GRAPH RAG · SERVERLESS AI NETWORKING R&D · WI-FI STANDARDS

A natural-language knowledge assistant for the Wi-Fi specification, running fully serverless

We built an AI assistant that turns the IEEE 802.11 specification, including Wi-Fi 7, into citation-backed answers, structured technical documents and causal analysis of protocol dependencies. The full LLM, vector retrieval and graph reasoning stack runs on AWS Lambda, with cold start cut from 10-15+ seconds to about 5 and complex queries from 13+ seconds to about 8.

01 CLIENT & NEED

The client drives AI-based innovation in networking, aiming to raise productivity and cut effort for **network and embedded engineers**. They needed a responsive, automated assistant to help engineers and researchers navigate the dense **Wi-Fi specifications**, from Wi-Fi 7 back through earlier amendments.

- **Intelligent Q&A and documents.** Precise, citation-backed answers and frame-level and field-level detail straight from the standard.
- **Causal graph analysis.** Show how Wi-Fi features affect throughput, latency and energy, and how they constrain or amplify one another.
- **Serverless by design.** Run the LLM, vector retrieval and graph stack on AWS Lambda to keep costs low and scale with demand.
- **Hard latency targets.** Cold start of 5.0 s at most, and under 8.0 s for complex queries with real-time streaming.
- **Zero-downtime CI/CD.** Isolated beta testing and production rollouts without interrupting live users.

BEFORE IMPLEMENTATION [T-0]

Engineers spent hours parsing thousands of pages of IEEE 802.11 PDFs to extract frame formats, trace protocol dependencies and judge feature impact.

Early Lambda deployments of the LLM, LangChain, NetworkX and vector stack hit 10-15+ s cold starts and 13+ s queries, caused by CPU-bound XML graph parsing, sequential imports and hundreds of database and API connections opened per query.

AFTER IMPLEMENTATION [4 phases]

A four-phase production pipeline covering knowledge ingestion, semantic routing, serverless optimization and enterprise CI/CD.

A hybrid Graph RAG engine answers in conversational time, parallel lazy loading absorbs cold starts, pooled connections protect query latency, and stable and beta Lambda aliases ship releases with no downtime.

02 RESULTS

~5.0 s
Lambda cold start
Init duration down from 10-15+ s, and absorbed in the background while the user types.

~8.0 s
Complex query time
Graph RAG streaming queries down from 13+ s.

50 ms
Knowledge graph load
Down from over 2 s after moving from XML to a native binary graph.

Zero
Release downtime
Beta stress-tested in isolation, then promoted to production instantly.

03 SOLUTION: A FOUR-PHASE PRODUCTION PIPELINE

PHASE 01

Hybrid Graph RAG and entity resolution

A retrieval engine combining **ChromaDB** for semantic chunk retrieval with **PostgreSQL**. A deterministic LLM router classifies intent (impact, comparison, backpressure) and maps it to a custom **NetworkX** knowledge graph that preserves causal links between Wi-Fi features and network primitives.

PHASE 02

Serverless latency optimization

The compute layer was re-engineered to use Lambda's unthrottled CPU burst during init. Heavy ML libraries sit behind import shields and load lazily, a **ThreadPoolExecutor** runs database connections, LLM warm-up and graph loading in parallel, and the XML graph became a native binary, cutting parse time from **2+ s to 50 ms**.

PHASE 03

Persistent connections and network resilience

TCP socket exhaustion was removed to protect the 8-second query budget. Thread-safe global singletons cache OpenAI clients so keep-alive connections are reused, and a **ThreadedConnectionPool** in the PostgreSQL pipeline replaces N × M connection growth with O(1) pooled socket leasing.

PHASE 04

Responsive UX and CI/CD infrastructure

A wake-up endpoint absorbs the cold start in the background while the user types their first query. Releases use immutable **ECR** Docker tags and Lambda aliases (stable and beta) on permanent function URLs, so **Locust** stress tests run in beta without touching production data.

04 PERFORMANCE, BEFORE AND AFTER

METRIC	BEFORE	AFTER	WHAT CHANGED
Lambda cold start	10-15+ s	~5.0 s	Import shields, lazy loading, parallel init
Complex query execution	13+ s	~8.0 s	Cached clients, pooled connections, streaming
Knowledge graph parse	2+ s	50 ms	XML graph converted to native Python binary
Database and LLM connections	N × M	O(1)	Singletons and ThreadedConnectionPool
Release downtime	At risk	Zero	Stable and beta aliases, immutable ECR tags

— OUTCOME

The IEEE 802.11 specification is now a fast, structured protocol research and market intelligence tool, replacing hours of manual reading. Pooled connections and parallel loading handle high-concurrency traffic, so the client gets low-cost serverless compute with real-time AI responsiveness.

TECHNOLOGY STACK

AWS Lambda · Amazon ECR · OpenAI · LangChain · ChromaDB · PostgreSQL · NetworkX · Docker · Locust