PROYECTO DE GRADO

Presentado ante la ilustre UNIVERSIDAD DE LOS ANDES como requisito final para obtener el Título de INGENIERO DE SISTEMAS

DESARROLLO DE UN ALGORITMO DE APRENDIZAJE POR REFUERZO PROFUNDO PARA LA GESTIÓN DEL INTERNET TÁCTIL

Por

Br. DUNIA DARISNEY MARQUINA MARQUINA

Tutor: Dr. José L. Aguilar Castro

Junio 2025

DESARROLLO DE UN ALGORITMO DE APRENDIZAJE POR REFUERZO PROFUNDO PARA LA GESTIÓN DEL INTERNET TÁCTIL

Br. DUNIA DARISNEY MARQUINA MARQUINA

Proyecto de Grado — Sistemas Computacionales

Resumen: En el presente proyecto de grado se propone el desarrollo de un algoritmo de aprendizaje por refuerzo profundo (DRL) para optimizar la gestión del Internet Táctil (TI), una tecnología que demanda requisitos de red muy estrictos. Dado que los protocolos clásicos no satisfacen esos requisitos, se proponen técnicas avanzadas de aprendizaje automático para una gestión dinámica y adaptable del TI. Específicamente, se implementaron dos enfoques de DRL: Deep Q-Network (DQN) y Advantage Actor-Critic (A2C), evaluados en un entorno de simulación que emula diversas condiciones de red. Los resultados demostraron que ambos métodos logran un rendimiento parecido, aunque A2C supera a DQN en la reducción de latencia y el manejo de pérdida de paquetes, posicionándose como la alternativa más eficiente, con la capacidad de satisfacer las exigencias de las aplicaciones TI.

Palabras clave: aprendizaje por refuerzo, redes neuronales, baja latencia, internet táctil, aprendizaje automático.

Índice general

Índice de Figuras

Índice de Tablas

Agradecimientos

A Dios y la virgen, por darme salud y ser fuente inagotable de fortaleza y sabiduría, guiándome especialmente en los momentos más difíciles, permitiéndome seguir adelante con determinación y fe.

A mis padres, Ninso y María, por brindarme su incondicional apoyo, cariño y comprensión en cada paso que he dado durante mi vida académica, por siempre estar a mi lado en los momentos más difíciles. Gracias a su amor y dedicación, este logro es una realidad. Son y siempre serán mis mejores guías de vida.

A mi hermana, Sofía, por su ayuda incansable, apoyo incondicional y su motivación constante. Gracias por creer en mí y siempre estar a mi lado.

A cada miembro de mi familia, que con su ejemplo, consejos y motivación han sido una fuente de inspiración y fuerza. Gracias por su apoyo y cariño en este proceso.

A mis amigos y compañeros, en especial a Mari, Eugi, Juli, Jorge y Josmar; con quienes compartí incontables horas de esfuerzo, estudio y trabajo en equipo, así como momentos de risas y experiencias inolvidables. Su apoyo y motivación han sido esencial en este proceso. Mi cariño, admiración y respeto para ustedes.

A mi tutor, el Dr. José Aguilar, cuyos conocimientos, dedicación y compromiso han sido claves para el desarrollo de este proyecto. Gracias por su invaluable guía, paciencia, motivación y apoyo.

A todos aquellos profesores comprometidos y dedicados que han formado parte de mi trayectoria universitaria, gracias por compartir su conocimiento y experiencia, por ser guías en mi formación, y por dejar en mí enseñanzas que llevaré siempre conmigo.

A los profesores Y. Rivero y A. Pinto por la ayuda prestada para la elaboración de esta tesis.

Finalmente, expreso mi gratitud a la ilustre Universidad de los Andes, por su compromiso con la educación y la formación de excelentes profesionales, a pesar de los desafíos actuales. Agradezco también a todos aquellos que contribuyen diariamente al funcionamiento y desarrollo de cada una de sus áreas, pues han hecho posible nuestra formación como profesionales.

Capítulo 1
Introducción

En la actualidad, la interacción con dispositivos a través de interfaces táctiles se está convirtiendo en una parte de nuestra vida cotidiana. Es por ello que, a medida que el uso del Internet Táctil (TI) se expande, es esencial garantizar una experiencia de usuario satisfactoria y la usabilidad de las aplicaciones en tiempo real. Esto requiere de comunicaciones ultra confiables de baja latencia (URLLC), alta disponibilidad y seguridad extrema. Sin embargo, estos requisitos no pueden ser garantizados por los enfoques tradicionales de gestión de la Internet. En este sentido, dado a que el aprendizaje por refuerzo profundo (DRL) ha demostrado eficacia en la toma de decisiones en entornos complejos, se propone el desarrollo de un algoritmo que mediante el uso de técnicas de DRL permita mejorar el rendimiento de las redes para permitir el TI.

1.1. Planteamiento del Problema

La gestión eficiente del TI debe garantizar la transmisión fluida y precisa de las sensaciones táctiles para asegurar la calidad de servicio (QoS) y de la experiencia (QoE) en las aplicaciones de TI. Particularmente, la latencia y la congestión de red pueden afectar significativamente la QoS y la QoE en las aplicaciones. A su vez, los enfoques tradicionales de la capa de transporte, como el protocolo de control de transmisión (TCP), aunque es ampliamente utilizado, no es adecuado para

estas aplicaciones debido a su limitación en el manejo de la congestión y la pérdida de paquetes. Otros protocolos, como el protocolo de datagramas de usuario (UDP), no ofrecen una solución óptima debido a su peor manejo de paquetes y menor confiabilidad.

Cabe mencionar que, en una reciente investigación realizada por Shahzad y colaboradores en 2023 [1], aborda este problema mediante un enfoque de aprendizaje por refuerzo (RL) simple para optimizar la selección entre dos esquemas de codificación de red lineal aleatoria (RLNC), en función de las condiciones de la red. Específicamente, la técnica de RLNC permite que la estrategia de 'almacenar y reenviar', la cual es una de las soluciones actuales de la capa de transporte, que consiste en poner en cola primero los paquetes y luego reenviarlos al destino, sea sustituida por la estrategia de 'computar y reenviar', ya que esta proporciona mayor resiliencia a la perdida de datos y permite a los nodos en la red procesar los paquetes entrantes a medida que los reciben. De esta forma mejora el rendimiento general de la red. El modelo propuesto muestra resultados favorables, ya que gracias al uso de técnicas de RL maximiza el rendimiento general mientras minimiza la latencia de entrega.

Basado en esto, esta tesis propone el desarrollo de un algoritmo utilizando técnicas avanzadas de RL para mejorar el rendimiento de las redes, en particular, para manejar eficientemente problemas como la congestión y la pérdida de paquetes, y así superar las limitaciones de la capa de transporte para aplicaciones táctiles. Lo novedoso de esta propuesta es la incorporación de DRL, una evolución del RL simple. Particularmente, el DRL emerge como una solución prometedora, capaz de adaptarse a condiciones variables y optimizar la gestión de manera autónoma. Esa autonomía se puede usar para hacer un análisis del espacio continuo más eficiente que lo propuesto por Shahzad y colaboradores [1], o para aprender la función de recompensa adecuada para el TI.

1.2. Justificación

La QoS y la QoE de las aplicaciones de TI están estrechamente relacionadas a condiciones de red optimas, factores críticos como la latencia y la congestión de red juegan un papel fundamental en su rendimiento. Una latencia alta puede causar retrasos perceptibles lo que deteriora la experiencia del usuario, ya que estas aplicaciones requieren una respuesta inmediata. Además, la congestión de red puede generar la perdida de paquetes, lo que reduce el rendimiento y aumenta la latencia, con lo cual se afecta de forma negativa la QoS y la QoE.

Por su parte, el protocolo TCP está diseñado para asegurar la fiabilidad en la transmisión de los datos, pero no es adecuado para el manejo de los estrictos requerimientos de alta disponibilidad y baja latencia que demandan las aplicaciones de TI. TCP proporciona una alta confiabilidad a cambio de la sobrecarga de paquetes, lo que introduce latencia adicional debido a sus mecanismos de control de flujo y congestión, los cuales son útiles para asegurar la integridad de los datos, pero generan ineficiencias en aplicaciones sensibles al tiempo. Por estas razones, es necesario considerar y/o proponer soluciones innovadoras que puedan optimizar la gestión de recursos para mejorar la QoS y la QoE en las aplicaciones de TI.

En este sentido, se espera que al aplicar DRL en la gestión del TI resulte en mejoras en la eficiencia y adaptabilidad en los sistemas de red. Para ello, se propone el uso de dos enfoques de DRL, *Deep Q Network* (DQN) y *Advantage Actor-Critic* (A2C), estos algoritmos tienen la capacidad de adaptarse a diversas condiciones del entorno. DQN permite manejar espacios de estados grandes y complejos, lo que lo hace ideal en situaciones donde se deben tomar decisiones secuencialmente. Asimismo, considera las consecuencias futuras de las acciones actuales; esto es

de gran utilidad en el caso del TI donde se deben tomar decisiones continúas y se debe tener en cuenta como afectaran en el rendimiento de la red. Por su parte, A2C utiliza dos redes neuronales, una para la determinar las acciones y otra para evaluar el valor de las acciones, lo que mejora la estabilidad y eficiencia del aprendizaje; esto es crucial en un entorno dinámico como el del TI, donde la incertidumbre puede afectar los resultados.

De esta manera, al llevar a cabo esta investigación se pretende contribuir en el área del DRL por medio de su aplicación a la gestión del TI y, en el avance de la tecnología de redes.

1.3. Objetivos

1.3.1. Objetivo general:

Desarrollar un algoritmo de aprendizaje por refuerzo profundo para la gestión del Internet táctil.

1.3.2. Objetivos específicos:

- Investigar sobre la gestión del TI y las técnicas de DRL.
- Diseñar y desarrollar un algoritmo de DRL para la gestión del TI.
- Desarrollar pruebas experimentales para evaluar el rendimiento del algoritmo propuesto.
- Analizar y comparar los resultados obtenidos con otros enfoques de gestión del TI para identificar ventajas y desafíos del algoritmo.

1.4 Antecedentes

En la actualidad, el TI promete revolucionar la interacción humana con máquinas mediante la transmisión de sensaciones táctiles en tiempo real. Así, resulta de gran importancia la

comprensión de los trabajos relacionados y las tendencias actuales para abordar los desafíos y oportunidades en esta área en constante evolución.

Sharma y colaboradores en 2020 [2] llevaron a cabo un estudio cuyo objetivo fue ofrecer una visión integral del TI y sus avances recientes, presentando un marco que comprende la identificación y análisis de los principales problemas técnicos involucrados, la arquitectura TI, las áreas de aplicación, los tres paradigmas principales de TI, y las tecnologías habilitadoras. Además, han proporcionado algunos temas para futuras direcciones de investigación. El resultado muestra un estudio bastante completo que permite una mejor visión o comprensión de los diferentes aspectos relacionados al TI, destacando que los objetivos más desafiantes de las próximas investigaciones o sistemas 5G están dirigidos a lograr una latencia ultra baja de aproximadamente 1ms y una confiabilidad ultra alta para así generar un mejor rendimiento de la red y proporcionar QoS y QoE en las aplicaciones táctiles.

En el trabajo realizado por Shahzad y colaboradores en 2023 [1], se presenta un sistema de gestión para TI, llamado marco de codificación de red lineal aleatorio selectivo basado en RL (RS-RLNC), con el fin de mejorar el rendimiento de las aplicaciones ejecutándose en el TI. Su objetivo fue diseñar un modelo de gestión que pueda adaptarse a las condiciones de red cambiantes y que minimice el retraso en la entrega de paquetes. Para ello, la solución aplicada usa un esquema de RLNC selectivo basado en RL para tomar decisiones sobre cuándo cambiar entre RLNC de bloque y RLNC deslizante según las condiciones de la red. Los resultados de la simulación muestran que RS-RLNC supera a las soluciones actuales de capa de transporte, y es capaz de minimizar la pérdida de paquetes y mejorar el rendimiento y la QoS en aplicaciones de TI.

Ahora bien, dicha propuesta es basada en un esquema que discretiza las acciones (no lo ve como un espacio continuo), por lo que no realiza un análisis exhaustivo del contexto. Además, presupone el esquema de recompensa. Sin embargo, este estudio proporciona una base sólida a la presente investigación, ya que demuestra cómo la integración de técnicas de RL y codificación de red pueden resultar en una transmisión de datos más eficiente y robusta. De esta forma, sobre estas bases se pueden incorporar técnicas que permitan mejorar el mecanismo de aprendizaje.

Ramírez y colaboradores en 2022 [3] presentan el desarrollo de un modelo de despacho económico hidrotérmico basado en DRL, el cual considera la incertidumbre en los flujos de agua y la demanda de energía. La finalidad de este estudio es minimizar los costos de suministro eléctrico utilizando de manera eficiente los recursos energéticos disponibles. Para ello, formulan el problema como un proceso de decisión de Markov (MDP) y proponen varios enfoques de DRL, como los algoritmos DQN y A2C, los cuales permiten abordar problemas de optimización complejos que involucran incertidumbre. DQN se utiliza para aprender una política óptima en un espacio de acción discreto, el cual utiliza una red neuronal para aproximar la función Q actualizando sus valores mediante un proceso de aprendizaje basado en la experiencia acumulada, lo que permite adaptarse a diferentes escenarios hidrológicos y demandas energéticas, y así aprende a seleccionar las acciones que minimizan el costo de suministro. Por su parte, A2C se utiliza para el caso continuo, este aprende directamente del espacio de observación por medio de un método de gradiente de políticas, lo que mejora la eficiencia del aprendizaje y así permite una adaptación más rápida a cambios en las condiciones del entorno, lo cual optimiza el despacho económico hidrotérmico. Los resultados muestran que los métodos propuestos pueden aprender políticas robustas que manejan diferentes escenarios de afluencia y demanda, con lo cual se demuestra que supera las limitaciones de los métodos deterministas tradicionales. En definitiva, la

importancia de este enfoque radica en la capacidad de mejorar la toma de decisiones en la planificación y operación de sistemas energéticos, lo cual facilita la gestión de riesgos.

Aunque este estudio se enfoca en el problema del despacho económico hidrotérmico, los conceptos y técnicas de DRL desarrollados podrían ser aplicables en otras áreas que requieren la gestión de decisiones secuenciales bajo incertidumbre, como es el caso de la gestión del TI, donde la incertidumbre de variables claves como las condiciones de red, son factores críticos que deben ser gestionados para generar políticas operativas más robustas y confiables, y de esta manera, mejorar el rendimiento de las redes.

Li y colaboradores en 2018 [4] plantean en su estudio el uso de DRL como una solución prometedora capaz de gestionar de forma eficiente los recursos en segmentación de redes 5G. Su objetivo principal es investigar y demostrar si el DRL puede optimizar la asignación de recursos tanto en el acceso radioeléctrico como en el núcleo de la red, y así superar las limitaciones de los métodos tradicionales, como es el caso de la asignación equitativa o los algoritmos de predicción manual. Para ello, la solución propuesta consiste en aplicar algoritmos de Deep Q-Learning (DQL) que combinan redes neuronales con aprendizaje por refuerzo, lo que permite al sistema aprender de forma dinámica la mejor estrategia de asignación de recursos adaptándose en tiempo real a la variabilidad de la demanda. Los resultados demuestran que DRL alcanza una mayor eficiencia en el uso de recursos en comparación con los métodos tradicionales. A pesar de ello, reconoce que aún existen limitaciones como la necesidad de una gran cantidad de datos para entrenar los modelos. No obstante, este estudio evidencia que DRL es una herramienta prometedora para la gestión eficaz de recursos.

Ssengonzi y colaboradores en 2022 [5] presentan una investigación sobre la aplicación de DRL en la gestión y optimización de la segmentación de redes en 5G y redes futuras. Su objetivo principal es abordar los vacíos que existen en la asociación entre DRL y la gestión que plantea la segmentación y virtualización de redes en entornos 5G. Con este fin, realiza una revisión detallada de los conceptos fundamentales de DRL, los principios de segmentación y virtualización de redes, los desafíos actuales, las soluciones propuestas y las posibles líneas de investigación. Dentro de las soluciones planteadas destacan los algoritmos basados en DQL; estos se implementan en diferentes niveles, como en la asignación de recursos a segmentos, el control de admisión de nuevos segmentos, la reconfiguración de segmentos, la gestión de la movilidad y seguridad de los segmentos, entre otros. Cabe mencionar que los trabajos evaluados demuestran que el DRL ofrece soluciones de gran potencial en la gestión de redes 5G. Sin embargo, aún existen desafíos por resolver, como el uso limitado que tiene en la previsión y predicción de tráfico, y la necesidad de continuar con la investigación para mejorar la estabilidad y escalabilidad de estos modelos.

En una reciente investigación llevada a cabo por Kokkinis y colaboradores en 2025 [6], proponen un sistema basado en DRL para la administración dinámica de recursos de radio en el TI, concretamente para aplicaciones de teleoperación video-háptica. Para ello, utilizan el método Soft Actor Critic (SAC), el cual se caracteriza por su eficacia en entornos dinámicos. Este utiliza las redes neuronales actor y crítico que son las encargadas de estabilizar y mejorar las actualizaciones de políticas durante el proceso de entrenamiento, además, incorpora un coeficiente de entropía que ayuda a mantener la estabilidad entre la exploración y explotación. Este enfoque permite que un agente aprenda mediante la retroalimentación de recompensas a asignar los recursos de radio entre los flujos de datos hápticos y de video considerando los requisitos de latencia, pérdida de paquetes, tasa de datos y la sincronización entre ambas modalidades, de esta

manera maximizar la satisfacción del usuario en escenarios de red cambiantes. Los resultados mostraron que el marco propuesto permite una gestión eficiente de la sincronización video-háptica en condiciones variables de red, con lo cual logra un aumento significativo en la satisfacción del usuario en comparación con los métodos convencionales.

En definitiva, este estudio no solo muestra la utilidad del DRL en la gestión de redes de teleoperación video-háptica, la cual se considera una rama tecnológica del TI, sino que, además, su enfoque en la sincronización y adaptabilidad a condiciones dinámicas posibilita sistemas autónomos más robustos. Sin embargo, no se compara con otros algoritmos de DRL, lo que impide identificar las ventajas especificas del SAC en este caso.

1.5 Organización de la tesis

En el capítulo 2 se presenta el marco teórico, con los conceptos más importantes relacionados al trabajo y el modelo base. Luego, en el capítulo 3 se describe el enfoque de gestión inteligente del TI donde se explica la formulación del problema como un proceso de decisión de Markov, las estrategias de gestión basadas en DRL, y las métricas a utilizar. En el capítulo 4 se explica la implementación y el análisis de los resultados. Finalmente, en el capítulo 5 se presentan las conclusiones y recomendaciones para trabajos futuros.

Capítulo 2
Fundamentación Teórica

En este capítulo, se explorarán los fundamentos teóricos que sustentan este estudio sobre la gestión del TI mediante algoritmos de DRL. Se abordarán conceptos claves relacionados al tema necesarios para comprender la propuesta de este trabajo, como los conceptos de aprendizaje automático, RL, DRL, entre otros.

2.1 Internet Táctil (TI)

El término “Internet táctil” fue definido por la Unión Internacional de Telecomunicaciones (UIT) en un informe en agosto de 2014, como la red que permitirá la interacción háptica, es decir, la percepción y manipulación de objetos mediante el tacto. Además, señala que el carácter del TI es definido por la latencia extremadamente baja combinada con alta disponibilidad, confiabilidad y seguridad [7].

Otra definición es la dada por la IEEE P1918.1, en la cual lo describe como “Una red o red de redes para acceder, percibir, manipular o controlar de forma remota objetos o procesos reales o virtuales en tiempo real percibido por humanos o máquinas” [8]. También, es considerada como la siguiente fase de la evolución del internet de las cosas (IoT), la cual ha emergido como un

paradigma de comunicación crucial que ofrece una solución eficiente para mejorar las interacciones entre humanos y máquinas (H2M) y entre máquinas (M2M), facilitando las comunicaciones hápticas como aplicación principal, proporcionando un canal en tiempo real para transmitir sensaciones táctiles y movimientos [1].

En particular, la TI se describe como una red que debe cumplir con ciertos requerimientos técnicos claves como [2]:

- Conectividad Ultrasensible: Necesita una latencia de extremo a extremo muy baja, de alrededor de 1ms.
- Conectividad Ultra Confiable: Requiere alta confiabilidad en la red que garantice un rendimiento estable bajo diversas condiciones.
- Inteligencia de borde distribuida: debe implementar técnicas de inteligencia artificial (IA) en el borde de las redes inalámbricas, que ayuden o permitan predecir y calcular las acciones futuras de los usuarios.
- Transmisión y procesamiento de datos táctiles: Necesita mecanismos de codificación táctil que permitan transmitir información háptica a través de redes de conmutación de paquetes.
- Seguridad y privacidad: Requiere que la autenticación sea parte integral de la transmisión física, ya que el método actual de separar la autenticación de la transmisión física no permite alcanzar una baja latencia de extremo a extremo.

Estos requisitos aseguran que las aplicaciones de TI funcionen de manera eficiente. Sin embargo, demandan una revisión de los protocolos tradicionales.

2.2 Introducción al Aprendizaje Automático (ML)

El aprendizaje automático o aprendizaje de máquina es un área de la IA, que comprende el proceso mediante el cual las máquinas adquieren conocimiento y mejoran su rendimiento a partir de datos o experiencia, sin la necesidad de la intervención humana directa. Es decir, en lugar de programar las reglas o algoritmos, el aprendizaje automático permite que a través de ejemplos de datos las computadoras aprendan patrones y relaciones.

Una de las definiciones más reconocidas es la de Arthur Samuel en 1959, que dice que el "*Aprendizaje automático es el campo de estudio que da al computador la habilidad de aprender sin haber sido explícitamente programado para ello"* [9].

El aprendizaje automático comprende principalmente tres paradigmas de aprendizaje: aprendizaje supervisado, aprendizaje no supervisado y aprendizaje por refuerzo. El aprendizaje supervisado utiliza datos preclasificados con etiquetas que especifican sus características. Este principalmente se usa para clasificación y regresión. El aprendizaje no supervisado procesa grandes cantidades de datos sin etiquetas identificando patrones y similitudes sin intervención humana. Este se usa para agrupar datos. El aprendizaje por refuerzo se describe más adelante.

En general, los algoritmos de aprendizaje automático basados en datos al entrenarse generan un modelo de conocimiento (ya sea predictivo, de diagnóstico, de optimización, prescriptivo, entre otros), que dependerá del tipo de problema al que se esté aplicando. Sin embargo, cualquiera que sea el problema la aplicación del mismo requiere la realización de una serie de etapas [9]:

- Preprocesamiento de datos: Implica la limpieza y transformación de los datos para que puedan ser utilizados por el algoritmo de aprendizaje automático.

- Separación en conjunto de entrenamiento y pruebas: El conjunto de datos (dataset) se divide en dos subconjuntos, el de entrenamiento para entrenar y estimar los parámetros del modelo y, el de prueba para probar el modelo de conocimiento construido.

- Configuración del algoritmo: Se definen los hiperparámetros del algoritmo de aprendizaje con valores que se deben ajustar adecuadamente, como el número de épocas y la tasa de aprendizaje en el caso del aprendizaje profundo.

- Entrenamiento del modelo: Consiste en construir el modelo de conocimiento con el algoritmo de aprendizaje, a su vez optimizando sus hiperparámetros.

- Validación: Consiste en probar el modelo de conocimiento usando el conjunto de datos de pruebas, para determinar numéricamente qué tan efectivo es el modelo de conocimiento. Para ello, existen diferentes métricas de calidad, dependiendo del tipo de problema.

2.3 Aprendizaje por refuerzo (RL)

El RL es considerado el tercer paradigma del ML. Este consiste en que un agente mediante la interacción con un entorno sea capaz de percibir el estado del mismo, y aprenda a tomar decisiones que maximicen una recompensa numérica. Para ello, el agente debe descubrir qué acciones proporcionan una mayor recompensa a través del ensayo y error. Dichas acciones no solo afectan la recompensa inmediata, sino también los estados futuros y las recompensas posteriores [10].

2.3.1 Bases de RL

Un sistema de RL incluye elementos clave como una política, una señal de recompensa, una función de valor y, opcional, un modelo del entorno. La política guía el comportamiento del agente, ya que puede ser un conjunto de reglas o asociaciones entre estados y acciones. La señal

de recompensa indica los efectos positivos o negativos para el agente, similar a las experiencias de placer o dolor en sistemas biológicos; esta es proporcionada por el entorno, lo cual el agente intenta maximizar a largo plazo. La función de valor, por su parte, evalúa las recompensas a largo plazo. Finalmente, un modelo del entorno puede predecir el comportamiento del entorno, permitiendo inferencias y la planeación de acciones. Juntos estos elementos permiten que el agente aprenda y tome decisiones óptimas para maximizar las recompensas acumuladas [10].

El RL emplea la estructura formal de los procesos de decisión de Markov (MDP), que permite describir la interacción entre un agente que aprende y su entorno mediante el uso de estados, acciones y recompensas. Esta interacción puede observarse en la Figura 2.1.

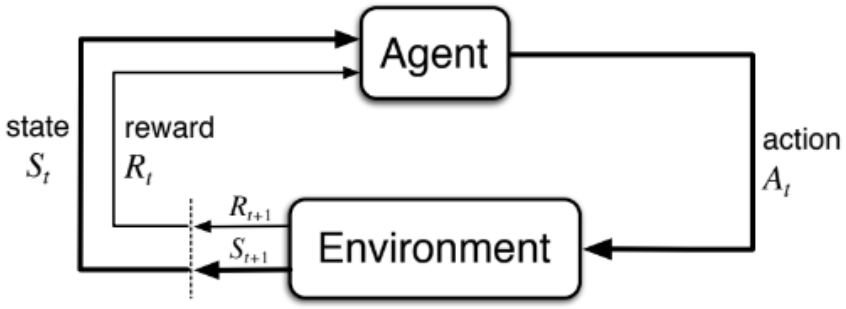


Figura 2.1 Interacción de un agente con su entorno en un modelo RL. Fuente: [10]

Donde:

- Agente (Agent): Es el encargado de interactuar con el entorno, hace observaciones, ejecuta acciones, y recibe recompensas por ello.
- Entorno (Environment): Comprende lo que está fuera del agente, es decir, con lo que interactúa.
- Acción (Action): Son los ajustes que el agente puede realizar al entorno.

- Estado (State): Son las condiciones u observaciones que el entorno proporciona al agente.
- Recompensa (Reward): Es un valor numérico que obtiene el agente al ejecutar una acción sobre el entorno y, que busca maximizar con el tiempo. Puede ser positiva o negativa, para indicar que tan buena fue la acción ejecutada.

De esta manera, en cada paso de tiempo t, el agente observa el estado S_t de un espacio de estados S del entorno, y selecciona una acción A_t del espacio de acciones A(s) siguiendo una política $\pi(a_t/s_t)$. Como consecuencia de ejecutar dicha acción, un paso de tiempo después, el agente recibe una recompensa numérica $R_{t+1} \in R \subset \mathbb{R}$ y se encuentra ante un nuevo estado S_{t+1}. Esta interacción da como resultado la siguiente trayectoria:

$$S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, ...$$

Cabe resaltar que en un MDP, la probabilidad de cada posible valor para S_t y R_t dependen únicamente del estado y acción inmediatamente anterior, S_{t-1} y A_{t-1}.

Por otro lado, la mayoría de los algoritmos de RL implican la estimación de funciones de valor, las cuales miden que tan bueno es estar en un estado dado. Esta valoración se basa en las recompensas futuras o retorno esperado, que dependen de las acciones del agente, las cuales son guiadas por políticas específicas. Una política es un mapeo de estados a probabilidades de seleccionar cada acción posible [10]. Ejemplo, si un agente sigue la política π en el tiempo t, entonces $\pi(a|s)$ es la probabilidad de que $A_t = a$ si $S_t = s$.

En general, la solución a un problema de RL consiste en identificar una política que maximice la recompensa a largo plazo, satisfaciendo la función de estado-valor óptima denotada como v_* y definida como:

$$v_*(s) \doteq max_\pi v_\pi(s) \quad (1)$$

Asimismo, la función óptima de acción-valor $q*$ que representa el valor óptimo que se logra al tomar la acción a en el estado s siguiendo la política óptima, y está definida como:

$$q*(s,a) \doteq max_\pi q_\pi(s,a) \quad (2)$$

Y en términos de $q*$ quedaría:

$$q*(s,a) = \mathbb{E}[R_{t+1} + \gamma\, v_*(s_{t+1}) | S_t = s, A_t = a] \quad (3)$$

2.3.2 Algoritmo Q-Learning

Este algoritmo fue uno de los primeros avances del RL, introducido por Watkins en 1989, y es uno de los más utilizados en este campo [10]. El objetivo de Q-Learning es encontrar una política de acción óptima para maximizar la recompensa total a lo largo del tiempo. Se define como:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma max_a Q(S_{t+1}; a_t) - Q(S_t, A_t)] \quad (4)$$

Donde S y A es el conjunto de estados y acciones, respectivamente, R es la función de recompensa, α es el parámetro de aprendizaje, γ es el factor de descuento, S_t representa el estado actual, S_{t+1} el estado siguiente al estado S_t al ejecutar la acción a_t, y $Q(S_{t+1}; a_t)$ es la mejor estimación de Q para una acción a ejecutar en el estado S_{t+1}.

2.4 Aprendizaje por refuerzo profundo (DRL)

El DRL combina redes neuronales profundas con algoritmos de RL para aprender de datos complejos y/o de mayor dimensión, ya que las redes neuronales son capaces de extraer características complejas que permiten resolver problemas de alta dimensionalidad o de estados continuos [11]. Estos métodos utilizan las redes neuronales para representar el estado o para aproximar alguno de los componentes del RL como la función de valor ($v\hat{}(s; \theta)$), la política ($\pi(a|s; \theta)$), el modelo del entorno (función de transición de estado), o la función de recompensa, donde los parámetros θ corresponde a los pesos de las redes neuronales [11].

En este proyecto se plantea implementar dos algoritmos de DRL: DQN y A2C.

2.4.1. Deep Q-Network (DQN)

DQN es un algoritmo que consiste en utilizar dos redes neuronales para aproximar la función Q, denominadas red objetivo (Target Network) y red Q (Q-Network), las cuales se encargan de estimar la recompensa futura y el valor de la función Q, respectivamente. Se denotan como red objetivo a $\theta^{Q'}$ y red Q a θ^{Q}, respectivamente. Entonces, la regla de actualización de DQN es:

$$Q_{n+1}(s_t, a_t) = Q_n(s_t, a_t) + \alpha[R(s_t, a_t, s_{t+1}) + \gamma max_{a\epsilon\, At+1} Q_n(s_{t+1}, a) - Q_n(s_t, a_t)] \quad (5)$$

Donde θ^{Q} estima el valor de la función Q y es equivalente a $Q_n(s_t, \alpha_t)$. Por su parte, $\theta^{Q'}$ estima la recompensa futura de tomar una acción a y, al agregar esta estimación a la recompensa actual se obtiene una estimación de la recompensa total en el tiempo t. Entonces, al considerar la red neuronal y reescribir $R(s_t,\alpha_t,s_{t+1}) + \gamma \cdot max_{a\epsilon At+1}\, Q_n(s_{t+1}, \alpha)$, se tiene la siguiente expresión equivalente:

$$R(s_t, a_t, s_{t+1}) + \gamma max_{a\epsilon\, A_{t+1}} \theta^{Q'} \quad (6)$$

Ahora, para calcular el error por medio de una función de pérdida (Loss) se utilizan como argumentos la predicción de θ^{Q} y la suma entre la recompensa y la predicción de $\theta^{Q'}$, con lo cual, suponiendo que la función de perdida es la ecuación de mínimos cuadrados (MSE), el cálculo del error quedaría:

$$L(\theta) = \left[\left(R(s_t, a_t, s_{t+1}) + \gamma . \theta^{Q'} - \theta^{Q}\right)^2\right] \quad (7)$$

Por otro lado, la red Q y la red objetivo generalmente tienen la misma arquitectura, por ello solo se entrena la red Q y usando esta se actualiza la red objetivo a intervalos regulares para estabilizar el proceso de aprendizaje [12]. Considerando esto, en la Figura 2.2 se muestra la arquitectura DQN.

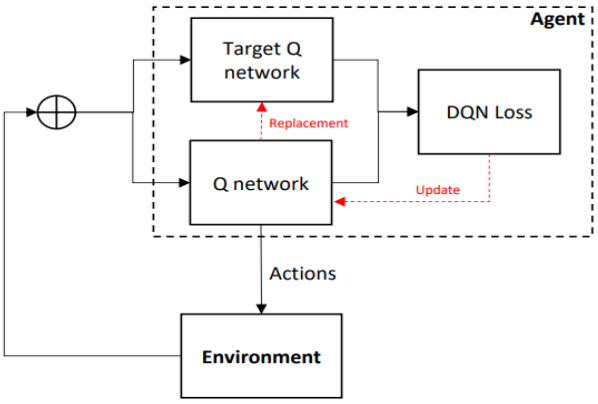


Figura 2.2 Arquitectura DQN. Fuente [3]

2.4.2 Advantage Actor Critic (A2C)

Los algoritmos actor-crítico están basados en los métodos de descenso de gradientes, es decir, estos aprenden directamente del espacio de observación mediante el uso de técnicas que

permiten ajustar continuamente la política del agente para maximizar la recompensa acumulada en el tiempo. Cabe mencionar que estos son uno de los más utilizados en un marco RL [3].

El actor-crítico con ventaja utiliza dos redes neuronales, una red actor que decide que acción tomar en un estado dado, y una red critico que evalúa que tan buena fue tomar esa acción. Ella calcula la diferencia entre el valor estimado de una acción y el valor esperado del estado actual, lo cual ayuda a disminuir la varianza en las estimaciones y mejorar el aprendizaje. Se calcula con la siguiente expresión:

$$A(s_t\,, a_t\,) = r_{t+1} + \gamma \cdot V(s_{t+1}\,) - \, V(s_t) \tag{8}$$

Donde $A(s_t, \alpha_t)$ estima el interés de tomar la acción α_t en el estado s_t, r_{t+1} es la recompensa obtenida después de tomar la acción α_t en el estado s_t, γ es el factor de descuento, $V(s_{t+1})$ representa el valor estimado del siguiente estado s_{t+1} y $V(s_t)$ es el valor estimado del estado actual s_t. Cabe mencionar que, si $A(s_t, \alpha_t)$ es positivo indica que tomar la acción α_t en el estado s_t es mejor que lo esperado, y si es negativo es peor que lo esperado.

En general, el algoritmo actor-critico combina el gradiente de política para el actor y la función de valor para el crítico. La expresión del gradiente de política es:

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=0}^{N} \nabla_\theta log\, \pi_\theta(a_i|s_i) \cdot A(s_i, a_i) \tag{9}$$

Donde $J(\theta)$ es el rendimiento esperado de la política basada en los parámetros θ, $\pi_\theta(\alpha|s)$ es la función de política que indica la probabilidad de elegir la acción α en el estado s, N es el número total de experiencias muestreadas, $A(s_i, \alpha_i)$ es la ventaja de tomar la acción α_i en el estado s_i, i

representa el índice de la muestra y $\nabla_\theta log\ \pi_\theta(a_i|s_i)$ es la dirección en la que se deben modificar los parámetros θ para mejorar la política. Por otro lado, la expresión de la función de valor es:

$$\nabla_w J(w) \approx \frac{1}{N} \sum_{i=1}^{N} \nabla_w (V_w(s_i) - Q_w(s_i, a_i))^2 \quad (10)$$

Donde $\nabla_w J(w)$ es el gradiente de la función de pérdida basada en los parámetros w, $V_w(s_i)$ es la estimación del crítico sobre el valor del estado s_i con parámetro w, $Q_w(s_i, a_i)$ es la estimación del crítico sobre el valor de la acción α_i en el estado s_i, N es el número de muestras utilizadas, i el índice de la muestra y $\nabla_w (V_w(s_i) - Q_w(s_i, a_i))^2$ mide el error entre la estimación del valor del estado V_w y el valor de acción Q_w, lo que determina en qué dirección ajustar los parámetros w para mejorar su evaluación de valores minimizando errores.

Ahora, para la actualización del actor se utiliza el ascenso de gradiente, lo que significa que modifica sus parámetros para maximizar las recompensas futuras de la siguiente manera:

$$\theta_{t+1} = \theta_t + \alpha \nabla_\theta J(\theta_t) \quad (11)$$

Donde α es la tasa de aprendizaje del actor y t es el paso de tiempo dentro de un episodio. En cuanto a la actualización del crítico, se utiliza el descenso de gradiente, lo que implica modificar sus parámetros para reducir el error en sus estimaciones y así mejorar sus predicciones sobre el valor de los estados y acciones. Se actualiza de la siguiente forma:

$$w_t = w_t - \beta \nabla_w J(w_t) \quad (12)$$

Donde w son los parámetros de la red crítico y β es la tasa de aprendizaje del crítico.

El esquema actor-crítico proporciona una mejor función de puntuación, ya que en lugar de esperar hasta el final del episodio permite realizar una actualización en cada paso del proceso, similar al aprendizaje por diferencia temporal (TD Learning) [13].

En general, en este enfoque se observa que el actor se encarga de seleccionar la acción a tomar y de estimar $\nabla_\theta log\ \pi_\theta(a_i|s_i)$ en la ecuación (9). Por otro lado, el crítico se encarga de estimar $V_w(s_i)$ y $Q_w(s_i, a_i)$ de la ecuación (10), evaluando que tan buena fue la acción tomada y cómo debe ajustarse. La arquitectura del actor-crítico se muestra en la Figura 2.3.

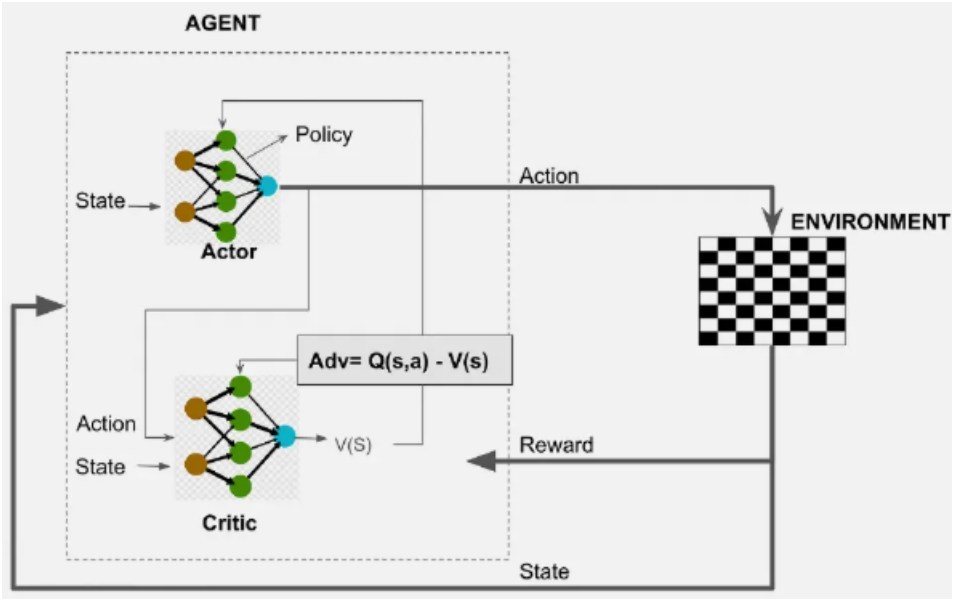


Figura 2.3 Arquitectura Actor-Critico (tomado de [18])

2.5 Modelo RL base de nuestra propuesta

El diseño de RL del sistema de RS-RLNC para la gestión del TI propuesto en [1], comprende:

- Un espacio de acción, donde esta predeterminado RLNC deslizante, y la acción consiste en continuar utilizando RLNC deslizante o cambiar a RLNC en bloque, o viceversa.
- Un espacio de estado con dos variables, la congestión incipiente (IC_t) y la tasa estimada de error del canal (pe).
- Una función de recompensa, definida de la siguiente forma:

$$r_t = \frac{goodput_t}{rTT}$$

Donde goodput es el rendimiento útil de la red, definido como la cantidad de datos útiles transmitidos correctamente por unidad de tiempo y rTT es el tiempo de ida y vuelta de los paquetes entregados en el momento t.

El algoritmo de RL implementado en este modelo fue el algoritmo Q-learning debido a su técnica sencilla de actualización de iteración de valores, junto al método ϵ-greedy para la exploración durante la fase de aprendizaje inicial. Dicho método toma una acción aleatoria a con probabilidad ϵ y una acción avara (greedy) dada por a= max $Q_t(a)$ con probabilidad 1- ϵ.

La evaluación del algoritmo se realiza en términos del rendimiento, latencia y complejidad de decodificación, utilizando las métricas de rendimiento útil acumulado, el retardo promedio móvil (MA) y la recompensa, ya que esta relación entre el buen rendimiento y el rTT hace referencia a mejorar el rendimiento y mantener un valor bajo de rTT.

Los resultados de la simulación muestran que RS-RLNC supera las soluciones actuales de la capa de transporte mostrando un mejor rendimiento cuando las condiciones de la red cambian, minimizando la pérdida de paquetes.

Capítulo 3

Enfoque de Gestión Inteligente del Internet Táctil

En este capítulo, se formaliza el problema de decisión de Markov aplicado al contexto de la gestión del TI, detallando los elementos fundamentales que lo conforman. Se describe al agente encargado de tomar decisiones, las posibles acciones que puede ejecutar, y el entorno en el que actúa. De igual manera, se presentan los enfoques de gestión basados en DRL y las métricas de calidad utilizadas para evaluar el rendimiento de los algoritmos y del sistema.

3.1 Formulación del problema como un proceso de decisión de Markov

3.1.1 Agente (Agent)

El agente es el responsable del aprendizaje, la toma de decisiones y la interacción con el entorno. En este esquema, el agente se comporta como un controlador de red. Para ello, basándose en el estado actual determinado por las condiciones de la red ejecuta una acción (a) cuya finalidad es mejorar el rendimiento de la misma. El entorno procesa esta acción para optimizar la red, con lo cual alcanza un nuevo estado (Next State), y proporciona una recompensa al agente. Esta información le permite al agente ajustar su política y mejorar sus decisiones futuras.

El diseño del agente (agente DRL) se lleva a cabo utilizando algoritmos de DRL, específicamente los descritos en el capítulo 2, DQN y A2C.

3.1.2 Acciones (Action)

Ajustar el número de segmentos en que se fragmenta un paquete o flujo de datos permite adaptar la transmisión a las condiciones de la red. Un valor alto genera mayor tolerancia a errores, lo que es ideal en redes con alta tasa de pérdida, pero aumenta la latencia. A su vez, un valor bajo minimiza el procesamiento, pero lo hace vulnerable a pérdidas de paquetes. En base a esto, el agente debe aprender a ajustar dinámicamente el número de segmentos para adaptarse a las condiciones variables de la red, y así minimizar la latencia y maximizar el rendimiento. Con este fin, se propone que el agente ajuste el número de segmentos entre -2 y 2, es decir que pueda seleccionar entre mantener, o aumentar o disminuir en 1 o 2 segmentos. Ahora, de acuerdo a los algoritmos a utilizar, se define la acción para el caso continuo y discreto:

- Para el caso continuo, el agente genera un valor dentro del rango [-2,2] que se utiliza para modificar la cantidad de segmentos. Así, si el valor es positivo se incrementa el número de segmentos y si es negativo se disminuye. Por ejemplo, si el valor es 1 se incrementa en 1 el número de segmentos.
- Para el caso discreto, el conjunto de acciones que el agente puede elegir es {-2, -1, 0, 1, 2}. Por ejemplo, en el caso de seleccionar 0 se mantiene el número de segmentos.

3.1.3 Entorno (Environment)

El entorno es con lo que interactúa el agente. Este representa la red y permite la interacción de las entradas, que es la acción, y las salidas, que corresponde al estado y la recompensa. Estos se describen a continuación.

3.1.3.1 Estados (State)

Los estados corresponden a las diferentes condiciones de la red que el entorno da al agente, y se representan como combinaciones de las siguientes variables:

- Ancho de banda (BW), el cual representa la capacidad de transmisión de datos disponible o que la red puede manejar. Un mayor ancho de banda permite transmitir más datos en menor tiempo. Esta varía entre 10 - 500 Mbps.

- Tasa de error del canal (pe), que indica el número de bits recibidos de un flujo de datos con errores. Esta varía de forma aleatoria entre 0 - 50 %.
- Número de segmentos del paquete (ns), el cual indica la cantidad de segmentos en que se divide la transmisión de datos. Un número elevado de segmentos puede aumentar la sobrecarga y afectar el rendimiento.
- Índice de congestión (IC), representa el nivel de saturación de la red. Una alta congestión puede generar retrasos y la pérdida de paquetes, con lo cual afecta la QoS y la QoE.

Estas variables son fundamentales ya que permiten identificar problemas en la red. Por lo tanto, ofrecen la información necesaria para que el agente decida la acción a ejecutar. De esta manera, el espacio de estados queda así:

$$S_t = (Bw, pe, ns, IC_t) \tag{13}$$

Donde Bw es el ancho de banda (por sus siglas en inglés, bandwidth), pe es la tasa de error, ns el número de segmentos, y IC_t es el índice de congestión en el tiempo t.

3.1.3.2 Recompensa (Reward)

La recompensa es una salida del entorno, específicamente un valor numérico que el entorno da al agente por tomar una acción en un estado. El agente busca obtener la mayor recompensa posible, identificando a su vez la política que minimice la función objetivo. Para el contexto del proyecto, la función de recompensa se plantea de forma similar a [1]:

$$r_t = \frac{goodput_t}{rTT} \tag{14}$$

Donde goodput es el rendimiento útil de la red y rTT es el tiempo de ida y vuelta de un subconjunto de paquetes entregados en el tiempo t. Estas métricas son de gran relevancia ya que se busca aumentar la cantidad de datos útiles transmitidos y mantener un tiempo de envío y respuesta bajo para mejorar la eficiencia de la red. Particularmente, goodput será definido en la sección 3.3.2.

3.1.4 Exploración – Explotación

La exploración y la explotación son conceptos fundamentales en el diseño de algoritmos RL, ya que ayudan al agente a decidir cuándo probar nuevas acciones para descubrir estrategias más efectivas en busca de mejores recompensas, y cuándo aprovechar lo que ya sabe para elegir la mejor acción que le permita obtener beneficios inmediatos. En este trabajo se utiliza el método épsilon – greedy y el de exploración por distribución gaussiana.

3.1.4.1 Epsilon – Greedy

El método **ε** - greedy es una estrategia común en algoritmos de RL como Q-Learning. Este funciona eligiendo la mejor acción con probabilidad 1 – **ε,** es decir explota el conocimiento

existente para maximizar la recompensa [10]. Así mismo, con probabilidad ε selecciona una acción aleatoria, permitiendo descubrir nuevas estrategias (ver algoritmo en Tabla 3.1). El parámetro ε varía entre 0 y 1, y se reduce con el tiempo para favorecer progresivamente la explotación sobre la exploración. Este esquema se utiliza en DQN.

Tabla 3.1.

Algoritmo ε-greedy para la elección de exploración y explotación del agente

Algoritmo 1 Algoritmo *ε-greedy para seleccionar exploración y explotación*

Entrada: valor aleatorio, valor de decaimiento, valor mínimo.

Parámetros: parámetro ε de *exploración inicial*

Salida: decisión si *explora* o *explota*

```
Initialize r = 0, ε = 1
r = rand(0,1)
ε = ε * decay
ε = max(ε, ε_min)
if  r < ε:
    explore
else:
    exploit
```

3.1.4.2 Exploración por distribución Gaussiana

Este método consiste en elegir acciones aleatorias utilizando una distribución normal de probabilidad, y se implementa en algoritmos con un espacio de acción continuo. Por lo tanto, este se utiliza en el A2C, en el cual la red actor genera acciones aleatorias alrededor de la media (μ) con una dispersión controlada por la varianza (σ^2). Al inicio, la varianza es alta y permite explorar

diversas acciones, pero a medida que el algoritmo aprende, la media se acerca a la mejor acción y la varianza disminuye favoreciendo la explotación.

3.2. Estrategias de Gestión basadas en DRL

En esta sección se explican los detalles de los algoritmos utilizados en este proyecto.

3.2.1 DQN

En este caso, el agente aprende de sus experiencias acumuladas. Para ello, almacena las experiencias en un buffer o memoria en cada episodio $e_t = (S_t, A_t, R_t, S_{t+1}, \text{done})$, y luego selecciona una muestra aleatoria para continuar con el entrenamiento de la red Q. Para el cálculo de la estimación de la recompensa futura se usa la red objetivo, y se actualizan los pesos de la red principal a través de descenso de gradiente. Además, cada 100 iteraciones se actualizan los pesos de la red objetivo con la red entrenada. Por otro lado, para la exploración y explotación se utiliza el método **ε** – greedy que se mencionó anteriormente. Para este algoritmo, los hiperparámetros utilizados se muestran en la Tabla 3.2, y el algoritmo DQN desarrollado se detalla en la Tabla 3.3, donde se puede ver que en el paso 10 se selecciona la acción usando **ε** –greedy, en el paso 11 se ejecuta la acción, y se observa la recompensa y el nuevo estado. Asimismo, entre los pasos 17 al 21 se actualiza el valor Q(s,α), y en los pasos 23 y 24 se ajustan los parámetros del modelo utilizando descenso de gradiente, minimizando el error cuadrático medio (MSE).

Tabla 3.2.

Hiperparámetros DQN

Parámetro	***Valor***
1: Optimizador	*ADAM*
2: Maximo número de pasos:	*3495*
3: Tasa de aprendizaje (α)	*0.01*
4: Factor de descuento (ɣ)	*0.9*
5: Tasa de exploración (ε)	*1.0*
6: Factor de decaimiento para ε	*0.995*
7: Valor mínimo de ε	*0.01*
8: Tamaño del buffer ()	*2000*
9: Número de muestras por lote	*32*
10: Número de capas ocultas	*2*
11: Número de unidades por capa	*16*
12: Número de salidas	*1*
13: Función de activación de capas ocultas	*LeakyReLU*
14: Función de activación de salida	*Lineal*

Tabla 3.3.

Algoritmo DQN

Algoritmo 2 *Algoritmo DQN*

Entrada: *estado, acción*
Parámetros: *parámetro de aprendizaje, factor de descuento, parámetro de exploración*
Salida: *política óptima*

Initialize Tactile Internet model
Initialize replay memory $\mathcal{D}$ to capacity N
Initialize Q network (θ)
Initialize target Q network (θ') with θ weights

for *each episode I* ***do****:*
  Initialize state s_0
  while *t= 1…T_1 in episode I* ***do****:*
    action = ***call*** *ε-greedy Algorithm*
    execute action a_t and observe reward r_t and state s_{t+1}
    store transition (s_t, a_t, r_t, s_{t+1}, done) in $\mathcal{D}$
    Update $s_t \leftarrow s_{t+1}$

    if *size batch < $\mathcal{D}$ size:*
      sample random batch (s_j, a_j, r_j, s_{j+1}, done) uniformly from $\mathcal{D}$
      if *done = True:*
        $Q_j(s_t, a_t) = r_j$
      else:
        $Q_j(s_t, a_t) = r_j + \gamma \cdot max_a(\theta'(s_{j+1}))$
      end if
      Update *model with Gradient Descent*
      loss $\leftarrow$ $MSE\ (y_j, \theta(s_j))$
      every ω step update $\theta' \leftarrow \theta$
    end if
  end while
end for

3.2.2. A2C

Este algoritmo combina una red actor que genera una distribución de probabilidad (media μ y desviación estándar σ) para cada acción continua, y una red crítico que estima el valor del estado actual para calcular la ventaja, que indica cuánto mejor es una acción respecto al promedio. Luego, se calculan la pérdida de la red actor y crítica, y se agrega un coeficiente de entropía para fomentar la exploración. Este modelo se actualiza en cada paso con la experiencia obtenida, utilizando una pérdida combinada que incluye la maximización de la probabilidad de buenas acciones, la minimización del error en la estimación de valores, y la penalización de políticas deterministas, lo que permite equilibrar el rendimiento y estabilidad. En la Tabla 3.4 se muestran los hiperparámetros utilizados para este algoritmo y en la Tabla 3.5 se describe el algoritmo A2C desarrollado. En este algoritmo se puede ver que en el paso 9 se selecciona una acción desde una distribución normal, y en el paso 10 se ejecuta la acción y se observa la recompensa y el nuevo estado. Después, en los pasos 11 y 12 se calcula el objetivo TD y la ventaja, en los pasos 13 y 17 se actualizan el crítico y el actor, respectivamente. Finalmente, en el paso 19 se actualizan los parámetros del modelo mediante descenso de gradiente.

Tabla 3.4.

Hiperparámetros A2C

Parámetro	***Valor***
1: Optimizador	*ADAM*
2: Máximo número de pasos:	*3495*
3: Tasa de aprendizaje (α)	*0.001*
4: Factor de descuento (ɤ)	*0.99*
5: Coeficiente de entropía (exploración)	*0.01*
6: Número de capas ocultas actor	*2*
7: Número de unidades por capa actor	*64*
8: Número de salidas actor	*2, media y std*
9: Número de capas ocultas critico	*2*
10: Número de unidades por capa crítico	*64*
11: Número de salidas crítico	*1*
12: Función de activación de capas ocultas actor	*ReLU* *ReLU*
13: Función de activación de capas ocultas critico	*Softplus para σ > 0* *Lineal*
14: Función de activación de salida actor	
15: Función de activación de salida crítico	

Tabla 3.5.

Algoritmo A2C

Algoritmo 3 *Algoritmo A2C*

Entrada: *estado, acción.*
Parámetros: *parametro de aprendizaje, factor de descuento, coeficiente de entropía.*
Salida: *política óptima, función de valor.*

Initialize Tactile Internet model
Initialize Actor π_θ *(mu, sigma)*
Initialize Critic V_ϕ^π
Initialize ADAM optimizer with learning rate

for *each episode I* ***do****:*
  Initialize state s_0
  while $t = 1 \ldots . T$ *in episode I* ***do****:*
    sample action $a_t \sim \pi(a|\mu,\sigma) = \mathcal{N}(a|\mu, \sigma)$ *according to current policy*
    execute action a_t *and observe reward* r_t *and state* s_{t+1}
    set TD target $y_t = r + \gamma V_\phi^\pi(s_{t+1}) * (1 - done)$
    compute advantage $\delta_t = y_t - V_\phi^\pi(s_t)$
    Update *critic minimizing loss* $\mathcal{L}_{critic} = {\delta_t}^2$

    Compute log probability $log\pi = -0.5 * \left(\frac{(a-\mu)}{\sigma}\right)^2 - log(\sigma)$
    Compute entropy $H = 0.5 * (log(2\pi\sigma^2) + 1)$
    Update *actor policy minimizing loss*
    $$\mathcal{L}_{actor} = -log\pi \cdot \delta_t - entropy_{coef} * H$$
    Update *model parameters using gradient descent on total loss*
    Update $S_t \leftarrow s_{t+1}$
  end while
end for

3.3. Métricas de evaluación

Las métricas de calidad utilizadas son de dos tipos, relacionadas al rendimiento del algoritmo y asociadas a la calidad del servicio, es decir, evalúan el desempeño del sistema controlado por el agente.

3.3.1. Recompensa acumulada

La recompensa acumulada corresponde a la suma total de las recompensas obtenidas en cada episodio. Esta evalúa si el agente está aprendiendo a maximizar su objetivo [10], y se calcula mediante la siguiente expresión:

$$Rt_e = \sum_n (R_1 + R_2 + \cdots + R_n) \tag{15}$$

Donde, Rt_e es la recompensa total acumulada en el episodio, R_n es la recompensa obtenida en el paso n, y n corresponde al número de pasos. Finalmente, se calcula la recompensa promedio global donde se incluyen todos los episodios desde la fase de exploración inicial hasta alcanzar su convergencia. Este análisis global se obtiene con la siguiente expresión:

$$R = \frac{\sum_e (R_1 + R_2 + \cdots + R_e)}{N} \tag{16}$$

Donde, R_e representa la recompensa del episodio e y N el número de episodios.

3.3.2. Rendimiento útil acumulado

El rendimiento útil (goodput) es una medida de la eficiencia de la red. Hace referencia a la cantidad de datos útiles entregados por unidad de tiempo y a diferencia del rendimiento (throughput), no incluye los datos redundantes o los retransmitidos [15]. Se calcula mediante la siguiente expresión:

$$goodput_t = \frac{Datos\ útiles\ entregados}{Tiempo\ total} \quad (17)$$

Donde el tiempo total es la latencia y los datos útiles entregados se calculan de la siguiente manera:

$$Datos\ útiles\ entregados = \frac{tamaño\ paquete}{ns} * (1 - pe) \quad (18)$$

Donde ns es el número de segmentos y pe es la tasa de error. Ahora, el rendimiento útil acumulado se define como el rendimiento útil obtenido en cada episodio, quedando así:

$$goodputT = \sum_n (g_1 + g_2 + \cdots + g_n) \quad (19)$$

Donde, el g_n es el rendimiento útil en el paso de tiempo n. Para el análisis global se plantea el promedio del rendimiento útil acumulado en cada episodio de la siguiente manera:

$$G = \frac{\sum_n (g_1 + g_2 + \cdots + g_n)}{N} \quad (20)$$

Donde, G es el rendimiento útil promedio, g_n es el rendimiento obtenido en el episodio n y N es el número de episodios.

3.3.3. Retardo promedio móvil

El retardo promedio móvil (Moving Average Delay - DMA) es una métrica que permite analizar la tendencia del retardo en una ventana móvil de los últimos W pasos o paquetes. Se calcula así:

$$DMA_t = \frac{1}{W} \sum_{k=t-W+1}^{t} L_k \quad (21)$$

Donde, DMA_t es el retardo promedio móvil en el tiempo t, L_k es el retardo o la latencia del paquete en el paso k, y W es el tamaño de la ventana. Esta medida permite identificar patrones de comportamiento en el tiempo, por lo cual es útil en redes donde los retardos variables pueden afectar la QoS y QoE. **Interpretación**: Tiempo promedio de transmisión por

3.3.4. Pérdida de paquetes

Esta métrica se representa como el porcentaje de paquetes perdidos en el momento de su transmisión respecto al total de paquetes enviados. Se calcula mediante la siguiente expresión:

$$P_{loss} = \frac{Número\ de\ paquetes\ perdidos}{Número\ de\ paquetes\ enviados} * 100 \quad (22)$$

Esta métrica permite conocer la tasa de pérdida de paquetes y con ello el estado de la red. Un alto porcentaje de pérdida indica congestión o problemas en la red, lo cual afecta la QoS y QoE de las aplicaciones.

Cabe mencionar que en la simulación de pérdida de paquetes se considera que un paquete se pierde si un valor aleatorio es mayor a la tasa de pérdida de paquetes o, si el IC_t es mayor a BW/ns, ya que si esto se cumple indica que hay más tráfico del que puede manejar el ancho de banda asignado a cada segmento, lo que resulta en la pérdida de paquetes.

3.3.5. Paquetes demorados

Esta métrica hace referencia a los paquetes que superan un umbral de tiempo. Los paquetes con retraso suelen ser inútiles para aplicaciones sensibles como las aplicaciones TI, lo cual afecta la QoS y QoE. Esta se calcula similar a la pérdida de paquetes, pero con los paquetes retrasados, así:

$$P_{delay} = \frac{Número\ de\ paquetes\ demorados}{Número\ de\ paquetes\ enviados} * 100 \quad (23)$$

Capítulo 4
Implementación y Análisis de Resultados

En este capítulo, evaluamos el rendimiento de los algoritmos propuestos utilizando nuestro enfoque de gestión, mediante simulaciones.

4.1. Configuración de las simulaciones

El modelo de red planteado simula una red táctil donde el agente debe controlar dinámicamente el número de segmentos (entre 1 y 10) en que se fragmenta un paquete o flujo de datos para ser transmitido bajo condiciones variables de BW, IC y pe. En cada paso se simula el envío de paquetes considerando pérdidas aleatorias con una probabilidad del 80%, y se calculan las métricas de rendimiento útil y retardo promedio móvil.

Para su evaluación, entrenamos nuestros agentes realizando diversas simulaciones en las que se debe completar la entrega de 4000 paquetes considerando diferentes parámetros de red. Estos parámetros utilizados se seleccionaron tomando en cuenta la investigación base de nuestro estudio [1] y la configuración de red en la categoría de TI para sistemas de teleoperación establecidos en el estándar IEEE1918.1 [16]. La configuración de los escenarios y los parámetros seleccionados se muestran en la Tabla 4.1.

Tabla 4.1

Parámetros de la simulación

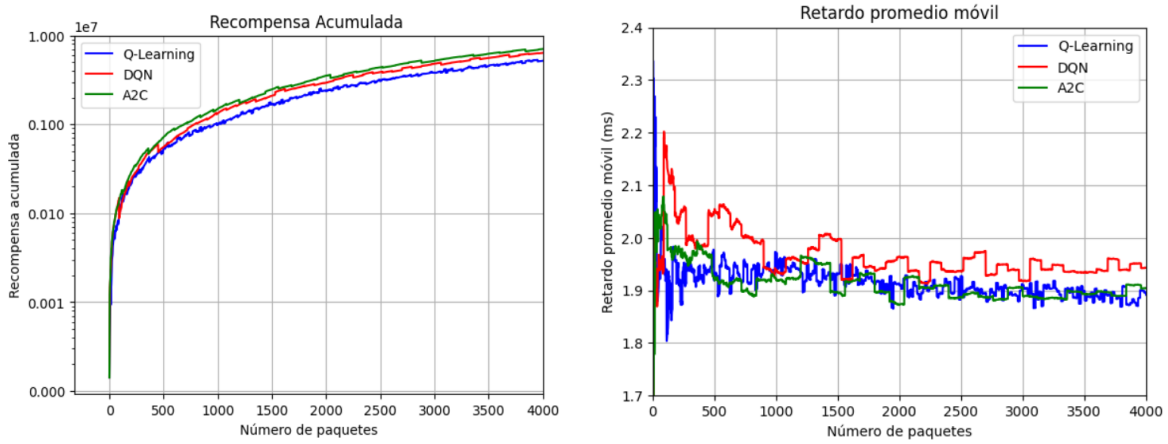
Parámetros		**Valor**	
Retardo de propagación extremo a extremo (tPD)		1 – 50 ms	
	Escenario 1		1 ms
	Escenario 2		50 ms
	Escenario 3		1 - 50 ms
Tasa de error (pe)		0 % - 50 %	
	Escenario 1		0% - 50%
	Escenario 2		5%
	Escenario 3		10%
Ancho de banda (BW)		10 – 500 Mbps	
	Escenario 1		100 – 500 Mbps
	Escenario 2		10 Mbps
	Escenario 3		100 Mbps
N° Paquetes entregados		4000	
Tipo de Trafico		TCP	
Algoritmos	Q-Learning	DQN	A2C
Tasa de aprendizaje (α)	0.1	0.01	0.001
Factor de descuento (γ)	0.99	0.9	0.99
Valor para la exploración	1	1	0.01

4.2. Experimentos

4.2.1. Prueba inicial (Escenario 1)

Realizamos una prueba inicial utilizando el dataset de parámetros de red definidos en [17], con un BW variable entre 100 y 500 Mbps, un retardo de propagación de extremo a extremo (tPD) de 1ms y pe seleccionado de forma aleatoria entre 0 y 50%, lo que lo hace un entorno dinámico.

La Figura 4.1(a) representa la curva de aprendizaje de los agentes mediante la recompensa acumulada, donde se observa que A2C tiene un mejor aprendizaje, aunque muy similar a DQN. Mientras que en la Figura 4.1(b) se puede ver que Q-Learning y A2C tienen un retardo menor en comparación con DQN. Por su parte, la curva del rendimiento útil acumulado (ver Figura 1 del Apéndice) tiene un comportamiento similar a la curva de recompensa, donde A2C muestra mejor rendimiento. En la Tabla 4.2 se pueden ver los valores promedios finales (de todas las corridas) de las medidas de evaluación, en la cual se observa que A2C tiene mejor recompensa y rendimiento acumulado, superando a DQN y Q-Learning, debido a su capacidad para adaptarse de forma dinámica a las condiciones variables de la red. En cuanto al retardo promedio móvil, las diferencias entre los algoritmos son mínimas donde Q-Learning tiene un menor retardo, aunque similar a A2C. Esto se debe a que Q-Learning puede ser más eficiente en tiempo de respuesta por su simplicidad algorítmica, sin embargo, A2C mantiene un retardo competitivo a pesar de su mayor complejidad.



(a) (b)

Figura 4.1 Comparación de curva de aprendizaje (a) y retardo promedio móvil (b) entre los algoritmos.

Tabla 4.2.

Estadísticas de las medidas de evaluación para la prueba inicial

Algoritmo	Promedio de Recompensa Acumulada	Promedio de Rendimiento útil Acumulado	Promedio de Retardo promedio móvil (ms)
Q-Learning	2.472.184,84	7.624,84	1,89
DQN	3.055.374,99	9.565,11	1,94
A2C	3.417.649,11	10.520,87	1,90

4.2.2. Resultados

Para las siguientes simulaciones planteamos 3 escenarios con las configuraciones mostradas en la Tabla 4.1, utilizando datos generados en Mininet.

La Figura 4.2 representa el **escenario 1**, el cual se caracteriza por ser un entorno dinámico. En la Figura 4.2(a) y 4.2(b) se muestra la curva de aprendizaje de los algoritmos utilizados y el rendimiento útil de la red, donde se observa que A2C y DQN tienen un comportamiento casi similar, mientras que el de Q-Learning es menor considerablemente. En la Figura 4.2(c) se compara los porcentajes de paquetes perdidos, que muestra que A2C tiene una menor pérdida de paquetes y es más estable, a diferencia de DQN y Q-Learning con porcentajes mayores muy similares. En la Figura 4.2(d) se muestra la evolución del retardo promedio móvil en el que se observan fluctuaciones por parte de los diferentes algoritmos, sin embargo, A2C tiene menor retardo con una mayor estabilidad, DQN muestra un mayor retardo, aunque más estable que Q-Learning, con un retardo bastante variable debido a su convergencia lenta. Esto se debe a las condiciones cambiantes de la red.

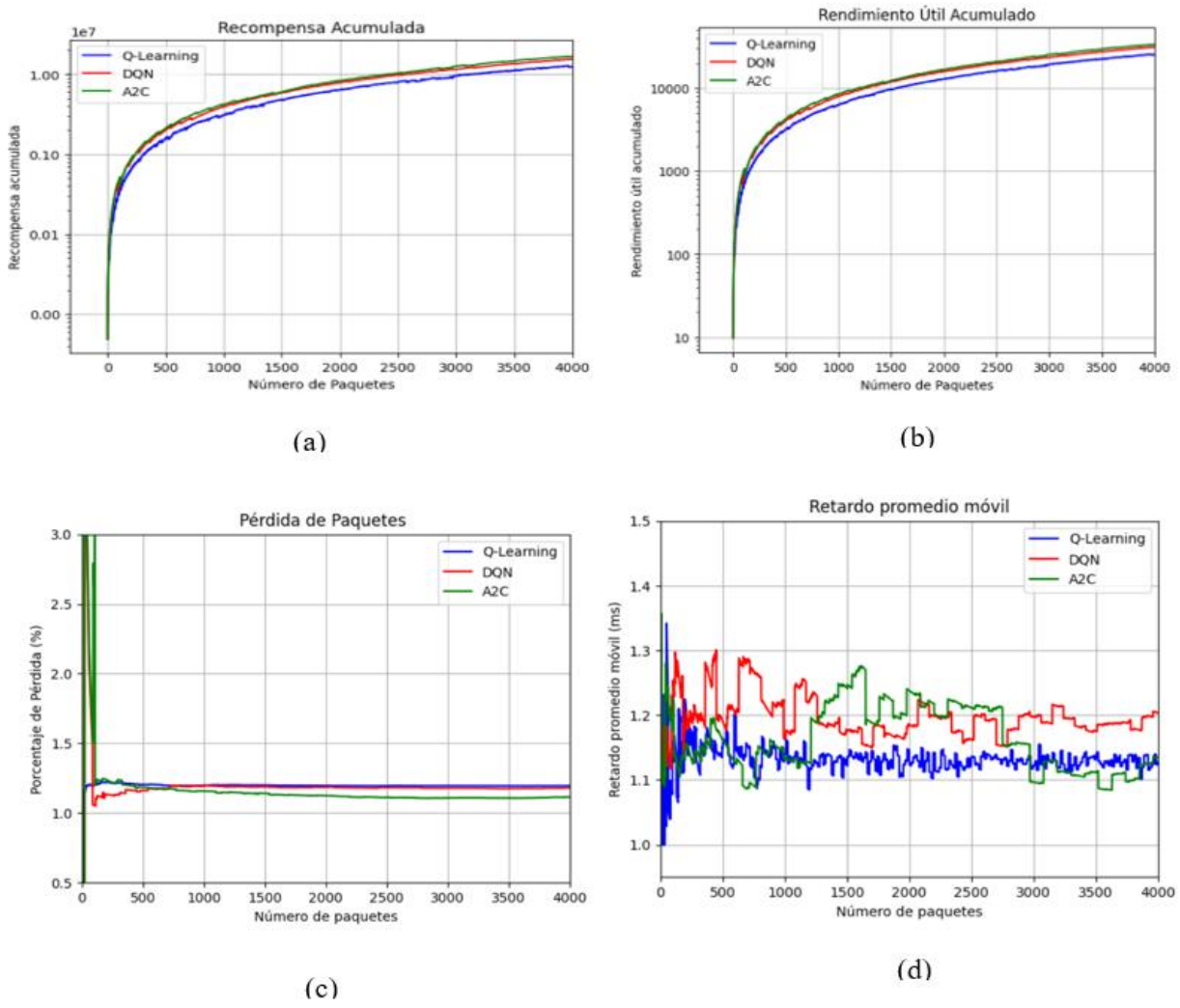


Figura 4.2 Comparación de la recompensa acumulada (a), rendimiento útil (b), pérdida de paquetes (c) y retardo promedio móvil (d).

En la Tabla 4.3 podemos ver los valores promedios finales de estas métricas, donde se observa que A2C es más eficiente y equilibrado en entornos dinámicos, ya que logra una mayor recompensa y rendimiento útil acumulado, manteniendo un menor retardo promedio móvil y un menor porcentaje de pérdida de paquetes respecto a DQN y Q-Learning. Esto se debe gracias a su enfoque basado en política y valor que le permite tener capacidad de adaptación a condiciones dinámicas y optimizar decisiones en tiempo real. Por su parte, DQN mejora a Q-Learning en cuanto a recompensa acumulada, rendimiento útil acumulado y pérdida de paquetes, pero no

alcanza a A2C, debido a que, aunque maneja mejor la variabilidad que Q-Learning no es tan robusto frente a una aleatoriedad extrema en comparación con A2C. Finalmente, Q-Learning aunque tiene un retardo menor que DQN y similar a A2C, muestra un peor desempeño en las otras métricas, lo que evidencia sus limitaciones en entornos dinámicos.

***Tabla 4.3*.**

Estadísticas de las medidas de evaluación para el escenario 1

Algoritmo	**Promedio de Recompensa Acumulada**	**Promedio de Rendimiento útil Acumulado**	**Promedio de Retardo promedio móvil (ms)**	**Promedio de Pérdida de paquetes (%)**
Q-Learning	6.358.068,33	12.877,19	1,13	1,20
DQN	7.779.402,42	15.828,0	1,20	1,19
A2C	8.328.648,86	16.871,55	1,12	1,16

Para el **escenario 2**, en la Figura 4.3 se muestra el retardo promedio móvil y la pérdida de paquetes con un BW = 10 Mbps, tPD = 50 ms y pe = 5%. Estos parámetros representan un entorno menos dinámico. La Figura 4.3(a) muestra que A2C tiene un retardo notablemente menor en relación a DQN y Q-Learning. Asimismo, en la figura 4.3(b) se observa que DQN tiene una mayor pérdida de paquetes, similar a Q-Learning, mientras que A2C muestra un mejor manejo de la pérdida de paquetes. Esto se debe a que A2C, gracias a su arquitectura distribuye el tráfico de forma equilibrada, con lo cual ajusta dinámicamente la política para evitar congestión y errores en enlaces con bajo BW, lo que minimiza tanto el retardo como la pérdida de paquetes. Por su parte, DQN aprende a evitar congestiones mediante su historial de experiencias, por lo tanto, se ve afectada si no es tan grande su memoria; mientras que Q-Learning no gestiona adecuadamente la saturación y añade retrasos por tomar decisiones subóptimas. Las curvas de recompensa

acumulada y rendimiento útil acumulado muestran un comportamiento parecido, donde A2C y DQN son casi similar (ver Figura 2(a) y 2(b) en el Apéndice, respectivamente).

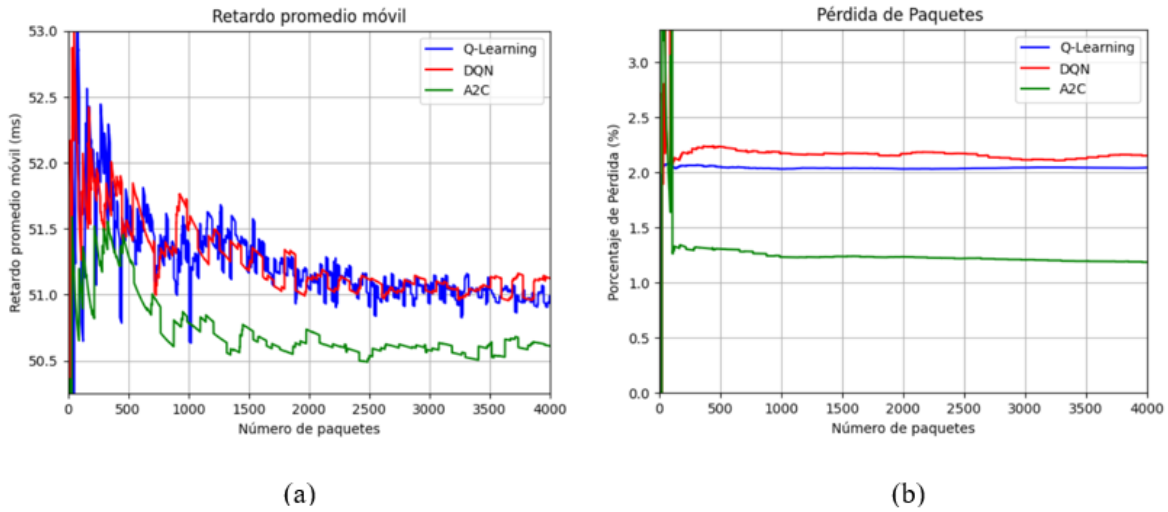


Figura 4.3 Comparación de retardo promedio móvil (a) y pérdida de paquetes (b) con BW = 10 Mbps y tPD = 50ms.

Los valores promedios finales de las métricas se muestran en la Tabla 4.4, en la cual se puede ver que A2C tiene mejor desempeño en todas las variables gracias a su enfoque que le permite adaptarse a las condiciones, y en este caso, optimizar el uso de BW limitado a la vez que minimiza el retardo. DQN tiene una mayor recompensa y rendimiento útil acumulado, aunque con un retardo y pérdida de paquetes ligeramente mayor respecto a Q-Learning; esto debido a su dependencia de aproximaciones basadas en valor, mientras que Q-Learning muestra un desempeño inferior, lo que muestra sus limitaciones en entornos complejos y restrictivos.

Tabla 4.4.

Estadísticas de las medidas de evaluación para el escenario 2

Algoritmo	**Promedio de Recompensa Acumulada**	**Promedio de Rendimiento útil Acumulado**	**Promedio de Retardo promedio móvil (ms)**	**Promedio de Perdida de paquetes (%)**
Q-Learning	2.801,76	282,14	50,93	2,04
DQN	3.494,09	352,23	51,12	2,18
A2C	3.711,44	372,65	50,61	1,26

La Figura 4.4 representa el **escenario 3**. En esta se muestra el retardo promedio móvil con un BW = 100 Mbps, pe = 10% y tPD de (a) 1 ms, (b) 2 ms, (c) 4 ms y (d) 50 ms. Esta configuración permite representar un entorno con mayor dinamismo como ocurre en las aplicaciones de teleoperación TI.

En este escenario, también se evaluó el retraso y la pérdida de paquetes. Para ello se estableció un tiempo límite definido como tPD + 10%, de esta manera si un paquete supera este tiempo de entrega, se considera demorado o retrasado. Este porcentaje de demora es importante ya que nos permite identificar que algoritmo cumple mejor con los estrictos requisitos de TI o, se adecua mejor a las necesidades de transmisión de datos de cada aplicación de TI. Los detalles en cuánto a demora y pérdida de paquetes se presentan en la Tabla 4.5, incluyendo en la columna [1] el mejor valor obtenido en el trabajo [1] como referencia comparativa. En esta Tabla se observa que A2C tiene el mejor desempeño en todas las configuraciones de retardo (tPD = 1ms, 2ms, 4ms, 50ms), ya que minimiza la pérdida y demora de paquetes, gracias a su enfoque que optimiza políticas en tiempo real para adaptarse a las condiciones priorizando la confiabilidad. También, se puede ver que las demoras son menores en A2C, especialmente en altos tPD; esto se debe a que A2C evita retransmisiones innecesarias. Por su parte, DQN y Q-Learning tiene mayores demoras

y pérdida de paquetes, algunas ligeramente similares, esto debido a que DQN tiende a sobreestimar valores Q lo que lleva a decisiones subóptimas que aumentan las pérdidas, mientras que Q-Learning al ser más simple tiende a ser más robusto en algunos casos por su simplicidad. Además, se observa que nuestra propuesta DRL tiene mejor desempeño, con menor demora y menor pérdida de paquetes respecto a [1], a excepción del caso cuando tPD toma el valor de 50 ms. En ese caso, el enfoque propuesto en [1] tiene menor pérdida de paquetes. Las representaciones gráficas del porcentaje de pérdida y demora de paquetes se pueden observar en el Apéndice (Figura 4 y 5, respectivamente).

Tabla 4.5.

Datos de demora y pérdida de paquetes

Parámetros	**Valores**															
Algoritmo	Q-L	DQN	A2C	[1]	Q-L	DQN	A2C	[1]	Q-L	DQN	A2C	[1]	Q-L	DQN	A2C	[1]
tpD	1 ms				2 ms				4 ms				50 ms			
Perdidos (%)	2,04	2,10	1,15	1,6	1,19	1,23	1,18	1,4	1,22	1,19	1,19	1,4	1,99	2,09	1,66	1,3
Demorados (%)	1,76	3,29	1,06	8	1,20	0,82	0,81	6,4	0,72	0,80	0,72	6,2	0,48	0,52	0,26	5

Por otro lado, en la Tabla 4.6 se muestran los valores promedios finales de las métricas principales, en la que se observa que DQN supera ligeramente a A2C en cuanto a la recompensa y rendimiento útil acumulado, y ambas superan considerablemente a Q-Learning. Sin embargo, en cuanto al retardo promedio móvil, A2C logra mejores resultados en todos los casos de tPD, gracias a su enfoque de política directa. Cabe destacar que se observa que el retardo de propagación es un factor crítico en la medida del retardo promedio móvil que afecta el rendimiento de la red a medida que aumenta, pero A2C demuestra mejor equilibrio ante esto. Además, se puede ver que DQN a pesar de tener mayor recompensa tiene mayor retardo en comparación con Q-Learning, esto es

debido a su complejidad computacional, ya que añade pequeñas demoras al procesar los datos, mientras que Q-Learning realiza búsquedas sencillas en tabla.

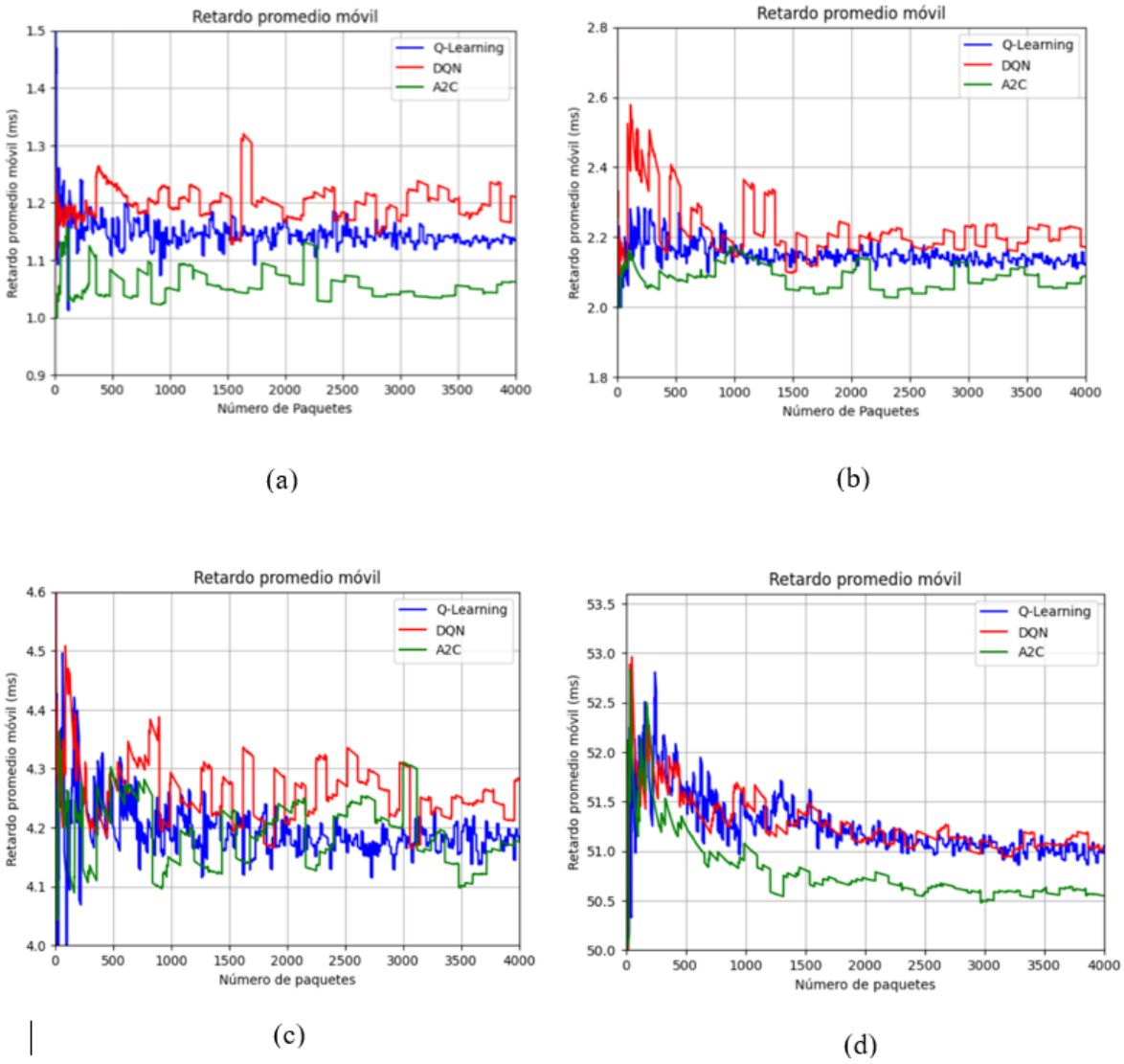


Figura 4.4 Comparación del retardo promedio móvil para tPD (a) 1 ms, (b) 2 ms, (c) 4 ms y (d) 50 ms.

En la Figura 3 del Apéndice se puede ver la representación del rendimiento útil acumulado para este escenario donde DQN y A2C tienen un comportamiento casi similar. En general, los

resultados muestran que A2C tiene el mejor desempeño adaptándose a las diferentes configuraciones de red y tPD.

Tabla 4.6.

Estadísticas de las medidas de evaluación para el escenario 3

Algoritmo	**Promedio de Recompensa Acumulada**				**Promedio de Rendimiento útil Acumulado**				**Promedio de Retardo promedio móvil (ms)**			
tpD	1ms	2ms	4ms	50ms	1ms	2ms	4ms	50ms	1ms	2ms	4ms	50ms
Q-Learning	6.960.622,93	1.818.502,53	458.715,22	2.948,01	14.273,84	7.352,37	3.697,65	296,92	1,13	2,12	4,19	50,98
DQN	8.861.177,94	2.283.765,20	569.903,70	3.674,08	18.021,32	9.200,73	4.600,26	370,57	1,21	2,17	4,28	51,02
A2C	8.851.985,48	2.223.089,0	554.557,90	3.512,37	17.825,05	8.942,19	4.461,69	352,85	1,06	2,08	4,17	50,54

4.3. Comparación cuantitativa

En esta sección, se presenta una comparación cuantitativa para evaluar el rendimiento de nuestra propuesta en relación al enfoque RS-RLNC propuesto en [1], mediante una evaluación numérica de diferencias. En este caso, la comparación se realiza con los datos de *porcentaje de pérdida* y *demora de paquetes* expuestos en la Tabla 4.5, de los cuales se tomaron los mejores valores obtenidos para cada valor de tPD, es decir, los valores de A2C junto a los datos de [1].

Para realizar la comparación se utilizaron las siguientes medidas estadísticas:

1. Diferencia absoluta, la cual consiste en restar el valor de referencia del valor comparado, para medir la variación entre estos dos valores. Su expresión es:

$$Diferencia\ absoluta = valor\ comparado - valor\ de\ referencia \quad (24)$$

2. Diferencia relativa, se expresa como un porcentaje para comparar la magnitud del cambio con respecto a un valor inicial. Se calcula así:

$$Diferencia\ relativa = \frac{Diferencia\ absoluta}{valor\ de\ referencia} x100 \quad (25)$$

También, se utilizó la prueba t de Student, la cual compara la media de dos conjuntos de datos y evalúa si la diferencia entre ellos es suficientemente grande para afirmar que hay una diferencia significativa. En esta se plantean dos hipótesis:

- Hipótesis nula (H_0): No hay diferencia entre las medias ($\mu_1 = \mu_2$).
- Hipótesis alternativa (H_1): Existe diferencia ($\mu_1 \neq \mu_2$, $\mu 1 < \mu_2$ o $\mu_1 > \mu_2$).

Luego, se calcula el estadístico t según el tipo de prueba. En este caso, como se trata de muestras independientes, al ser las medias de dos grupos distintos mide la diferencia entre estas, y se calcula con la siguiente expresión:

$$t = \frac{X_1 - X_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}} \quad (26)$$

Donde, X_1 y X_2 son las medias de las muestras, s_1^2 y s_2^2 son las varianzas muestrales, y n_1 y n_2 son los tamaños de las muestras. Un valor grande de t indica mayor diferencia entre los grupos. Ahora, se busca el valor de p asociado al valor estadístico, con un nivel de significancia

comúnmente de 0.05 ($\alpha = 0.05$). Si $p < 0.05$ se rechaza H_0 e indica que hay una diferencia significativa, y si $p \geq 0.05$ indica que las diferencias no son significativas.

4.3.1. Pérdida de paquetes

En la Tabla 4.7 se muestran los mejores valores de cada enfoque (A2C y [1]) con su respectiva media y desviación estándar.

Tabla 4.7.

Datos de pérdida de paquetes

Enfoque	**% de pérdida de paquetes**				**Media**	**Desviación estándar**
	1 ms	2 ms	4 ms	50 ms		
[1]	1,6	1,4	1,4	1,3	1,42	0,12
Este trabajo	1,15	1,18	1,19	1,66	1,29	0,24

Tomando como valor referencial la media de los datos de [1] se calcularon la diferencia absoluta y relativa según las ecuaciones (24) y (25), quedando:

$$Diferencia\ absoluta = 1.42\ \% - 1.29\ \% = \quad 0.13\ \%$$

$$Diferencia\ relativa = \frac{0.13}{1.42} x100 = 9.12\ \%$$

Esto demuestra que nuestro enfoque reduce un 0.13% los paquetes perdidos con una mejora de 9.12% respecto a la propuesta en [1].

4.3.2. Paquetes demorados

Para la comparación en cuanto a la demora de paquetes en la Tabla 4.8, se muestran los mejores valores de cada enfoque (A2C y [1]) con su respectiva media y desviación estándar.

Tabla 4.8.

Datos de paquetes demorados

Enfoque	**% de paquetes demorados**				**Media**	**Desviación estándar**
	1 ms	2 ms	4 ms	50 ms		
[1]	8	6,4	6,2	5	6,4	1,23
Este trabajo	1,06	0,81	0,72	0,26	0,71	0,33

Al calcular la diferencia absoluta y relativa, queda:

$$Diferencia\ absoluta = 6.4\ \% - 0.71\% = \quad 5.69\ \%$$

$$Diferencia\ relativa = \frac{5.69}{6.4} x100 = 88.90\ \%$$

Esto demuestra que nuestro enfoque reduce un 5.69 % los paquetes retrasados y tiene una mejora del 88.9 % respecto a la propuesta en [1].

4.3.3. Prueba t de Student

Para validar los resultados anteriores se utilizó la prueba t de Student con los datos de las tablas 12 y 13. Los resultados se muestran en la Tabla 4.9, en la cual se observa que, en el caso de la métrica de pérdida de paquetes, $p \geq 0.05$, lo que indica que no hay diferencia significativa, o no hay evidencia suficiente para demostrarlo. Sin embargo, en el caso de la métrica de paquetes

demorados, $p < 0.05$, con lo cual hay diferencias significativas y, por lo tanto, muestra que nuestro enfoque reduce significativamente el porcentaje de paquetes demorados respecto a [1].

Tabla 4.9.

Datos estadísticos de la prueba t de Student

Métrica	**Media ([1])**	**Media (Este trabajo)**	**p**	**Significativo**
%Pérdida paquetes	1,42 ± 0,12	1.29 ± 0,24	0,38	No
%Paquetes demorados	6,4 ± 1.23	0.71 ± 0,33	0,0001	Sí

4.4. Comparación cualitativa

En esta sección, comparamos nuestra propuesta con otros trabajos relacionados lo que nos permite destacar las similitudes y diferencias entre las distintas propuestas (ver Tabla 4.10). Para ello, se definen tres criterios:

- Criterio 1: utiliza algoritmos de DRL.
- Criterio 2: está enfocado en minimizar la latencia.
- Criterio 3: busca mejorar la pérdida de paquetes.

Tabla 4.10.

Comparación cualitativa

Trabajos	Criterios		
	C1	C2	C3
[1]		✓	✓
[4]	✓		
[6]	✓		
Este trabajo	✓	✓	✓

Tomando como referencia estos criterios, en la Tabla 4.10 se presenta un análisis cualitativo de nuestro enfoque y trabajos afines. En [1] se buscó minimizar la latencia en la entrega de paquetes para así mejorar el rendimiento y QoS en las aplicaciones TI. Asimismo, mediante simulaciones de diferentes condiciones de red buscó identificar que esquema de codificación de los utilizados tiene mejor desempeño respecto a la pérdida de paquetes. No obstante, el sistema planteado utiliza un enfoque RL simple. En [4] se utiliza el algoritmo DQL para optimizar la asignación de recursos en la segmentación de redes. Este, a pesar de mostrar resultados favorecedores, no busca mejorar la pérdida de paquetes ni está enfocado en minimizar la latencia, sin embargo, la considera como un requisito importante en la obtención de la recompensa. En [6] se utiliza el algoritmo DRL Soft Actor-Critic (SAC) para garantizar una distribución eficiente de los recursos de radio entre los flujos de datos hápticos y de video, para ello considera la pérdida de paquetes y los requisitos de latencia, aunque no se enfocó en minimizarla. Por otro lado, ese trabajo está enfocado en aumentar la satisfacción del usuario mediante la optimización de la asignación de los recursos en las aplicaciones de teleoperación video-háptica la cual es una rama o proceso del TI.

Como se observa en la Tabla 4.10, los trabajos anteriores no cumplen con todos los criterios. Es por ello que este estudio representa un avance importante al integrar estos aspectos claves, como el enfoque en minimizar la latencia y la mejora en la pérdida de paquetes. Considerar estos criterios permite una mayor eficiencia en la transmisión de datos al reducir errores y retransmisiones innecesarias, lo que optimiza el rendimiento de la red, y mejora la QoS y QoE de las aplicaciones sensibles a la latencia y a la fiabilidad en la entrega de datos. Además, el uso de

técnicas avanzadas de DRL permite lograr un equilibrio óptimo entre los criterios, superando así las limitaciones de los enfoques existentes.

Capítulo 5
Conclusiones y Trabajos Futuros

En esta investigación desarrollamos dos algoritmos DRL para la gestión del TI. Para ello, se llevó a cabo un diseño integral del entorno que simula la red TI teniendo en cuenta sus principales componentes (como la capacidad de BW, tasa de error, tPD, simulación de congestión y pérdida, entre otros), y su interacción con los algoritmos DRL. Particularmente, se propusieron los algoritmos DQN para el manejo de acciones discretas y A2C para el manejo de acciones continuas.

Los resultados obtenidos mediante simulaciones en diversos escenarios demostraron que A2C y DQN logran una recompensa y rendimiento útil acumulado muy similar en varios de los casos. En el escenario 1, que simula un escenario dinámico con BW y pe variable, y en el escenario 2, que simula un entorno menos dinámico con un BW limitado, A2C obtuvo mejores resultados en todas las métricas. En el escenario 3, que simula dinamismo con variación en el tPD, A2C mostró mejores resultados en el tiempo de entrega, demora y pérdida de paquetes; mientras que DQN mostró mejor resultado en cuanto a la recompensa y rendimiento útil acumulado, aunque con un valor ligeramente mayor a A2C. En conclusión, se demostró la eficacia de los algoritmos DRL demostrando su potencial en entorno complejos.

Las limitaciones de este trabajo fueron las siguientes: primero, el estudio solo se probó con características de sistemas de teleoperación, por lo cual se debe evaluar su utilidad en otros entornos de redes con requisitos diferentes. Además, el entorno de simulación estuvo limitado a un número reducido de nodos, lo que dificulta la aplicación de los resultados a redes más grandes y complejas. Por otro lado, no se tomaron en cuenta factores externos como interferencias, lo que es común en entornos reales.

Finalmente, para trabajos futuros se propone:

- Aplicar otros algoritmos como Soft Actor-Critic (SAC) o Deep Determinist Policy Gradient (DDPG), los cuales consideran diferentes características durante el proceso de aprendizaje. Por ejemplo, el SAC incorpora entropía máxima, lo que incentiva a explorar diferentes políticas sin caer en soluciones subóptimas, ofreciendo mejor estabilidad. En cuanto a DDPG con su enfoque determinístico y uso de replay buffer, aprovecha mejor los datos. Además, estos algoritmos ofrecen un mejor manejo de espacios de búsqueda continuos.

- Considerar otras acciones como: la asignación de ancho de banda disponible de acuerdo a las necesidades de las diferentes aplicaciones TI para optimizar el rendimiento y evitar cuellos de botella; o la implementación de técnicas para el manejo de la congestión o priorización de tráfico que reducirían la pérdida de paquetes y latencia para así garantizar un servicio más estable.

- Estudiar otros casos en la TI como aplicaciones en sistemas de control industrial remoto, que requieren fiabilidad extrema, es decir, cero pérdida de paquetes en

comandos críticos; o realidad virtual/aumentada colaborativa, la cual requiere latencia menor a 20 ms y sincronización entre múltiples usuarios. Sus requisitos extremos podrían ser útiles para validar algoritmos de optimización DRL.

Referencias

[1] Shahzad, R. Ali, A. Haider, H. S. Kim, "RS-RLNC: A Reinforcement Learning-Based Selective Random Linear Network Coding Framework for Tactile Internet," *IEEE Access*, vol. 11, pp. 141277-141288, 2023. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10347212&isnumber=10005208

[2] S. K. Sharma, I. Woungang, A. Anpalagan, S. Chatzinotas, "Toward Tactile Internet in Beyond 5G Era: Recent Advances, Current Issues, and Future Directions," *IEEE Access*, vol. 8, pp. 56948-56991, 2020, doi: 10.1109/ACCESS.2020.2980369.

[3] A. Ramírez, J. Aguilar, M. R-Moreno, "Deep reinforcement learning approaches for the hydro-thermal economic dispatch problem considering the uncertainties of the context," *Sustainable Energy, Grids and Networks*, vol. 35, 2023, doi.org/10.1016/j.segan.2023.101109.

[4] R. Li, Z. Zhao, Q. Sun, C. Lin, C. Yang, X. Chen, M. Zhao, H. Zhang, "Deep reinforcement learning for resource management in network slicing," *IEEE Access*, vol. 6, pp. 74429- 74441, 2018, doi: 10.1109/ACCESS.2018.2881964.

[5] C. Ssengonzi, O. P. Kogeda, O. Olwal, "A survey of deep reinforcement learning application in 5G and beyond network slicing and virtualization," *Array*, vol. 14, 2022 Available: https://doi.org/10.1016/j.array.2022.100142

[6] G. Kokkinis, A. Iosifidis, Q. Zhang, "Deep reinforcement learning-based video-haptic radio resource slicing in tactile internet," *DIGIT and Department of Electrical and Computer Engineering*, Aarhus University, 2025. Available: https://doi.org/10.48550/arXiv.2503.14066

[7] G. Fettweis, H. Boche, T. Wiegand, E. Zielinski, H. Schotten, P. Merz, S. Hirche, A. Festag, W. Haffner, M. Meyer y E. Steinbach, "The tactile internet ITUT technology watch report," Unión Internacional de Telecomunicaciones (UIT), Ginebra, Suiza, Tech. Rep., 2014. Available: https://www.itu.int/oth/T2301000023/en

[8] J. Wagner, H. Winger, C. Cherif y F. Ellinger, "Smart Glove With Fully Integrated Textile Sensors and Wireless Sensor Frontend for the Tactile Internet," IEEE Sensors Letters, vol. 7, no. 2, 2023., doi: 10.1109/LSENS.2023.3239991.

[9] C. Pineda, *Aprendizaje Automático y Profundo en Python. Una mirada hacia la inteligencia artificial*. 1ª ed. Ra-Ma Editorial, 2022.

[10] R. S. Sutton y A. G. Barto, *Reinforcement Learning. An Introduction*, 2ª ed. Cambridge, Massachusetts: The MIT Press, 2018.

[11] Y. Li, "Deep Reinforcement Learning: An overview," Cornell University, 2018, doi.org/10.48550/arXiv.1810.06339.

[12] S. Ohnishi, E. Uchibe, Y. Yamaguchi, Y. Yasui, S. Ishii, "Constrained Deep Q-Learning Gradually Approaching Ordinary Q-Learning," Frontiers in Neurobotics, 2019, doi: 10.3389/fnbot.2019.00103.

[13] Y. Chen, F. Zhang, Z. Liu, "Adaptive Advantage Estimation for Actor-Critic Algorithms," International Joint Conference on Neural Networks (IJCNN), Shenzhen, China, 2021, doi: 10.1109/IJCNN52387.2021.9534005.

[14] C. Szepesvari. *Algorithms for Reinforcement Learning*. Morgan &Claypool Publishers. Synthesis Lectures on Artificial Intelligence and Machine Learning, 2009.

[15] A. Tanenbaum and D. Wetherall, *Computer Networks*, 5ª ed. Pearson Education, Inc., 2012.

[16] O. Holland, E. Steinbach, R. V. Prasad, Q. Liu, Z. Dawy, A. Aijaz, N. Pappas, K. Chandra, V. S. Rao, S. Oteafy, M. Eid, "The IEEE 1918.1 'tactile internet' standards working group and its standards", Proc. IEEE, vol. 107, no. 2, pp. 256-279, 2019, doi: 10.1109/JPROC.2018.2885541.

[17] Y. Rivera, "Dataset de parámetros de red," 2024. Dataset no publicado. Contacto: yaircostac@hotmail.com.

[18] "Algoritmo actor-crítico en el aprendizaje por refuerzo". Available: https://www-geeksforgeeks-org.translate.goog/actor-critic-algorithm-in-reinforcement-learning/?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=tc

Referencias

Apéndice

Figura 1. Curva del rendimiento útil acumulado de la prueba inicial

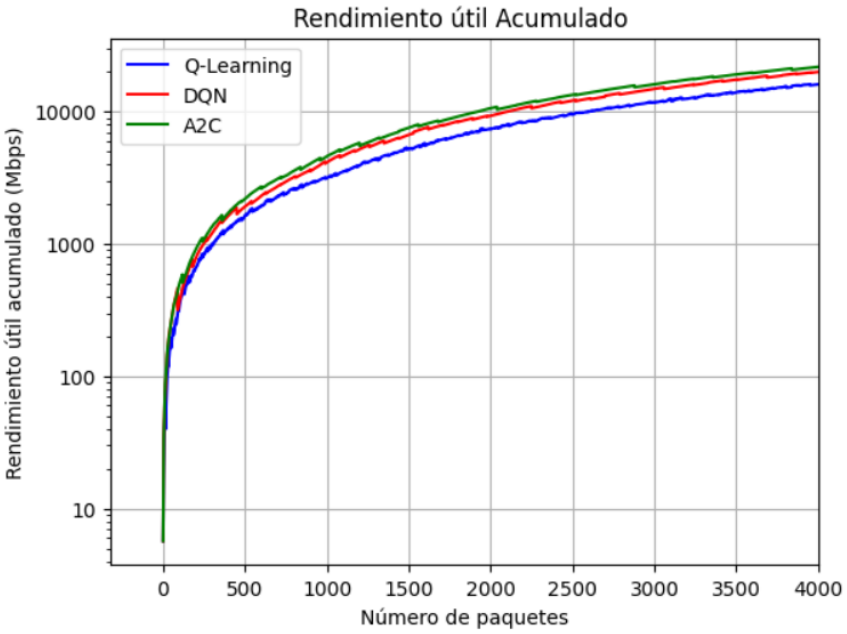
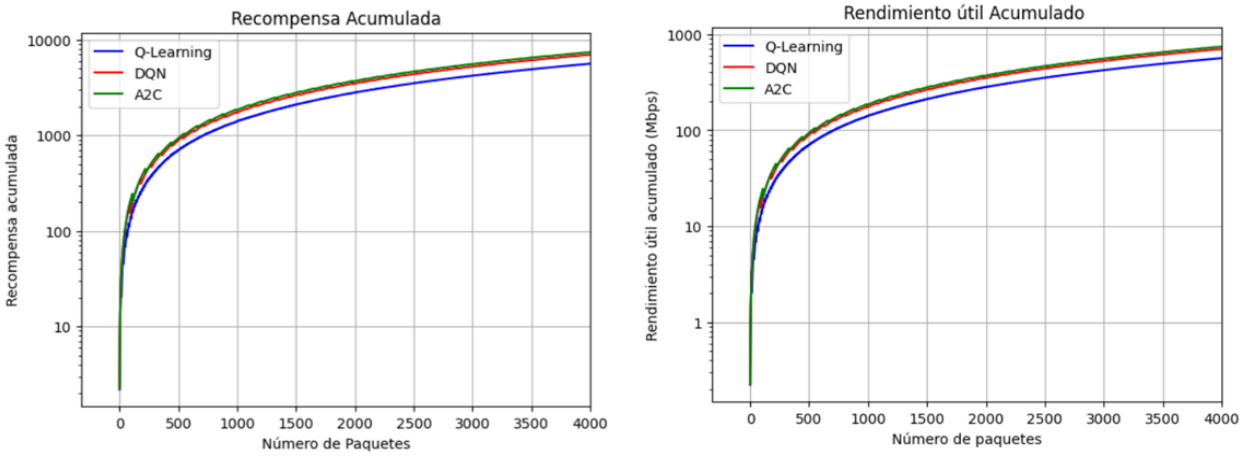
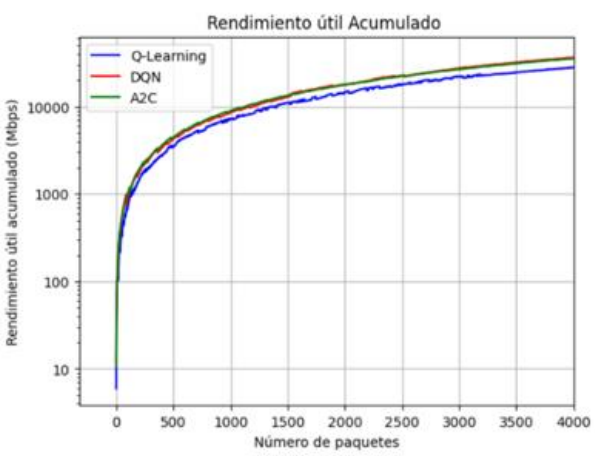


Figura 2. Curvas de la recompensa acumulada (a) y rendimiento útil acumulado (b) del escenario 2

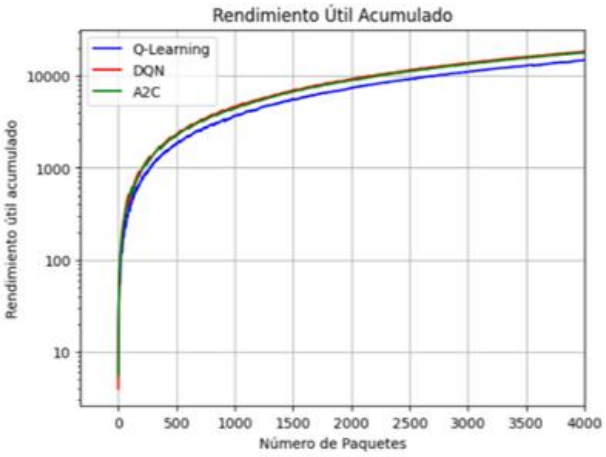


(a) (b)

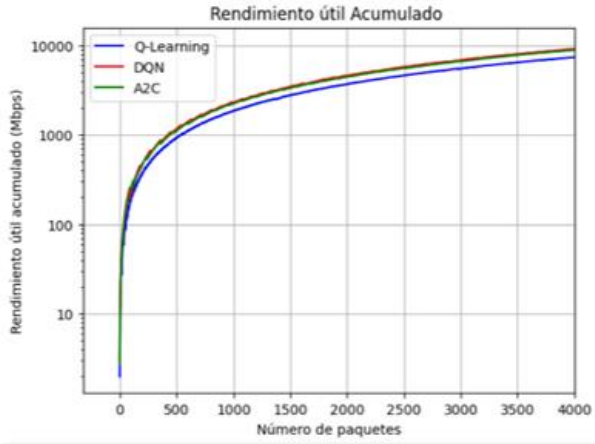
Figura 3. Rendimiento útil acumulado del escenario 3 para (a) 1 ms, (b) 2 ms, (c) 4 ms y (d) 50 ms



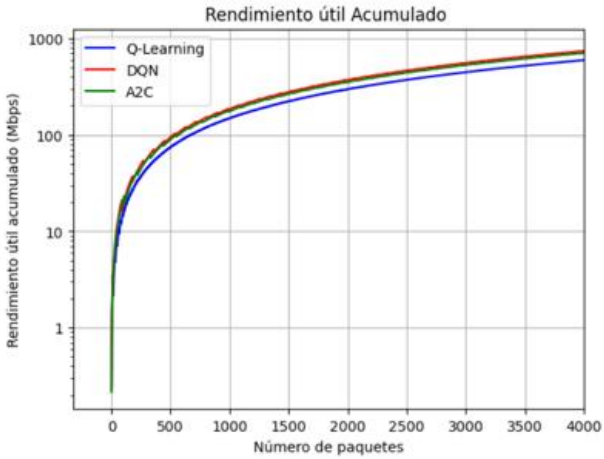
(a)



(b)



(c)



(d)

Figura 4. Porcentaje de pérdida de paquetes del escenario 3 para (a) 1 ms, (b) 2 ms, (c) 4 ms y (d) 50 ms

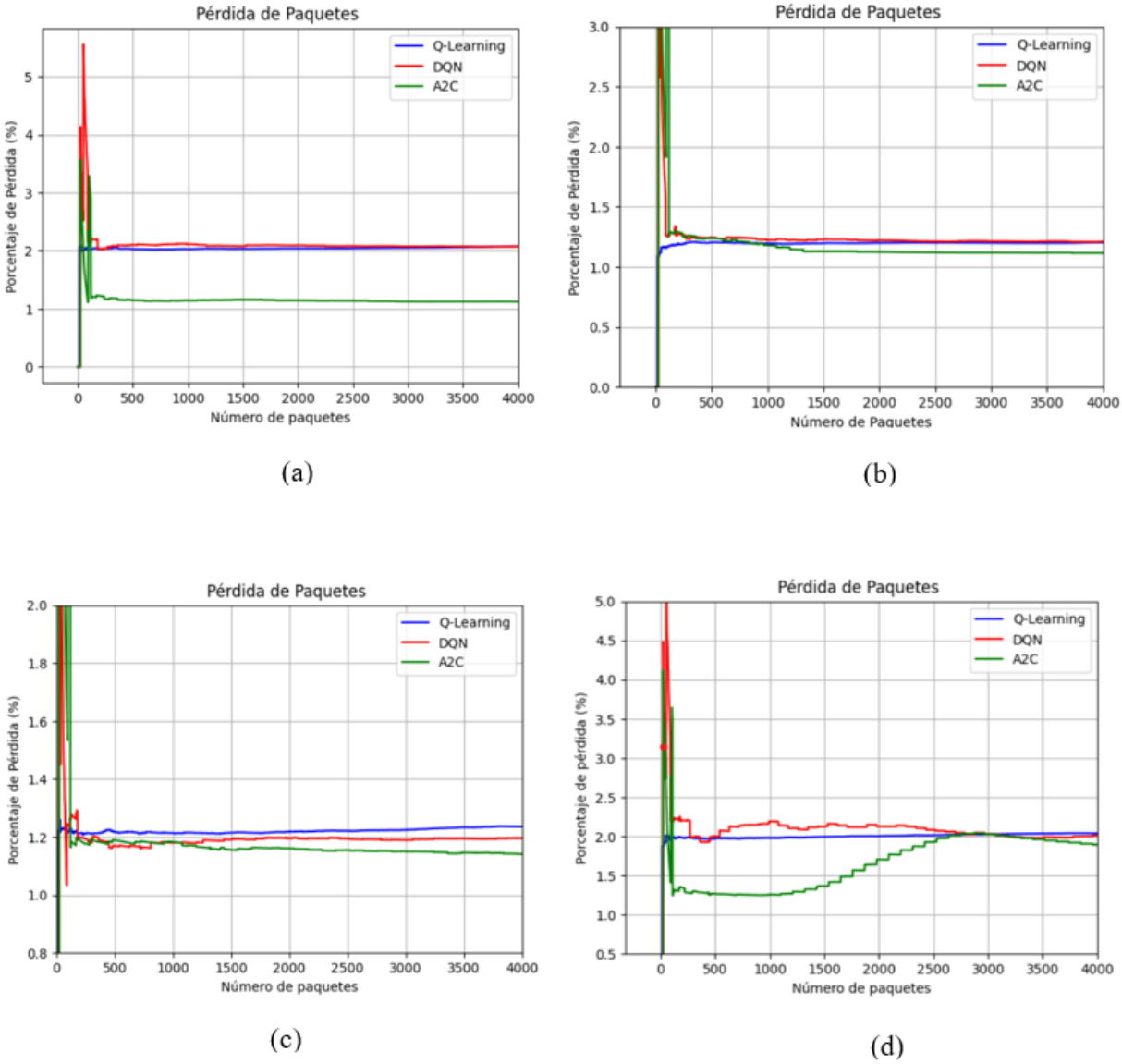


Figura 5. Porcentaje de paquetes demorados del escenario 3 para (a) 1 ms, (b) 2 ms, (c) 4 ms y (d) 50 ms