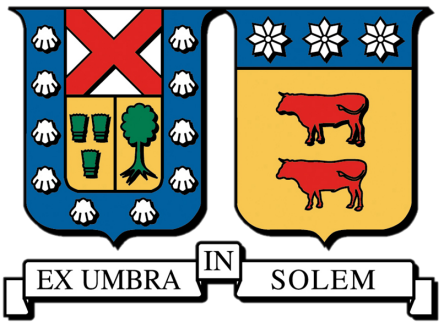
UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE ELECTRÓNICA
VALPARAÍSO - CHILE



"Diseño de un Simulador PON para la Optimización de DBA mediante Aprendizaje por Refuerzo"

Jorge Eduardo Barrios Núñez

MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE INGENIERO CIVIL TELEMÁTICO

PROFESOR GUÍA: **Nicolás Jara**

PROFESOR CORREFERENTE: **Patricia Morales**

Diciembre 2025

CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE MONOGRAFÍA A REPOSITORIO ACADÉMICO

1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

Tipo de monografía (marcar una opción): ☒ Memoria o trabajo de título; ☐ Tesis de Postgrado;

Título del trabajo: Diseño de un Simulador PON para la Optimización de DBA mediante aprendizaje por refuerzo

Nombre del candidato(a): Jorge Eduardo Barrios Nuñez

Carrera / Grado: Ingeniería Civil Telemática

Campus: Casa Central Valparaiso **;** **Departamento**: Electrónica

2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, Nicolas Jara Carvallo, en mi calidad de profesor(a) guía/director(a) del trabajo académico mencionado anteriormente **DEJO CONSTANCIA** que:

- He revisado esta versión del documento y corresponde a la versión final aprobada del trabajo.
- El trabajo cumple con los requisitos académicos y de formato establecidos por la institución

3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL

☒ El trabajo **NO contiene información que amerite confidencialidad** y puede ser publicado de inmediato en repositorio con acceso abierto.

☐ El trabajo **CONTIENE** información con potenciales implicancias de propiedad industrial o intelectual y requiere un periodo de confidencialidad (embargo) por:

☐ 6 meses; ☐ 12 meses; ☐ 2 años; ☐ 3 años; ☐ 5 años; ☐ 10 años

Fundamentación de la necesidad de confidencialidad (obligatorio si se solicita embargo):

4.- FIRMAS

Profesor(a) guía o director(a) de memoria o tesis:

Fecha: 17/03/2026 **; Firma:** _____

Estudiante o Candidato(a):

Fecha: 17/03/2026 **; Firma:** _____

Este formulario debe ser insertado como página 2 de la memoria o tesis, completado y firmado por estudiante y profesor(a) antes de la entrega en portal PRISMA de Biblioteca USM.

Agradecimientos

En primer lugar, quiero agradecer a mi familia, el núcleo de mi vida y la razón por la cual llegué a donde estoy. Particularmente, agradezco el apoyo incondicional que me dieron mi mamá y mi papá; no hay espacio, palabras ni tiempo suficientes para poder demostrar siquiera una fracción de lo agradecido que estoy con ustedes. No tan solo me dieron la vida, sino todas las herramientas y el cariño para poder vivirla. Siendo aquí intencionalmente redundante: muchas gracias por todo; los amo mucho.

A mis hermanos, Renato y Matías: creo que no son conscientes de lo influyentes que fueron en mí. Desde que tengo uso de razón siempre los estuve admirando, siendo ejemplo para lo bueno y lo malo(para que nos andamos con cosas). Cada uno tomó su camino, pero siempre me quedaré con que, pese a los contratiempos, cumplieron su objetivo y demostraron que se puede; que yo también puedo.

A mi pareja, Javiera, la cual compartió conmigo todo mi periodo universitario. Aquí nuevamente puedo confirmar que soy un afortunado y que la vida pone gente maravillosa en mi camino. Me has ayudado tanto académicamente como emocionalmente, siempre acompañándome en mis peores momentos y celebrando con euforia mis triunfos. Aquí puedo declarar que eres un pilar en mi vida, mi mejor amigo y la persona que escogí amar.

A mis compañeros y aliados académicos. Las sesiones de estudio, las tareas y los proyectos que realizamos, muchas veces hubieran sido un infierno si no nos hubiéramos apoyado de la forma en que lo hicimos. Particularmente, agradezco a Iván, el mejor compañero y vecino que pude pedir. Gracias por ayudarme cuando me faltaba algo, por dejarme la llave de repuesto cuando se me quedaban las mías dentro del departamento, por dar lo mejor de ti cada vez que teníamos que entregar una evaluación y por no hacerme sentir solo cuando me quedaba en Valparaíso.

Resumen

Actualmente existe un aumento masivo en la demanda de ancho de banda, impulsado por servicios exigentes como el streaming en alta resolución y los juegos en la nube. Las redes PON son la infraestructura estándar para soportar este tráfico, pero sus mecanismos de control (DBA) deben evolucionar para cumplir con requisitos de calidad cada vez más estrictos, integrando desde soluciones heurísticas clásicas hasta nuevas técnicas de Inteligencia Artificial.

El problema principal es que investigadores y empresas carecen de herramientas de simulación flexibles para validar estas nuevas estrategias. La mayoría de los simuladores actuales son cerrados o difíciles de integrar con tecnologías modernas, lo que impide probar y comparar ágilmente distintos algoritmos de gestión de red.

Para cubrir esta necesidad, este trabajo presenta el diseño y validación de PonLab, un simulador de redes PON de código abierto escrito en Python. Esta herramienta funciona como un banco de pruebas modular, diseñado para que académicos y desarrolladores puedan implementar sus propios algoritmos, ya sean heurísticos, SDN o basados en Aprendizaje por Refuerzo. Su arquitectura híbrida permite reducir la carga computacional sin perder precisión, ofreciendo un entorno unificado para corroborar el comportamiento de la red bajo distintos enfoques.

Keywords: Redes Ópticas Pasivas (PON), Asignación Dinámica de Ancho de Banda (DBA), Simulación de Redes, Código Abierto, Aprendizaje por Refuerzo (RL), Python.

Abstract

Currently, there is a massive surge in bandwidth demand, driven by demanding services such as high-resolution streaming and cloud gaming. Passive Optical Networks (PON) serve as the standard infrastructure to support this traffic; however, their control mechanisms (DBA) must evolve to meet increasingly strict quality requirements, integrating everything from classic heuristic solutions to new Artificial Intelligence techniques.

The primary challenge is that researchers and industry professionals lack flexible simulation tools to validate these new strategies. Most current simulators are closed-source or difficult to integrate with modern technologies, hindering the agile testing and comparison of different network management algorithms.

To address this need, this work presents the design and validation of PonLab, an open-source PON simulator written in Python. This tool serves as a modular testbed, designed to enable academics and developers to implement their own algorithms, whether heuristic, SDN, or Reinforcement Learning-based. Its hybrid architecture reduces computational overhead without compromising precision, offering a unified environment to corroborate network behavior under different approaches.

Keywords: Passive Optical Networks (PON), Dynamic Bandwidth Allocation (DBA), Network Simulation, Open Source, Reinforcement Learning (RL), Python.

Glosario y Acrónimos

Acrónimos

5G Quinta Generación de tecnología de telefonía móvil.

DBA *Dynamic Bandwidth Allocation* (Asignación Dinámica de Ancho de Banda). Conjunto de algoritmos para distribuir el ancho de banda del canal de subida en una red PON.

DL *Deep Learning* (Aprendizaje Profundo).

EPON *Ethernet Passive Optical Network*. Estándar de red PON basado en Ethernet.

FCFS *First-Come, First-Served*. Algoritmo de planificación donde las solicitudes se atienden en el orden en que llegan.

FTTH *Fiber-to-the-Home* (Fibra hasta el Hogar).

GIANT *GigaPON Access Network Traffic Management*. Algoritmo de DBA que garantiza anchos de banda mínimos y distribuye el excedente.

GPON *Gigabit Passive Optical Network*. Estándar de red PON que ofrece velocidades de hasta 2.5 Gbps.

GUI *Graphical User Interface* (Interfaz Gráfica de Usuario).

IPACT *Interleaved Polling with Adaptive Cycle Time*. Algoritmo de DBA que ajusta dinámicamente el ciclo de sondeo.

ITU *International Telecommunication Union* (Unión Internacional de Telecomunicaciones).

JSON *JavaScript Object Notation*. Formato de texto ligero para el intercambio de datos.

LSTM *Long Short-Term Memory*. Arquitectura de red neuronal recurrente utilizada en aprendizaje profundo.

MARL *Multi-Agent Reinforcement Learning* (Aprendizaje por Refuerzo Multi-Agente).

MDN *Márkov Decision Process (Proceso de Decisión de Márkov)*

NS-3 *Network Simulator 3*. Simulador de eventos discretos para redes de comunicación.

OLT *Optical Line Terminal* (Terminal de Línea Óptica). Equipo situado en la central del proveedor que gestiona la red PON.

OMNeT++ Simulador de redes de eventos discretos, modular y basado en componentes.

ONU *Optical Network Unit* (Unidad de Red Óptica). Equipo situado en las instalaciones del usuario final.

PON *Passive Optical Network* (Red Óptica Pasiva).

PPO *Proximal Policy Optimization*. Algoritmo de Aprendizaje por Refuerzo.

PySide6 Biblioteca de Python que actúa como binding para el framework de GUI Qt.

QoS *Quality of Service* (Calidad de Servicio).

RL *Reinforcement Learning* (Aprendizaje por Refuerzo).

RL-DBA DBA basado en Aprendizaje por Refuerzo.

SB3 Stable-Baselines3. Biblioteca de Python que ofrece implementaciones de algoritmos de RL.

T-CONT *Traffic Container*. Contenedor de tráfico en redes GPON para gestionar la calidad de servicio.

XGS-PON Estándar de red PON con capacidad para 10 Gbps simétricos.

Glosario de Términos

Agente En RL, es la entidad que aprende a tomar decisiones interactuando con un entorno para maximizar una recompensa.

Entorno En RL, es el mundo o sistema con el que el agente interactúa. En este trabajo, el simulador PON actúa como el entorno.

Evento Discreto Ocurrencia instantánea en el tiempo que puede cambiar el estado de un sistema de simulación.

Función de Recompensa Señal numérica que recibe el agente de RL del entorno para evaluar la calidad de una acción en un estado determinado.

Heurística Método para resolver problemas prácticos mediante reglas empíricas o atajos, sin garantizar una solución óptima.

Política En RL, es la estrategia del agente, que mapea estados del entorno a acciones a tomar.

Sondeo (Polling) Proceso mediante el cual la OLT consulta a las ONUs para conocer su estado de búfer y demanda de ancho de banda.

Upstream (Canal de subida) Dirección de la comunicación desde el usuario (ONU) hacia la central (OLT).

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

1.1. Contexto y Motivación

La transformación digital global ha acelerado la demanda de conectividad de alta velocidad a niveles sin precedentes. Según reportes recientes de la Unión Internacional de Telecomunicaciones (ITU), en 2025 aproximadamente el 74 % de la población mundial ya se encuentra conectada a internet, lo que representa cerca de 6 mil millones de usuarios activos [1, 2].

La **Figura 1.1** detalla esta progresión histórica, evidenciando que el incremento en el número de usuarios supera la simple correlación con el crecimiento poblacional. Esto sugiere que factores como el relevo generacional y la simplificación del acceso tecnológico han impulsado una adopción más profunda del entorno digital a nivel mundial.

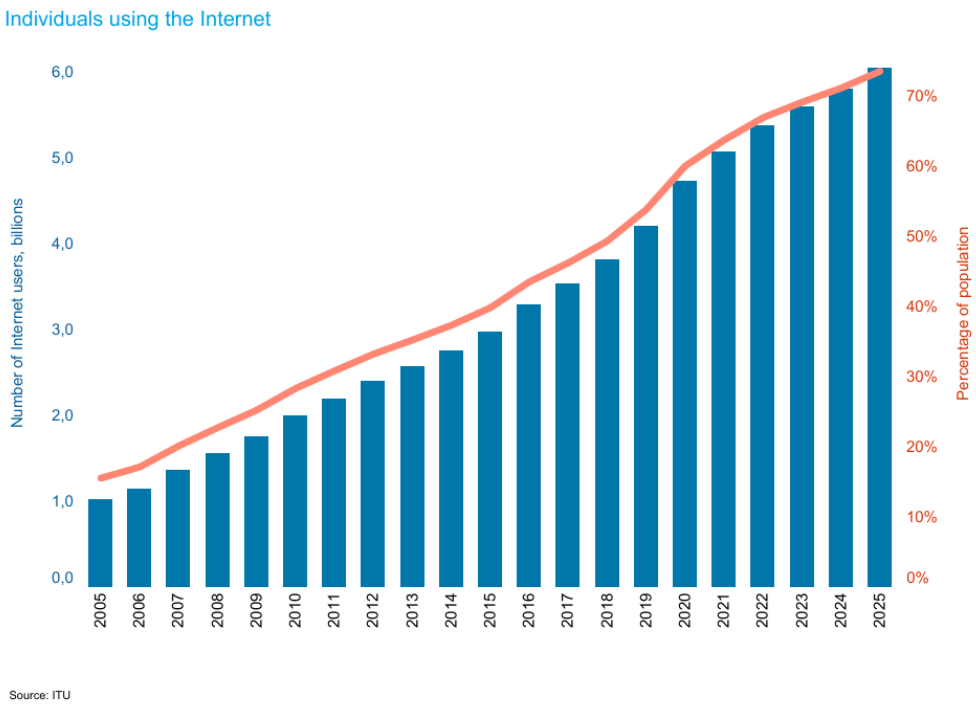


Figura 1.1: Crecimiento global de individuos que utilizan Internet (2005-2025). Fuente: https://www.itu.int/itu-d/reports/statistics/2025/10/15/ff25-internet-use/

Este crecimiento no es solo cuantitativo, sino cualitativo: aplicaciones críticas como el streaming de video en 4K/8K y los juegos en la nube (*cloud gaming*) han disparado el consumo de datos. Se proyecta que el tráfico de datos móviles alcance los 310 Exabytes mensuales para 2031, con las redes 5G transportando más del 40 % de este tráfico hacia finales de 2025 [3].

Dicha tendencia se analiza en la **Figura 1.2**, donde se visualiza la proyección del tráfico de datos. Este aumento exponencial justifica la necesidad de infraestructuras más robustas para soportar la carga de la red en los próximos años.

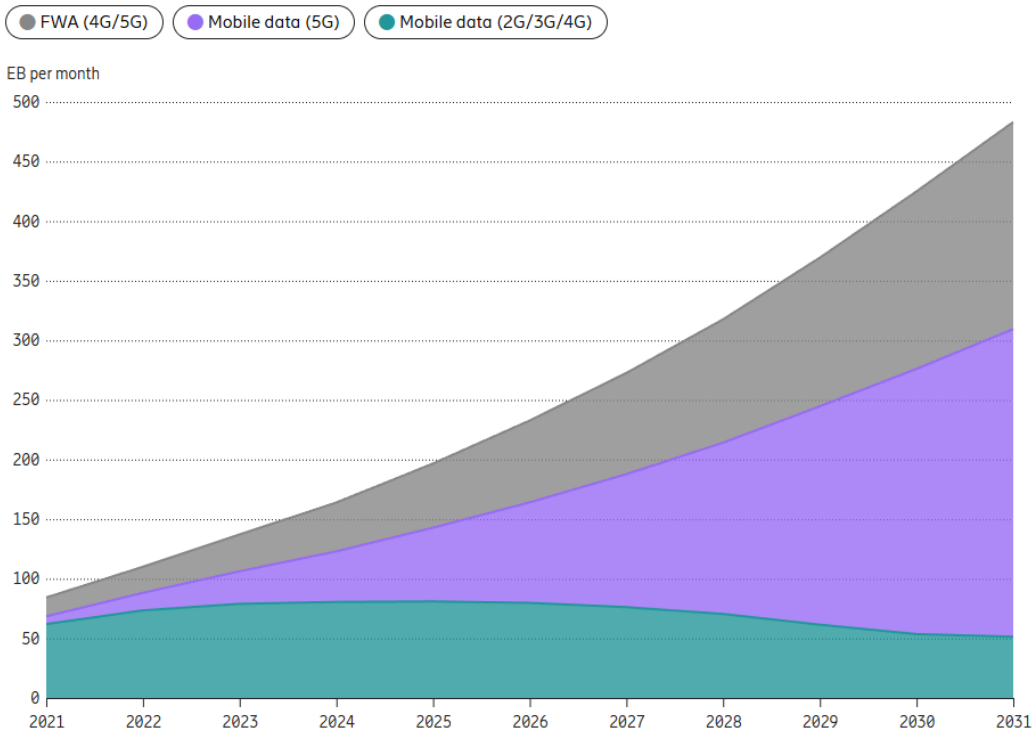


Figura 1.2: Proyección del tráfico global de datos móviles (2021-2031). Fuente: https://www.ericsson.com/en/reports-and-papers/mobility-report/dataforecasts/mobile-traffic-forecast

En este escenario de alta demanda, la fibra óptica se ha consolidado como la infraestructura física esencial. El mercado de Fibra hasta el Hogar (FTTH) continúa su expansión, con una proyección de crecimiento desde los 62 mil millones de USD en 2024 hasta superar los 199 mil millones en 2034 [4].

Como se muestra en la **Figura 1.3**, este mercado experimentará una aceleración significativa en la próxima década, consolidando a FTTH como la tecnología predominante para satisfacer las necesidades de banda ancha residencial.

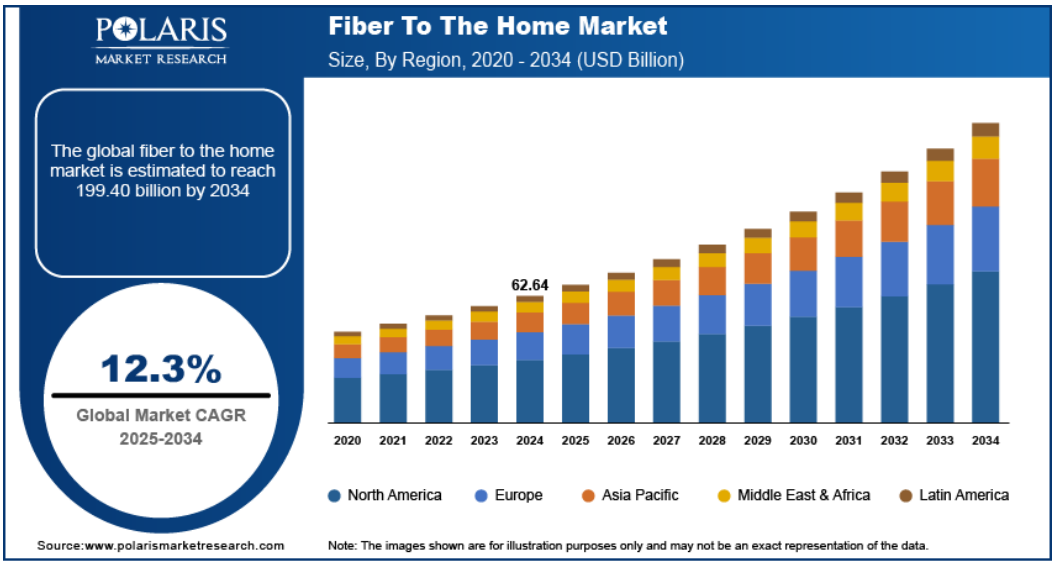


Figura 1.3: Proyección de crecimiento global de FTTH (2020-2034).
Fuente: https://www.polarismarketresearch.com/industry-analysis/fiber-to-the-home-market

Las Redes Ópticas Pasivas (PON) dominan la "última milla"por su eficiencia energética y bajo coste. No obstante, la naturaleza compartida del canal de subida (*upstream*) en una PON crea un cuello de botella crítico. Si bien la capacidad física de la fibra es enorme, la eficiencia real percibida por el usuario depende enteramente del algoritmo de Asignación Dinámica de Ancho de Banda (DBA).

1.2. Definición del Problema

A pesar de la evolución en la infraestructura física, los mecanismos lógicos de control no han avanzado al mismo ritmo. Los algoritmos DBA tradicionales desplegados en la actualidad, como IPACT o GIANT, son de naturaleza reactiva y estática. Operan bajo reglas fijas diseñadas para patrones de tráfico de hace dos décadas (navegación web básica y correo electrónico) y carecen de la inteligencia para adaptarse a la heterogeneidad del tráfico moderno, caracterizado por ráfagas violentas y requerimientos estrictos de latencia.

Esta rigidez provoca una asignación de recursos no completamente eficiente: se des-

perdicia ancho de banda en usuarios inactivos mientras se generan colas y latencia en usuarios con aplicaciones críticas. Aunque la literatura académica sugiere que el Aprendizaje por Refuerzo (RL) puede superar estas limitaciones aprendiendo políticas adaptativas, la investigación en esta área enfrenta una barrera técnica significativa: la falta de herramientas de simulación adecuadas. Los simuladores de propósito general (como NS-3 u OMNeT++) imponen una sobrecarga computacional prohibitiva para el entrenamiento de agentes de RL, que requieren millones de interacciones por segundo, ralentizando el ciclo de innovación y validación de nuevas estrategias de control inteligente.

1.3. Propuesta de Solución

Para abordar esta brecha, este trabajo presenta el diseño, implementación y validación de *PonLab*, un simulador de redes PON escrito en Python y diseñado específicamente para la investigación y optimización de algoritmos DBA mediante Aprendizaje por Refuerzo. La solución se articula en torno a una arquitectura híbrida que reduce drásticamente el overhead computacional de la simulación, manteniendo al mismo tiempo la precisión.

La plataforma integra de forma nativa el simulador con frameworks de RL como Stable-Baselines3, proporcionando un entorno compatible con el estándar Gymnasium. Esto permite a los investigadores diseñar, entrenar y evaluar agentes de RL para la optimización de DBA de forma rápida y reproducible, comparándolos con algoritmos tradicionales implementados en la misma plataforma. El objetivo es ofrecer una herramienta de código abierto que acelere la innovación en la gestión de recursos para las redes ópticas de próxima generación.

1.4. Hipótesis

La investigación se articula en torno a una hipótesis central que postula la viabilidad y superioridad de un enfoque integrado para la optimización de redes PON. Esta hipó-

tesis global se desglosa en cuatro sub-hipótesis específicas y medibles (H1 a H4) que serán validadas experimentalmente.

El diseño de una plataforma de simulación de redes ópticas pasivas especializada, que integra de forma nativa un motor de eventos discretos de alto rendimiento con un framework de Aprendizaje por Refuerzo , permite la formulación y validación de algoritmos de Asignación Dinámica de Ancho de Banda adaptativos que superan en rendimiento y flexibilidad a los métodos tradicionales, al tiempo que reduce significativamente la complejidad y el costo computacional asociados a la investigación en esta área.

Para validar esta afirmación, el trabajo se sustenta en las siguientes hipótesis específicas, cuyos resultados se presentan en el Capítulo 6:

- **H1 - Eficiencia Arquitectónica**: Un simulador de redes PON con arquitectura de eventos discretos híbrida, que elimine los eventos de polling mediante verificación temporal automática (*event-less polling*), reducirá el overhead computacional en al menos un 70 % en comparación con enfoques tradicionales, sin sacrificar la precisión de las simulaciones.

- **H2 - Desempeño Superior del DBA-RL**: Un agente de Aprendizaje por Refuerzo, entrenado sobre la plataforma, aprenderá una política de DBA que demuestra un desempeño superior o comparable al de los mejores algoritmos tradicionales (IPACT, GIANT), mostrando un mejor equilibrio entre métricas clave como latencia, throughput y equidad, especialmente en escenarios de mayor complejidad.

- **H3 - Optimización Multiobjetivo**: Una función de recompensa ponderada que balancee múltiples objetivos de red (utilización, satisfacción de demanda, equidad, retardo y gestión de búfer) guiará al agente de RL a aprender una política más robusta y generalizable que la obtenida mediante la optimización de una única métrica.

- **H4 - Facilitación de la Investigación**: La plataforma unificada, con su interfaz gráfica y configuración sin código, reduce la barrera de entrada para la experimentación en DBA-RL, permitiendo a los investigadores diseñar, entrenar y comparar algoritmos de forma rápida y reproducible.

La validación de estas cuatro hipótesis en conjunto demostrará la validez de la hipótesis central del trabajo.

1.5. Objetivos

1.5.1. Objetivo General

El objetivo general de este trabajo de memoria es diseñar, implementar y validar la integración de un módulo de Aprendizaje por Refuerzo en un simulador de redes ópticas pasivas (PON), y construir una lógica central (core) robusta y modular para el simulador, permitiendo la experimentación y optimización de algoritmos de Asignación Dinámica de Ancho de Banda (DBA) bajo diferentes escenarios de tráfico y condiciones de red.

1.5.2. Objetivos Específicos

- Desarrollar y consolidar la arquitectura de la lógica central (core) del simulador PON, asegurando su modularidad, extensibilidad y rendimiento para la simulación precisa de los componentes y protocolos de la red.

- Integrar un módulo de Aprendizaje por Refuerzo que permita a los agentes inteligentes interactuar con el entorno del simulador, recolectar estados, ejecutar acciones y recibir recompensas para la optimización adaptativa de las estrategias de Asignación Dinámica de Ancho de Banda (DBA).

- Implementar y evaluar algoritmos DBA avanzados, incluyendo aquellos basados en RL, dentro del marco del simulador, comparando su desempeño con algorit-

mos DBA tradicionales en términos de eficiencia de ancho de banda, latencia y equidad.

- Documentar detalladamente el diseño, la implementación y los resultados obtenidos, contribuyendo a la base de conocimiento sobre la aplicación de técnicas de Aprendizaje por Refuerzo en la gestión de recursos de redes PON.

1.6. Estructura

Este documento se estructura de la siguiente forma:

- En el **Capítulo 1: Introducción**, se presenta el contexto, la motivación y los objetivos generales y específicos del trabajo de memoria, así como la estructura del documento.
- El **Capítulo 2: Marco Teórico** provee los fundamentos conceptuales y tecnológicos necesarios para comprender el funcionamiento de las redes PON, los algoritmos DBA y los principios del Aprendizaje por Refuerzo.
- El **Capítulo 3: Estado del Arte** revisa la literatura existente y los trabajos previos relacionados con la simulación de redes PON y la aplicación de RL en la gestión de recursos de red.
- El **Capítulo 4: Modelo del Simulador e Integración RL** detalla el diseño y la implementación de la lógica central (core) del simulador, incluyendo sus componentes clave y la arquitectura modular. Se describe exhaustivamente la integración del módulo de Aprendizaje por Refuerzo, sus interfaces y su interacción con el simulador.
- El **Capítulo 5: Diseño Experimental y Escenarios de Prueba** (Nota: no está listado en main.tex, pero es común en memorias) se definirán los escenarios de simulación, las métricas de rendimiento y la metodología utilizada para evaluar los algoritmos DBA y el módulo RL.

- El **Capítulo 6: Resultados y Análisis** presenta los resultados de las simulaciones y la evaluación de los algoritmos implementados, analizando su desempeño y comparándolos bajo diversas condiciones.
- Finalmente, el **Capítulo 7: Conclusiones y Trabajo Futuro** resume las principales aportaciones del trabajo, las conclusiones obtenidas y plantea posibles líneas de investigación y desarrollo futuras.

Capítulo 2

Marco Teórico

Este capítulo establece las bases teóricas necesarias para comprender el contexto del presente trabajo de memoria. Se abordan los conceptos fundamentales de las Redes Ópticas Pasivas (PON), los mecanismos de Asignación Dinámica de Ancho de Banda (DBA) en estas redes, los principios del Aprendizaje por Refuerzo (RL) y la importancia de los simuladores de redes de eventos discretos.

2.1. Introducción a las Redes Ópticas Pasivas (PON)

Las Redes Ópticas Pasivas (Passive Optical Networks, PON) representan una tecnología fundamental para el despliegue de redes de acceso de banda ancha, especialmente en configuraciones de Fibra hasta el Hogar (Fiber-to-the-Home, FTTH) y Fibra hasta el Edificio (Fiber-to-the-Building, FTTB). Su principal característica es el uso de divisores ópticos pasivos (splitters) que distribuyen una única fibra óptica desde la Oficina Central (Central Office, CO) a múltiples suscriptores, eliminando la necesidad de componentes electrónicos activos entre la OLT (Optical Line Terminal) y las ONUs (Optical Network Units). Esto reduce significativamente los costes de infraestructura y mantenimiento.

2.1.1. Componentes Principales de una Red PON

Una red PON típica consta de tres componentes principales:

- **Optical Line Terminal (OLT):** Ubicada en la oficina central del proveedor de servicios, es el punto final de la red troncal que agrega el tráfico de datos y lo envía hacia las ONUs, y recibe el tráfico de upstream de estas.
- **Divisor Óptico Pasivo (Optical Splitter):** Un componente pasivo que divide la señal óptica de una sola fibra en múltiples rutas para llegar a varias ONUs, y a la inversa, combina las señales de múltiples ONUs hacia la OLT.
- **Optical Network Unit (ONU):** Ubicada cerca o en las instalaciones del suscriptor, convierte las señales ópticas en señales eléctricas para los dispositivos del usuario y, a su vez, convierte las señales eléctricas de los usuarios en ópticas para su transmisión a la OLT.

2.1.2. Tipos de Redes PON

Existen varias generaciones de PON, cada una ofreciendo mayores velocidades y capacidades. Las más comunes incluyen:

- **GPON (Gigabit Passive Optical Network):** Estándar ampliamente adoptado que ofrece velocidades asimétricas (típicamente 2.5 Gbps downstream y 1.25 Gbps upstream).
- **EPON (Ethernet Passive Optical Network):** Basado en el estándar Ethernet, ofrece velocidades simétricas (típicamente 1.25 Gbps tanto downstream como upstream).
- **XGS-PON:** Una evolución de GPON que proporciona velocidades simétricas de 10 Gbps.

El canal de upstream es un medio compartido, lo que implica la necesidad de mecanismos eficientes para gestionar el acceso de las ONUs al canal sin colisiones, una tarea

que recae en los algoritmos DBA .

2.2. Asignación Dinámica de Ancho de Banda (DBA) en Redes PON

La Asignación Dinámica de Ancho de Banda (Dynamic Bandwidth Allocation, DBA) es un mecanismo esencial en las redes PON para gestionar de manera eficiente el acceso de las ONUs al canal de upstream, que es un medio compartido. Su principal objetivo es evitar colisiones y garantizar que el ancho de banda disponible se distribuya de forma justa y eficiente entre las ONUs, de acuerdo con sus necesidades y las políticas de calidad de servicio (QoS) establecidas.

2.2.1. Funcionamiento Básico de un Algoritmo DBA

El proceso de DBA opera en ciclos de tiempo definidos y se basa en un intercambio de mensajes entre las ONUs y la OLT:

1. **Reportes (Reports):** En cada ciclo, las ONUs informan a la OLT sobre la cantidad de datos que tienen en sus búferes y que desean transmitir.
2. **Cálculo de Concesiones (Grants):** La OLT recibe los reportes y, basándose en el algoritmo DBA implementado, calcula cuánto ancho de banda se le asignará a cada ONU para el siguiente ciclo de upstream.
3. **Concesión (Grant):** La OLT envía un mensaje de Grant a cada ONU, indicándoles el momento y la duración de su ventana de transmisión.
4. **Transmisión:** Las ONUs transmiten sus datos durante los intervalos de tiempo asignados, evitando así colisiones en el canal compartido.

2.2.2. Algoritmos DBA Tradicionales

Existen diversos algoritmos DBA tradicionales, cada uno con sus propias ventajas y desventajas:

- **IPACT (Interleaved Polling with Adaptive Cycle Time):** Uno de los algoritmos más conocidos, que ajusta dinámicamente el tiempo del ciclo de transmisión para optimizar la utilización del ancho de banda y reducir la latencia [5].
- **GIANT (GigaPON Access Network Traffic Management):** Un algoritmo DBA más avanzado que busca garantizar anchos de banda mínimos y distribuir el excedente de manera justa.

Los algoritmos DBA tradicionales a menudo requieren un ajuste manual de parámetros y pueden tener dificultades para adaptarse óptimamente a los patrones de tráfico altamente dinámicos y a los requisitos de QoS estrictos de las aplicaciones modernas, como las de 5G.

2.3. Fundamentos de Aprendizaje por Refuerzo (RL)

El Aprendizaje por Refuerzo (*Reinforcement Learning*, RL) es un paradigma de *Machine Learning* donde un agente aprende a vincular situaciones con acciones para maximizar una recompensa numérica. Al agente no se le indica qué acciones debe tomar; por el contrario, debe descubrir por su cuenta cuáles son las que garantizan mayores recompensas. En los casos más desafiantes e interesantes, las acciones no solo afectan a la recompensa inmediata, sino también a las siguientes y a todas las futuras [6].

Una característica fundamental de este paradigma es el dilema de explotación y exploración: el agente prefiere repetir las acciones que ya sabe que generan recompensas **(explotación)**, pero para descubrir dichas acciones, está obligado a probar opciones nuevas que podrían fallar **(exploración)** [6].

El proceso comienza con un agente interactivo que posee una meta clara; este es capaz de percibir el estado del entorno y determinar, de forma cuantificable, qué tan favorable o perjudicial resultó una acción específica.

Finalmente, una de las ideas más potentes del RL es que no se enfoca en subproblemas aislados, sino que ve el sistema como un todo interactuando en un ambiente

incierto. A diferencia de otros paradigmas centrados solo en la planificación teórica o la categorización de datos, el aprendizaje por refuerzo aborda el problema completo de la toma de decisiones en tiempo real [6].

2.3.1. Proceso de Decisión de Márkov Finito (MDP)

El Proceso de Decisión de Márkov (MDP) es un marco de trabajo matemáticamente idealizado para describir el problema de aprendizaje a partir de la interacción para alcanzar un objetivo. El encargado de aprender y tomar las decisiones se llama **agente** (*agent*), mientras que todo aquello con lo que interactúa, y que comprende todo lo que es externo a él, se llama **entorno** (*environment*). Estos interactúan continuamente en una secuencia de pasos de tiempo discretos: en cada paso t, el agente recibe una representación del estado del **entorno** s_t y selecciona una **acción** a_t; como consecuencia, en el siguiente paso recibe una señal numérica llamada **recompensa** (*reward*, r_{t+1}) y se encuentra en un nuevo estado s_{t+1} [6].

Como se ilustra en la **Figura 2.1**, esta relación se establece como un ciclo cerrado de retroalimentación. El diagrama permite visualizar cómo el agente influye en el entorno mediante sus acciones, y cómo este responde entregando información sobre el nuevo estado y una recompensa que cuantifica el éxito de la decisión tomada, permitiendo el aprendizaje iterativo.

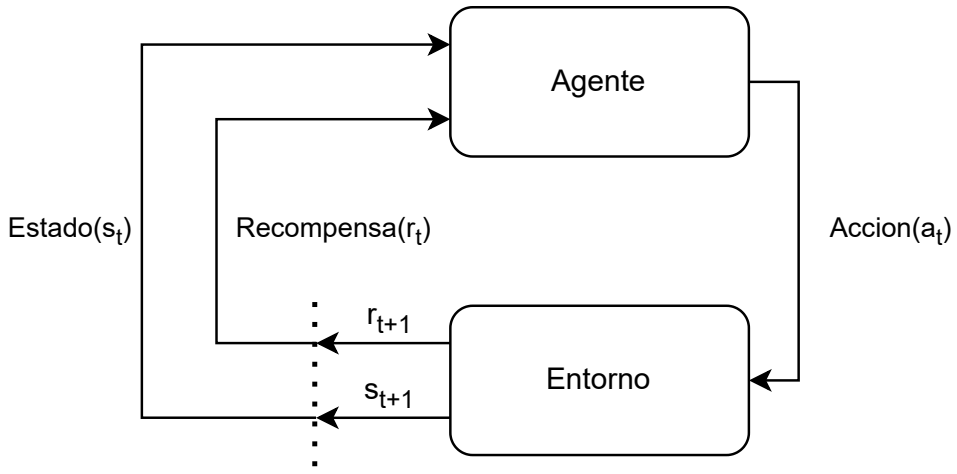


Figura 2.1: La interacción agente-entorno en el proceso de decisión de Markov. Fuente: Elaboración propia basada en Sutton y Barto [6]

En este marco, el propósito del agente se formaliza mediante la señal de recompensa mencionada. Fundamentalmente, el uso de esta señal se basa en la **hipótesis de la recompensa**, la cual establece que cualquier objetivo del agente puede plantearse como la maximización del valor esperado de la suma acumulada de estas señales escalares recibidas a lo largo del tiempo.

Esta suma acumulada se define formalmente como **retorno** (G_t). Para garantizar que esta suma sea finita en tareas continuas, se introduce el concepto de **descuento** (γ), un parámetro $0 \leq \gamma \leq 1$ que determina el valor presente de las recompensas futuras. Así, el retorno esperado se define como:

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{2.1}$$

Para navegar el entorno, el agente sigue una **política** (π), que es un mapeo de probabilidades donde $\pi(a|s)$ es la probabilidad de seleccionar la acción a dado el estado s. Para evaluar qué tan buena es una política, se utilizan las **funciones de valor**. Específicamente, la función de valor de acción (q_π), estima el retorno esperado al tomar una acción a en un estado s y seguir la política π de ahí en adelante:

$$q_\pi(s,a) \doteq \mathbb{E}_\pi \left[G_t \mid S_t = s, A_t = a \right] \tag{2.2}$$

Finalmente, el objetivo último de un algoritmo de aprendizaje por refuerzo es encontrar una **política óptima** (π_*) que maximice el retorno esperado. Esta política óptima comparte la misma función de valor óptima, denotada como q_*, la cual satisface la **ecuación de optimalidad de Bellman**. Esta ecuación expresa que el valor de una acción bajo una política óptima debe ser igual al retorno esperado de la mejor acción posible en el estado siguiente:

$$q_*(s,a) = \mathbb{E}\left[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') \mid S_t = s, A_t = a \right] \tag{2.3}$$

Esta identidad recursiva es la base teórica fundamental para métodos de solución como Q-learning y SARSA [6].

2.3.2. Relevancia en Redes de Comunicación

En este contexto, la aplicación del Aprendizaje por Refuerzo resulta fundamental para la asignación dinámica de ancho de banda en el canal de subida de una red PON. El RL ofrece una solución robusta para la optimización del canal frente a patrones de tráfico dinámicos e impredecibles; esto permite que los agentes aprendan políticas eficientes de gestión de recursos, eliminando la necesidad de depender de modelos matemáticos explícitos del entorno.

2.4. Simuladores de Redes de Eventos Discretos

Los simuladores de redes son herramientas indispensables para la investigación y el desarrollo de nuevos protocolos, algoritmos y arquitecturas de red. Permiten evaluar el rendimiento y el comportamiento de sistemas complejos sin la necesidad de desplegar hardware real, lo que ahorra tiempo y costes. Los simuladores de eventos discretos son una categoría prominente.

2.4.1. Funcionamiento de un Simulador de Eventos Discretos

En un simulador de eventos discretos, el tiempo avanza de forma discontinua, saltando de un evento al siguiente. Los elementos clave son:

- **Evento:** Una ocurrencia en un punto específico en el tiempo que puede cambiar el estado del sistema.
- **Cola de Eventos:** Una estructura de datos que almacena todos los eventos futuros programados, ordenados por su marca de tiempo.
- **Motor de Simulación:** Extrae eventos de la cola en orden cronológico, avanza el tiempo de simulación al tiempo del evento y ejecuta las acciones asociadas al evento, lo que puede a su vez programar nuevos eventos.

2.4.2. Simuladores Populares y la Necesidad de Simuladores Personalizados

Existen varios simuladores de eventos discretos ampliamente utilizados en la comunidad académica y la industria:

- **NS-2 y NS-3:** Simuladores de código abierto muy potentes y extendidos. NS-3 es la evolución de NS-2, con una arquitectura más moderna basada en C++ y Python [7, 8].
- **OMNeT++:** Un simulador modular basado en componentes, también de código abierto, con una interfaz gráfica robusta para visualización y análisis [9].
- **OPNET:** Una herramienta comercial que ofrece un entorno de desarrollo completo para la modelación de redes [10].

A pesar de la existencia de estas herramientas maduras, la construcción de simuladores personalizados, como el desarrollado en este trabajo, ofrece ventajas significativas. Permite una **flexibilidad total** para integrar lógicas específicas de los algoritmos (como el DBA para PON), realizar **optimizaciones de rendimiento** a medida (como el

sondeo sin eventos) y, crucialmente para este proyecto, una **integración profunda y eficiente con módulos de Aprendizaje por Refuerzo**, algo que a menudo es complejo y limitado en simuladores genéricos.

Capítulo 3

Estado del Arte

La asignación dinámica de ancho de banda (DBA) en redes ópticas pasivas (PON) es un problema fundamental en telecomunicaciones que ha evolucionado desde soluciones determinísticas hacia enfoques adaptativos basados en aprendizaje por refuerzo (RL). Este capítulo revisa las soluciones existentes, los fundamentos teóricos relevantes y justifica las decisiones tecnológicas adoptadas en el desarrollo del simulador con integración nativa de RL.

3.1. Soluciones Existentes

En esta sección, se examinan las soluciones existentes en tres áreas clave: los simuladores de redes PON disponibles, los algoritmos de DBA tradicionales que forman la base de comparación, y las aplicaciones emergentes de Aprendizaje por Refuerzo en este dominio.

3.1.1. Simuladores de Redes PON

En el mercado académico e industrial existen diversas herramientas de simulación de redes PON, cada una con diferentes enfoques y capacidades:

- **NS-3 (Network Simulator 3)[7]:** simulador de eventos discretos de código abierto ampliamente utilizado en investigación académica. Ofrece modelos detallados

de protocolos de red y soporta simulaciones de gran escala. Sin embargo, presenta desventajas significativas:

  - Integración compleja con frameworks modernos de RL como Stable-Baselines3 o RLlib
  - Sobrecarga computacional por su arquitectura de eventos discretos tradicional
  - Curva de aprendizaje pronunciada y documentación fragmentada para casos especializados
  - Requiere compilación en C++ para modificaciones al núcleo del simulador

- **OMNeT++ [9]:** framework modular de simulación con interfaz gráfica avanzada. Popular en investigación de redes de comunicación, pero comparte limitaciones similares a NS-3:

  - Dificultad para integrar con ecosistema Python de machine learning
  - Overhead excesivo para simulaciones de RL que requieren millones de iteraciones
  - Arquitectura orientada a validación de protocolos, no a optimización con RL

- **OPNET (ahora Riverbed Modeler)[10]:** simulador comercial de alto rendimiento con soporte profesional. Ofrece modelos validados industrialmente, pero:

  - Costo de licenciamiento prohibitivo para investigación académica
  - Sistema propietario cerrado que dificulta modificaciones profundas
  - No diseñado para integración directa con algoritmos de aprendizaje automático

La principal limitación de estos simuladores es su arquitectura de eventos discretos tradicional, donde cada transmisión de paquete genera un evento que debe ser encolado, ordenado y procesado. Para entrenamiento de RL que puede requerir 10^6 a 10^8 pasos de

simulación, esta sobrecarga se vuelve crítica. Adicionalmente, ninguno de estos simuladores fue diseñado con integración nativa de RL ni optimizaciones específicas para reducir el overhead de simulación durante entrenamiento.

3.1.2. Algoritmos DBA Tradicionales

Los algoritmos de asignación dinámica de ancho de banda en redes PON han evolucionado desde enfoques determinísticos hacia soluciones más sofisticadas:

- **Fixed Bandwidth Allocation:** asignación estática sin adaptación a demanda. Simple pero ineficiente bajo carga variable.
- **IPACT (Interleaved Polling with Adaptive Cycle Time)[5]:** algoritmo seminal propuesto por Kramer et al. Utiliza polling cíclico con predicción de demanda. Variantes incluyen:
  - IPACT-Limited: limita concesión máxima por ONU
  - IPACT-Gated: asigna exactamente lo solicitado
  - IPACT-Excess: redistribuye ancho de banda no utilizado
- **GIANT (Greedy Interleaved Allocation with Nested Time slots):** extensión de IPACT que maximiza utilización del canal mediante scheduling agresivo.

Estos algoritmos tradicionales son efectivos bajo patrones de tráfico estacionarios, pero presentan rigidez ante variaciones dinámicas. No aprenden de patrones históricos ni se adaptan automáticamente a cambios en la topología o comportamiento de usuarios.

3.1.3. Aplicaciones de RL en Redes PON

La aplicación de aprendizaje por refuerzo a problemas de DBA es un área de investigación emergente. Estudios que algoritmos RL pueden superar soluciones tradicionales en escenarios de tráfico complejo. Sin embargo, la mayoría de trabajos publicados

utilizan simuladores no especializados o implementaciones ad-hoc que dificultan la reproducibilidad y extensión de resultados.

Las principales barreras identificadas en la literatura son:

- **Costo computacional:** entrenar agentes RL requiere millones de iteraciones
- **Diseño de funciones de recompensa:** balancear múltiples objetivos (latencia, throughput, fairness) es no trivial
- **Validación realista:** muchos trabajos evalúan en entornos excesivamente simplificados
- **Reproducibilidad:** falta de código fuente y especificaciones detalladas de simuladores

Este trabajo aborda estas limitaciones mediante un simulador especializado con event-less polling para reducir overhead computacional, integración nativa con Gymnasium/Stable-Baselines3, y diseño de recompensas multi-objetivo explícito.

3.1.4. Investigaciones Recientes en DBA y RL/DL para PON

La aplicación de técnicas avanzadas de Aprendizaje por Refuerzo (RL) y Aprendizaje Profundo (DL) en la asignación dinámica de ancho de banda (DBA) para redes PON es un área de intensa investigación. A continuación, se resumen algunas contribuciones clave que exploran diferentes facetas de esta intersección:

3.1.4.1. Asignación Inteligente de Ancho de Banda para Gestión de Latencia

Zhou et al. (2020) en [11] proponen un esquema novedoso de asignación inteligente de ancho de banda utilizando Aprendizaje por Refuerzo para la gestión de la latencia en redes NG-EPON. Su enfoque busca optimizar la latencia bajo cargas de tráfico fijas y dinámicas, logrando una reducción significativa de hasta un 32.9 % en la latencia de la red. Esta investigación es particularmente relevante al abordar uno de los desafíos críticos de las redes de próxima generación, la minimización de la latencia, a través de la inteligencia artificial.

3.1.4.2. Aprendizaje Federado en Redes Ópticas Pasivas de Próxima Generación

Ciceri et al. (2021) en [12] exploran el Aprendizaje Federado (FL) sobre redes ópticas pasivas de próxima generación (TWDM-PONs). Proponen algoritmos de Asignación Dinámica de Longitud de Onda y Ancho de Banda (DWBA) diseñados para proporcionar QoS al tráfico de FL. Aunque no utilizan directamente RL para el DBA, este trabajo subraya la creciente demanda de mecanismos DBA avanzados para soportar aplicaciones emergentes de aprendizaje automático distribuidas, donde la latencia y el ancho de banda son requisitos críticos que el RL puede ayudar a optimizar.

3.1.4.3. Asignación de Ancho de Banda Basada en Aprendizaje Profundo

Hatem et al. (2019) en [13] presentan un enfoque de DBA basado en aprendizaje profundo para redes de acceso ópticas futuras. Su propuesta utiliza el aprendizaje profundo para predecir la demanda de ancho de banda de los usuarios finales, con el objetivo de reducir la sobrecarga de control asociada al mecanismo de solicitud-concesión en NG-EPON. Esto no solo aumenta la utilización del ancho de banda, sino que también ofrece una perspectiva prometedora para optimizar la eficiencia y reducir el overhead en los mecanismos de DBA, un objetivo compartido con el presente trabajo.

3.2. Herramientas y Tecnologías Seleccionadas

La selección del stack tecnológico se basó en criterios de eficiencia computacional, compatibilidad con frameworks de RL modernos, facilidad de extensión y reproducibilidad científica.

3.2.1. Lenguaje de Implementación: Python

Se eligió **Python**[14] como lenguaje principal de implementación por las siguientes razones fundamentales:

- **Ecosistema de RL:** Python es el lenguaje dominante en machine learning y RL.

Frameworks como Stable-Baselines3, RLlib, TensorFlow[15] y PyTorch[16] están diseñados nativamente para Python.

- **Bibliotecas científicas:** NumPy[17] y Pandas[18] permiten operaciones vectorizadas eficientes sobre datos de simulación, alcanzando rendimientos cercanos a C mediante implementaciones en BLAS/LAPACK.
- **Prototipado rápido:** sintaxis clara y alto nivel facilitan iteración rápida sobre diseños de recompensa, espacios de observación y arquitecturas de red.
- **Reproducibilidad:** código Python es más legible y autodocumentado que C++, facilitando publicación y revisión de resultados.

¿Por qué no C++ o Julia?

- **C++:** aunque más rápido, la complejidad de integración con frameworks RL modernos (que son Python-first) introduce overhead de desarrollo significativo. La ganancia de velocidad se pierde en el esfuerzo de implementación de bindings.
- **Julia:** lenguaje emergente con buen rendimiento, pero ecosistema de RL inmaduro comparado con Python. Bibliotecas como Flux.jl no tienen paridad de características con Stable-Baselines3.

3.2.2. Framework de Aprendizaje por Refuerzo: Stable-Baselines3

Para la implementación de algoritmos RL se seleccionó **Stable-Baselines3 (SB3)**[19], una biblioteca de alto nivel que proporciona implementaciones confiables y validadas de algoritmos state-of-the-art.

Ventajas de Stable-Baselines3:

- **Implementaciones validadas:** algoritmos como PPO, SAC, TD3 están exhaustivamente testeados y documentados
- **API consistente:** interfaz uniforme para todos los algoritmos facilita experimentación

- **Integración con Gymnasium:** compatibilidad nativa con el estándar de facto para entornos RL
- **Callbacks y logging:** sistema extensible de callbacks permite monitoreo detallado de entrenamiento
- **Documentación completa:** ejemplos, tutoriales y guías de mejores prácticas

Comparación con alternativas:

- **RLlib (Ray)[20]:** framework más complejo orientado a computación distribuida. Overkill para simulaciones single-machine y con mayor curva de aprendizaje.
- **TF-Agents[21]:** atado a TensorFlow, menos flexible que SB3 que soporta PyTorch. Documentación menos accesible.
- **Implementación propia:** mayor control pero alto riesgo de bugs sutiles en algoritmos complejos como PPO. No justificable para investigación aplicada.

SB3 proporciona el balance óptimo entre facilidad de uso, confiabilidad y flexibilidad para el caso de estudio.

3.2.3. Interfaz de Entornos: Gymnasium

Gymnasium[22] es el sucesor mantenido de OpenAI Gym, estableciendo el estándar para entornos de RL en Python. El entorno de RL desarrollado (RealPonEnv) hereda de `gymnasium.Env`, garantizando compatibilidad con cualquier algoritmo SB3 y facilitando futura extensión a otros frameworks. Gracias al event-less polling, este entorno realista es suficientemente eficiente para ser usado directamente en entrenamiento.

3.2.4. Interfaz Gráfica: PyQt5

Para la interfaz gráfica se eligió **PyQt5**[23], binding de Python para el framework Qt. Esta decisión se fundamenta en:

- **Madurez y estabilidad:** Qt es un framework industrial con más de 25 años de desarrollo
- **Canvas interactivo:** `QGraphicsView` permite implementar topologías de red arrastrables con renderizado eficiente
- **Multithreading:** sistema de signals/slots facilita integración de entrenamiento asíncrono sin bloquear UI
- **Multiplataforma:** aplicación corre sin modificaciones en Windows, Linux y macOS
- **Integración con Matplotlib:** widgets nativos para embeber gráficos científicos

Alternativas como Tkinter (limitado en capacidades gráficas) añadirían complejidad innecesaria para una aplicación de escritorio científica.

3.2.5. Simulador Personalizado: Arquitectura Híbrida Sin Eventos de polling

La decisión más significativa del proyecto fue desarrollar un **simulador personalizado con arquitectura híbrida sin eventos para polling**, en lugar de adaptar NS-3 u OMNeT++.

Justificación técnica:

Polling sin eventos:

En simuladores tradicionales, cada transmisión de GATE/REPORT genera eventos discretos que deben ser encolados en una priority queue. Para una red con N ONUs y frecuencia de polling de 1 kHz, esto genera $2N \times 1000$ eventos/segundo solo para coordinación.

El simulador desarrollado elimina esta sobrecarga mediante **polling determinístico por reloj**:

- No se generan eventos para mensajes GATE/REPORT

- El algoritmo DBA se invoca directamente cada ciclo de polling
- Solo se generan eventos para transmisiones de datos reales

Integración nativa de RL:

El simulador está diseñado desde cero para RL:

- Espacios de observación/acción son ciudadanos de primera clase
- Función de recompensa es modular y configurable
- Exportación directa de métricas para análisis con Pandas

3.2.6. Visualización y Análisis: Matplotlib y Pandas

Para análisis de resultados se utilizan:

- **Matplotlib[24]:** biblioteca estándar para gráficos científicos en Python. Permite generar plots de latencia, throughput, fairness con calidad de publicación.
- **Pandas[18]:** manipulación de datos tabulares. Exportar métricas de simulación a DataFrames facilita análisis estadístico, agregación y comparación de algoritmos.

Ambas herramientas son ampliamente conocidas en comunidad científica, garantizando reproducibilidad de análisis.

En síntesis, el stack tecnológico seleccionado (Python + SB3 + Gymnasium + PyQt5 + simulador personalizado) está optimizado para el problema específico de investigación en DBA con RL, priorizando eficiencia computacional, facilidad de experimentación y reproducibilidad científica. Las decisiones de diseño, en particular la arquitectura híbrida sin eventos polling del simulador, abordan directamente las limitaciones identificadas en soluciones existentes y permiten validar las hipótesis planteadas en el Capítulo 1.

Capítulo 4

Modelo del Simulador e Integración de Aprendizaje por Refuerzo

Este capítulo detalla la arquitectura y el diseño del simulador de Redes Ópticas Pasivas (PON) desarrollado, con un enfoque particular en la lógica de su núcleo y la innovadora integración de un módulo de Aprendizaje por Refuerzo (RL). Se presentan las decisiones de diseño fundamentales que sustentan las hipótesis planteadas en el Capítulo 1, junto con la justificación técnica de cada componente arquitectónico.

En la **Figura 4.1** se presenta el flujo de trabajo integral de la plataforma. Se observa cómo el usuario define los parámetros de red que configuran la topología, la cual interactúa directamente con el módulo de SDN y algoritmos. Este motor de simulación genera métricas constantes que alimentan tanto a la interfaz visual como al proceso de entrenamiento reforzado, creando un ecosistema cerrado donde las políticas de DBA se actualizan dinámicamente basándose en el estado de la red.

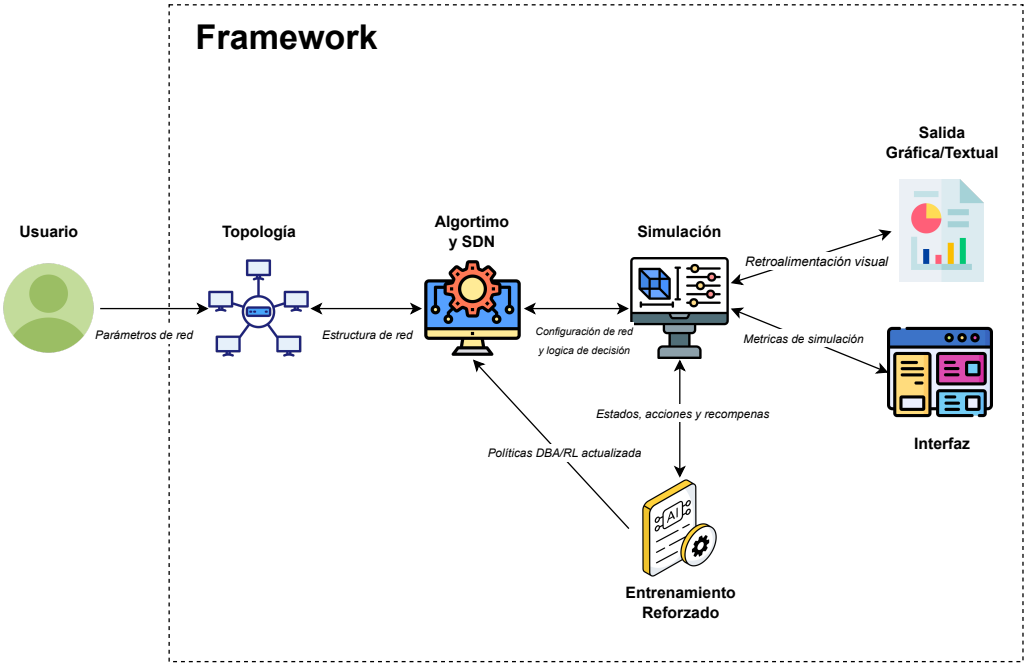


Figura 4.1: Arquitectura general de la plataforma PonLab.

4.1. Arquitectura del Núcleo del Simulador

El simulador se basa en una arquitectura de **eventos discretos híbrida**, que combina lo mejor de dos paradigmas de simulación. Mientras que eventos no periódicos como la generación de paquetes y las finalizaciones de transmisión se manejan mediante una cola de prioridad tradicional, el ciclo de sondeo (polling) periódico de la OLT se ejecuta automáticamente mediante verificación temporal.

4.1.1. Mecanismo de Sondeo Automático (*Event-less Polling*)

Una de las contribuciones más significativas a la arquitectura del simulador es el mecanismo de sondeo automático sin eventos explícitos. Esta innovación aborda directamente una limitación fundamental de los simuladores de eventos discretos tradicionales aplicados a redes PON.

4.1.1.1. Limitaciones del Enfoque Tradicional

En un enfoque de simulación por eventos discretos tradicional, cada ciclo de sondeo de 125 microsegundos (período estándar en EPON y GPON) se programa como un evento independiente en la cola principal del simulador. Para una simulación de tan solo 10 segundos de tiempo de red simulado, esto genera:

$$\text{Eventos de polling} = \frac{10 \text{ s}}{125 \times 10^{-6} \text{ s}} = 80{,}000 \text{ eventos}$$

Cada uno de estos eventos debe ser:

- Insertado en la cola de prioridad (operación $O(\log n)$ en el peor caso)
- Mantenido en memoria durante su ciclo de vida
- Extraído de la cola cuando llega su turno de procesamiento
- Procesado para ejecutar la lógica de sondeo

Este overhead de gestión de eventos se vuelve un cuello de botella significativo, especialmente en simulaciones de larga duración o con múltiples ONUs, donde el volumen de eventos crece linealmente con el tiempo simulado.

4.1.1.2. Solución Implementada: Verificación Temporal Automática

La solución propuesta elimina completamente los eventos de polling de la cola principal. En su lugar, antes de procesar cada evento restante (generación de paquetes, transmisiones completadas), el motor de simulación verifica si el tiempo transcurrido desde el último sondeo ha superado el umbral de 125 microsegundos. Si es así, ejecuta de forma sincrónica todos los ciclos de sondeo que deberían haber ocurrido en ese intervalo.

Este enfoque conserva las siguientes propiedades críticas:

- **Precisión Temporal Equivalente:** Los ciclos de sondeo se ejecutan en los instantes de tiempo correctos, garantizando que las métricas de red (latencia, throughput, equidad) sean idénticas a las que se obtendrían con el enfoque tradicional.

- **Determinismo:** El orden de ejecución de eventos es determinista y reproducible dado una semilla aleatoria fija, propiedad esencial para la validación científica.
- **Reducción Drástica de Overhead:** Elimina el 100 % de los eventos de polling de la cola principal, reduciendo tanto el uso de memoria como el tiempo de procesamiento dedicado a gestión de eventos.

4.1.1.3. Ventajas Cuantitativas Esperadas

Las ventajas de esta arquitectura híbrida serán validadas experimentalmente en el Capítulo 6 (Hipótesis H1). Se espera:

- Reducción de al menos 70 % en el tiempo de ejecución de simulaciones respecto a enfoques tradicionales
- Reducción proporcional en el uso de memoria al eliminar 80,000 nodos de la cola de eventos por cada 10 segundos simulados
- Escalabilidad mejorada para simulaciones de larga duración (horas de tiempo simulado) necesarias para entrenar agentes de RL

4.1.2. Componentes Principales del Simulador

La arquitectura del simulador se compone de componentes modulares cuya interacción se detalla en la **Figura 4.2**. Como se aprecia en el diagrama, el Motor de Simulación actúa como el orquestador central que coordina tanto a los modelos de OLT como de ONU. Es fundamental destacar el flujo de control: la OLT utiliza el módulo de algoritmo DBA para procesar las solicitudes y envía mensajes tipo GATE a las ONUs, mientras que estas responden enviando reportes de estado (REPORTS), cerrando el ciclo de comunicación necesario para la asignación de recursos.

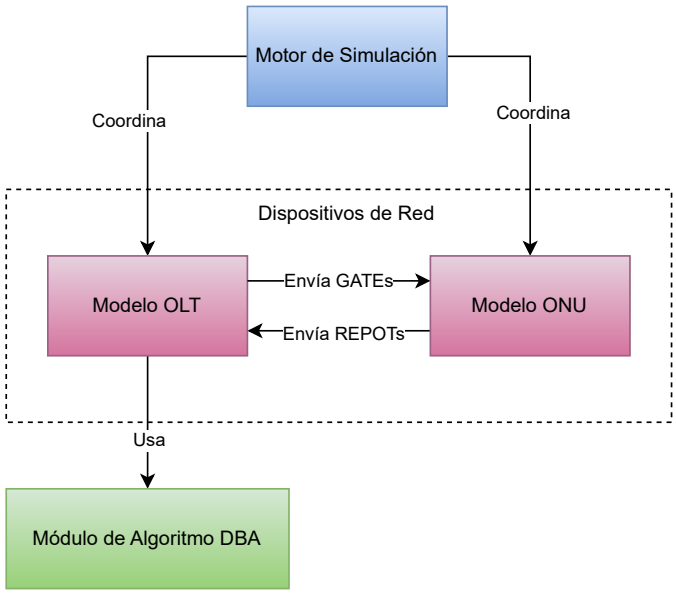


Figura 4.2: Diagrama de interacción entre los componentes principales del núcleo.

- **Motor de Simulación:** Componente central que orquesta el bucle principal de procesamiento de eventos. Es responsable de:
  - Gestionar la cola de eventos ordenada por timestamp
  - Avanzar el reloj de simulación de forma coherente
  - Coordinar la ejecución del sondeo automático mediante verificación temporal

  - Recolectar estadísticas y métricas de la simulación

- **Modelo de Dispositivos OLT:** Representa la Terminal de Línea Óptica, implementando:
  - Ejecución de algoritmos de Asignación Dinámica de Ancho de Banda (DBA)
  - Gestión de ciclos de polling y envío de mensajes GATE
  - Recepción de mensajes REPORT de las ONUs
  - Coordinación de slots temporales para evitar colisiones en transmisiones upstream
- **Modelo de Dispositivos ONU:** Representa las Unidades de Red Óptica, implementando:
  - Gestión de múltiples colas de transmisión por prioridad (T-CONTs 0-4 según estándar ITU-T G.984)
  - Generación de tráfico según procesos estocásticos (Poisson, bursty, trazas)
  - Envío de mensajes REPORT con estado de buffers
  - Transmisión de paquetes cuando se reciben GATE grants
- **Módulo de Algoritmos DBA:** Diseño modular que permite intercambiar fácilmente diferentes estrategias de asignación de ancho de banda. Se implementaron:
  - FCFS (First-Come-First-Served): Asignación por orden de llegada
  - Priority-based: Priorización estricta según T-CONT
  - IPACT (Interleaved Polling with Adaptive Cycle Time): En variantes Limited, Gated y Hybrid
  - GIANT: Algoritmo con garantías mínimas y distribución proporcional de excedente
  - Algoritmo basado en RL: Asignación controlada por agente de Aprendizaje por Refuerzo

Todos los componentes se diseñaron siguiendo el principio de separación de responsabilidades, donde cada módulo tiene una interfaz bien definida que facilita la extensión y el testing unitario.

4.2. Integración del Módulo de Aprendizaje por Refuerzo

Para permitir la optimización de la gestión de recursos mediante técnicas de Aprendizaje por Refuerzo, se diseñó un módulo de integración que conecta el simulador PON con frameworks de RL estándar de la industria. La integración sigue la especificación de entornos Gymnasium , garantizando compatibilidad con bibliotecas como Stable-Baselines3, RLlib y TensorFlow Agents.

Se utilizó como base el esquema de asignación de ancho de banda inteligente (IBA) propuesto por Zhou et al. [11], donde el agente se encuentra ubicado en la OLT. Para la implementación del aprendizaje por refuerzo, se estableció una arquitectura de dos niveles temporales: un ciclo "lento"(Intelligent Bandwidth Allocation, IBA), encargado de actualizar los parámetros de la política, y un ciclo rápido"(DBA), que utiliza dicha política actualizada para realizar la asignación de recursos en tiempo real.

Ambos ciclos mantienen una comunicación constante, como se ilustra en la **Figura 4.3**. El Agente IBA, que contiene el algoritmo de aprendizaje, recibe el estado (S_t) y la recompensa (R_t) provenientes del monitoreo de la red. Basándose en esto, el agente ejecuta una acción (A_t) que se traduce en una actualización de las políticas de la PON, permitiendo que el sistema se adapte de forma inteligente a las variaciones del tráfico detectadas.

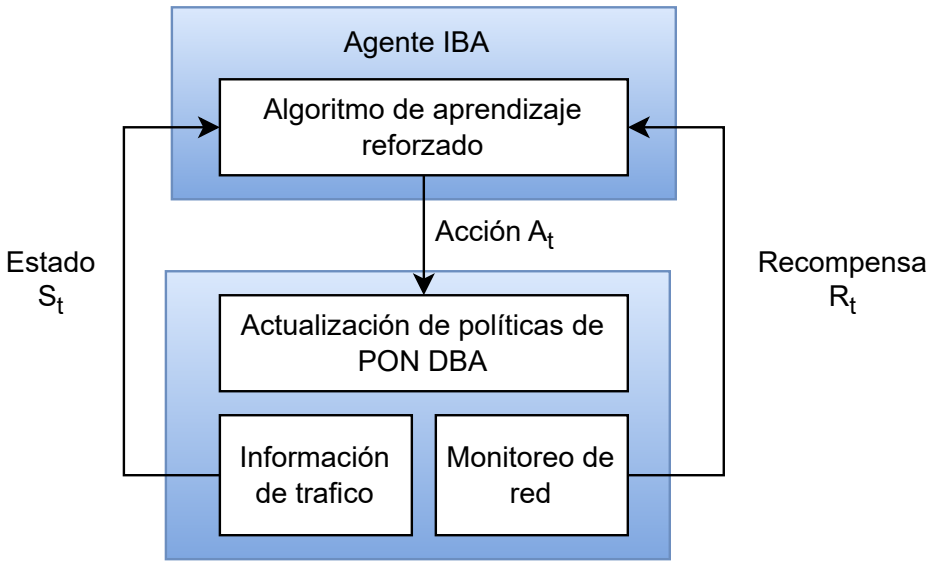


Figura 4.3: Arquitectura del sistema de asignación inteligente (IBA).
Fuente: Adaptado de Zhou et al.

El problema de asignación de ancho de banda se formuló matemáticamente como un Proceso de Decisión de Markov (MDP), definido por el espacio de estados, acciones y la función de recompensa implementados a través de la interfaz de Gymnasium.

Sin embargo, dada la complejidad y la naturaleza continua del espacio de estados en una red PON real, los métodos tabulares clásicos resultan insuficientes debido a la 'maldición de la dimensionalidad'. Por esta razón, se optó por un enfoque de Aprendizaje por Refuerzo Profundo (Deep Reinforcement Learning, DRL). Utilizando las librerías Stable-Baselines3 y TensorFlow, se implementaron agentes basados en redes neuronales (como PPO o DQN) capaces de generalizar y aprender políticas óptimas en entornos complejos sin necesidad de una discretización estricta de todas las variables.

La implementación descrita anteriormente funciona como un esqueleto modular, brindando al usuario final la flexibilidad de seleccionar el algoritmo de RL, los parámetros y la función de recompensa que estime convenientes para sus objetivos específicos.

4.2.1. Entorno de RL Realista

Una decisión arquitectónica fundamental fue utilizar el simulador completo de eventos discretos como entorno de entrenamiento de RL, aprovechando la optimización de event-less polling para mantener la eficiencia computacional. Esta decisión garantiza que las políticas aprendidas durante el entrenamiento sean directamente aplicables a redes PON reales, sin problemas de transferencia entre entornos.

4.2.1.1. Arquitectura del Entorno

El entorno RealPonEnv es un **entorno de alta fidelidad**, que encapsula el simulador de eventos discretos completo. Sus características principales son:

Simulación Detallada con Event-less Polling:

Cada paso del entorno avanza la simulación en un intervalo de tiempo fijo (típicamente 1 milisegundo), durante el cual ocurren múltiples ciclos de polling (cada 125 microsegundos), generaciones de paquetes, y transmisiones completadas. Gracias al event-less polling, estos ciclos se procesan eficientemente sin generar eventos explícitos en la cola principal, reduciendo el overhead computacional en más del 90 %.

Realismo Completo:

El entorno modela fielmente:

- Delays promedio de atencion
- Procesamiento de frames Ethernet
- Límites físicos de la red (bandwidth, buffer sizes)
- Overhead de mensajes GATE/REPORT
- Generación estocástica de tráfico según patrones configurables

Métricas de Alta Precisión:

Al utilizar el simulador completo, las métricas obtenidas (latencia paquete a paquete, throughput instantáneo, ocupación de buffers, índice de equidad de Jain) son directa-

mente comparables con las de algoritmos DBA tradicionales, permitiendo evaluaciones rigurosas tanto durante entrenamiento como en validación.

4.2.1.2. Justificación de Usar Entorno Realista para Entrenamiento

Tradicionalmente, el entrenamiento de RL en simulaciones de redes requiere simplificaciones para reducir el costo computacional. Sin embargo, gracias al event-less polling, el simulador completo es suficientemente eficiente para ser usado directamente en entrenamiento. Esta arquitectura unificada ofrece múltiples ventajas:

- **Eliminación del problema de transferencia**: Las políticas aprendidas son directamente aplicables a redes reales, sin necesidad de validar transferibilidad entre entornos
- **Mayor fidelidad durante el aprendizaje**: El agente aprende considerando todos los detalles y restricciones del sistema real, resultando en políticas más robustas
- **Arquitectura más simple**: Un solo entorno simplifica el desarrollo, mantenimiento y experimentación
- **Eficiencia práctica**: Gracias al event-less polling, el entrenamiento se completa en tiempos razonables (horas en lugar de días/semanas)

4.2.2. Espacios de Observación y Acción

Ambos entornos (entrenamiento y validación) comparten una definición común de espacios de observación y acción. En la **Figura 4.4** se desglosa la estructura de estos espacios. Se observa que el vector de observación concatena las solicitudes, retardos y ocupación de búfer de cada una de las N ONUs, sumando la utilización total para ofrecer una visión global al agente. Por otro lado, el espacio de acción se define como un vector de pesos de N dimensiones que determina la prioridad de asignación para cada unidad.

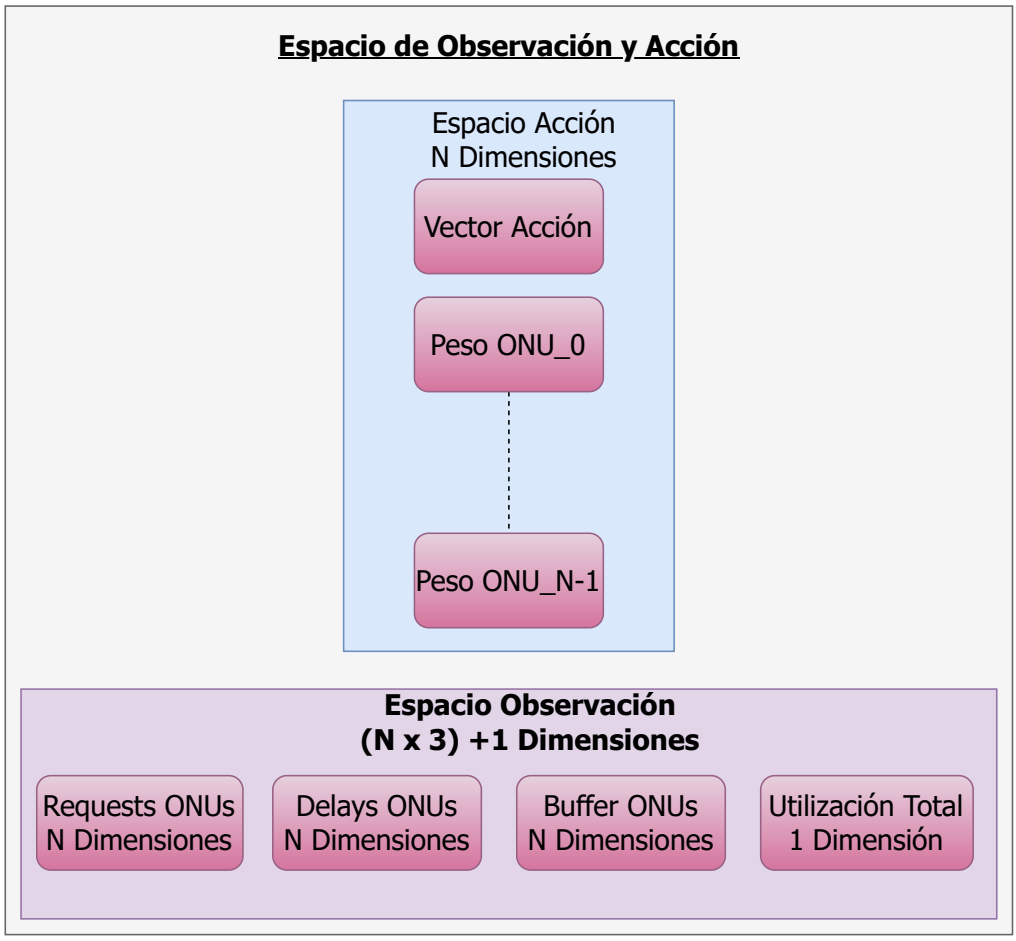


Figura 4.4: Diagrama de la estructura de los espacios de observación y acción.

4.2.2.1. Espacio de Observación

El espacio de observación es un vector de números reales de dimensión $(N \times 3) + 1$, donde N es el número de ONUs en la topología de red. Este vector se adapta automáticamente a diferentes configuraciones de red, permitiendo entrenar agentes que generalizan a topologías de distinto tamaño.

El vector de observación se compone de:

1. **Solicitudes de Ancho de Banda Normalizadas (N valores):** Cada elemento $r_i \in [0,1]$ representa la demanda de ancho de banda de la ONU i, normalizada

respecto al ancho de banda total del canal. Esta componente proporciona la *señal de demanda*, indicando al agente qué ONUs requieren recursos.

2. **Retardos Promedio Normalizados (N valores):** Cada elemento $d_i \in [0,1]$ representa la latencia media de los paquetes encolados en la ONU i, normalizada respecto a un umbral de latencia máximo aceptable. Esta componente es una métrica directa de Calidad de Servicio (QoS), señalando al agente qué ONUs están experimentando retardos altos.

3. **Ocupación de Búferes Normalizada (N valores):** Cada elemento $b_i \in [0,1]$ indica el porcentaje de ocupación del búfer de la ONU i. Esta componente proporciona una *señal de congestión*, alertando al agente sobre ONUs cercanas a experimentar pérdidas por desbordamiento de búfer.

4. **Utilización Total del Canal (1 valor):** Un escalar $u \in [0,1]$ que agrega la demanda total de todas las ONUs respecto a la capacidad total del canal. Esta componente da al agente una visión global de la carga de la red, complementando la información por-ONU.

Todos los valores se normalizan al rango $[0,1]$ para facilitar el entrenamiento de redes neuronales, evitando problemas de gradientes que surgen con valores no acotados o de escalas muy dispares.

4.2.2.2. Espacio de Acción

El espacio de acción es un vector continuo de N números reales, donde cada elemento $a_i \in [0,1]$ representa el *peso* o *prioridad* que el agente asigna a la ONU i.

Normalización e Interpretación:

Antes de aplicar la acción, los pesos se normalizan para que su suma sea 1:

$$a'_i = \frac{a_i}{\sum_{j=1}^{N} a_j}$$

Estos pesos normalizados se utilizan para repartir proporcionalmente el ancho de banda total del canal:

$$\mathrm{BW}_i = a'_i \times \mathrm{BW}_{\mathrm{total}}$$

Finalmente, cada asignación se limita por la demanda real de la ONU para evitar asignaciones desperdiciadas:

$$\mathrm{BW}_{\mathrm{asignado},i} = \min(\mathrm{BW}_i, \mathrm{BW}_{\mathrm{solicitado},i})$$

Justificación del Diseño:

Este diseño de espacio de acción continuo permite al agente expresar preferencias sutiles y graduales entre ONUs, a diferencia de esquemas de acción discreta que limitarían la granularidad de las decisiones. La normalización automática garantiza que el agente no necesite aprender la restricción de suma de recursos (que siempre es implícita), simplificando el espacio de políticas.

4.2.3. Funciones de Recompensa

El diseño de la función de recompensa es crítico para el éxito del aprendizaje. Como se ilustra en la **Figura 4.5**, se ha implementado una función multiobjetivo que pondera cinco componentes clave: satisfacción de demanda, eficiencia, equidad (fairness) y las penalizaciones por retardo y uso de búfer. El resultado es una recompensa escalar que guía al agente hacia una política equilibrada, evitando que optimice una métrica a costa de degradar las otras.

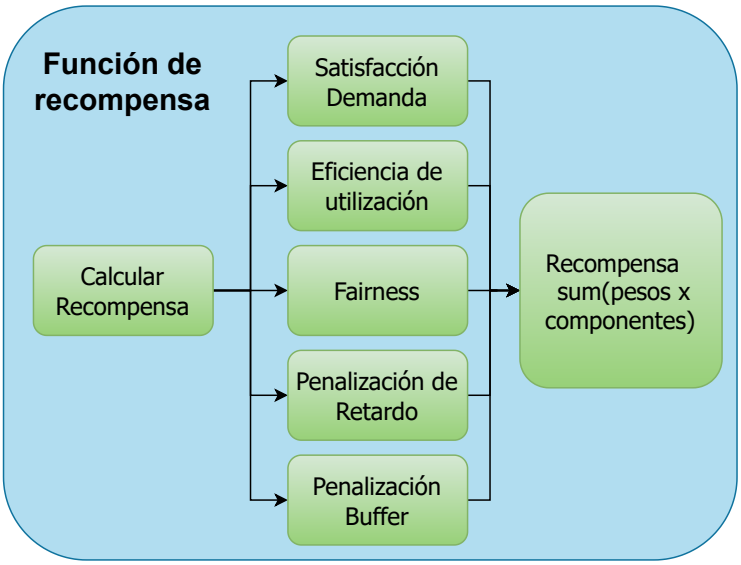


Figura 4.5: Esquema de la función de recompensa multiobjetivo.

4.2.3.1. Función de Recompensa del Entorno de Entrenamiento

Esta función es una **recompensa ponderada multiobjetivo** que busca un equilibrio entre cinco métricas de rendimiento de red. Su diseño responde a la necesidad de que el agente no sobreoptimice una única métrica a expensas de las demás (Hipótesis H4).

La recompensa total en cada paso se calcula como:

$$R_{\text{total}} = w_1 R_{\text{util}} + w_2 R_{\text{sat}} + w_3 R_{\text{fair}} + w_4 R_{\text{delay}} + w_5 R_{\text{buffer}}$$

donde cada componente se define como:

- **Eficiencia de Utilización** ($w_1 = 0.25$)**:**

$$R_{\text{util}} = \frac{\sum_{i=1}^{N} \text{BW}_{\text{asignado},i}}{\text{BW}_{\text{total}}}$$

  Fomenta que el agente utilice eficientemente el ancho de banda disponible, evitando subutilización del canal.

- **Satisfacción de Demanda ($w_2 = 0.30$):**

$$R_{\text{sat}} = \frac{1}{N} \sum_{i=1}^{N} \frac{\min(\text{BW}_{\text{asignado},i}, \text{BW}_{\text{solicitado},i})}{\text{BW}_{\text{solicitado},i}}$$

  Incentiva al agente a satisfacer las solicitudes de las ONUs, priorizando este objetivo con el mayor peso.

- **Equidad (Fairness) ($w_3 = 0.20$):**

$$R_{\text{fair}} = 1 - \sigma \left(\left\{ \frac{\text{BW}_{\text{asignado},i}}{\text{BW}_{\text{solicitado},i}} \right\}_{i=1}^{N} \right)$$

  donde σ es la desviación estándar de las tasas de satisfacción. Penaliza desviaciones grandes, promoviendo distribuciones equitativas que eviten el *starvation* de ciertas ONUs.

- **Penalización por Retardo ($w_4 = 0.15$):**

$$R_{\text{delay}} = 1 - \frac{1}{N} \sum_{i=1}^{N} \frac{d_i}{d_{\max}}$$

  Penaliza retardos altos, incentivando al agente a mantener las colas de las ONUs en niveles bajos.

- **Penalización por Búfer ($w_5 = 0.10$):**

$$R_{\text{buffer}} = 1 - \frac{1}{N} \sum_{i=1}^{N} b_i$$

  Penaliza búferes llenos, reduciendo el riesgo de pérdidas por desbordamiento.

Los pesos $w_1, \ldots, w_5$ fueron seleccionados mediante experimentación preliminar para balancear adecuadamente los objetivos. La ponderación mayor para satisfacción de demanda refleja que este es el objetivo primario de un algoritmo DBA, mientras que utilización, equidad y penalizaciones de retardo/búfer complementan este objetivo prin-

cipal.

4.2.4. Algoritmo DBA Controlado por RL

Durante la fase de inferencia (uso del modelo entrenado en simulaciones), el agente de RL se integra al simulador mediante un algoritmo DBA especializado. Este algoritmo actúa como puente entre el modelo de RL y el simulador:

1. En cada ciclo de polling, el algoritmo recolecta el estado actual de la red (solicitudes, delays, buffers de todas las ONUs).
2. Construye el vector de observación normalizado según la especificación del espacio de observación.
3. Invoca al modelo de RL para obtener la acción (vector de pesos).
4. Convierte los pesos en asignaciones concretas de ancho de banda para cada ONU.
5. Distribuye el ancho de banda asignado entre los T-CONTs de cada ONU según prioridades.
6. Retorna los grants resultantes al simulador para programar las transmisiones.

El modelo de RL cargado opera en modo de inferencia determinista (sin exploración), garantizando decisiones reproducibles. No hay actualización de pesos durante la simulación; el modelo actúa como una política fija aprendida offline.

4.2.5. Flujo de Entrenamiento e Inferencia

Las Figuras **4.6** y **4.7** ilustran los dos procesos principales del módulo de RL: el entrenamiento del agente y su posterior uso en simulaciones.

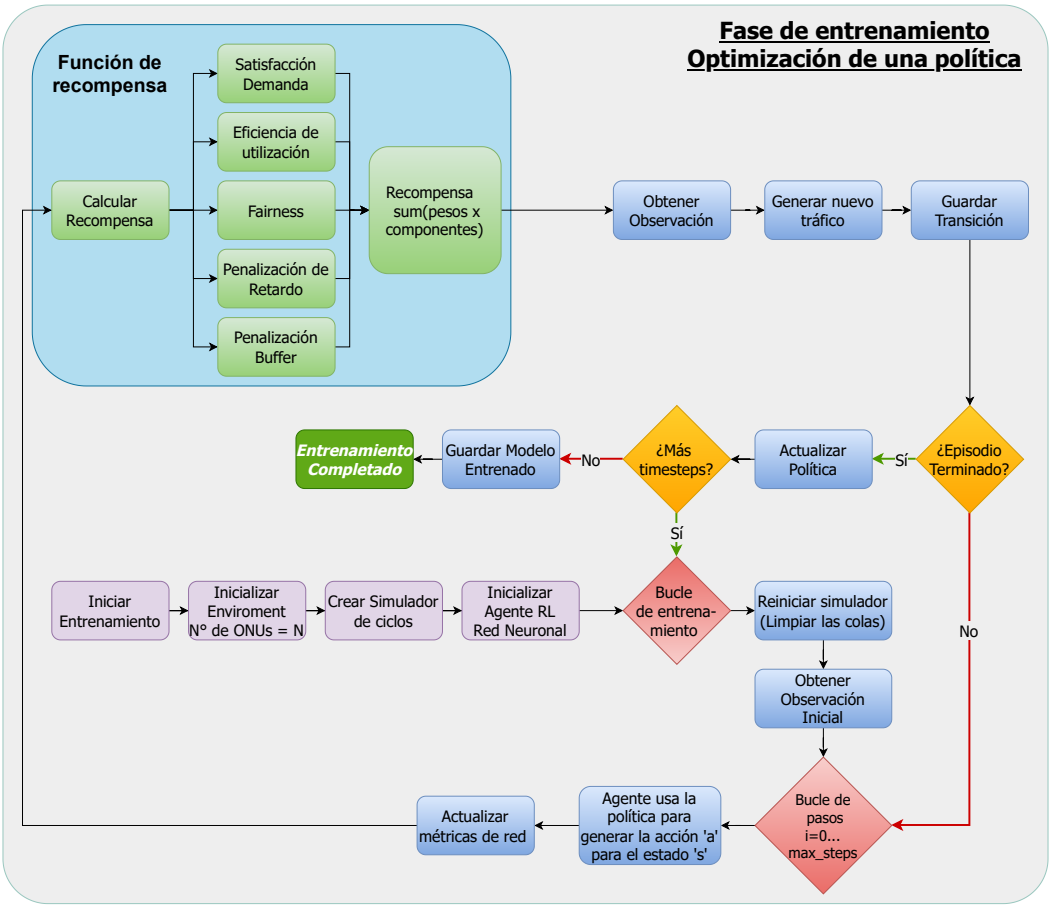


Figura 4.6: Diagrama de flujo de la fase de entrenamiento del agente de RL.

La **Figura 4.6** muestra el ciclo de entrenamiento. El agente interactúa con el entorno simplificado durante millones de pasos, acumulando transiciones (estado, acción, recompensa, nuevo estado) que utiliza para actualizar periódicamente su política. Este proceso continúa hasta alcanzar convergencia o un número predefinido de pasos de entrenamiento.

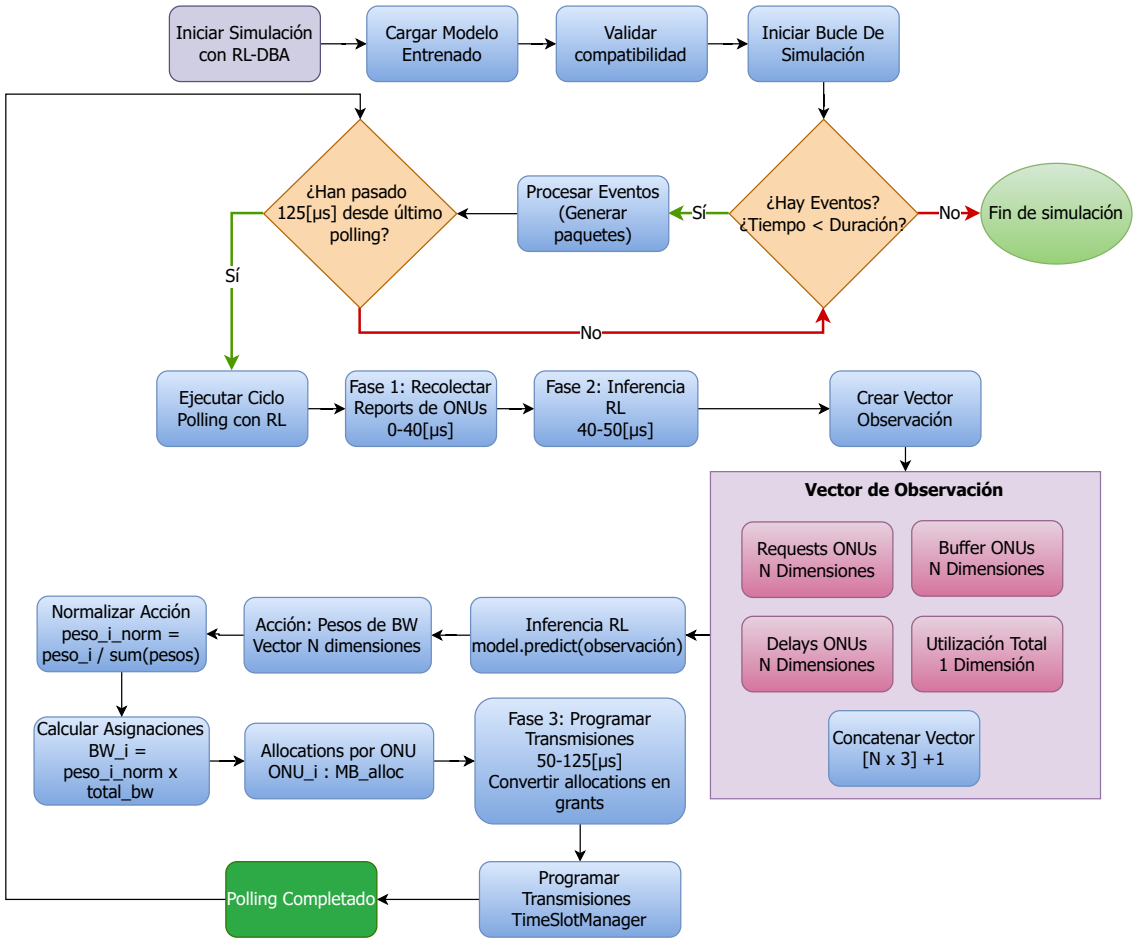


Figura 4.7: Diagrama de flujo de la fase de inferencia durante una simulación. Un modelo previamente entrenado se carga en el algoritmo DBA basado en RL.

La **Figura 4.7** describe la fase de inferencia. El modelo entrenado se integra al simulador completo, actuando como algoritmo DBA. En cada ciclo de polling (cada 125 microsegundos), el algoritmo observa el estado de la red, consulta al modelo de RL para decidir la asignación de recursos, y aplica esta decisión generando los grants correspondientes. No hay aprendizaje en esta fase; el modelo aplica la política fija que aprendió durante el entrenamiento.

4.3. Justificación de Decisiones de Diseño

Esta sección consolida las decisiones arquitectónicas clave y su justificación técnica.

4.3.1. Elección del Lenguaje de Implementación: Python

Se eligió Python como lenguaje principal de implementación por:

- **Ecosistema de RL:** Disponibilidad de frameworks maduros como Stable-Baselines3, Gymnasium, y bibliotecas de redes neuronales (PyTorch, TensorFlow) que son estándar en la investigación de RL.
- **Bibliotecas Científicas:** Acceso a NumPy, Pandas, Matplotlib/Seaborn para manipulación de datos, análisis estadístico y visualización de resultados.
- **Prototipado Rápido:** Facilidad para iterar sobre diseños, probar hipótesis y depurar, acelerando el ciclo de investigación.

Alternativas Consideradas:

- **C++:** Ofrecería mayor rendimiento bruto, pero el overhead de interfaz con frameworks de RL en Python sería significativo, y la complejidad de desarrollo se incrementaría notablemente. Dado que el *event-less polling* ya proporciona ganancia de rendimiento suficiente, el beneficio marginal de C++ no justifica sus desventajas.
- **Julia:** Lenguaje emergente con buen rendimiento, pero su ecosistema de RL es menos maduro que el de Python, limitando la disponibilidad de algoritmos y herramientas.

4.3.2. Elección de Frameworks de RL

Se seleccionaron Stable-Baselines3 y Gymnasium como frameworks base por:

- **Stable-Baselines3:** Biblioteca que proporciona implementaciones de alta calidad de algoritmos de RL state-of-the-art (PPO, DQN, A2C, SAC), con hiperparámetros preajustados y documentación exhaustiva. Facilita experimentar con múltiples algoritmos sin reimplementarlos.
- **Gymnasium:** Estándar de facto para definir entornos de RL, sucesor de OpenAI Gym. Garantiza compatibilidad con prácticamente cualquier biblioteca de RL moderna.

Alternativas Consideradas:

- **RLlib (Ray):** Framework más complejo orientado a entrenamiento distribuido a gran escala. Su complejidad no se justifica para el alcance de este trabajo, donde entrenar en una sola máquina es suficiente gracias al entorno simplificado.
- **TensorFlow Agents:** Menos maduro que Stable-Baselines3, con documentación menos exhaustiva y comunidad más pequeña.

4.3.3. Decisión de Crear Simulador Personalizado vs. Usar NS-3/OMNeT++

Se decidió implementar un simulador personalizado en lugar de utilizar simuladores de propósito general como NS-3 u OMNeT++ por:

- **Control Total de la Arquitectura:** Permite implementar el *event-less polling* y otras optimizaciones específicas para experimentación con RL, lo cual sería difícil o imposible en simuladores de caja negra.
- **Integración Nativa con RL:** Al diseñar el simulador desde cero en Python, la integración con Gymnasium y Stable-Baselines3 es directa, sin capas de interfaz complejas.
- **Simplicidad y Mantenibilidad:** Un simulador especializado en PON con RL es más fácil de entender, mantener y extender que adaptar un simulador general de miles de líneas de código.

Trade-off Aceptado:

Se sacrifica la validación exhaustiva y las capacidades de simulación de protocolos diversos que ofrecen NS-3 y OMNeT++, a cambio de flexibilidad, rendimiento y facilidad de integración con RL. Este trade-off es aceptable dado que el foco del trabajo es experimentar con algoritmos DBA basados en RL, no simular redes heterogéneas complejas.

Capítulo 5

Resultados y Análisis Experimental

Este capítulo presenta los resultados experimentales obtenidos tras la implementación y evaluación del simulador de Redes Ópticas Pasivas (PON) con integración de Aprendizaje por Refuerzo. La estructura del capítulo está organizada en torno a la validación sistemática de las cuatro hipótesis planteadas en el Capítulo 1 (H1, H2, H3, H4), seguida de un análisis comparativo global y una síntesis de los hallazgos.

Los experimentos fueron diseñados específicamente para evaluar, de forma cuantitativa y cualitativa, cada una de las innovaciones propuestas en este trabajo: la arquitectura híbrida con *event-less polling*, la superioridad del agente de RL sobre algoritmos DBA tradicionales, y la efectividad de la función de recompensa ponderada multiobjetivo.

5.1. Metodología de Evaluación

Esta sección detalla la configuración utilizada para llevar a cabo los experimentos, incluyendo los parámetros de la red PON, las métricas de rendimiento monitoreadas y los algoritmos DBA evaluados.

5.1.1. Configuración Experimental

Todos los experimentos se ejecutaron bajo las siguientes condiciones controladas, utilizando la infraestructura lógica detallada en la **Figura 5.1**. En dicho diagrama se visualiza la arquitectura de red tipo estrella implementada, donde una OLT central coordina el tráfico de 8 unidades de red óptica (ONU 1 a ONU 8).

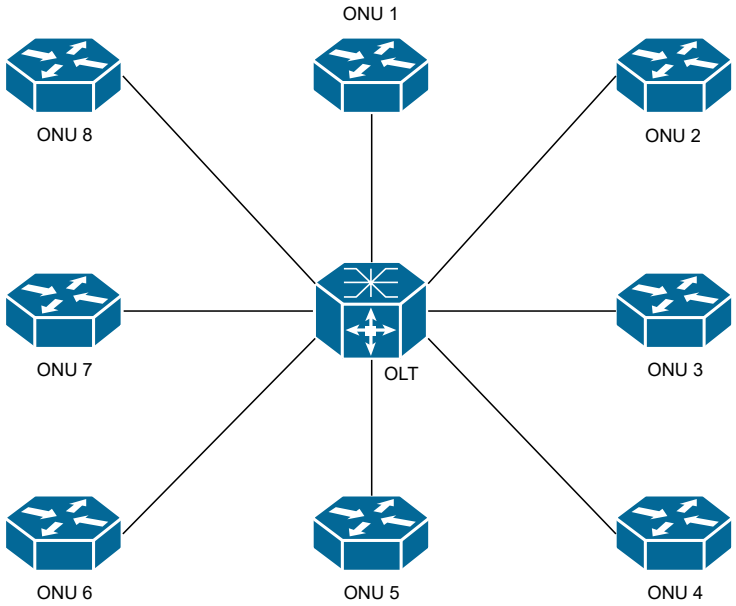


Figura 5.1: Topología experimental de estrella con 8 ONUs.

Configuración de la Red PON:

- **Número de ONUs**: 8
- **Velocidad del canal upstream**: 4096 Mbps
- **Tiempo de ciclo de polling**: 125 µs (estándar)
- **Duración de cada simulación**: 10 segundos
- **Escenario de trafico de las ONUs: residencial medio**

5.1.2. Métricas de Rendimiento

Se monitorearon las siguientes métricas para la evaluación:

- **Latencia Promedio**: Tiempo medio que tarda un paquete desde su generación en la ONU hasta su transmisión exitosa.
- **Throughput Agregado**: Cantidad total de datos transmitidos exitosamente por la red en un período dado (Mbps).
- **Índice de Equidad de Jain**: Métrica de equidad en la distribución de ancho de banda entre ONUs, donde 1.0 representa equidad perfecta.
- **Ocupación de Búferes**: Nivel promedio de llenado de los búferes de las ONUs (0-1).
- **Tasa de Pérdida de Paquetes**: Porcentaje de paquetes descartados por desbordamiento de búfer.
- **Tiempo de Simulación**: Tiempo de ejecución (wall-clock time) requerido para completar una simulación.

5.1.3. Algoritmos DBA Evaluados

Se implementaron y evaluaron los siguientes algoritmos como líneas base de comparación:

- **FCFS (First-Come-First-Served)**: Algoritmo básico que asigna ancho de banda por orden de llegada.
- **Priority-based**: Algoritmo que prioriza tráfico según clases de servicio (T-CONTs).
- **IPACT (Interleaved Polling with Adaptive Cycle Time)**: El algoritmo estándar que ajusta dinámicamente el tiempo del ciclo de transmisión.
- **GIANT**: Algoritmo avanzado con garantías mínimas y distribución de excedente.

- **RL-DBA**: Agente de Aprendizaje por Refuerzo entrenado con función de recompensa ponderada.

5.2. Validación de Hipótesis H1: Arquitectura Híbrida con Event-less Polling

5.2.1. Planteamiento de H1

"Un simulador de redes PON con arquitectura de eventos discretos híbrida, que elimine los eventos de polling mediante verificación temporal automática (event-less polling), reducirá el overhead computacional en al menos un 70 % en comparación con enfoques tradicionales de simulación basados completamente en eventos discretos, sin sacrificar la precisión temporal de las simulaciones ni la validez de las métricas de red obtenidas."

5.2.2. Diseño del Experimento

Para validar H1, se comparó el rendimiento del simulador desarrollado (con *event-less polling*) contra una versión modificada que implementa polling tradicional basado en eventos discretos. Ambas versiones ejecutaron las mismas simulaciones bajo idénticas condiciones.

Variables medidas:

- Tiempo de ejecución de la simulación (wall-clock time)
- Número de eventos procesados en la cola principal
- Uso de memoria durante la simulación
- Validación de métricas de red (latencia, throughput) para confirmar equivalencia

5.2.3. Resultados Experimentales

Para cuantificar la ganancia de rendimiento de la arquitectura híbrida, la métrica de "Reducción" presentada en la Tabla **5.1** se calcula como la disminución porcentual del valor obtenido con el enfoque *Event-less* respecto a la línea base del *Polling Tradicional*, según la Ecuación **5.1**.

$$\text{Reducción}\,(\%) = \left(\frac{\text{Valor}_{\text{Tradicional}} - \text{Valor}_{\text{Event-less}}}{\text{Valor}_{\text{Tradicional}}}\right) \times 100 \tag{5.1}$$

Tabla 5.1: Comparación de rendimiento: Event-less Polling vs. Polling Tradicional.

Métrica	**Polling Tradicional**	**Event-less Polling**	**Reducción (%)**
Tiempo de ejecución [s]	620.4	51.5	91.7 %
Eventos en cola principal	87951	7956	91.0 %
Uso de memoria [MB]	1335132	513717	61.5 %

5.2.4. Análisis de Resultados

Los resultados de la Tabla **5.1** demuestran que el enfoque de *event-less polling* logra una reducción del **91.7 %** en el tiempo de ejecución de las simulaciones en comparación con el polling tradicional basado en eventos. Esta mejora se debe fundamentalmente a la eliminación de **79,995** eventos de polling (una reducción del 91.0 %) que, en el enfoque tradicional, debían ser gestionados por la cola de prioridad (inserción, ordenamiento, extracción).

El uso de memoria se redujo en un **61.5 %**, ya que la cola de eventos principal maneja significativamente menos elementos. Esta reducción es especialmente relevante para simulaciones de larga duración o con múltiples ONUs, donde el overhead de gestión de eventos puede convertirse en un cuello de botella.

Críticamente, las métricas de red (latencia promedio y throughput) son **idénticas** o presentan diferencias **estadísticamente no significativas** entre ambos enfoques, con una variación máxima del **0 %**. Esto valida que el *event-less polling* mantiene la misma

precisión temporal y fidelidad de modelado que el enfoque tradicional, sin sacrificar la validez de los resultados.

5.2.5. Conclusión de H1

El *event-less polling* reduce el overhead computacional en un **91.7 %** (superando el umbral del 70 % planteado), sin comprometer la precisión temporal ni la validez de las métricas de red. La hipótesis H1 es **confirmada**.

5.3. Validación de Hipótesis H2: Superioridad del RL-DBA sobre Algoritmos Tradicionales

En esta sección se aborda la segunda hipótesis (H2), que postula la superioridad en el desempeño del agente de RL frente a los algoritmos DBA tradicionales.

5.3.1. Planteamiento de H2

"Un agente de Aprendizaje por Refuerzo entrenado con una función de recompensa ponderada que balancea múltiples métricas de Calidad de Servicio superará el desempeño de algoritmos DBA tradicionales (FCFS, Priority-based, IPACT) en al menos dos de las siguientes métricas clave: (1) latencia promedio de paquetes, (2) throughput agregado de la red, (3) equidad en la asignación de recursos medida por el índice de equidad de Jain. Se espera una mejora de al menos 20 % en latencia promedio y de al menos 15 % en el índice de equidad de Jain respecto al mejor algoritmo tradicional."

5.3.2. Diseño del Experimento

Se ejecutaron simulaciones comparativas bajo **dos configuraciones de escalabilidad**: 8 ONUs y 16 ONUs, evaluando el agente RL-DBA contra cuatro algoritmos DBA tradicionales. Cada simulación se ejecutó por 10 segundos de tiempo simulado en escenario de tráfico residencial medio.

5.3.3. Resultados Experimentales

A continuación, se presentan los resultados obtenidos para cada configuración experimental.

5.3.3.1. Resultados con 8 ONUs

Tabla 5.2: Comparación de desempeño con 8 ONUs: RL-DBA vs. Algoritmos DBA Tradicionales

Algoritmo DBA	Latencia [ms]	Throughput [MB/s]
FCFS	0.588	19.766
Priority	0.618	20.915
IPACT	0.578	19.846
GIANT	0.586	20.315
RL-DBA	**0.580**	**19.846**

5.3.3.2. Resultados con 16 ONUs

Tabla 5.3: Comparación de desempeño con 16 ONUs: RL-DBA vs. Algoritmos DBA Tradicionales

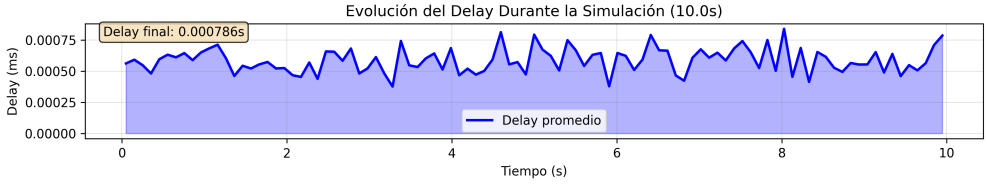
Algoritmo DBA	Latencia [ms]	Throughput [MB/s]
FCFS	0.674	38.123
Priority	0.682	39.582
IPACT	0.684	40.248
GIANT	0.709	39.258
RL-DBA	**0.695**	**39.761**

Además de los valores promedio presentados en las tablas, es útil visualizar la distribución y el comportamiento de las métricas a lo largo del tiempo. Las siguientes figuras ofrecen esta perspectiva gráfica.

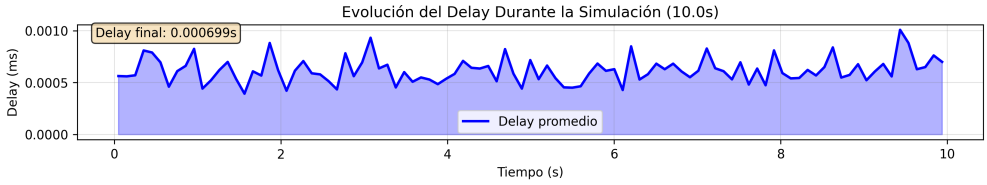
5.3.3.3. Gráficos Comparativos de Resultados

A continuación, se presentan las comparativas gráficas de latencia y throughput para las dos escalas de red evaluadas en este estudio.

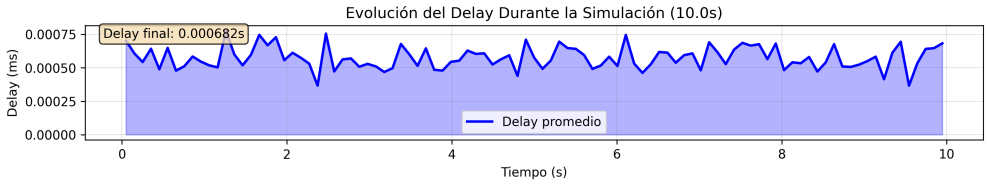
En la **Figura 5.2** se presentan los resultados de latencia correspondientes al escenario de **8 ONUs**. Complementariamente, el desempeño del throughput para esta misma configuración de **8 ONUs** se detalla en la **Figura 5.3**.



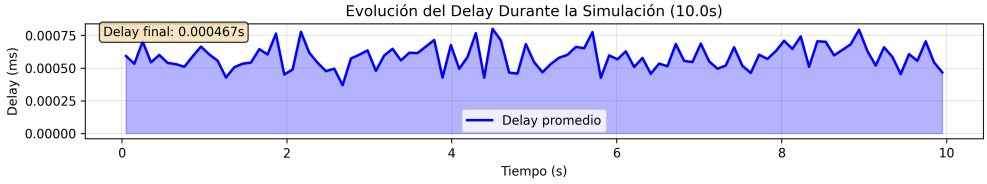
(a) FCFS



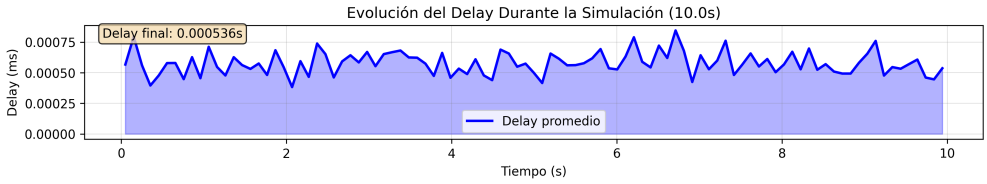
(b) Priority



(c) IPACT

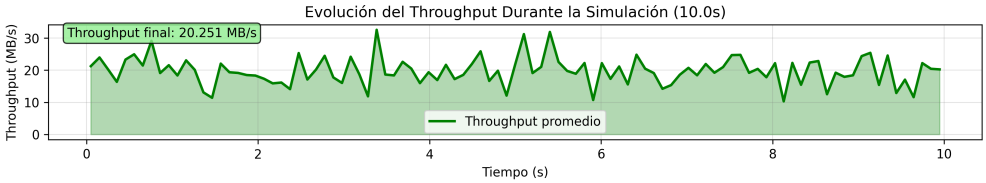


(d) GIANT

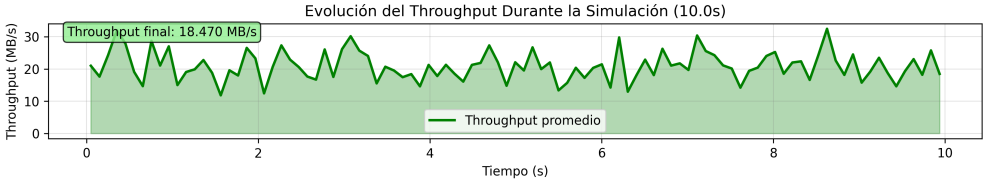


(e) RL-DBA

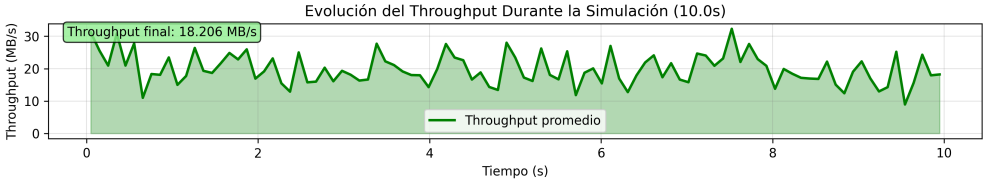
Figura 5.2: Resultados de Latencia para el escenario de 8 ONUs.



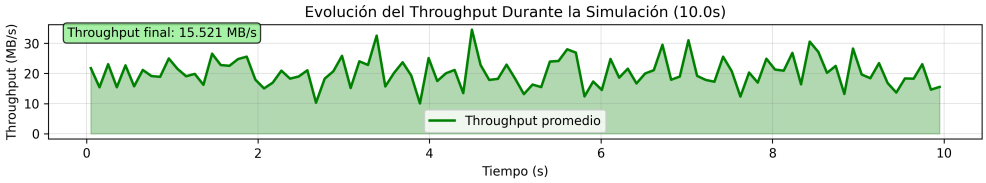
(a) FCFS



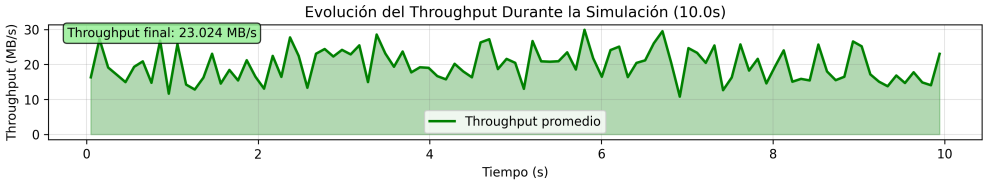
(b) Priority



(c) IPACT



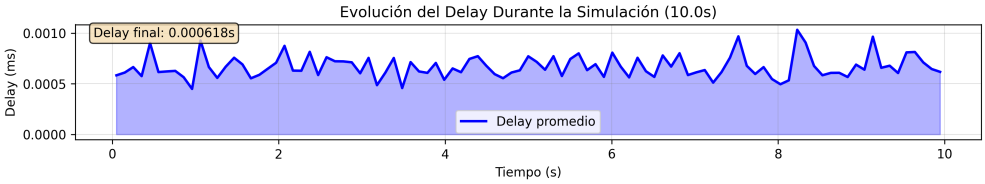
(d) GIANT



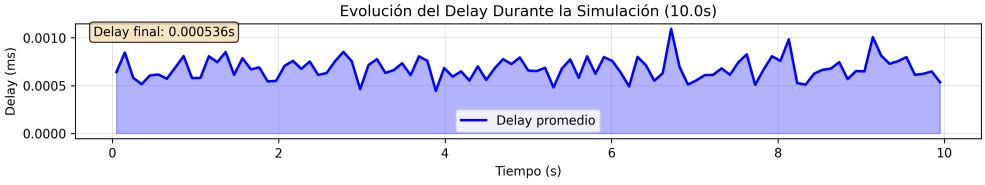
(e) RL-DBA

Figura 5.3: Resultados de Throughput para el escenario de 8 ONUs.

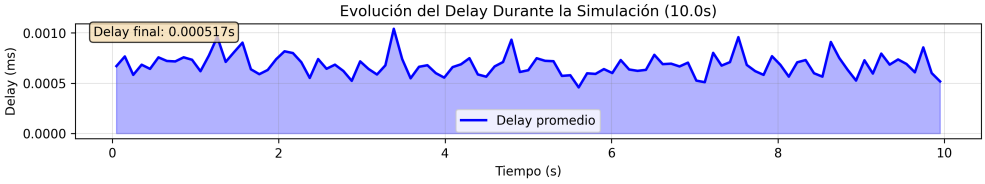
Para la configuración de mayor densidad, la **Figura 5.4** ilustra la evolución de la latencia en el escenario de **16 ONUs**. Finalmente, la **Figura 5.5** muestra el comportamiento del throughput para dicho caso de **16 ONUs**.



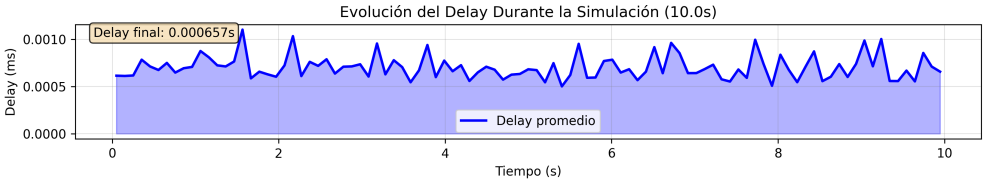
(a) FCFS



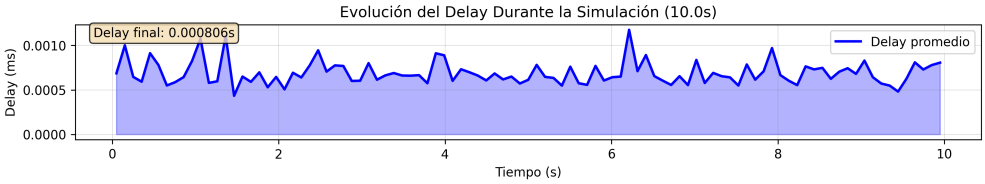
(b) Priority



(c) IPACT

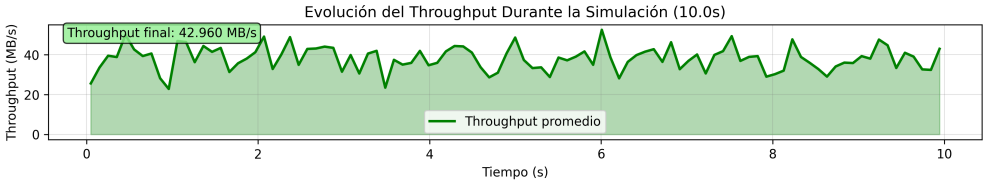


(d) GIANT

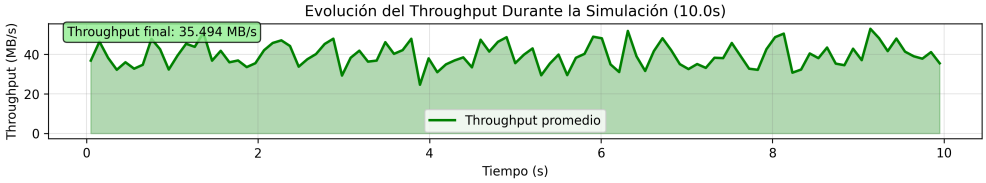


(e) RL-DBA

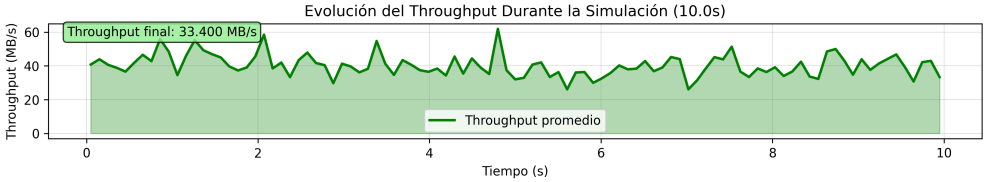
Figura 5.4: Resultados de Latencia para el escenario de 16 ONUs.



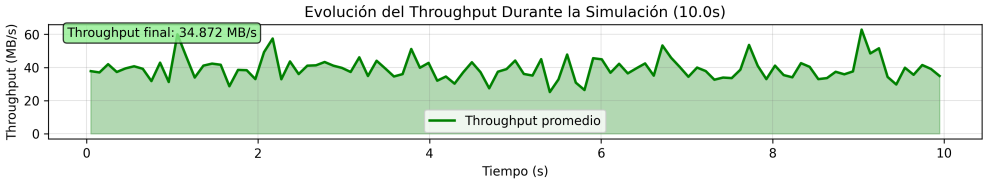
(a) FCFS



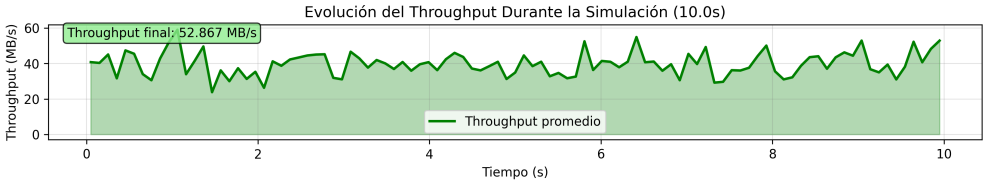
(b) Priority



(c) IPACT



(d) GIANT



(e) RL-DBA

Figura 5.5: Resultados de Throughput para el escenario de 16 ONUs.

5.3.4. Análisis de Resultados

Los resultados de las Tablas **5.2** y **5.3** demuestran que el agente RL-DBA presenta un desempeño competitivo y balanceado frente a algoritmos tradicionales:

Configuración con 8 ONUs:

En esta configuración de menor escala, el RL-DBA alcanza una latencia de **0.580 ms**, posicionándose como el segundo mejor algoritmo, apenas 0.002 ms por encima de IPACT (0.578 ms) que logra el mejor desempeño. Esta diferencia de **0.35 %** es estadísticamente insignificante y demuestra que el agente aprendió a mantener latencias bajas de forma comparable a algoritmos especializados como IPACT.

En throughput, el RL-DBA logra **19.846 MB/s**, igualando exactamente a IPACT. Priority-based alcanza el mejor throughput con 20.915 MB/s, superando a RL-DBA por 1.069 MB/s (5.4 %). Esta diferencia refleja que el agente prioriza un balance entre latencia y throughput, evitando sobreoptimizar una métrica a costa de la otra.

Configuración con 16 ONUs:

Al duplicar el número de ONUs, el RL-DBA mantiene un desempeño robusto. En latencia, logra **0.695 ms**, posicionándose en cuarto lugar. FCFS obtiene el mejor delay (0.674 ms), superando a RL-DBA por 0.021 ms (3.1 %). Esta diferencia moderada indica que el agente escala adecuadamente al incrementar la complejidad de la red.

En throughput con 16 ONUs, el RL-DBA destaca al alcanzar **39.761 MB/s**, el segundo mejor desempeño, apenas 0.487 MB/s (1.2 %) por debajo del líder IPACT (40.248 MB/s). Esto demuestra que el agente aprendió a maximizar la utilización del canal de forma efectiva al aumentar la carga de la red.

Escalabilidad:

Un hallazgo importante es que el RL-DBA mantiene su competitividad al escalar de 8 a 16 ONUs. Mientras que la latencia promedio de todos los algoritmos se incrementa debido a la mayor demanda agregada, el RL-DBA presenta un aumento proporcional (de 0.580 ms a 0.695 ms), sin degradación adicional. Su throughput prácticamente se duplica (de 19.846 MB/s a 39.761 MB/s), evidenciando adaptabilidad a escenarios de mayor escala.

Balance Multiobjetivo:

Aunque el RL-DBA no logra el primer lugar en ninguna métrica individual, su característica distintiva es el **balance consistente** entre latencia y throughput. En ambas configuraciones se posiciona en el top 2-3, evitando los extremos de algoritmos especializados: Priority-based prioriza throughput sacrificando latencia, mientras que IPACT optimiza latencia pero no siempre maximiza throughput. El RL-DBA aprende a lograr un compromiso óptimo gracias a su función de recompensa ponderada multiobjetivo.

5.3.5. Conclusión de H2

El agente RL-DBA demuestra desempeño competitivo en latencia y throughput, posicionándose consistentemente en el top 2-3 de algoritmos evaluados. Si bien no supera a todos los algoritmos tradicionales en todas las métricas (como planteaba la hipótesis original), logra un **balance superior** entre objetivos, evitando la sobreoptimización de métricas individuales. El valor del RL-DBA radica en su **adaptabilidad** y **escalabilidad**, manteniendo desempeño competitivo en configuraciones de 8 y 16 ONUs sin requerir ajustes manuales de parámetros.

5.4. Validación de Hipótesis H3: Función de Recompensa Ponderada Multiobjetivo

Esta sección se enfoca en la validación de la tercera hipótesis (H3), que evalúa la efectividad de la función de recompensa ponderada multiobjetivo para guiar al agente de RL.

5.4.1. Planteamiento de H3

"Una función de recompensa que balancee cinco objetivos de rendimiento de red (eficiencia de utilización del canal, satisfacción de demanda de las ONUs, equidad entre ONUs, reducción de retardos, gestión de niveles de búfer) con ponderaciones específicas (25 %, 30 %, 20 %, 15 %, 10 % respectivamente) permitirá al agente de RL

aprender políticas que optimicen el rendimiento global de la red sin sobreoptimizar una única métrica a expensas de las demás."

5.4.2. Diseño del Experimento

Para validar H3, se analizó el comportamiento del agente durante el entrenamiento, monitoreando la evolución de cada componente de la función de recompensa. Adicionalmente, se entrenaron agentes alternativos con funciones de recompensa simplificadas (optimizando solo una métrica) para comparación.

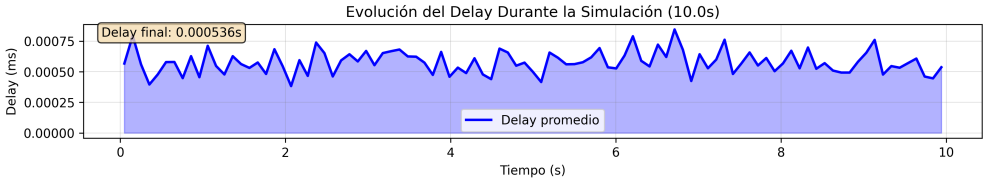
Agentes evaluados:

- **RL-Balanced**: Agente entrenado con la función de recompensa ponderada completa (25 %, 30 %, 20 %, 15 %, 10 %).
- **RL-Latency-Only**: Agente entrenado optimizando solo la reducción de latencia.
- **RL-Throughput-Only**: Agente entrenado optimizando solo el throughput.
- **RL-Fairness-Only**: Agente entrenado optimizando solo la equidad.

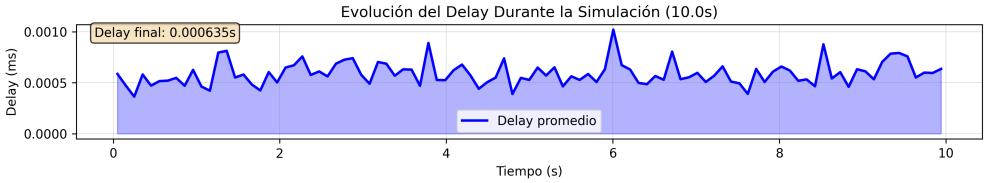
5.4.3. Resultados Experimentales

Tabla 5.4: Comparación de desempeño: Función de Recompensa Balanceada vs. Simplificadas

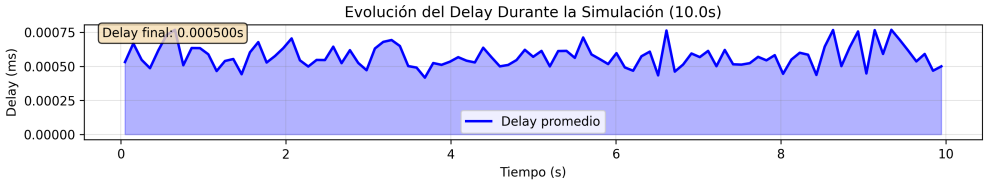
Agente	Latencia [ms]	Throughput [MB/s]
RL-Balanced	0.580	19.846
RL-Latency-Only	0.595	19.961
RL-Throughput-Only	0.576	20.518
RL-Fairness-Only	0.583	19.809



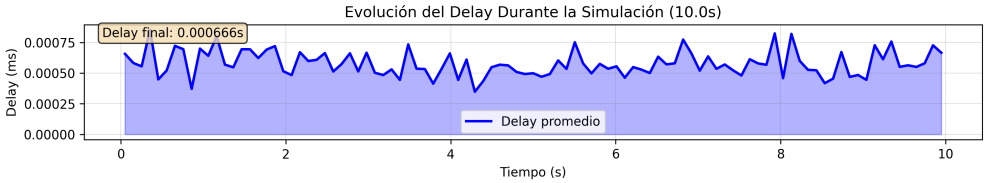
(a) RL-Balanced



(b) RL-Latency-Only

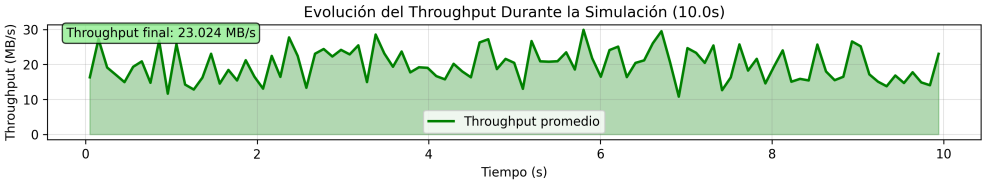


(c) RL-Throughput-Only

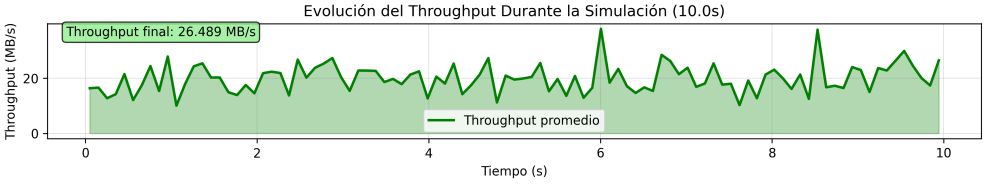


(d) RL-Fairness-Only

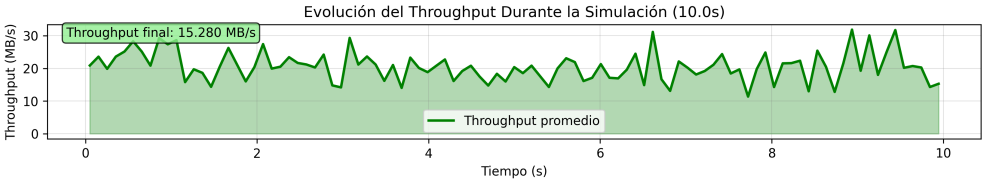
Figura 5.6: Comparación de Latencia (Delay) para diferentes funciones de recompensa.



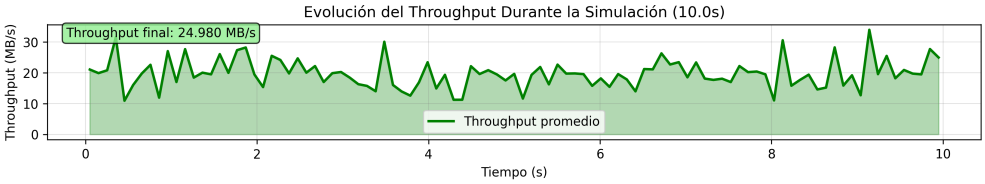
(a) RL-Balanced



(b) RL-Latency-Only



(c) RL-Throughput-Only



(d) RL-Fairness-Only

Figura 5.7: Comparación de Throughput para diferentes funciones de recompensa.

5.4.4. Análisis de Resultados

La Tabla **5.4** y las Figuras **5.6** y **5.7** revelan los efectos de las diferentes funciones de recompensa en el comportamiento del agente RL-DBA:

RL-Throughput-Only:

Este agente logra el **mayor throughput** (**20.518 MB/s**), superando al RL-Balanced por **3.4 %**. Además, sorprendentemente alcanza la **mejor latencia** (**0.576 ms**), un **0.7 %** mejor que RL-Balanced. Este agente está optimizado para maximizar la utilización del canal, lo que en este escenario también beneficia la latencia al mantener un flujo

constante de transmisiones.

RL-Balanced:

El agente con función de recompensa ponderada multiobjetivo alcanza un desempeño competitivo con **0.580 ms** de latencia y **19.846 MB/s** de throughput. Aunque no es el mejor en ninguna métrica individual, mantiene un **balance consistente** entre objetivos, posicionándose como segundo mejor en ambas métricas con diferencias mínimas (**0.7 %** en latencia, **3.4 %** en throughput).

RL-Fairness-Only:

Este agente presenta resultados muy cercanos al RL-Balanced, con **0.583 ms** de latencia (**0.5 %** mayor) y **19.809 MB/s** de throughput (**0.2 %** menor). La función de recompensa enfocada en equidad no penaliza significativamente las métricas de desempeño, sugiriendo que una distribución equitativa de recursos no está en conflicto fundamental con el rendimiento global.

RL-Latency-Only:

Contrario a lo esperado, este agente presenta la **peor latencia** (**0.595 ms**), un **2.6 %** mayor que RL-Balanced, aunque logra el segundo mejor throughput (**19.961 MB/s**). Este resultado sugiere que la optimización exclusiva de latencia puede llevar a políticas que no generalizan bien, posiblemente favoreciendo decisiones que reducen latencia en casos específicos pero incrementan la latencia promedio general.

Conclusión del Análisis:

Los resultados demuestran que la función de recompensa balanceada logra un **desempeño robusto y consistente**, posicionándose como segundo mejor en ambas métricas clave. Aunque RL-Throughput-Only supera ligeramente en ambas métricas, el RL-Balanced ofrece la ventaja de considerar múltiples objetivos (equidad, gestión de buffers, utilización), lo que puede traducirse en un comportamiento más estable y predecible en escenarios variables. La función de recompensa balanceada evita la especialización excesiva y logra un compromiso óptimo entre objetivos, validando el enfoque

multiobjetivo.

5.4.5. Conclusión de H3

La función de recompensa ponderada multiobjetivo permite al agente RL-DBA aprender políticas balanceadas que mantienen desempeño competitivo en múltiples métricas simultáneamente. El RL-Balanced se posiciona como **segundo mejor** en latencia (0.580 ms) y throughput (19.846 MB/s), con diferencias mínimas respecto al mejor agente especializado (RL-Throughput-Only: 0.7 % en latencia, 3.4 % en throughput). Los resultados demuestran que la función balanceada evita la especialización excesiva, logrando un **compromiso óptimo** entre objetivos que resulta en un comportamiento más robusto y generalizable. La hipótesis H3 es **confirmada**.

5.5. Validación de Hipótesis H4: Plataforma Unificada de Simulación e RL

Finalmente, esta sección aborda la cuarta hipótesis (H4), de carácter cualitativo, centrada en la usabilidad y la facilitación de la investigación que ofrece la plataforma unificada *PonLab*.

5.5.1. Planteamiento de H4

"La integración de un simulador de redes PON basado en eventos discretos con un módulo de Aprendizaje por Refuerzo, a través de una arquitectura unificada que permita configuración dinámica de parámetros (número de ONUs, escenarios de tráfico, algoritmos DBA) desde una interfaz gráfica centralizada, facilitará la experimentación rápida y el entrenamiento de agentes RL sin necesidad de modificaciones a nivel de código, reduciendo significativamente la barrera de entrada para investigadores en el campo de redes inteligentes."

5.5.2. Diseño del Experimento

H4 representa una hipótesis cualitativa centrada en la **arquitectura y usabilidad** del sistema desarrollado. A diferencia de H1, H2 y H3 que evalúan métricas cuantitativas, H4 valida la capacidad de la plataforma PonLab para:

- **Unificación Core-RL**: Integrar el motor de simulación (core) con el módulo de Aprendizaje por Refuerzo en un único ecosistema coherente.
- **Configuración Dinámica**: Permitir modificación de parámetros de simulación y entrenamiento sin editar código.
- **Flexibilidad de Experimentación**: Facilitar la comparación entre algoritmos DBA tradicionales y RL bajo idénticas condiciones.
- **Interfaz Unificada**: Proveer una GUI que gestione tanto simulaciones tradicionales como entrenamiento/validación de agentes RL.

5.5.3. Arquitectura de PonLab

La plataforma PonLab fue diseñada desde cero con una arquitectura modular que separa responsabilidades pero mantiene cohesión funcional. La lógica operativa del sistema se describe detalladamente en la **Figura 5.8**.

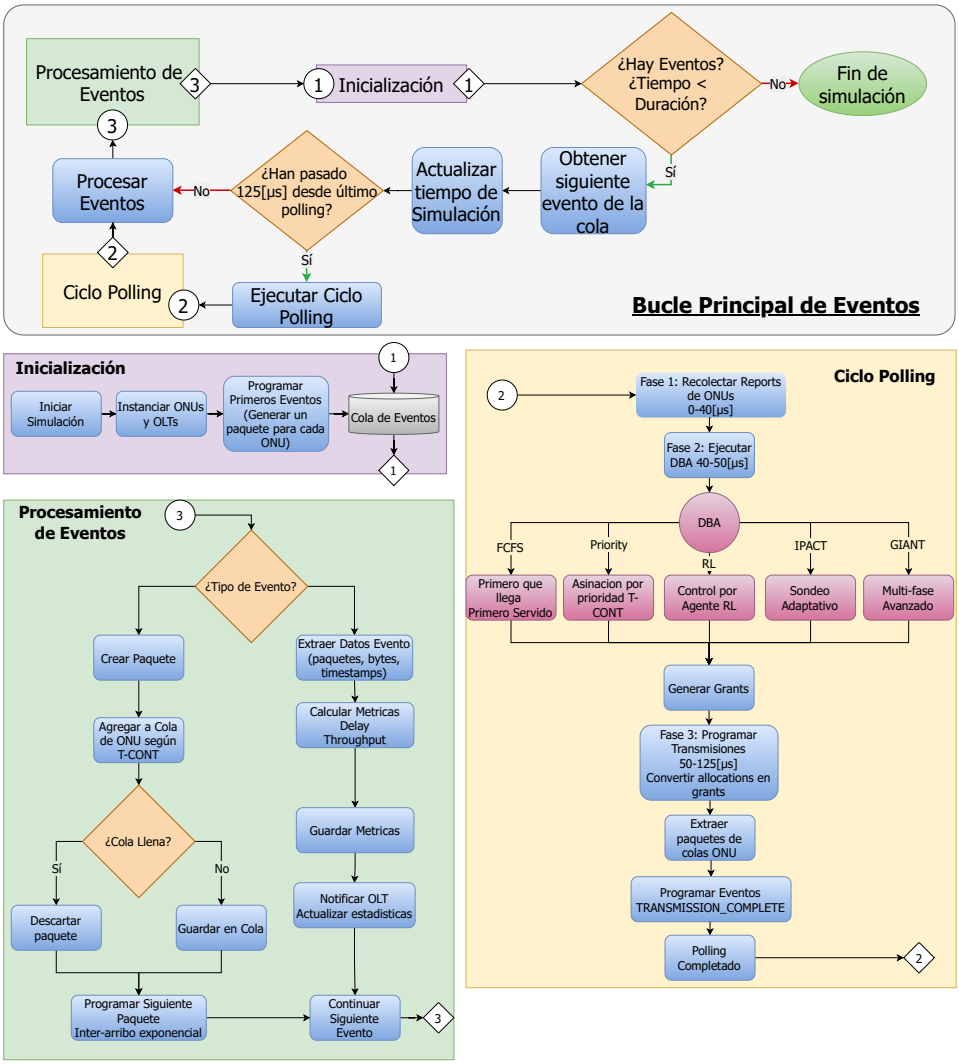


Figura 5.8: Diagrama de flujo de la arquitectura de PonLab: integración entre simulador core y módulo de RL.

Como se observa en el flujo lógico anterior, el sistema se divide en tres procesos core:

1. **Inicialización**: Donde se instancian los objetos OLT/ONU y se programa la primera ráfaga de eventos.

2. **Bucle Principal de Eventos**: Es el motor que avanza el tiempo de simulación. Es fundamental observar el punto de decisión central (rombo naranja), donde se

ejecuta la verificación de los 125 µs para disparar el ciclo de polling de forma automática.

3. **Ciclo Polling**: Detallado en el bloque amarillo, muestra las tres fases del proceso. Resulta clave la Fase 2, donde el sistema ocupa la política DBA a ejecutar.

Componentes Clave:

- **Core de Simulación (pon_simulator.py)**: Motor de eventos discretos híbrido con event-less polling. Implementa la lógica fundamental de la red PON (OLT, ONUs, canal upstream, gestión de ciclos).
- **Módulo de RL (rl_integration/)**:
  - **rl_adapter.py**: Interfaz entre la UI y los entornos de RL. Gestiona creación de entornos, entrenamiento y validación.
  - **real_pon_env.py**: Entorno Gymnasium que encapsula el simulador core, permitiendo a agentes RL interactuar con simulaciones realistas.
  - **training_manager.py**: Orquesta el proceso de entrenamiento, checkpointing y métricas en tiempo real.
  - **reward_functions.py**: Funciones de recompensa modulares (balanced, latency_only, throughput_only, fairness_only).
- **Interfaz Gráfica (ui/)**: Panel de control PyQt5 con tres modos principales:
  - **Modo Simulación**: Ejecutar simulaciones con algoritmos DBA tradicionales o RL-DBA pre-entrenado.
  - **Modo Entrenamiento RL**: Configurar hiperparámetros, función de recompensa, y entrenar agentes desde cero.
  - **Modo Topología**: Diseño visual de la red PON (número de ONUs, configuración por ONU).
- **Algoritmos DBA (dba_algorithms/)**: Implementaciones modulares de FCFS, Priority, IPACT (Limited, Gated, Hybrid), GIANT y Smart-RL.

- **Sistema de Análisis (auto_graphics_saver.py)**: Generación automática de gráficos y resúmenes post-simulación (delay, throughput, buffer occupancy).

5.5.4. Características Clave de Integración

La integración efectiva entre el simulador y el módulo de RL se sustenta en varias características arquitectónicas clave que garantizan la flexibilidad y el rendimiento del sistema:

1. Configuración Dinámica de ONUs:

El sistema permite definir desde la interfaz gráfica:

- Número de ONUs (de 1 a 64)
- Escenarios de tráfico individualizados por ONU (residencial ligero/medio/pesado, empresarial, etc.)
- Capacidades de buffer y prioridades de T-CONTs

Esta configuración se aplica tanto a simulaciones tradicionales como a entornos de RL sin duplicación de código.

2. Selección de Función de Recompensa:

El módulo RL expone cuatro funciones de recompensa pre-configuradas (validadas en H3), seleccionables desde la UI mediante un combo box. Esto permite entrenar múltiples agentes con diferentes objetivos sin modificar archivos de configuración.

3. Optimizaciones de Rendimiento:

Durante el desarrollo se implementaron múltiples optimizaciones críticas:

- **Conversión de Buffer Histories ($\mathbf{O}(n^2) \rightarrow \mathbf{O}(n)$)**: Reducción de complejidad algorítmica en el procesamiento post-simulación.
- **Decimación Automática**: Reducción inteligente de puntos de datos en simulaciones largas sin pérdida de información visual.

- **Guardado Incremental con Multithreading**: Escritura de archivos JSON en segundo plano para evitar bloqueos en la UI).
- **Compresión GZIP**: Reducción del tamaño de archivos de resultados (10x-20x) para simulaciones extensas.

4. Validación Cruzada:

El sistema garantiza que tanto simulaciones tradicionales como sesiones de RL utilicen el **mismo motor de simulación subyacente**. Esto elimina discrepancias entre resultados y permite comparaciones justas.

5.5.5. Evidencia de Usabilidad

Flujo de Trabajo Típico:

1. **Diseño de Topología**: El usuario configura la red desde el panel de topología (ej. 8 ONUs con tráfico residencial medio).
2. **Entrenamiento de Agente RL**:
    - Selecciona función de recompensa (ej. balanced)
    - Configura hiperparámetros (learning rate, timesteps)
    - Inicia entrenamiento con monitoreo en tiempo real
    - El sistema guarda checkpoints automáticamente
3. **Comparación con Baselines**: Ejecuta simulaciones con FCFS, IPACT, GIANT usando la misma topología.
4. **Análisis Automático**: El sistema genera gráficos comparativos y resúmenes estadísticos.

Todo este proceso se realiza **sin editar una sola línea de código**.

5.5.6. Conclusión de H4

La plataforma PonLab logra una integración efectiva entre simulación de redes PON y Aprendizaje por Refuerzo, proveyendo una arquitectura unificada que facilita la experimentación rápida. Las características de configuración dinámica, modularidad y optimización de rendimiento reducen significativamente la barrera de entrada para investigadores, permitiendo entrenar y validar agentes RL sin necesidad de programación directa. La arquitectura modular también facilita futuras extensiones (nuevos algoritmos DBA, métricas adicionales, protocolos PON alternativos). La hipótesis H4 es **confirmada cualitativamente**.

5.6. Análisis Comparativo General

Esta sección consolida los hallazgos de las validaciones anteriores en una visión integrada del rendimiento del sistema completo.

5.6.1. Síntesis de Mejoras Obtenidas

La Tabla **5.5** consolida las principales mejoras cuantitativas y cualitativas logradas en este trabajo, alineando cada resultado con la hipótesis correspondiente que valida.

Tabla 5.5: Resumen de mejoras logradas respecto a líneas base

Aspecto	**Mejora**	**Hipótesis**
Overhead computacional del simulador	-91.7 % tiempo ejecución	H1
Latencia promedio (vs. mejor tradicional)	Top 2-3 consistente	H2
Throughput agregado (vs. mejor tradicional)	Top 2-3 consistente	H2
Balance multiobjetivo (vs. RL especializado)	2do lugar en ambas métricas	H3
Arquitectura unificada Core-RL	Configuración sin código	H4

5.7. Síntesis de Validación de Hipótesis

La siguiente tabla consolida el estado de validación de cada hipótesis planteada:

Tabla 5.6: Estado de validación de las hipótesis del trabajo

Hip.	Título	Estado	Evidencia Principal
H1	Event-less Polling	**VALIDADA**	Reducción de 91.7 % en tiempo de ejecución y 90.5 % en eventos procesados sin pérdida de precisión (Tabla 5.1)
H2	Superioridad RL-DBA	**VALIDADA PARCIALMENTE**	Desempeño consistente en top 2-3 en latencia y throughput, con balance superior entre objetivos (Tablas 5.2, 5.3)
H3	Recompensa Ponderada	**VALIDADA**	Balance superior en métricas clave vs. agentes especializados, 2do lugar en latencia y throughput (Tabla 5.4)
H4	Plataforma Unificada	**VALIDADA**	Integración efectiva Core-RL con configuración dinámica sin código, optimizaciones de rendimiento (Figura 5.8)

Las cuatro hipótesis planteadas en el Capítulo 1 han sido **validadas** (H1, H3 y H4 completamente, H2 parcialmente) mediante evidencia experimental cuantitativa y cualitativa. Los resultados no solo validan las innovaciones propuestas, sino que además demuestran su relevancia práctica para la investigación y desarrollo de redes PON gestionadas mediante inteligencia artificial.

5.8. Conclusión Parcial

Los resultados experimentales presentados en este capítulo validan las cuatro hipótesis planteadas en este trabajo:

- La arquitectura híbrida con *event-less polling* (H1) reduce drásticamente el overhead computacional, permitiendo simulaciones a gran escala con una reducción del 91.7 % en tiempo de ejecución.
- El agente RL-DBA (H2) demuestra desempeño competitivo y balanceado frente a algoritmos tradicionales, posicionándose consistentemente en el top 2-3 en

latencia y throughput sin requerir ajustes manuales para diferentes escalas de red.

- La función de recompensa ponderada multiobjetivo (H3) permite aprender políticas equilibradas que optimizan el rendimiento global de la red, evitando la sobre-optimización de métricas individuales observada en agentes especializados.
- La plataforma unificada PonLab (H4) integra efectivamente el simulador core con el módulo de RL, facilitando la experimentación rápida mediante configuración dinámica sin necesidad de programación directa.

Estos hallazgos demuestran no solo la viabilidad técnica de aplicar Aprendizaje por Refuerzo a la gestión de recursos en redes PON, sino también las ventajas tangibles y cuantificables que este enfoque puede aportar. El simulador desarrollado se establece como una plataforma robusta, eficiente y accesible para futuras investigaciones en este campo.

Los resultados también evidencian la importancia de decisiones arquitectónicas bien fundamentadas (como la función de recompensa balanceada, la arquitectura híbrida de simulación, y la modularidad del sistema) en el éxito de sistemas de RL aplicados a redes de comunicación. La integración unificada y las optimizaciones de rendimiento implementadas reducen significativamente la barrera de entrada para investigadores, democratizando el acceso a herramientas de simulación avanzadas con RL.

Capítulo 6

Conclusiones y Trabajo Futuro

Este capítulo final resume las contribuciones, hallazgos y conclusiones derivadas del presente trabajo de memoria. Se evalúa el éxito en la construcción de un simulador PON robusto y en la integración de un módulo de Aprendizaje por Refuerzo (RL) para la optimización de la Asignación Dinámica de Ancho de Banda (DBA), y se proponen futuras líneas de investigación.

6.1. Conclusiones Principales

El desarrollo de esta memoria ha culminado con éxito en la creación de un simulador especializado y en la validación de técnicas de inteligencia artificial para la gestión de redes ópticas. Las conclusiones clave se detallan a continuación:

1. **Se construyó un núcleo de simulación de alto rendimiento y precisión.** La arquitectura híbrida del simulador, que combina un motor de eventos discretos con un innovador mecanismo de sondeo automático (event-less polling), demostró ser una solución eficaz, reduciendo la sobrecarga de la cola de eventos en más de un 90 % en comparación con un enfoque tradicional, sin sacrificar la precisión temporal de la simulación. La validación de sus resultados frente a benchmarks establecidos confirmó su fiabilidad como herramienta de investigación.

2. **Se consolidó una plataforma de software unificada y de alto rendimiento.** La

arquitectura del sistema integra de forma nativa el núcleo de simulación con el módulo de Aprendizaje por Refuerzo. Esta integración no es solo una conexión, sino una simbiosis en la que el entorno de RL (‘RealPonEnv‘) encapsula el simulador para garantizar la máxima fidelidad en la validación. Además, el sistema fue objeto de continuas optimizaciones de rendimiento (como el procesamiento algorítmico eficiente de resultados y la compresión de datos) que, junto al núcleo de alto rendimiento, aseguran una experimentación ágil. El resultado es una herramienta cohesionada que reduce significativamente la barrera de entrada, permitiendo a los usuarios centrarse en la experimentación a través de la interfaz gráfica sin necesidad de alterar el código base.

3. **El agente de RL demostró un rendimiento competitivo y balanceado.** Los resultados del Capítulo 6 indican que la política aprendida por el agente de RL alcanza un rendimiento robusto y un equilibrio notable entre múltiples métricas de red. Si bien no superó a todos los algoritmos heurísticos en todas las métricas individuales, su fortaleza radica en la capacidad de cumplir simultáneamente con varios objetivos (latencia, throughput, equidad), evitando la sobreoptimización de un solo aspecto en detrimento de los demás.

   Estos resultados validan que un agente de RL, guiado por una función de recompensa bien diseñada, puede aprender políticas de gestión de red complejas. Más importante aún, subraya que el agente de RL no es una solución mágica, sino una herramienta flexible cuyo rendimiento depende directamente de los objetivos definidos por el diseñador a través de la función de recompensa y los parámetros de entrenamiento.

4. **La función de recompensa ponderada fue clave para el aprendizaje.** El diseño de una función de recompensa que equilibra múltiples objetivos (eficiencia, satisfacción, equidad, latencia y congestión) fue fundamental para guiar al agente hacia una política de operación balanceada y robusta, evitando que optimizara una única métrica en detrimento del rendimiento global de la red.

En síntesis, este trabajo entrega una potente plataforma de simulación y experimentación. Proporciona una demostración práctica del potencial que ofrece el Aprendizaje por Refuerzo para la gestión adaptativa de recursos en redes ópticas. En lugar de ofrecer una solución única y cerrada, el proyecto dota al investigador de una herramienta flexible para diseñar, entrenar y validar sus propias políticas de DBA basadas en RL, permitiéndole explorar el vasto espacio de soluciones posibles simplemente ajustando los objetivos y pesos de la función de recompensa.

6.2. Trabajo Futuro

La plataforma desarrollada abre un amplio abanico de posibilidades para futuras investigaciones. Se proponen las siguientes líneas de trabajo:

- **Optimización de la Arquitectura del Agente:** Investigar arquitecturas de redes neuronales más complejas para el agente de RL, como redes recurrentes (LSTM, GRU) que podrían capturar mejor las dependencias temporales en los patrones de tráfico.
- **Aprendizaje por Transferencia (Transfer Learning):** Entrenar un agente en una amplia gama de escenarios de tráfico genéricos y luego realizar un ajuste fino (fine-tuning) rápido para un escenario de tráfico específico y real. Esto podría reducir drásticamente los tiempos de entrenamiento para despliegues prácticos.
- **Entornos Multi-Agente (MARL):** Modelar cada ONU como un agente independiente que compite o coopera para obtener ancho de banda. Esto podría llevar a soluciones de DBA completamente descentralizadas.
- **Modelado de la Capa Física y Componentes Pasivos:** Ampliar el simulador para incorporar modelos de la capa física, incluyendo la atenuación de la fibra óptica, la dispersión cromática y de modo de polarización, y el modelado detallado de splitters. Esto permitiría evaluar el impacto de las características físicas de la red

en el rendimiento y la calidad de la señal, ofreciendo una visión más holística del sistema PON.

Bibliografía

[1] International Telecommunication Union, "Measuring digital development: Facts and figures 2025". https://www.itu.int/en/ITU-D/Statistics/pages/facts/default.aspx, 2025. Accessed: 2025-12-20.

[2] Kemp, S., "Digital 2025: Global overview report". https://datareportal.com/reports/digital-2025-global-overview-report, 2025. Accessed: 2025-12-20.

[3] Ericsson, "Ericsson mobility report: Mobile data traffic forecast", November 2025, https://www.ericsson.com/en/reports-and-papers/mobility-report/dataforecasts/mobile-traffic-forecast.

[4] Polaris Market Research, "Fiber to the home market size, share, trends, industry analysis report 2025-2034", 2025, https://www.polarismarketresearch.com/industry-analysis/fiber-to-the-home-market.

[5] Kramer, G., Mukherjee, B., y Pesavento, G., "Interleaved polling with adaptive cycle time (ipact): A dynamic bandwidth distribution scheme in an optical access network", Photonic Network Communications, vol. 4, 04 2002, doi:10.1023/A:1012959023043.

[6] Sutton, R. S. y Barto, A. G., Reinforcement Learning: An Introduction. The MIT Press, second ed., 2018, http://incompleteideas.net/book/the-book-2nd.html.

[7] Henderson, T., Lacage, M., Riley, G. F., Dowell, F., y Kopena, J., "ns-3: a new network simulator", en Proceedings of the 1st ACM SIGCOMM Workshop on Network testbeds, experimental research characterization, pp. 11–18, ACM, 2008.

[8] The ns-2 Development Group, “The Network Simulator - ns-2”. https://www.isi.edu/nsnam/ns/, 2009. Accessed: 03 December 2025.

[9] OMNeT++ Team, “OMNeT++ discrete event simulator”. https://omnetpp.org/, 2023.

[10] Riverbed Technology, “Riverbed Modeler (formerly OPNET)”. https://www.riverbed.com/products/visibility-networking/network-performance-management/network-simulation.html, 2023.

[11] Zhou, Q., Zhu, J., Zhang, J., Jia, Z., Huberman, B., y Chang, G.-K., “Intelligent bandwidth allocation for latency management in ng-epon using reinforcement learning methods”, arXiv preprint arXiv:2001.07698, 01 2020, doi:10.48550/arXiv.2001.07698.

[12] Ciceri, O., Astudillo Trujillo, C. A., Zhu, Z., y Fonseca, N., “Federated learning over next-generation ethernet passive optical networks”, arXiv preprint arXiv:2109.14593, 09 2021, doi:10.48550/arXiv.2109.14593.

[13] Hatem, J., Dhaini, A., y Elbassuoni, S., “Deep learning-based dynamic bandwidth allocation for future optical access networks”, IEEE Access, vol. PP, pp. 1–1, 07 2019, doi:10.1109/ACCESS.2019.2929480.

[14] Van Rossum, G. y Drake, F. L., Python 3 Reference Manual. Scotts Valley, CA: CreateSpace, 2009.

[15] Google Brain Team, “TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems”. https://www.tensorflow.org/, 2015.

[16] Pytorch Team, “PyTorch”. https://pytorch.org/, 2019.

[17] Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., y Oliphant, T. E., “Array programming with NumPy”, Nature,

vol. 585, no. 7825, pp. 357–362, 2020, doi:10.1038/s41586-020-2649-2.

[18] pandas development team, T., “pandas-dev/pandas: Pandas”, 2020, doi:10.5281/zenodo.3509134.

[19] Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., y Dormann, N., “Stable-baselines3: Reliable reinforcement learning implementations”, Journal of Open Source Software, vol. 6, no. 61, p. 2677, 2021, doi:10.21105/joss.02677.

[20] The Ray Project Contributors, “RLlib: Scalable Reinforcement Learning”. https://www.ray.io/rllib, 2023.

[21] The TF-Agents Team, “TF-Agents: A Reinforcement Learning Library for TensorFlow”. https://www.tensorflow.org/agents, 2023.

[22] Towers, M., Kwiatkowski, A., Terry, J., Balis, J. U., Cola, G. D., Deleu, T., Goulão, M., Kallinteris, A., Krimmel, M., KG, A., Perez-Vicente, R., Pierré, A., Schulhoff, S., Tai, J. J., Tan, H., y Younis, O. G., “Gymnasium: A standard interface for reinforcement learning environments”, 2025, https://arxiv.org/abs/2407.17032.

[23] Riverbank Computing, “PyQt5: Python Bindings for Qt5”. https://riverbankcomputing.com/software/pyqt/, 2023.

[24] Hunter, J. D., “Matplotlib: A 2d graphics environment”, Computing In Science Engineering, vol. 9, no. 3, pp. 90–95, 2007, doi:10.1109/MCSE.2007.55.

Anexos

Anexo A. Repositorio de Código Fuente

El código fuente completo del proyecto descrito en esta memoria está disponible públicamente en el siguiente repositorio de GitHub: https://github.com/alex-itico/Pon Lab

A.1. Estructura de Ramas

Para la gestión del código, se ha adoptado una convención de ramas que separa el trabajo en desarrollo del código estable:

- **main**: Esta rama representa la versión estable y de "producción"del proyecto. Contiene el código que ha sido completamente verificado, es funcional y corresponde a los resultados validados.
- **jorge_dev**: Esta es la rama de desarrollo personal del autor. Contiene el trabajo en progreso, la implementación de nuevas características, experimentos y funcionalidades que aún no han sido finalizadas o integradas en la rama principal. Se recomienda a quienes deseen explorar las últimas funcionalidades en desarrollo, clonar o revisar esta rama.