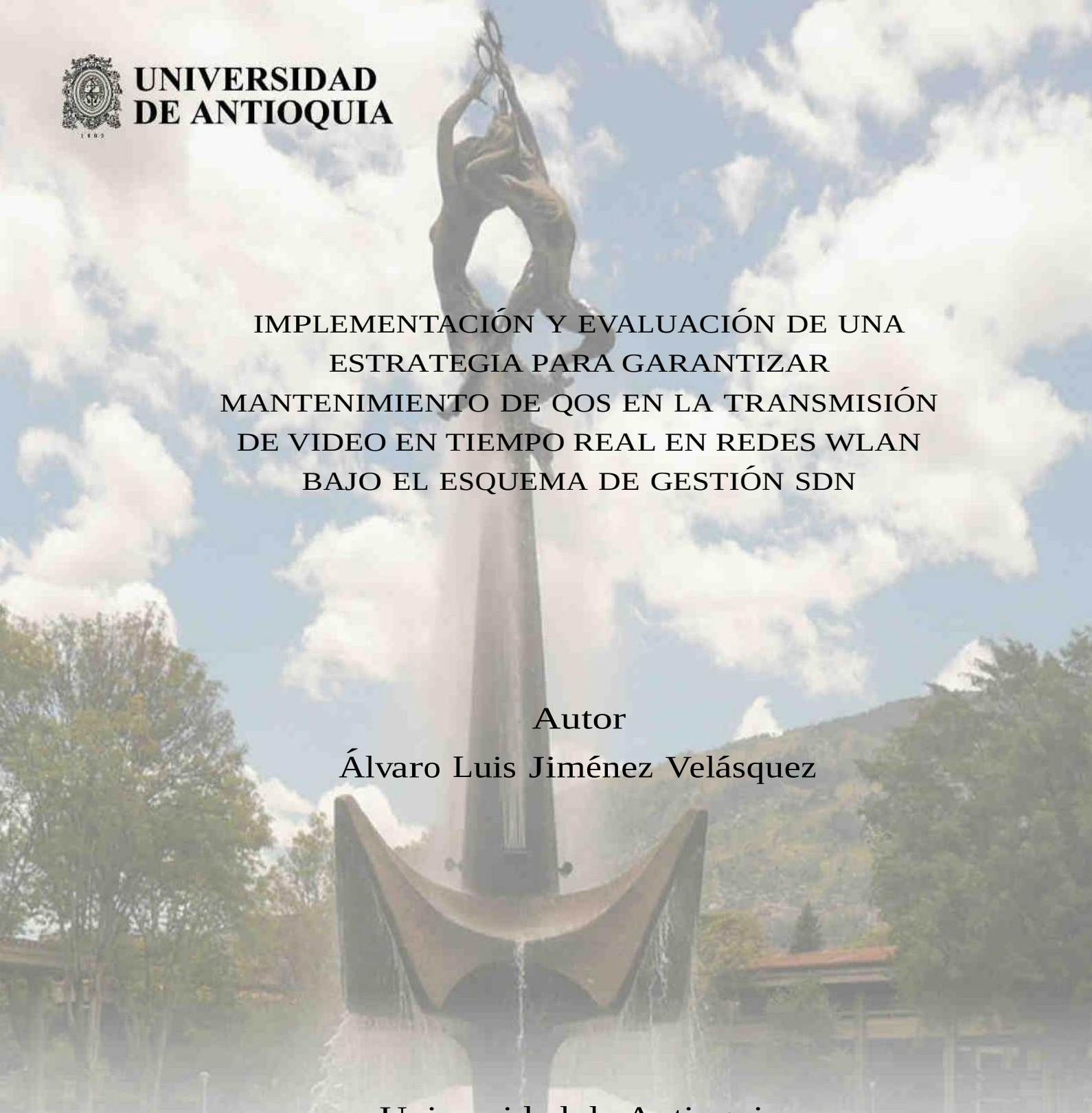

UNIVERSIDAD DE ANTIOQUIA

IMPLEMENTACIÓN Y EVALUACIÓN DE UNA ESTRATEGIA PARA GARANTIZAR MANTENIMIENTO DE QOS EN LA TRANSMISIÓN DE VIDEO EN TIEMPO REAL EN REDES WLAN BAJO EL ESQUEMA DE GESTIÓN SDN

Autor

Álvaro Luis Jiménez Velásquez

Universidad de Antioquia

Facultad de Ingeniería, Departamento de Electrónica y Telecomunicaciones

Medellín, Colombia

2019

Implementación y evaluación de una estrategia para garantizar mantenimiento de QoS en la transmisión de video en tiempo real en redes WLAN bajo el esquema de gestión SDN

Álvaro Luis Jiménez Velásquez

Trabajo de investigación como requisito para optar al título de:
Magister en Ingenier´ıa de Telecomunicaciones

Codirectores:
Juan Felipe Botero Vega, PhD
Juan Pablo Urrea Duque, PhD

Universidad de Antioquia
Facultad de Ingeniería, Departamento de Electrónica y Telecomunicaciones
Grupo de Investigación de Telecomunicaciones Aplicadas, GITA
Medell´ın, Colombia

2019

"La educación genera confianza. La confianza genera esperanza. La esperanza genera paz".

Confucio

Abstract

Local Area Networks based on IEEE 802.11 standard represent the largest deployment of wireless network technology worldwide. Such networks have accelerated the growth of multimedia applications. This type of content requires special settings to guarantee Quality of Service (QoS), specially for the case of real-time video streaming. Currently, the state-of-the-art mostly focuses on solving the QoS problem over traditional IEEE 802.11 networks, but few works try to evaluate the performance of QoS mechanisms over software-defined WLAN network scenarios. The Software Defined Networking (SDN) architecture helps to improve, simplify, and program the network management according to specific needs, making easier and faster the design and deployment of new control policies. This thesis presents the implementation of a non-standardized QoS mechanism for IEEE 802.11 networks, independent of a manufacturer that can be configured and adapted to the needs of an institution. This solution will allow the flows in the active stations to maintain a good throughput. In addition, it will present throughput and delay analysis from the admission control mechanism selected, taking into account the operation of the emulation scenario based on NS-3 integrated with SDN.

Agradecimientos

A Dios por darme la fortaleza espiritual para afrontar los obstáculos y favorecerme con las oportunidades para alcanzar mis metas. A mi familia por el apoyo incondicional y por sacrificar tiempo valioso de compartir en mi ausencia. A la Universidad de Antioquia por brindar el apoyo económico a través del estímulo de Estudiante Instructor. Al Grupo de Investigación de Telecomunicaciones Aplicadas - GITA. Al profesor Juan Felipe Botero por todo el acompañamiento y asesoría durante estos tres años, además por permitirme ser parte del proyecto de Colciencias. Al profesor Juan Pablo Urrea por su dedicación y esfuerzo en ayudarme a sacar adelante este proyecto y comprender las dificultades a las que nos enfrentamos. A los compañeros Daniel Ospina, Juan Fernando González, Paola Soto y Nelson Cardona por enriquecer con sus aportes el desarrollo del proyecto. A los estudiantes Edison Aza y Steven Mart´ınez por continuar explorando mayores alcances del proyecto.

Este trabajo se desarrolló dentro del proyecto *"Desarrollo e implementación de una red inalámbrica definida por software para la ampliación de cobertura de eventos en redes institucionales"* financiado por el Departamento Administrativo de Ciencia, Tecnolog´ıa e Innovación - COLCIENCIAS.

Índice general

Índice de figuras

Índice de tablas

Abreviaturas

ACA	Algoritmo de Control de Admisión (Admission Control Algorithm)
ACs	Categor´ıas de Acceso (Access Categories)
AC_BE	AC de Mejor esfuerzo (AC BEst effort)
AC_BG	AC de Segundo plano (AC Background)
AC_VI	AC de Video (AC VIdeo)
AC_VO	AC de Voz (AC VOice)
ACK	Acuse de Recibo (Acknowledgement)
AIFS	Arbitraje de Espacios Intertramas (Arbitration Inter-Frame Space)
AIFSN	Arbitraje de Número de Espacios Intertramas (AIFS Number)
A-MSDU	MSDU Agregado (Aggregation MSDU)
AP	Punto de Acceso (Access Point)
API	Interfaz de Programación de Aplicaciones (Application Programming Interface)
BW	Ancho de Banda (Bandwidth)
BEB	*Backoff* Exponencial Binario (Binary Exponential Backoff)
CAPWAP	Protocolo de Control y Aprovisionamiento de Puntos de Acceso Inalámbrico (Control And Provisioning of Wireless AP)
CDMA	Acceso Múltiple por División de Códigos (Code Division Multiple Access)
CSMA	Acceso Múltiple por Detección de Portadora (Carrier Sense Multiple Access)
CSMAC	MAC basado en Planificación Centralizada (Centralized Scheduling based MAC)
CSMA/CA	Acceso Múltiple por Detección y Evasión de Colisiones (CSMA with Collision Avoidance)

CSP	Ruta Más Corta Restringida (Constrained Shortest Path)
CW	Ventana de Contención (Contention Window)
DAC	Control de Admisión Distribuido (Distributed Admission Control)
DCF	Función de Coordinación Distribuida (Distributed Coordination Function)
DIFS	Espacio Intertrama Distribuido (Distributed Inter-Frame Space)
DSCP	Punto de Código de Servicios Diferenciados (Differentiated Services Codepoint)
DSSS	Espectro Ensanchado por Secuencia Directa (Direct Sequence Spread Spectrum)
EDCA	Acceso Distribuido Mejorado al Canal (Enhanced Distributed Channel Access)
EIFS	Espacio Intertrama Extendido (Extended Inter-Frame Space)
FIFO	Primero en Entrar Primero en Salir (First In First Out)
FQA	Agente de QoS Justo (Fair QoS Agent)
GCR	Groupcast con Reintentos (Groupcast with Retries)
HCF	Función de Contención Híbrida (Hybrid Contention Function)
HCCA	Acceso Controlado por HCF al Canal (HCF Controlled Channel Access)
HTTP	Protocolo de Transferencia de HiperTexto (Hypertext Transfer Protocol)
IBSS	Conjunto de Servicios Básicos Independientes (Independent Basic Service Set)
IEEE	Instituto de Ingeniería Eléctrica y Electrónica (Institute of Electrical and Electronics Engineers)
ICMP	Protocolo de Mensajes de Control de Internet (Internet Control Message Protocol)
IMAL	L´ımite de Reintento Adaptativo Mejorado para MAC (Improved MAC Adaptative Retry Limit)
IP	Protocolo de Internet (Internet Protocol)
ITU-T	Sector de Normalización de las Telecomunicaciones de la UIT (Telecommunication Standardization Sector)
LAN	Red de Área Local (Local Area Network)

LVAP	AP Virtual Ligero (Light Virtual AP)
LARAC	Relajación Lagrangiana basada en el Costo Agregado (Lagrangian Relaxation based Aggregated Cost)
LEMON	Biblioteca de Modelamiento y Optimización Eficiente de Redes (Library for Efficient Modeling and Optimization Networks)
MAC	Control de Acceso al Medio (Medium Access Control)
MIMO	Múltiple Entrada Múltiple Salida (Multiple Input Multiple Output)
MPDU	Unidad de Protocolo de Datos MAC (MAC Protocol Data Unit)
MSDU	Unidad de Servicio de Datos MAC (MAC Service Data Unit)
NFV	Virtualización de Funciones de Red (Network Functions Virtualization)
NNS	Espacios de Nombres de Redes Linux (Linux Network Namespaces)
NTXOP	Número de Oportunidades de Transmisión (Number of TXOP) OF Protocolo OpenFlow (OpenFlow Protocol)
OFDM	Multiplexación por División de Frecuencia Ortogonal (Orthogonal Frequency Division Multiplexing)
ONF	Open Networking Foundation
OQAM	Qos Optimizado Adaptativo Multicapa (Optimized Qos Adaptive Multilayer)
OSI	Interconexión de Sistemas Abiertos (Open Systems Interconnection)
OVS	Open vSwitch
PCF	Función de Coordinación de Punto (Point Coordination Function)
PIFS	Espacios Intertramas PCF (PCF Inter-Frame Space)
PHY	Capa F´ısica (Physical)
QoE	Calidad de la Experiencia (Quality of Experience)
QoS	Calidad de Servicio (Quality of Service)
RF	Radiofrecuencia (Radio Frequency)
RFC	Solicitud de Comentarios (Request For Comments)
RTP	Protocolo de Tiempo Real (Real-time Protocol)
RTS/CTS	Petición de Envío/Libre para el Envío (Request To Send)/Clear To Send)
SCS	Servicio de Clasificación de Flujo (Stream Classification Service)
SDN	Redes Definidas por Software (Software Defined Networking)

SDWN	Redes Inalámbricas Definidas por Software (Software Defined Wireless Network)
SNR	Relación Señal a Ruido (Signal to Noise Ratio)
SIFS	Espacios Intertramas Corto (Short Inter-Frame Space)
ToS	Tipo de Servicio (Type of Service)
TSPEC	Especificación del Tráfico (Traffic Specification)
TXOP	Oportunidad de Transmisión (Transmission Opportunity)
TXOPLimit	Límite de Oportunidad de Transmisión (Transmission Opportunity Limit)
VA-MSDU	Agregación de Video MSDU (Video Aggregation MSDU)
VoIP	Voz sobre Protocolo de Internet (Voice over Internet Protocol)
Wi-Fi	Wireless Fidelity
WLAN	Red inalámbrica de Área local (Wireless Local Area Network)
WME	Extensión Multimedia Wi-Fi (WiFi Multimedia Extension)
WMM	Wi-Fi Multimedia

Dedicado a la memoria de mis abuelos

Capítulo 1

Introducción

1.1. Contexto

Aparte de las redes celulares, las redes inalámbricas de área local (*Wireless LAN*, WLAN) -basadas en el estándar IEEE 802.11- representan el mayor despliegue de tecnología inalámbrica en todo el mundo, debido a la proliferación de dispositivos habilitados con esta tecnolog´ıa [4] [5]. Con estas redes se da un crecimiento acelerado de aplicaciones de audio y/o video, tales como video en l´ınea, videoconferencias, voz sobre IP, etc., las cuales cada vez tienen mayores requerimientos para garantizar la calidad de servicio (QoS). Sin embargo, la naturaleza poco confiable del medio inalámbrico se traduce en la necesidad de un control de tráfico más exigente que en las redes cableadas [6]. Esto hace de gran importancia la tarea de proporcionar QoS en redes IEEE 802.11 [4].

En la primera generación del estándar IEEE 802.11, las aplicaciones multimedia no contaban con servicios o mecanismos explícitos para la diferenciación del tipo de tráfico. Por este motivo, se dificultaba que la red ofreciera QoS [7]. A partir de la enmienda IEEE 802.11e con EDCA (*Enhanced Distributed Channel Access*), el estándar ofrece mecanismos para soportar QoS en la capa MAC, incorporando priorización y políticas de colas. Estas políticas permiten garantizar servicios diferenciados entre las aplicaciones de la red [4]. A pesar de esto, el estándar IEEE 802.11e no provee garantías para asegurar niveles adecuados de rendimiento y retardo para los servicios de voz y/o video [8].

Han surgido también otras adendas al estándar de las redes inalámbricas WLAN para manejar las crecientes exigencias del tráfico en la red. Entre estas, se pueden mencionar IEEE 802.11g, IEEE 802.11n e IEEE 802.11ac. En dichas adendas se incluyen servicios como el control de acceso, gestión de movilidad, equilibrio de carga, reconfiguración dinámica de canales, entre otros. De igual manera existen soluciones WLAN propuestas

por diferentes proveedores con algunos de los servicios antes mencionados (p.ej. Cisco CleanAir [9], HP-Aruba AirMatch [10]), pero la mayoría son de código cerrado y alto costo. Esto limita la flexibilidad y expansión de la red [11].

Es decir, para redes a gran escala con cientos de conmutadores, enrutadores y *hosts*, que en el peor de los casos pueden ser de distintos fabricantes con la lógica de control embebida y varios protocolos de red, requieren para su configuración y administración personal altamente capacitado y gran cantidad de trabajo. Debido a que cada dispositivo de red debe configurarse por separado cuando se requiere implementar una nueva política global. Por lo tanto, cambiar la red o probar nuevas políticas será costoso [11].

Una posible alternativa para afrontar la lógica de control embebida y propietaria, además de la poca flexibilidad de los dispositivos tradicionales son las redes definidas por software (*Software Defined Networking*, SDN). Este es un paradigma emergente que promete cambiar el estado de las cosas al romper la integración vertical, promover la centralización lógica del control e introducir la capacidad de programar las funciones de la red. La separación de las tareas para la formulación de políticas, su implementación en el hardware de conmutación y el reenvío del tráfico son clave para permitir la flexibilidad deseada al dividir el problema de control en piezas tratables. SDN posibilita la creación e introducción de nuevas abstracciones como virtualización, topologías ó información de los estados de enlaces, simplificando su administración y facilitando su evolución [12].

Además, SDN puede ayudar a los administradores de red a crear plataformas de administración con mecanismos de QoS automatizados reconfigurables para la reserva de recursos, el monitoreo de la actividad de las colas [13] y control de admisión previo a las colas [14]. Aunque el aprovisionamiento de aplicaciones de QoS requiere mecanismos de control bien definidos debido a la naturaleza dinámica de los recursos de red. En el paradigma SDN el control para la configuración de QoS se vuelve más sencillo en comparación con las arquitecturas tradicionales, ya que SDN permite tener una visión global del estado actual de toda la red gracias a que el controlador recibe información del estado de todos los equipos de red de la topología [15] [16].

Tradicionalmente SDN ha sido usado para redes cableadas, especialmente en escenarios LAN, WAN y centros de datos. Este paradigma también se ha expandido a las redes inalámbricas definida por software (*Software Defined Wireless Network*, SDWN). La principal ventaja de usar SDWN es su capacidad para admitir la implementación de nuevas aplicaciones en la red; esta caracter´ıstica ayuda a SDWN a resolver los desaf´ıos presentes en las arquitecturas de redes inalámbricas convencionales y, también, a promover una rápida innovación en la red [17]. SDWN se implementó en arquitecturas WLAN [18] y celulares [19], luego se expandió a redes *mesh* [20], Ad hoc [21] [22], entre otras.

En redes WLAN, SDN ha sido un campo importante de investigación en proyectos tales como: Odin [18], CloudMAC [23], Thor [24], OpenSDWN [25], Mininet-WiFi [26] y EmPOWER [27], que promueven el diseño, la implementación y evaluación de plataformas de gestión lógicamente centralizadas para las redes inalámbricas de área local bajo el paradigma SDN.

1.2. Planteamiento del problema

El crecimiento de la demanda por acceso de dispositivos inalámbricos se ha incrementado de manera continua y con ello el despliegue de redes inalámbricas de área local IEEE 802.11 (WLAN) [28]. Al mismo tiempo se ha incrementado el uso de aplicaciones de tiempo real como VoIP, videollamadas, videoconferencias, juegos en l´ınea, entre otros, con exigencias cada vez mayores de QoS [4]. Para proporcionar garant´ıas de QoS, los desarrolladores de redes deben integrar mecanismos que permitan la diferenciación entre los tipos de tráfico, de manera que se pueda garantizar la asignación de recursos a las diferentes aplicaciones de tiempo real con requisitos estrictos en términos de retardos y pérdidas de paquetes [28].

Aunque los dispositivos de red tradicionales que incorporan EDCA han demostrado soportar QoS, las decisiones de admisión de flujos son tomadas localmente en cada AP sin ningún mecanismo de coordinación con las celdas vecinas. Esta falta de coordinación no da garantías de que los flujos admitidos encuentren, en última instancia, recursos suficientes para su adecuado desempeño [29]. Otra limitación de EDCA surge respecto al uso del mismo *buffer* para almacenar todos los flujos con diferentes prioridades, lo que ocasiona la pérdida del tráfico de más baja prioridad [30].

Por otra parte, el efecto de la sobrecarga de canales no es deseable para el tráfico inelástico, como el de las aplicaciones multimedia que requieren garant´ıas adecuadas de rendimiento y retardo [4]. Es decir, el desempeño de la red se ve degradado bajo una gran carga de tráfico de alta prioridad [31]. Esto se debe a que los recursos del canal deben ser distribuidos entre los diferentes flujos activos de acuerdo a su prioridad, lo que significa que los servicios de voz/video no serán soportados adecuadamente, debido a que no pueden adaptarse a estas limitaciones [8].

Además, es importante indicar que resulta complejo tomar decisiones de control sobre los recursos de red basados solo en el rendimiento de los flujos activos. Esto debido a que la capa MAC tiene funcionalidades de control preestablecidas (generación de *beacons*, clasificación, manejo de colas, planificación, etc.), que son propietarias y de código cerrado [11]. Las redes inalámbricas aún tienen retos en su administración, configuración

y control que los fabricantes tratan de resolver por medio de soluciones propietarias. La mayoría de esas soluciones comparten dos elementos comunes: dividen las funcionalidades que proporcionan los APs, y añaden funciones centralizadas para la supervisión y el control remoto de la red.

El protocolo de control y aprovisionamiento de puntos de acceso inalámbrico (*Control And Provisioning of Wireless Access Points*, CAPWAP) propone separar las funcionalidades de los APs para implementar infraestructuras de red más flexibles [32].

El grupo IETF (*Internet Engineering Task Force*) ha intentado promover el control y la administración centralizada mediante la estandarización de CAPWAP [33], pero este protocolo sigue dependiendo únicamente del controlador de acceso -tanto para el plano de control, como para el plano de datos- en un mismo dispositivo físico, lo que limita la flexibilidad requerida en la red [34].

Debido a problemas como la complejidad que impone la gestión centralizada sobre el despliegue de infraestructura WLAN tradicional en ambientes institucionales, la modificación de soluciones propietarias y cerradas, la edición de los parámetros por defecto de la capa MAC del protocolo 802.11 con las restricciones del estándar EDCA, la demanda del uso de mecanismos de control flexibles y las exigencias de QoS de las aplicaciones de tiempo real. Demandan el uso de mecanismos de control flexibles y programables que se puedan implementar en redes inalámbricas gestionadas bajo el paradigma SDN.

Además, para proveer calidad de servicio en escenarios de distribución de video con un gran número de dispositivos solicitando aplicaciones multimedia que requieren mecanismos específicos de control [3]. Las técnicas que han sido propuestas tradicionalmente para manejarlas no han sido completamente estandarizadas [35] o ajustadas para operar en escenarios WLAN gestionadas con SDN.

Dentro de este contexto surge la siguiente pregunta de investigación: *¿La implementación de un mecanismos de mantenimiento de QoS en redes inalámbricas institucionales bajo el esquema de gestión SDN, garantizará los niveles de rendimiento y retardo exigidos en la transmisión exitosa de video en tiempo real?*.

Se buscará una respuesta a esa pregunta mediante la evaluación y selección de un mecanismo de QoS no estandarizado orientado a aplicaciones multimedia, propuesto en la literatura. Se describirá y analizará su modo de operación para obtener los parámetros de rendimiento necesarios para la toma de decisiones. Seguidamente, se replicará para fines comparativos el modo de operación de EDCA definido por el estándar IEEE 802.11. Finalmente, se adaptará el escenario WLAN tradicional a un escenario gestionado con SDN, que ejecutará aplicaciones de QoS sobre el tráfico de video en tiempo real.

1.3. Motivación

En un esfuerzo por llevar nuevas funcionalidades de control que garanticen QoS en redes gestionadas bajo el paradigma SDN, a partir de mecanismos no estandarizados orientado inicialmente a redes WLAN tradicionales, surge el propósito de esta investigación.

Es decir, el propósito de esta investigación es la implementación de un mecanismo de QoS de bajo costo e independiente del fabricante, que se pueda configurar y adaptar a las necesidades de una institución. Esta solución posibilitará que los flujos admitidos en las estaciones seleccionadas mantengan un buen desempeño en cuanto al *throughput*, sin llegar a la condición de saturación sobre un entorno WLAN institucional.

Considerando lo anterior, se implementó un mecanismo de control de admisión que ofrece garantía de QoS sobre topologías de red en modo infraestructura y se evaluó su desempeño en un conjunto de estaciones inalámbricas transmitiendo video en tiempo real en escenarios SDWN. Este desarrollo compara el comportamiento del flujo con y sin garantías de QoS en un entorno de simulación y emulación WLAN.

1.4. Objetivo de Investigación

Implementar y evaluar una estrategia que garantice el mantenimiento de QoS para transmisión de video en tiempo real en redes inalámbricas institucionales bajo el esquema de gestión SDN.

1.5. Organización del proyecto de investigación

En el Capítulo 2 se presenta el marco teórico y estado del arte. Específicamente, lo relacionado al estándar IEEE 802.11, calidad de servicio, mecanismos de control de admisión y las definiciones que tienen que ver con las redes inalámbricas gestionadas mediante SDN, así como los protocolos de operación y algunas investigaciones relacionadas con QoS en SDN.

En el Capítulo 3 se presentan los fundamentos para la implementación de un modelo analítico de control de admisión y la formulación matemática para la obtención del rendimiento en condiciones de saturación de la red bajo el estándar IEEE 802.11e. Estos resultados son el fundamento en la toma de decisión del control de admisión implementado para la arquitectura de red inalámbrica basada en SDN.

En el Capítulo 4 se presenta la principal contribución de esta investigación. Se explica la implementación de un escenario de simulación del mecanismo, por defecto, EDCA para fines comparativos a la operación del estándar. Luego, se adapta el escenario WLAN tradicional a un escenario de emulación para tráfico de video en tiempo real. Este último sobre una arquitectura de red inlámbrica gestionada con SDN. Se evidencian, a su vez, los resultados de la evaluación del mecanismo de control de admisión propuesto respecto al estándar, en términos de métricas de QoS, como rendimiento y retardo.

Finalmente, en el Cap´ıtulo 5 se presentan las conclusiones y trabajos futuros.

Capítulo 2

Calidad de servicio y mecanismos de control de admisión en redes WLAN y SDN

En este capítulo se presentará el marco teórico y el estado del arte de los mecanismos de calidad de servicio orientados a las redes tradicionales WLAN y SDN. En la primera sección del marco teórico se muestran las generalidades de las redes WLAN con su respectivo estándar IEEE 802.11. Luego se define calidad de servicio (QoS) y se muestra, a partir de la literatura, la clasificación de las principales métricas de evaluación de desempeño, como rendimiento, retardo, jitter, tasa de pérdida de paquetes, entre otras. Después se muestra brevemente la evolución de las adendas que ha tenido el estándar desde IEEE 802.11b. Se resalta el modo de operación de la enmienda denominada EDCA, la cual permite categorizar y diferenciar las clases de tráfico bajo prioridades y, de esta manera, distinguir los flujos multimedia con QoS.

Posteriormente, se presentan los conceptos asociados al mecanismo de control de admisión y su respectiva clasificación. Por otra parte, se presenta el protocolo CAPWAP como un desarrollo intermedio antes de SDN, el cual propone centralizar la administración de la red. Luego se definen los conceptos asociados a las redes definidas por software, su arquitectura y principal protocolo de operación. Esta sección finaliza describiendo la integración de SDN a las redes WLAN bajo el término SDWN.

En otro apartado de este cap´ıtulo se presenta el estado del arte -centrado en algunas de las propuestas encontradas en la literatura- sobre los mecanismos tradicionales de garantías de QoS no estandarizados para IEEE 802.11. También se presentan algunas propuestas sobre plataformas que utilizan los principios de operación de SDN para redes cableadas.

Luego esta sección presenta algunas investigaciones enfocadas en la implementación de mecanismos que ofrecen garant´ıa de QoS en plataformas gestionadas con la arquitectura emergente SDN para redes WLAN.

Finalmente, como resultado de la revisión de la literatura se selecciona el modelo de mantenimiento de calidad de servicio para el estándar IEEE 802.11 que mejor se ajusta a las condiciones de replicabilidad, adaptabilidad y no intervención en la modificación de los parámetros por defecto de la capa MAC del estándar.

2.1. Marco Teórico

2.1.1. Estándar IEEE 802.11 WLAN

El estándar IEEE 802.11 para Redes Inalámbricas de Área Local (WLAN) -comúnmente conocidas como redes WiFi- es una tecnología madura con más de 20 años de desarrollo y estandarización. La primera versión del estándar IEEE 802.11 se publicó en 1997 como una alternativa o extensión inalámbrica de las LAN que utilizaban tecnología Ethernet. Desde su aparición, la especificación IEEE 802.11 ha evolucionado continuamente para incluir nuevas tecnologías y funcionalidades. Además, se han desarrollado modificaciones al estándar IEEE 802.11 original. Actualmente las WLAN no son solo la tecnología de acceso a Internet más común, sino que también se han expandido en una amplia variedad de mercados como el móvil y automotriz. De hecho las WLAN están ampliamente disponibles en diversos espacios (hogares, zonas públicas, entornos empresariales) y las interfaces de radio basadas en IEEE 802.11 se encuentran en una inmensa variedad de dispositivos [36].

La mayoría de las adendas al estándar IEEE 802.11 funcionan en las bandas de frecuencia de 2,4 GHz y 5 GHz, que están disponibles a nivel mundial. Por lo tanto, cualquiera puede implementar una WLAN en esas bandas cumpliendo restricciones básicas como la potencia de transmisión máxima, la ganancia de las antenas o el tipo de modulación. El aspecto negativo es que la mayor´ıa de las WLAN se implementan de forma descontrolada, con problemas de interferencia o sin consideraciones técnicas. Esto ha hecho especialmente dif´ıcil garantizar l´ımites adecuados de rendimiento y niveles razonables de calidad de servicio (QoS) [36].

2.1.2. Calidad de servicio (QoS)

QoS se define como la capacidad de proporcionar garantías específicas al tráfico, respecto a características, tales como pérdida de paquetes, retardo y jitter experimentado por

los flujos [37]. Con la demanda de aplicaciones multimedia y las exigencias de niveles adecuados de QoS para las redes de telecomunicaciones, es necesario garantizarle a estas aplicaciones la diferenciación de servicios mediante la asignación justa de recursos de red. Es importante reconocer que sin una adecuada asignación de recursos, el desempeño de la red y la calidad del servicio se deterioran rápidamente [4].

2.1.3. Parámetros para evaluar QoS

Los requerimientos de calidad de servicio son generalmente especificaciones de las aplicaciones, algunas son sensibles al retardo y al rendimiento -como las videoconferencias-, mientras que otras son susceptibles a la variación en la tasa de transmisión, por ejemplo: volumen de transferencia de archivos. Los requerimientos de servicios son expresados frecuentemente usando métricas como: ancho de banda, rendimiento, retardo, jitter y tasa de pérdida. La taxonomía propuesta por Malik [4] que se muestra en la Figura 2.1 reúne las principales métricas que se encuentran en la literatura.

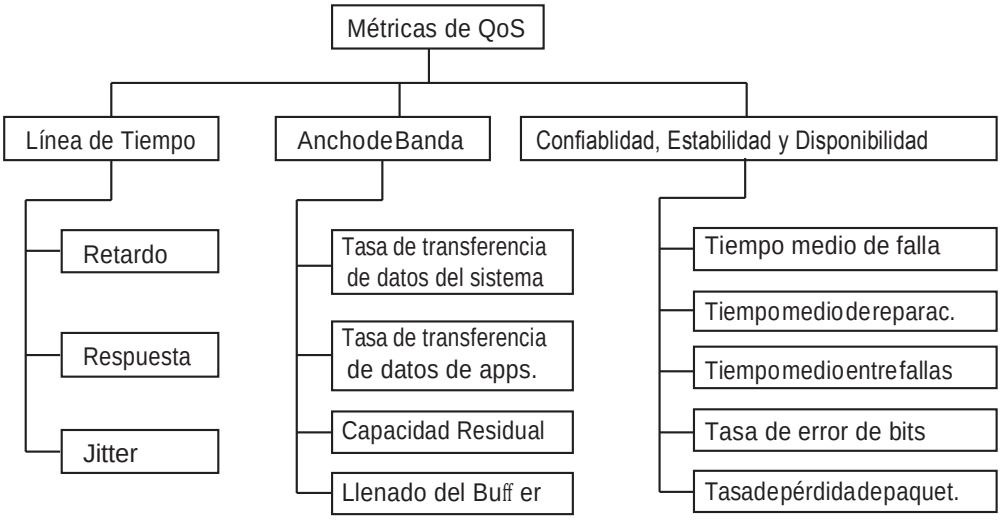

Figura 2.1: Métricas de QoS. [4]

El retardo (*delay*), representa el tiempo que tarda un paquete en llegar a su destino [37]. El jitter es la variación del tiempo de llegada entre los paquetes. Mientras que el rendimiento (*throughput*) es el número de paquetes entregados con éxito por unidad de tiempo [4].

2.1.4. Estándar IEEE 802.11e

El primer estándar que apareció para WLAN define las especificaciones de capas físicas (PHY) y de control de acceso al medio (MAC) que proporcionan velocidades de transmisión de hasta 2 Mbps en el rango de 2.4 GHz. El estándar IEEE 802.11b (1999) que

opera en la banda de 2.4 Ghz proporciona hasta 11 Mbps usando la técnica de modulación DSSS. IEEE 802.11a (1999) maneja velocidades hasta 54 Mbps usando el esquema de modulación OFDM y el estándar IEEE 802.11g (2003) soporta velocidades de hasta 54 Mbps usando los esquemas de modulación OFDM y DSSS.

En el estándar IEEE 802.11, el mecanismo de acceso al medio fundamental se denomina Función de Coordinación Distribuida (DCF) y un mecanismo de acceso opcional es la Función de Coordinación de Puntos (PCF). El DCF se basa en el método de acceso múltiple por detección de portadora y prevención de colisiones (CSMA/CA), desarrollado para los servicios *best effort* [28]. Para PCF, el problema de la calidad del servicio no se pudo resolver y en consecuencia, los servicios prestados a los usuarios no tienen un rendimiento óptimo para varias aplicaciones, incluidas las aplicaciones de audio y video durante cargas pesadas en la red [4].

La especificación IEEE 802.11e (2005) para redes WLAN se desarrolla con el fin de proporcionar una capa MAC con QoS. La calidad de servicio se da por la discriminación del tráfico a través de la definición de cuatro categorías de acceso a nivel de capa MAC con diferentes prioridades entre ellas. Tal categorización se logra gracias al acceso distribuido o basado en contienda de 802.11e denominado EDCA (*Enhanced Distributed Channel Access*), una versión compatible con DCF [3].

EDCA considera cuatro clases de tráfico (o categorías de acceso, ACs): voz (AC_VO), video ($AC\ VI$), *best effort* (mejor esfuerzo, $AC\ BE$) y *background* (segundo plano, $AC\ BK$). Cada una de estas ACs maneja una cola independiente en cada estación, las cuales tiene diferentes parámetros entre los que se encuentran el Arbitraje de Espacios Intertramas (AIFS) y Ventana de Contención (CW), como muestra la Figura 2.2, y un contador binario exponencial llamado *backoff* asociado a cada cola independientemente (ver Figura 2.3). Las colisiones entre paquetes de la misma estación (colisiones virtuales) se resuelven a favor del AC con mayor prioridad (ver Tabla 2.1).

Además de AIFS se utilizan otros parámetros para diferenciar las clases de flujo como: máxima y mínima ventana de contención (CW_{min}, CW_{max}), tiempo de *backoff* y la máxima cantidad de tiempo que cada AC puede utilizar el canal durante cada oportunidad de acceso al canal (TxOP, *Transmission Oportunity*) [38].

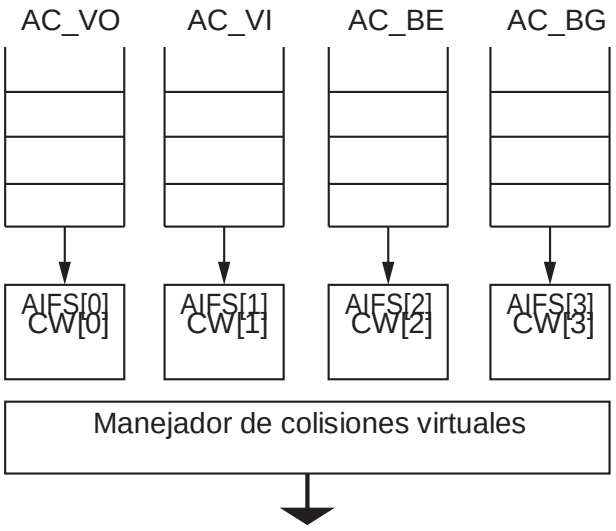

Figura 2.2: Manejador de AIFS y CW por cada AC. Adaptado de [1].

Tabla 2.1: Asignación de las prioridades de usuario de IEEE 802.1D a las categorías de acceso IEEE 802.11e [1]

Prioridad de Usuario	Designación 802.1D	ACs	Descripción
1	BK	AC_BK	Background
2	-	AC_BK	Background
0	BE	AC_BE	Best Effort
3	EE	AC_BE	Best Effort
4	CL	AC_VI	Video
5	VI	AC_VI	Video
6	VO	AC_VO	Voz
7	NC	AC_VO	Voz

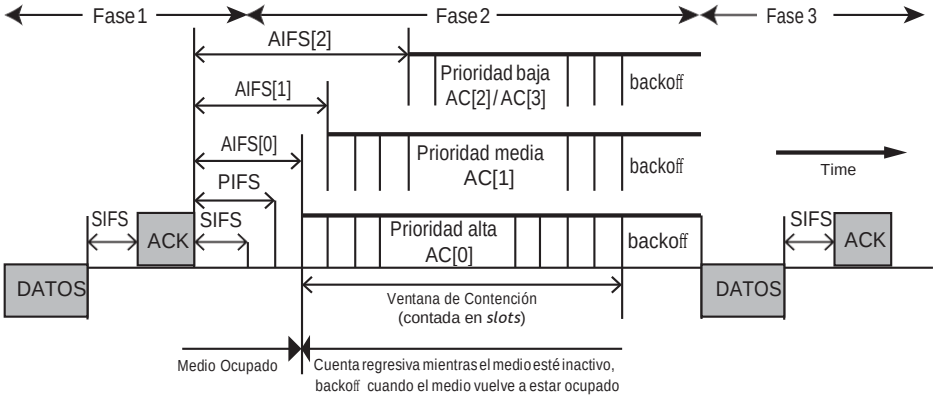

Figura 2.3: Operación del *Backoff* por AC.

Aunque con EDCA se anunciaron esquemas de diferenciación de servicios, las aplicaciones de baja prioridad se ven afectadas debido a que son retenidas en la cola del *buffer* de la capa MAC. Pero las aplicaciones con requerimientos muy estrictos logran únicamente mejoras en el desempeño, pero sin resolver el problema de la degradación en el momento en el que el canal se satura [3].

Otras enmiendas al estándar IEEE 802.11 han surgido para soportar aplicaciones multimedia y altas tasas de transferencia, entre las principales están: IEEE 802.11n (2009), IEEE 802.11ac (2013), IEEE 802.11ah (2016), IEEE 802.11ax (2019), IEEE 802.11aa (en desarrollo).

Para admitir aplicaciones multimedia u otras aplicaciones de alto consumo de recursos de red, surge la necesidad de proveer funcionalidades de QoS que proporcionen garant´ıas de recursos junto a la diferenciación de servicios. De manera general, la literatura clasifica los mecanismos de QoS en: clasificación (priorización), control de admisión y planificación [28]. En primer lugar, la priorización de paquetes se utiliza para organizar los flujos en diferentes tipos de tráfico. Una vez clasificados los paquetes, se realiza un mecanismo de control de admisión para tomar la decisión de aceptar o rechazar el flujo entrante de acuerdo con sus requerimientos de QoS y los recursos de red disponibles; después los paquetes se planifican y se colocan en cola de los *buffers* de memoria para su transmisión [39].

Malik et al. [4] proponen otros esquemas de clasificación de mecanismos como reserva de recursos, control de congestión, diferenciación de servicios y otras como las de administración centralizada orientadas a las arquitecturas de redes modernas. En las siguientes subsecciones se describirán los mecanismos que presentan mayor relación con el alcance propuesto para este proyecto de investigación.

2.1.5. Asignación de recursos

Una red tiene diferentes recursos, tales como ancho de banda, tamaño del *buffer*, rutas, entre otros. Estos recursos deben ser gestionados y asignados adecuadamente, de lo contrario, el rendimiento de la red y la calidad del servicio se enfrentarán a problemas como la pérdida de paquetes, alta congestión o bajo desempeño.

Existen dos enfoques arquitectónicos sobresalientes que garantizan calidad de servicio mediante la asignación de recursos en Internet: Servicios integrados (IntServ, RFC 1633 [40]) y Servicios diferenciados (DiffServ, RFC 3260 [41]). En IntServ el planificador de paquetes se usa para reservar los recursos en todo el camino que debe atravesar un flujo individual. En DiffServ, al hacer uso de la priorización, se reservan recursos por

clases para la diferenciación del servicio, el tráfico se trata de acuerdo a su respectiva clase. Este último enfoque arquitectónico ha sido usado para implementar QoS en redes inalámbricas basadas en IEEE 802.11 [4].

2.1.5.1. Servicios integrados (IntServ)

En IntServ, la reserva de recursos se basa en el flujo de paquetes. Este flujo tiene direcciones IP y puertos, tanto de origen como de destino junto al protocolo usado [42]. IntServ utiliza un planificador para asignar recursos a flujos de paquetes individuales junto con el soporte de priorización. El planificador se utiliza para obtener los retardos límites (de manera estadística o determinista). Además, IntServ tiene dos abstracciones principales, las cuales son recursos reservados y recursos estándares. En el primer caso, los enrutadores deberian ser conscientes de la existencia de los recursos reservados para el proceso de transmisión y en el segundo caso se le indica al enrutador que debe tener en cuenta la capacidad del enlace y del *buffer* usado [40].

2.1.5.2. Servicios diferenciados (DiffServ)

DiffServ es una reserva de recursos por clase que clasifica el tráfico, utiliza priorización y es capaz de reenviar el tráfico a múltiples clases. DiffServ utiliza la regla de manejo de paquetes por salto. A lo largo de la ruta cada salto se realiza de acuerdo con los valores de prioridad de un flujo de paquetes específico [41].

2.1.6. Priorización

Este método utiliza colas con distintas prioridades en la capa MAC, donde los paquetes de datos se separan en función de las prioridades. Siempre que una estación particular tenga acceso al canal, se transmiten los paquetes que estén en la cola con mayor prioridad. Todas las estaciones deben competir entre sí para acceder al medio. La cola de prioridad se hace de forma que haya diferentes niveles de prioridad y, por lo tanto, se deben mantener diferentes colas [4].

2.1.7. Planificación

La planificación es la clave para compartir recursos de red de forma equitativa entre los usuarios activos y proporcionar garantías de servicio a aplicaciones con restricciones de tiempo. El planificador gestiona las colas y decide primero el orden de las solicitudes que

deben notificarse. La planificación puede proporcionar diferentes servicios a los flujos usando parámetros como el ancho de banda, sirviendo un solo flujo en un intervalo particular de tiempo de acuerdo con el nivel de prioridad definido y su tasa de pérdida. La asignación de espacio en *buffers* se adapta dependiendo las condiciones de la red [4].

2.1.8. Control de admisión

El control de admisión es una técnica en sistemas de comunicaciones que realiza un proceso de verificación antes de establecer una conexión y, de esta manera, confirma que los recursos presentes en la red sean los suficientes para garantizarle los requerimientos a esta conexión [43].

El estándar IEEE 802.11e tiene mecanismos de acceso al medio habilitados para QoS. Uno de ellos es el control de admisión. Este es un componente importante para la provisión de garantías de parámetros de QoS debido a su capacidad de limitar la cantidad de tráfico admitido en una clase de servicio particular. Así posibilita que el QoS de los flujos ya admitidos no se degrade y, al mismo tiempo, se aprovechen al máximo los recursos del medio [1].

Originalmente, los sistemas de control de admisión de EDCA podían clasificarse en dos categor´ıas: basados en medidas y basados en modelos [44] [45]. En los sistemas basados en medidas las decisiones de control de admisión se toman según las condiciones de la red, como son el rendimiento y el retardo. Por otra parte, los esquemas basados en modelos construyen parámetros de rendimiento en saturación para predecir el estado de la red. En literatura más reciente se habla de una tercera clasificación llamada híbrida, como la combinación de ambas categorías [4].

2.1.8.1. Control de admisión basado en medidas

En este esquema, las decisiones se toman a través del monitoreo continuo de las condiciones de la red, bajo la evaluación de parámetros como el rendimiento y el retardo. Un primer esquema de esta categoría es el Control de Admisión Distribuido (DAC) propuesto por el grupo de trabajo IEEE 802.11e para proteger los flujos activos con QoS. En el DAC, el AP anuncia a través de *beacons* el presupuesto de transmisión, que es la cantidad de tiempo adicional disponible para cada AC durante el siguiente intervalo *beacon*[1]. Para calcular este presupuesto, el AP necesita medir la cantidad de tiempo que ocupa la transmisión de cada AC durante intervalos *beacons*.

[1]Tiempo transcurrido entre transmisiones *beacon*. *Beacon* es una señal enviada cada determinada cantidad de tiempo para indicar a los clientes inalámbricos que la red inalámbrica aún está activa.

Un problema del esquema DAC es la dificultad para estimar las variaciones del rendimiento de la red, ya que una estación siempre necesita ajustar sus parámetros de transmisión en cada intervalo *beacon* [1].

2.1.8.2. Control de admisión basado en modelos

En los esquemas basados en modelos, el estado de la red se mide en función de dichos modelos matemáticos. En este enfoque, el AP se basa en el cálculo numérico para predecir las métricas de rendimiento antes de tomar cualquier decisión. Las métricas de QoS se predicen anal´ıticamente sin necesidad de introducir el nuevo flujo [8].

Los de cadena de Markov son importantes en los intentos de modelar el comportamiento del estándar IEEE 802.11 [4]. Enmarcados en este concepto Taher et al. [8] desarrollaron un modelo anal´ıtico para predecir los niveles de QoS que se pueden lograr una vez que se introduce un nuevo flujo de voz/video en la WLAN.

El esquema propuesto por Jeonghoon Mo [46] estima el ancho de banda que los flujos presentes alcanzarían si se admitieran nuevos flujos con garantías de QoS. Solo podrá aceptarse una nueva solicitud de flujo de tráfico en la red si la estimación demuestra que el ancho de banda requerido se puede alcanzar y no se ponen en peligro las garant´ıas de ancho de banda de los flujos ya admitidos. El modelo anal´ıtico se deriva bajo condiciones de saturación, ya que el control de admisión generalmente se vuelve asertivo cuando la red está saturada [8].

La ventaja de utilizar algoritmos de control de admisión basados en modelos es la garantía cuantitativa de predecir parámetros como el ancho de banda, el rendimiento o el retardo. Sin embargo, una estimación precisa del ancho de banda requiere procesos computacionales pesados que son inaceptables para la resolución de problemas de admisión en tiempo real [44].

2.1.8.3. Control de admisión híbrido

Para los mecanismos de control de admisión, la fusión de la metodología basada en medidas y basada en modelos produce la metodología híbrida. Aquí el AP obtiene en cada intervalo de tiempo los valores de desempeño de la red para ajustar sus parámetros. Pero las decisiones del mecanismo se basan en los valores de rendimiento proporcionados por el modelo analítico según las condiciones suministradas por la red [8]. Este mecanismo aprovecha los beneficios de los dos mecanismos anteriores.

2.1.9. CAPWAP

El control y aprovisionamiento de puntos de acceso inalámbrico CAPWAP es un protocolo WLAN para coordinar la distribución física de los APs mediante una gestión lógica centralizada. Una de las principales características de este protocolo es que al ofrecer una administración centralizada del sistema WLAN garantiza una mejor implementación de políticas de control y aplicaciones consolidadas. A través de un controlador de acceso, CAPWAP ofrece el monitoreo y el control de la red, así como desempeño adecuado de QoS mediante manejo eficiente del balanceo de carga entre los diferentes APs, consolidando así la distribución y la integración de la carga de tráfico entre los diferentes dispositivos [33].

Por otra parte, el protocolo CAPWAP permite que el AP ofrezca calidad de servicio sobre los flujos de datos y los mensajes de control de la red, de manera que los datos son etiquetados con diferentes prioridades para garantizar QoS, tal como lo propone la WMM (*Wireless Multimedia*) [33].

Aunque el protocolo intenta resolver tanto el control como el aprovisionamiento y gestión de la red en un solo dispositivo físico, CAPWAP presenta limitaciones de interoperabilidad entre fabricantes de APs para usar el mismo controlador de acceso [34] y no permite la programabilidad sobre los dispositivos de red que posibilite la implementación y despliegue de políticas de control dinámicas y personalizadas para una organización.

2.1.10. Redes definidas por software (SDN)

SDN está cambiando la forma en que se diseñan y administran las redes. Tiene dos características que la definen. Primero, SDN separa el plano de control (que decide cómo manejar el tráfico) del plano de datos (que reenvía el tráfico de acuerdo con las decisiones que toma el plano de control). Segundo, SDN consolida el plano de control, de modo que un solo programa de control de software gestiona múltiples elementos del plano de datos. El plano de control SDN ejerce el control directo sobre el estado en los elementos del plano de datos de la red (es decir, enrutadores, conmutadores y otros equipos intermedios de manipulación del tráfico) a través de una API [47]. Un esquema de la arquitectura de SDN se puede observar en la Figura 2.4.

En una red definida por software, el plano de control encargado de las operaciones de gestión de la red lo conforman controladores o sistemas operativos de red [12] (p. ej. NOX [48], POX [49], Floodlight [50], OpenDaylight [51]). El controlador se comunica

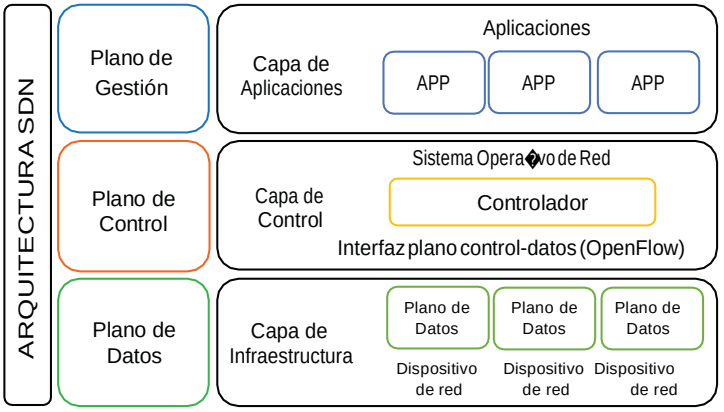

Figura 2.4: Arquitectura SDN adaptada de [12].

con los dispositivos del plano de datos a través de un protocolo *southbound*[2] (p. ej. OpenFlow [52]) y expone un conjunto de APIs *northbound*[3] (p. ej. REST [53]) a través de las cuales se desarrollan mecanismos de control de red. De hecho, la programabilidad del plano de datos posibilita el diseño y la implementación de los mecanismos de control de la red. Estos mecanismos también se conocen como aplicaciones de red, ya que en realidad se ejecutan en el sistema operativo de red [54].

2.1.11. Protocolo OpenFlow

OpenFlow es un protocolo estandarizado impulsado por la Open Networking Foundation (ONF). La ONF se concibió como una organización impulsada para promover la adopción de SDN a través del desarrollo del protocolo OpenFlow como un estándar abierto para comunicar las decisiones de control a los dispositivos del plano de datos [12].

OpenFlow es una interfaz de comunicaciones estándar, definida entre las capas de control e infraestructura para una arquitectura SDN. OpenFlow permite el acceso directo a la manipulación del plano de infraestructura de los dispositivos de red tales como conmutadores y enrutadores, tanto f´ısicos como virtuales.

Openflow tiene una o más tablas de reglas de paquetes. Cada regla corresponde a un flujo del tráfico y realiza ciertas acciones (reenviar, modificar, etc.) sobre los paquetes que pertenecen a ese flujo. Dependiendo de las reglas instaladas por una aplicación del controlador, un conmutador que soporta OpenFlow puede configurarse como un

[2] APIs SDN usadas para la comunicación entre el controlador SDN y dispositivos conmutadores o enrutadores de la red

[3] APIs Rest SDN, usadas para la comunicación entre el controlador SDN y los servicios o aplicaciones que se ejecutan en la red

enrutador, conmutador, *firewall* o desempeñar otras funciones, por ejemplo: balancear la carga, configurar el tráfico, etc. [12].

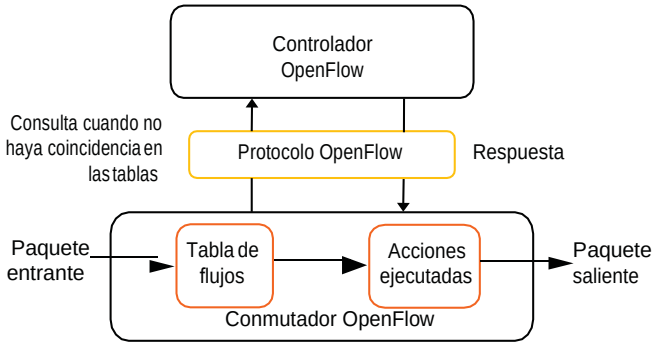

Figura 2.5: Cómo un conmutador OpenFlow maneja los paquetes entrantes [55].

Cada conmutador con soporte de OpenFlow tiene una cadena de tablas de flujo y cada tabla almacena una colección de entradas de flujo. En la Figura 2.5 se evidencia el procedimiento mediante el cual un conmutador OpenFlow maneja un paquete entrante. Cuando un paquete llega a un conmutador, este busca coincidencias en las tablas de flujo y ejecuta las correspondientes listas de acciones. Si no se encuentra ninguna coincidencia para el paquete, el paquete se pone en cola y se envía un evento de consulta al controlador OpenFlow. El controlador responde con una nueva entrada de flujo para manejar ese paquete en cola. Los paquetes subsecuentes en el mismo flujo serán manejados por el conmutador sin contactar al controlador y serán reenviados a la velocidad de operación del conmutador [12].

2.1.12. Redes inalámbricas definidas por software (SDWNs)

Si bien los esfuerzos en SDN se han enfocado principalmente en redes cableadas y centros de datos, se considera que los beneficios de este paradigma se pueden trasladar a redes de acceso inalámbrico y entornos de *backhaul* [4], ya que se pueden lograr ventajas a partir de la flexibilidad proporcionada por la gestión a través de SDN [56].

Entre los principales beneficios en la integración de SDN en las redes inalámbricas se encuentran: optimización de recursos, convergencia de redes heterogéneas, control programable, innovación en las funcionalidades de la red, entre otras [57]. La flexibilidad que ofrece SDWN puede mejorar el rendimiento general y la seguridad de las redes inalámbricas, debido a la configuración de reglas para asignar diferentes niveles de privilegios a distintos tipos de usuarios [17].

[4]Para este contexto se considera el medio por el que las estaciones celulares se conectan al centro de conmutación del operador móvil.

En general, SDN en las redes de acceso inalámbrico se puede considerar como una extensión de la red cableada [58]. Esta extensión se denomina a veces SDWN [58] [56], por SDN inalámbrica, la cual permite una coordinación centralizada para la asignación de recursos en una WLAN, programar la red para facilitar el uso y el rendimiento de la misma.

La consolidación de servicios o funciones inalámbricas en un software controlador lógicamente centralizado es posible gracias a los principios de arquitectura SDN. En la literatura se encuentran algunas propuestas de plataformas que utilizan el paradigma SDN en redes WLAN, tales como: Odin [18], OpenRoads [59], CloudMAC [23], EmPOWER [27], Mininet-WiFi [26], SDWMN [60] [61], etc.

2.2. Estado del Arte

2.2.1. Mecanismos tradicionales de QoS en redes WLAN

La capa MAC en el entorno inalámbrico es de particular relevancia ya que es la responsable de compartir el medio y asegurar la comunicación. Además, las WLAN presentan desafíos adicionales, como la diferenciación de servicios, retrasos mínimos, asignación equitativa del ancho de banda y la garantía de requerimientos de desempeño para aplicaciones de tiempo real. Por las razones anteriormente descritas, algunos autores centran sus trabajos en mecanismos de QoS que mitiguen esos desafios como: agregación de tramas, planificación, priorización y control de admisión sobre la capa MAC [4].

2.2.1.1. Mecanismos de agregación de tramas

Homann et al. [62] proponen una mejora en la capa MAC con un esquema de agregación en el estándar IEEE 802.11n, donde se envían dos o más tramas simultáneamente en una misma transmisión en EDCA para soportar requerimientos estrictos de QoS de algunas aplicaciones. Los autores demuestran que el rendimiento y el retardo pueden ser mejorados cuando se aplican prioridades estrictas, las colisiones se reducen a cero y el tiempo de espera en cola puede minimizarse para ciertos escenarios típicos. Además, Kuo et al. [63] presentan un diseño de alto desempeño para la capa MAC que soporta requerimientos de QoS combinando HCF de 802.11e con el esquema de agregación de tramas de 802.11n. Su propuesta incluye mecanismos para soportar QoS como: control de admisión, cálculos asignados para oportunidades de transmisión (TXOP, período de tiempo cuando una estación tiene el derecho de iniciar las transmisiones en el medio inalámbrico) y un planificador que ayuda a mejorar el desempeño del tráfico *best effort*

mientras los mecanismos de calidad del servicio soportan tráfico de tiempo real. Se mejora, de esta manera, la capacidad y utilización del canal y, además, se minimizan las pérdidas de paquetes.

2.2.1.2. Mecanismos de planiftcación MAC

Autores como Char et al. [28] indican que existen estudios como: planificación adaptativa de colas, donde se adaptan los valores de las ventanas de contención, las oportunidades de transmisión y la separación entre tramas; planificación CSMAC (*Centralized Scheduling based MAC*), donde se reducen las colisiones de la red en el control de acceso al medio. La planificación centralizada FQA (*Fair QoS Agent*) centrada en el método de planificación para mejorar el desempeño de QoS, garantiza una asignación justa del tráfico de tiempo real en redes inalámbricas. De igual forma, Luo y Shyu [64] proponen un mecanismo de planificación centralizada para sobrellevar deficiencias como: control descentralizado del AP, valores fijos para las oportunidades de transmisión y servicio de tasa de bits constantes del protocolo HCCA de 802.11e no adecuadas para aplicaciones multimedia con tasa de bits variables. Mediante simulación los autores demuestran que su esquema ofrece mejor calidad de servicio para tráfico de tiempo real con una asignación eficiente del ancho de banda.

2.2.1.3. Mecanismos de priorización

Se propone para el estándar IEEE 802.11e una clasificación desde las perspectivas de encolamiento con prioridad definiendo colas con diferentes niveles de prioridad para almacenar y ordenar los paquetes de los distintos tipos de tráfico. Al tráfico con más alto requerimiento de QoS se le asigna la más alta prioridad [65]. Chen y Yang [66] proponen una asignación justa proporcional sobre los recursos de red para el tráfico de menos prioridad y garantizan los requerimientos de QoS en los servicios diferenciados mediante un algoritmo que calcula la ventana de contención mínima óptima y otro de control adaptativo que calcula las probabilidades de colisión. Demuestran así que el enfoque de control propuesto tiene una alta velocidad de convergencia y precisión.

2.2.1.4. Mecanismos de control de admisión

Xiao y Li [67] proponen una medida basada en control de admisión para tráfico *best effort* que garantiza QoS en transmisiones de tiempo real en IEEE 802.11e. El control de admisión distribuido, asistido de manera centralizada, proporciona buena diferenciación entre categorías de acceso, además permite equidad entre el tráfico de tiempo

real y otras ACs. Para la transmisión, los APs controlan dinámicamente los parámetros globales basados en las condiciones del tráfico. Estos parámetros se utilizan para guiar las estaciones en el acceso al canal usando CSMA/CA de una manera distribuida. Los autores afirman que con el esquema propuesto, la QoS es mejorada ya que no se permite la degradación mediante el control del tráfico pesado.

Taher et al. [8] proponen un modelo analítico preciso para predecir las métricas de calidad de servicio en futuros flujos. Para esto se utiliza un mecanismo de control de admisión que permite mantener la red lejos de un estado de saturación mediante baja carga computacional. Es decir, el mecanismo considera los efectos producidos por los flujos que ingresan por nuevas conexiones. Además, tiene en cuenta los niveles máximos de rendimiento y retardo que toleraría la red para el estándar IEEE 802.11e. De esta manera se evita que la red caiga en el estado de saturación que afecta los valores de los parámetros de QoS. El modelo utiliza cadenas de Markov para predecir el rendimiento y el retardo antes de tomar las decisiones de admisión. Esas métricas, a su vez, están basadas en requerimientos de QoS del flujo transmitido. Los principales parámetros que considera en EDCA son: a) ventana de contención (i.e. el tiempo en el que la red está operando en el modo de contención); b) control de espacios inter-tramas (AIFS) que están basados en el período de tiempo que tiene que esperar un nodo inalámbrico antes de que se le permita transmitir su trama siguiente; y c) oportunidad de transmisión, que funciona como el intervalo de tiempo limitado durante el cual una estación puede enviar tantas tramas como sea posible.

En resumen, en la literatura existen estrategias, mecanismos o técnicas de mantenimiento de QoS para redes WLAN tradicionales. En la Tabla 2.2 se resaltan algunas de las más citadas en la literatura y con relación directa sobre la orientación de este trabajo de investigación. Allí se presentan algunos mecanismos tradicionales que se tuvieron en cuenta para la selección de una estrategia que garantice el mantenimiento de QoS en redes inalámbricas institucionales bajo el esquema de gestión SDN.

La Tabla 2.2 presenta algunos trabajos organizados de la siguiente manera: Trabajo de referencia, Metodología, las limitaciones identificadas y las métricas evaluadas. La “Metodología” identifica cada una las técnicas o estrategias usadas por los autores (p. ej. Priorización, Planificación, Control de acceso, entre otras), la descripción, los métodos, el escenario, las condiciones, herramientas con que fueron desarrollados los experimentos y los algoritmos implementados en la estrategia. La columna de “limitaciones” propone mostrar los puntos débiles del trabajo analizado, permitiendo la selección adecuada del mecanismo que mejor se ajusta al marco de este proyecto de investigación. La columna “métricas” permite identificar los parámetros de evaluación de desempeño usados,

Tabla 2.2: Mecanismos tradicionales de QoS para IEEE 802.11

Trabajo	Metodología	Limitación	Métricas
SRA-MSDU [68]	Control de Acceso Mejoramiento A-MSDU control de secuencia de tramas para incrementar el rendimiento y el retardo. Agrega un campo de control a la cabecera. Chequea la integridad de las subtramas. Hace retransmisión selectiva. Simulación en NS-2.	- No está enfocado a ofrecer QoS y se limita a maximizar la transmisión de paquetes. - No es específtca para tráftco multimedia en el contexto de este trabajo.	- Rendimiento - Retardo
Modelo analítico de Admisión. [8]	Control de Admisión Modelo analítico preciso para predecir las métricas de QoS para futuros flujos. Algoritmo que para cada AC resuelve modelo matemático para predecir rendimiento y retardo. Basados en Cadenas de Markov. Simulación en NS-2 y Matlab.	- Estaciones con una sola AC. - Limitación de evaluación del modelo. Escenario considerado no tiene en cuenta las condiciones de canal de una red WLAN.	- Rendimiento - Retardo
Equidad en WLAN [66]	Priorización Enfoque teórico de control para asignar equidad proporcional de tráftco en IEEE 802.11e. Utilizan un método de control *feedback* multivariable para asignar un control adaptativo centralizado para la asignación justa a cada ACs. Simulación en Matlab	- Un solo AP con n clientes en su modelo de red. - Asumen un canal libre de errores. - Cada cliente utiliza una sola AC.	- Rendimiento - Retardo
Planiftcador [69]	Planiftcación Algoritmo de planiftcación de paquetes para AP en ambientes concurridos. Balance mediante el ajuste automático de CW sin dejar a los paquetes de baja prioridad sin una asignación justa de recursos. *Testbed*	- Arquitectura simplift-cada de un solo AP. - No orientado a tráftco multimedia.	- Rendimiento - Eftciencia
OQAM [70]	Multicapa OQAM Optimiza el desempeño de la red inalámbrica para la transmisión de video digital en tiempo real. Algoritmos de compresión de video en la capa 7. Capa MAC utiliza el algoritmo IMAL. Control de error en la capa física. Simulación en NS-2 y EvalVid.	- Cadenas de Markov bidimensional (consideran solo dos estados). - Un solo AP con n clientes .	- Rendimiento - Retardo - Pérdida de paquetes

además, posibilita la identificación de las principales métricas que se usan en la literatura para evaluar QoS.

Por otra parte, el estudio de los mecanismos tradicionales que ofrecen garant´ıas de QoS -abordado en secciones anteriores- permitió identificar las métricas más relevantes para medir el desempeño de la red inalámbrica. También se reveló la importancia de considerar la replicabilidad de los modelos matemáticos propuestos y la fundamentación teórica que facilite su comprensión.

2.2.2. QoS en redes cableadas gestionadas con SDN

Karakus y Durresi [71] presentan una compilación de los principales mecanismos de QoS para redes cableadas en arquitecturas SDN, clasificándolas en siete tipos de mecanismos:

Enrutamiento de flujos multimedia, Enrutamiento de QoS inter-dominio, Reservación de recursos, Planificación y gestión de colas, Monitoreo de redes, QoE-aware (Conscientes de QoE) y Mecanismos relacionados con QoS. En la Tabla 2.3 se identificaron los trabajos relacionados con tráfico multimedia que manejan los mecanismos de QoS en redes gestionadas mediante SDN, como HiQoS [72], OpenQoS [73], FlowQoS [74], entre otros.

Los trabajos indicados en la Tabla 2.3 dan evidencia de la orientación del manejo de la calidad de servicio para redes cableadas que se gestionan bajo el paradigma SDN, resaltando que las métricas con mayor frecuencia identificada son la de ancho de banda (BW), rendimiento y retardo requeridos sobre los flujos multimedia. También se destaca la tendencia en la implementación de mecanismos de planificación para que las rutas seleccionadas, dependiendo del tipo de tráfico solicitado por las aplicaciones, maximicen el desempeño mediante la reservación de recursos.

2.2.3. QoS en redes inalámbricas gestionadas con SDN

Como expresan Costanzo et al. [79] y Bernardos et al. [56], hay trabajos que aprovechan los principios operativos de SDN para mejorar las estrategias de QoS en redes inalámbricas, proporcionando de esta manera los beneficios de un aprovisionamiento ágil, dinámico y flexible en la gestión del tráfico. En la Tabla 2.4 se presentan algunos trabajos que sobresalen en la literatura, relacionados con la gestión de SDN orientada a las redes WLAN. Esta tabla muestra una descripción, con su respectiva metodología y escenario usado. Resalta algunas limitaciones identificadas y muestra las métricas de desempeño.

Por su parte, Dely [80] también propone una arquitectura de gestión de redes inalámbricas basadas en SDN. El propósito es usar OpenFlow para monitorizar el flujo de tráfico y permitir a los usuarios controlar la red a través de una interfaz gráfica. El autor se refiere explícitamente a la virtualización de la red y sugiere dividirla en función de las características de la capa IP y de Aplicación. Por ejemplo, un servicio de *streaming* de video puede crear su propio segmento de red con configuraciones específicas de QoS. El autor desarrolló un método para estimar el ancho de banda disponible de un enlace inalámbrico. Este método a través de sondeos logra gran precisión de la estimación del ancho de banda. El método propuesto utiliza el tráfico de datos normal para obtener estadísticas del canal. Mediante el uso de conexiones simultáneas a múltiples APs y un *backhaul* alámbrico habilitado para SDN, las estaciones pueden realizar entregas rápidas. Demuestran que el sistema propuesto es capaz de proporcionar mayor desempeño de la transmisión de video en tiempo real a los usuarios WLAN móviles con una calidad superior a la de los sistemas tradicionales.

Tabla 2.3: Mecanismos de QoS en SDN

Trabajo	Mecanismo, descripción y metodología	Limitación	Métrica
Civanlar [75]	Enrutamiento de flujos multimedia (Planiftcación). Una arquitectura de enrutamiento habilitada para QoS para la transmisión de video escalable. Formulación basada en programación lineal para reducir la pérdida de paquetes y mantener el retraso tolerable para la codiftcación de video escalable. Testbed con NOX.	- Centrados en el plano de control.	- Ancho de banda - Retardo - Pérdidas
HiQoS [72]	Enrutamiento de flujos multimedia (Priorización). Diseño de aplicación HiQoS para enrutamiento de múltiples rutas y mecanismos de puesta en cola. Encuentra múltiples rutas entre origen y destino, con un mecanismo que proporciona garantías de BW y gestión de colas para diferentes clases de tráftco en los conmutadores SDN. Mininet con 5 conmutadores, 2 servidores y 11 clientes. Controlador Floodlight.	- Centrado en el plano de apps.	- Rendimiento - Retardo - Ancho de Banda
OpenQoS [73]	Enrutamiento de flujos multimedia (Priorización). OpenQoS transmite video a través de redes OpenFlow con QoS. Se basa en enrutamiento QoS, donde las rutas del tráftco multimedia se optimizan dinámicamente para cumplir con las exigencias de QoS requeridas. Minimiza los efectos adversos (pérdida de paquetes y latencia) entre otros tipos de flujos. Clasiftca los flujos entrantes como flujos multimedia y flujos de datos, usando campos de encabezado de paquete. Estos flujos se enrutan dinámicamente. Los flujos de datos están sujetos al enrutamiento de *best effort*. Testbed: 3 conmutadores, 1 controlador, 3 host. Floodlight	- Limitación de evaluación: No se evalua con aplicaciones reales. - Enrutamiento de QoS basados en métodos tradicionales como LARAC	- Rendimiento - SNR - Costo
Egilmez [76]	Enrutamiento de flujos multimedia (Planiftcación) Plataforma analítica para optimizar las decisiones de reenvío en la capa de control para habilitar QoS dinámico sobre las redes OpenFlow. El controlador calcula nuevas rutas para el flujo de video resolviendo el problema de CSP, mediante optimización con el algoritmo LARAC implementado en LEMON. *Testbed*	- No garantiza algunos parámetros (rendimiento y retardo) de QoS para tráftco multimedia. - Limitado a los parámetros del modelo de optimización.	- Ancho de Banda - SNR
FlowQoS [74]	Reserva de recursos (Priorización) FlowQoS permite a los usuarios especiftcar prioridades de flujo de aplicaciones de alto nivel. Los usuarios de la red de acceso simplemente especiftcan las aplicaciones que deberían tener mayor prioridad en comparación con otras. El controlador de FlowQoS realiza la conftguración de QoS apropiada con las preferencias de un usuario. Testbed: OpenWrt con OVS, Raspberry Pi. Controlador POX.	- QoS implementado en enrutador doméstico - No compara resultados contra otros *frameworks* - Conftguración manual del traftco de control de Linux	- Rendimiento - Retardo - Ancho de banda
QoSFlow [77]	Planiftcación y gestión de colas (Planiftcación y Priorización) Una plataforma de control de QoS que usa múltiples planiftcadores de paquetes. Manipula los planiftcadores de múltiples paquetes usando FIFO. Combina los planiftcadores de paquetes de Linux junto con las redes OpenFlow. Testbed: 3 Conmutadores TPLink. Cualquier controlador.	- Centrado solo en el plano de datos.	- Rendimiento - Ancho de banda
Jarschel [78]	Consciente de QoE (Planiftcación) Cambio dinámico de los flujos de aplicaciones entre los canales disponibles para mejorar la calidad de las aplicaciones críticas. Implementación de la gestión de recursos de red para YouTube mediante API de *northbound*. Tesbed: 2 conmutadores, un servidor. Floodlight.	- Limitado optimizar el flujo de tráftco de acuerdo con la información disponible. - Dependencia al banco de pruebas SDN habilitado para YouTube.	- Rendimiento - Ancho de banda

Tabla 2.4: QoS en redes LAN inalámbricas gestionadas con SDN

Trabajo	Mecanismo, descripción y metodología	Limitación	Métrica
MAC as NFV [81]	Priorización Capa MAC como un servicio NFV en un servidor en la nube. Estrategias de priorización de paquetes mediante estrategia de cola. Basado en OpenDaylight sobre una nube privada de OpenStack.	- No se enfoca directamente sobre tráftco de tiempo real. - Enfocado en mantener un equilibrio entre tráftco de *background* y tráftco de alta prioridad.	- Rendimiento - Latencia
CloudMAC [23]	Plataforma sin mecanismos Habilita los APs para redireccionar la capa MAC, procesando los datos MAC en una infraestructura de computación en la nube. Virtualización completa de la MAC, crea AP virtuales en servidores. Máquinas virtuales. APs con OpenWRT.	- El desempeño se ve un poco afectado por la latencia y pérdida de paquetes. No enfocado en QoS.	- Rendimiento
WLANs with Odin [82]	Plataforma sin mecanismos Odin introduce la idea de los APs Virtuales Ligeros (LVAPs) en el entorno WLAN de la empresa para crear una conexión continua entre los APs. Los usuarios utilizando un único BSSID asociado con el usuario. Cada usuario está asociado con un LVAP y varios de estos LVAPs están alojados en un AP. Múltiples agentes se ejecutan en cada AP y un controlador centralizado con una vista global de la red gestiona la movilidad (sin activar la reasociación), el balanceo de carga y la interferencia. Testbed: Un solo cliente, 2 APs, un servidor para ejecutar el controlador OpenFlow. Floodlight.	- Plataforma no orientado a ofrecer QoS .	- Rendimiento
SWAN [11]	Plataforma sin mecanismos SWAN utiliza un controlador lógicamente centralizado para los AP en la WLAN del campus. Utiliza el protocolo SWAN para adquirir estadísticas e información de red y vincula dicha información con las operaciones en la API para aplicaciones de capa superior. Cada AP ejecuta un agente que puede manejar algunas tareas de control. *Testbed*: Diferentes escenarios. Servidor, 2 APs, 2 Estaciones.	- La optimización del algoritmo de decisión de *handover* inteligente y balanceo de carga eftciente. - Las aplicaciones de red para mejorar la gestión y conftguración de la WLAN. - No está orientado a garantizar QoS en las aplicaciones.	- Ninguna
QoSAC [83]	Asociación Investiga el problema de la asociación a nivel de flujo para abordar las demandas de calidad de servicio de los clientes. Propone un mecanismo para la asociación concurrente de un cliente a múltiples APs y soporta enrutamiento de tráftco a nivel de flujo. Este mecanismo facilita la gestión específta de cada flujo. Por ejemplo, un cliente puede utilizar un AP para la transmisión de video mientras utiliza otro AP para cargar un archivo. Testbed: Varios escenarios con 10 APs y 30 clientes. Controlador no especiftcado.	- No considera conjuntamente los problemas de asignación de canales y asociación de clientes en la red para mejorar la eftciencia de la red.	- Retardo - Rendimiento
Planiftcador de AP [84]	Planiftcación Mejorar el mecanismo DCF para asegurar que cualquier transmisión de paquetes en el enlace de descarga tenga la tasa de éxito más alta posible. Una controlador central toma el control de todos los APs a través de OpenFlow. Mediante la instalación de reglas el controlador puede realizar una programación detallada de los paquetes en el enlace de descarga de los APs. El algoritmo obtiene una alta tasa de recepción de paquetes a ftn de mejorar la eftciencia del DCF. Simulación: 10 Aps, 50 Estaciones. No especiftcado, cualquier controlador.	- Centrado solo en el enlace de descarga para realizar la planiftcación del tráftco.	- Rendimiento - Retardo
Empower [27]	Plataforma multipropósito Retoma las ideas de Odin y construye su propia plataforma para agregar nuevas funcionalidades como: gestión de recursos de red, energía, interferencia, *handover*, entre otros. Tesbed: 3 APs, un servidor, 3 estaciones. Controlador POX.	- Su operación no está orientada a ofrecer QoS de manera explicita - Requiere que los APs soporten OpenWRT par su operación. - Solo opera con controladora inalámbrica Atheros.	- Rendimiento - Ancho de banda

Zhao et al. [85] proponen un controlador SDN que gestione todos los APs a través de la interfaz OpenFlow. El esquema propuesto utiliza OpenFlow para mitigar la interferencia de las redes WLAN empresariales entre APs. El *framework* planteado añade normas específicas en varios puntos de acceso para la planificación de paquetes, sin modificar el mecanismo DCF convencional. La idea básica de esta propuesta es mantener una competencia moderada entre los AP para lograr una alta tasa de éxito de transmisión, evitando retransmisiones excesivas.

En otra investigación Lee et al. [86] amplían la red del controlador SDN a dispositivos móviles, logrando la detección en tiempo real de las demandas de QoS de una red inalámbrica y proporcionando control de QoS de extremo a extremo.

Por otra parte, con la revisión de estas últimas secciones, en el Estado del arte se evidencia que existe una multitud de trabajos orientados a garantizar QoS sobre arquitecturas SDN cableadas. El despliegue en redes inalámbricas WLAN institucionales con esta misma arquitectura es un poco más limitado. Se pudo evidenciar que sí existen plataformas de trabajo (*frameworks*) para gestionar las redes IEEE 802.11 bajo el paradigma SDN, pero sin una orientación particular a brindar mecanismos explícitos de QoS en los casos estudiados. Solo investigaciones como las presentadas por Zhao et al. [84] muestran evidencias del uso de mecanismos tradicionales como el de planificación, que ofrece garant´ıas de QoS implementadas en plataformas gestionadas con SDN.

Para concluir este capítulo, se pudo identificar la orientación de las investigaciones dirigidas a ofrecer mecanismos que garantizan QoS sobre redes inalámbricas que operan con el estándar IEEE 802.11 y redes cableadas e inalámbricas gestionadas con SDN. De las investigaciones enfocadas en redes tradicionales es escogida la de Taher [3] [8], ya que ofrece una adecuada formulación matemática de su modelo, un buen soporte argumental basado en las publicaciones y un mecanismo de referencia replicable para ofrecer garantías de QoS. El modelo propuesto está orientado al tráfico multimedia y es adaptable a las condiciones que ofrecen las redes inalámbricas gestionadas con SDN sin tener que incurrir en la reconfiguración de la capa MAC por defecto en un AP. Dentro de las caracter´ısticas del proyecto, el modelo se ajusta a los requerimientos para su implementación como mecanismo que garantizará el mantenimiento de QoS en la transmisión de video en tiempo real sobre redes WLAN, bajo el esquema de gestión SDN.

En el siguiente capítulo se presentarán los detalles de la formulación del modelo analítico de rendimiento en redes IEEE 802.11e (EDCA) que garantizará los límites máximos en que debe operar una red antes de caer en condiciones de saturación, requeridos para la toma de decisiones de un algoritmo de control de admisión.

Capítulo 3

Modelo analítico de rendimiento en redes IEEE 802.11e (EDCA)

3.1. Introducción

Luego de la revisión detallada de la literatura en el capítulo anterior, se seleccionó el mecanismo de control de admisión que ofrece garantías de QoS para tráfico de video en entornos de redes inalámbricas tradicionales. Las decisiones del algoritmo de control de admisión están soportadas en la predicción del rendimiento bajo condiciones de saturación obtenidas a partir de un modelo analítico representado por cadenas de Markov.

Este modelo describe los detalles de operación para EDCA en el estándar IEEE 802.11e, las transiciones entre estados que ocurren en una AC en cada período de tiempo y la formulación del modelo de rendimiento propuesto por Taher [3]. También se presentan los detalles en la implementación del algoritmo que calcula el rendimiento mediante un *solver* de Matlab [87] para obtener la solución al sistema de ecuaciones no lineales propuesto, a través de las probabilidades de ocupación de canal, de colisión y de transmisión exitosa. A partir de estos últimos valores se obtienen los límites máximos en los que debe operar una red antes de caer en condiciones de saturación.

El "rendimiento de saturación" o rendimiento bajo condiciones de saturación es una cifra de desempeño fundamental que representa la carga máxima (en bps) que el sistema puede soportar en condiciones estables [88]. Este se define como el límite alcanzado por el rendimiento del sistema a medida que la carga ofrecida aumenta.

El modelo de Bianchi [88] fue el primero en dar una formulación matemática para representar el DCF en IEEE 802.11 [89], siempre que una estación esté bajo condición de

saturación. Este modelo bidimensional, basado en cadenas de Markov, abstrae el proceso de *backoff* y predice el comportamiento del mecanismo DCF. El propósito de este modelo es estimar el rendimiento del protocolo IEEE 802.11 en estado de saturación al aumentar el número de estaciones activas en la red.

Por otra parte, IEEE 802.11e EDCA proporciona un mecanismo -basado en contención- para soportar calidad de servicio (QoS), mejorando así el protocolo DCF. En EDCA se definen cuatro categorías de acceso parametrizadas para diferenciar los servicios. Cada AC se comporta como un DCF y tiene su propia cola de transmisión y cuatro parámetros de contención ajustable: ventanas de contención mínima y máxima ($CWmin$ y $CWmax$), espaciado inter tramas arbitrario (AIFS) y longitudes de ráfagas de paquetes o límite de oportunidad de transmisión ($TXOP\ Limit$). Una adecuada sincronización conjunta de los cuatro parámetros produce la diferenciación de servicios al otorgar distintas prioridades de acceso a los diferentes tráficos. Los parámetros de contención proporcionan un mecanismo flexible para asignar estas prioridades a los canales, sin embargo, es muy difícil predecir el rendimiento de cualquier configuración de parámetros elegidos [90].

3.2. Modelo analítico de EDCA

3.2.1. Mecanismo de acceso MAC de EDCA en IEEE 802.11e

Como se había mencionado en el capítulo anterior, EDCA define cuatro categorías de acceso (AC): voz, video, *best effort* y *background* ; y tres parámetros de priorización del tráfico que son: AIFS, ventana de contención (CW) y $TxOP\ Limit$.

En EDCA, cada AC dentro de una estación particular se comporta como una estación virtual que debe escuchar el canal antes de iniciar su transmisión. Si el canal se detecta libre por un período de tiempo igual a su AIFS específico (el AIFS particular de la AC), comienza su procedimiento de *backoff* eligiendo un valor aleatorio de su ventana de contención y comienza a disminuir el contador de *backoff*. En cada intervalo de tiempo durante el procedimiento de *backoff*, si la AC detecta una actividad en el canal, detiene su contador de *backoff* y espera hasta que el canal se libere de nuevo durante un período completo de AIFS antes de disminuir su contador de *backoff*. Una vez que el contador de *backoff* llega a cero, la AC transmite en el canal.

La AC continúa transmitiendo hasta que su límite de transmisión $TXOP\ Limit$ expira mediante el intercambio de secuencias de datos y acuse de recibo (ACK) con su destino, separado por SIFS (*Short Interframe Space*). Los ACK se utilizan para notificar a la

estación fuente acerca de una buena recepción de datos. Si la estación no recibe un acuse de recibo para una trama determinada, considera que hay una colisión o un error en el canal. En este caso, espera un período AIFS antes de iniciar la retransmisión de esta trama. Las demás estaciones que no están implicadas en la colisión, retrasan su transmisión por un período de tiempo igual a EIFS - DIFS + AIFS. Este período de tiempo se llama período posterior a la colisión (*post-colisión*).

En DCF, las estaciones esperan un tiempo igual a EIFS (*Extended Inter Frame Space*). Por definición del estándar, el período de espera es mayor que el AIFS y el DIFS (*Distributed Inter Frame Space*). Debido a que es necesario dar a la estación fuente la prioridad de retransmitir su trama lo antes posible. Cuando la colisión ocurre entre diferentes ACs de la misma estación, la AC de mayor prioridad obtiene una transmisión física en el canal.

Después de cada transmisión fallida la ventana de contención se duplica hasta que se alcanza la *CW max*. El valor inicial de CW es *CW min*. Una trama de datos es rechazada después de un número máximo de reintentos sin éxito (llamado *Retry Limit*, m). Después de cada transmisión exitosa, la AC realiza una copia de seguridad aleatoria, incluso si no tiene otras tramas que transmitir en su cola. Esto se llama *post-backoff*, ya que se hace después y no antes de la transmisión. Después de este procedimiento de *post-backoff*, si la AC tiene datos para transmitir, intentará acceder al canal directamente. De lo contrario, permanecerá en un estado de espera para la llegada de nuevas tramas [8]. El diagrama mostrado en la Figura 3.1 esquematiza el procedimiento del *backoff* exponencial binario para IEEE 802.11e EDCA.

Para modelar el comportamiento previamente descrito se pueden utilizar cadenas de Markov discretas de cuatro dimensiones [2]. La primera dimensión $a(t)$ indica el período ocupado por la AC en el tiempo t, la segunda dimensión $s(t)$ indica la etapa del *backoff*, la tercera dimensión $b(t)$ representa el valor del contador de *backoff* en el momento t; finalmente, la cuarta dimensión $r(t)$ indica el tiempo restante para dejar el período ocupado por la AC, tal como se explica en este cap´ıtulo.

3.2.2. Modelo analítico EDCA para rendimiento

En esta sección se presenta la formulación del modelo analítico que proporciona los valores de referencia del rendimiento bajo condiciones de saturación necesarios para la toma de decisiones en el algoritmo de control de admisión que se propondrá en el Cap´ıtulo 4.

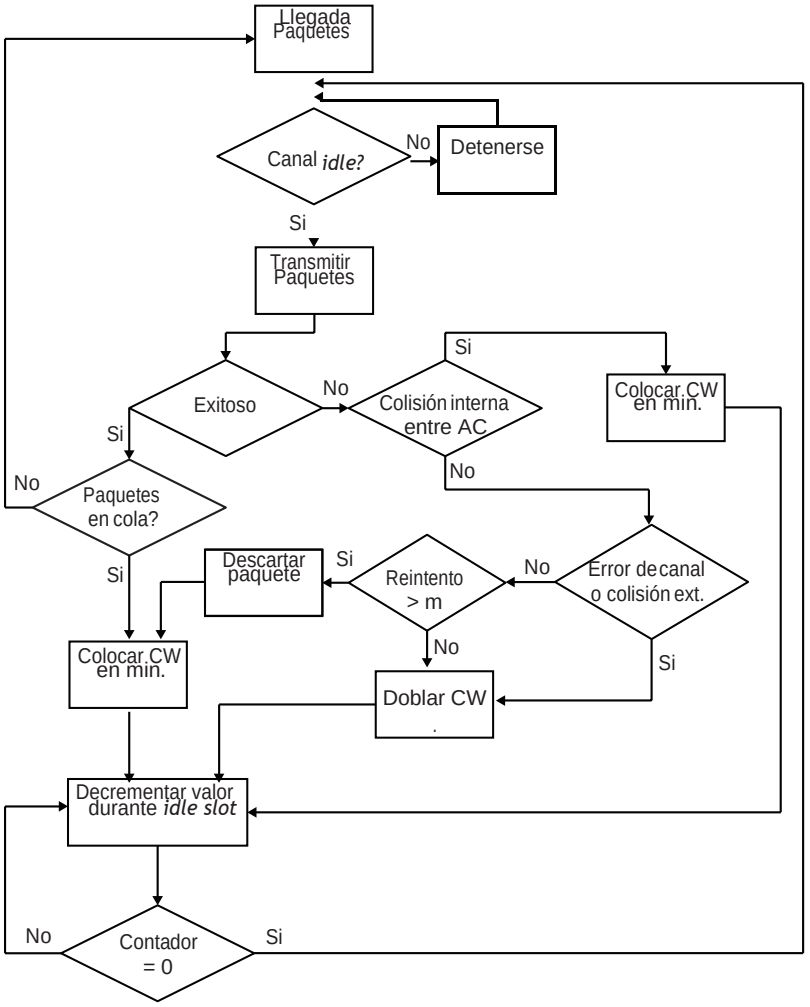

Figura 3.1: Diagrama del procedimiento del *backoff* exponencial binario para IEEE 802.11e EDCA [91].

En el modelo basado en Taher et al.[3], cada estado representa una AC en un *slot* de tiempo. Al final de cada *slot*, un evento puede activar una transición a otro estado. Los seis estados que puede ocupar una AC en cada *slot* de tiempo, seleccionado aleatoriamente, representan las operaciones de EDCA: período AIFS, período *backoff*, período *frozen*, período de colisión, período de *post-colisión* y período de transmisión.

En cada intervalo de tiempo el estado de cada AC está determinado por las variables (i, j, k, d).

(i) La primera variable, i, representa el período ($i = A$ para AIFS, F para *frozen*, B para *backoff*, C para colisión, PC para *post-colisión* y T para transmisión). Esta variable es necesaria para la comprensión de la transición entre los estados y para que sea posible integrar todas las caracter´ısticas de EDCA.

(ii) La segunda, j, representa la etapa de retroceso o el número de retransmisiones ($j = 0, 1, 2...m$).

(iii) La tercera, k, contiene el valor del contador de *backoff* y toma su valor de manera uniforme en el intervalo $[0, w_j]$. w_j depende de la etapa de *backoff* y satisface las relaciones; $w_{j+1} = 2w_j + 1$ si $w_j < w_{max}$ y $w_{j+1} = w_j$ si $w_j = w_{max}$.

(iv) La última, d, indica el tiempo restante en *slots* de tiempo para abandonar el período. d depende del valor de i ya que el tiempo restante para salir de un período y entrar en otro, dependiendo en el que se encuentre.

En cada *slot* de tiempo los estados de cada AC están determinados por:

(1) Período AIFS: este período aparece varias veces en el ciclo de transmisión. Hay un AIFS al principio de cada intento de transmisión antes del primer *backoff* y un AIFS después de cada suspensión del contador de *backoff*. En los estados del período AIFS, la AC debe detectar un canal libre durante un período completo igual a AIFS. De este modo, la tupla (i, j, k, d) es igual a (A, j, k, A), $(A, j, k, A-1)$, $(A, j, k, A-2)$ ... $(A, j, k, 1)$. A para indicar el período del AIFS donde A es igual a AIFS + 1, se añade el 1 para garantizar que se espera un período completo de AIFS y para reflejar la definición del estándar. El período AIFS siempre comienza con el estado (A, j, k, A). Si el canal se encuentra ocupado con una probabilidad P_b, la AC permanece en este estado, de lo contrario se producirá una transición a $(A, j, k, A-1)$ y en cada estado del período del AIFS, $1-P_b$ es la probabilidad de transición de (A, j, k, d) a $(A, j, k, d-1)$ y P_b es la probabilidad de transición de (A, j, k, d) a (A, j, k, A), $d = A, A-1 \ldots 1$. Si se detecta que el canal está ocupado durante el período AIFS, la AC debe esperar a que el AIFS esté completo antes de entrar en el procedimiento de *backoff*, lo que se ajusta a la definición del estándar. El modelo considera que hay una suspensión del contador *frozen* durante el período AIFS en el caso del canal ocupado [2].

(2) Período *backoff*: para $i = B$, j contiene el valor de la etapa de *backoff* y k contiene el valor actual del contador de *backoff* que disminuye con cada intervalo de tiempo. $(i, j, k, d) = (B, j, k, k)$; $j = 0 \ldots m$, $k = d = 0 \ldots w_j$. En cada estado del período de *backoff*, si una transmisión es realizada por otra estación o AC, el canal se determina ocupado y la AC en cuestión cambia a un estado de suspensión del contador de *backoff* manteniendo la misma etapa y lectura de este contador. Una vez que el contador de *backoff* llega a cero, la AC intenta transmitir. Si el resultado de este intento es una colisión, la AC cambia a este estado siempre que no se haya alcanzado el límite de retransmisión. Si no es así, borra la trama y cambia a la transmisión de las siguientes tramas. En el caso en que el intento de transmisión tenga éxito, se realiza una transición directa al estado de transmisión [2].

(3) Período *frozen*: para $(i, j, k, d) = (F, j, k, N), (F, j, k, N-1) \ldots (F, j, k, 1)$; j y k son la etapa de *backoff* y el contador suspendido respectivamente. j y k se mantienen iguales durante este período mientras que el parámetro d solo disminuye hasta que finaliza el período de suspensión del contador de *backoff* antes de volver a entrar en el período AIFS y, a continuación, en el procedimiento de *backoff*. Así que aquí d representa el tiempo de suspensión restante antes de reactivar el contador de *backoff*. Este tiempo es aleatorio, está en relación directa con el tiempo de transmisión de la AC que realmente transmite en el canal. De hecho, durante la transmisión de una AC dada, todos los contadores de *backoff* de las otras ACs están bloqueados durante un período igual al tiempo de transmisión de la presente AC. No es fácil saber qué AC está transmitiendo durante este período si se reemplaza este tiempo aleatorio por su valor medio se puede lograr una forma más sencilla de tener un valor representativo de esta variable sin afectar significativamente la validez de la variable modelo. N, representa el tiempo medio de suspensión [2].

(4) Período de colisión: con $(i, j, k, d) = (C, j, 0, Tc), (C, j, 0, Tc-1) \ldots (C, j, 0, 1)$; aquí $k = 0$, d inicia en Tc, j es la etapa de *backoff* donde ocurrió la colisión y Tc es el tiempo promedio que tarda en detectar una colisión. Se calculará de la siguiente manera para los modos de acceso básico y RTS/CTS: después de la colisión, la AC detecta un error en el nivel de datos recibidos e introduce directamente un estado de *post-colisión* para incrementar su etapa de *backoff* e intentar una nueva retransmisión. La Figura 3.2 muestra las transiciones en estados de colisión para $0 \leq j \leq m-1$ [2].

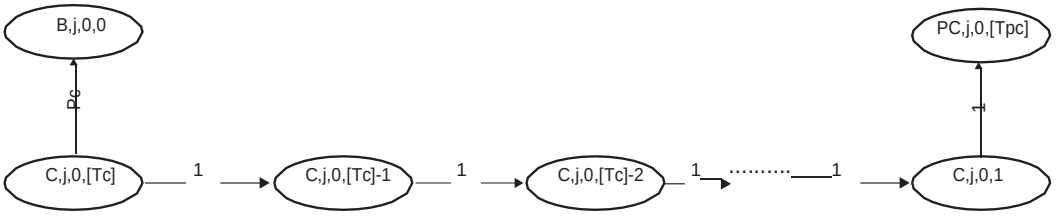

Figura 3.2: Estados de colisión y sus transiciones ($0 \leq j \leq m-1$)[2].

(5) Período de *post-colisión*: con $(i, j, k, d) = (PC, j, 0, Tpc), (PC, j, 0, Tpc-1) \ldots (PC, j, 0, 1)$. PC indica el período de *post-colisión* y Tpc es el tiempo de *post-colisión*, es igual a AIFS si la AC está involucrada en la colisión y si no lo está será EIFS - DIFS + AIFS. Debido a esta consideración, las estaciones involucradas en la colisión tienen la oportunidad de retransmitir lo antes posible. El estado posterior a la colisión es muy similar al Período AIFS, con una diferencia en los tiempos de

espera. j es la etapa de *backoff* donde se produjo la colisión, después de la colisión la etapa de *backoff* se incrementa en 1 para $0 \leq j \leq m-1$ [2].

(6) Período de transmisión: con $(i, j, k, d) = (T, 0, 0, Ts)$, $(T, 0, 0, Ts-1)$ ... $(T, 0, 0, 1)$, $j = k = 0$ (Ts es el tiempo de transmisión promedio), porque el *backoff* no interviene una vez que la AC entra al período de transmisión, $d = Ts...1$. Aquí la AC realiza una transmisión exitosa. Una vez que comienza la transmisión sobre el canal, ésta continúa hasta alcanzar el $TXOP\ Limit$ (bajo la premisa de que la AC tiene suficientes tramas en su cola). [2].

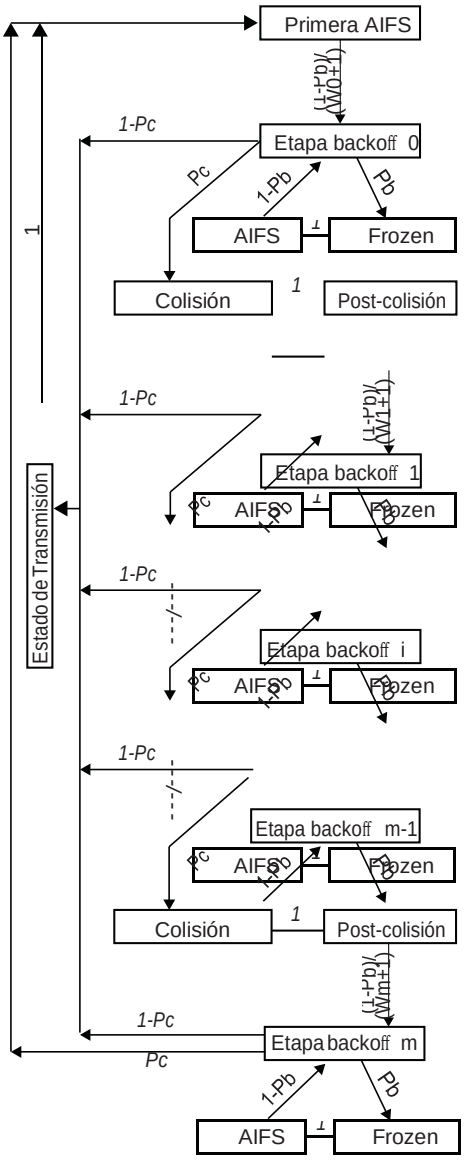

Figura 3.3: Diagrama de bloques de transiciones entre estados [2].

Teniendo todos los estados posibles para una AC con sus transiciones, en la Figura 3.3 se presenta un diagrama de bloques de la cadena de Markov donde se evidencian las

transiciones entre los períodos. El esquema completo de la cadena de Markov se muestra en la Figura 3.4.

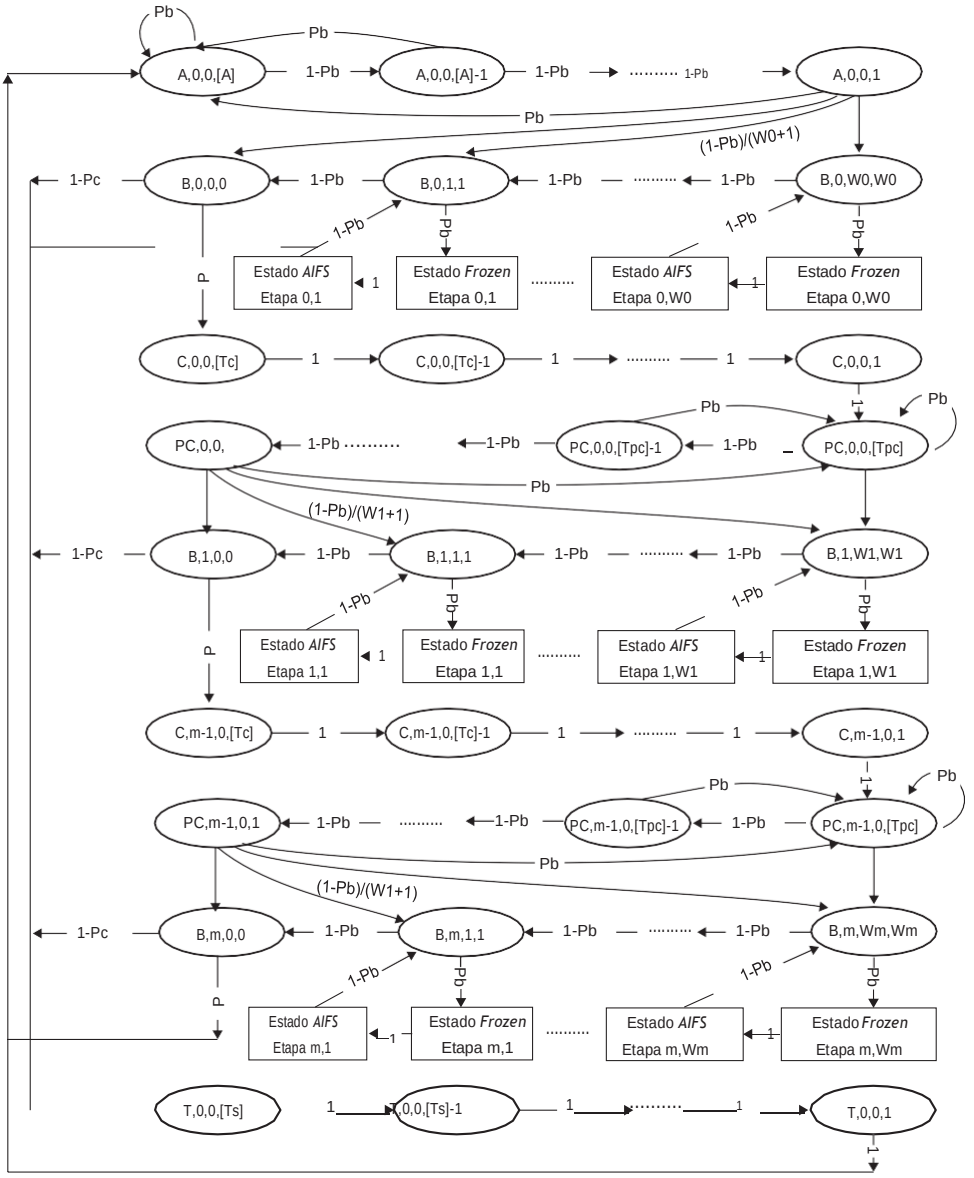

Figura 3.4: Esquema completo de la cadena de Markov bajo condiciones de saturación [8].

3.2.3. Consideraciones para la solución del modelo

Para obtener límites máximos del rendimiento bajo condiciones de saturación se procede como indica la Figura 3.5. Primero, se calculan las probabilidades de estado estable en términos de las probabilidades inciales de backoff $P_{B,0,0,0}$ aplicando la condición de normalización. Luego, se utilizan los parámetros del estándar IEEE 802.11e para los requerimientos temporales de la cadena de Markov, y se procede a resolver el sistema

de ecuaciones no lineales propuesto por Taher, para determinar el conjunto de probabilidades de canal ocupado, de canal libre, y de transmisión exitosa.

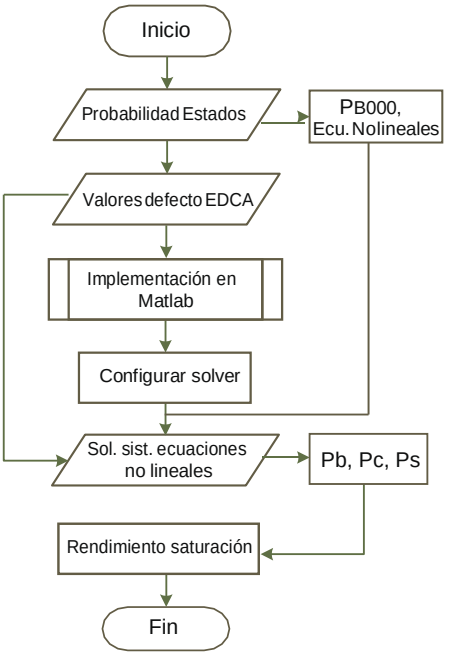

Figura 3.5: Procedimiento solución del sistema.

3.2.3.1. Cálculo de las probabilidades de los estados de la cadena de Markov

Taher [2] parte de la premisa de que el valor de la probabilidad de transición $P_{B,0,0,0}$ es conocido y es la probabilidad de estar en la primera etapa de *backoff* con un contador en cero. Para calcular las probabilidades de todos los estados es necesario recurrir a las probabilidades de transición y los parámetros por defecto que suministra EDCA. Después de calcular las probabilidades de todos los estados (ver referencia [2]), se obtienen las ecuaciones mostradas en la Tabla 3.1.

Se considera $P_{i,j,k,d}$ la probabilidad estacionaria del estado (i, j, k, d). Para resolver el sistema y calcular las métricas de rendimiento se deben encontrar las probabilidades de todos los estados de la cadena de Markov. Para obtener la probabilidad de un estado dado es suficiente sumar las probabilidades de todas las transiciones que conducen al mismo. Por lo tanto, aplicamos las ecuaciones de balance global para cada estado de la cadena de Markov. Recordamos aquí que P_c es la probabilidad de colisión total vista por una AC y es la combinación entre la probabilidad de colisión externa y la probabilidad de colisión interna para esta AC, mientras que P_b es la probabilidad de que el canal esté ocupado por todas las ACs. P_b es, por lo tanto, común para todas las ACs mientras que P_c es específico para cada AC. No se incluyen detalles para cada AC al momento de simplificar

las ecuaciones. Esto significa que cuando se escriba P_c, se refiere indirectamente a P_{ci}, con $0 \leq i \leq 3$, para cada una de las cuatro ACs [2].

Tabla 3.1: Probabilidades de todos los estados [2]

Probabilidad	Ecuación	
Inicial	$P_{B,j,0,0} = P_c^j P_{B,0,0,0}; \quad 0 \leq j \leq m$	(3.1)
Backoff	$P_{B,j,k,k} = \frac{w_j - k + 1}{w_j + 1} \frac{1}{1-p_b} P_{B,j,0,0}; \quad 0 \leq j \leq m$	(3.2)
Colisión	$P_{C,j,0,d} = P_c P_{B,j,0,0}; \quad 0 \leq j \leq m-1$	(3.3)
Post-colisión	$P_{B,j,k,k} = \frac{P_c}{1-p_b} P_{B,j,0,0}; \quad 0 \leq j \leq m-1 \quad 1 \leq d \leq T_{pc}$	(3.4)
Frozen	$P_{F,j,k,d} = \frac{w_j}{w_j+1} \frac{P_b}{1-p_b} P_{B,j,0,0}; \quad 0 \leq j \leq m \quad 1 \leq k \leq w_j \quad 1 \leq d \leq N$	(3.5)
AIFS trans.	$P_{A,j,k,d} = \frac{w_j}{w_j+1} \frac{P_b}{(1-p_b)^{d+1}} P_{B,j,0,0}; \; 0 \leq j \leq m \; 1 \leq k \leq w_j \quad 1 \leq d \leq A$	(3.6)
AIFS Inicial	$P_{A,0,0,d} = \frac{1}{(1-p_b)^d} P_{B,0,0,0}; \quad 1 \leq d \leq A$	(3.7)
Transmisión	$P_{T,0,0,d} = (1 - P_c^{m+1}) P_{B,0,0,0}; \quad 1 \leq d \leq Ts$	(3.8)

Teniendo todas las probabilidades de estados (Tabla 3.1) expresadas como una función de $P_{B,0,0,0}$, se requiere aplicar la condición de normalización, es decir, la sumatoria de todas las probabilidades de la cadena de Markov debe ser igual a 1.

$$\left[\sum_{j=0}^{m-1}\sum_{d=1}^{[Tpc]} P_{PC,j,0,d} + \sum_{j=0}^{m-1}\sum_{d=1}^{[Tc]} P_{Pc,j,0,d} + \sum_{j=0}^{m} P_{B,j,0,0} + \sum_{j=0}^{m}\sum_{k=1}^{w_j} P_{B,j,k,k} + \sum_{j=0}^{m}\sum_{k=1}^{w_j}\sum_{d=1}^{[N]} P_{F,j,k,d} + \sum_{d=1}^{[Ts]} P_{T,0,0,d}\right] + \sum_{j=0}^{m}\sum_{k=1}^{w_j}\sum_{d=1}^{[A]} P_{A,j,k,d} + \sum_{d=1}^{[A]} P_{A,0,0,d} = 1 \tag{3.9}$$

La resolución de las probabilidades de estado mediante el uso de las ecuaciones de la Tabla 3.1 y la ecuación 3.9 de normalización dan la siguiente expresión:

$$P_{B,0,0,0} = \left[\frac{1}{p_b}\left(\frac{(1-P_b)^{[A]}}{(1-P_b^m)} - 1\right) + (1 - P_c^{m+1})\left(\frac{1}{1-P_c}\right) + P_c\left([Tc] + \frac{1}{(1-P_c)}\right) + \frac{1}{2(1-P_b)}\left(\frac{P_b}{(1-P_b)^{[A]}}\right)\right]^{-1} \sum_{j=0}^{m} P_c^j w_j \tag{3.10}$$

La Tabla 3.2 muestra las ecuaciones propuestas por Taher [2] [3] para la resolución del sistema de ecuaciones no lineales del modelo analítico de control de admisión. Encontramos variables cómo: Longitud del preámbulo (H), Tiempo de paquete (P), Retardo de propagación (δ), entre otras.

Tabla 3.2: Sistema de ecuaciones no lineales [3]

Descripción	Ecuación	
Tiempo de transmisión	$Ts_1 = H + P + SIFS + ACK + 2\delta$	(3.11)
Tiempo de colisión	$Tc = H + P + SIFS + ACK + \delta$	(3.12)
Número de TxOP	$N_{TXOP} = \begin{cases} \left\lfloor \frac{TXOP\ Limit}{Ts_1 + SIFS} \right\rfloor & \text{si } TXOP\ Limit \neq 0, \\ 1, & \text{si } TXOP Limit = 0, \end{cases}$	(3.13)
Tiempo de transmisión por AC	$Ts_i = N_{TXOP_i}(Ts_1 + AIFS) \ \forall\ 0 \leq i \leq 3$	(3.14)
Probabilidad acceso al canal por cada AC	$\tau_i = \sum_{j=0}^{m} P_{B,j,0,0} = \frac{1-p_{ci}^{m+1}}{1-p_{ci}}, \ \forall\ 0 \leq i \leq 3$	(3.15)
Probabilidad acceso al canal por una AC	$\tau = 1 - \prod_{i=0}^{3}(1-\tau_i)$	(3.16)
Probabilidad de colisión interna	$P_{cint_i} = 1 - \prod_{j>i}(1-\tau_j)$	(3.17)
Probabilidad de colisión externa	$P_{cext_i} = 1 - (1-\tau)^{M-1}$	(3.18)
Probabilidad total de colisión	$P_{ci} = 1 - (1-\tau)^{M-1} \prod_{j>1}(1-\tau_j)$	(3.19)
Probabilidad de canal ocupado por otra AC	$v_i = ([Ts_i](1-p_{ci}^{m+1}) + [Tc_i]\frac{p_{ci}-p_{ci}^{m+1}}{1-p_{ci}} p_{cext_i})P_{B,0,0,0}$	(3.20)
Probabilidad de canal ocupado por una estación	$v = \sum_{i=0}^{3} v_i \prod_{j \neq i}(1-v_j)$	(3.21)
Probabilidad de canal ocupado	$P_b = 1 - (1-v)^M$	(3.22)

3.2.3.2. Estimación del rendimiento

Sea S_i el rendimiento normalizado del sistema de las AC_i. i indica la AC evaluada.

$$S_i = \frac{E[Carga\ útil\ transmitida\ con\ éxito\ por\ AC_i]}{E[Tiempo\ transmisión\ exitosa]} \tag{3.23}$$

La probabilidad P_{si} para que una transmisión de AC_i tenga éxito es igual a la probabilidad de que se transmita exactamente el tráfico para una AC_i en una sola estación. P_{si} depende de la probabilidad de canal ocupado por una estación (v) u otras ACs (v_j) respectivamente, de modo que:

$$P_{si} = M \sum_{d=1}^{\lceil T_s \rceil} P_{T,0,0,d}(1-v)^{M-1} \prod_{j>1}(1-v)_j \tag{3.24}$$

Ahora, si E[P] es el valor esperado del promedio de la carga útil del paquete, la cantidad media de información de carga útil transmitida con éxito es $P_{si}E[P\,]N_{T\,XOP_i}$. El intervalo de duración promedio en *slot* de tiempo entre dos transmisiones consecutivas se obtiene al considerar que con una probabilidad $(1-P_b)$ el canal estará libre y con probabilidad P_bP_{si} si este tiene una transmisión exitosa. Con probabilidad $P_b(1-P_{si})$ si tiene una colisión, por lo que la ecuación (3.23) se convierte en:

$$S^i = \frac{P_{si}E[P\,]N_{T\,XOP_i}}{(1-P_b) + P_b\sum_{j=0}^{3} P_{sj}T\,s_i + P_b(1-\sum_{j=0}^{3} P_{sj})T\,c_i} \quad 0 \le i \le 3 \tag{3.25}$$

La ecuación 3.25 dará los valores del rendimiento bajo condiciones de saturación para una AC_i calculada con las probabilidades P_b, P_c y P_{si}, que a la vez son obtenidas de la solución del sistema de ecuaciones no lineales propuesto en la Tabla 3.2.

3.2.3.3. Parámetros por defecto en IEEE 802.11e

Los parámetros del estándar 802.11e mostrados en las Tablas 3.3 y 3.4 son necesarios para resolver las ecuaciones de transición de estados $P_{B,0,0,0}$, que incluyen las probabilidades de colisión (P_c) y canal ocupado (P_b). Además, el tiempo de transmisión $T\,s_1$ y el tiempo de colisión $T\,c$ para una trama de datos dependen de las longitudes del preámbulo (H), el tiempo de paquete (P) y demás parámetros incluidos en la Tabla 3.3 suministrados por el estándar IEEE 802.11e para el modo de acceso básico sobre una arquitectura WLAN en modo infraestructura.

Tabla 3.3: Parámetros de referencia en IEEE 802.11e

Parámetro	Valores	
Longitud Preámbulo (H)	144 bits	72 μs
Tiempo de paquete (P)		800 μs
SIFS		10 μs
Retardo de propagación (δ)		64 μs
Longitud ACK (ACK)	112 bits	56 μs
Tasa básica (referencia)	2 Mbps	

3.2.3.4. Implementación para solución del sistema

Una vez que se tienen los parámetros por defecto de EDCA de las Tablas 3.3 y 3.4, se pueden encontrar las soluciones para $T\,s_1$, $T\,c$, $T\,s_i$, $N_{T\,XOP}$ de la Tabla 3.2 para obtener a su vez los valores de $T\,s$, $T\,c$, $T\,pc$, A , N, m, w_j, requeridos como parámetros de entrada para la solución del sistema de ecuaciones no lineales descritos en la tabla antes

Tabla 3.4: Parámetros por defecto en EDCA [3]

Prioridades −→	*AC -V O*	*AC -V I*	*AC -BE*	*AC -Btt*
AIFS (A)	2	2	3	7
$CW_{min}(w0)$	7	15	31	31
$CW_{max}(w)$	15	31	1023	1023
TXOPLimit	3264 μs	6016 μs	0	0
L´imite reintentos (m)	2	2	6	6
NTxOP	3	5	1	1
Tiempo frozen (N)	3264 μs	6016 μs	0	0
post-colisión (Tpc)	3	5	1	1

mencionada. Las probabilidades P_c y P_b dependerán de las estaciones activas y de las ACs en la estación.

Tabla 3.5: Variables y parámetros por defecto de EDCA para solución del sistema

Parámetro	Descripción	Valores de referencia
P	Carga útil del paquete[1]	[1382] Bytes
λ	Tasa de llegada estimada[1]	[200]
M	Número de estaciones activas[1]	[4,6,8,10,12,14,16,18]
m	Límite de reintentos	[2, 2, 6, 6]
w_j	Ventana de contención máxima	[15, 31, 1023, 1023]
w_0	Ventana de contención mínima	[7, 15, 31, 31]
W	Tiempo medio de espera	2690 μs
A	AIFSN	[2, 2, 3, 7]
N_{TXOP}	Número de oportunidades de transmisión	[3, 5, 1, 1]
Tc	Tiempo de colisión	1602 μs
Ts	Tiempo de transmisión exitosa	[3228, 5380, 1076, 1076] μs
T_{PC}	Períodos de *post-colisión* por AC	[3, 5, 1, 1]
N	Tiempo *frozen*	[3352, 5028, 1076, 1076] μs

Para calcular el conjunto de soluciones de la formulación del modelo de ecuaciones no lineales, anteriormente descrito, se implementó el Algoritmo 1 en Matlab (para replicar el experimento del autor [3]). Este algoritmo toma los parámetros de entrada de la Tabla 3.5 y ejecuta el *solver* [87] con los parámetros de la Tabla 3.6, la cual determina las raíces de una ecuación no lineal simultánea.

Algoritmo 1: Medida del rendimiento

Input : *Default EDCA parameters* // Tabla 3.5

$M \leftarrow [2, 4, 6, 8, 10, 12, 14, 16, 18]$ // Número estaciones activas

for k in M do // Repetir de 2 hasta 18

  $[P_{bk}, P_{ck}, P_{sk}] = solver(M, Input, equations)$ // Resolver sistema ecuaciones

  $S_{ki} = Throughput(P_{bk}, P_{ck}, P_{sk})$ // Obtener rendimiento con 3.25

  $S_{ktotal} \leftarrow \sum_{j=0}^{3} S_{kj}$ // Sumar S_{ki}, i para cada AC

end for

Tabla 3.6: Parámetros configurados en el *solver*

Parámetro	Valor	Observación
FunctionTolerance	1e-6	Criterio de parada
FunValCheck	'on'	Verificar si los valores objetivos de la función son válidos
MaxFunctionEvaluations	'default'	Número máximo de evaluaciones de funciones permitidas
MaxIterations	1e8	Número máximo de iteraciones permitidas
PlotFcn	'optimplotx'	Traza varias medidas de progreso mientras el algoritmo se ejecuta

3.3. Resultados del modelo de saturación

Después de implementar y resolver el sistema de ecuaciones no lineales (Tabla 3.2) del modelo de rendimiento bajo condiciones de saturación, se encuentran los valores de referencia necesarios para la toma de decisiones del algoritmo de control de admisión que se propondrá en el siguiente capítulo.

Los resultados de las Figuras 3.6 y 3.7 indican el comportamiento de las probabilidades de colisión y canal ocupado a medida que se aumenta el número de estaciones activas. La Figura 3.6 tiene un comportamiento creciente que se extiende más allá de 18 estaciones activas, luego tiene un comportamiento asintótico en Y = 0.4860 para más de 18 estaciones. En la Figura 3.7 también se tiene un comportamiento asintótico en Y = 0.4970 para más de 18 estaciones. El *solver* usado encuentra un conjunto de soluciones de probabilidades P_c y P_b para el conjunto de parámetros de entrada suministrados.

La Figura 3.8 muestra el rendimiento bajo condiciones de saturación para cada una de las cuatro categor´ıas de acceso: voz (*AC V O*), video (*AC V I*), *best effort* (*AC_BE*) y *background* (*AC Btt*). Los valores obtenidos por las cuatro categor´ıas de acceso indican el valor máximo de rendimiento que soportaría la red IEEE 802.11 para operar por debajo del estado de saturación. También se identifica cómo las ACs de alta prioridad

[1] Son los valores que no dependen explicitamente de EDCA, se definen para el tipo de tráfico que se usará en el experimento del siguiente capítulo.

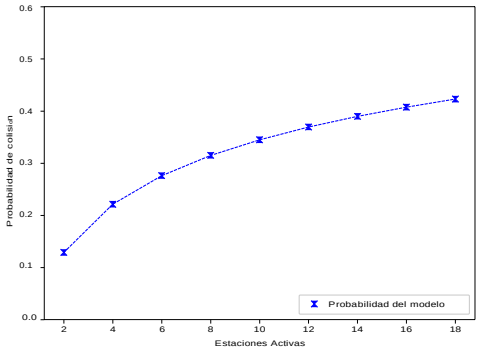

Figura 3.6: Probabilidad de colisión.

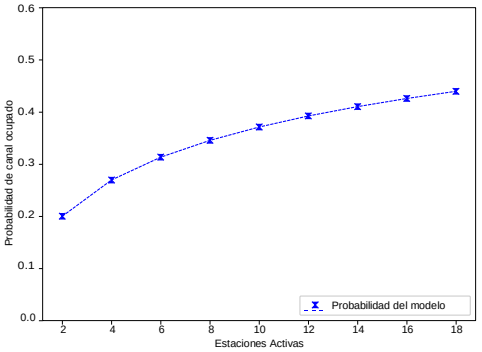

Figura 3.7: Probab. canal ocupado.

(voz y video) tendrán mayor límite de los valores de rendimiento bajo condiciones de saturación.

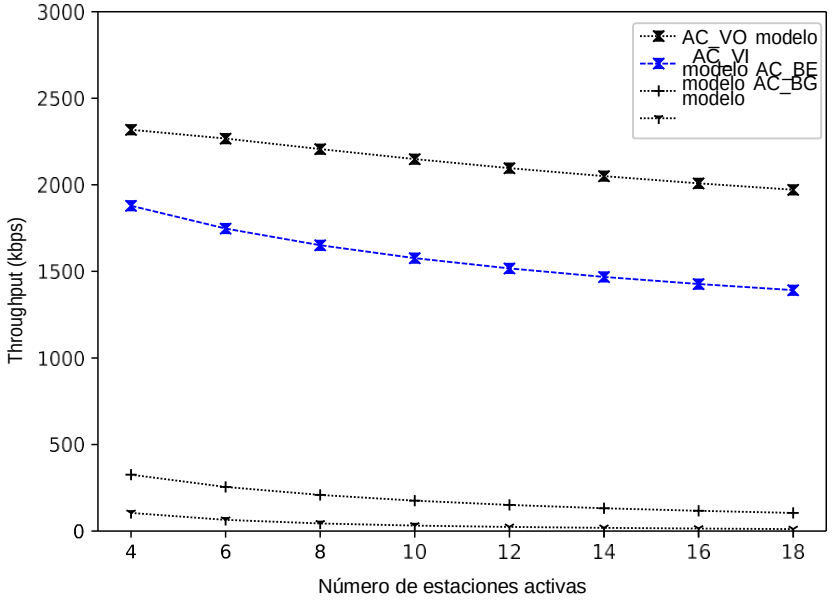

Figura 3.8: Rendimiento bajo condiciones de saturación por AC.

La Figura 3.9 muestra la sumatoria de los valores l´ımites de rendimiento bajo condiciones de saturación de todas las ACs para cada número de estaciones activas evaluadas. Cuando el número de estaciones aumenta, el rendimiento total alcanzable en la red disminuye aproximadamente hasta los 3.5 Mbps con 18 estaciones activas. Cuando más estaciones activas comparten los mismos recursos se genera más contención, la probabilidad de colisión aumenta y la capacidad de la red se degrada.

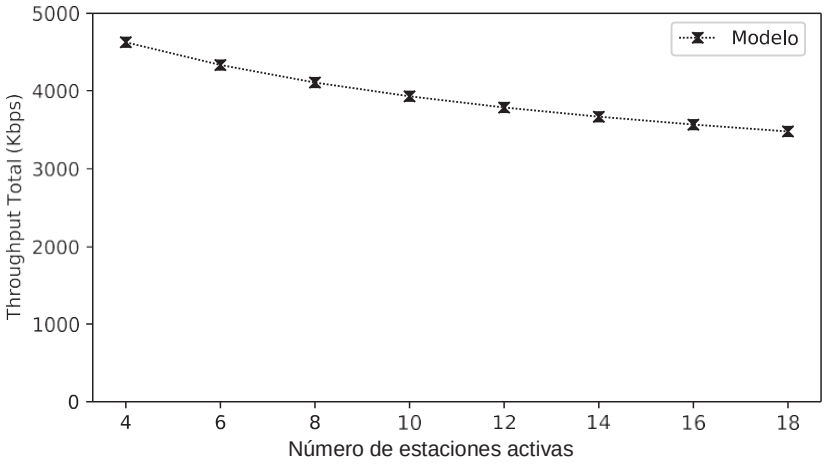

Figura 3.9: Rendimiento total bajo condiciones de saturación.

Tabla 3.7: Rendimiento bajo condiciones de saturación requerido

# Estaciones	Rendim. Sat. AC_VI (Kbps)	Rendim. Sat. Total (Kbps)
2	2056,04	4962,03
4	1878,75	4626,06
6	1746,85	4332,83
8	1650,27	4107,69
10	1575,95	3930,41
12	1516,46	3786,59
14	1467,43	3666,96
16	1426,09	3565,44
18	1390,61	3477,85

La Tabla 3.7 muestra los valores de rendimiento proporcionados por el modelo anal´ıtico evaluado. Los valores indicados aquí son los valores teóricos de rendimiento que soportaría la red antes de alcanzar la condición de saturación.

Este cap´ıtulo estuvo centrado en el desarrollo del modelo anal´ıtico propuesto por Taher para el cálculo de los valores de rendimiento bajo condiciones de saturación sobre el estándar IEEE 802.11e. Se partió de la comprensión del modo de operación de EDCA, se representó la formulación matemática de los estados de transición obtenidos a partir de las cadenas de Markov. Se usaron los parámetros por defecto de EDCA, como variables de entrada para la resolución del modelo. Se obtuvieron las probabilidades de colisión, transmisión y canal ocupado. Estos últimos valores son las variables principales para calcular el rendimiento bajo condiciones de saturación requeridas, para el futuro algoritmo de control de admisión que se propondrá en el siguiente capítulo, para una red inalámbrica gestionada con SDN, sobre la que se transmitirá video en tiempo real.

Capítulo 4

Implementación y evaluación de estrategias que garantizan QoS en la transmisión de video en tiempo real en redes WLAN basadas en SDN

En el capítulo anterior se formuló un modelo analítico para IEEE 802.11e EDCA. Tal modelo generó los valores de rendimiento bajo condiciones de saturación de una red WLAN para la AC de video como se indicó en la Tabla 3.7. Los valores límite obtenidos son la base para la toma de decisiones del mecanismo de control de admisión para QoS, como se evidenciará en este capítulo. Aquí se dan los detalles de implementación del control de admisión como estrategia que garantiza QoS del tráfico de video en tiempo real sobre un escenario experimental de redes WLAN basadas en SDN. Antes se simulará el escenario de red inalámbrica en NS-3 para validar el comportamiento de los flujos de video respecto a las otras categorías de acceso, sobre el estándar IEEE 802.11e. Luego, el entorno de red WLAN tradicional en NS-3 se reconfigurará a un esquema WLAN basado en SDN. Una vez implementada la arquitectura inalámbrica basada en SDN se ejecutaran dos aplicaciones que ofrecen mantenimiento de QoS sobre la transmisión de tráfico de video en tiempo real. A partir del uso de tráfico de tiempo real el entorno será considerdo emulación. Después, al variar el número de estaciones inalámbricas activas, se evaluará el comportamiento del rendimiento y retardo sobre los escenarios propuestos. La Tabla 4.1 resume de manera general las características que se van a evaluar (métricas) en cada uno de los escenarios que se configurarán.

Tabla 4.1: Escenarios implementados y evaluados

Escenario	Descripción	Factores	Métricas
WLAN+EDCA	Escenario de simulación Diseñado para veriftcar el comportamiento de cada AC de acuerdo a parámetros por defecto de EDCA. Simulación solo con NS-3.	Tasa de llegada (λ). ACs (AC_VO, AC_VI, AC_BE, AC_BG).	Rendimiento Retardo
WLAN+SDN+QoS	Escenario de emulación. Cada uno con un número de estaciones predeftnidas. Se crean escenarios WLAN con topología estrella. Cada escenario estará conftgurado con un número de estaciones activas, una estrategia de QoS y un video. SDN no se ejecuta cuando no hay estrategia de QoS. Emulación con tráftco de tiempo real con NS-3 y NNS adaptado para soportar SDN y aplicaciones de transmisión y reproducción de video.	Número de estaciones activas. Estrategia de QoS	Rendimiento Retardo

4.1. Operación de la arquitectura IEEE 802.11e EDCA en NS-3

Para simular el escenario en las capas de red inalámbrica PHY y MAC se utiliza NS-3 [92], configurado con las características por defecto del modelo de propagación de canal para el estándar IEEE 802.11e. Los bloques del escenario propuesto se pueden observar en la Figura 4.1. Esta simulación nos permite validar el comportamiento del rendimiento y retardo que ofrece EDCA por defecto a los diferentes tipos de flujo, según su categoría de acceso.

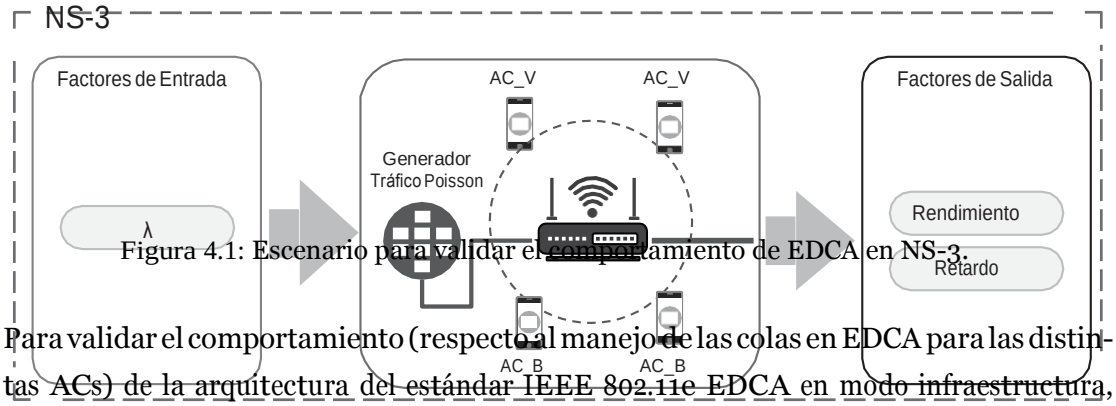

Figura 4.1: Escenario para validar el comportamiento de EDCA en NS-3.

Para validar el comportamiento (respecto al manejo de las colas en EDCA para las distintas ACs) de la arquitectura del estándar IEEE 802.11e EDCA en modo infraestructura, se crea una topología con cuatro nodos inalámbricos diferentes que están ubicados a la

misma distancia del AP. Cada uno de ellos transmitirá un tipo de tráfico con su respectiva categor´ıa de acceso (voz, video, *best effort* y *background*) como se muestra en la Figura 4.2, para fines comparativos (los detalles de la implementación están adjuntos en el Apéndice C).

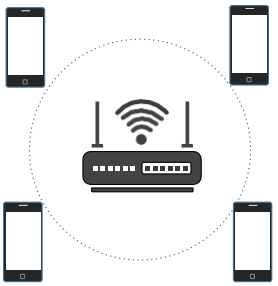

Figura 4.2: Topología modo infraestructura de la red inalámbrica propuesta.

La generación de tráfico multimedia para fines comparativos con la propuesta de Taher [3] requiere de una distribución de Poisson con paquetes de 800 bytes y tasas de llegada variable (λ entre 20 y 1800 kbps).

La función generadora y de gestión del tráfico en cola de la capa MAC, según el estándar IEEE 802.11e EDCA para las distintas categor´ıas de acceso se configura con tiempo de llegada entre paquetes con distribución exponencial (ver Anexo C.1). Es así, como las Figuras 4.3 y 4.4 muestran el comportamiento de EDCA con todos los parámetros activados por defecto (AIFS, CW y TxOP). Se observa que las ACs de video y voz obtienen mejor rendimiento cuando se entra al estado de saturación (tasas de transmisión superiores a los 200 kbps). El rendimiento promedio de las ACs de menor prioridad comienza a disminuir cuando se aumenta la tasa de transmisión. Como se esperaría que ocurriera según el funcionamiento de EDCA.

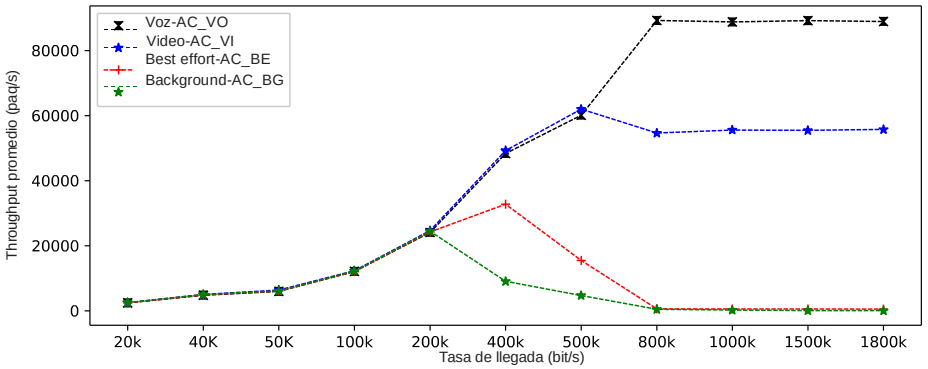

Figura 4.3: Comportamiento del rendimiento por AC en EDCA.

La Figura 4.3 indica que hasta 200 kbps, todas las categor´ıas de acceso tienen el mismo comportamiento (debido a que todas tienen el mismo tiempo en cola y la misma oportunidad de transmisión). Cuando va aumentando la tasa de transmisión, el tráfico con

mayor prioridad se va apoderando de los recursos de la red. Esto garantiza QoS para las AC de voz y video, siempre y cuando compitan con flujos de *best effort* y *background*. El caso de competencia entre tráfico de video se evaluará en secciones posteriores. A partir de 800 kbps, el rendimiento se mantiene aproximadamente constante, debido a que el flujo de mayor prioridad obtiene más recursos en la trasmisión.

De igual forma, la Figura 4.4 muestra el comportamiento del retardo en el escenario WLAN propuesto, donde las AC de más baja prioridad son retenidas en el *buffer* de capa MAC. De esta manera se observa el incremento del retardo por encima de los 400 ms después de configurar tasas de transmisión por encima de los 200 kbps.

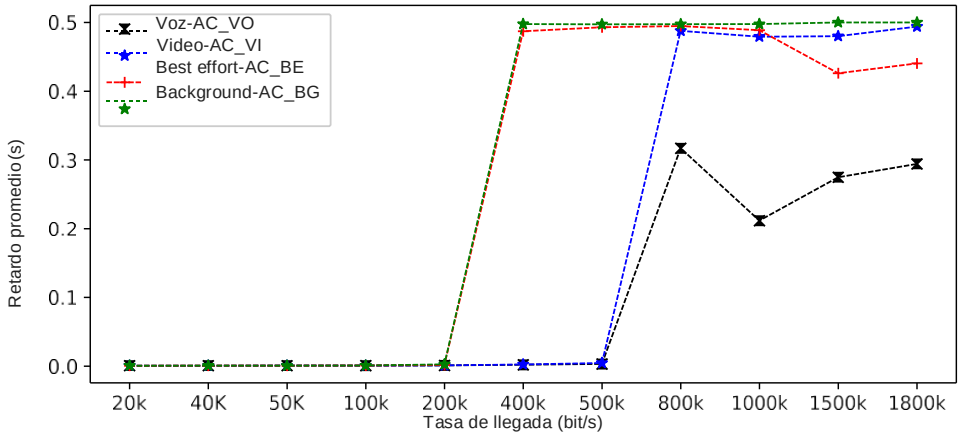

Figura 4.4: Comportamiento del retardo por AC en EDCA.

Una vez desarrollada la topología de red WLAN, bajo el esquema de operación del estándar tradicional IEEE 802.11e, se procederá a adaptarla a un entorno de emulación con un escenario de red inalámbrica basado en SDN, como se explicará en detalle en la siguiente sección.

4.2. Operación de la arquitectura SDN con NS-3

Ahora para crear el entorno de emulación se generará tráfico de tiempo real con el servidor de VLC Player en contenedores aislados de Linux. Para el desarrollo del experimento se consideran tres bloques como indica la Figura 4.5. Cada escenario experimental estará constituido por un AP, un número de estaciones activas, el uso o no de políticas de QoS y un video de referencia que se transmitirá en el entorno de red inalámbrica gestionada con SDN. Los parámetros de evaluación y comparación serán el rendimiento y el retardo para cada situación en cada uno de los escenarios.

El AP de cada escenario estará configurado en modo infraestructura de un solo salto, con un número específico de estaciones inalámbricas activas solicitando video (ver Figura

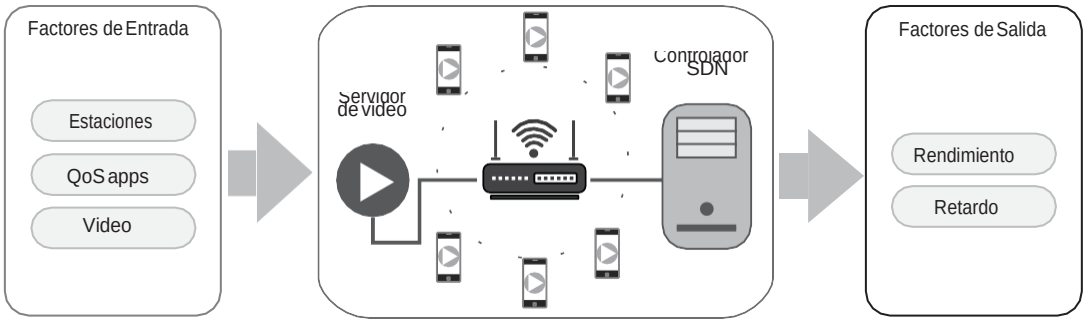

Figura 4.5: Elementos del experimento.

4.6). El primer escenario con 4 estaciones activas, el segundo con 6, y así sucesivamente. Se van incrementando 2 estaciones para cada escenario[1].

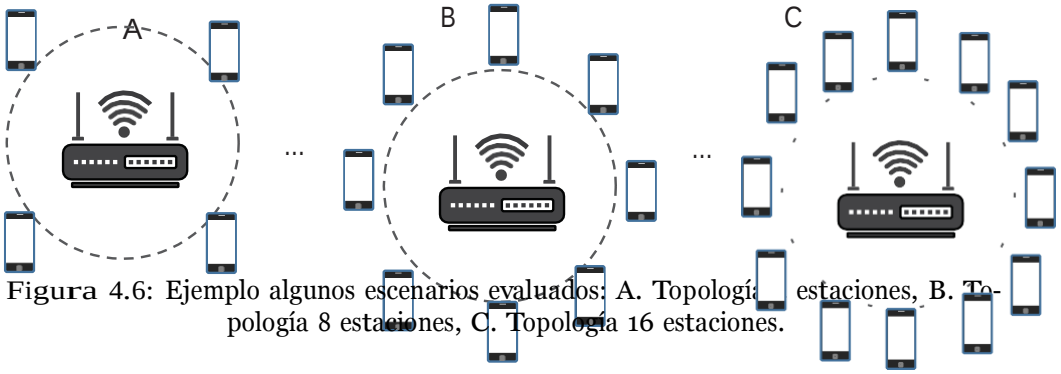

Figura 4.6: Ejemplo algunos escenarios evaluados: A. Topología 4 estaciones, B. Topología 8 estaciones, C. Topología 16 estaciones.

A cada escenario se le configurarán cuatro modos de operación, el modo representará la estrategia de QoS usada en la emulación (ver Tabla 4.2 y Figura 4.7): Modo 1, no usar ningún algoritmo de QoS; Modo 2, usar el mecanismo de QoS por defecto de la aplicación VLC Player; Modo 3, estaciones que solo usarán el algoritmo de control de admisión y finalmente se considerará el Modo 4, para una estación que usará simultáneamente los dos algoritmos de QoS. Los detalles de los algoritmos usados se explicarán en las siguientes secciones.

Tabla 4.2: Modos para las estrategias de QoS usadas

Modos	Descripción
1	No usar ningún algoritmo de QoS
2	Usar el mecanismo de QoS por defecto de VLC Palyer
3	Usar solo el algoritmo de control de admisión (ACA)
4	Usar simultáneamente dos algoritmos de QoS (ACA y QoS-Tag)

[1]El escenario máximo configurado fué de 18 estaciones activas por las limitaciones de recursos de hardware con que se contaba para las emulaciones.

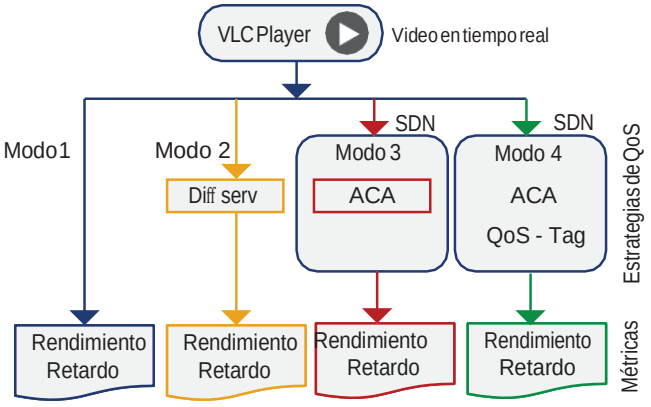

Figura 4.7: Estrategias de QoS implementadas por escenario.

Para realizar la implementación del entorno de emulación de la red inalámbrica basada en SDN, se utiliza NS-3 en modo emulación. La integración del controlador SDN gestionará las aplicaciones de QoS. El entorno de experimentación propuesto se divide en tres bloques principales:

- Bloque de *Linux network namespaces* (NNS) [93] para la separación en contenedores aislados de las estaciones móviles y servidores de video.
- Bloque NS-3 para emulación [94] de la red inalámbrica con las propiedades del estándar IEEE 802.11e.
- Bloque de SDN para la separación del plano de control y datos de la red inalámbrica.

Se usa *Linux network namespaces* para suministrar una copia independiente del *stack* de red con sus propias rutas, reglas de *flujo* y elementos de red. Es decir, cada NNS genera una capa de abstracción que reproduce aplicaciones independientes dentro de un espacio de usuario con recursos de red aislados [95].

NS-3 en modo emulación está conectado a aplicaciones de video de tiempo real (cliente-servidor). Se utiliza NNS para proporcionar aislamiento entre las interfaces de red, el *stack* TCP/IP y la capa de aplicación, tal como se muestra en la Figura 4.8. Open vSwitch (OVS) [96], es un conmutador virtual de código abierto a las principales plataformas hipervisoras. Se utiliza el protocolo OpenFlow, para permitir la comunicación entre el controlador y los dispositivos de red. OVS es un conmutador reprogramable a través de OpenFlow [96], soporta una amplia variedad de interfaces incluyendo *Tap* y V_{eth} (*ethernet virtual*).

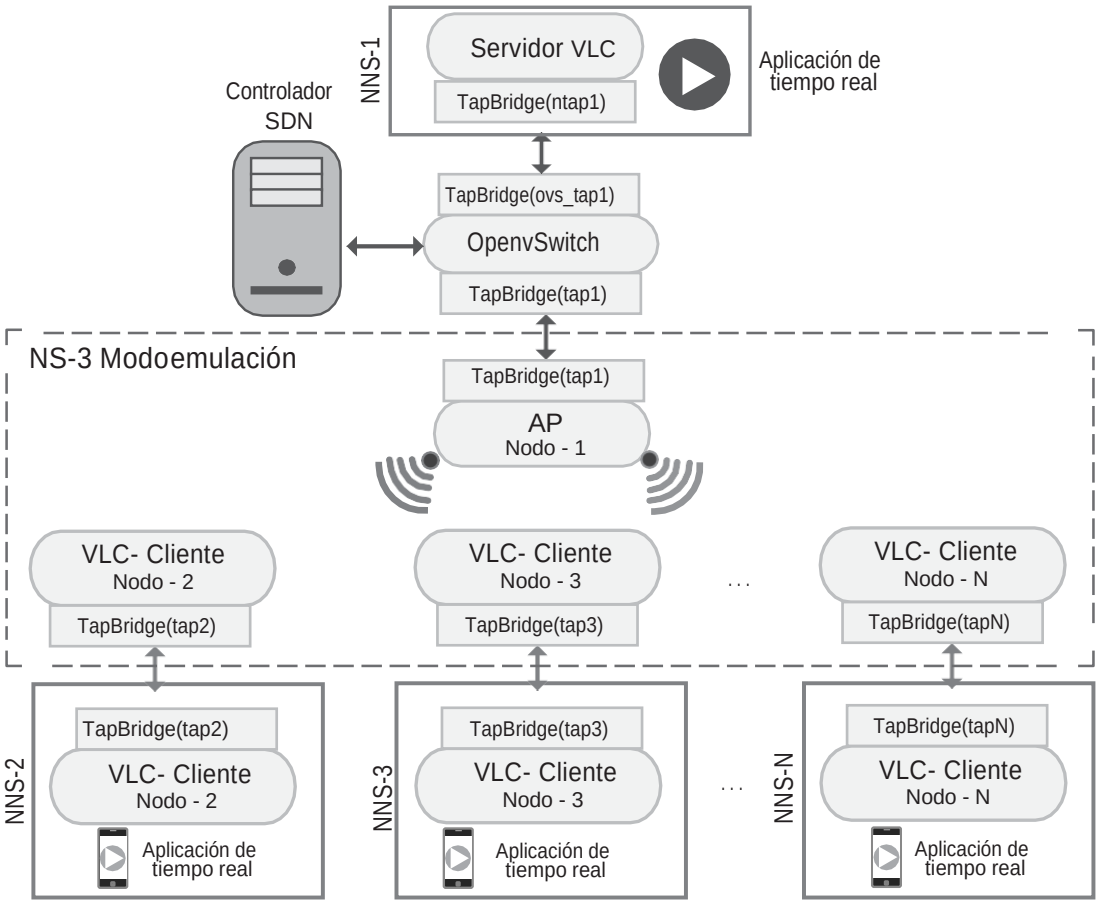

Figura 4.8: Entorno de experimentación propuesto.

Para ejecutar la aplicación VLC Player, tanto en las estaciones *host* como en el servidor de video, el escenario propuesto requiere un dispositivo *Tap bridge* para integrar aplicaciones en tiempo real a NS-3. El *Tap bridge* está diseñado para integrar *hosts* "reales"(*hosts* que admiten dispositivos *Tun/Tap*) en emulaciones de NS-3 [95]. El objetivo es hacerle creer a NS-3 que tiene un nodo *host* "real".

En el escenario de emulación se utiliza un servidor VLC Player que se ejecuta como una aplicación en un servidor independiente. El conmutador OVS con OpenFlow del AP se comunica con el controlador. Este último ejecuta las aplicaciones externas de etiquetado de QoS y de control de admisión. Las emulaciones se ejecutan en los escenarios indicados en la Figura 4.6, utilizando las estrategías indicadas en la Figura 4.7. Del experimento se selecionan 20 ejecuciones para cada escenario con cada uno de los modos de las estrategias usadas.

El entorno de emulación se configura de acuerdo a los parámetros de red indicados en la Tabla 4.3. En esta tabla muestra la configuración por defecto de EDCA. La Tabla 4.4 muestra las caracterísicas del tipo de video usado. Y la Tabla 4.5 los recursos de hardware y software usados en la emulación.

Tabla 4.3: Parámetros de Red

Grupo	Parámetro	Valor
EDCA	AIFSN	2
	MinCW	15
	MaxCW	1024
	TxOPLimit	*Default*

Tabla 4.4: Caracter´ısticas del video

Grupo	Característica	Valor
VIDEO	Nombre	”bigbunny.mp4”[2]
	Resolución	720x480 px
	Codec	H.264
	Tasa de bits	494 kbps
	Duración	60 s
	Frames	30 fps

Tabla 4.5: Recursos de Hardware y Software

Grupo	Recurso	Valor
HARDWARE	RAM	8 GB
	CPU	Intel
SOFTWARE	OS	Ubuntu 14.04.5 LTS
	NS-3	3.26
	OpenFlow	1.0 [3]

En la Figura 4.9 se detalla el procedimiento general realizado para ejecutar las estrateg´ıas de QoS en los escenarios descritos.

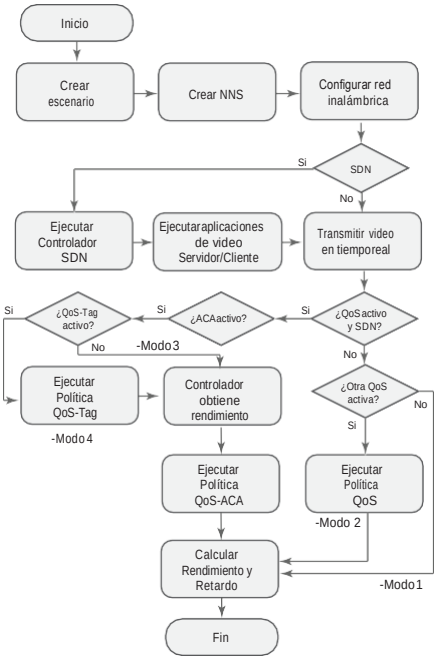

Figura 4.9: Diagrama de flujos de la ejecución del experimento.

[2]Proporciona una variedad de opciones de configuración para los investigadores [97]
[3]Controlador SDN usado soporta OpenFlow 1.0

4.3. Mecanismo para mantenimiento de QoS

Para desarrollar la estrategia del mantenimiento de QoS se propone el Algoritmo 2 de control de admisión (ACA) que aprovecha las funciones del controlador para obtener las estadísticas del tráfico en tiempo real. ACA calcula el rendimiento actual de cada flujo a partir de la información obtenida de OpenFlow (en Byte/s).

El Algoritmo 2 compara los valores teóricos del rendimiento bajo condiciones de saturación del modelo (ver Tabla 3.7) del capítulo anterior, respecto a los valores obtenidos de OVS. Estos valores permiten al Algoritmo 2 tomar la decisión de rechazar o aceptar el nuevo flujo entrante. Es decir, el Algoritmo ACA obtiene las estad´ısticas de flujo y los valores del rendimiento bajo condiciones de saturación en cada ejecución del escenario. Las estadísticas del flujo las suministra la función FLOWSTATS de OpenFlow. Se filtra solo al flujo que pasa por el puerto habilitado para VLC Player, se cuentan los bytes en el rango de tiempo establecido (cada segundo) y se calcula el rendimiento promedio del flujo de video por unidad de tiempo. Por último se rechaza el flujo si el rendimiento calculado está por encima de los valores del rendimiento bajo condiciones de saturación suministradas por el modelo anal´ıtico.

Se aclara que el Algoritmo de control de admisión (ACA) para entornos de redes WLAN gestionadas con SDN es una adaptación del algoritmo de Taher [8] que estaba orientado a redes inalámbricas tradicionales.

Algoritmo 2: Control de admisión (ACA)

```
procedure flowstats(stats, satthroug)          // Ejecutar función
    for flow in stats do                       // Repetir para todos los flujos
        if flow = VLCPort then                 // Filtrar solo tráfico de video
            bytecount ← deltabytecount(flow)   // Contar tráfico en bytes del flujo
            time ← deltatime(flow)             // Tiempo transcurrido
            throughput ← bytecount/time        // Obtener rendimiento
            if throughput > satthroug then     // Rendimiento mayor al modelado
                Reject(flow)                   // Se rechaza flujo
            end if
        end if
    end for
end procedure
```

Otra aplicación que se ejecuta simultáneamente es la de etiquetado de QoS (QoS *Tag*), que clasifica los flujos entrantes -utilizando clases preconfiguradas- según el campo Tipo de servicio (ToS) en la cabecera de los paquetes IP (Ver Algoritmo 3). En primer lugar, el Algoritmo 3 guarda un diccionario con las direcciones IP de origen y destino a las que se les modifica el campo ToS. Cuando el controlador informa de un evento de conexión (*PACKET_IN*), se realiza una comparación para cada dirección IP del diccionario (dupla

IP origen y destino). Después de eso, si el resultado de la comparación es verdadero, se modificará el campo ToS con un cambio de prioridad sobre los flujos.

Algoritmo 3: Etiquetado campo de ToS (QoS *Tag*)

```
procedure QoS(Src, Dst)                          // Obtener IP origen-destino para QoS
    qoslist ←dictionary{Src, Dst}                // Listar IP origen y destino
    while HandleConectionUp do                   // Cuando se crea evento de conexión
        for ip in qoslist do                     // Repetir para toda la lista
            match (IP v4, newSrc, newDst)        // Comparar coincidencias de IP
            actions.append (T OS = 128)          // DSCP-Flash Override Video Data
            actions.append (OF P P ALL)          // Aplicar a todas las coincidencias
            send (Msg)                           // Enviar regla de flujo
        end for
    end while
end procedure
```

Como se había definido en el Capítulo 2, el rendimiento representa el número de paquetes entregados con éxito por unidad de tiempo y el retardo representa el tiempo que tarda un paquete en ser transmitido con éxito.

Para analizar el rendimiento y el retardo de los resultados de las estrategias de QoS usadas en cada repetición del experimentos se utilizaron las ecuaciones 4.1 y 4.2:

$$Rendimiento\ promedio = \frac{Carga\ útil\ total}{\Delta[T\ iempo\ T\ x\ total]} = \frac{\sum_{x=1}^{x=N} P^x}{t_{fN} - t_{i1}} \qquad (4.1)$$

Donde:

N Número de paquetes transmitidos con éxito
P_x Carga útil de cada paquete
t_{fN} Tiempo donde termina la transmisión
t_{i1} Tiempo donde inicia la transmisión

$$Retardo\ promedio = \frac{\Delta[T\ iempo\ paquetes\ Rx]}{N} = \frac{1}{N}\sum_{j=1}^{j=N} t_{f\,j} - t_{ij}\ \ Con\ N > 0. \qquad (4.2)$$

Donde:

N Número de paquetes transmitidos con éxito, si N = 0 no existió transmisión
t_{fj} Tiempo de llegada exitosa del paquete j
t_{ij} Tiempo inicio en cola del paquete j

4.4. Evaluación del desempeño

En esta sección se mostrarán los resultados obtenidos del análisis de las trazas que resultaron de las emulaciones para cada uno de los escenarios con y sin la implementación de estrategias de QoS. En la primera parte se indica el rendimiento general promedio sin y con la implementación del algoritmo de control de admisión de todo el tráfico de la red. En la segunda parte, se muestra la variación del *throughput* con y sin la implementación de cada uno de los mecanismos de QoS. Finalmente, igual que para el rendimiento se muestra el retardo promedio experimentado por el tráfico de video.

4.4.1. Validación del funcionamiento del Algoritmo ACA propuesto

Los archivos de captura (*.pcap* de Wireshark) contienen la información del tráfico transmitido por cada una de las estaciones activas. Un ejemplo de esta captura es la Figura 4.10 que muestra el comportamiento del rendimiento con la entrada consecutiva de nuevas estaciones[4]. A manera de ejemplo se toma el escenario con 10 estaciones activas solicitando video y se puede evidenciar la disminución del rendimiento general de la red. Aquí todo el tráfico compite por los recursos de la red, en el intervalo entre los segundos 40 y 50, el rendimiento total de las estaciones disminuye debido a las limitaciones del canal y a la superación de los límites de rendimiento máximo que soporta la red con los parámetros por defecto de EDCA (Tabla 4.3). La transmisión en esos intervalos de tiempo hace que los flujos sean retenidos más tiempo en el *buffer* de la MAC, incrementando las probabilidades de colisión. Las condiciones del rendimiento se reestablecen cuando finaliza el requerimiento para la entrada de nuevos flujos.

[4]La transmisión inicia a partir del segundo 16 en el eje del tiempo. El rendimiento medido inicia en este tiempo debido a que en los primeros 15 segundos se realizan todas las configuraciones del escenario y todas las peticiones ICMP que validan las conexiones de la topología de red

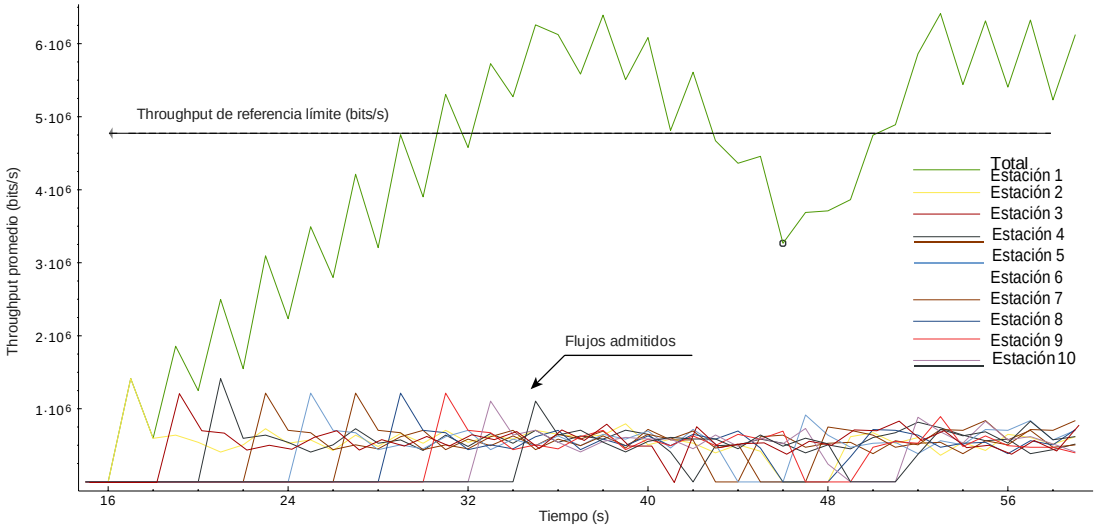

Figura 4.10: Rendimiento de la red sin QoS. Flujo entrante de video por estación.

La Figura 4.10 muestra que el rendimiento total se incrementa de manera escalonada en el eje de las ordenadas, cada vez que ingresa una nueva estación solicitando transmisión de video hasta que finaliza el ingreso de nuevas estaciones -dibujando así un patrón diente de sierra-. Sobre la misma figura se traza una l´mea de referencia horizontal para comparar los niveles de rendimiento alcanzado cuando están o no habilitadas las políticas de QoS implementadas.

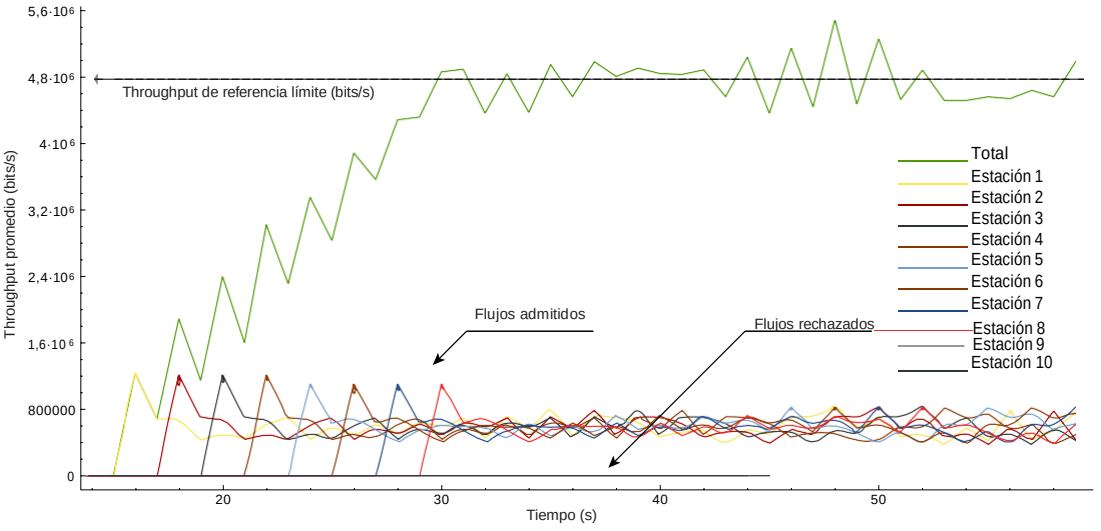

Figura 4.11: Rendimiento de la red con QoS. Flujo entrante de video por estación con control de admisión.

Bajo el mismo escenario, en la Figura 4.11, donde se ejecuta el Algoritmo 2 de control de admisión, se demuestra que de las 10 estaciones que intentan recibir el video desde

el servidor VLC, solo 8 logran tener éxito. El mecanismo de control de admisión mantiene controlado el rendimiento total de la red sin dejar que ésta llegue a los límites de rendimiento máximo soportados. Los flujos de las estaciones que saturan la red se ven penalizados y el control de admisión, con base en el rendimiento, solo les permitirá transmitir cuando existan los recursos disponibles.

4.4.2. Evaluación del rendimiento (*Throughput*) de la estrategia implementada

En esta sección se mostrarán los resultados de las emulaciones para cada uno de los escenarios establecidos (según Figura 4.6) y para cada estrategia implementada sobre las estaciones según la Tabla 4.2. Se realiza un comparativo entre los escenarios para cada modo como muestra la Tabla 4.6.

Tabla 4.6: Comparativa del rendimiento en los distintos escenarios

Escenario	Estaciones Evaluadas	Media (kbps)	Desviación Estándar (kbps)	Intervalo de Confianza(kbps) 95 %
4 Estaciones	Modo 1	557.20	19.41	(508.94 - 605.40)
	Modo 2	565.71	25.29	(525.47 - 605.95)
	Modo 3	556.17	52.47	(503.69 - 608.64)
	Modo 4	602.70	0.58	(602.42 - 602.97)
6 Estaciones	Modo 1	532.29	47.34	(484.95 - 579.63)
	Modo 2	544.30	45.61	(496.44 - 592.17)
	Modo 3	531.75	39.39	(482.84 - 580.66)
	Modo 4	603.08	0.90	(602.66 - 603.50)
8 Estaciones	Modo 1	505.65	55.45	(454.37 - 556.94)
	Modo 2	472.90	62.68	(420.50 - 525.23)
	Modo 3	491.14	48.58	(440.15 - 542.13)
	Modo 4	579.39	34.89	(563.06 - 595.72)
10 Estaciones	Modo 1	441.04	79.58	(379.87 - 502.21)
	Modo 2	428.36	111.43	(348.65 - 508.06)
	Modo 3	481.41	69.85	(408.11 - 554.71)
	Modo 4	577.57	47.87	(555.17 - 599.97)
12 Estaciones	Modo 1	385.69	116.54	(307.40 - 463.98)
	Modo 2	378.09	144.28	(286.42 - 469.77)
	Modo 3	465.12	72.91	(388.61 - 541.63)
	Modo 4	578.51	42.27	(557.49 - 599.53)
14 Estaciones	Modo 1	342.26	139.63	(257.88 - 426.63)
	Modo 2	330.25	168.73	(232.83 - 427.68)
	Modo 3	453.57	80.12	(369.49 - 537.65)
	Modo 4	568.41	51.09	(543.78 - 593.03)
16 Estaciones	Modo 1	294.11	148.53	(211.86 - 376.36)
	Modo 2	301.04	176.87	(203.09 - 398.99)
	Modo 3	445.15	99.27	(340.97 - 549.32)
	Modo 4	595.01	60.60	(566.65 - 623.37)
18 Estaciones	Modo 1	285.66	154.58	(200.06 - 371.26)
	Modo 2	284.69	175.21	(187.66 - 381.72)
	Modo 3	429.14	93.23	(331.31 - 526.98)
	Modo 4	597.65	54.10	(571.57 - 623.72)

A partir de los valores mostrados de la Tabla 4.6, se obtienen las Figuras 4.12, 4.13, 4.14 y 4.15 que representan el comportamiento específico del rendimiento promedio para cada uno de los escenarios evaluados y las respectivas estrategias implementadas.

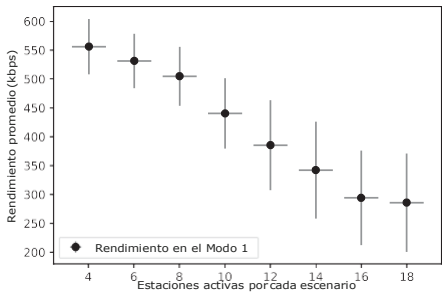

Figura 4.12: Rendimiento en Modo 1.

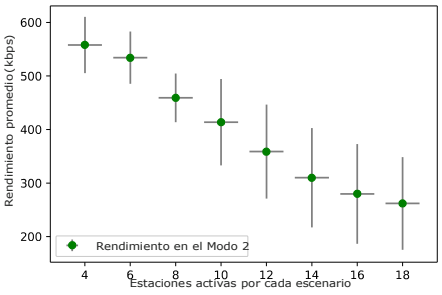

Figura 4.13: Rendimiento en Modo 2.

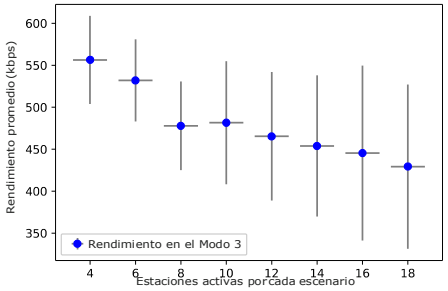

Figura 4.14: Rendimiento en Modo 3.

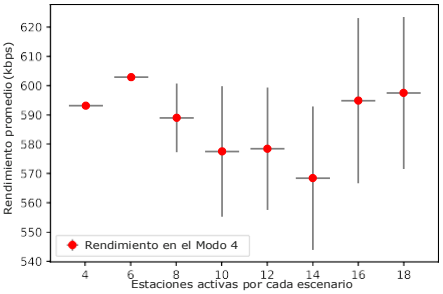

Figura 4.15: Rendimiento en Modo 4.

Para la Figura 4.12, con la estrategia en Modo 1, se observa un rendimiento con tendencia decreciente que parte en 557 kbps y finaliza con 285 kbps en el último escenario. La variación total entre el primer y último escenario es de 272 kbps.

Para tener como referencia otro mecanismo de QoS, se utilizó el que viene por defecto en la aplicación de VLC Player (ToS/DSCP), que usa las características del mecanismo Diffserv mediante la modificación de las cabeceras del flujo en su campo de ToS para asignar las prioridades en la reservación de recursos (Modo 2). La Figura 4.13 muestra el comportamiento del rendimiento promedio de todas las estaciones para cada escenario evaluado. De manera similar, como ocurrió en el Modo 1, aquí todos los flujos compiten de igual forma por los recursos limitados. Así las primeras estaciones logran el mejor desempeño, pero se degradan en la medida en que aparecen nuevas solicitudes de admisión. En esta figura el rendimiento pasó de 565 kbps a 284 kbps, con una desviación estándar entre todos los escenarios de 107 kbps.

En la Figura 4.14, con el Modo 3 activado, se muestra un rendimiento promedio que parte en 556 kbps para el primer escenario y decrece hasta los 429 kbps sobre el último escenario. La variación de este modo alcanza un valor de 127 kbps de diferencia entre el primer y último escenario con desviación estándar de 166 kbps.

La Figura 4.15 muestra los efectos de ejecutar el Modo 4 sobre una estación en particular. Esta figura da cuenta de los valores promedio del rendimiento entre los que se mantiene

una estación. Su valor máximo (603 kbps) se alcanza en el segundo escenario, el mínimo (568 kbps) en el escenario con 14 estaciones y finaliza con un rendimiento de 597 kbps para el último caso estudiado. La desviación estándar de los promedio del rendimiento entre los escenarios evaluados es de 11.84 kbps, miestras que para el primer y segundo escenario la desviación de las medidas del rendimiento estuvieron por debajo de 1 kbps.

Para analizar el comportamiento del rendimiento entre las estrategias usadas se realiza un comparativo entre los valores obtenidos en las figuras 4.12, 4.14 y 4.15 respectivamente, lo que produce la Figura 4.16. En esta se puede identificar la tendencia que tienen tres de los cuatro modos estudiados, resaltando que para la estación en la que se aplican los dos algoritmos de QoS el rendimiento oscila alrededor de 568 y 597 kbps mientras que para las estaciones que no ejecutan los algoritmos de QoS su rendimiento disminuye hasta los 272 kbps en el escenario con 18 estaciones activas. Como se indica en la Figura 4.17, los algoritmos de QoS presentan una ventaja de hasta el 117 % en el escenario con 18 estaciones activas.

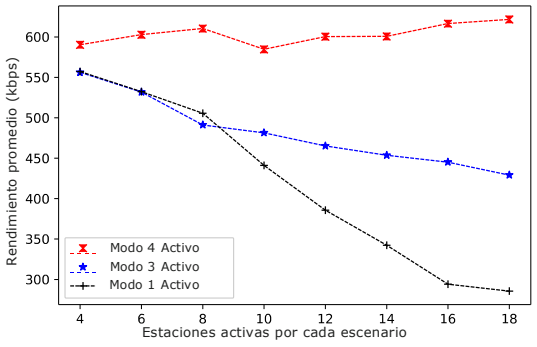

Figura 4.16: Comparativa del rendimiento entre Modos 1, 3 y 4.

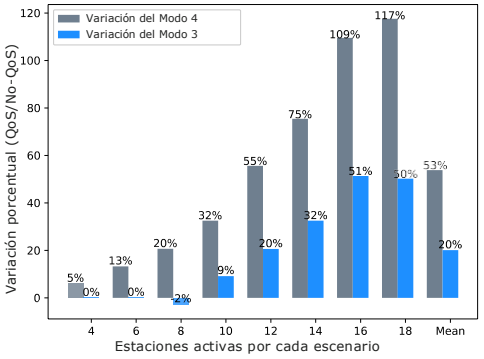

Figura 4.17: Variación porcentual rendimiento Modos 3 y 4 respecto a 1.

Para mostrar la tasa de variación porcentual en cada escenario logrado en el Modo 3 y 4 respecto al Modo 1 (referencia) se presenta la Figura 4.17. Aqu´ıse evidencia la mejora obtenida mediante el uso de las aplicaciones de QoS gestionadas con SDN. La mejora va desde un 5 %, en un escenario con 4 estaciones, hasta un 117 % para el escenario con 18 estaciones inalámbricas, siendo este último el mejor de los casos evaluados en el alcance de este proyecto.

De forma independiente utilizando el procedimiento anterior se evaluó el rendimiento de todas las estaciones en Modo 2 y Modo 3 respecto al Modo 1 como referente. Se identifica una mejora del rendimiento promedio en el Modo 3 respecto al Modo 2 en 168 kbps como se observa en la Figura 4.18. La variación porcentual indicada en la Figura 4.19 muestra que el mecanismo (Diffserv) usado por VLC no gestiona adecuadamente el rendimiento cuando existen aplicaciones compitiendo con la misma prioridad en sus flujos.

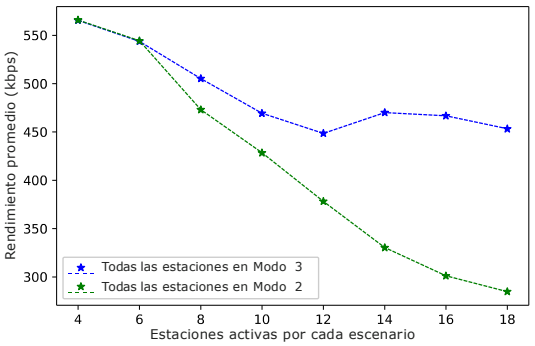

Figura 4.18: Comparativa rendimiento entre los Modos 2 y 3.

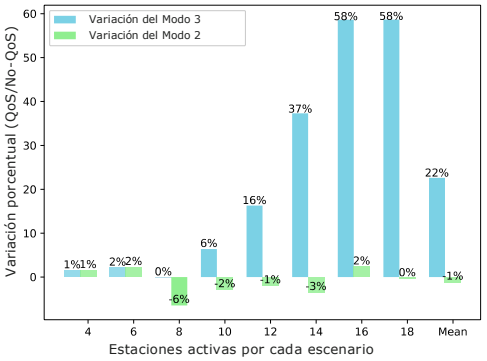

Figura 4.19: Variación porcentual Modos 2 y 3 respecto al Modo 1.

4.4.3. Evaluación del retardo (*Delay*) de la estrategia implementada

Como en la subsección anterior, tomando los mismos datos de las trazas, se realizó un comparativo entre los cuatro modos de estrategias de QoS (Tabla 4.2) aplicadas en cada escenario (ver Tabla 4.7). El resultado se resume en la Figura 4.20 que demuestra que existe, en algunos casos, un mayor retardo al usar los algoritmos de QoS, pero que éste no crece a medida que se aumenta el número de estaciones activas, como en el caso del rendimiento. S´ıes evidente que se agrega en la mayor´ıa de los escenarios un retardo leve al adicionar nuevas reglas al flujo entrante, pero este no supera un milisegundo en el peor de los casos de los escenarios evaluados. La Figura 4.21 mide la variación porcentual del retardo para las estaciones que aplican el Modo 3 y 4. Las estaciones se ven afectadas por un retardo que se incrementa hasta un 19 % respecto a ningún uso de QoS. Caso distinto se observa para el promedio del retardo en las estaciones que están en el Modo 3, que presentan una disminución en el valor de esta métrica de hasta el 7 % para el escenario con 14 estaciones activas (un retardo menor implica mejor desempeño).

Tabla 4.7: Comparativa del retardo en los distintos escenarios

Escenario	Estrategias Evaluadas	Media (s)	Desviación Estándar (s)	Intervalo de Conftanza (s) 95 %
4 Estaciones	Modo 1	0.00252	0.000118	(0.002227 - 0.002813)
	Modo 2	0.00258	6.41E-05	(0.002474 - 0.002678)
	Modo 3	0.00285	0.000272	(0.002176 - 0.003528)
	Modo 4	0.00300	0.000682	(0.002681 - 0.003319)
6 Estaciones	Modo 1	0.00288	0.000162	(0.002683 - 0.003085)
	Modo 2	0.00269	0.000110	(0.002578 - 0.002808)
	Modo 3	0.00293	0.000350	(0.002499 - 0.003368)
	Modo 4	0.00331	0.000838	(0.002915 - 0.003699)
8 Estaciones	Modo 1	0.00308	0.000137	(0.002957 - 0.003211)
	Modo 2	0.00273	0.000130	(0.002620 - 0.002838)
	Modo 3	0.00299	0.000264	(0.002753 - 0.003241)
	Modo 4	0.00325	0.000630	(0.002945 - 0.003552)
10 Estaciones	Modo 1	0.00301	0.000143	(0.002903 - 0.003122)
	Modo 2	0.00277	0.000221	(0.002616 - 0.002932)
	Modo 3	0.00286	0.000152	(0.002700 - 0.003019)
	Modo 4	0.00305	8.35E-05	(0.003007 - 0.003085)
12 Estaciones	Modo 1	0.00302	0.000231	(0.002869 - 0.003179)
	Modo 2	0.00268	0.000246	(0.002525 - 0.002837)
	Modo 3	0.00285	0.000147	(0.002697 - 0.003007)
	Modo 4	0.00327	0.000399	(0.003075 - 0.003472)
14 Estaciones	Modo 1	0.00302	0.000160	(0.002921 - 0.003114)
	Modo 2	0.00254	0.000255	(0.002395 - 0.002690)
	Modo 3	0.00278	9.25E-05	(0.002685 - 0.002879)
	Modo 4	0.00326	0.000329	(0.003101 - 0.003418)
16 Estaciones	Modo 1	0.00281	0.000192	(0.002706 - 0.002919)
	Modo 2	0.00253	0.000331	(0.002348 - 0.002715)
	Modo 3	0.00270	0.000146	(0.002543 - 0.002850)
	Modo 4	0.00324	0.000271	(0.003109 - 0.003362)
18 Estaciones	Modo 1	0.00272	0.000201	(0.002611 - 0.002834)
	Modo 2	0.00237	0.000254	(0.002234 - 0.002515)
	Modo 3	0.00256	0.000227	(0.002324 - 0.002801)
	Modo 4	0.00324	0.000225	(0.003132 - 0.003348)

Un caso particular del retardo se presenta en la Figura 4.22 donde se evidencia menor retardo en los escenarios donde las estaciones usan el mecanismo por defecto que ofrece el servidor de video, respecto a las que no usan QoS y tampoco ACA. Según la Figura 4.23 el retardo más alto (12 %) se alcanzó en el primer escenario para el Modo 3. Otra situación contraria también se observa en el escenario con 14 estaciones activas recibiendo tráfico de video, donde se tiene un retardo menor (en un 15 %) en las estaciones configuradas en el Modo 2. El promedio del retardo (usando la ecuación 4.2) en los escenarios emulados para el mecanismo de VLC muestra una variación porcentual entre el 6 % y 15 % de mejora en el rendimiento. Como ya se hab´ıa indicado, todos los comparativos

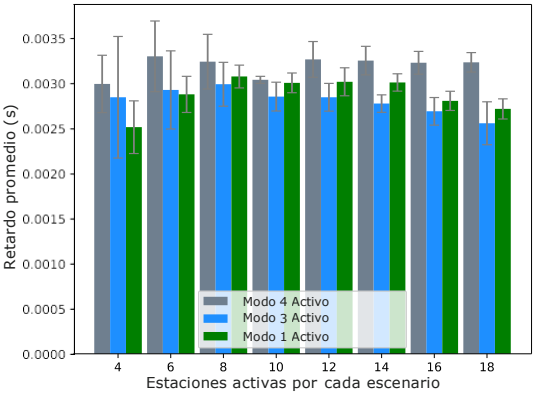

Figura4.20: Comparativa del retardo entre los Modos 1, 3 y 4.

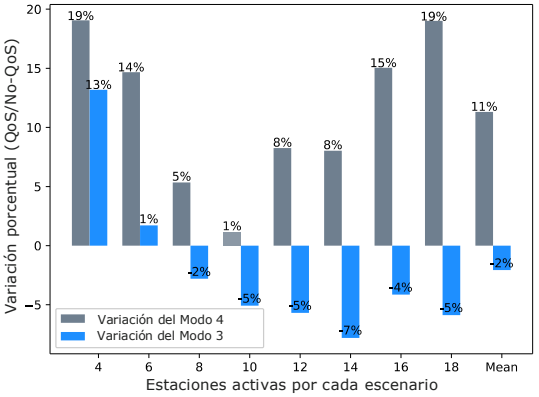

Figura 4.21: Variación porcentual retardo Modos 3 y 4 respecto Modo 1.

porcentuales son respecto a los valores de rendimiento alcanzado por las estaciones que no usan ningún mecanismo de QoS (Modo 1).

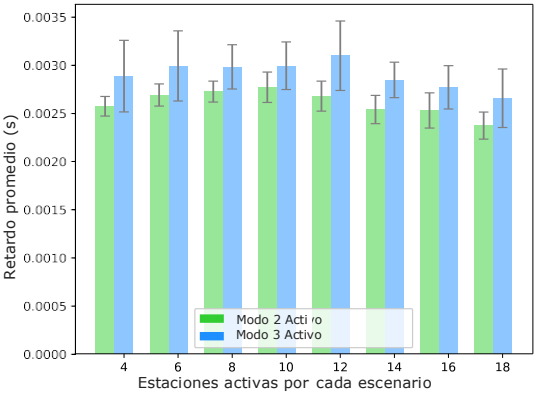

Figura4.22: Comparativa del retardo entre los Modos 2 y 3..

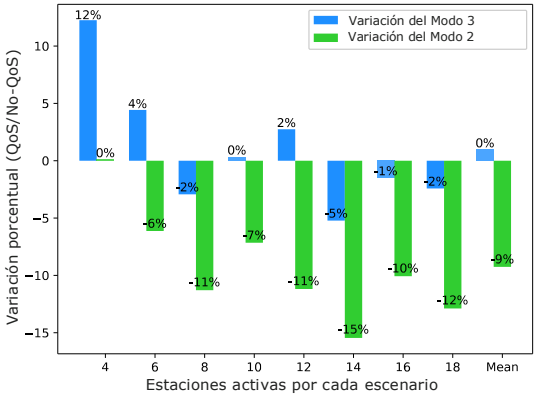

Figura 4.23: Variación retardo Modos 2 y 3 respecto al Modo 1.

En el desarrollo de este capítulo se simuló -para efectos comparativos- un primer escenario que validó el comportamiento por defecto del estándar IEEE 802.11e, variando la tasa de transmisión. Una vez validado el comportamiento del estándar se adaptó este a un escenario de emulación de red inalámbrica gestionada con SDN para la transmisión de video en tiempo real usando las caracter´ısticas que ofrece *Linux network namespaces* para el manejo de aplicaciones. Se implementaron dos algoritmos como mecanismos de garantía de QoS. Se comparó el rendimiento y el retardo para diferentes modos de estrategias como: a) no usar ningún algoritmo de QoS, b) usar un mecanismo de QoS propietario de la aplicación del servidor de video, c) el comportamiento sobre estaciones que solo usan el algoritmo de control de admisión y d) el comportamiento sobre una estación que usa los dos algoritmos de QoS propuestos. Los resultados indican (ver Tabla 4.8 y Tabla 4.9) que para el Modo 4 se obtuvo mejor rendimiento, pero para el Modo

Tabla 4.8: Resumen comparativa del rendimiento (máximos y mínimos) por escenarios y modos (kbps)

ESCENARIOS	MODO 1	MODO 2	MODO 3	MODO 4
4	557.2	565.71	556.17	602.7
6	532.29	544.3	531.75	603.08
8	505.65	472.9	491.14	579.39
10	441.04	428.36	481.41	577.57
12	385.69	378.09	465.12	578.51
14	342.26	330.25	453.57	568.41
16	294.11	301.04	445.15	595.01
18	285.66	284.69	429.14	597.65

MAX	557.2	565.71	556.17	603.08
MIN	285.66	284.69	429.14	568.41

RENDIM/MODO	
MAX	MIN
MODO 4	MODO 3
MODO 4	MODO 3
MODO 4	MODO 2
MODO 4	MODO 2
MODO 4	MODO 2
MODO 4	MODO 2
MODO 4	MODO 1
MODO 4	MODO 2

Tabla 4.9: Resumen comparativa del retardo (máximos y mínimos) por escenarios y modos (s)

ESCENARIOS	MODO 1	MODO 2	MODO 3	MODO 4
4	0.00252	0.00258	0.00285	0.00300
6	0.00288	0.00269	0.00293	0.00331
8	0.00308	0.00273	0.00299	0.00325
10	0.00301	0.00277	0.00286	0.00305
12	0.00302	0.00268	0.00285	0.00327
14	0.00302	0.00254	0.00278	0.00326
16	0.00281	0.00253	0.0027	0.00324
18	0.00272	0.00237	0.00256	0.00324

MAX	0.00308	0.00277	0.00299	0.00331
MIN	0.00252	0.00237	0.00256	0.00300

RETARDO/MODO	
MAX	MIN
MODO 4	MODO 1
MODO 4	MODO 2
MODO 4	MODO 2
MODO 4	MODO 2
MODO 4	MODO 2
MODO 4	MODO 2
MODO 4	MODO 2
MODO 4	MODO 2

2 se obtuvo menor retardo para las estaciones que lograron transmitir, debido a que los flujo no pasaban por la gestión del controlador SDN.

Capítulo 5

Conclusiones y Trabajos futuros

5.1. Conclusiones

En este trabajo se identificó, seleccionó, implementó y evaluó un mecanismo de control de admisión como estrategia para garantizar QoS en la transmisión de tráfico de video en tiempo real para un entorno de red inalámbrica gestionada con SDN. Esta estrategia funciona sin la necesidad de reconfigurar los parámetros por defecto de la capa MAC del estándar IEEE 802.11e.

Las bases del modelo analítico seleccionado suministraron los límites teóricos de rendimiento que permitieron tomar la decisión de rechazar o aceptar secuencias de flujos de video en tiempo real con el algoritmo de control de admisión propuesto para los escenarios SDWN emulados. También se desarrolló en NS-3 un escenario que mostró el comportamiento de EDCA respecto al manejo de cada uno de los tipos de flujos (voz, video, *best effort* y *background*) como referencia del funcionamiento por defecto del estándar IEEE 802.11e. Luego, este entorno de simulación se adaptó a otro de emulación para la transmisión de tráfico de video en tiempo real sobre una red inalámbrica gestionada con SDN. Para esto se aprovecharon las capacidades de los bloques de *Linux network namespaces* (NNS) para la separación en contenedores aislados donde se ejecutaron las aplicaciones de las estaciones móviles y de los servidores de video.

Se implementaron dos aplicaciones para QoS ejecutadas por el controlador SDN. Primero, una aplicación de etiquetado del campo ToS que modifica la prioridad del tráfico de video transmitido sobre las estaciones móviles, teniendo en cuenta las direcciones IP de origen y destino. En la aplicación de control de admisión, que obtiene en tiempo real la estadística del rendimiento de los flujos suministrada por OpenFlow, se calculó el rendimiento en intervalos de tiempo deseados. Con base en esta estad´ıstica se toma la

decisión de bloquear los flujos de las últimas estaciones que intentan acceder al medio inalámbrico. El rechazo del nuevo flujo de las aplicaciones de una estación se realiza mediante la modificación de las reglas de flujos en el conmutador Open vSwitch del AP.

Los resultados indicaron que las estrategias de mantenimiento de QoS para la red inalámbrica basada en SDN garantizan el rendimiento hasta un 117 % en el mejor de los casos estudiados, esto en comparación con la misma red inalámbrica sin los mecanismos de QoS gestionados con SDN. Esto indica que el rendimiento promedio en las estaciones que están afectadas con las políticas de calidad de servicio gestionadas bajo el paradigma SDN se mantiene en los niveles tolerables por debajo de las condiciones de saturación, incluso cuando se aumentan el número de estaciones activas. Además, el rendimiento promedio de todas las estaciones activas para cada uno de los escenarios mejora hasta un 50 % aplicando solo una estrategia de QoS (ACA, para este caso). Para este último caso el rendimiento promedio fue superior al rendimiento medio ofrecido por el mecanismo Diffserv utilizado por VLC Player.

El retardo promedio para todos los escenarios presentó una variación de incremento de hasta un 19 % cuando se usaron los dos algoritmos de mantenimiento de QoS. En términos generales el retardo no mostró una tendencia creciente continua significativa a medida que se aumentaba el número de estaciones activas con aplicaciones que solicitan video. En el peor de los casos estudiados el retardo no superó los 0,8 ms adicionales con el uso de las aplicaciones de QoS gestionadas con SDN. Un caso contrario del comportamiento del retardo se presentó al usar un mecanismo de QoS de VLC Player que para este caso particular presentó una leve mejora en la disminución del valor de esta métrica respecto al promedio de las estaciones que usaron los algoritmos de garantías de QoS.

Aunque el controlador SDN involucra un procesamiento adicional sobre los flujos entrantes al principio de la transmisión, este retardo mantiene niveles temporales aceptables exigidos para garantizar QoS en el tráfico de video en tiempo real transmitido.

Como parte de la divulgación de los resultados obtenidos en este trabajo de investigación, se presentó el siguiente artículo de conferencia:

- *Admission control implementation for QoS performance evaluation over SDWN (2018) [98].*

5.2. Trabajos futuros

Este trabajo expone ideas hacia las que se pueden orientar trabajos futuros dentro de los grupos de investigación, tales como:

- Evaluar el comportamiento de otros tipos de tráfico como el de voz (AC_VO) en tiempo real, junto con el uso de otro tipo de tráfico de manera simultánea que intenten saturar aún más el canal.
- Implementar las aplicaciones de QoS emuladas sobre escenarios con equipos físicos.
- Implementar y comparar el diseño del escenario propuesto sobre otros controladores como RYU u OpenDaylight que soportan versiones superiores de OpenFlow.
- Expandir el alcance del entorno de emulación a las siguientes enmiendas al estándar como: IEEE 802.11aa, IEEE 802.11ac, IEEE 802.11ax, entre otras, usadas y próximas a ser implementadas en la industria.
- Aprovechar la flexibilidad y programabilidad que ofrece SDN para evaluar estrategias que aprovechen el potencial de *Machine Learning* e integrarlo para construir modelos de QoS autoajustables a las condiciones de tiempo real que impone la red.

Apéndice A

Tablas de resultados del modelo

Tabla A.1: Probabilidades de canal ocupado y de colisión para AC_VI

#Estaciones	Pb	Pc
2	0,20	0,13
4	0,27	0,22
6	0,31	0,27
8	0,34	0,31
10	0,37	0,34
12	0,39	0,37
14	0,41	0,39
16	0,42	0,41
18	0,44	0,42

Tabla A.2: Valores de referencia del rendimiento de saturación (Kbps)

# Estaciones	AC_VO	AC_VI	AC_BE	AC_BG	Total
2	2271,35	2056,04	438,25	196,38	4962,03
4	2317,60	1878,75	325,39	104,31	4626,06
6	2266,39	1746,85	255,14	64,45	4332,83
8	2205,51	1650,27	208,36	43,54	4107,69
10	2148,01	1575,95	175,22	31,23	3930,41
12	2096,08	1516,46	150,65	23,40	3786,59
14	2049,64	1467,43	131,78	18,12	3666,96
16	2008,06	1426,09	116,89	14,40	3565,44
18	1970,68	1390,61	104,87	11,69	3477,85

Tabla A.3: Rendimiento por defecto en la simulación de EDCA (Kbps)

Tasa de llegadas	AC_VO	AC_VI	AC_BE	AC_BG
20k	161,787	159,049	161,366	161,366
40K	314,375	326,072	321,541	328,542
50K	398,455	415,021	400,989	400,317
100k	808,045	815,193	817,135	788,785
200k	1603,88	1615,87	1597,67	1623,69
400k	3226,15	3237,98	2161,51	828,885
500k	3998,67	4045,26	1038,31	1002,27
800k	5979,62	3736,95	54,6895	2612,57
1000k	5952,19	3966,85	32,7206	1010,16
1500k	5929,25	4140,26	40,1767	56,094
1800k	5916,56	4142,71	44,4209	59,7217

Tabla A.4: Retardo por defecto en la simulación EDCA (ms)

Tasa de llegadas	AC_VO	AC_VI	AC_BE	AC_BG
20k	0,625	0,629	0,638	0,641
40K	0,653	0,657	0,679	0,689
50K	0,668	0,676	0,699	0,718
100k	0,75	0,775	0,868	0,922
200k	0,994	1,1	1,751	2,612
400k	1,929	2,433	487,336	497,748
500k	3,067	4,659	493,054	497,259
800k	316,982	487,878	494,772	497,458
1000k	211,564	479,364	488,769	497,71
1500k	274,744	480	426,232	500
1800k	294,137	494,031	440,443	500

Apéndice B

Procedimiento cálculo de los promedios

La Tabla B.1 muestra el procedimiento que se aplicó para cada escenario (este es un caso part´ıcular, escenario con cuatro estaciones, con la estrategia de aplicar dos algoritmos de QoS -Modo 4-). Cada escenarios tiene 20 repeticiones, un número de estaciones activas (para este caso A, B, C y D). A cada estación se le calculó el promedio del rendimiento, la desviación estándar, los intervalos de confianza con límite inferior y superior, la media del promedio de tadas las estaciones con su respectiva desviación. Esto se realizó para los 8 escenarios y los 4 modos de estrategia de QoS usada.

Tabla B.1: Procedimiento cálculo de los promedios del rendimiento

Escenario	Iteración	Rendimientos (Kbps)			
#	#	A	B	C	D
1: MODO 4	1	593.72	578.15	554.23	536.89
1: MODO 4	2	593.71	578.13	555.46	532.79
1: MODO 4	3	593.74	578.14	554.15	536.5
1: MODO 4	4	592.04	578.06	553.89	536.04
1: MODO 4	5	592.05	577.92	553.21	536.01
1: MODO 4	6	592.04	576.71	553.97	536.33
1: MODO 4	7	591.44	576.95	553.32	538.29
1: MODO 4	8	593.72	577.96	554.34	536.76
1: MODO 4	9	593.71	578.73	553.9	536.71
1: MODO 4	10	594.13	578.8	555.87	536.33
1: MODO 4	11	593.64	577.89	554.68	536.37
1: MODO 4	12	593.72	578.61	555.63	537.41
1: MODO 4	13	593.72	578.65	554.53	536.03
1: MODO 4	14	592.04	578.38	554.06	530.3
1: MODO 4	15	595.83	579.37	555.63	535.32
1: MODO 4	16	593.63	578.19	555.11	536.75
1: MODO 4	17	593.72	578.13	554.36	536.93
1: MODO 4	18	593.73	578.69	554.92	537.14
1: MODO 4	19	591.43	575.92	551.77	534
1: MODO 4	20	594.13	578.94	554.94	536.74
PROMEDIO		593.29	578.12	554.4	535.98
DESVIACIÓN		1.0979	0.8043	0.969	1.7801
INTER. CONF		0.5138	0.3764	0.4535	0.8331
I.C. LIM. INF.		592.78	577.74	553.94	535.15
I. C. LIM. SUP.		593.81	578.49	554.85	536.82
MEDIA OTRAS (BCD)			556.166		
DESVIA. OTRAS (BCD)			21.12		
MEDIA TODAS (ABCD)		565.45			
DESVIA. TODAS (ABCD)		25.34			

De igual forma se procedió con la obtención de los retardos promedios. En este caso para efectos demostrativos se tomó el escenario 5, con la estrategia de no aplicar QoS (Modo 1). Las 12 estaciones se etiquetaron como: A, B, C, D, E, F, G, H, I, J, K y L.

Tabla B.2: Procedimiento cálculo de los promedios del retardo

Escenario	Iteración	Rendimientos (Kbps)											
#	#	A	B	C	D	E	F	G	H	I	J	K	L
1: MODO 4	1	0.003	0.0025	0.0025	0.0053	0.0025	0.0029	0.003	0.0026	0.0029	0.00253	0.0026	0.00337
1: MODO 4	2	0.0026	0.0027	0.0033	0.003	0.0029	0.0057	0.0028	0.0027	0.0025	0.00266	0.00565	0.00258
1: MODO 4	3	0.0029	0.0026	0.0026	0.0032	0.0026	0.0054	0.003	0.0049	0.0026	0.00364	0.00233	0.00235
5: MODO 1	4	0.0026	0.0031	0.0025	0.0027	0.0026	0.0026	0.0025	0.0038	0.0048	0.00327	0.00311	0.00248
5: MODO 1	5	0.0027	0.0025	0.0026	0.0026	0.0026	0.0025	0.0025	0.0025	0.0055	0.0069	0.00257	0.00249
5: MODO 1	6	0.0032	0.0028	0.0026	0.0025	0.0025	0.0025	0.0025	0.0025	0.0039	0.00255	0.00253	0.00248
5: MODO 1	7	0.0027	0.003	0.0027	0.0027	0.0029	0.0024	0.0026	0.0026	0.0025	0.00306	0.00247	0.00232
5: MODO 1	8	0.0025	0.0025	0.0025	0.0026	0.0025	0.0032	0.0025	0.0046	0.0033	0.00232	0.00237	0.00244
5: MODO 1	9	0.0026	0.0026	0.0028	0.0025	0.0027	0.0025	0.0027	0.0026	0.0026	0.00466	0.00263	0.00255
5: MODO 1	10	0.0026	0.0026	0.0028	0.0051	0.0026	0.0029	0.0026	0.0044	0.0031	0.0027	0.00276	0.00256
5: MODO 1	11	0.0032	0.0027	0.0026	0.0033	0.005	0.0035	0.0028	0.0026	0.0059	0.0042	0.00267	0.00288
5: MODO 1	12	0.0028	0.0026	0.0026	0.0025	0.0025	0.0025	0.0025	0.0027	0.0025	0.00422	0.00258	0.00298
5: MODO 1	13	0.0042	0.0025	0.0056	0.0034	0.0028	0.0029	0.0026	0.0044	0.0031	0.00256	0.00248	0.00252
5: MODO 1	14	0.0045	0.0026	0.0026	0.003	0.003	0.0026	0.0026	0.0044	0.0026	0.00222	0.00225	0.00388
5: MODO 1	15	0.0031	0.0028	0.0025	0.0026	0.0026	0.003	0.0028	0.0025	0.0026	0.00354	0.00276	0.00408
5: MODO 1	16	0.0026	0.0025	0.003	0.0025	0.0025	0.004	0.0025	0.004	0.0025	0.00439	0.00429	0.00244
5: MODO 1	17	0.0032	0.0048	0.0026	0.0033	0.0026	0.0026	0.0025	0.0055	0.0025	0.00318	0.00251	0.00641
5: MODO 1	18	0.0027	0.0029	0.0048	0.0026	0.0026	0.0026	0.003	0.0026	0.0028	0.00243	0.00245	0.00405
5: MODO 1	19	0.0027	0.0028	0.003	0.0029	0.0036	0.0049	0.0028	0.0025	0.0026	0.0024	0.00255	0.00284
5: MODO 1	20	0.0027	0.0027	0.0027	0.0031	0.0037	NO Tx	0.0026	0.0026	0.0026	NO Tx	0.00236	0.00378
PROMEDIO		0.003	0.0028	0.0029	0.0031	0.0029	0.0032	0.0027	0.0033	0.0032	0.00334	0.00279	0.00307
DESVIACIÓN		0.0005	0.0005	0.0008	0.0008	0.0006	0.001	0.0002	0.001	0.001	0.00116	0.0008	0.00099
INTER. CONF		0.0003	0.0002	0.0004	0.0004	0.0003	0.0005	8E-05	0.0005	0.0005	0.00056	0.00037	0.00046
I.C. LIM. INF.		0.0027	0.0026	0.0026	0.0027	0.0026	0.0027	0.0026	0.0029	0.0027	0.00278	0.00242	0.00261
I. C. LIM. SUP.		0.0032	0.003	0.0033	0.0034	0.0031	0.0037	0.0027	0.0038	0.0037	0.00389	0.00317	0.00353
MEDIA OTRAS (B...L)			0.00302395										
DESVIA. OTRAS (B...L)			0.0002313										
MEDIA TODAS (A...L)		0.00301887											
DESVIA. TODAS (A...L)		0.00022123											

El resumen de los cálculos del promedio de los rendimientos y retardos para los otros escenarios se puede encontrar en el Cap´ıtulo 4.

Apéndice C

Detalles de implementación

Esta sección mostrará algunos de las funciones de mayor importancia usados en la simulación y emulación de los escenarios experimentales. Los programas completos se adjuntarán en formato digital de este documento para posteriores usos.

C.1. Función generadora de tráfico para EDCA

El siguiente fragmento de código muestra la función generadora iterativa de tráfico con tiempo de llegada entre paquetes con distribución exponencial, necesaria para emular el comportamiento del flujo transmitido en un entorno de red inalámbrica IEEE 802.11.

```
static void GenerateTraffic (Ptr<Socket> socket, uint32_t pktSize,
    Ptr<ExponentialRandomVariable> NextArrival)
{
    socket->Send (Create<Packet> (pktSize));
    double interPacketInterval = NextArrival->GetValue ();
    Simulator::Schedule (Seconds(interPacketInterval), &GenerateTraffic,
    socket, pktSize, NextArrival);
}
```

Función en C++ sobre NS-3 para generación de tráfico con tiempo de llegada entre paquetes con distribución exponencial

```
void SimulaEdca::Poisson()
{
    uint32_t sourceNode, dest_port, stations;
    stations=m_xSize-1;
    double mean = 1/(lambda/stations); //Seconds/packet
```

```
std::cout<< "Arrival rate: " << lambda << " packets/second " <<std::endl;
std::cout<< "E(x): " << 1/lambda << " seconds/packet " <<std::endl;
std::cout<< "Packet Size: "<< m_packetSize << " bytes"<<std::endl;
Ptr<ExponentialRandomVariable> PoissonTraffic =
CreateObject<ExponentialRandomVariable> ();
PoissonTraffic->SetAttribute ("Mean", DoubleValue (mean));
sourceNode=0;
Ptr<UniformRandomVariable> deltastart =
CreateObject<UniformRandomVariable>  ();
deltastart->SetAttribute ("Min", DoubleValue (1.0));
deltastart->SetAttribute ("Max", DoubleValue (stations));

Ptr<Socket> recvSink[stations]; // Define array of sockets
Ptr<Socket> source;
Ipv4Address DestAddress,SrcAddress ;
//Get the unique id of the factory class that can create udp sockets
TypeId tid = TypeId::LookupByName ("ns3::UdpSocketFactory"); //UDP
for (uint32_t sinkNode = 0; sinkNode < stations; sinkNode++) {
    dest_port = 6100;
    DestAddress = stainterfaces.GetAddress(sinkNode);    //Address of
destination node

    //Bind the socket we just created to the 'any' 0.0.0.0 ipv4 address
    recvSink[sinkNode] = Socket::CreateSocket (nodes.Get (sinkNode), tid);
    InetSocketAddress local = InetSocketAddress(Ipv4Address::GetAny (),
dest_port );
    recvSink[sinkNode]->Bind (local);
    //Create a udp socket on the sending node the ap
    source = Socket::CreateSocket (ap.Get (sourceNode), tid);
    //Connect the udp socket to the ip address and port number of the udp
socket that was created on the sink node.
    //This just sets the default "to" ip address for packets that are sent
over this socket
    InetSocketAddress remote = InetSocketAddress
(stainterfaces.GetAddress(sinkNode), dest_port );
    if (sinkNode == 0)
    { remote.SetTos (0xC0); //AC_V0 : 192
    }
    if (sinkNode == 1)
    { remote.SetTos (0xb8); //AC_VI :  184
    }
    if (sinkNode == 2)
    {  remote.SetTos (0x68); //AC_BE : 104
    }
```

```
        if (sinkNode == 3)
        {  remote.SetTos (0x38); //AC_BG:56
        }

        source->Connect (remote);
        Simulator::Schedule (Seconds(0.0 + deltastart->GetValue()),
    &GenerateTraffic, source, m_packetSize, PoissonTraffic);
    }
}
```

Funciones en C++ sobre NS-3 para generación de tráfico Poisson

C.2. Generador de topología de red inalámbrica emulada

El siguiente *script* muestra los pasos para crear la topología de la red integrada entre NS-3 y SDN, crear los *Linux network namespaces*, instala y ejecuta el servidor de *VLC Player*, ejecuta el controlador e invoca a los *subscript* de ejecución de las aplicaciones (en el cliente) y servidores de video.

```
#!/bin/bash
#Test Openflow with NS-3 emulated mode.
#Run: sudo bash netns-ns3-edca.sh
# Only Node 1 is ovswitch: AP node
COLLECTOR_IP="127.0.0.1"
COLLECTOR_PORT="6343"
AGENT="switch1"
HEADER=128
SAMPLING=512
POLLING=10
BRIDGE_NAME="switch1"
:> /home/alvaro/pox/pox/forwarding/estaciones.dat
N=$1  # Nodes including AP
TIME=60 # Emulation time
NS3_FOLDER=/home/alvaro/ns3/ns-allinone-3.26/ns-3.26
rm -f $NS3_FOLDER/edca-running.dat
bash create-taps.sh $N
Nestaciones=$((N-1))
# Run netns in Background
IP=10.1.1
# start netns after  NS-3
(
   echo "Waiting NS3..."
```

```
while [ ! -e $NS3_FOLDER/edca-running.dat ]; do # A flag indicating that
 the simulation has taken the taps
# Sleep until file does exists/is created
sleep 1
done
echo "NS3 running. Running containers."
echo "Create namespaces"
for ((i=1; i <= N ; i++))
do
   ip netns add "node-$i"
   ip netns exec "node-$i" ip link set dev lo up
   MAC=`printf "%0.2x\n" $i` #Convert to HEX
   if [[ $i -eq 1 ]]; then  # -eq equal to
      echo "Node $i"
      # Remove switch
      ovs-vsctl del-br switch$i
       ovs-vsctl add-br switch$i # create a bridge
       # ovs-vsctl set bridge switch$i protocols=OpenFlow10, OpenFlow13 #
 Try modify the Openflow version for bridge but POX doesn't support
       ovs-vsctl set-fail-mode switch$i secure  # If fail-mode set to
 secure, ovs-vswitchd will not set up flows nor clear all flows
       ip link add ntap1 type veth peer name ovs-tap1
         ip link set ntap1 netns "node-$i"
      ovs-vsctl add-port switch$i ovs-tap1
      ip netns exec "node-$i" ifconfig ntap1 down
      ip netns exec "node-$i" ifconfig ntap1 hw ether "00:00:00:00:00:$MAC"
      ip netns exec "node-$i" ifconfig ntap1 $IP.$i/24
      ip netns exec "node-$i" ifconfig ntap1 up
      ip netns exec "node-$i" ifconfig
      ovs-vsctl add-port switch$i tap$i
       ovs-vsctl show
   else
      ip link set tap$i netns "node-$i"
      ip netns exec "node-$i" ifconfig tap$i hw ether "00:00:00:00:00:$MAC"
      ip netns exec "node-$i" ifconfig tap$i $IP.$i/24 #ip addr add
 10.1.1.$i/24 dev tap$i
      ip netns exec "node-$i" ip link set tap$i up
      ip netns exec "node-$i" ifconfig tap$i
   fi
done
echo "Run controller:"
# AP is switch1
ovs-vsctl set-controller switch1 tcp:127.0.0.1:6633
echo "Network Namespaces created"
```

```
    ip netns list
    echo "Done."
    echo "Terminate simulation with Ctrl+C, (simulation time=$TIME)"
    for((i=2;i<=$N;i++))
    do
    echo "ping del nodo $i"
    ip netns exec node-$i ping -c2 10.1.1.1
    done
    ovs-ofctl --more dump-flows switch1
    #Run video server
    for((i=2;i<=$N;i++))
    do
     #nodeip ="10.1.1.$i"
     sleep 2
     ip netns exec node-1 bash vlc-server.sh $i  #
     ip netns exec node-$i bash vlc-client.sh
    done
)&
pushd $NS3_FOLDER
pwd
# NS3 simulation creates a file: touch wmn_running.dat: this a flag indicating
    that the simulation has taken the taps.
# Then the namespaces start without problem.
./waf --run "SimulaEdca --x_size=$N --time=$TIME --pcap=1 --RealTime=1"
    if [ `echo $?` -ne 0 ]; then    # Check last command error condition (in
    this case waf)
        echo "NS-3  error. Exit. " #CTtrl+C cancel the simulation and
    terminates the program.
    fi
popd
# Remove namespaces
for ((i=1; i <= N ; i++))
    do
    echo "Deletenamespaces"
    ip netns del "node-$i"
done
ifconfig "tap1" down
tunctl -d "tap1"
ifconfig "ovs-tap1"down
ip link delete ovs-tap1
killall vlc
```

Segmento código bash para crear la topología de red emulada

C.3. Aplicación control de admisión

```
"""
Based on:
A simple POX controller application used to gather different parameters from
    the network topology.
https://github.com/hfskappel/ext/blob/master/Statistics.py
https://github.com/hip2b2/poxstuff/blob/master/flow_stats.py
https://github.com/TUDelftNAS/SDN-OpenNetMon/blob/master/monitoring.py
"""
from pox.lib.addresses import IPAddr
from pox.lib.addresses import EthAddr
from pox.core import core
import pox.openflow.libopenflow_01 as of
from pox.lib.util import dpidToStr
from pox.lib.recoco import Timer
from pox.lib.packet.ipv4 import ipv4
from pox.lib.packet.ethernet import ethernet
import pox.lib.packet as pkt
import csv
from collections import defaultdict
prev_stats = defaultdict(lambda:defaultdict(lambda:None))
log = core.getLogger()
ETH_IPV4 = 0x0800
ETH_ARP = 0x0806
IP_PROTO_UDP = 0x11
IP_PROTO_TCP = 0x06
VLC_PORT=1234
TOS=128 #128
Throsat=[0, 4962000, 4626100, 4332800, 4107700, 3930400, 3786600, 3666900,
    3565400, 3477800] # Valores del modelo en bps
PATH = '/home/alvaro/pox/pox/misc/'
with open(PATH+'numno.dat') as f1:
   data1 = f1.readlines()
N=int(data1[0])
f1.close()
waitime = [0]*N
listips = []
for i in range(N):
   listips.append('10.1.1.'+str(i+2))
nodes = []
def _handle_ConnectionUp(event):
    nodes.append(event)
```

```
    print "Switch discovered with DIPID: ", event.connection.dpid
def _handle_flowstats(event):
   # Handle flow stats
    dpid = event.connection.dpid
    port_bytes = 0
    port_flows = 0
    port_packet = 0
    tot_throug = 0
    global waitime
    global listips
    for f in event.stats:
        ip = f.match.nw_dst
        if f.match.tp_dst == VLC_PORT and f.match.dl_type==ETH_IPV4 and
    f.match.nw_proto == IP_PROTO_UDP:
            if ip not in prev_stats or dpid not in prev_stats[ip]:
                prev_stats[ip][dpid] = 0, 0, 0
            prev_byte_count, prev_duration_sec, prev_duration_nsec =
    prev_stats[ip][dpid]
            delta_byte_count = f.byte_count - prev_byte_count
            delta_duration_sec = f.duration_sec - prev_duration_sec
            delta_duration_nsec = f.duration_nsec - prev_duration_nsec
            if delta_byte_count != 0:
                port_bytes = f.byte_count
                port_packet = f.packet_count
                port_flows += 1
                tot_throug += 8*delta_byte_count / (delta_duration_sec +
    (delta_duration_nsec / 1000000000.0))
                log.info("traffic from %s to %s: %s bytes (%s packets) over
    %s flows, qos %s", f.match.nw_src, f.match.nw_dst, port_bytes,
    port_packet, port_flows, f.match.nw_tos)
                prev_stats[ip][dpid] = f.byte_count, f.duration_sec,
    f.duration_nsec
                if  tot_throug >= Throsat[N/2]:
                    for s in range(N):
                        if ip == listips[s]:
                           waitime[s]+=1
                if  tot_throug >= Throsat[N/2] and (wt for wt in waitime)>=3:
                    waitime=[0]*N
                    if str(ip) not in ip2fire:
                        ip2fire.append(str(ip))
    fi=open('/home/alvaro/pox/pox/forwarding/estaciones.dat','r') #Read active
    stations
                    lista=fi.readlines()
                    fi.close()
```

```
                    if len(lista)>0:
                        sock=lista[-1].rstrip('\n')
    ip_destino=sock.split('-')[0].split('.')[3]
                        print "Se va a bloquer la ip "+ip_destino+ " de
    "+str(len(lista))
                       lista.pop()
    fi=open('/home/alvaro/pox/pox/forwarding/estaciones.dat','w')
                        for i in lista:
                            fi.write(i)
                        fi.close()
                        msg = of.ofp_flow_mod()
    msg.actions.append(of.ofp_action_output(port=of.OFPP_NONE))
                        match = of.ofp_match()
                        msg.match.priority = 65535
                        msg.idle_timeout = 60
                        match.nw_src = IPAddr('10.1.1.1')
                        match.nw_dst = IPAddr(ipblock)
                        msg.match = match
                        event.connection.send(msg)
                        print ("Install rule to "+ipblock)
                else:
                    log.info(" ")
            log.info("Throughput acc: %s ", tot_throug)
def _on_timer():
    for n in nodes:
    n.connection.send(of.ofp_stats_request(body=of.ofp_flow_stats_request()))
def launch():
   from pox.openflow.discovery import launch
   launch()
   core.openflow.addListenerByName("ConnectionUp", _handle_ConnectionUp)
   core.openflow.addListenerByName("FlowStatsReceived",_handle_flowstats)
   Timer(1, _on_timer, recurring=True)
```

Segmento código python para estadística del rendimiento y bloqueo de estaciones

C.4. Captura orden de llegada de las estaciones con POX

```
  def _handle_PacketIn (self, event):
    dpid = event.connection.dpid
    inport = event.port
    packet =event.parsed
    if not packet.parsed:
```

```
    log.warning(" %i  %i ignoring unparsed packet", dpid, inport)
    return
  datos = event.parsed.find('udp')
  datst=str(datos)
  if datst != 'None':
    mipto=datst.split(">")[1].split(" ")[0] #Get comunications port
    estacionIP=packet.next.dstip
    if str(estacionIP) !='10.1.1.1':
      sock=str(estacionIP)+'-'+str(mipto)+'\n'
      f=open('/home/alvaro/pox/pox/forwarding/estaciones.dat','r') #lRead
blocked stations
      VideCon=f.readlines()
      f.close()
      if sock not in VideCon:
        ar=open('/home/alvaro/pox/pox/forwarding/estaciones.dat','a')
        ar.write(sock)
        print VideCon
  #if (type(datos) is pox.lib.packet.udp.udp):
  #print datos.split('>')
  if dpid not in self.arpTable:
    # New switch -- create an empty table
    self.arpTable[dpid] = {}
    for fake in self.fakeways:
      self.arpTable[dpid][IPAddr(fake)] = Entry(of.OFPP_NONE,
      dpid_to_mac(dpid))
  if packet.type == ethernet.LLDP_TYPE:
    # Ignore LLDP packets
    return
  if isinstance(packet.next, ipv4):
    log.debug(" %i %i IP %s => %s", dpid,inport,
              packet.next.srcip,packet.next.dstip)
    # Send any waiting packets...
    self._send_lost_buffers(dpid, packet.next.srcip, packet.src, inport)
    # Learn or update port/MAC info
    if packet.next.srcip in self.arpTable[dpid]:
      if self.arpTable[dpid][packet.next.srcip] != (inport, packet.src):
        log.info(" %i %i RE-learned %s", dpid,inport,packet.next.srcip)
    else:
      log.debug(" %i %i learned %s", dpid,inport,str(packet.next.srcip))
    self.arpTable[dpid][packet.next.srcip] = Entry(inport, packet.src)
    # Try to forward
    dstaddr = packet.next.dstip
    if dstaddr in self.arpTable[dpid]:
      # We have info about what port to send it out on...
```

```
        prt = self.arpTable[dpid][dstaddr].port
        mac = self.arpTable[dpid][dstaddr].mac
        if prt == inport:
          log.warning(" %i %i not sending packet for %s back out of the " +
                      "input port" % (dpid, inport, str(dstaddr)))
        else:
          log.debug(" %i %i installing flow for %s => %s outport %i"
                    % (dpid, inport, packet.next.srcip, dstaddr, prt))
          actions = []
          actions.append(of.ofp_action_dl_addr.set_dst(mac))
          actions.append(of.ofp_action_output(port = prt))
          match = of.ofp_match.from_packet(packet, inport)
          match.dl_src = None # Wildcard source MAC
          msg = of.ofp_flow_mod(command=of.OFPFC_ADD,
                                idle_timeout=FLOW_IDLE_TIMEOUT,
                                hard_timeout=of.OFP_FLOW_PERMANENT,
                                buffer_id=event.ofp.buffer_id,
                                actions=actions,
                                match=of.ofp_match.from_packet(packet,inport))
          event.connection.send(msg.pack())
      elif self.arp_for_unknowns:
...
```

Modificación función en l3_learning.py de POX

Los programas completos desarrollados para la simulación y la emulación se adjuntarán en formato digital a la base de datos bibliográfica de la Universidad de Antioquia.

Bibliografía

[1] Deyun Gao, Jianfei Cai, and King Ngi Ngan. Admission control in ieee 802.11e wireless lans. *IEEE network*, 19(4):6–13, 2005.

[2] Nada Chendeb Taher. *Modélisation analytique et contrôle d'admission dans les réseaux 802.11e pour une maîtrise de la Qualité de Service*. PhD thesis, Université d'Evry-Val d'Essonne, 2009.

[3] Nada Chendeb Taher, Yacine Ghamri Doudane, Bachar El Hassan, and Nazim Agoulmine. An accurate analytical model for 802.11e edca under different traffic conditions with contention-free bursting. *Journal of Computer Networks and Communications*, 2011:24, 2011.

[4] Aqsa Malik, Junaid Qadir, Basharat Ahmad, Kok-Lim Alvin Yau, and Ubaid Ullah. Qos in ieee 802.11-based wireless networks: a contemporary review. *Journal of Network and Computer Applications*, 55:24–46, 2015.

[5] Hassan Aboubakr Omar, Khadige Abboud, Nan Cheng, Kamal Rahimi Malekshan, Amila Tharaperiya Gamage, and Weihua Zhuang. A survey on high efficiency wireless local area networks: Next generation wifi. *IEEE Communications Surveys & Tutorials*, 18(4):2315–2344, 2016.

[6] Juan Pablo Urrea Duque. *Performance evaluation model of streaming video in wireless mesh networks*. PhD dissertation, Universidad de Antioquia, 2016.

[7] Daqing Gu and Jinyun Zhang. Qos enhancement in ieee 802.11 wireless local area networks. *IEEE Communications Magazine*, 41(6):120–124, 2003.

[8] Nada Chendeb Taher, Yacine Ghamri Doudane, Bachar El Hassan, and Nazim Agoulmine. Towards voice/video application support in 802.11e wlans: A model-based admission control algorithm. *Computer Communications*, 39:41–53, 2014.

[9] Wi-Fi Becomes Mission Critical. Cisco cleanair technology: Intelligence in action. 2014.

[10] Hewlett Packard Enterprise Development LP. Aruba airmatch technology. URL https://www.arubanetworks.com/assets/tg/TB_AirMatch.pdf.

[11] Tao Lei, Zhaoming Lu, Xiangming Wen, Xing Zhao, and Luhan Wang. Swan: An sdn based campus wlan framework. In *Wireless Communications, Vehicular Technology, Information Theory and Aerospace & Electronic Systems (VITAE), 2014 4th International Conference on*, pages 1–5. IEEE, 2014.

[12] Diego Kreutz, Fernando MV Ramos, Paulo Esteves Verissimo, Christian Esteve Rothenberg, Siamak Azodolmolky, and Steve Uhlig. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76, 2015.

[13] Shiva Rowshanrad, Sahar Namvarasl, and Manijeh Keshtgari. A queue monitoring system in openflow software defined networks. *Journal of Telecommunications and Information Technology*, 2017.

[14] Luca Boero, Marco Cello, Chiara Garibotto, Mario Marchese, and Maurizio Mongelli. Beaqos: Load balancing and deadline management of queues in an openflow sdn switch. *Computer Networks*, 106:161–170, 2016.

[15] Martin Casado, Nate Foster, and Arjun Guha. Abstractions for software-defined networks. *Communications of the ACM*, 57(10):86–95, 2014.

[16] Murat Karakus and Arjan Durresi. Quality of service (qos) in software defined networking (sdn): A survey. *Journal of Network and Computer Applications*, 80: 200–218, 2017.

[17] Danda B Rawat and Swetha Reddy. Recent advances on software defined wireless networking. In *SoutheastCon, 2016*, pages 1–8. IEEE, 2016.

[18] Lalith Suresh, Julius Schulz-Zander, Ruben Merz, Anja Feldmann, and Teresa Vazao. Towards programmable enterprise wlans with odin. In *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, HotSDN '12, pages 115–120, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1477-0. doi: 10.1145/2342441.2342465. URL http://doi.acm.org/10.1145/2342441.2342465.

[19] Kun Wang, Yihui Wang, Deze Zeng, and Song Guo. An sdn-based architecture for next-generation wireless networks. *IEEE Wireless Communications*, 24(1):25–31, 2017.

[20] August Betzler, Ferran Quer, Daniel Camps-Mur, Ilker Demirkol, and Eduard Garcia-Villegas. On the benefits of wireless sdn in networks of constrained edge devices. In *Networks and Communications (EuCNC), 2016 European Conference on*, pages 37–41. IEEE, 2016.

[21] Mehran Abolhasan, Justin Lipman, Wei Ni, and Brett Hagelstein. Software-defined wireless networking: centralized, distributed, or hybrid? *IEEE Network*, 29(4):32–38, 2015.

[22] C Yu Hans, Giorgio Quer, and Ramesh R Rao. Wireless sdn mobile ad hoc network: From theory to practice. In *Communications (ICC), 2017 IEEE International Conference on*, pages 1–7. IEEE, 2017.

[23] P. Dely, J. Vestin, A. Kassler, N. Bayer, H. Einsiedler, and C. Peylo. Cloudmac; an openflow based architecture for 802.11 mac layer processing in the cloud. In *2012 IEEE Globecom Workshops*, pages 186–191, Dec 2012. doi: 10.1109/GLOCOMW.2012.6477567.

[24] Roberto Riggio, Cigdem Sengul, Lalith Suresh, Julius Schulz-Zander, and Anja Feldmann. Thor: Energy programmable wifi networks. In *Computer Communications Workshops (INFOCOM WKSHPS), 2013 IEEE Conference on*, pages 21–22. IEEE, 2013.

[25] Julius Schulz-Zander, Carlos Mayer, Bogdan Ciobotaru, Stefan Schmid, and Anja Feldmann. Opensdwn: programmatic control over home and enterprise wifi. In *Proceedings of the 1st ACM SIGCOMM symposium on software defined networking research*, page 16. ACM, 2015.

[26] Ramon R Fontes, Samira Afzal, Samuel HB Brito, Mateus AS Santos, and Christian Esteve Rothenberg. Mininet-wifi: Emulating software-defined wireless networks. In *Network and Service Management (CNSM), 2015 11th International Conference on*, pages 384–389. IEEE, 2015.

[27] Roberto Riggio, Mahesh K Marina, Julius Schulz-Zander, Slawomir Kuklinski, and Tinku Rasheed. Programming abstractions for software-defined wireless networks. *IEEE Transactions on Network and Service Management*, 12(2):146–162, 2015.

[28] Emna Charfi, Lamia Chaari, and Lotfi Kamoun. Phy/mac enhancements and qos mechanisms for very high throughput wlans: A survey. *IEEE Communications Surveys and Tutorials*, 15(4):1714–1735, 2013.

[29] Katarzyna Kosek-Szott, Marek Natkaniec, Szymon Szott, Artem Krasilov, Andrey Lyakhov, Alexander Safonov, and Ilenia Tinnirello. What's new for qos in ieee 802.11? *IEEE Network*, 27(6):95–104, 2013.

[30] Robson Costa, Paulo Portugal, Francisco Vasques, Carlos Montez, and Ricardo Moraes. Limitations of the ieee 802.11 dcf, pcf, edca and hcca to handle real-time traffic. In *Industrial Informatics (INDIN), 2015 IEEE 13th International Conference on*, pages 931–936. IEEE, 2015.

[31] Chang Wen Chen and FENG Zhengyong. Video over ieee802. 11 wireless lan: A brief survey. *China Communications*, 10(5):1–19, 2013.

[32] Elena Agostini, Massimo Bernaschi, Massimo Vellucci, and Luca Vollero. Opencapwap v2. 0: the new open-source implementation of the capwap protocol. *International Journal of Network Management*, 26(6):537–552, 2016.

[33] Dorothy Stanley, Pat Calhoun, and Michael Montemurro. Control and provisioning of wireless access points (capwap) protocol specification. 2009.

[34] Anyfi Networks AB. Software-defined wireless networking: Concepts, principles and motivations, 2014. URL http://static.anyfinetworks.com/anyfi-sdwn-concepts-wp-20140514.pdf.

[35] IEEE. Standards process, 2018. URL https://www.ietf.org/standards/process/.

[36] Boris Bellalta, Luciano Bononi, Raffaele Bruno, and Andreas Kassler. Next generation ieee 802.11 wireless local area networks: Current status, future directions and open challenges. *Computer Communications*, 75:1–25, 2016.

[37] Mario Marques Freire; Manuela Periera. *Encyclopedia of Internet technologies and applications*. Gale virtual reference library. Information Science Reference, 2008. ISBN 9781591409946,1591409942,9781591409939,1591409934.

[38] Fernanda Rodríguez. Análisis del desempeño de redes 802.11. Master's thesis, Universidad de la República, Montevideo, 11 2015.

[39] Emna Charfi, Cédric Gueguen, Lamia Chaari, Bernad Cousin, and Lotfi Kamoun. Dynamic frame aggregation scheduler for multimedia applications in ieee 802.11n networks. *Transactions on Emerging Telecommunications Technologies*, 28(2):e2942–n/a, 2017. ISSN 2161-3915. doi: 10.1002/ett.2942. URL http://dx.doi.org/10.1002/ett.2942. e2942 ett.2942.

[40] IETF Network Working Group et al. Rfc 1633 integrated services in the internet architecture: An overview. *IETF*, 1994.

[41] Dan Grossman. New terminology and clarifications for diffserv. Technical report, 2002.

[42] Sijo Joy and Amiya Nayak. Improving flow completion time for short flows in datacenter networks. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 700–705. IEEE, 2015.

[43] Ahmad Reza Montazerolghaem, Mohammad Hossein Yaghmaee, Alberto Leon-Garcia, Mahmoud Naghibzadeh, and Farzad Tashtarian. A load-balanced call admission controller for ims cloud computing. *IEEE Trans. Network and Service Management*, 13(4):806–822, 2016.

[44] Conroy Smith. *An admission control scheme for IEEE 802.11e wireless local area networks*. PhD thesis, University of Cape Town, 2008.

[45] Orlie T Brewer and Arun Ayyagari. Comparison and analysis of measurement and parameter based admission control methods for quality of service (qos) provisioning. In *Military Communications Conference, 2010-Milcom 2010*, pages 184–188. IEEE, 2010.

[46] Jeonghoon Mo. Performance modeling of communication networks with markov chains. *Synthesis Lectures on Data Management*, 3(1):1–90, 2010.

[47] Nick Feamster, Jennifer Rexford, and Ellen Zegura. The road to sdn: an intellectual history of programmable networks. *ACM SIGCOMM Computer Communication Review*, 44(2):87–98, 2014.

[48] Natasha Gude, Teemu Koponen, Justin Pettit, Ben Pfaff, Mart´ın Casado, Nick McKeown, and Scott Shenker. Nox: towards an operating system for networks. *ACM SIGCOMM Computer Communication Review*, 38(3):105–110, 2008.

[49] McCauley et al. Pox controller, 2015. URL https://noxrepo.github.io/pox-doc/html/.

[50] Floodlight Project. Floodlight, 2018. URL http://www.projectfloodlight.org/.

[51] The OpenDaylight Project, Inc. OpenDaylight, 2018. URL http://www.opendaylight.org/.

[52] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. Openflow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74, 2008.

[53] Wei Zhou, Li Li, Min Luo, and Wu Chou. Rest api design patterns for sdn northbound api. In *Advanced Information Networking and Applications Workshops (WAINA), 2014 28th International Conference on*, pages 358–365. IEEE, 2014.

[54] Behnam Dezfouli, Vahid Esmaeelzadeh, Jaykumar Sheth, and Marjan Radi. A review of software-defined wlans: Architectures and central control mechanisms. *IEEE Communications Surveys & Tutorials*, 2018.

[55] Dong Jin. *Network-simulation-based evaluation of smart grid applications*. PhD thesis, University of Illinois at Urbana-Champaign, 2013.

[56] Carlos J Bernardos, Antonio De La Oliva, Pablo Serrano, Albert Banchs, Luis M Contreras, Hao Jin, and Juan Carlos Zúñiga. An architecture for software defined wireless networking. *IEEE wireless communications*, 21(3):52–61, 2014.

[57] Mao Yang, Yong Li, Depeng Jin, Lieguang Zeng, Xin Wu, and Athanasios V. Vasilakos. Software-defined and virtualized future mobile and wireless networks: A survey. *Mobile Networks and Applications*, 20(1):4–18, Feb 2015. doi: 10.1007/s11036-014-0533-8. URL https://doi.org/10.1007/s11036-014-0533-8.

[58] Fabrizio Granelli, Anteneh A Gebremariam, Muhammad Usman, Filippo Cugini, Veroniki Stamati, Marios Alitska, and Periklis Chatzimisios. Software defined and virtualized wireless access in future wireless networks: scenarios and standards. *IEEE Communications Magazine*, 53(6):26–34, 2015.

[59] Kok-Kiong Yap, Masayoshi Kobayashi, David Underhill, Srinivasan Seetharaman, Peyman Kazemian, and Nick McKeown. The stanford openroads deployment. In *Proceedings of the 4th ACM International Workshop on Experimental Evaluation and Characterization*, WINTECH '09, pages 59–66, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-740-0.

[60] Ahmed Abujoda, David Dietrich, Panagiotis Papadimitriou, and Arjuna Sathiaseelan. Software-defined wireless mesh networks for internet access sharing. *Computer Networks*, 93(Part 2):359 – 372, 2015. ISSN 1389-1286. Community Networks.

[61] Huawei Huang, Peng Li, Song Guo, and Weihua Zhuang. Software-defined wireless mesh networks: architecture and traffic orchestration. *IEEE network*, 29(4):24–30, 2015.

[62] Oliver Hoffmann, Falk-Moritz Schaefer, Rüdiger Kays, Christian Sauer, and Hans-Peter Loeb. Prioritized medium access in ad-hoc networks with a systemclick model of the ieee 802.11n mac. In *Personal Indoor and Mobile Radio Communications (PIMRC), 2010 IEEE 21st International Symposium on*, pages 2805–2810. IEEE, 2010.

[63] Yaw-Wen Kuo, Tsern-Huei Lee, Yu-Wen Huang, and Jing-Rong Hsieh. Design and evaluation of a high throughput mac with qos guarantee for wireless lans. In *Communications (MICC), 2009 IEEE 9th Malaysia International Conference on*, pages 869–873. IEEE, 2009.

[64] Hongli Luo and Mei-Ling Shyu. An optimized scheduling scheme to provide quality of service in 802.11e wireless lan. In *2009 11th IEEE International Symposium on Multimedia*, pages 651–656. IEEE, 2009.

[65] Nitya Sundareswaran, George F Riley, Ken Boyd, and Amit Nainani. Improving quality of service in mac 802.11 layer. In *2007 15th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication systems*, pages 38–45. IEEE, 2007.

[66] Xiaomin Chen, Ibukunoluwa Akinyemi, and Shuang-Hua Yang. A control theoretic approach to achieve proportional fairness in 802.11e edca wlans. *Computer Communications*, 75:39–49, 2016.

[67] Yang Xiao and Haizhon Li. Voice and video transmissions with global data parameter control for the ieee 802.11e enhance distributed channel access. *IEEE Transactions on parallel and distributed systems*, 15(11):1041–1053, 2004.

[68] Anwar Saif and Mohamed Othman. Sra-msdu: Enhanced a-msdu frame aggregation with selective retransmission in 802.11n wireless networks. *Journal of Network and Computer Applications*, 36(4):1219–1229, 2013.

[69] Yi Yao, Bo Sheng, and Ningfang Mi. A new packet scheduling algorithm for access points in crowded wlans. *Ad Hoc Networks*, 36:100–110, 2016.

[70] SM Koli, RG Purandare, SP Kshirsagar, and VV Gohokar. Qos-optimized adaptive multi-layer (oqam) architecture of wireless network for high quality digital video transmission. *Journal of Visual Communication and Image Representation*, 26: 210–221, 2015.

[71] Murat Karakus and Arjan Durresi. Quality of service (qos) in software defined networking (sdn): A survey. *Journal of Network and Computer Applications*, 80:200 – 218, 2017. ISSN 1084-8045. doi: https://doi.org/10.1016/j.jnca.2016.12.019. URL http://www.sciencedirect.com/science/article/pii/S1084804516303186.

[72] Yan Jinyao, Zhang Hailong, Shuai Qianjun, Liu Bo, and Guo Xiao. Hiqos: An sdn-based multipath qos solution. *China Communications*, 12(5):123–133, 2015.

[73] Hilmi E Egilmez, S Tahsin Dane, K Tolga Bagci, and A Murat Tekalp. Openqos: An openflow controller design for multimedia delivery with end-to-end quality of service over software-defined networks. In *Signal & Information processing association annual summit and conference (APSIPA ASC), 2012 Asia-Pacific*, pages 1–8. IEEE, 2012.

[74] M Said Seddiki, Muhammad Shahbaz, Sean Donovan, Sarthak Grover, Miseon Park, Nick Feamster, and Ye-Qiong Song. Flowqos: Qos for the rest of us. In *Proceedings of the third workshop on Hot topics in software defined networking*, pages 207–208. ACM, 2014.

[75] Seyhan Civanlar, Murat Parlakisik, A Murat Tekalp, Burak Gorkemli, Bulent Kaytaz, and Evren Onem. A qos-enabled openflow environment for scalable video streaming. In *GLOBECOM Workshops (GC Wkshps), 2010 IEEE*, pages 351–356. IEEE, 2010.

[76] Hilmi E Egilmez, Seyhan Civanlar, and A Murat Tekalp. An optimization framework for qos-enabled adaptive video streaming over openflow networks. *IEEE Transactions on Multimedia*, 15(3):710–715, 2013.

[77] Airton Ishimori, Fernando Farias, Eduardo Cerqueira, and Antônio Abelém. Control of multiple packet schedulers for improving qos on openflow/sdn networking. In *Software Defined Networks (EWSDN), 2013 Second European Workshop on*, pages 81–86. IEEE, 2013.

[78] Michael Jarschel, Florian Wamser, Thomas Hohn, Thomas Zinner, and Phuoc Tran-Gia. Sdn-based application-aware networking on the example of youtube video streaming. In *Software Defined Networks (EWSDN), 2013 Second European Workshop on*, pages 87–92. IEEE, 2013.

[79] Salvatore Costanzo, Laura Galluccio, Giacomo Morabito, and Sergio Palazzo. Software defined wireless networks: Unbridling sdns. In *Software Defined Networking (EWSDN), 2012 European Workshop on*, pages 1–6. IEEE, 2012.

[80] Peter Dely. *Architectures and Algorithms for Future Wireless Local Area Networks*. PhD thesis, 2012. URL http://urn.kb.se/resolve?urn=urn:nbn:se:kau:diva-15393.

[81] Jonathan Vestin and Andreas Kassler. Qos enabled wifi mac layer processing as an example of a nfv service. In *1st IEEE Conference on Network Softwarization (NetSoft), 13-17 April 2015, London*, pages 1–9. IEEE conference proceedings, 2015.

[82] Lalith Suresh, Julius Schulz-Zander, Ruben Merz, Anja Feldmann, and Teresa Vazao. Towards programmable enterprise wlans with odin. In *Proceedings of the first workshop on Hot topics in software defined networks*, pages 115–120. ACM, 2012.

[83] Jiacheng Chen, Bo Liu, Haibo Zhou, Quan Yu, Lin Gui, and Xuemin Sherman Shen. Qos-driven efficient client association in high-density software-defined wlan. *IEEE Transactions on Vehicular Technology*, 66(8):7372–7383, 2017.

[84] Dong Zhao, Ming Zhu, and Ming Xu. Leveraging sdn and openflow to mitigate interference in enterprise wlan. *Journal of Networks*, 9(6):1526, 2014.

[85] Ming Zhu Dong Zhao and Ming Xu. Leveraging sdn and openflow to mitigate interference in enterprise wlan. In *JOURNAL OF NETWORKS*, volume 9, pages 1526–1533, Jun 2014. doi: 10.4304. URL https://goo.gl/sVvPRd.

[86] Jeongkeun Lee, Mostafa Uddin, Jean Tourrilhes, Souvik Sen, Sujata Banerjee, Manfred Arndt, Kyu-Han Kim, and Tamer Nadeem. mesdn: Mobile extension of sdn. In *Proceedings of the fifth international workshop on Mobile cloud computing & services*, pages 7–14. ACM, 2014.

[87] MATLAB Optimization Toolbox. Matlab fsolve toolbox, 2016.

[88] Giuseppe Bianchi. Performance analysis of the ieee 802.11 distributed coordination function. *IEEE Journal on selected areas in communications*, 18(3):535–547, 2000.

[89] Emad Felemban and Eylem Ekici. Single hop ieee 802.11 dcf analysis revisited: Accurate modeling of channel access delay and throughput for saturated and unsaturated traffic cases. *IEEE Transactions on Wireless Communications*, 10(10): 3256–3266, 2011.

[90] Qinglin Zhao, Danny HK Tsang, and Taka Sakurai. A scalable and accurate non-saturated ieee 802.11e edca model for an arbitrary buffer size. *IEEE Transactions on mobile computing*, 12(12):2455–2469, 2013.

[91] G Prakash and P Thangaraj. Performance comparison of ieee 802.11e edca and 802.11b dcf under non-saturation condition using network simulator. *Research Journal of Applied Sciences, Engineering and Technology*, 4(22):4748–4754, 2012.

[92] National Science Foundation et al. The ns-3 network simulator, 2017. URL https://www.nsnam.org/.

[93] openstack.org. OpenStack Docs: Network namespaces linux network namespaces, 2018. URL https://docs.openstack.org/newton/networking-guide/intro-network-namespaces.html.

[94] George F Riley and Thomas R Henderson. The ns-3 network simulator. In *Modeling and tools for network simulation*, pages 15–34. Springer, 2010.

[95] J. F. G. Orrego and J. P. U. Duque. Throughput and delay evaluation framework integrating sdn and ieee 802.11s wmn. In *2017 IEEE 9th Latin-American Conference on Communications (LATINCOM)*, pages 1–6, Nov 2017. doi: 10.1109/LATINCOM.2017.8240186.

[96] Ben Pfaff, Justin Pettit, Teemu Koponen, Ethan Jackson, Andy Zhou, Jarno Rajahalme, Jesse Gross, Alex Wang, Joe Stringer, Pravin Shelar, et al. The design and implementation of open vswitch. In *12th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 15)*, pages 117–130, 2015.

[97] Stefan Lederer, Christopher Müller, and Christian Timmerer. Dynamic adaptive streaming over http dataset. In *Proceedings of the 3rd Multimedia Systems Conference*, pages 89–94. ACM, 2012.

[98] Alvaro Jiménez, Juan F Botero, and Juan P Urrea. Admission control implementation for qos performance evaluation over sdwn. In *2018 IEEE Colombian Conference on Communications and Computing (COLCOM)*, pages 1–6. IEEE, 2018.