerpo completo del paquete
AAA Separado	No (mezcla Auth + Authz)	Sí (Auth, Authz, Acct)
Uso típico	Wi-Fi, VPN, 802.1X	Acceso CLI / dispositivos

---

2 Configuración paso a paso (RADIUS)

2.1 Definir el servidor

```
radius server ISE1
 address ipv4 10.0.0.50 auth-port 1812 acct-port 1813
 key SuperSekret!
 timeout 5
 retransmit 2
```

2.2 Habilitar AAA nueva modalidad

```
aaa new-model
aaa group server radius ISE-GRP
 server name ISE1

aaa authentication login REMOTE_LOGIN group ISE-GRP local
aaa authorization exec  REMOTE_AUTH  group ISE-GRP local
aaa accounting exec      ACCT_START  start-stop group ISE-GRP
```

2.3 Aplicar métodos a VTY y consola

```
line vty 0 4
 login authentication REMOTE_LOGIN
 authorization exec REMOTE_AUTH
 accounting exec ACCT_START
```

```
line con 0
 login local
```

---

3 Configuración básica (TACACS+)

```
tacacs server ACS1
 address ipv4 10.0.0.60
 key 0 Str0ngKey

aaa group server tacacs+ TAC-GRP
 server name ACS1

aaa authentication login REMOTE_LOGIN group TAC-GRP local
aaa authorization exec  REMOTE_AUTH  group TAC-GRP local
aaa accounting commands 15 ACCT_CMDS start-stop group TAC-GRP
```

---

4 Verificación rápida

```
Router# test aaa group radius ISE-GRP username admin password ****
Router# show aaa servers
Router# show accounting
```

Campos clave: **Reachable**, **AuthZ OK**, **Acct pkts sent/recv**.

---

5 Troubleshooting

Síntoma / Log	Causa probable	Comando útil
`%AUTHMGR-5-AAA_FAIL`	Clave o puerto incorrecto	`debug radius` / `debug tacacs`
Delay >5 s en login	Timeout alto / DNS	`show aaa servers`
Usuarios sin privilegios exec	Falta atributo *shell*	Revisar perfil en ACS/ISE

---

6 Buenas prácticas

1. Definir ruta de **backup local** (`local`) en método AAA.

2. Usar **NTP** preciso en clientes y servidores para integridad de registros.
3. Activar **command accounting** (nivel 15) para auditoría completa.
4. Segmentar tráfico AAA en **VRF o VLAN de gestión** con ACL.
5. Rotar claves compartidas y usar **IPv6** si el servidor lo soporta.

Lecturas recomendadas

- Cisco IOS — *AAA and RADIUS/TACACS+ Configuration Guide*.
- RFC 2865 / 2866 — *Remote Authentication Dial-In User Service*.
- Cisco Press — *AAA Identity Management Security*, cap. 4-6.

Buenas prácticas de nomenclatura en red

Un sistema de nombres consistente acelera el *troubleshooting*, facilita la automatización y reduce errores humanos al documentar o generar configuraciones de forma dinámica.

---

1 Objetivos de un buen esquema

- **Único**: cada elemento (dispositivo, VLAN, interfaz) se identifica sin ambigüedad.
- **Legible**: alguien nuevo puede entender la función con un vistazo.
- **Escalable**: admite crecimiento sin renombrar en masa.
- **Automatizable**: fácil de tokenizar para scripts y CMDB.

---

2 Componentes típicos en nombres de host

Componente	Ejemplo	Descripción
Sitio	MAD	Código IATA/ciudad (Madrid)
Función	DIST	CORE / DIST / ACC / FW / SRV
Número	01	Secuencial o par/ impar por chasis
Sufijo	-A	Stack, TOR-A/B, POD-1/2 (opcional)

Patrón resultante: `<SITE>-<FUNC>-<NN><SUF>` → `MAD-DIST-01-A`

---

3 Ejemplo de tabla de prefijos

Objeto	Prefijo	Ejemplo
VLAN datos	V-D-	**V-D-010**
VLAN voz	V-V-	V-V-025
VRF	VRF-	VRF-CORPORATE
ACL	ACL-	ACL-DENY-ICMP
SSID Wi-Fi	SSID-	SSID-CORP-5G

---

4 Convenciones de interfaz (Cisco)

```
Gi0/0/1     UPLINK-DIST01
Gi0/0/2     ACC-SRV-01
Po10        PC-CORE-TRUNK
Lo0         RID_OSPF
```

Incluye el propósito en `description`; evita abreviaturas crípticas.

---

5 Buenas prácticas adicionales

1. Usar **ceros a la izquierda** para orden alfanumérico (`VLAN010`).
2. Reservar rangos de números por propósito (10–99 datos, 100–199 voz).
3. Mantener un **glosario central** (Markdown/Wiki) con todos los prefijos.
4. Validar nombres vía *lint* en CI/CD antes de aplicar cambios.
5. Evitar caracteres especiales; apégate a **A-Z, 0-9, guion**.

Lecturas recomendadas

- Cisco – *Network Naming Conventions Design Guide.*
- Google SRE – *Production Best Practices: Naming Things.*
- ITIL – *Configuration Management Database (CMDB) Guidelines.*

Planificación de direccionamiento IPv4 público y privado

Diseñar un plan IP ordenado evita solapamientos, simplifica ACL, facilita la summarización y agiliza el crecimiento futuro.
Combinar rangos **privados** (RFC 1918) con bloques **públicos** requiere una estrategia clara y documentación detallada.

---

1 Rangos privados estándar

Rango RFC 1918	Tamaño (/bits)	Hosts	Uso típico
10.0.0.0/8	/8 – /30	16 777 214	Corporativos grandes
172.16.0.0/12	/12 – /30	1 048 574	Campus medianos
192.168.0.0/16	/16 – /30	65 534	Sucursales / hogar

> RFC 6598 define 100.64.0.0/10 para **CGNAT**; evita usarlo internamente salvo que implementes NAT444.

---

2 Pasos metodológicos

1. **Inventario de sitios**: listar HQ, data centers, oficinas, DMZ.
2. **Clasificar roles**: usuarios, servidores, infra, gestión, IoT.
3. **Asignar bloques** jerárquicos (ej. /16 por país, /24 por VLAN).
4. Reservar subredes para **loopbacks**, **túneles** y **point-to-point** (/30 o /31).
5. Definir políticas NAT bloques públicos (PAT, DNAT, static NAT).

---

3 Ejemplo de esquema jerárquico (empresa multinacional)

Sitio	Bloque /16	Descripción
MAD	10.10.0.0/16	HQ Madrid
NYC	10.20.0.0/16	HQ Nueva York
BUE	10.30.0.0/16	Oficina Buenos Aires

Cada /16 se divide en /24 por VLAN:

```
10.10.10.0/24  → Usuarios planta 1
10.10.20.0/24  → VoIP
10.10.30.0/24  → Servidores
10.10.254.0/24 → Gestión (OOB)
```

---

4 Asignación de bloques públicos

```
198.51.100.0/24 ← Web, VPN, MX
203.0.113.0/27  ← BGP loopbacks
```

Usa NAT 1:1 para servidores DMZ; PAT (overload) para clientes internos.

---

5 Summarización y anuncios

- OSPF interno: `area 0 range 10.10.0.0 255.255.0.0`.
- BGP hacia ISP: anunciar solo `198.51.100.0/24`.
- Filtrar RFC 1918 en rutas de salida (`ip prefix-list OUT deny 10.0.0.0/8 le 32`).

---

6 Buenas prácticas

1. Documentar rangos en **IPAM** (Gestor IP) con atributos (VLAN, VRF, ACL).
2. Evitar superposición con partners: usa bloque ULA único (hash SHA-1).
3. Reservar **/24** contiguo para futuros servicios y pruebas.
4. Etiquetar subredes críticas con `description` en interfaces y VRF.
5. Revisar el plan al menos cada 12 meses; crecimiento >70 % = expandir.

Lecturas recomendadas

- RFC 1918 — *Address Allocation for Private Internets.*

- RIPE-690 — *IPv4 Addressing Plan Best Current Practice.*
- Cisco Press — *Enterprise IPv4/V6 Addressing Design Guide*, cap. 2-3.

Redistribución OSPF EIGRP

La **redistribución** permite intercambiar rutas entre procesos de enrutamiento distintos (OSPF, EIGRP, BGP…), imprescindible en fusiones, migraciones o entornos multi-vendor. Mal aplicada puede crear bucles, inundar tablas y provocar rutas sub-óptimas.

---

1 Riesgos habituales

- **Bucles** – rutas devueltas al protocolo de origen.
- **Inflación de LSDB** – exceso de prefijos externos (*E2*).
- **Métricas incoherentes** – tráfico toma trayectos no deseados.

---

2 Métricas por defecto

Protocolo origen	Destino	Métrica asignada por defecto
EIGRP → OSPF	20 (E2)	Cost = 20
OSPF → EIGRP	BW 100000 Kb	BW, Delay, Reliab, etc. = 0

Ajusta métricas manualmente para evitar caminos de baja capacidad.

---

3 Configuración básica (IOS)

3.1 Redistribuir OSPF dentro de EIGRP

```
router eigrp 100
 redistribute ospf 1 metric 100000 10 255 1 1500 \
  route-map OSPF-IN
```

3.2 Redistribuir EIGRP dentro de OSPF

```
router ospf 1
 redistribute eigrp 100 subnets \
  metric 100 type 1 route-map EIGRP-IN
```

---

4 Control de rutas con *route-map*

```
route-map OSPF-IN permit 10
 match ip address prefix-list HQ_ONLY
 set metric 50

ip prefix-list HQ_ONLY seq 5 permit 10.10.0.0/16 le 24
```

Solo redes /24 del HQ se inyectan con coste 50.

---

5 Verificación esencial

```
Router# show ip route eigrp
Router# show ip ospf database external | inc 0.0.0.0
Router# show route-map
```

Campos clave: **External Type 1/2**, **FD/AD** en EIGRP, **Age** en LSAs.

---

6 Troubleshooting

Síntoma	Causa probable	Comando útil
Rutas externas faltantes	Falta `subnets`	`show run | sec redis`
AD preferida al protocolo eq.	AD no ajustada	`distance eigrp ...`
Uso de enlace lento inesperado	Métrica default 20	Ajustar `metric 10 type 1`

---

7 Buenas prácticas

1. **Filtrar** prefijos con *prefix-lists* antes de redistribuir.
2. Usar **métricas coherentes**; en OSPF, *Type 1* considera costo interno.
3. **Evitar doble redistribución** (EIGRP→OSPF→EIGRP) salvo con tags.
4. Etiquetar rutas (`set tag 100`) y bloquear feedback (`match tag 100 deny`).
5. Documentar reglas en diagramas y CMDB; probar en sandbox primero.

Lecturas recomendadas

- Cisco IOS — *Route Redistribution Configuration Guide*.
- Doyle, J. — *Routing TCP/IP Vol.1*, cap. 11.
- RFC 5838 — *OSPFv2 Multiple Instances* (multi-IGP coexistencia).

Diseño de DMZ y segmentación de servidores

Una **DMZ (Demilitarized Zone)** aloja servicios expuestos a Internet (web, correo, VPN) separándolos de la LAN interna para mitigar riesgos de compromiso. La segmentación añade capas (zonas) con reglas de firewall estrictas que limitan el movimiento lateral.

---

1 Zonas típicas en un diseño de tres niveles

Zona	Tráfico permitido	Ejemplos de equipos
Externa	Internet público → DMZ	Routers ISP, DDoS scrubs
DMZ	Externa DMZ, DMZ Core	Reverse proxy, WAF, MX
Core / LAN	DMZ Core, Core interno	Servidores internos, DB

> *Regla general*: "Menos es más" → solo los puertos y direcciones imprescindibles se abren entre zonas.

---

2 Asignación de VLAN y subred

```
VLAN 100  192.0.2.0/28     → DMZ pública
VLAN 110  10.10.100.0/24   → Servidores App (segmento interno)
VLAN 120  10.10.200.0/24   → Base de datos
```

Separar App y DB evita que un atacante desde la DMZ llegue directo a la DB.

---

3 Políticas de firewall de ejemplo

Fuente	Destino	Puerto/Proto	Acción	Motivo
Internet	DMZ-WEB	443/TCP	Permit	HTTPS público
DMZ-WEB	APP-SRV	8080/TCP	Permit	Tráfico API interno
APP-SRV	DB-SRV	3306/TCP	Permit	MySQL
Cualquier	Cualquier	*	Deny	Política por defecto

---

4 Hardening adicional

- **IPS/WAF** delante de servidores web para filtrar ataques L7.
- **Reverse Proxy** (NGINX/Envoy) termina TLS y oculta direcciones reales.
- **Port Security / DHCP Snooping** en switch DMZ para evitar MAC spoof.
- **Logs centralizados** (Syslog, SIEM) de FW y servidores DMZ.

---

5 DMZ en nube híbrida

Implementa la misma lógica:
- **Subnet pública** en VPC/VNet para Load Balancer/ALB.
- **Subnet privada DMZ** para instancias front-end con SG restringidos.
- **Transit Gateway / ExpressRoute** hacia el data center core.

---

6 Troubleshooting rápido

Síntoma	Causa probable	Comando / Herramienta
Web accesible pero sin backend	FW bloquea 8080	`packet-tracer` ASA
Conexión DB intermitente	Time-wait / NAT ports	Aumentar `xlate`
Escaneo detecta puertos extra	Servicio no deshabilitado	`nmap`, `systemctl`

---

7 Buenas prácticas

1. **Modelo Zero Trust**: nunca asumas que la DMZ es "segura".
2. Usar **NAT estático** 1:1, no PAT, para facilitar trazabilidad.
3. Actualizar parches OS/aplicación en DMZ con *window* dedicado.
4. Revisar reglas cada 6 meses; eliminar puertos/protocolos obsoletos.
5. Documentar flujos en diagrama (Visio, draw.io) + CMDB.

Lecturas recomendadas

- Cisco — *Secure Data Center Design Guide* (DMZ chapter).
- NIST SP 800-41 rev.1 — *Guidelines on Firewalls and Firewall Policy.*
- OWASP — *Secure Deployment Cheat Sheet.*

Alta disponibilidad con VRRP

VRRP (Virtual Router Redundancy Protocol, RFC 5798) ofrece un gateway virtual compartido por varios routers. Es similar a HSRP, pero estándar y soportado por múltiples fabricantes.

---

1 Componentes y estados

Elemento	Descripción
Virtual IP	Dirección que usan los hosts (gateway)
Master	Router que responde ARP, reenvía tráfico
Backup	Toma rol si el Master falla

Estado VRRP	Significado
INIT	Grupo recién habilitado
MASTER	En control de la IP virtual
BACKUP	Esperando hellos

---

2 Temporizadores y prioridad

Parámetro	Valor por defecto	Ajuste típico
Advertisement	1 s (v3)	1 s (Fast)
Skew time	(256-prio)/256	Automático
Priority	100	101–254 Master preferido

El router con **mayor prioridad** gana; en empate, la IP más alta.

---

3 Configuración mínima en Cisco IOS/IOS XE

```
interface Gig0/0
 ip address 192.0.2.2 255.255.255.0
 vrrp 10 ip 192.0.2.1
 vrrp 10 priority 110
 vrrp 10 preempt         ! Recupera control al volver
 vrrp 10 authentication text S3cr3t
```

En el router de respaldo:

```
interface Gig0/0
 ip address 192.0.2.3 255.255.255.0
 vrrp 10 ip 192.0.2.1
 vrrp 10 priority 90
 vrrp 10 preempt
 vrrp 10 authentication text S3cr3t
```

---

4 Verificación rápida

```
R1# show vrrp
R2# show vrrp brief
```

Campos clave: **State**, **Master Router**, **Master Down timer**, **Adv Interval**.

---

5 Troubleshooting

Síntoma / Log	Causa probable	Comando útil
Ambos routers en MASTER	Auth o prioridad dif.	`show vrrp`
Estado INIT permanente	Hellos bloqueados (ACL)	Revisar ACL/L2
Cambio frecuente de Master	Flapping interfaz LAN	`show log`, cables

---

6 Buenas prácticas

1. Igualar **authentication** en todos los miembros para evitar split-brain.
2. Distribuir carga con **VRRP load-sharing** (varios grupos, distintas prioridades).
3. Supervisar interfaces críticas (`track Gig0/1`) y reducir prioridad al fallar.
4. Sincronizar NTP; facilita análisis de logs de cambio de estado.

Lecturas recomendadas

- RFC 5798 — *Virtual Router Redundancy Protocol (VRRP) Version 3.*

- Cisco IOS — *VRRP Configuration Guide.*
- Juniper — *Understanding VRRP for IPv4.*

Túneles GRE punto-a-punto

GRE (Generic Routing Encapsulation, RFC 2784/2890) permite crear un túnel L3 sobre una red IP cualquiera, transportando protocolos IPv4, IPv6 o incluso tramas multicast que no podrían cruzar Internet nativa.

---

1 Casos de uso habituales

- Conectar sitios que usan **ruteo dinámico** (OSPF, EIGRP) a través de Internet.
- Encapsular **multicast** o protocolos no enrutable (iSCSI, OSPF NBMA).
- Soporte para **VPN "hub-and-spoke"** simple cuando IPsec solo cifra.

---

2 Vista rápida del encabezado GRE

Campo	Bits	Descripción
Flags	2	C (Checksum) · K (Key) opcionales
Protocol	16	Tipo de paquete interno (0x0800 = IPv4)
Key	32	Identificador de túnel (opcional)

Para evitar colisiones se recomienda habilitar el **Key** (RFC 2890).

---

3 Configuración IOS clásica

```
interface Tunnel0
 ip address 172.16.0.1 255.255.255.252
 tunnel source Gig0/0                ! 198.51.100.2
 tunnel destination 203.0.113.2
 tunnel key 1234
 keepalive 5 3                       ! opcional
```

En el router remoto:

```
interface Tunnel0
 ip address 172.16.0.2 255.255.255.252
 tunnel source Gig0/0                ! 203.0.113.2
 tunnel destination 198.51.100.2
 tunnel key 1234
```

3.1 Añadir IPsec para cifrado

```
crypto map GRE-IPSEC 10 ipsec-isakmp
 set peer 203.0.113.2
 set transform-set AES256-SHA
 match address 110

access-list 110 permit gre host 198.51.100.2 host 203.0.113.2

interface Gig0/0
 crypto map GRE-IPSEC
```

---

4 Verificación rápida

```
R1# show interface Tunnel0
R1# show ip route | inc 172.16
R1# ping 172.16.0.2 source 172.16.0.1
R1# show crypto ipsec sa           ! si se cifra
```

Campos clave: **Tunnel0 up/up**, **TX/RX encaps/decaps** creciendo.

---

5 Troubleshooting

Síntoma / Log	Causa probable	Comando útil
Tunnel down (line protocol)	Ping al destino falla	`traceroute` WAN
GRE encaps OK pero sin cifrado	Crypto map no aplica	`show crypto map`
MTU issues / fragmentación	Falta `ip tcp adjust-mss`	Ajustar MSS 1360

---

6 Buenas prácticas

1. Usar **keepalive** para derribar rutas si el túnel cae.
2. Habilitar **tunnel key** único por par para facilitar filtros ACL.
3. Ajustar **MSS/PMTUD**: `ip tcp adjust-mss 1360` en interfaces LAN.
4. Documentar túneles en IPAM y diagrama — origen, destino, ID, Key.
5. Migrar a **DMVPN/GETVPN** en topologías malla o >50 sitios.

Lecturas recomendadas

- Cisco IOS — *GRE Tunneling Configuration Guide.*
- RFC 2784 — *Generic Routing Encapsulation (GRE).*
- RFC 2890 — *Key and Sequence Number Extensions to GRE.*

Firewall en borde: reglas básicas

Un **firewall de perímetro** es la primera línea de defensa entre Internet y la red interna. Una política mínima—bien documentada y auditada—reduce la superficie de ataque y facilita el *troubleshooting*.

---

1 Modelo de reglas por capas (outside → DMZ → inside)

Prioridad	Acción	Tráfico	Motivo
1	Permit	Established / Related	Respuestas a conexiones salientes
2	Permit	HTTPS, 443/TCP → DMZ-WEB	Publicar sitio web
3	Permit	SMTP, 25/TCP → DMZ-MX	Entrada correo
4	Deny	RFC 1918 from outside	Ocultar redes privadas
5	Deny	IP spoof (0.0.0.0/8, 127.0.0.0/8)	Protección básica
6	Permit	OUTBOUND any → Internet (NATed)	Navegación usuarios
99	Deny	Any Any	Política por defecto

> **Regla 1** se aplica con "stateful inspection" (`established` en ASA, `ctstate RELATED,ESTABLISHED` en iptables).

---

2 Ejemplo de ACL en Cisco ASA (object groups)

```
object network OBJ_WEB
 host 192.0.2.10
object network OBJ_MX
 host 192.0.2.20

access-list OUTSIDE_IN extended permit tcp any object OBJ_WEB eq 443
access-list OUTSIDE_IN extended permit tcp any object OBJ_MX  eq smtp
access-list OUTSIDE_IN extended deny   ip any 10.0.0.0 255.0.0.0
access-list OUTSIDE_IN extended deny   ip any 192.168.0.0 255.255.0.0
access-list OUTSIDE_IN extended deny   ip any 172.16.0.0 255.240.0.0
access-list OUTSIDE_IN extended deny   ip any host 127.0.0.0
access-list OUTSIDE_IN extended permit ip any any established
access-group OUTSIDE_IN in interface outside
```

---

3 Buenas prácticas operativas

- **Menos es más**: habilita solo puertos necesarios; revisa cada 6 meses.

- **Object-groups** y **service-groups** evitan reglas duplicadas.
- Activa **logging** de descartes críticos (`informational`, Syslog 106023).
- Usa **descripciones** en reglas (`remark`) para justificar la apertura.
- Implementa **MFA** en acceso de administración y segmenta Mgmt-VRF.

---

4 Verificación y monitoreo

```
ASA# show access-list OUTSIDE_IN
ASA# show conn detail | include 192.0.2.10
ASA# show service-policy flow
```

Métricas clave: *hitcnt*, *bytes*, *drops*, *age* por entrada ACL.

---

5 Troubleshooting rápido

Síntoma / Log	Causa probable	Herramienta
`%ASA-4-106023 Deny tcp…`	Puerto no permitido	Revisar ACL / packet-tracer
Conex. sale pero no vuelve	NAT/established faltante	`show xlate`, `packet-capture`
Alta CPU en ASA	Syslog nivel “debug” alto	Bajar logging o filtrar

---

Lecturas recomendadas

- Cisco ASA — *General Operations CLI Configuration Guide.*
- NIST SP 800-41 rev.1 — *Guidelines on Firewalls and Firewall Policy.*
- SANS Institute — *Firewall Rule-set Review Checklist.*

Summarización de rutas y optimización

La **summarización** (o agregación) combina prefijos contiguos en uno más amplio, reduciendo el tamaño de las tablas de enrutamiento, el tráfico de actualizaciones y el uso de CPU en los routers.

---

1 Ventajas principales

- Menos entradas → Cálculos SPF/EIGRP más rápidos.
- Oculta inestabilidad de subredes detrás de un resumen estable.
- Disminuye el ancho de banda consumido por LSAs/DUAL.

---

2 Requisitos para un buen resumen

1. Todos los prefijos deben ser **contiguos**.
2. La máscara del resumen debe cubrir exactamente ese rango; no incluir subredes de terceros.
3. Debe haber **puntos de discontigüidad** mínimos para evitar blackholing.

---

3 Ejemplo rápido: sumarizar /28 contiguos

Prefijos originales	Binario (último octeto)
192.168.1.0/28	0000 0000
192.168.1.16/28	0001 0000
192.168.1.32/28	0010 0000
192.168.1.48/28	0011 0000

» Los cuatro comparten 26 bits → resumen **192.168.1.0/26**.

---

4 Configuración en protocolos comunes

4.1 OSPF (ABR)

```
router ospf 1
```

```
 area 10 range 10.20.0.0 255.255.252.0   ! /22
```

4.2 EIGRP (manual)

```
interface Gig0/0
 ip summary-address eigrp 100 172.16.32.0 255.255.248.0
```

4.3 BGP (agregate-address)

```
router bgp 65010
 aggregate-address 198.51.100.0 255.255.254.0 summary-only
```

---

5 Verificación esencial

```
Rtr# show ip route 10.20.0.0
Rtr# show ip ospf database summary | include 10.20.0.0
Rtr# show ip eigrp topology | inc 172.16.32.0
```

Campos clave: `O IA`, `D EX` y el marcador `*>i` para agregados BGP.

---

6 Troubleshooting

Síntoma	Causa probable	Comando útil
Tráfico se pierde tras resumen	Super-resumen incluye redes ajenas	Revisar máscara / regla
Rutas específicas siguen apareciendo	Falta *summary-only* (BGP)	`show ip bgp`
OSPF envía many LSAs	Resumen no en ABR correcto	Ver topología / área

---

7 Buenas prácticas

1. Implementar en bordes (ABR/ASBR) y no en todos los routers.
2. Mantener bloques de direccionamiento **jerárquicos** desde el diseño inicial.
3. Documentar cada rango resumido en la CMDB/IPAM.
4. Evitar sumarizar rutas de loopbacks utilizadas como Router-ID.

5. Probar en laboratorio la convergencia y posibles black-holes.

Lecturas recomendadas

- Cisco IOS — *Route Summarization Configuration Guide*.
- Doyle, J. — *Routing TCP/IP Vol.1*, cap. 12.
- RFC 4632 — *Classless Inter-Domain Routing (CIDR)*.

Guía rápida de troubleshooting (ping & traceroute)

El dúo **ping / traceroute** es la primera herramienta para aislar problemas de conectividad. Esta ficha explica cómo interpretarlos y qué pasos seguir antes de escalar el incidente.

---

1 Flujo de decisión básico

Paso	Prueba	Acción si falla
1	`ping 127.0.0.1`	Revisar pila IP local
2	`ping <Gateway>`	Verificar NIC, VLAN, ARP
3	`ping 8.8.8.8`	Comprobar ruta, NAT, ACL
4	`ping www.google.com`	Resolver DNS

---

2 Sintaxis IOS útil

```
ping 10.10.10.1 repeat 5 timeout 2
ping 2001:db8::1 source Loopback0
traceroute 8.8.8.8 numeric
```

Parámetros comunes:
* `repeat <n>` → paquetes enviados (por defecto 5).
* `timeout <s>` → esperar X segundos por respuesta.
* `source <IF/IP>` → fuerza interfaz/IP de salida.

---

3 Interpretación rápida del *traceroute*

Salida típica	Significado
`* * *`	Timeout; router no responde a TTL.
Saltos >30	Posible loop o ruta larga (MPLS).
Picos de RTT	Congestión o shaping en ese salto.

> *Tip*: en redes MPLS, algunos routers esconden saltos (penultimate hop).

---

4 Ejemplo de secuencia de comandos (Bash)

```
#!/usr/bin/env bash
TARGET=$1
echo "== PING $TARGET =="
ping -c 5 -W 2 $TARGET
echo
echo "== TRACEROUTE $TARGET =="
traceroute -n $TARGET
```

Guárdalo como `netdiag.sh`; ejecútalo con `./netdiag.sh 8.8.8.8`.

---

5 Troubleshooting típico

Síntoma	Causa probable	Siguiente paso
`Destination host unreachable`	Sin ruta en router intermedio	`show ip route`
1%+ packet loss en último salto	Congestión/buffer drop	QoS, enlace saturado
Primer salto muestra * * *	ACL ICMP en gateway	Revisar reglas FW

---

6 Buenas prácticas

1. Registrar resultados con fecha (*snippet* en ticket).
2. Usar `ping sweep` (`for i in {1..254}`) para detectar host vivos.
3. Ajustar DSCP/ToS en ping para probar QoS (`ping tos 184`).
4. En IPv6, recordar `%IF` para link-local (`ping fe80::1%Gig0/0`).

Lecturas recomendadas

- Cisco IOS — *Ping and Traceroute Command Reference*.
- RFC 792 — *Internet Control Message Protocol (ICMP)*.
- PacketLife.net — *ICMP Cheat Sheet*.

Syslog y niveles de logging en Cisco IOS

Syslog centraliza los mensajes de eventos generados por routers, switches y firewalls, facilitando auditoría, alertas y análisis forense. Un buen esquema de severidad y filtrado evita inundar el colector con ruido.

---

1 Niveles de severidad

Nº	Severidad	Descripción	Ejemplo de mensaje
0	**EMERGENCY**	Sistema no usable	Kernel panic
1	ALERT	Acción inmediata requerida	PSU FAIL
2	CRITICAL	Condición crítica	Fan speed low
3	ERROR	Función no disponible	OSPF adjacency down
4	WARNING	Condición no crítica	High CPU 85 %
5	NOTICE	Normal pero importante	Line protocol up
6	INFO	Información general	DHCP lease granted
7	DEBUG	Mensajes detallados de depuración	OSPF LSA floods

> **Regla práctica**: almacenar de 0 a 5 en el SIEM; 6–7 solo en sesión de debug o colector secundario con rotación frecuente.

---

2 Configuración básica de Syslog

```
logging buffered 100000           ! buffer local (bytes)
logging trap warnings             ! severidad mínima que se envía
logging host 10.0.0.50 transport udp port 514
service timestamps log datetime msec localtime
service sequence-numbers
```

Parámetros clave:
* `trap` – severidad hacia el servidor.
* `buffered` – tamaño del log en RAM (consulta con `show logging`).

---

3 Ejemplo de filtros avanzados (IOS XE)

```
logging discriminator OSPF msg-body includes OSPF
logging host 10.0.0.51 discriminator OSPF
```

Solo eventos que contengan “OSPF” se envían al colector 10.0.0.51.

---

4 Comandos de verificación

```
Rtr# show logging
Rtr# show logging history
Rtr# show logging | include 10.0.0.50
```

Campos a revisar: **Log Buffer**, **Logging to host**, **filtered by**.

---

5 Troubleshooting rápido

Síntoma / Log	Causa probable	Acción sugerida
Syslog server sin logs	ACL o UDP 514 bloqueado	Revisar FW, `ping`
Buffer local se llena rápido	Nivel trap demasiado bajo	Aumentar severidad
Eventos repetidos (log loop)	Config flapping	`logging rate-limit`

---

6 Buenas prácticas

1. Sincronizar **NTP** en todos los nodos para sellos de tiempo coherentes.
2. Habilitar `service timestamps msec` para correlación precisa.
3. Usar **TLS / TCP 6514** si el colector lo soporta (`logging host ... transport tcp port 6514`).
4. Separar logs de **seguridad** y **operación** en colectores distintos.
5. Rotar y comprimir logs del SIEM; conservar 6–12 meses según políticas.

Lecturas recomendadas

- Cisco IOS — *Syslog Messages and Severity Levels.*
- RFC 5424 — *The Syslog Protocol.*
- NIST SP 800-92 — *Guide to Computer Security Log Management.*

Supervisión con SNMP v2c y v3 en Cisco IOS

El **Simple Network Management Protocol (SNMP)** permite recopilar métricas (CPU, memoria, interfaces), enviar *traps* y realizar cambios remotos (read-write) en los equipos de red. Usar **v3** aporta autenticación y cifrado; v2c se mantiene para compatibilidad con herramientas antiguas.

---

1 Comparativa de versiones

Versión	Seguridad	Transporte	Uso típico
v1	Comunidad (texto)	UDP 161/162	Obsoleto, solo lab/historic
v2c	Comunidad (texto)	UDP 161/162	Herramientas legacy
v3	USM (auth, priv)	UDP/TCP	Producción, compliance PCI/GDPR

---

2 Configuración paso a paso

2.1 SNMP v2c (solo lectura)

```
snmp-server community PUBLIC_RO RO 99
snmp-server host 10.0.0.50 version 2c PUBLIC_RO
!
access-list 99 permit 10.0.0.50
```

2.2 SNMP v3 (auth + privacidad AES)

```
snmp-server group NMS-GRP v3 priv read SYSTEM-RO write SYSTEM-RW
snmp-server user nmsadmin NMS-GRP v3 auth sha StrOngAuth! priv aes 256 StrOngPriv!

snmp-server host 10.0.0.60 version 3 priv nmsadmin
snmp-server enable traps snmp linkdown linkup coldstart warmstart
```

---

3 Vistas y MIBs recomendadas

```
snmp-server view SYSTEM-RO iso included
snmp-server view SYSTEM-RO iso.org.dod.internet.private excluded

! Limita RW a solo ifTable y ipRoute
snmp-server view SYSTEM-RW iso.org.dod.internet.mgmt.mib-2.if included
snmp-server view SYSTEM-RW iso.org.dod.internet.mgmt.mib-2.ip.ipRouteTable included
```

Tip: exporta solo las OID necesarias para cumplir **principio de mínimo privilegio**.

---

4 Verificación rápida

```
Router# show snmp user
Router# show snmp group
Router# show snmp engineID
Router# show snmp statistics
```

En el NMS:

```
snmpwalk -v3 -u nmsadmin -l authPriv -a sha -A StrOngAuth! \
 -x aes -X StrOngPriv! 198.51.100.1 1.3.6.1.2.1.1.5.0
```

---

5 Troubleshooting rápido

Síntoma / Log	Causa probable	Acción sugerida
`snmp authentication failure`	Comunidad/USM errónea	Revisar credenciales
Timeout al hacer *snmpwalk*	ACL o UDP 161 bloqueado	Comprobar FW / `ping`
Traps no llegan al NMS	Host mal definido	`show snmp host`
Contadores vacíos (0)	Vista restringe OID	`show snmp view`

---

6 Buenas prácticas

1. Usar **SNMP v3** con AES-256 y SHA-256 siempre que el NMS lo soporte.
2. Limitar acceso por **ACL** a servidores de monitorización.
3. Cambiar comunidades *public/private* por strings aleatorias en v2c.
4. Habilitar **linkUp/linkDown** y **coldStart/warmStart** traps para alertas básicas.
5. Integrar **NetFlow/IPFIX** o **gRPC** para métricas de rendimiento detalladas.

Lecturas recomendadas

- Cisco IOS — *SNMP Version 3 Configuration Guide.*
- RFC 3411-3418 — *SNMP Framework (v3).*
- NIST SP 800-115 — *Technical Guide to Information Security Testing and Assessment* (cap. SNMP).

Eficiencia energética en redes (EEE, PoE y EnergyWise)

Reducir el consumo eléctrico de switches, routers y puntos de acceso baja el OPEX y contribuye a los objetivos de sostenibilidad de TI. Las funciones clave son **Energy Efficient Ethernet (EEE)**, la gestión de **PoE** y la plataforma **Cisco EnergyWise**.

---

1 Energy Efficient Ethernet (IEEE 802.3az)

- **Modo LPI (Low-Power Idle)**: el enlace pasa a reposo cuando no hay tráfico.
- Ahorro típico: 30-40 % en puertos de acceso 1 Gb/s con baja utilización.
- Compatible con cobre (1000BASE-T) y algunos módulos SFP/SFP+.

1.1 Habilitar EEE en IOS XE

```
interface GigabitEthernet1/0/5
 power efficient-ethernet auto
```

Verificación:

```
SW# show eee interface gi1/0/5
```

---

2 Gestión de PoE (IEEE 802.3af/at/bt)

Comando típico	Función
`power inline static 15400`	Limita puerto a 15,4 W (af)
`power inline auto`	Negociación automática
`power inline police`	Deshabilita puerto si supera presupuesto
`power inline consumption default 7000`	Ajuste manual de presupuesto (mW)

2.1 Ejemplo: apagar AP fuera de horario

```
event manager applet POE_OFF_NIGHT
 event timer cron name NIGHT cron-entry "0 22 * * 1-5"
 action 1.0 cli command "enable"
 action 2.0 cli command "configure terminal"
 action 3.0 cli command "interface range Gi1/0/1-4"
 action 4.0 cli command "power inline never"
```

```
 action 5.0 cli command "end"
```

---

3 Cisco EnergyWise (EoX pero aún presente en campus)

```
energywise domain GREEN secret MyKey!
!
interface range Gi1/0/1-48
 energywise level 5 interval 60
```

Los niveles (0-10) definen potencia asignada; un EMS recoge los datos por UDP 43440/43441.

---

4 Métricas de consumo (show power)

```
SW# show power inline
 Interface  Admin  Oper   Power(W)  Device
 Gi1/0/1    auto   on     12.2      AIR-AP2802I
 Gi1/0/2    auto   off     0.0      n/a
```

Para chasis:

```
SW# show environment power all
```

---

5 Buenas prácticas

1. **Auditoría trimestral**: revisar puertos activos sin enlace y deshabilitarlos.

2. Ajustar **timer EEE** solo en enlaces < 30 % utilización; evitar en uplinks críticos.

3. Habilitar **PoE policing** para proteger frente a dispositivos que exceden af/at.

4. Usar **802.3bt (PoE++)** solo donde se requiera (>60 W), planificando la carga del PSU.

5. Documentar perfiles de energía en CMDB y reflejarlos en monitoreo (SNMP OID CISCO-POWER-ETHERNET-EXT-MIB).

Lecturas recomendadas

- IEEE 802.3az-2010 — *Energy Efficient Ethernet*.
- Cisco — *Catalyst PoE Configuration Guide*.
- Cisco — *EnergyWise Implementation Best Practices* (archived).

Automatización avanzada con Ansible (git + CI/CD)

Una vez dominada la copia de seguridad y los cambios simples, el siguiente paso es orquestar la red mediante **Git** y pipelines CI/CD, garantizando revisiones, pruebas y despliegue controlado.

---

1 Flujo de trabajo recomendado

```
GitLab / GitHub
    main        ← Producción estable
    feature/X   ← Cambios (pull request + review)
             ↓  (merge)
CI Pipeline
    ansible-lint
    ansible-playbook --syntax-check
    pytest (test de plantillas)
    despliegue escalonado (staging → prod)
```

---

2 Ejemplo de pipeline (GitLab CI)

```yaml
stages:
  - lint
  - test
  - deploy

variables:
  ANSIBLE_FORCE_COLOR: '1'

lint:
  image: python:3.11
  stage: lint
  script:
    - pip install ansible ansible-lint
    - ansible-lint playbooks/

test:
  stage: test
  script:
    - ansible-playbook -i inventory.yml playbooks/config_vlan.yml --syntax-check

deploy:
```

```
  stage: deploy
  when: manual
  script:
    - ansible-playbook -i inventory.yml playbooks/config_vlan.yml --diff
```

manual en la etapa “deploy” evita cambios no revisados en horas no laborales.

---

3 Patrón idempotente con `state=present/absent`

```
- name: Asegurar que VLAN 30 existe
  cisco.ios.ios_vlan:
    vlan_id: 30
    name: USERS
    state: present
```

Ejecutar el playbook dos veces no provoca cambios adicionales.

---

4 Uso de variables por sitio (group_vars)

```
# group_vars/madrid.yml
vlans:
  - { id: 30, name: USERS_MAD }
  - { id: 40, name: VOICE_MAD }
```

Loop en el playbook:

```
- name: Crear VLANs locales
  cisco.ios.ios_vlan:
    vlan_id: "{{ item.id }}"
    name: "{{ item.name }}"
    state: present
  loop: "{{ vlans }}"
```

---

5 Verificación automática con `assert`

```
- name: Comprobación post-config
  ansible.netcommon.cli_command:
    command: show vlan id 30
  register: vlan30

- name: Validar salida
  assert:
```

```
that:
  - "'30' in vlan30.stdout"
  - "'active' in vlan30.stdout"
```

---

6 Buenas prácticas específicas

1. Habilitar `strategy: linear` para despliegues deterministas.
2. Usar **"check mode"** en producción antes del *commit final*.
3. Registrar **"diff"** en el pipeline; guardar artefactos HTML/JSON.
4. Etiquetar releases (tag v1.2.0) y asociar a cambios en CMDB.
5. Rotar credenciales con **Ansible Vault** + variables de entorno dentro del runner.

Lecturas recomendadas

- Ansible — *Idempotence and Configuration Drift Guide*.
- GitLab — *Best Practices for Network CI/CD*.
- Red Hat — *Event-Driven Ansible Overview*.

Seguridad en switches: Port-Security, BPDU Guard y DHCP Snooping

Los ataques en el nivel de acceso —MAC flooding, BPDU spoof o rogue DHCP— se mitigan con funciones nativas de los switches Cisco. Esta ficha resume la configuración esencial, verificaciones y buenas prácticas.

---

1 Funciones clave

Característica	Amenaza mitigada	Acción realizada
Port-Security	CAM overflow / MAC spoof	Limita y valida MAC por puerto
BPDU Guard	Switch rogue / STP manipulación	Pone el puerto en *err-disable* al recibir BPDU
DHCP Snooping	Servidor DHCP no autorizado	Filtra ofertas DHCP y crea tabla de bindings

---

2 Port-Security paso a paso

```
interface FastEthernet0/10
 switchport mode access
 switchport access vlan 20
 switchport port-security
 switchport port-security maximum 2
 switchport port-security violation restrict
 switchport port-security mac-address sticky
```

Verificación:

```
SW# show port-security interface fa0/10
```

---

3 BPDU Guard y Root Guard

```
interface range Fa0/1-24
 spanning-tree portfast
 spanning-tree bpduguard enable

interface Gi0/1        ! Trunk a distribución
 spanning-tree guard root
```

Si un puerto con BPDU Guard recibe un BPDU → **err-disable**.

---

4 DHCP Snooping + Option 82

```
ip dhcp snooping
ip dhcp snooping vlan 10,20

! Puertos trusted (hacia servidor DHCP)
interface Gig0/1
 ip dhcp snooping trust

! Límite de solicitudes
interface range Fa0/1-24
 ip dhcp snooping limit rate 25

ip dhcp snooping information option   ! Inserta Option 82
```

Verificación:

```
SW# show ip dhcp snooping
SW# show ip dhcp snooping binding
```

---

5 Troubleshooting rápido

Síntoma / Log	Causa probable	Comando útil
Puerto en **err-disable**	BPDU o violación MAC	`show errdisable status`
Clientes sin DHCP	Puerto no *trust*	`show ip dhcp snooping`
MAC sticky no persiste tras reboot	Faltó `write memory`	`wr mem`

---

6 Buenas prácticas

1. Habilitar **PortFast + BPDU Guard** en todos los puertos de acceso.
2. Ajustar **máximo de MAC** al tipo de dispositivo (1–2 PCs, 3–5 VoIP+PC).
3. Registrar violaciones (`logging event port-security`).
4. Configurar **errdisable recovery** con un temporizador (300 s) para autosanación.

5. Documentar puertos *trusted* de DHCP en la CMDB y revisarlos tras cambios.

Lecturas recomendadas

- Cisco IOS — *Switch Security Configuration Guide*.
- Cisco SAFE — *Campus Security Validation*.
- IEEE 802.1X-2020 — *Port-based Network Access Control*.

Topologías de campus: Core-Distribution-Access y variantes

Un campus bien diseñado asegura alta disponibilidad, escalabilidad y facilita la implementación de QoS y seguridad. El modelo clásico Core-Distribution-Access sigue vigente, aunque surgen variantes como Collapsed Core o Fabric (SD-Access).

---

1 Capas del modelo jerárquico

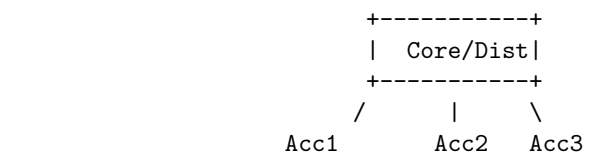
Capa	Función principal	Características clave
Access	Conectar dispositivos finales	PoE, seguridad 802.1X, port-security
Distribution	Agregación y políticas L3	ACL, QoS, routing, span VLANs por site
Core	Conmutación L3 de alta velocidad	Redundancia, enlaces 10/40/100 GbE

> La separación de funciones simplifica el *troubleshooting* y permite escalar cada capa de forma independiente.

---

2 Variante "Collapsed Core"

Cuando el campus es pequeño (< 2 edificios) se fusionan **Core+Dist**:



Ventajas: Menos switches y menor coste.
Desventaja: Solo dos chasis concentran todo el tráfico.

---

3 Enlaces y protocolos recomendados

- **Access Dist**: Ethernet 10 GbE, LACP Port-Channel, Rapid-PVST+ o MST.

- **Dist Core**: 40/100 GbE, routing L3 (OSPF/BGP), ECMP.
- **FHRP**: HSRP o VRRP en distribución (gateway VIP por VLAN).
- **Spine-Leaf** (campus nuevo): VXLAN EVPN para movilidad L2/L3.

---

4 Diseño de redundancia

```
Usuarios -> AccSW -----\
                        Dist1 === Core === Dist2
Usuarios -> AccSW -----/             \---- AccSW (otro edificio)
```

- 2 enlaces por Access (dual-home).
- Distribución en *vPC/MEC* (NX-OS) o StackWise Virtual (Catalyst).
- Core con enlaces cruzados para evitar fallo único (uplink, PSU, supervisor).

---

5 Buenas prácticas operativas

1. **Bloquear spanning-tree** en core: todo enrutado L3 hacia el acceso.
2. Usar **/30 o /31** en enlaces punto-a-punto para sumarizar fácil.
3. **QoS consistente**: clasificar en Access, marcar en Dist, colas en Core.
4. Separar **gestión (OOB)** en VRF y VLAN dedicadas, sin rutas a Internet.
5. Documentar diagramas (Visio/draw.io) con nombres, velocidades y etiquetas.

Lecturas recomendadas

- Cisco SAFE — *Campus Design Guide 3-Tier & Collapsed Core.*
- Cisco Press — *Campus Network Architecture*, cap. 2-3.
- RFC 7432 — *EVPN VXLAN for Campus Fabrics.*

Balanceadores de carga L4 vs L7

Los **load balancers** distribuyen solicitudes entre varios servidores para garantizar disponibilidad y escalar aplicaciones. Existen dos grandes categorías: **L4 (transport)** y **L7 (aplicación)**.

---

1 Diferencias clave

Capa	Algoritmo → decisión	Contenido analizado	Ejemplos de equipos
L4	IP + puerto destino	TCP/UDP header	Cisco CSM, HAProxy TCP-mode, AWS NLB
L7	Host, URI, cookie, header	HTTP/S, gRPC, TLS SNI	F5 BIG-IP, NGINX Plus, AWS ALB

> L7 permite *path-based routing* y reescritura; L4 ofrece mayor rendimiento y menor latencia.

---

2 Ejemplo rápido HAProxy (L4 TCP)

```
frontend mysql-in
 bind *:3306
 mode tcp
 default_backend mysql-servers

backend mysql-servers
 mode tcp
 balance source
 server db01 10.10.30.11:3306 check
 server db02 10.10.30.12:3306 check backup
```

---

3 Ejemplo NGINX (L7 HTTP con TLS)

```
http {
 upstream web-app {
   zone backend 64k;
   least_conn;
   server 10.10.20.11 max_fails=3 fail_timeout=5s;
   server 10.10.20.12 backup;
 }
```

```
 server {
   listen 443 ssl;
   server_name app.example.com;
   ssl_certificate     /etc/ssl/app.crt;
   ssl_certificate_key /etc/ssl/app.key;

   location /api/v1 {
     proxy_pass http://web-app;
     proxy_set_header X-Forwarded-For $remote_addr;
   }
 }
}
```

---

4 Políticas de salud (*health checks*)

Tipo	Capa	Ejemplo	Uso recomendado
ping	3	ICMP echo	Solo validar reachability
TCP	4	`telnet host 80`	Servicios sin protocolo extra
HTTP	7	`GET /status`	APIs, apps web, custom codes

---

5 Troubleshooting rápido

Síntoma / Log	Posible causa	Herramienta
Conexión abre pero cierre inmediato	Mismatch L4/L7	`tcpdump -nn`
Servidor marcado “DOWN”	Health check falla	Revisar firewall, logs
Afinidad de sesión rota	Cookie no insertada	Inspect DevTools

---

6 Buenas prácticas

1. En L4, habilitar **TCP keepalive** para detectar servidores colgados.
2. En L7, usar **TLS offload** y reencriptar si necesitas inspección IDS.
3. Documentar políticas de **stickiness** (IP-hash, cookie, header).

4. Registrar métricas (5xx rates, latency P95) en Prometheus/Grafana.
5. Mantener playbooks IaC (Terraform/Ansible) para cambios reproducibles.

Lecturas recomendadas

- F5 — *Load Balancing 101 White Paper.*
- NGINX — *Comparing Layer 4 vs Layer 7 Load Balancing.*
- Cisco — *Server Load Balancing Configuration Guide.*

Arquitectura Spine-Leaf (CLOS) para centros de datos

La topología **Spine–Leaf** (Clos de tres etapas) ofrece alta capacidad este-oeste, latencia predecible y escalabilidad horizontal mediante enlaces **ECMP**. Es la base de fabrics VXLAN-EVPN y redes SDN modernas (ACI, NSX-T, Cumulus).

---

1 Componentes básicos

Rol	Función principal	Características clave
Leaf	Conecta servidores / TOR	10/25/40 GbE downlinks; 40/100 GbE uplinks
Spine	Núcleo L3; reenvío ECMP	Sólo interconecta leafs; Tabla MAC/IP pequeña

Regla de oro: servidores **NUNCA** se conectan a spines.

---

2 Relación fan-out (oversubscription)

Fórmula rápida:

```
Oversub = (Leaf  Spine uplink BW) / (Leaf  Server downlink BW total)
```

Ejemplo: Leaf con 48×25 GbE hacia hosts (1200 Gb/s) y 8×100 GbE hacia spines (800 Gb/s) → **1.5 : 1**. Diseña $3:1$ para tráfico HPC.

---

3 Protocolos comunes

Capa	Protocolo	Motivo
L2.5	**VXLAN EVPN**	Aislación L2, movilidad, multitenant
L3	**eBGP** (MP-BGP EVPN)	ECMP + control-plane escalable
L3	OSPF/IS-IS (alt.)	Solo si no se usa EVPN

eBGP best-path + *next-hop-self* simplifica tablas de ruta en leafs.

---

4 Configuración de ejemplo (NX-OS, eBGP EVPN)

```
interface Ethernet1/1
  description  Spine-01
  no switchport
  ip address 172.16.0.1/31

router bgp 65010
  router-id 10.255.0.1
  address-family ipv4 unicast
  neighbor 172.16.0.0 remote-as 65000
  neighbor 172.16.0.0 next-hop-self
  address-family l2vpn evpn
    neighbor 172.16.0.0 activate
    advertise-l2vpn-evpn
```

El spine repite la configuración cambiando IP y AS (común o distinto).

---

5 Verificación rápida

```
Leaf# show bgp l2vpn evpn summary
Leaf# show interface brief status | include Eth1/
Spine# show ip route 10.255.0.0/24
```

Estado clave: **Established**, **ECMP paths**, **VXLAN VNIs up**.

---

6 Ventajas vs core jerárquico

- Cada salto leaf→spine→leaf es **3 µs** con ASIC recientes.
- Agregar spines = **lineal** incremento de capacidad (Nx100 GbE).
- Simplifica **cableado**: sólo vertical; sin cross-links laterales.

---

7 Desafíos y mitigaciones

Problema	Mitigación recomendada
Tabla ARP/MAC enorme	EVPN control-plane (MAC/IP route)
Design oversub alto	Añadir spines / 400 GbE uplinks
Source-leaf black-hole	BFD/Graceful Restart

8 Buenas prácticas

1. **Paridad**: mismo nº de uplinks de cada leaf a *todos* los spines.
2. **ASN idéntico** por rol (65000 spines, 65010 leafs) o ebgp-multi-hop.
3. Documentar **cable-matrix** (Excel / NetBox) para instalación sin errores.
4. Automatizar configs (Ansible/pyATS) para miles de puertos.
5. Supervisar métricas **buffer/queue** en p-fabrics (Telemetry gRPC).

Lecturas recomendadas

- RFC 7938 — *Use of BGP for Routing in Large-Scale Data Centers.*
- Cisco — *VXLAN EVPN Design Guide (CVD).*
- Juniper — *Spine-Leaf Architecture Best Practices.*

Estrategias de migración a IPv6

Adoptar **IPv6** implica coexistir con IPv4 durante años. Las organizaciones suelen combinar **dual-stack**, **túneles** y **traducción NAT64/DNS64** según su infraestructura y los servicios externos que consumen.

---

1 Fases de un plan realista

Paso	Objetivo	Resultado esperado
1	Inventario de redes y equipos	Lista de compatibilidad v6
2	Activar dual-stack en core/DC	Enrutamiento IPv6 interno operativo
3	Exponer servicios públicos en v6	DNS AAAA, certs SAN, WAF adaptado
4	Habilitar NAT64/DNS64 para sólo-v6	Clientes IPv6 acceden a destinos IPv4
5	Retirar v4 en segmentos seleccionados	Ahorro de NAT, simplificación ACL

---

2 Métodos disponibles

Técnica	Cuándo usar	Notas rápidas
Dual-Stack	Hardware soporta ambas pilas	Ruta más flexible; consume recursos v4
6RD / 6to4	ISP solo IPv4, despliegue rápido	Túneles automáticos, MTU +20 B
ISATAP	LAN IPv4, hosts Windows legacy	Decreciente; considerar desuso
NAT64/DNS64	Solo clientes IPv6, servidores IPv4	Traducción stateful de 64 → 32
464XLAT	Apps hard-coded IPv4 (móvil)	CLAT en CPE o dispositivo

---

3 Configuración de muestra NAT64/DNS64 (IOS XE)

```
ipv6 nat64 prefix-stateful 64:ff9b::/96

interface Gig0/0
 description WAN-IPv4
 ip address 198.51.100.2 255.255.255.252
 ip nat inside
```

```
interface Gig0/1
 description LAN-IPv6
 ipv6 address 2001:db8:10::1/64
 ipv6 nat64 enable

! DNS64 en IOS XE SD-WAN / external BIND:
window inet
  dns64 64:ff9b::/96
```

---

4 Métricas clave de éxito

Métrica	Herramienta	Objetivo
% de tráfico IPv6	NetFlow/IPFIX, SLAAC stats	> 50 % primer año
Fallos DNS AAAA	SIEM, resolver logs	< 0.5 %
Latencia extra NAT64	`ping6`, HTTP RUM	< 10 ms sobre dual-stack

---

5 Troubleshooting rápido

Síntoma	Causa probable	Comando útil / acción
`ping6` falla, `ping` OK	Filtros ICMPv6	Revisar ACL / FW
DNS AAAA resuelve, no conecta	Ruta / ND faltante	`traceroute6`, `show ipv6 route`
App legacy sin IPv6	Falta 464XLAT / DNS64	Implementar CLAT

---

6 Buenas prácticas

1. Reservar bloques /48 per site y /64 por VLAN; documentar en IPAM.
2. Habilitar **RA Guard** y **ND Inspection** desde el inicio.
3. Configurar **IPv6 ACL espejo** de políticas IPv4 (least privilege).
4. Medir consumo NAT64 para dimensionar CPU/ASIC.
5. Formar a soporte: comandos `ping6`, `%IF` para link-local, etc.

Lecturas recomendadas

- RIPE-690 — *Best Current Practices IPv6* (Addressing Plan).
- Cisco — *IPv6 Transition Guide: Dual-Stack, NAT64, 464XLAT.*
- RFC 6180 — *Guidelines for IPv6 Deployment.*

Diagramas de red dinámicos y documentación viva

Los diagramas "estáticos" en Visio pierden vigencia en semanas.
Una práctica moderna es generar **diagramas dinámicos** a partir de los datos reales de la red (inventario, LLDP, routing). Así la documentación se actualiza de forma automática o con mínimas intervenciones.

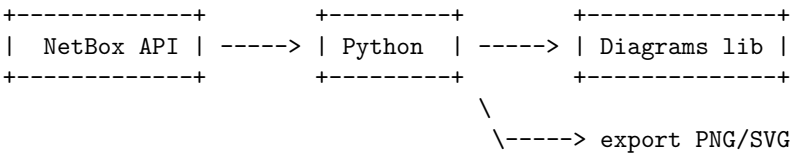
---

1 Fuentes de verdad típicas

Origen	Datos aportados	Herramientas
NetBox	Dispositivos, interfaces, cables	API REST/GraphQL
Nautobot	Similar a NetBox + plugins	Jobs GitOps
SNMP/LLDP	Vecindad física	Netshot, Nornir, PySNMP
BGP/OSPF	Topología L3, prefijos	Batfish, ExaBGP

---

2 Flujos de trabajo recomendados



El script Python consulta NetBox y dibuja con ***diagrams*** *o* ***Graphviz****.*

---

3 Ejemplo mínimo con la librería *diagrams*

```
from diagrams import Diagram, Edge
from diagrams.cisco.network import Router, Switch

with Diagram("Campus Core", show=False, filename="campus_core"):
    core1 = Router("Core-1")
    core2 = Router("Core-2")
    dist1 = Switch("Dist-1")
    dist2 = Switch("Dist-2")

    core1 >> Edge(label="100G") >> dist1
    core1 >> Edge(label="100G") >> dist2
```

```
core2 >> Edge(label="100G") >> dist1
core2 >> Edge(label="100G") >> dist2
```

Resultado: **campus_core.png** generado automáticamente.

---

4 Herramientas open-source populares

Herramienta	Lenguaje	Destacado
Diagrams	Python	Iconos AWS, GCP, Cisco, etc.; simple
PlantUML	Texto	Compatible con CI; export SVG/PNG
Graphviz / DOT	Texto	Gran control de layout
Network-Topology-Mapper (NTM.js)	JS	Render en navegador en tiempo real

---

5 Integración CI/CD

1. Repositorio Git → *pre-commit* ejecuta script de generación.
2. Pipeline (`.gitlab-ci.yml`) exporta PNG/SVG y los guarda como artefactos.
3. Docs en **MkDocs / Sphinx** incluyen `<img>` de los artefactos más recientes.

---

6 Buenas prácticas

- Mantén **"source of truth" único** (NetBox, CMDB); no edites PNG a mano.
- Usa **etiquetas (tags)** para filtros (solo dispositivos `role=core`).
- Programar regeneración **nightly** o tras *merge* a rama `main`.
- Exporta **SVG** para zoom infinito y anotaciones en Confluence/Wiki.
- Versiona diagramas en Git para rastrear cambios visuales a lo largo del tiempo.

Lecturas recomendadas

- NetBox — *Graphing and Diagram Automation Cookbook*.
- Diagrams Python — *Official Documentation*.
- GitLab — *Automated Network Diagrams with CI/CD Blog Post*.