Publicaciones en Ciencias y Tecnología Vol. 11 Nº 2, Jul-Dic (2017) 63–77

Artículo de investigación

Implementación de tareas de analítica de datos para mejorar la calidad de servicios en redes de comunicaciones

Implementation of data analytics tasks to improve the quality of services in communications networks

José Aguilar[a], Kristell Aguilar[b], Marxjhony Jerez[a], Carlos Jiménez[a]

[a] *Universidad de Los Andes, Venezuela*
[b] *Université de Pau et des Pays de l'Adour, Francia*

Recibido: 01-12-2017 Aceptado: 30-06-2018

Resumen

Este trabajo especifica ciclos autónomos (CA) de tareas de análisis de datos para optimizar la calidad de servicios (QoS) en Internet. Los mecanismos para mejorar son importantes para los proveedores de servicios de Internet (ISP) y se basan en el análisis del contexto, en la inspección profunda de paquetes (DPI), en el uso de la minería de datos y semántica, entre otras. Los CA de análisis de datos propuestos en este trabajo integran esos aspectos, para realizar tareas para mejorar la QoS en Internet, tales como la tarea de clasificación del tráfico en la red. En este trabajo se utiliza la metodología MIDANO, para especificar los dos CA que se proponen, uno con el objetivo de mejorar la QoS en Internet, y otro con el objetivo de aprender el patrón del tráfico en la red. Además, en este trabajo se implementa el CA que mejora la QoS en Internet. Este CA monitorea el estado del tráfico en Internet, determina el comportamiento de las aplicaciones, caracteriza los patrones de tráfico, genera reglas de optimización del tráfico, entre otras cosas, usando técnicas de DPI, minería semántica, aprendizaje automático, entre otras.

Palabras clave: analítica de datos, minería semántica, minería de datos, telecomunicaciones, QoS.
Código UNESCO: 120304- Inteligencia artificial

Abstract

This work specifies Autonomous Cycles (AC) of data analysis tasks, to optimize the Quality Of Services (QoS) on the Internet. The mechanisms to improve QoS on the Internet are important for Internet Service Providers (ISP). These mechanisms should be based on context analysis, Deep Packet Inspection (DPI), the use of data mining and semantics, among others. The ACs of data analysis proposed in this work integrate these aspects, to perform tasks to improve QoS on the Internet, such as the task of classifying traffic on the network. In this paper the MIDANO methodology is used, to specify the two ACs that are proposed, one with the aim of improving the QoS on the Internet, and another with the objective of learning the traffic pattern in the network. In addition, this work implements the AC that improves the QoS on the internet. This AC monitors the state of Internet traffic, determines the behavior of applications, characterizes traffic patterns, generates traffic optimization rules, among other things, using DPI techniques, semantic mining, machine learning, among others.

Key words: data analytics, semantic mining, data mining, telecommunications, QoS.
UNESCO Code: 120304 - Artificial intelligence

Impresa:ISSN 1856-8890. Dep. Legal pp200702LA2730. Digital: ISSN 24779660 Dep. Legal ppi201402LA4590.
Email addresses: aguilar@ula.ve (José Aguilar), kristell153@gmail.com (Kristell Aguilar), marxjhony@ula.edu.ve (Marxjhony Jerez), solracjh@gmail.com (Carlos Jiménez)

1. Introducción

La Analítica de Datos (AdD) es una ciencia que permite analizar datos y convertirlos en conocimiento, con el fin de mejorar el entorno donde se producen esos datos [1]. La misma ha sido usada en múltiples ámbitos, por sus capacidades para descubrir conocimiento útil, el cual, además, es usado en los procesos de toma de decisión. Se han desarrollado mecanismos, estrategias, metodologías para facilitar el uso de la AdD. Debido a la cantidad de datos en la red de datos, la AdD puede ser de mucha utilidad. Algunos trabajos previos han estudiado problemas específicos en Internet desde el análisis de sus datos, pero sin considerarlos de manera integral. Por ejemplo, los trabajos [2], [3] analizan diferentes técnicas de Aprendizaje Automático (ML, por sus siglas en inglés Machine Learning) para la clasificación de tráfico, Roughan [4] hace uso de ML para la detección de intrusos, igual que Sommer et al. [5], y Lee et al. [6] usan ML para mejorar los servicios de mensajería de texto. En este artículo, se propone usar el concepto de CA de Tareas de AdD propuesto en [1] [7], el cual es un tipo de supervisión inteligente autónoma que permite plantear objetivos estratégicos alrededor del problema de QoS en las redes de comunicación. Los CAs integran un conjunto de tareas de AdD, que de manera autónoma trabajan colectivamente para alcanzar los objetivos estratégicos que persiguen dichos ciclos [1] [7]. En particular, en este trabajo se proponen dos CAs para la adaptación del tráfico de red para mejorar la QoS de las redes comunicacionales. Dichos CAs están compuestos de diversas tareas de AdD, las cuales se alimentan entre sí, para transformar datos en conocimiento. El primer CA, llamado CA de Mejora de la QoS en Internet, tiene como objetivo mejorar la QoS, para lo cual usa DPI para determinar los tramos de red y tipos de paquetes que necesitan ser priorizados. El segundo CA, llamado de Aprendizaje de Tráfico tiene como objetivo descubrir nuevos patrones de tráfico en los paquetes, y determinar para cada uno de ellos, cual herramienta de Inspección Profunda de Paquetes (DPI) lo caracteriza de mejor manera. Ambos CAs son complementarios, donde la salida de uno alimenta la otra. En este artículo solo se implementan las tareas del primer CA. La primera tarea consiste en determinar el comportamiento de cada aplicación, mientras que la segunda tarea consiste en identificar sus patrones de tráfico. Este artículo se organiza de la siguiente manera, en la primera parte se presenta la metodología usada en este trabajo; seguidamente se presenta el marco teórico de base de nuestro trabajo. La sección cuatro presenta la problemática, los trabajos relacionados a la problemática, la definición de los ciclos autonómicos, y las tareas de análisis de datos del CA de mejora de la QoS. Se presenta una sección con el análisis de los resultados obtenidos, y finalmente, las conclusiones y trabajos futuros.

2. Desarrollo

2.1. Metodología

La metodología empleada para la realización de este trabajo se denomina MIDANO [8] [9] la cual es utilizada para el Desarrollo de Aplicaciones de Minería de Datos (MD) basada en el Análisis Organizacional. Esta es una metodología diseñada para el desarrollo de aplicaciones basadas en AdD [10] [11] y está compuesta por tres fases, ver figura 1.

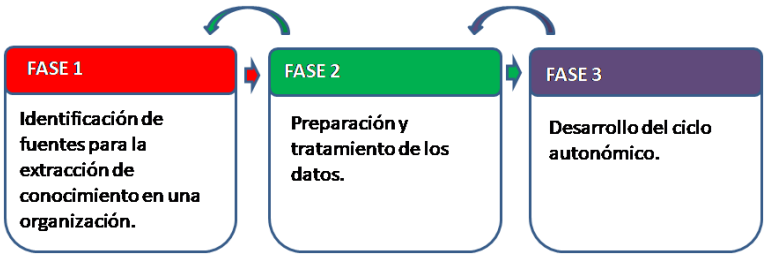

Figura 1: Fases de Midano [12] [13]

Fase 1. Identificación de fuentes para la extracción de conocimiento en una organización: El principal objetivo de esta fase es conocer la organización, sus procesos, sus expertos, entre otros aspectos, para definir el objetivo de la aplicación de AdD en la organización. Además, se hace una especificación de los CAs de AdD a desarrollar.

Fase 2. Preparación y tratamiento de los datos: Este proceso se basa en el paradigma ETL: Extracción de los datos desde sus fuentes, Transformación de los datos, y Carga (Load) de los mismos en el almacén de datos del CA. Para realizar este proceso se crea una vista minable, la cual está compuesta por la descripción de todas las variables de interés, y algunos campos adicionales de importancia para realizar el proceso de tratamiento. Con esta vista minable conceptual se crea el modelo de datos a ser usado por el medio de almacenamiento (almacén de datos). Finalmente, el

medio de almacenamiento es cargado con los datos. El medio de almacenamiento cargado con los datos, se le llama vista minable operativa.

Fase 3. Desarrollo del CA de treas de AdD: esta fase tiene como objetivo implementar las diferentes tareas de AdD del CA, que generan los modelos de conocimientos requeridos (por ejemplo, modelos predictivos, modelos descriptivos, etc.). Esta etapa culmina con la implementación de un prototipo del CA. Durante esta fase se realizan experimentos para validar los modelos de conocimiento generados por las tareas de AdD.

En nuestro caso en particular, las fuentes de datos analizadas, que permiten conocer el estado en la red, fueron trazas de tráfico en la red, comportamientos en las redes sociales, incidencias en las comunicaciones, entre otras. A partir de las mismas, se determinan las variables para identificar lo que sucede en la red. En la tabla 1 se resume el uso de MIDANO en este trabajo.

Cuadro 1: Uso de las Fases de MIDANO en este trabajo

Fase	Uso
Fase 1	Análisis del proceso de "Gestión y Monitoreo de la Red" Especificación de los CAs: Mejora de la QoS en la red, Aprendizaje del Tráfico en la red
Fase 2	Identificación de las fuentes de datos: Trazas, redes sociales, entre otros
Fase 3	Implementación del CA de mejora de la QoS de la red

2.2. Estado del arte

Crotti et al. [14] estudian la característica de los flujos. Estas características son tamaño del paquete IP, tiempo de inter-llegada de los paquetes, y la cardinalidad de los flujos. Paxson [15] propone un conjunto de variables para una clasificación de flujos y paquetes. Estos trabajos son de interés en nuestro trabajo, ya que determinan los descriptores/características a considerar para describir los flujos de comunicación. Karagiannis et al. [16] proponen una estrategia de clasificación basada en la observación estadística de las propiedades del flujo. Sus resultados no logran identificar algunas aplicaciones, como las Peer to Peer (P2P). Dewes et al. [17] consideran que el tráfico de chat es dominado para la interacción de los humanos. Ellos realizan la identificación de ese tráfico usando una clasificación estadística. Este método logra clasificar un 93.1 % del tráfico. Los trabajos anteriores describen enfoques estadísticos para analizar el tráfico en las redes comunicación, lo cual es relevante en nuestro trabajo para caracterizar el producto a generar en algunas de las tareas de monitoreo de los ciclos autónomos.

Por otro lado, existen diversos trabajos sobre la clasificación del tráfico de Internet basado en ML [18] [19] [13] [20] [21] [22] [23] [24]. En [19] Kim et al. proponen una herramienta para la estimación de la QoS y la clasificación del tráfico basada en técnicas de aprendizaje supervisado. Proponen un clasificador basado en la técnica Support Vector Machine (SVM) que analiza algunos descriptores del tráfico, como las características de los tres primeros bits de los paquetes, para clasificar los flujos. En Tan et al. [23], los autores proponen un enfoque para identificar los flujos de tráfico utilizando un método de ML híbrido. Resuelven dos problemas para la identificación del tráfico de Internet: la clasificación de las diferentes aplicaciones, y para cada característica, la definición de sus propiedades e importancia. Utilizan una red neuronal artificial de backpropagation y un algoritmo PSO (Particle Swarm Optimization), para identificar el flujo del tráfico.

El objetivo del trabajo de Hadjadj-Aoul et al. [13] es aumentar la precisión del DPI y los métodos estadísticos, al ensamblarlos conjuntamente en un solo clasificador utilizando RNAs. Proponen una metodología para la caracterización del tráfico de la red basada en discriminadores estadísticos y en DPI, que proporcionan información suficiente para distinguir varias aplicaciones. Otro método de ML utilizado para clasificar el tráfico son las redes bayesianas, por ejemplo, Nechay [22] utiliza los siguientes discriminadores como parámetros de entrada: la duración del flujo, el puerto TCP y el tiempo entre llegadas de paquetes. Ellos proponen dos algoritmos bayesianos, para calcular la probabilidad de pertenecer a una clase de tráfico, dada la observación de los discriminadores. También, en Tan et al. [25] definen dos tipos de clasificadores de tráfico de Internet en línea basado en SVM, uno adecuado para aplicaciones tipo P2P, y otro para aplicaciones del tipo VoIP. Por otro lado, Yamansavascilar et al. [24] estudian métodos de ML para la identificación de aplicaciones populares como Facebook, Twitter y Skype. Usaron el conjunto de datos UNB ISCX Network Traffic y evaluaron cuatro algoritmos: J48, Random Forest, k-NN y Bayes Net. Los datos tenían 111 descriptores, pero después lo redujeron a 12 funciones. Según el número de descriptores la técnica de ML que da mejor resultados cambia, k-NN fue la mejor para 111 descriptores, y Random Forest para 12 descriptores.

En Trivedi et al. [26] se propone un sistema de clasificación del tráfico de red haciendo uso de un hibrido entre DPI y técnicas de ML. Por otro lado, Shafiq et al. [27] [28] realizan un análisis comparativo de diferentes algoritmos de ML en tareas de clasificación de tráfico de red, y Arcia et al. [18] [20] proponen un esquema de identificación de aplicaciones a través de la clasificación del tráfico de red, en especial para aplicaciones Skype y P2P. Los anteriores trabajos muestran la diversidad de enfoques basados en ML, que se han usado para analizar el tráfico en las redes de comunicación. Esos trabajos son un importante insumo para la especificación de las tareas de AdD de nuestros ciclos autónomos, que tienen que ver con el análisis de tráfico.

A su vez, en algunos trabajos se han hecho propuestas de servicios de análisis de tráfico, para la clasificación o detección del mismo [29]. Por ejemplo, Alhamazani et al. [30] presentan una plataforma multicapas en la nube, para el monitoreo de tráfico usando el paradigma de big data, llamado CLAMS (Cross-Layer Multi-Cloud Application Monitoring-as-a-Service). Roughan et al. [4] proponen un esquema de clasificación del tráfico como servicio, conocido como Class of Service (CoS), en donde se crean las CoS basado en el análisis estadístico del tráfico. Finalmente, en Le et al. [6] usan la técnica de ML no supervisada denominada SOM (Self Organizing Map), para detectar tráfico desconocido. Estos trabajos nos permiten comprender la complejidad del problema de mejora de la QoS, asi de su posible implementación usando la noción de servicios, lo cual es fácilmente integrado en nuestros CAs, ya que las tareas de AdD son vistas como servicios. Finalmente, algunas herramientas computacionales que han implementado soluciones DPI son: PACE 2.0 [31], SolarWinds Network Performance Monitor [32], NDPI [33], Bro [12]. Estas herramientas son de interés en nuestro trabajo, porque serán utilizadas en las especificaciones de algunas de las tareas de AdD de nuestros CAs.

Como se ha podido constatar, existen diversos trabajos que han estudiado problemas específicos de gestión de tráfico en las redes: identificación, detección, etc. Además, han estado orientados a estudiar tráficos específicos, usando enfoques estáticos (clasificación), entre otras cosas. En este trabajo se propone un enfoque que integra múltiples tareas de AdD para resolver el problema de optimización de la QoS, bajo un enfoque orientado a servicios que permite su fácil escalabilidad y actualización, en un CA que le confiere la capacidad de auto-adaptación a las redes de comunicación.

2.3. Inspección Profunda de Paquetes (DPI)

En general, un proceso de clasificación del tráfico de la red consiste en la captura del tráfico desde una interface de red (Network Interface Card, NIC), su análisis, y la determinación del tipo del tráfico (ver figura 2), tal que sus resultados puedan ser guardados (análisis pasivo) o usados para tomar acciones (análisis activo). En específico, se capturan paquetes (PCAP) de la red, los cuales son guardados en un archivo, o leídos directamente por el analizador de tráfico de red en tiempo real. En cuanto al análisis, es el que permite realizar la identificación y clasificación del tráfico de red. En particular, existen básicamente cinco técnicas para lograr la identificación del tráfico [34] [35]: i) Métodos basados en DPI; ii) Métodos basados en el comportamiento del tráfico; iii) Métodos basados en las características del flujo de la red; iv) Métodos basados en el análisis de los Puertos; y v) Métodos basados en ML.

En cuanto al método DPI, analiza de manera automatizada la carga útil del paquete IP, pudiéndolo hacer de dos maneras [35]:

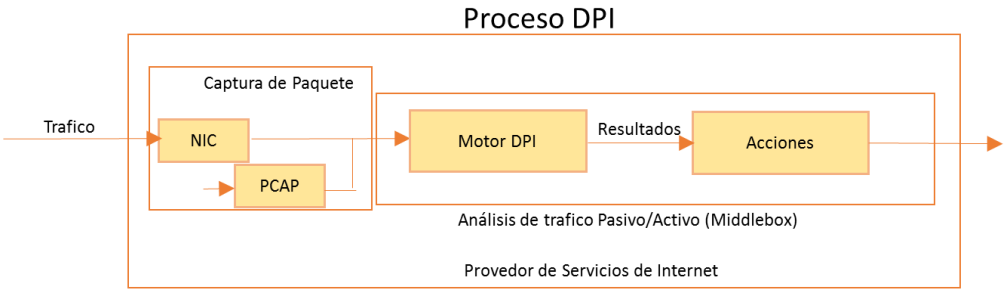

Figura 2: Proceso general de análisis de tráfico (elaboración propia)

a) Coincidencia de patrones: se basa en la búsqueda a través de los datos de la red, de secuencias conocidas de bytes (patrones), modeladas de diferentes formas, como por ejemplo como expresiones regulares (ver figura 3).

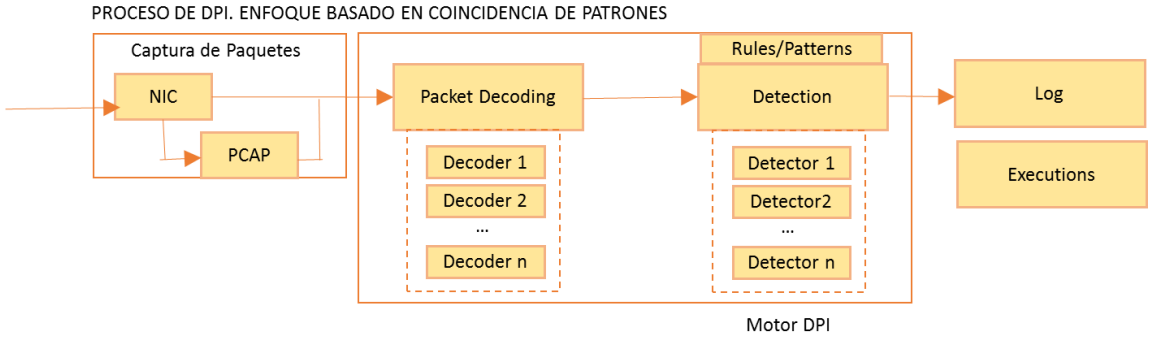

Figura 3: DPI con coincidencia de patrones (elaboración propia)

b) Análisis de eventos: Los paquetes son procesados por un "motor de eventos", el cual filtra el stream y lo convierte en una serie de eventos de alto nivel, y luego, a través de una política de interpretación de scripts, se identifican los eventos en la secuencia de datos (ver figura 4) [36].

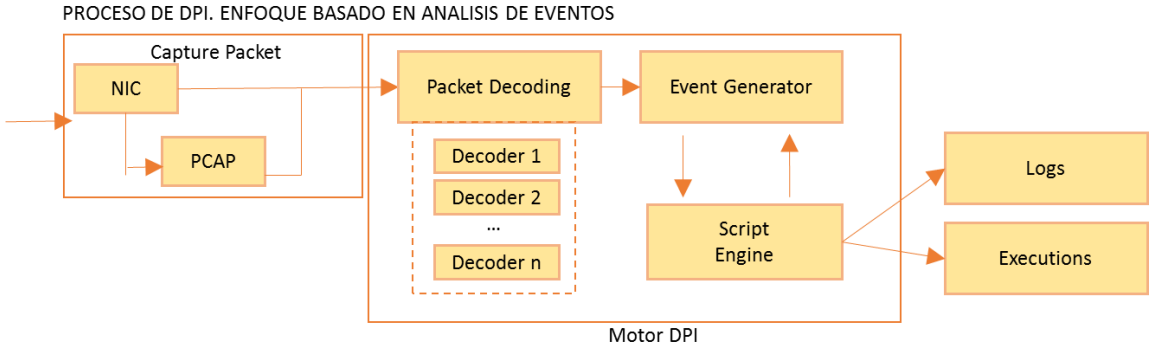

Figura 4: DPI con Análisis de eventos (elaboración propia)

2.4. Ciclos autónomos de tareas de análisis de datos

El alto grado de datificación incrustado en la sociedad, exige nuevas herramientas y mecanismos para la manipulación y la representación de los datos, que faciliten la extracción de información significativa, lo que ha dado lugar al dominio denominado AdD. La AdD es la ciencia que se encarga de la interpretación de los datos, con el fin de descubrir información y conocimiento [1]. Los objetivos principales de la AdD son ayudar a ver los problemas del negocio desde una perspectiva de los datos, y comprender los principios de extracción de conocimiento útil a partir de los datos. En general, genera modelos descriptivos, de identificación y predictivos, entre otros, con el fin de producir conocimiento importante a partir de datos, que se utilizará para guiar la toma de decisiones. Por otro lado, un "CA de Tareas de AdD" es un concepto definido en [1] [7], que consiste en un conjunto de tareas de AdD que deben realizarse juntas con el propósito de alcanzar un objetivo en el proceso que supervisan. Este conjunto de tareas interactúa entre sí, y tienen diferentes roles en el ciclo: Observar el proceso, analizar e interpretar lo que pasa en él, y tomar decisiones sobre el proceso, que permitan alcanzar el objetivo para el que fue diseñado el ciclo.

Esta integración de tareas de AdD en un ciclo cerrado, permite resolver problemas complejos, que requieren una gran cantidad de conocimiento para la solución. En este sentido, es muy importante integrar las tareas de AdD de forma coherente para generar conocimiento útil, estratégico para el logro de los objetivos. La descripción detallada de los roles de cada tarea en un CA es:

a) Tareas de observación: son un conjunto de tareas, encargadas de monitorear el proceso. Deben capturar datos e información acerca del comportamiento del entorno. Además, son responsable de la preparación de dichos datos.
b) Tareas de análisis del sistema: son un conjunto de tareas cuyo propósito es interpretar, entender y diagnosticar, a partir de los datos, lo que puede estar sucediendo en el contexto supervisado por el ciclo. Esto significa que estas tareas construyen modelos de conocimiento acerca de las dinámicas observadas por el ciclo.

c) Toma de decisiones: Este conjunto de tareas se encarga de implementar las tareas necesarias para la toma de decisiones, que conllevan a la mejora del proceso donde es aplicado el ciclo. Este conjunto de tareas impacta la dinámica del proceso, con la intención de mejorarlo. Los efectos de estas tareas, son nuevamente evaluadas en las etapas de observación y análisis del ciclo, recomenzando una nueva iteración del mismo.

2.5. Desarrollo de la propuesta

Hoy día la variedad de tráfico de datos en las redes de comunicación y la información generada por diferentes tipos de aplicaciones se ha incrementado. Por otro lado, para algunos tipos de tráficos, es vital alcanzar una determinada QoS, definida por los tiempos de transmisión, la pérdida de paquetes, entre otras cosas. Por ello es necesario realizar tareas de análisis para llegar a una adecuada y precisa identificación, clasificación y tratamiento de dicho tráfico, haciendo uso de técnicas de DPI, las cuales permiten inspeccionar detalladamente los paquetes, con el fin de indicar la aparición de patrones [26] [33]. En este trabajo se propone mejorar la QoS en las redes de comunicación. En particular, se definen tareas de AdD y técnicas de DPI para extraer el conocimiento necesario para dicho proceso. En ese sentido, se plantean dos ciclos autonómicos de tareas de AdD:

a) CA de mejora de la QoS, el cual tiene como objetivo optimizar la QoS usando técnicas de DPI, las cuales permiten determinar los tramos de red y tipos de paquetes que necesitan ser priorizados.
b) CA de aprendizaje de tipos de tráfico, cuyo objetivo es descubrir nuevos patrones de tráfico en los paquetes, y determinar para cada nuevo patrón, que herramienta DPI debería ser utilizada para su análisis.

Los dos CAs son complementarios, en particular, la salida del segundo alimenta al primero. La especificación de los ciclos autonómicos será presentada a continuación.

2.6. Especificación de los ciclos autónomos propuestos en este trabajo

En este trabajo se proponen dos CAs de tareas de AdD para gestionar los problemas de QoS para el tráfico en Internet. Cada CA se compone de un grupo de tareas de AdD agrupadas según los roles que cumplen en el mismo (monitoreo, análisis, o toma de decisiones). Algunas de las tareas son de análisis de datos vinculadas al tráfico (agrupación, clasificación, asociación, etc.), y otras de análisis de datos sociales (enriquecimiento semántico, procesamiento de texto, etc.). Dependiendo del tipo de tarea de AdD, las técnicas a usar son diferentes, por ejemplo, de minería de datos, enlazado de datos, minería de texto, etc. Por otro lado, la relación entre los dos CAs ocurre en la etapa de análisis del CA de Mejora de QoS, cuando el patrón de tráfico actualmente analizado no coincide con ninguno de los patrones de tráfico conocidos. En ese momento, ese ciclo invoca al CA de aprendizaje, para determinar y aprender ese nuevo patrón. A continuación, se describe cada ciclo.

a) Especificación del ciclo autónomo para mejorar la QoS.
Este CA tiene como objetivo mejorar la QoS del tráfico sobre internet (ver figura 5). Este ciclo se compone del siguiente grupo de tareas de AdD:

- Tareas de monitoreo.
Está compuesta por las tareas que permiten la detección e identificación de paquetes en la red, así como el estudio de los informes de incidencia realizados por los usuarios. Estas tareas nos permiten verificar el estado de la red y generar alarmas.
- Tareas de análisis.
Permiten analizar el actual contexto en la red. Entre otras cosas, hay tareas que permiten realizar la clasificación de los paquetes, para determinar el tipo de aplicación que el usuario está consumiendo. Agrupa los paquetes de acuerdo a patrones de tráfico, basándose en la información contenida en las trazas. También, hay tareas de enlazado de datos (linked data), para determinar problemas de QoS vinculados a los patrones de tráfico encontrados en la red. Con esta información recogida de caracterización del tráfico, por los procesos de enlazado de datos, se realiza/ actualiza el diseño de las reglas de optimización de la QoS, para conseguir el mejor rendimiento de la red.
- Tareas de toma de decisiones.
Son tareas que tienen como objetivo determinar las optimizaciones a realizar en la red, de tal manera de mejorar la QoS del tráfico. Para ello, usa un conjunto de reglas que caracterizan los problemas de QoS en las redes, las cuales son instanciadas con la información generada por las tareas anteriores.

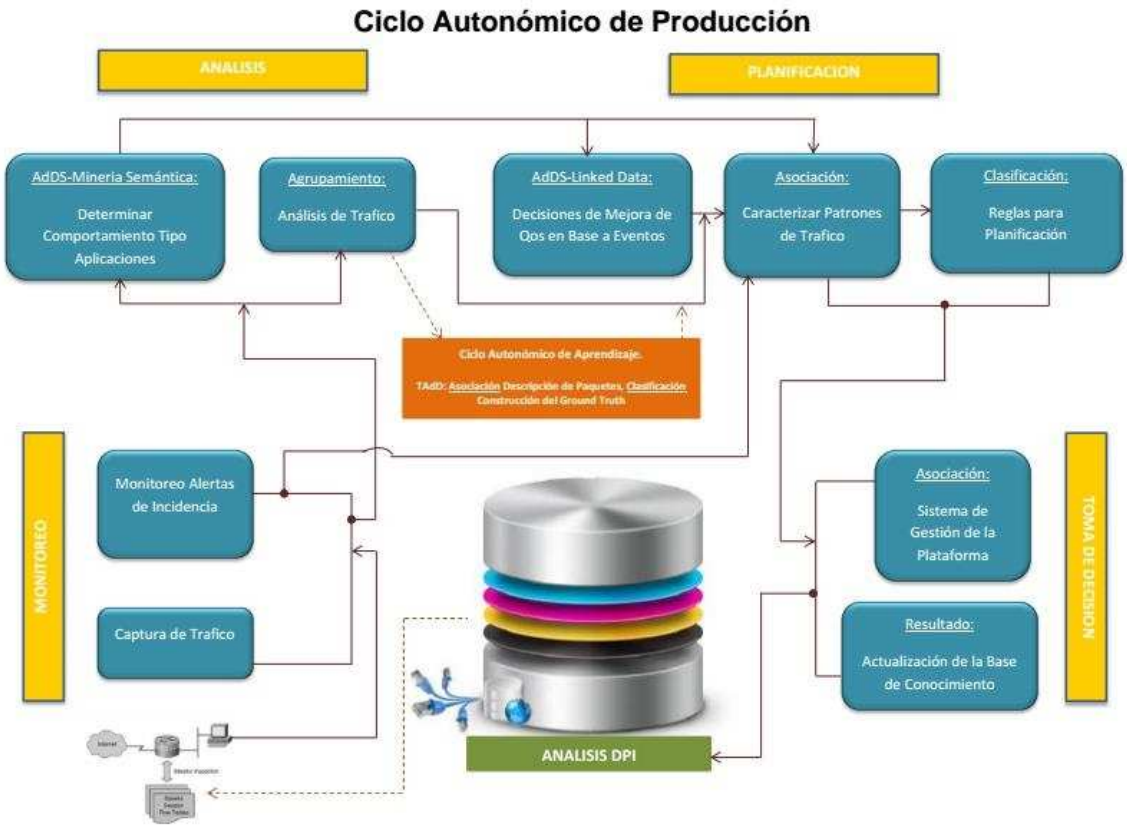

Figura 5: Ciclo Autonómico para mejorar la QoS (elaboración propia)

b) Especificación del ciclo autonómico de aprendizaje de tráfico.
El CA de aprendizaje de tráfico está compuesto por las siguientes tareas (ver figura 6):

- Tareas de monitoreo.
  Son las tareas que permiten detectar e identificar el tráfico en la red.
- Tareas de análisis.
  Son tareas que recogen las características específicas de los paquetes que están circulando en la red, y los protocolos asociados a cada paquete, usando diferentes aplicaciones DPI. Además, estas tareas preparan un informe sobre los diferentes tipos de aplicaciones DPI. A partir de los informes, se realiza un análisis comparativo de las diferentes herramientas DPI, para determinar la mejor (más precisa) para el análisis del patrón de tráfico estudiado. La escogencia de la mejor herramienta se basa en quien genera los menores falsos positivos y falsos negativos.
- Tareas de toma de decisiones.
  Con las herramientas DPI seleccionadas, se pasa a construir el patrón de tráfico bajo estudio.

2.7. Implementación del ciclo autónomo para mejorar la QoS

En esta sección, tomamos dos tareas de AdD del primer CA, para desarrollar una versión simplificada del mismo:

a) Tarea para determinar comportamiento de la aplicación (AdDs-Minería Semántica).
Para determinar el comportamiento de la aplicación que está usando un usuario, se realiza un análisis de los contenidos de las trazas (ver tabla 2). El análisis del contenido, permite determinar que está ocurriendo en un momento dado en Internet.
Las tareas con las que se relaciona son las de enlazado de datos (AdDS-Linked Data) para enriquecer semánticamente la información obtenida (buscando información específica vinculada a las características determinadas en el

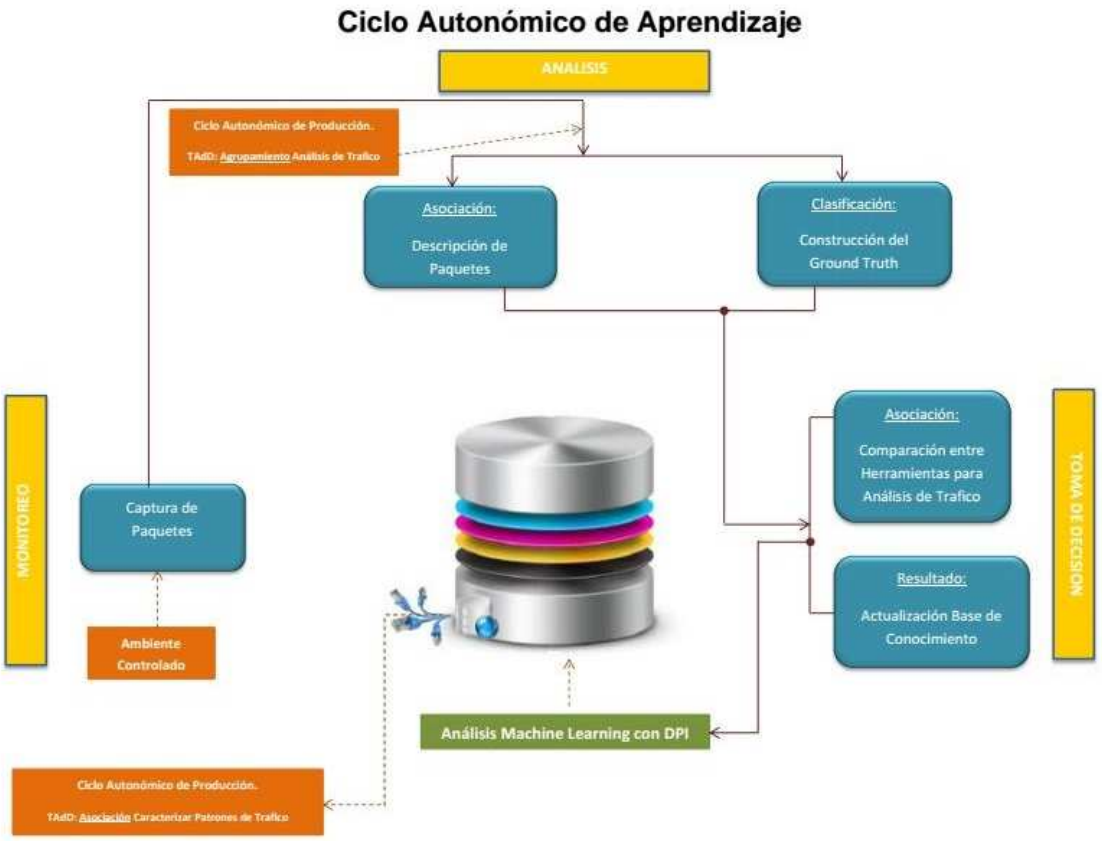

Figura 6: Ciclo autonómico de aprendizaje (elaboración propia)

Cuadro 2: Tarea que determina tipo de aplicación

Tarea de AdD	Determinar comportamiento de aplicación
Fuente de datos	Redes sociales, hashtag, etc.
Técnica de analítica de datos	AdD Minería Semántica.
Tipo de tarea de analítica de Datos	Asociación
Tipo de tarea del ciclo autonómico	Análisis

comportamiento de las trazas), y con la de asociación, que permite construir el patrón del tráfico actual en la red, para después buscar con que patrones conocidos se empareja (posteriores etapas, ver figuras 5 y 7). Específicamente busca determinar el comportamiento de las trazas, para poder tratarlo de la manera adecuada en las siguientes fases, para mejorar su QoS.

b) Tarea para caracterizar patrones de tráfico (Asociación).
Caracteriza los distintos paquetes, según la información que contienen las trazas (IP destino, IP fuente, tamaño, número de paquetes), la cual fue extraída por las herramientas DPI (ver tabla 3). Con este insumo, es posible calcular las métricas de rendimiento de los paquetes. Esto es fundamental, para diseñar el patrón del tráfico actual, y determinar las reglas de optimización de la QoS a usar.
Las tareas de AdD son (ver figura 8) las siguientes: a) Clasificación, usa el patrón parcial descubierto en esta fase para instanciar las diferentes reglas de gestión de la red de comunicación; b) Actualización de la base de conocimiento, actualiza la información en dicha base con ese patrón parcial del tráfico actual; y c) Sistema de gestión de la plataforma, que busca optimizar su funcionamiento.

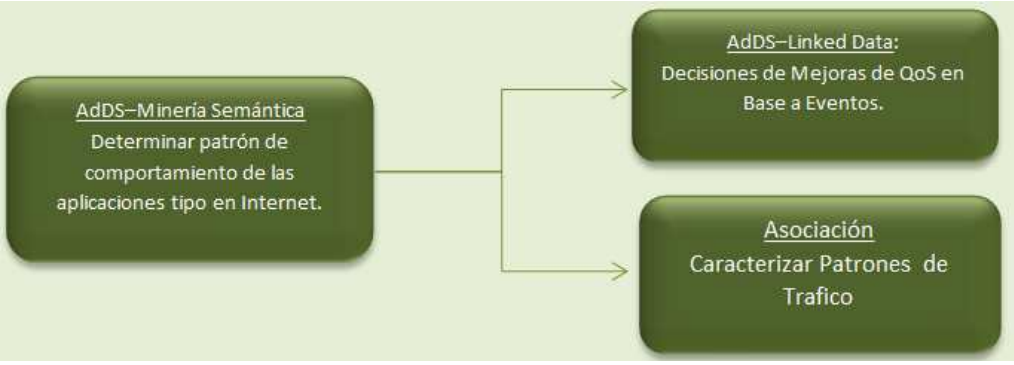

Figura 7: Sub-tareas relacionadas con la tarea comportamiento aplicación (elaboración propia)

Cuadro 3: Tarea para caracterizar patrones de tráfico

Tarea de AdD	Determinar comportamiento de aplicación
Fuente de datos	Captura a nivel lógico: PCAP. Reporte de herramientas DPI.
Técnica de analítica de datos	Reglas de asociación.
Tipo de tarea de analítica de datos	Asociación
Tipo de tarea del ciclo autonómico	Planificación.

2.8. Implementación de las tareas de AdD

Para las tareas de AdD anteriores, se desarrolló un programa en el lenguaje Python. Además, se usó Tweepy [11], la biblioteca de Python para acceder a la API de Twitter, y la Librería Panda [37] que realiza análisis de datos. Por otro lado, para caracterizar los patrones de tráfico se usa nDPI (herramienta que realiza la DPI) [33], Wireshark [38], que es un analizador de protocolos, usado para obtener el porcentaje de pérdidas en la traza y Weka como herramienta de minería de datos [39].

Figura 8: Sub-Tareas a relacionar con la tarea caracterizar patrones tráfico (elaboración propia)

a) Determinar el comportamiento de las aplicaciones.

Para dar un ejemplo de esta tarea de AdD, se supondrá que se usa Twitter como la fuente de información desde las redes sociales, pudiéndose usar otras fuentes. El Macro-algoritmo de esta tarea de AdD es:

- Capturar los tweets
- Filtrar los tweets capturados
- Generar gráficos basados en los datos capturados

La descripción del algoritmo es la siguiente: El primer paso es el proceso de captura de tweets. Se implemento un programa en Python, usando el API Tweepy.Mediante la API se conecta al flujo de Twitter, y captura los tweets generados. Para reducir la cantidad de tweets capturados, se filtran con palabras claves, como *redula* o *addpi_redula*

b) Caracterizar patrones de tráfico.

El Macro-algoritmo de esta tarea de AdD es:

(a) Capturar las trazas (PCAP)
(b) Calcular las métricas que evalúan la QoS usando las herramientas DPI
(c) Obtener distintas métricas
(d) Reportar comportamientos de las herramientas DPI
(e) Obtener reglas de asociación

Las trazas fueron proporcionadas por la red de comunicaciones de la Universidad de Los Andes, con un tamaño de aproximadamente 9GB. Esas trazas se pasaron por la herramienta de DPI, y se obtuvo un reporte sobre el análisis de ellas. Luego se generaron las métricas. Los atributos presentes inicialmente en dicho archivo son descriptores de la traza (ver tabla 4).

Cuadro 4: Algunos atributos de las trazas

Label	Protocolo de la aplicación
protocol	Identificador del protocolo de la aplicación
fpackets	Número total de paquetes enviados desde nodo inicial hasta nodo final
fbytes	Número total de bytes enviados desde nodo inicial hasta nodo final
bpackets	Número total de paquetes enviados desde nodo final hasta el nodo inicial
bbytes	Número total de bytes enviados desde nodo final hasta el nodo inicial
minfpktl	Mínimo tamaño de payload enviado desde nodo inicial hasta nodo final
Meanfpkt	Media del tamaño del payload enviado desde el nodo inicial hasta nodo final
Maxfpktl	Máximo del tamaño del payload enviado desde nodo inicial hasta el nodo final
Stdfpktl	Desviación estándar del tamaño del payload enviado desde el nodo inicial hasta el nodo final
Minbpktl	Mínimo tamaño del payload enviado desde nodo final hasta el nodo inicial
Stdbpktl	Desviación estándar del tamaño del payload enviado desde el nodo final al nodo inicial
minfiat	Tiempo mínimo de interconexión de paquetes para paquetes enviados desde el nodo inicial al nodo final
meanfiat	Tiempo medio de interconexión de paquetes para paquetes enviados desde el nodo inicial al nodo final
stdfiat	Desviación estándar del tiempo de interconexión de paquetes enviados desde el nodo inicial al nodo final
meanbiat	Tiempo medio de interconexión de paquetes para paquetes enviados desde el nodo final al nodo inicial
stdbiat	Desviación estándar del tiempo de interconexión de paquetes enviados desde el nodo final al nodo inicial
duration	Duración del flujo (en microsegundos)
timestamp	Tiempo de inicio del flujo

Con la información de las trazas, se realiza el cálculo de las métricas que evalúan la QoS. En específico, se calcula:

- Tasa de pérdida: para determinar esta métrica, se usó wireshark, el cual proporciona el número total de pérdidas en toda la traza, al usar el filtro tcp.analysis.lost_segment (ver figura 9).
- Retardo: en este caso se usaron las variables meanfiat y meanbiat, las cuales permiten determinar el tiempo promedio de duración del paquete viajando en la red desde el nodo de inicio hasta el nodo final, y viceversa:

$$retardoEmisor = meanfiat \tag{2.1}$$

$$retardoReceptor = meanbiat \tag{2.2}$$

- Rendimiento: viene dado por la cantidad de datos enviados o recibidos, fbytes y bbytes, en el tiempo promedio de duración del viaje de los paquetes de un nodo a otro.

$$rendimientoEmisor = \frac{fbytes}{meanfiat} \tag{2.3}$$

$$rendimientoEmisor = \frac{bbytes}{meanbiat} \tag{2.4}$$

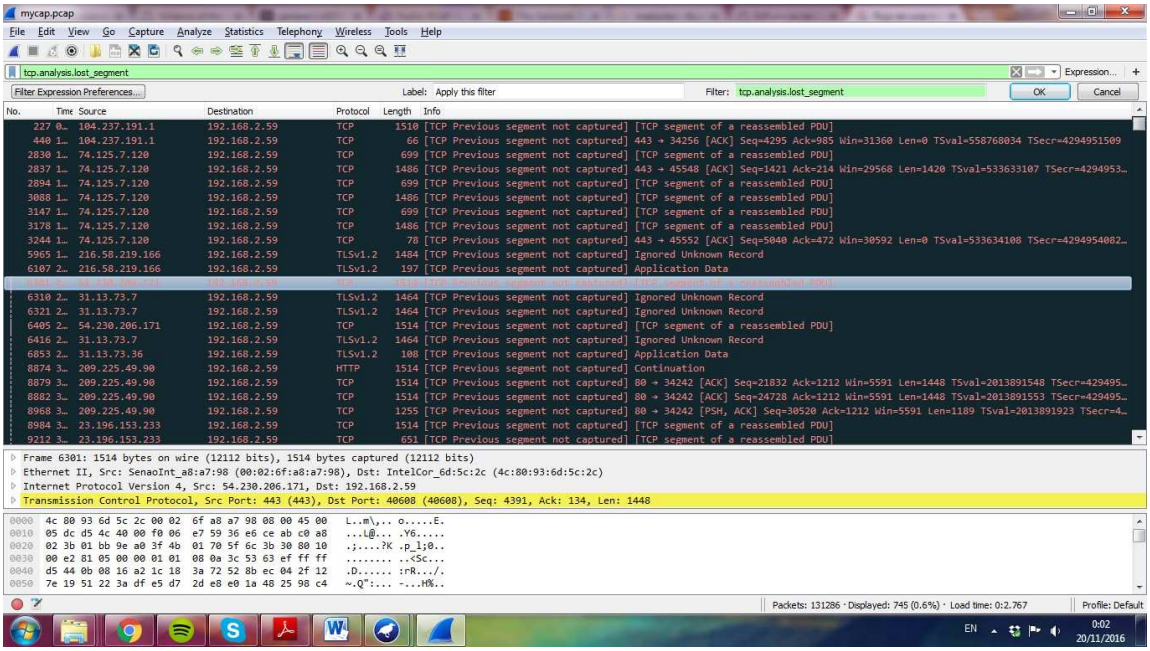

Figura 9: Tasa de pérdida según wireshark

Una vez se obtuvo las métricas, se obtienen los distintos protocolos usados por las aplicaciones. Con esto último, se tiene suficiente información para analizar el comportamiento y la eficiencia de las diferentes herramientas DPI. Finalmente, se obtienen reglas de asociación usando la herramienta de AdD llamada Weka [39]. En particular, se usó el algoritmo Apriori para generar las reglas de asociación (ver siguiente sección).

2.9. Análisis de los resultados

a) Determinar comportamiento de las aplicaciones.
Para este estudio se activó el programa de captura de tweets, y se filtraron los tweets usando la palabra clave Redula y las palabras especificadas en la sección anterior para hacer el filtrado. Los resultados obtenidos son mostrados en las figuras 10 y 11.
Con ese tipo de información, es posible construir una primera aproximación del comportamiento de las trazas (patrón). Vemos que, en lapso de tiempo de 4 horas, y con no más de 5000 tweet de los usuarios sobre la red de datos de la Universidad de los Andes, se lograron obtener resultados que nos permiten hacer una primera determinación de la QoS de la red. Esta tarea de analítica de datos social captura datos (tweets) en línea, para analizar la información en ellos. Por ejemplo, la queja por una velocidad lenta, indica que se está usando una aplicación que requiere mejorar su QoS (por ejemplo, de transmisión de videos). Además, si se detecta una irregularidad, por ejemplo, un 60 % de los usuarios reportando la red caída o lentitud en un servicio en específico (videos, servidores caídos, etc.), permite emitir un evento a las otras tareas del ciclo autonómico, con la finalidad de resolver el problema.

b) Caracterizar patrones de tráfico.
Esta tarea determina un conjunto de reglas sobre el comportamiento de las trazas basada en criterios de rendimiento (la figura 12 muestra las mejores reglas). Para determinar las mejores reglas, se usaron las siguientes métricas de Weka [39].

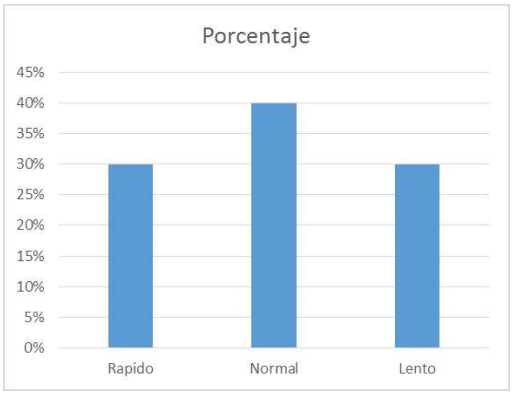

Figura 10: Calidad de servicio de red según los clientes (elaboración propia)

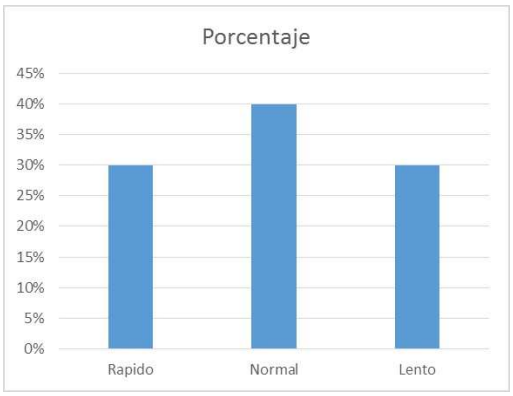

Figura 11: Velocidad de la red según los twitterm (elaboración propia)

- Lift: es la relación entre la probabilidad de que ocurran ambos lados de la regla, y la probabilidad de que ocurra cada lado de la regla individualmente. Es deseable ver valores mayores a 1, ya que 1 indica que las variables son totalmente independientes, mientras mayor es el valor es más probable que un lado ocurra dado el otro lado de la regla.
- Leverage: corresponde a la diferencia entre la probabilidad de que ocurran ambos lados de la regla, y la probabilidad de que ocurran ambos lados individualmente. Mide la proporción de casos cubiertos por ambos lados de la regla, adicionales a los esperados si ambos lados de la regla fueran independientes. Se esperan valores mayores a 0.
- Conviction: es parecido al lift, pero mide el efecto si el lado derecho de la regla no es verdad.

  Además de las anteriores, se usaron otras dos métricas clásicas para calcular la calidad de una regla:
- Soporte: se define como la proporción de ejemplos/instancias que cubre esa regla.
- Confianza: es la probabilidad de sí, al encontrar la parte derecha de una regla se encuentra también la parte izquierda.

Las reglas mostradas en la figura 11, son las que tienen los mejores valores en todas esas métricas de calidad. Las reglas permiten describir desde el punto de vista de criterios de rendimiento, el tráfico actual en un momento dado. Por ejemplo, una regla que señale que hay bajo rendimiento cuando hay poco retardo y mucha pérdida de paquetes, significa que a pesar de que hay poco retardo, existe una degradación de la QoS en la aplicación debido a la pérdida de paquetes. Este insumo es vital para determinar el patrón del tráfico en una situación dada, y poderlo hacer coincidir con los patrones conocidos que usan las otras etapas del ciclo autonómico.

En general, para determinar la calidad de las reglas que genera el algoritmo, es suficiente concentrarse en las métricas de soporte, confianza, y a veces, de lift o leverage. Del grupo de reglas que genera Weka en este caso de estudio, 26

```
Best rules found:

 1. brendimiento=bajo 549 ==> frendimiento=bajo 549    <conf:(1)>
lift:(1) lev:(0) [0] conv:(0)
 2. fpacketloss=mucho 506 ==> frendimiento=bajo 506    <conf:(1)>
lift:(1) lev:(0) [0] conv:(0)
 3. fpacketloss=mucho brendimiento=bajo 505 ==> frendimiento=bajo 505
<conf:(1)> lift:(1) lev:(0) [0] conv:(0)
 4. fretardo=poco 499 ==> fpacketloss=mucho 499    <conf:(1)> lift:(1.09)
lev:(0.07) [39] conv:(39.92)
 5. fretardo=poco 499 ==> frendimiento=bajo 499    <conf:(1)> lift:(1)
lev:(0) [0] conv:(0)
 6. fretardo=poco frendimiento=bajo 499 ==> fpacketloss=mucho 499
<conf:(1)> lift:(1.09) lev:(0.07) [39] conv:(39.92)
 7. fretardo=poco fpacketloss=mucho 499 ==> frendimiento=bajo 499
<conf:(1)> lift:(1) lev:(0) [0] conv:(0)
 8. fretardo=poco 499 ==> fpacketloss=mucho frendimiento=bajo 499
<conf:(1)> lift:(1.09) lev:(0.07) [39] conv:(39.92)
```

Figura 12: Mejores reglas obtenidas por la tarea caracterizar patrones de tráfico

tienen una confianza de 1, es decir, siempre que el lado izquierdo de la regla sucede, el lado derecho se cumple. Este valor es importante, porque representan reglas que nos permiten realizar la caracterización del tráfico.

3. Conclusiones

Los ciclos autonómicos de tareas de análisis de datos propuestos en este trabajo permiten la auto adaptación de las redes de comunicación para la mejora continua de la QoS que brindan. De esta manera, los ciclos autonómicos son mecanismos de reconfiguración de las redes de comunicación, con el fin de mejorar la QoS. Los ciclos autonómicos se comportan como esquemas de supervisión inteligente para mejorar el rendimiento de las plataformas comunicacionales. En el caso del ciclo de mejora de la QoS, al aparecer nuevas reglas debería hacer un llamado al ciclo de aprendizaje para determinar el patrón de tráfico de esta regla. Por otro lado, con la información de este ciclo se pueden proponer mejoras en la calidad de servicio usando técnicas de Inspección Profunda de Paquetes, para determinar los segmentos y tipos de paquetes que necesitan ser priorizados. Es importante resaltar que el ciclo autónomo que se implementó mezcla una tarea de análisis de datos sociales (minería de texto sobre tweets) para determinar el comportamiento de las aplicaciones, con una tarea de análisis de datos (regla de asociación) para construir los patrones de tráfico. Futuros trabajos pueden implementar el resto de tareas del ciclo autonómico, como también el otro ciclo autonómico para probar de manera integral a las dos propuestas de este trabajo. Dichas pruebas pueden realizarse en diferentes contextos operacionales, considerando diferentes tipos de aplicaciones.

Referencias

[1] J. Aguilar; O. Buendia; K. Moreno; D. Mosquera. Autonomous cycle of data analysis tasks for learning processes. In *Second International Conference on Technologies and Innovation (CITI)*, pages 187–202, Guayaquil, Ecuador, 2016. Springer International Publishing.

[2] T. Nguyen; G. Armitage. A survey of techniques for internet traffic classification using machine learning. *IEEE Communications Surveys & Tutorials*, 10(4):56–76, 2008.

[3] M. Soysal; E. Schmidt. Machine learning algorithms for accurate flow-based network traffic classification: Evaluation and comparison. *Performance Evaluation*, 67(6):451–467, 2010.

[4] M. Roughan; S. Sen; O. Spatscheck; N. Duffield. Class-of-service mapping for qos: a statistical signature-based approach to ip traffic classification. In *4th ACM SIGCOMM Conference on Internet measurement*, pages 135–148, New York, USA, 2004. ACM.

[5] R. Sommer; V. Paxson. Outside the closed world: On using machine learning for network intrusion detection. In *Security and Privacy, IEEE Symposium on*, pages 305–316. IEEE, 2010.

[6] D. Le; N. Zincir-Heywood; M. Heywood. Data analytics on network traffic flows for botnet behaviour detection. In *Computational Intelligence (SSCI), IEEE Symposium Series on*, pages 1–7. IEEE, 2016.

[7] J. Aguilar; O. Buendía; J. Cordero. Specification of the autonomic cycles of learning analytic tasks for a smart classroom. *Journal of Educational Computing Research*, 2017.

[8] F. Pacheco; J. Aguilar; C. Rangel; M. Cerrada; J. Altamiranda. Methodological framework for data processing based on the data science paradigm. In *Computing Conference (CLEI), XL Latin American*, pages 1–12. IEEE, 2014.

[9] C. Rangel; F. Pacheco; J. Aguilar; M. Cerrada; J. Altamiranda. Methodology for detecting the feasibility of using data mining in an organization. In *Computing Conference (CLEI), 2013 XXXIX Latin American*, pages 502–513. IEEE, 2013.

[10] G. Riofrio; E. Encalada; D. Guamán; J. Aguilar. Business intelligence applied to learning analytics in student-centered learning processes. In *Computing Conference (CLEI), Latin American*, pages 567–576. IEEE, 2014.
[11] Tweepy API. Libreria python para accesar la api de twitter. http://www.tweepy.org/, Última consulta 26 de mayo de 2018.
[12] R. Goss; R. Botha. Establishing discernible flow characteristics for accurate, real-time network protocol identification. In *INC*, pages 25–34, 2012.
[13] Y. Hadjadj-Aoul; K. Singh; S. Vaton; M. Sbai; S. Moteau. Technical specifications of the modules proposed within the wp2: Traffic classifier, qoe evaluator and their integration, technical report, projet vipeer ingénierie du trafic vidéo en intradomaine basèe sur les paradigmes du pair à pair. 2011.
[14] M. Crotti; F. Gringoli; P. Pelosato; L. Salgarelli. A statistical approach to ip-level classification of network traffic. In *Communications, ICC'06. IEEE International Conference on*, volume 1, pages 170–176. IEEE, 2006.
[15] V. Paxson. Empirically derived analytic models of wide-area tcp connections. *IEEE/ACM Transactions on Networking (TON)*, 2(4):316–336, 1994.
[16] T. Karagiannis; K. Papagiannaki; M. Faloutsos. Blinc: multilevel traffic classification in the dark. In *ACM SIGCOMM computer communication review*, volume 35, pages 229–240. ACM, 2005.
[17] C. Dewes; A. Wichmann; A. Feldmann. An analysis of internet chat systems. In *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, pages 51–64. ACM, 2003.
[18] A. Arcia-Moret; A. Araujo; J. Aguilar; L. Molina; A. Sathiaseelan. Intelligent network discovery for next generation community wireless networks. In *12th conference on Wireless On-demand Network systems and Services Conference*, pages 81–87, Italy, 2016. IEEE.
[19] H. Kim; K. Claffy; M. Fomenkov; D. Barman; M. Faloutsos; K. Lee. Internet traffic classification demystified: myths, caveats, and the best practices. In *Proceedings of the 2008 ACM CoNEXT conference*, page 11. ACM, 2008.
[20] A. Arcia-Moret; A. Sathiaseela; A. Araujo; J. Aguilar; L. Molina. Assisted network discovery for next generation wireless networks. In *Consumer Communications & Networking Conference (CCNC), 13th IEEE Annual*, pages 800–801. IEEE, 2016.
[21] A. Moore; D. Zuev. Internet traffic classification using bayesian analysis techniques. In *ACM SIGMETRICS Performance Evaluation Review*, volume 33, pages 50–60. ACM, 2005.
[22] D. Nechay. *Controlling False Alarm/Discovery Rates in Online Internet Traffic Classification*. PhD thesis, McGill University, 2009.
[23] J. Tan; X. Chen; M. Du; K. Zhu. A novel internet traffic identification approach using wavelet packet decomposition and neural network. *Journal of Central South University*, 19(8):2218–2230, 2012.
[24] B. Yamansavascilar; M. Guvensan; A. Yavuz; M. Karsligil. Application identification via network traffic classification. In *Computing, Networking and Communications (ICNC),International Conference on*, pages 843–848. IEEE, 2017.
[25] J. Tan; X. Chen; M. Du. An internet traffic identification approach based on ga and pso-svm. *JCP*, 7(1):19–29, 2012.
[26] U. Trivedi; M. Patel. A fully automated deep packet inspection verification system with machine learning. In *Advanced Networks and Telecommunications Systems (ANTS), IEEE International Conference on*, pages 1–6. IEEE, 2016.
[27] M. Shafiq; X. Yu; A. Laghari; L. Yao; N. Karn; F. Abdesssamia. Wechat text and picture messages service flow traffic classification using machine learning technique. In *High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS), IEEE 18th International Conference on*, pages 58–62. IEEE, 2016.
[28] M. Shafiq; X. Yu; A. Laghari; L. Yao; N. Karn; F. Abdessamia. Network traffic classification techniques and comparative analysis using machine learning algorithms. In *Computer and Communications (ICCC), 2nd IEEE International Conference on*, pages 2451–2455. IEEE, 2016.
[29] L. Caviglione; N. Celandroni; M. Collina; H. Cruickshank; G. Fairhurst; E. Ferro; A. Gotta; M. Luglio; C. Roseti; A. Abdel Salam; A. Vanelli. A deep analysis on future web technologies and protocols over broadband geo satellite networks. *International Journal of Satellite Communications and Networking*, 33(5):451–472, 2015.
[30] K. Alhamazani; R. Ranjan; P. Jayaraman; K. Mitra; M. Wang; Z. Huang; L. Wang; F. Rabhi. Real-time qos monitoring for cloud-based big data analytics applications in mobile environments. In *Mobile Data Management (MDM), IEEE 15th International Conference on*, volume 1, pages 337–340. IEEE, 2014.
[31] Pace: Dpi engine. https://www.ipoque.com/en/products/pace, Ultima consulta 26 de mayo de 2018.
[32] SolarWinds. Solarwinds network performance monitor. http://www.solarwinds.com/network-performance-monitor, Ultima consulta 26 de mayo de 2018.
[33] NDPI: herramienta de dpi. https://github.com/ntop/nDPI/issues/249, Última consulta 26 de mayo de 2018.
[34] L. Deri; M. Martinelli; T. Bujlow; A. Cardigliano. ndpi: Open-source high-speed deep packet inspection. In *Wireless Communications and Mobile Computing Conference (IWCMC), International*, pages 617–622. IEEE, 2014.
[35] P. C. Lin; Y. D. Lin; Y. C. Lai; T. H. Lee. Using string matching for deep packet inspection. *Computer*, 41(4):23–+, 2008.
[36] K. Kyriakopoulos; D. Parish; J. Whitley. Flowstats: An ontology based network management tool. In *Computing Technology and Information Management (ICCTIM), Second International Conference on*, pages 13–18. IEEE, 2015.
[37] Pandas API. Libreria de python para analítica de datos. http://pandas.pydata.org/, Ultima consulta 26 de mayo de 2018.
[38] Wireshark project. Analizador de paquetes de red gratuito y abierto. https://www.wireshark.org/, Ultima consulta 26 de mayo de 2018.
[39] Weka 3: Data mining software in java. https://www.cs.waikato.ac.nz/ml/weka/, Última consulta 26 de mayo de 2018.

Sobre los autores

José Aguilar
Profesor- investigador de la Universidad de los Andes, Venezuela. PhD en Informática de la Universidad Renne Descartes, Francia. Maestría en Informática, Paul Sabatier, Toulouse Francia. Correo: aguilar@ula.ve - ORCID

Kristell Aguilar
Estudiante de Maestría en Informática en la Université de Pau et des Pays de l'Adour, Francia. Ingeniero de Sistemas de la Universidad de Los Andes, Mérida Venezuela. Correo: kristell153@gmail.com

Marxjhony Jerez
Profesor- investigador de la Universidad de los Andes, Venezuela. Estudiante de Doctorado. Maestría en Computación de la Universidad de Los Andes, Mérida, Venezuela. Ingeniero de Sistemas de la Universidad de Los Andes, Mérida, Venezuela. Correo: marxjhony@ula.ve - ORCID

Carlos Jiménez
Estudiante de Ingeniería de Sistemas de la Universidad de Los Andes, Mérida Venezuela. Correo: solracjh15@gmail.com