Enabling Adaptive Probing for SDN Monitoring

Edwin Ferney Castillo * Armando Ordonez, and Oscar Mauricio Caicedo Rendon
Department of Telematics, Universidad del Cauca, Popayán, Cauca, Colombia.
e-mail: {efcastillo, jaordonez, omcaicedo}@unicauca.edu.co

***Resumen*—Traffic Monitoring assists in achieving the stability of networks by observing and quantifying their behavior. A proper traffic monitoring solution requires the accurate and timely collection of flow statistics. Many approaches have been proposed to monitor Software-Defined Networks. However, these approaches have diverse shortcomings. First, they are unconcerned about the trade-off between the probing interval and the Monitoring Accuracy (MA). Second, they lack intelligent mechanisms intended to optimize this trade-off by learning from network behavior. This paper introduces an approach, called IPro, to address these shortcomings. Our approach is formed by an architecture that follows the Knowledge-Defined Networking paradigm, an algorithm based on Reinforcement Learning, and an IPro prototype. In particular, IPro uses Reinforcement Learning to determine the probing interval that keeps within thresholds (target values) the Control Channel Overhead (CCO) and the Extra CPU Usage of the Controller (CUC). An extensive quantitative evaluation corroborates that IPro is an efficient approach for SDN Monitoring regarding CCO, CCU, and MA.**

***Keywords*—Knowledge-Defined Networking, Machine Learning, Probing Interval, Software-Defined Networking, Traffic Monitoring**

I. Introducción

Redes Definidas por Software (Software-Defined Networking - SDN) separan el plano de datos de la capa de control, ubican las funciones de control en una entidad de software centralizada (*i.e*, controlador) e introducen interfaces estandarizadas que permiten la programación de los dispositivos de red [1]. SDN permite mejorar y simplificar las tareas de gestión [2], ya que posibilita el desarrollo de aplicaciones que pueden reconfiguran la red automáticamente [3]. No obstante, en escenarios dinámicos las aplicaciones de gestión plantean nuevos requisitos al proceso de monitorización [4], especialmente en términos de frecuencia de medición y estadísticas de tráfico (detalladas, precisas y oportunas), indispensables para tomar decisiones, solucionar fallas, detectar anomalías y proporcionar una vista actualizada del estado de la red.

Para recuperar las estadísticas de trafico de los conmutadores, los controladores SDN utilizan tradicionalmente dos enfoques [5]: basado en inserción (Push-based) y basado en extracción (Pull-based). En Push-based, el controlador observa la información estadística suministrada por un módulo colector sin influir en el rendimiento de la red. Sin embargo, este enfoque tiene algunos inconvenientes en su implementación. Primero, requiere módulos adicionales de hardware y software en los conmutadores, que pueden generar problemas de seguridad [6]. Segundo, cuando el tráfico varía dinámicamente, los conmutadores detectan paquetes no coincidentes en la tabla de flujo y, como resultado, envían informes estadísticos masivos al controlador. Estos informes masivos pueden causar una sobrecarga significativa del canal de control (Control Channel Overhead - CCO) [7] y un uso adicional de CPU del controlador (CPU Usage of the Controller - CUC) [8].

En Pull-based, el controlador sondea uno o todos los conmutadores utilizando mensajes de Estado-Lectura para recuperar estadísticas de flujo. Este enfoque proporciona mayor flexibilidad que Push-based, ya que puede comunicarse de manera asíncrona con los conmutadores y solicitar información específica, controlando de este modo el tamaño de los informes estadísticos. Además, este enfoque no requiere cambios de software y hardware en los conmutadores.

Cabe señalar que, Pull-based introduce CCO, en función del intervalo de sondeo que utiliza el controlador para enviar los mensajes de Estado-Lectura, que puede sobrecargar el controlador e interferir de manera significativa con las funciones esenciales de SDN, tales como el reenvió de paquetes y la actualización de rutas. Se han llevado a cabo varios trabajos de investigación para tratar con el CCO y CUC [8-21]. En particular, algunos trabajos reducen el CCO utilizando técnicas adaptativas, comodines, métodos basados en umbrales e información de enrutamiento a expensas de disminuir la exactitud de la monitorización (Monitoring Accuracy - MA) [9-13]. Otros trabajos disminuyen el CCO agregando módulos y/o modificando las tablas de flujo en los conmutadores [8-14-16] y haciendo uso de controladores distribuidos [17-21]. No obstante, estos trabajos reducen el CCO a expensas de incrementar los costos operativos. Además, es importante recalcar que, en las soluciones Pull-based actuales, no se ha considerado mecanismos inteligentes que optimicen el balance entre el intervalo de sondeo y la MA.

Para superar estas deficiencias, proponemos un enfoque llamado IPro. IPro incluye una arquitectura que sigue el paradigma de Redes definidas por el conocimiento (Knowledge-Defined Networking - KDN), un agente basado en aprendizaje por refuerzo (Reinforcement Learning - RL) y un prototipo IPro. KDN es utilizado para aplicar el aprendizaje automático (Machine Learning - ML) en SDN. En particular, RL permite optimizar el intervalo de sondeo de IPro con el objetivo de mantener CCO y CUC dentro de valores objetivo. Evaluamos el prototipo IPro en un entorno emulado mediante el uso de una topología de red campus. Los resultados de la evaluación revelaron que IPro mantiene el CCO en menos del 1.23 % y el CUC en menos del 8 %. Además, IPro tiene una MA mejor ($\geq$ 90 %) que el enfoque de sondeo periódico (Periodic Probing Approach - PPA), un método Pull-based para monitorizar los conmutadores con un intervalo de sondeo predefinido.

El resto de este documento está organizado de la siguien-

te manera. En la sección II, se presenta los principios de operación de KDN y RL. Después, en la Sección III, se presentan los estudios que han abordado la monitorización en SDN. En la sección IV, se expone IPro. En la sección V, se describe y discute el estudio de caso planteado para evaluar IPro. Finalmente, la conclusión del artículo es presentada en la sección VI.

II. Marco Conceptual

En esta sección se resumen los principios de operación de KDN y RL.

2.1. Redes definidas por conocimiento

KDN impulsa el uso de ML para cambiar la forma en que los operadores de red programan, optimizan y solucionan problemas de SDN [22]. En particular, mediante el uso de técnicas de ML, es posible aprender del comportamiento de la red para proporcionar automatización (reconocer-actuar) y recomendaciones (reconocer-explicar-sugerir) que respalden las tareas de gestión. A su vez, SDN ofrece control total y una vista de toda la red desde un punto lógicamente centralizado (*i.e*, controlador). En este trabajo, argumentamos que RL es una técnica de ML útil para mantener la MA y disminuir el CCO y CUC del proceso de monitorización en SDN dado que permite optimizar el intervalo de sondeo al interactuar con la red (*i.e,* entorno en términos de RL).

2.2. Aprendizaje por refuerzo

En RL, un agente aprende un proceso de toma de decisiones al interactuar con un entorno modelado típicamente como un proceso finito de decisiones de Markov (Markov Decision Processes - MDP) [23], donde el agente envía acciones y recibe resultados (observaciones y recompensas). En un MDP finito, el agente y el entorno interactúan en pasos de tiempo discretos $t = 0, 1, 2, ..., N$. En cada paso de tiempo t, el agente recibe alguna representación del estado del entorno, $S_t \in S$, donde S es el conjunto de estados posibles. Basado en S_t, el agente selecciona una acción, $A_t \in A(S_t)$, donde $A(S_t)$ es el conjunto de acciones disponibles en el estado S_t. La ejecución de la acción A_t pone al agente en el nuevo estado S_{t+1}. Además, el agente recibe una recompensa numérica del entorno, $R_{t+1} \in \mathbb{R}$ en el paso $t \in N$.

Los algoritmos de RL encuentran una política óptima $\Pi^* : S \rightarrow A$ que maximiza la recompensa acumulativa esperada para cada estado, utilizando los métodos de exploración (*e.g*, ε-greedy, Boltzmann [23]). Las principales características de RL son su capacidad para ejecutarse sin ningún conocimiento previo del entorno (aquí, la red monitorizada) y tomar sus propias decisiones en tiempo de ejecución (en línea). No obstante, RL requiere un período de entrenamiento para modelar el entorno antes de converger en la política óptima.

Una de las técnicas mas importante de RL es Q-learning [24] porque, primero, esta técnica fue el método de RL pionero utilizado para fines de control. En segundo lugar, Q-learning tiene una curva de aprendizaje que tiende a converger rápidamente. Tercero, es la técnica más simple que calcula directamente la política de acción óptima sin un paso intermedio de evaluación de costos y sin el uso de un modelo *(i.e.*, sin modelo). Cuarto, tiene una capacidad de aprendizaje fuera de la política; esto significa que el agente puede aprender una política óptima (llamada función-Q [25]), incluso si no siempre elige acciones óptimas. La única condición es que el agente visite y actualice regularmente todos los pares (S_t, A_t). Q-learning comienza con una función-Q arbitraria Q_0. En cualquier estado S_t, un agente selecciona una acción A_t que determina la transición al siguiente estado S_{t+1} y con el valor asociado al par (S_t, A_t) ajusta los valores de la función-Q de acuerdo con:

$$Q_{t+1}(S_t, A_t) \leftarrow (1-\alpha) \cdot Q_t(S_t, A_t) + \alpha \cdot \left[R_{t+1} + \gamma \cdot \max_{\mathrm{A}} Q_t(S_{t+1}, A) \right] \quad (1)$$

donde R_{t+1} denota la recompensa recibida en el momento $t+1$, $\alpha \in [0, 1]$ es el factor de aprendizaje que determina la importancia de la recompensa adquirida. $\gamma \in [0, 1]$ es el factor de descuento que determina la importancia de futuras recompensas. La parte entre corchetes es el valor actualizado que representa la diferencia entre la estimación actual del valor Q óptimo $Q_t(S_t, A_t)$ para un par de estado-acción (S_t, A_t) , y la nueva estimación $\left[R_{t+1} + \gamma \max_{\mathrm{A}} Q_t(S_{t+1}, A) \right]$.

III. Monitorización en SDN

La monitorización de redes es un campo de investigación que atrae mucha atención de la academia y la industria [26]. La Tabla I resume algunos trabajos relevantes en el campo de la monitorización en SDN, revelando varios hechos. Primero, trabajos como [14-21] minimizan el CCO y mejoran la MA agregando módulos en el plano de datos o utilizando controladores distribuidos, lo que ocasiona un incremento en los recursos y costos de la red. Además, dichos trabajos no admiten una monitorización detallado de la red y carecen de flexibilidad y escalabilidad, necesaria para hacer frente a una gran cantidad de flujos. En segundo lugar, otros trabajos, como [9-13] reducen el CCO utilizando técnicas adaptativas, comodines, métodos basados en umbrales e información de enrutamiento a expensas de disminuir la MA (*i.e.*, no existe un balance entre la MA y el CCO).

A pesar del progreso en la monitorización en SDN, los enfoques existentes aún tienen algunas deficiencias. Primero, introducen CCO que degrada el rendimiento de la red o requieren inversiones económicas sustanciales. En segundo lugar, no se ha considerado el balance entre el intervalo de sondeo y la MA, lo que lleva a deteriorar el canal de control o la misma MA. En tercer lugar, no consideran mecanismos inteligentes que optimicen dicho balance al aprender del comportamiento de la red, lo que causa posibles cuellos de botella en el canal de control, pérdida de paquetes/flujo y caídas de rendimiento. En este trabajo, abordamos estas deficiencias siguiendo el paradigma de KDN. En particular, se utiliza RL para optimizar el balance entre el intervalo de sondeo y la MA en SDN mientras se mantiene CCO y CUC dentro de valores objetivo.

Tabla I: Monitorización de trafico en SDN – Alto (↑) y Bajo (↓)

Work	Accuracy	Overhead	Resources-Cost	Flexibility-Scalability	Description
[9]	↑	↑	↓	↑	Utilizan algoritmos de muestreo adaptativo para ajustar el nivel de carga generado por el proceso de monitorización
[10]	↑	↑	↓	↑	Emplean umbrales para ajustar el nivel de carga generado por el proceso de monitorización
[11]	↑	↑	↓	↑	Hacen uso de un mecanismo de sondeo adaptativo para monitorizar el rendimiento, el retraso y la pérdida de paquetes
[12]	↑	↑	↓	↑	Usan un método de recopilación estadística de flujos adaptativos para ajustar los intervalos de sondeo
[13]	↑	↑	↓	↑	Utiliza estadísticas OpenFlow (*e.g.*, bytes y contadores de paquetes) e información de enrutamiento de controlador para monitorizar la utilización del enlace
[14]	↑	↓	↑	↓	Hacen uso de agentes software en los hosts que interactúan con los dispositivos de red para realizar tareas de monitorización
[17]	↑	↓	↑	↓	Utilizan un conjunto de reglas de coincidencia y controladores secundarios para identificar y monitorizar flujos agregados
[15]	↑	↓	↑	↓	Separan la monitorización de las tablas de reenvío existentes y utilizan agentes software personalizados en los conmutadores para procesar su funcionalidad de monitorización
[16]	↑	↓	↑	↓	Inyectan paquetes LLDP con marca de tiempo en los conmutadores para la monitorización de la latencia de la red
[18]	↑	↓	↑	↓	Utilizan administradores locales y entidades para reconfigurar los recursos de la red y apoyar las tareas de monitorización en diferentes niveles de granularidad
[19]	↑	↓	↑	↓	Usan un mecanismo de monitorización de auto-ajuste para adaptar automáticamente su configuración en función del dinamismo del tráfico
[20]	↑	↓	↑	↓	Incluyen módulos adicionales en los conmutadores para distribuir las tareas de monitorización de forma equilibrada
[21]	↑	↓	↑	↓	Describen una jerarquía de dos capas de controladores. La capa más baja sondea la estadística de flujo y envía estadísticas a la capa superior. La capa más alta coordina los controladores ubicados en el nivel más bajo

IV. IPro

En esta sección se presenta la descripción general de IPro, sus elementos arquitectónicos y su funcionamiento.

4.1. Descripción general

IPro utiliza RL para optimizar el intervalo de sondeo con respecto a CCO y CUC. La Figura 1 muestra la operación general de IPro: (1) el Plano de Control (PC) recopila información estadística del Plano de Datos (PD) con algún intervalo de sondeo. Dado que esta recopilación de información afecta el comportamiento de la red, la red cae en un nuevo estado. (2) el Plano de Gestión (PG) extrae estas estadísticas para determinar dicho estado mediante el análisis de CCO y CUC. Posteriormente, en (3), PG envía este nuevo estado al Plano de Conocimiento (PCon). En (4), el agente RL toma dicho estado para calcular la recompensa. Es importante resaltar que una recompensa baja indica CCO y CUC alto. En función de la recompensa, el agente RL decide un nuevo intervalo de sondeo destinado destinado a mantener el CCO y CUC dentro de valores objetivo. (5) el agente RL comunica este nuevo intervalo de sondeo a PC. En (6), el PC aplica este intervalo que afecta nuevamente el comportamiento de la red (*i.e.,* la red cae en un nuevo estado). Esta operación se repite hasta que el agente RL perciba que la política no cambia. En este momento, el agente RL obtiene el intervalo de sondeo más gratificante que mantiene CCO y CUC dentro de los valores objetivo con la finalidad de minimizar la degradación del rendimiento de la red causada por las tareas de monitorización.

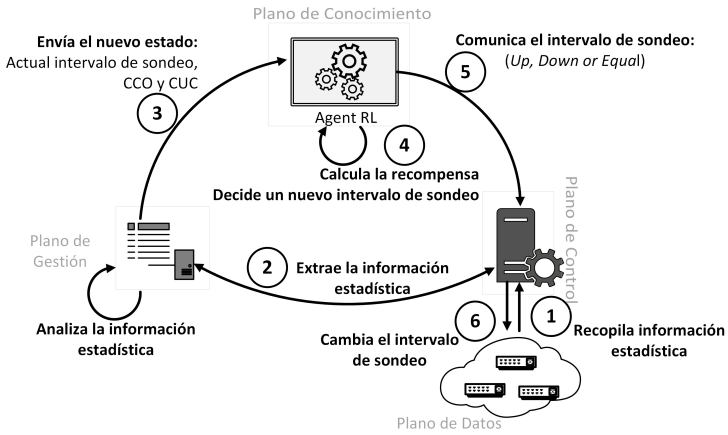

Figura 1. Operación general de IPro

4.2. Capas y Elementos Arquitectónicos

La Figura 2 presenta la arquitectura de IPro. IPro está constituido por cuatro planos que siguen los fundamentos de KDN. A continuación, detallamos cada uno de estos planos.

a) ***Plano de Conocimiento****:* Este plano es responsable de aprender el comportamiento de la red y de decidir automáticamente un nuevo intervalo de sondeo. PCon obtiene el estado actual de la red y controla el intervalo de sondeo interactuando con PG y PC, respectivamente. El corazón de PCon es el agente RL. Este agente está a cargo de determinar la estrategia de sondeo más gratificante destinada a mantener el

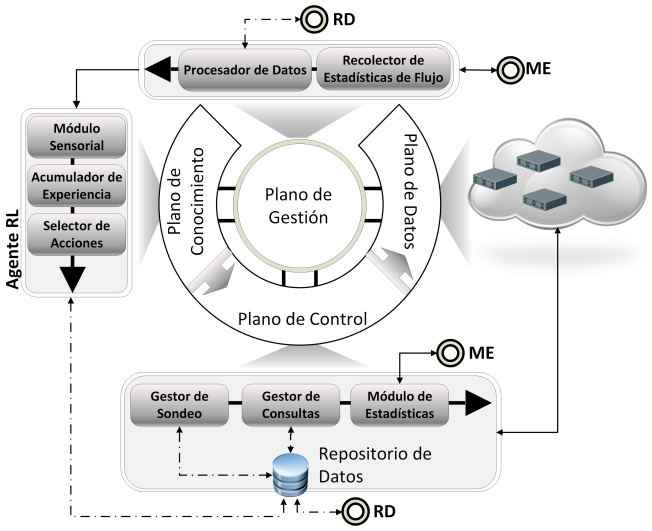

Figura 2. Descripción general de la arquitectura IPro y sus componentes

CCO y CUC dentro de los valores objetivo. IPro considera los umbrales ω y χ como políticas de monitorización configurables de acuerdo con los requisitos de la red. ω representa la política 1 cuyo fin es evitar un CCO excesivo. A su vez, χ representa la política 2 y tiene como objetivo evitar un CUC excesivo. Dicho brevemente, el agente RL maneja las políticas de monitorización controlando ω y χ.

Las políticas de monitorización se definen para evitar que los mensajes de monitorización afecten el rendimiento del controlador y el ancho de banda del canal de control. Por ejemplo, cuando CUC excede el 80 % de la capacidad de la CPU, el controlador aumenta su tiempo de respuesta, lo que genera demoras y pérdidas [27]. En el mismo sentido, cuando el uso del ancho de banda del canal de control excede el 80 %, los mensajes de monitorización pueden interferir con las funciones SDN [28].

La Figura 3 representa el modelo general del agente RL. Este modelo incluye tres elementos, *Módulo Sensorial*, *Acumulador de Experiencia* y *Selector de Acciones*. El *Módulo Sensorial* permite al agente RL obtener el estado de la red y la recompensa del proceso de monitorización de IPro. El elemento *Acumulador de Experiencia* guía la política (*i.e.*, qué acción tomar) del agente en dos pasos. En el primero, este elemento combina las observaciones hechas por el *Módulo Sensorial* y define la matriz de información que representa los estados internos de nuestro agente RL. En el segundo paso, el *Acumulador de Experiencia* mapea estos estados y asigna recompensas para indicar qué tan buenos o malos son estos estados según la experiencia acumulada. El *Selector de Acciones* guía la política de acción considerando la estrategia de sondeo más gratificante basada en el estado actual de la red.

La Figura 4 expone el procedimiento general realizado por el agente RL para optimizar el intervalo de sondeo. Como primera medida, se inicializan los parámetros intrínsecos de Q-learnig: el factor de aprendizaje α, el factor de descuento

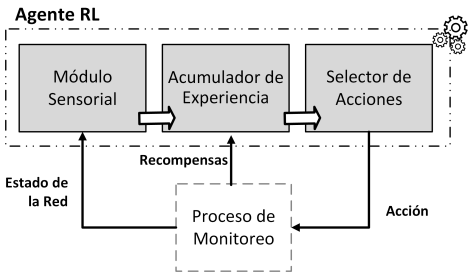

Figura 3. Modelo general del agente RL

γ, y el método de exploración ε-greedy. Además, el agente RL asume una condición inicial nula de $Q(\mathcal{S}, \mathcal{A})$ antes de que ocurra la primera actualización. Después de inicializar estos parámetros, comienza el proceso de optimización del intervalo de sondeo. En este proceso, el agente RL determina el intervalo de sondeo más gratificante al interactuar con la red realizando los siguientes pasos de manera iterativa:

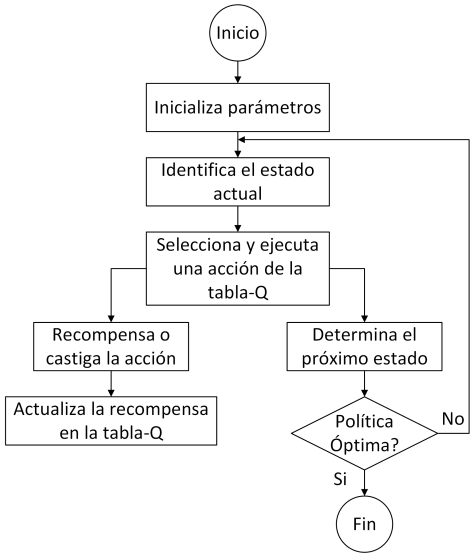

Figura 4. Optimización del intervalo de sondeo

1. Identifica el estado actual de la red (inicialmente selecciona un estado aleatorio que representa los valores iniciales del intervalo de sondeo, CCO, y CUC.) interactuando con el MP.
2. Selecciona una acción $A_t = \{up, down, keep\}$ de la tabla-Q mediante el método de exploración ε-greedy que modifica el intervalo de sondeo. Las acciones posibles son aumentar (up), reducir ($down$) o mantener ($keep$) el intervalo de sondeo. Luego, ejecuta la acción seleccionada que altera el comportamiento de la red y la emplaza a un nuevo estado. Este nuevo estado es percibido por el agente RL a través de MP.
3. El agente RL toma el estado percibido en el paso anterior para recomenzar o castigar la acción ejecutada.
4. Conforme con los parámetros iniciales, la recompensa,

y el nuevo estado de la red, el agente RL actualiza los valores de la tabla-Q de acuerdo con la Ecuación 1. Además, el agente RL determina y se mueve al nuevo estado.

5. El agente RL decide si el estado percibido en el paso 2 cumple con la política óptima (*i.e.,* que maximiza la recompensa acumulativa esperada para cada estado). Si no cumple, el agente RL pasa a la siguiente iteración $t+1$, de lo contrario, el agente RL obtiene el intervalo de sondeo más gratificante que mantiene CCO y CUC dentro de los valores objetivo, el cual minimiza la degradación del rendimiento de la red causada por las tareas de monitorización.

b) ***Plano de Control****:* Este plano programa la monitorización de la red realizando los pasos descritos en el Algoritmo 1. Inicialmente, este plano recibe la decisión de PCon sobre el nuevo intervalo de sondeo (I). Luego aplica dicha decisión para programar la tarea de recopilación de datos en la red monitorizada. Para recopilar los datos el PC envía mensajes de Estado-Lectura de solicitud a cada conmutador conectado (línea 3) a este plano cada I segundos (línea 7). Enseguida, recibe los mensajes de Estado-Lectura de respuesta de forma asíncrona (línea 5), que contienen la información estadística de los conmutadores. Finalmente, almacena la información estadística (línea 6) para mantener un registro de los cambios en la red. Estos pasos se repiten hasta alcanzar un criterio de parada (*e.g.,* errores y administrador).

Algorithm 1: Recopilación de datos

```
Require:
        Intervalo de sondeo I
1 while criterio de parada no alcanzado do
2     foreach conmutador ∈ conmutadores do
3         Enviar un mensaje de Estado-Lectura de solicitud al
            conmutador
4     end
5     Recibir los mensajes de Estado-Lectura de respuesta de los
        conmutadores // estos mensajes contienen la
        información estadística;
6     Almacenar la información estadística
7     Esperar I segundos
8 end
```

PC incluye cuatro elementos: el *Gestor de Sondeo, el Gestor de Consultas, el Módulo de Estadísticas, y el Repositorio de Datos*. El *Gestor de Sondeo* configura el intervalo de sondeo de acuerdo con la decisión tomada por el agente RL. El *Gestor de Consultas* programa la recopilación de datos en función del intervalo de sondeo calculado y los niveles de agregación deseados (*e.g.*, byte, paquete, flujo). Después de la recopilación de datos, el *Gestor de Consultas* combina y almacena la información estadística en el repositorio de datos. Por lo tanto, esta información puede ser utilizada posteriormente por aplicaciones de capa superior. El *Módulo de Estadísticas* es un servicio que se ejecuta en la parte superior del controlador SDN, que es útil para desarrollar aplicaciones de medición de red personalizadas. El *Repositorio de Datos* almacena información estadística de cada operación de monitorización y mantiene un rastro de los cambios en la red.

Es importante resaltar que el PC interactúa con el PD a través de OpenFlow [29] dado que este protocolo se ha convertido progresivamente en el estándar *de facto* para la interfaz sur (Southbound Interface - SBI) en SDN [30]. Además, este plano interactúa con PCon mediante un controlador del repositorio de datos.

c) ***Plano de Gestión****:* Este plano es responsable de extraer la información estadística de PC para proporcionar un análisis del estado de la red a PCon con respecto al CCO y CUC. PG incluye dos elementos llamados: *Recolector de Estadísticas de Flujo y Procesador de Datos*. El primero extrae la información estadística del *Módulo de Estadísticas* ubicado en PC. El segundo elemento es responsable de procesar y organizar la información recuperada por el *Recolector de Estadísticas de Flujo* para calcular el CCO y CUC. El *Procesador de Datos* también envía la información procesada al agente RL ubicado en PCon mientras mantiene un registro histórico del estado de la red en el *Repositorio de Datos*.

Es de destacar que PG interactúa con PC mediante una interfaz basada en REST. A su vez, PG se comunica con PCon mediante APIs específicas ya que, hasta ahora, no hay interfaces estandarizadas para PCon.

d) ***Plano de Datos****:* Este plano es responsable de reenviar flujos de la red SDN monitorizada a través de los dispositivos de red, desacoplados del PC. Los dispositivos de red cuentan con tablas en las que se determinan las reglas que guían el tráfico, y un cliente compatible con el protocolo OpenFlow, que se encarga de comunicarse con el controlador y realizar las acciones que éste le indique. Es importante mencionar que, en SDN, es importante tomar decisiones de monitorización inteligentes, mediante el PG, PC y PCon, con el objetivo de mejorar el rendimiento de la red [22].

V. Evaluación

Para evaluar el enfoque propuesto, se creó un entorno de prueba, incluida la red SDN para monitorizar, y se desarrolló un prototipo de IPro. Las pruebas se llevaron a cabo realizando mediciones experimentales del CCO, el CUC y la MA del prototipo construido. MA refleja la diferencia entre el valor real de una métrica y el valor medido por IPro. En particular, para evaluar la MA, medimos las métricas de rendimiento (Throughput) y retardo (Delay).

a) ***Entorno de Prueba****:* La Figura 5 presenta el entorno de prueba utilizado para evaluar IPro. Este entorno incluye la topología de red del campus de la Universidad Federal de Rio Grande do Sul [31], el Controlador Ryu y el prototipo IPro. La topología monitorizada incluye 11 conmutadores OpenFlow (versión 1.3) que conectan 230 hosts de 7 laboratorios (con 20, 30 y 40 hosts) y 4 oficinas de administración (cada una con 10 hosts) a través de enlaces con un ancho de banda de 10 Mbps cada uno. Además, dicha topología incluye un servidor web y un servidor de vídeo conectado en una de las oficinas de administración. Cada conmutador de la red está conectado, a través de un canal TCP dedicado, a un controlador Ryu remoto utilizado para coordinar las decisiones de reenvío de toda la red. El emulador Mininet [32] se usó para implementar la topología monitorizada, la cual se ejecutó en un servidor

Ubuntu con un Intel i7-4770 2.26 GHz y 8GB RAM. El controlador Ryu se ejecutó en un servidor Ubuntu con un procesador Core i7-4770 y 2 GB de RAM.

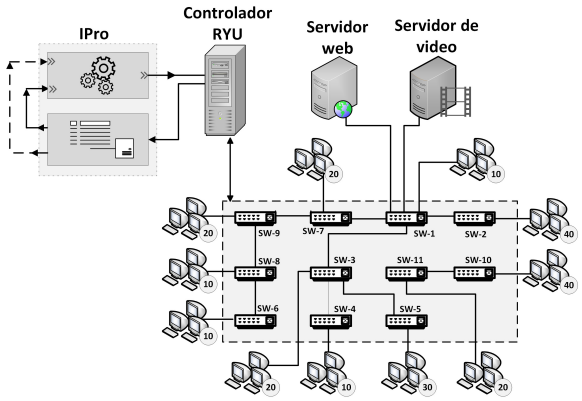

Figura 5. Entorno de prueba

Para evaluar IPro de manera confiable, necesitamos un marco de evaluación realista que refleje los patrones de tráficos actuales y previstos (*e.g.,* web, P2P y video). En este artículo, hemos recurrido a dos estudios de medición que investigan la composición de patrones de tráfico realistas [33-34]. Los resultados indican el dominio del tráfico web, que asciende al 52 % del tráfico medido en general, seguido por el tráfico de vídeo con el 25-40 % de todo el tráfico, mientras que el tráfico P2P constituye solo el 18.3 % del tráfico total. Teniendo en cuenta dichos resultados, en todos los experimentos, generamos solo tráfico de vídeo y web, en una proporción de 75 % a 25 %, respectivamente (dado que el tamaño de las solicitudes de vídeo genera más tráfico que las solicitudes web). Para generar este tráfico, incluimos en la topología del campus un servidor web y un servidor de vídeo. También asumimos que los 230 hosts están activos durante todo el tiempo del experimento y realizamos una solicitud en promedio cada 30 segundos. Además, todos los resultados del experimento tienen un nivel de confianza igual o superior al 95 %.

b) ***Prototipo****:* La Figura 6 representa el prototipo IPro que incluye: el *RL-agent*, el *Procesador de Datos*, el *Recolector de Estadísticas de Flujo*, el *Repositorio de Datos*, el *Gestor de Sondeo*, el *Gestor de Consultas* y el *Módulo de Estadísticas*. El *RL-agent* y el *Procesador de Datos* se desarrollaron e implementaron utilizando la versión 1.15 de Numpy, que es el paquete fundamental de Python para la informática científica. El *Gestor de Sondeo*, el *Gestor de Consultas* y el *Módulo de Estadísticas* se desarrollaron utilizando la API basada en REST proporcionada por Ryu. Esta API ayuda a recuperar las estadísticas del interruptor. El *Recolector de Estadísticas de Flujo* se desarrolló utilizando la API de Ryu basada en Python. El *Repositorio de Datos* fue desarrollado en MySQL.

Es importante resaltar que el *Módulo de Estadísticas* interactúa con PD mediante el Protocolo OpenFlow [29]. Utilizamos este protocolo porque se ha activado progresivamente el estándar SBI *de facto* en SDN [30]. OpenFlow describe un protocolo abierto que permite a las aplicaciones de software

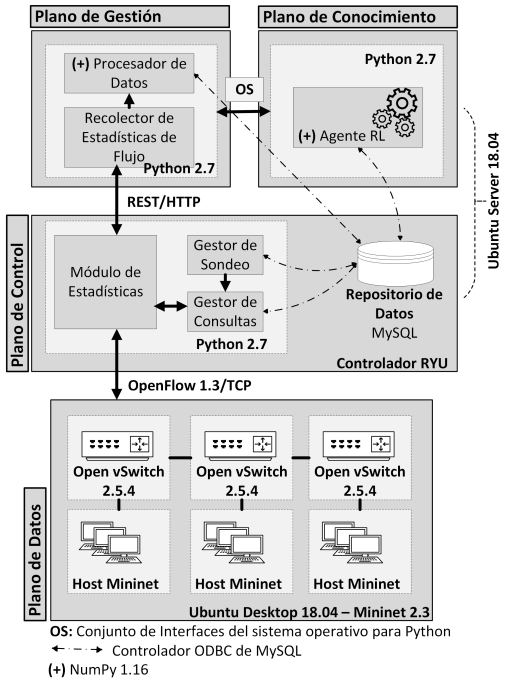

Figura 6. Prototipo IPro

programar (*i.e.*, agregar, actualizar y eliminar entradas de flujo) la tabla de flujo de los conmutadores compatibles con OpenFlow. En particular, el *Módulo de Estadísticas* utiliza la versión 1.3 de OpenFlow. Específicamente, este módulo utiliza dos mensajes de Estado-Lectura (solicitud-respuesta) para recopilar información del conmutador, como la configuración actual, estadísticas y capacidades. El controlador envía un mensaje de Estado-Lectura de solicitud a los conmutadores para solicitar las estadísticas de tráfico de los flujos. Los conmutadores comunican al controlador las estadísticas de tráfico solicitadas a través de mensajes de Estado-Lectura de respuesta. Para obtener detalles de OpenFlow, remitimos al lector a la especificación de Openflow [29].

c) ***Comparación****:* Comparamos IPro con PPA (*i.e.,* un enfoque basado en extracción dirigido a monitorizar los conmutadores de red con un intervalo de sondeo predefinido) con respecto a CCO, CUC y MA. Los intervalos de prueba probados oscilaron entre 1 y 15 segundos (en pasos de 1 segundo). Para cada intervalo, realizamos la prueba 32 veces durante 600 segundos con el objetivo de obtener el promedio de CCO, CUC y MA. A continuación, analizamos solo los intervalos de sondeo de 4, 5 y 6 segundos porque, en ellos, la MA de las mediciones de las métricas de rendimiento y retraso fueron superiores al 80 %.

Los resultados experimentales (*cf.* Tabla II) revelan hechos diversos. Primero, IPro tiene un CCO significativamente más pequeño que el PPA. La reducción en los intervalos de 4, 5 y 6 segundos es de alrededor de 16.17 %, 10.22 % y 10.1 %, respectivamente. En segundo lugar, en los mismos intervalos, IPro usa mejor la CPU del controlador que PPA aproximadamente 13.2 %, 4.3 % y 2.7 %, respectivamente. En tercer lugar,

Tabla II. TABLA COMPARATIVA

Intervalo sondeo [s]	MA of Throughput [%]	MA of Delay [%]	CUC [%]	CCO [%]
IPro	90.11	92.60	7.40	1.23
PPA with 4	83.83	82.70	20.60	17.40
PPA with 5	88.83	86.50	11.70	11.45
PPA with 6	86.06	83.02	10.10	11.33

en los intervalos mencionados, IPro logra una MA más alta cuando se usa para medir la métrica de rendimiento que PPA aproximadamente 6.28 %, 1.28 % y 4.05 %, respectivamente. Cuarto, en los intervalos analizados, IPro logra una MA más alta cuando se usa para medir la métrica de retraso que el PPA de aproximadamente 9.9 %, 6.1 % y 9.58 %, respectivamente. Estos hechos se deben a que, en cada paso de tiempo, IPro utiliza el estado de la red para mejorar sus políticas de control y, luego, toma las mejores medidas basadas en las políticas mejoradas. Estas políticas conducen a una mejor supervisión de CCO y CUC. En resumen, IPro proporciona un buen comportamiento en CCO y CUC sin comprometer la MA.

VI. CONCLUSIÓN

En este articulo fue presentado IPro, un enfoque que determina el intervalo de sondeo que mantiene a CCO y CUC dentro de umbrales aceptables en función del comportamiento de la red. Las principales contribuciones son, en primer lugar, la arquitectura IPro que proporciona una solución simple y eficiente para la monitorización de redes basadas en SDN siguiendo el concepto de KDN. En segundo lugar, el agente RL que determina el intervalo de sondeo considerando las variaciones del tráfico de red y mantiene CCO y CUC dentro de los valores objetivo. En tercer lugar, el prototipo IPro que logra un CCO menor que 1.23 %, CUC menor que 8 % y un MA mayor que 90 % (superando a PPA).

También se presentó un escenario de monitorización realista, en el que se combinaron múltiples elementos de interacción para implementar el agente RL, a saber, *Módulo Sensorial, Acumulador de Experiencia y Selector de Acciones*. La implementación de referencia y el agente RL implementado fueron capaces de superar los desafíos planteados, lo que demuestra la relevancia de esta propuesta. El prototipo desarrollado incluyó un agente y un Módulo de Estadísticas que recopilan información estadística detallada, exacta y oportuna basados en el comportamiento del canal de control y del controlador, una característica que la mayoría de los enfoques de monitorización de SDN no consideran. Teniendo en cuenta los resultados de la evaluación cuantitativa realizada en este escenario, se puede decir que, Ipro logró reducir el CCO y CUC, y con ello la degradación del rendimiento de la red causada por las tareas de monitorización, manteniendo una MA alta.

El trabajo futuro incluye la correlación de otros parámetros (*e.g.,* MA y recursos computacionales de los conmutadores) en la función de recompensa y el uso de otras técnicas RL (*e.g.,* Deep RL) con el objetivo de mejorar la estimación del intervalo de sondeo.

REFERENCIAS

[1] C. C. Machado, L. Z. Granville, A. Schaeffer-Filho, and J. A. Wickboldt, "Towards sla policy refinement for qos management in software-defined networking," in *IEEE 28th International Conference on Advanced Information Networking and Applications*, May 2014, pp. 397–404.

[2] H. Kim and N. Feamster, "Improving network management with software defined networking," *IEEE Communications Magazine*, vol. 51, no. 2, pp. 114–119, 2013.

[3] S. Agarwal, M. Kodialam, and T. Lakshman, "Traffic engineering in software defined networks," in *2013 Proceedings IEEE INFOCOM*. IEEE, 2013, pp. 2211–2219.

[4] O. M. C. Rendon, C. R. P. dos Santos, A. S. Jacobs, and L. Z. Granville, "Monitoring virtual nodes using mashups," *Computer Networks*, vol. 64, pp. 55–70, 2014.

[5] J. Suh, T. T. Kwon, C. Dixon, W. Felter, and J. Carter, "Opensample: A low-latency, sampling-based measurement platform for commodity sdn," in *IEEE 34th International Conference on Distributed Computing Systems*, June 2014, pp. 228–237.

[6] K. Dharsee, E. Johnson, and J. Criswell, "A software solution for hardware vulnerabilities," in *2017 IEEE Cybersecurity Development (SecDev)*. IEEE, 2017, pp. 27–33.

[7] M. Aslan and A. Matrawy, "On the impact of network state collection on the performance of sdn applications," *IEEE Communications Letters*, vol. 20, no. 1, pp. 5–8, 2016.

[8] Z. Su, T. Wang, Y. Xia, and M. Hamdi, "Flowcover: Low-cost flow monitoring scheme in software defined networks," in *Global Communications Conference (GLOBECOM), 2014 IEEE*. IEEE, 2014, pp. 1956–1961.

[9] S. R. Chowdhury, M. F. Bari, R. Ahmed, and R. Boutaba, "Payless: A low cost network monitoring framework for software defined networks," in *IEEE Network Operations and Management Symposium*, May 2014, pp. 1–9.

[10] D. Raumer, L. Schwaighofer, and G. Carle, "Monsamp: A distributed sdn application for qos monitoring," in *Federated Conference on Computer Science and Information Systems*, Sept 2014, pp. 961–968.

[11] N. L. M. van Adrichem, C. Doerr, and F. A. Kuipers, "Opennetmon: Network monitoring in openflow software-defined networks," in *IEEE Network Operations and Management Symposium*, May 2014, pp. 1–8.

[12] H. Tahaei, R. Salleh, S. Khan, R. Izard, K.-K. R. Choo, and N. B. Anuar, "A multi-objective software defined network traffic measurement," *Measurement*, vol. 95, pp. 317–327, 2017.

[13] A. Tootoonchian, M. Ghobadi, and Y. Ganjali, *OpenTM: Traffic Matrix Estimator for OpenFlow Networks*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 201–210. [Online]. Available: https://doi.org/10.1007/978-3-642-12334-4_21

[14] P. Sun, M. Yu, M. J. Freedman, J. Rexford, and D. Walker, "Hone: Joint host-network traffic management in software-defined networks," *Journal of Network and Systems Management*, vol. 23, no. 2, pp. 374–399, Apr 2015. [Online]. Available: https://doi.org/10.1007/s10922-014-9321-9

[15] X. T. Phan and K. Fukuda, "Sdn-mon: Fine-grained traffic monitoring framework in software-defined networks," *Journal of Information Processing*, vol. 25, pp. 182–190, 2017.

[16] L. Liao, V. C. M. Leung, and M. Chen, "An efficient and accurate link latency monitoring method for low-latency software-defined networks," *IEEE Transactions on Instrumentation and Measurement*, vol. 68, no. 2, pp. 377–391, Feb 2019.

[17] L. Jose and M. Yu, "Online measurement of large traffic aggregates on commodity switches," in *USENIX Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services, Boston, MA, USA,March 29, 2011*, 2011. [Online]. Available: https://www.usenix.org/conference/hot-ice11/online-measurement-large-traffic-aggregates-commodity-switches

[18] G. Tangari, D. Tuncer, M. Charalambides, and G. Pavlou, "Decentralized monitoring for large-scale software-defined networks," in

IFIP/IEEE Symposium on Integrated Network and Service Management, May 2017, pp. 289–297.

[19] G. Tangari, D. Tuncer, M. Charalambides, Y. Qi, and G. Pavlou, "Self-adaptive decentralized monitoring in software-defined networks," *IEEE Transactions on Network and Service Management*, vol. 15, no. 4, pp. 1277–1291, Dec 2018.

[20] X. T. Phan, I. D. Martinez-Casanueva, and K. Fukuda, "Adaptive and distributed monitoring mechanism in software-defined networks," in *2017 13th International Conference on Network and Service Management (CNSM)*, Nov 2017, pp. 1–5.

[21] H. Tahaei, R. B. Salleh, M. F. Ab Razak, K. Ko, and N. B. Anuar, "Cost effective network flow measurement for software defined networks: A distributed controller scenario," *IEEE Access*, vol. 6, pp. 5182–5198, 2018.

[22] A. Mestres, A. Rodriguez-Natal, J. Carner, P. Barlet-Ros, E. Alarcón, M. Solé, V. Muntés-Mulero, D. Meyer, S. Barkai, M. J. Hibbett *et al.*, "Knowledge-defined networking," *ACM SIGCOMM Computer Communication Review*, vol. 47, no. 3, pp. 2–10, 2017.

[23] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction-second edition*. Cambridge, Massachusetts, 2018, vol. 1, no. 1.

[24] E. Duryea, M. Ganger, and W. Hu, "Exploring deep reinforcement learning with multi q-learning," *Intelligent Control and Automation*, vol. 7, no. 04, p. 129, 2016.

[25] F. Farahnakian, M. Ebrahimi, M. Daneshtalab, P. Liljeberg, and J. Plosila, "Q-learning based congestion-aware routing algorithm for on-chip network," in *Networked Embedded Systems for Enterprise Applications (NESEA), 2011 IEEE 2nd International Conference on*. IEEE, 2011, pp. 1–7.

[26] R. Boutaba, M. A. Salahuddin, N. Limam, S. Ayoubi, N. Shahriar, F. Estrada-Solano, and O. M. Caicedo, "A comprehensive survey on machine learning for networking: evolution, applications and research opportunities," *Journal of Internet Services and Applications*, vol. 9, no. 1, p. 16, Jun 2018.

[27] S. Répás, V. Horváth, and G. Lencse, "Stability analysis and performance comparison of three 6to4 relay implementations," 07 2015.

[28] H. Xu, Z. Yu, C. Qian, X. Y. Li, and Z. Liu, "Minimizing flow statistics collection cost of sdn using wildcard requests," in *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, May 2017, pp. 1–9.

[29] O. N. Foundation, "Openflow switch specification v1.3.0." ONF, 2013.

[30] B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 3, pp. 1617–1634.

[31] P. H. Isolani, J. A. Wickboldt, C. B. Both, J. Rochol, and L. Z. Granville, "Interactive monitoring, visualization, and configuration of openflow-based sdn," in *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*. IEEE, 2015, pp. 207–215.

[32] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: rapid prototyping for software-defined networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*. ACM, 2010, p. 19.

[33] C. Labovitz, S. Iekel-Johnson, D. McPherson, J. Oberheide, and F. Jahanian, "Internet inter-domain traffic," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 75–86, 2011.

[34] G. Maier, A. Feldmann, V. Paxson, and M. Allman, "On dominant characteristics of residential broadband internet traffic," in *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement*. ACM, 2009, pp. 90–102.