Communication Network Model for a Computer Management and Control System implemented using FIWARE platform: Case Study

Washington Velasquez, *Member, IEEE,* Margarita Filian-Gomez

***Abstract*—In the last two decades, different organizations have used computing management resources systems based on client-server architecture. The Escuela Superior Politécnica del Litoral uses a computer management system; which has been adapted to versions Windows OS to manage computer labs. However, these adaptations have not considered improvements in the system architecture, to correct pre-existing communication problems. This paper states a solution based on a distributed model of computer management using the architecture of the FIWARE platform, to provide a smart system that converts a computer on an IoT device to improve the system communication. Finally, communication tests show that the proposed model can be implemented on a large scale, due to the distributed architecture, resources management, and real-time processing to manage events.**

***Index Terms*—Architecture, Computer, Draco, FIWARE, Management, Orion, System.**

I. Introducción

Sistemas de administración de computadoras que utilizan una arquitectura cliente-servidor; han sido ampliamente utilizados en los últimos veinte años [1], [2]. Este diseño es empleado por diferentes organizaciones como universidades, ministerios, y centros de investigación con la finalidad de que sus usuarios puedan utilizar diferentes recursos informáticos [3]. La Escuela Superior Politécnica del Litoral (ESPOL) utiliza un sistema de administración de computadoras que nació a través de un tema de tesis de pregrado entre los años 2000-2003 para el S.O. Windows XP llamado *"Sistema multi-plataforma para la administración de laboratorios de computación* (SMALC)" [4]. En el 2009, con la aparición de nuevos sistemas operativos como Windows Vista, y Windows 7 [5]; la arquitectura diseñada para ejecutarse en Windows XP presentó algunos problemas de comunicación, e.g., validar el tiempo disponible que un usuario utiliza una computadora, siendo un problema relacionado con el esquema de "autenticación y sesión de usuario" utilizado en estas versiones de Windows [6].

SMALC se adaptó a las nuevas plataformas de sistemas operativos; mejorando la comunicación y agregando nuevas funcionalidades de administración (https://www.fiec.espol.edu.ec/es/proyectos-implementados).

Washington Velasquez - Escuela Superior Politécnica del Litoral, Campus Gustavo Galindo Km 30.5 Vía Perimetral, P.O. Box 09-01-5863, Guayaquil, Ecuador, e-mail: wavelasq@espol.edu.ec

Margarita Filian-Gomez - Escuela Superior Politécnica del Litoral, Campus Gustavo Galindo Km 30.5 Vía Perimetral, P.O. Box 09-01-5863, Guayaquil, Ecuador, e-mail: mfilian@espol.edu.ec

SMALC esta actualmente instalado en diferentes ubicaciones dentro de la ESPOL, como en la Facultad de Ingeniería en Eléctricidad y Computación (FIEC), el Centro de Información Bibliotecaria (CIB) y la Facultad de Ciencias Sociales y Humanísticas (FCSH). Sin embargo, no hay un servidor centralizado que gestione todos estos recursos; al contrario, se debe instalar un servidor en cada ubicación (unidad o facultad) para administrar un conjunto de computadoras, e.g., FIEC tiene un servidor que maneja las solicitudes de alrededor de 400 computadoras distribuidas entre todos sus laboratorios. El diseño implementado en cada ubicación se muestra en la Fig. 1, donde se puede visualizar tres aplicaciones (cliente, servidor y aplicación de administración), desarrolladas en diferentes lenguajes de programación que cumplen una función específica dentro del sistema. Este diseño presenta algunas desventajas considerando la infraestructura tecnológica actual de la ESPOL, e.g., computadoras específicas destinadas a ejecutarse como un servidor en diferentes ubicaciones, personal dedicado a la administración del sistema y, lo más importante, el aumento en el flujo de datos procesados por cada comunicación entre *cliente-servidor* que causa la pérdida de paquetes en algunas funciones del sistema.

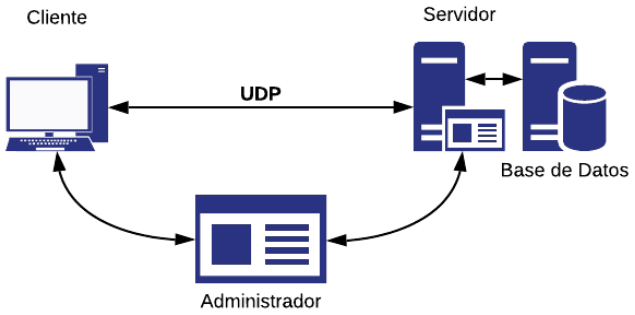

Fig. 1. Arquitectura SMALC implementada en cada ubicación

Este artículo describe un modelo de comunicación distribuido para la administración de computadoras basado en tecnologías de la plataforma FIWARE [7], con la finalidad de proporcionar un sistema inteligente que transforme a una computadora en un dispositivo de Internet de las Cosas (IdC), capaz de reconocer fallas, sucesos o incidencias que permitan alertar a los administradores de las facultades de manera inmediata. Además, la comprobación del modelo propuesto se lo realiza en un entorno de laboratorio virtual (simulación del sistema) que envía mensajes en diferentes

formatos utilizando un esquema basado en actores (Threads), proporcionando ventajas en la comunicación con respecto al modelo cliente-servidor.

Este documento se divide en cinco secciones: Sección II describe el estado del arte relacionado con el esquema actual del sistema SMALC, los componentes y problemas que han surgido con cada actualización. Sección III presenta la nueva arquitectura general del sistema que describe el esquema de comunicación, las tecnologías y el modo de uso dentro de la ESPOL. Sección IV presenta una prueba preliminar de concepto, que describe un escenario utilizado para el envío de mensajes entre los componentes del sistema. Por último, se presenta una discusión del modelo y la viabilidad de este esquema para ser incorporado a gran escala.

II. SMALC

Un sistema de gestión y control informático puede ser ampliamente utilizado en un cibercafé. [8]. Estos permiten realizar diferentes funciones, e.g., administrar el tiempo de un usuario, control remoto (inicio de sesión, bloqueo, apagado y reinicio), e incluso, la contabilidad empresarial en software especializado. Sin embargo, Internet se ha convertido en un servicio de necesidad básica, haciendo que estos servicios se trasladen a zonas urbano marginales de países en desarrollo [9]. SMALC encaja en la categoría de estos sistemas, sin embargo, realiza funciones específicas en el ámbito académico. El objetivo de SMALC es controlar el acceso a las computadoras, el tiempo disponible para cada usuario, y administrar y planificar las diferentes actividades que se realizan dentro de laboratorios de computación. [4]. Este proyecto se instaló inicialmente en la FIEC, permitiendo administrar lo siguiente:

- Préstamo de computadoras a estudiantes - reconociendo el tipo de licenciatura o diferentes carreras de grado.
- Préstamo de laboratorios destinado a clases, seminarios y talleres.
- Validación de los tiempos y usuarios en todo momento dentro de un laboratorio designado para tareas específicas.
- Administración y control de computadoras (tiempos, usuarios, control remoto de una computadora).
- Chat de mensajería (usuario-administrador).
- Monitoreo en tiempo real de las computadoras.

SMALC consta de tres componentes (Fig. 1): cliente, servidor y aplicación de administración.

- Cliente: Aplicación desarrollada en lenguaje de programación *C#* que se instala en cada computadora de los laboratorios.
- Servidor: Recibe las peticiones de los usuarios y las acciones que solicite realizar el administrador. Es una aplicación desarrollada en lenguaje de programación *Java* que maneja sesiones de usuarios; a través de una estructura de árbol binario, que le permite realizar una búsqueda basada en direccionamiento. En la Fig. 2, se muestra el algoritmo de búsqueda binaria, donde se utiliza el último octeto de la dirección IP para su incorporación al árbol binario, esta acción corresponde a una nueva sesión de usuario en el sistema [4]. Asimismo, el servidor se encarga de la administración y préstamo de las computadoras a estudiantes, y gestionar los horarios de los cursos/talleres que se imparten en las diferentes facultades de ESPOL.

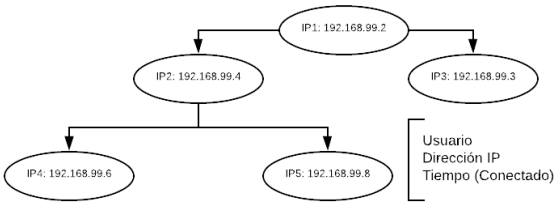

Fig. 2. Estructura de datos que gestiona las sesiones de los usuarios

- Administrador: Aplicación de escritorio desarrollada en lenguaje de programación *Java* para administrar todos los laboratorios y computadoras desde una ubicación (administrador) [4], [10]. La vista principal del sistema se muestra en la Fig. 3.

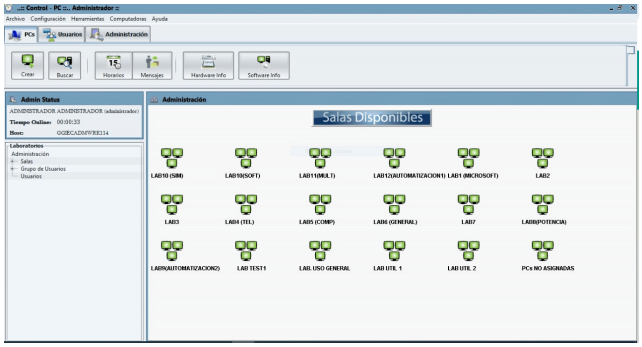

Fig. 3. Vista general de la aplicación de administrador.

A. Comunicación de Componentes SMALC

En la Fig. 4, el modelo de comunicación actual del sistema SMALC se muestra utilizando paquetes UDP en los sistemas operativos Windows [11], donde el cliente, el servidor y la aplicación de administración tienen puertos habilitados; *(SP: Puerto de envío, RP: Puerto de recepción)*, para establecer el flujo de comunicación. Las funcionalidades de mensajería de chat, monitoreo, envío y recepción de notificaciones (cliente-administrador) tienen puertos separados para evitar la saturación en los puertos de sesión del usuario *(SP, RP)*.

La Tabla I muestra la cantidad de computadoras para cada una de las ubicaciones donde está instalado el SMALC. Sin embargo, con el aumento de computadoras y laboratorios que actualmente posee la FIEC, y los cambios en la arquitectura del sistema operativo (Windows 10) [12], [13], ha habido algunos problemas de comunicación, e.g.:

- Pérdida de paquetes entre cliente-servidor.
- Validación del usuario (tiempo de uso diario).
- Monitoreo en tiempo real de las actividades del usuario.
- Fallos de autenticación del usuario.
- Fallas en las funciones básicas de control remoto.
- Envío y recepción de notificaciones.

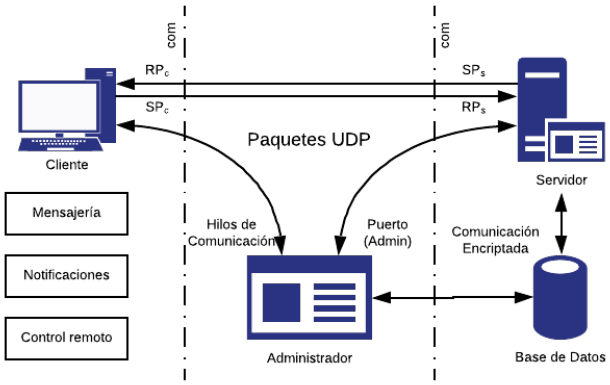

Fig. 4. Modelo de comunicación SMALC (cliente-administrador-servidor)

TABLA I
NÚMERO DE COMPUTADORAS DONDE SE UTILIZA SMALC

Ubicación	Cliente	Servidor	Administrador
FIEC	404	1	3
FCSH	80	1	1
Biblioteca General	100	1	3

B. Trabajos Relacionados

SMALC es un sistema de computo que provee funcionalidades de administración y control a laboratorios de computación [4], [14]. Este tipo de sistema ha tenido la necesidad de incorporar nuevas tecnologías que permitan establecer una mejor comunicación entre los elementos del sistema [15], mejorando la experiencia e interacción del usuario [16], [17]. Actualmente, la creciente demanda tecnológica y la necesidad de permanecer conectados ha llevado a que estos sistemas provean nuevas características, e.g., en [18] incorporan un sistema de acceso biométrico con lectores RFID para el acceso a un laboratorio de cómputo, y en [19] se presenta el desarrollo de una aplicación móvil para el acceso a los laboratorios, utilizando como medio de comunicación "Bluetooth" y hardware libre de bajo costo (Arduino). No obstante, las mejoras no solo se han dado en software de computación, también se han contemplado otros tipos de servicios para los usuarios, e.g., en [20] presentan un sistema inmótico empleado para la seguridad, comodidad e iluminación de los laboratorios de computación.

A pesar de todas las innovaciones tecnológicas y funciones incorporadas en los sistemas, cuando el incremento de dispositivos, y equipos de cómputo para el control y administración sobrepasa las capacidades de comunicación en un laboratorio, pueden ocasionar problemas de conexión entre cliente y servidor. Esto ha llevado el estudio de mejores técnicas de comunicación entre dispositivos, e.g., en [21] han desarrollado un prototipo de un sistema de comunicación utilizando un dron para ampliar la señal WiFi de los laboratorios de telecomunicaciones. Asimismo, en [22] presentan un sistema inalámbrico para la comunicación de datos dentro de un laboratorio. En este contexto, ésta propuesta se presenta como innovadora debido a que se pretende establecer un canal de comunicación en varias vías (cliente, servidor, y administrador) con la finalidad de incorporar el paradigma de IdC, técnicas de procesamiento de datos y aprendizaje automatizado en un sistema capaz de realizar todas las funciones actuales del SMALC, como también añadir componentes para analizar datos en tiempo real, preveer sucesos inesperados que permitan al administrador actuar de manera inmediata ante algún imprevisto.

III. ARQUITECTURA DEL MODELO DE COMUNICACIÓN

Para incorporar el paradigma IdC en los sistemas de gestión y control de computadoras, se deben contemplar algunas funciones adicionales para hacer que estos sistemas sean inteligentes. Las principales características necesarias para obtener este tipo de sistemas serían conectividad, análisis en tiempo real de los datos e integración con diferentes dispositivos independiente de la tecnología.

- Conectividad: El esquema de un sistema distribuido en varias aplicaciones, ofrece a un sistema la confiabilidad y robustez para la implementación a gran escala, permitiendo obtener:
  - Visualización: Monitoreo en tiempo real de todos los recursos del sistema, integrando todos los dispositivos en una plataforma web.
  - Mensajería de alta velocidad: Envío y recepción de paquetes dentro de la red, manteniendo un flujo constante que admite tolerancia a fallos.
  - Administración del sistema: Entidades, metadatos y estados del ciclo de vida de una computadora.
- Análisis en tiempo real: El sistema realiza el procesamiento en tiempo real de los datos generados por el sistema (usuarios y computadoras), lo que permite conocer eventos inesperados para el administrador, e.g.:
  - Procesamiento de flujo: Análisis en tiempo real de los flujos de datos entrantes con agregación, filtrado y correlación de eventos.
  - Enriquecimiento de datos (valor): Asigna un valor a los datos sin procesar con información contextual generando flujos compuestos.
  - Almacenamiento de eventos: Consulta y visualiza los datos almacenados permitiendo su análisis.
- Integración: Integra diferentes tecnologías para controlar dispositivos usando comandos, generando una respuesta óptima en aplicaciones.

Un esquema general del modelo de comunicación del sistema se presenta en la Fig. 5, donde el cliente trabaja en plataformas múltiples (Windows, MAC OS y Linux) estableciendo comunicación con el componente Fiware-Orion, y éste, dependiendo de los eventos redirige a Fiware-Draco para que realice cada una de las tareas específicas de gestión y control de computadoras, como del sistema en general. Asimismo, se contempla tener algunos *Kafka-brokers* para que cada tópico dentro de un *broker* se encargue de recopilar toda la información de una computadora, e.g., Una *computadora* posee un tópico específico *(PC1)* en el *Broker-0* (FIEC) para recibir esa información. Por otro lado, Fiware-Draco actúa como un sumidero para el flujo de datos al enviar toda la información al administrador, Kafka, y un componente de

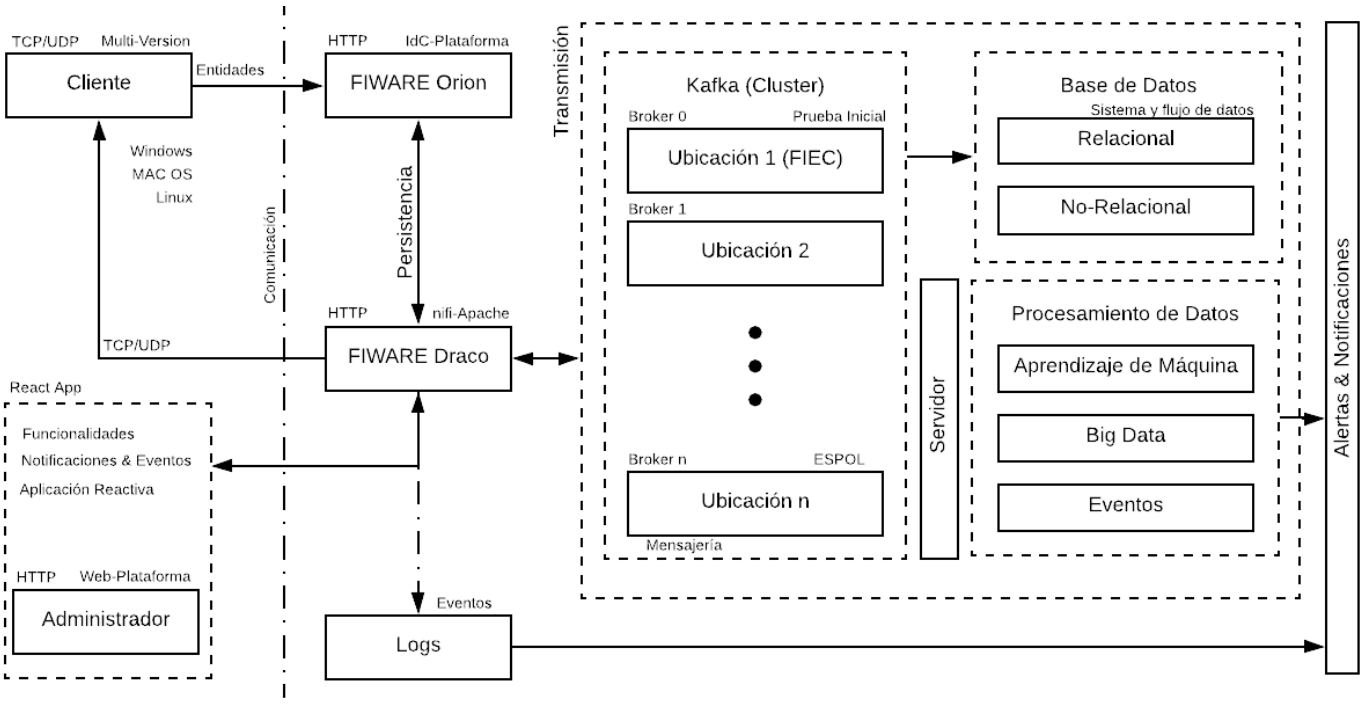

Fig. 5. Esquema de la propuesta del modelo de comunicación para SMALC

procesamiento de registros. La arquitectura completa y las acciones de cada uno de los componentes se describen en detalle a continuación.

A. Comunicación entre la Aplicación del Cliente con Fiware-Orion

Un esquema de comunicación cliente-servidor utilizado actualmente en SMALC; donde un *RP* espera por un paquete enviado a través del *SP*, y, a su vez, utiliza subprocesos de procesamiento, puede consumir un valor significativo de memoria utilizada por la máquina virtual Java (JVM), lo que provoca la pérdida de paquetes en este sistema [23]. Por esta razón, el modelo de comunicación se cambió para usar un esquema de publicación/suscripción usando Actores en Akka-Scala [24]. Akka es un kit de herramientas para crear aplicaciones basadas en mensajes altamente concurrentes, distribuidas y resistentes para Java y Scala, que proporciona a este sistema lo siguiente:

- Actores, permiten construir sistemas que se escalan, utilizando los recursos de un servidor de manera más eficiente.
- Akka, permite sistemas de escritura con tolerancia a fallos (resilientes).
- Procesamiento de flujo asincrónico.
- El servidor y el cliente totalmente asíncronos y con transmisión vía HTTP, proporciona una gran plataforma para crear una estructura basada en micro-servicios.

Este tipo de comunicación puede tener algunas instancias de Actores (Threads) trabajando en forma paralela y distribuida, logrando que el componente Fiware-Orion permita manejar eventos basados en contexto utilizando como enlace la API REST NGSIv2 desarrollado como parte de la plataforma FIWARE. Asimismo, permite administrar todo el ciclo de vida de la información de contexto, incluidas actualizaciones, consultas, registros y suscripciones; utilizando el *Orion Context Broker*, puede crear elementos de contexto y administrarlos mediante actualizaciones y consultas. Además, puede suscribirse a la información de contexto de una computadora, i.e., si los elementos de contexto han cambiado; recibe una notificación sobre esta acción.

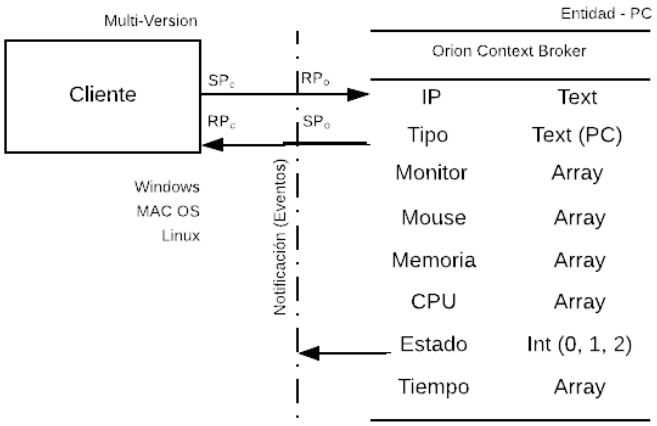

Fig. 6. Comunicación entre cliente-aplicación y Fiware-Orion

En la Fig. 6 se muestra el esquema de comunicación entre la aplicación cliente con Fiware-Orion. La aplicación cliente tiene puertos activos de SP y RP para establecer comunicación con Fiware-Orion usando Akka, e.g., usando el contexto -Estado- donde el significado es (0: Computadora habilitada y libre para ser utilizada por un usuario, 1: Computadora siendo utilizada por un usuario, 2: Computadora en mantenimiento). Si un usuario va a usar una computadora,

el contexto -Estado- cambia de 0 a 1, esta información se envía a Fiware-Orion para enviar notificaciones a todas las aplicaciones o procesos suscritos que necesitan recibir este cambio de contexto.

El modelo de comunicación interna de la aplicación del cliente basada en actores se muestra en la Fig. 7. Cuando los paquetes UDP se reciben utilizando un puerto *RP*, se enrutan al actor principal (MainActor). Luego, los mensajes se envían a los siguientes actores en función del mensaje recibido (Hardware, Software, Usuario, Tiempo, Administración). Finalmente, cuando hay un cambio en los atributos del usuario, se envían a través de una solicitud HTTP a Fiware-Orion.

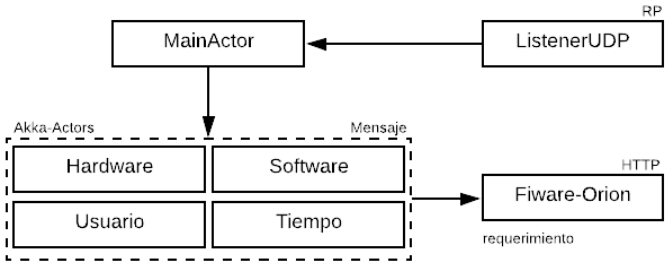

Fig. 7. Modelo de Actores en la aplicación cliente

El modelo de entrega de notificaciones se muestra en la Fig. 8, si el contexto -Estado- cambia, se envían notificaciones a todas las aplicaciones suscritas (Fiware-Draco). Después de realizar las acciones (procesamiento, almacenamiento, aprendizaje automático, etc.) según sea el caso, la aplicación cliente recibe el resultado de esta acción dependiendo de la funcionalidad.

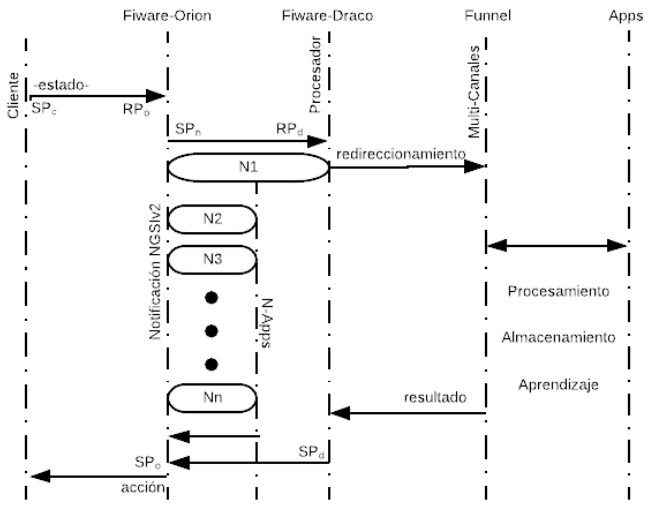

Fig. 8. Envío de notificaciones a las aplicaciones suscritas

El núcleo de Fiware-Draco son los procesadores, estos especifican el tipo de acción a realizar. En la Fig. 9 se muestra el proceso -Listening- que espera notificaciones NGSIv2 de Fiware-Orion, con estos valores un procesador con programación en python -Pre-Processing- desglosa el paquete para redirigir los datos a Kafka, almacenar la información y enviar a un modelo de Apache Spark. El mismo procedimiento se realiza cuando un componente desea enviar información a la computadora, los paquetes viajan a través de Fiware-Draco para ser redirigidos a la aplicación cliente.

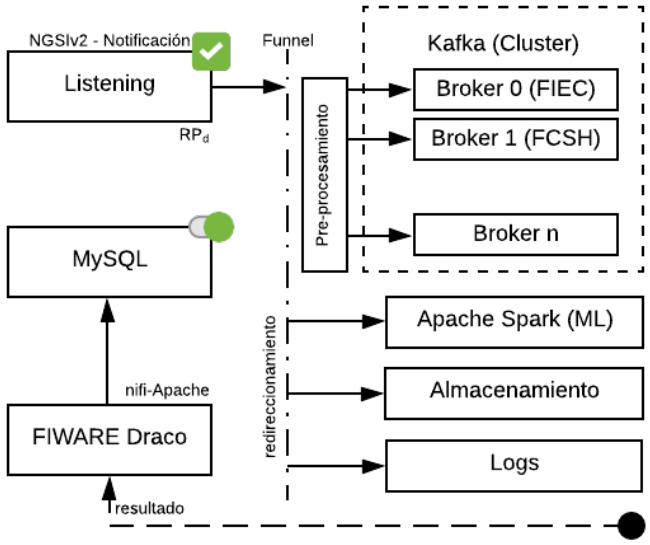

Fig. 9. Modelo de comunicación diseñado en Fiware-Draco

B. Comunicación entre la Aplicación del Cliente con Kafka

Los tópicos en Zookeeper son creados por cada computadora dentro de una Facultad, e.g., en el *Broker-0* (FIEC) los tópicos $PC1, PC2, PC3...PCn$ que corresponden a las computadoras de los laboratorios de la FIEC son creados en Kafka [25]. Este modelo se utiliza para la mensajería entre el cliente y el administrador, así como para el procesamiento de datos y el reconocimiento de patrones.

C. Comunicación Entre la Aplicación de Administración con Fiware-Draco

El módulo de administración es una parte fundamental del sistema, debido a las funciones de control y gestión de las computadoras como de predicción y detección de eventos. Asimismo, la comunicación de este módulo se realiza bajo el esquema de publicación/suscripción, desarrollando una aplicación web reactiva utilizando Node.js y React.js [26]. La aplicación web envía solicitudes a través de HTTP a Fiware-Draco, luego internamente se usa un procesador (ExecuteScript) para redirigir el paquete a un puerto del cliente a través de paquetes TCP/UDP.

D. Procesamiento de Datos & Almacenamiento

Las aplicaciones de IdC para el control y monitoreo en tiempo real generan un gran volumen de información; este esquema es aplicable al sistema de préstamo de computadoras propuesto, debido a que en promedio se atienden a 800 estudiantes por semana solo en la FIEC [27]. Los usuarios generan datos históricos donde el uso de técnicas de aprendizaje automatizado permiten descubrir tendencias y

patrones en el ámbito educativo. Este sistema pretende realizar el control y la administración de las computadoras, y trata de administrar la información de los usuarios convirtiéndola en un sistema de IdC con módulos de Fast-Data [28], [29]. La implementación de este módulo está fuera del alcance de este documento.

E. Alertas & Notificaciones

La incorporación del paradigma IdC en los sistemas de control y administración de computadoras utiliza la tecnología para comprender y explorar esquemas; monitorear y pronosticar eventos inminentes; procesamiento de datos en tiempo real; y procesar y difundir advertencias a las autoridades y administradores relevantes. Se utilizan tecnologías de comunicación que pueden servir para alertas y notificaciones, tales como: Telestax (SMS), Apifonica (llamada), y Vidio (WebRTC).

IV. Prueba de Concepto

El modelo de comunicación de acuerdo con la arquitectura descrita en la sección III utiliza algunas tecnologías de código abierto para incorporar el paradigma IdC en una computadora. Sin embargo, para la incorporación de actores en la aplicación cliente que utiliza la concurrencia en el flujo de comunicación a través de la transferencia de mensajes, se diseñó un modelo que obtiene paquetes UDP y luego los transforma en un mensaje de Hardware, Software, Usuario, Tiempo o Administración dependiendo del *Actor* que recibe la información. La verificación del modelo de comunicación se lleva a cabo mediante la programación de los componentes: cliente y administrador, la configuración de Fiware-Orion y Fiware-Draco, y la creación de un script para la transferencia de mensajes. El escenario de simulación y la interacción de cada uno de los componentes se describen a continuación.

A. Ambiente de Simulación

Se realizó una prueba utilizando diferentes equipos y redes de comunicación para verificar el tiempo de recepción de los paquetes. En la Tabla II se muestran los componentes y recursos que se utilizaron en la simulación, sin embargo, se realizó un escenario complementario utilizando solo dos computadoras dentro de una red WLAN. Computadora 1: Cliente y Fiware-Draco (192.168.100.5), y Computadora 2: Administrador y Fiware-Orion (192.168.100.21).

TABLA II
Componentes en la prueba del modelo de comunicación

Equipo	Dirección IP	Red	Ubicación	Componente
Servidor 1	200.9.176.XXX	LAN	FIEC	Fiware-Orion
Servidor 2	200.9.176.XXX	LAN	FIEC	Fiware-Draco
Computador 1	192.168.100.5	WLAN	Hogar	Cliente
Computador 2	192.168.100.21	WLAN	Hogar	Administrador

B. Prueba de Simulación

La verificación de la prueba de comunicación se realizó a través de una interfaz web utilizando *React.js* para enviar paquetes en formato JSON. Los paquetes de prueba llevan la dirección IP de destino y la información adicional enviada por el administrador. Cuando se envía un paquete desde la aplicación de administración, Fiware-Draco *(Servidor 2)* lo recibe a través del procesador *"HandleHTTP_ControlAdmin_FIEC"*, y entonces es procesado por *"EvaluateJsonPath_GET_IPAddress"* para obtener la dirección IP de destino, tal como se muestra en la Fig. 10. Luego, el procesador *"ExecuteScript_SendDataClient"* reenvía el paquete a través de UDP a la dirección de destino como se muestra en la Fig. 11. Además, en ésta se muestra el formato del paquete JSON recibido y el cambio a un tipo de mensaje para que sea entendido por el actor de Software (SoftwareMessage). Los campos del mensaje representan lo siguiente en el paquete:

- *ip* - Dirección IP de Fiware-Draco
- *message* - Diferentes tipos de mensajes. S: Software, H: Hardware, U: Usuario, T: Tiempo y A: Administración.
- *action* - Acción a realizar por la aplicación cliente, e.g., *'10001'* representa la obtención del software instalado en la computadora.
- *data* - La dirección IP de destino e información adicional en el paquete.

La otra ruta de comunicación se realiza enviando la información del Cliente a Fiware-Orion *(Servidor 1)*. En la Fig. 12 se muestra el envío del paquete con el cambio del atributo "estado". Además, cuando Fiware-Orion recibe un paquete de un cliente y hay un cambio en sus atributos, primero lo redirige a Fiware-Draco a través de diferentes suscripciones. El procesador a cargo de la recepción del paquete de datos es *"ListenHTTP_Orion"*, este recibe toda la información de los clientes para que pueda ser redirigida al procesador correspondiente. Finalmente, el procesador *NGSIToMongo* almacena los datos de acción enviados por Fiware-Orion en una base de datos no relacional *"Mongo"*, y el procesador *LogAttribute* almacena un registro de los atributos.

V. Discusión y Conclusiones

La idea general de crear un modelo de comunicación que se adapte a los nuevos requisitos de la ESPOL y que permita mantener el control y la administración de equipos informáticos en todo el campus, sin la necesidad de contar con un personal en sitio dedicado a la administración del sistema, y mejorar la comunicación entre todas las aplicaciones, ha llevado al estudio y la implementación de un modelo de comunicación con distribución de flujo y procesamiento de transmisión de una gran cantidad de datos (personal administrativo y estudiantes), como se describe en la sección III. El modelo propuesto tiene algunas ventajas debido a que incluye conceptos tales como concurrencia, gestión de paquetes, centralización, comunicación tipo publicación/suscripción, resiliencia y alto rendimiento de las aplicaciones desarrolladas.

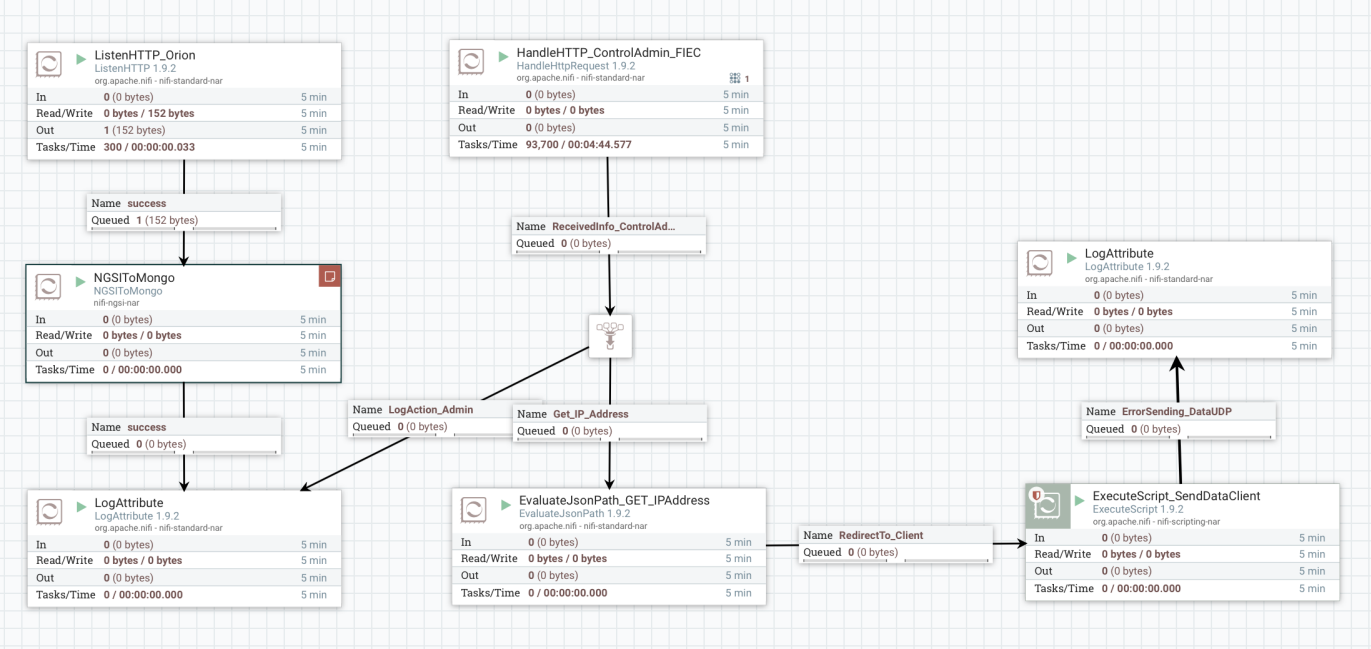

Fig. 10. Configuración del proceso en Fiware-Draco

Fig. 11. Información recibida por la aplicación cliente

Fig. 12. Información enviada por la aplicación cliente

TABLA III
TIEMPOS OBTENIDOS COMO PRUEBAS ALEATORIAS EN SMALC

Funcionalidad	***Computador con Windows 10 versión 9.9*** **192.168.30.166**	**200.9.176.191**
Visualizar pantalla de la PC	4.3s	3.55s
Obtener el Hardware y Software instalado	No funciona	No funciona
Apagar una PC	1.824s	2.496s
Reiniciar una PC	2.214s	2.815s
Cerrar sesión de PC	No funciona	No funciona
Enviar un mensaje	1.637s	1.64s
Visualizar información de: usuario conectado, PC (memoria, procesador, y disco duro)	0.478s	0.5s

En la Tabla III, se presentan diferentes pruebas obtenidas en forma aleatoria para realizar una comparación con los resultados del modelo propuesto. Hubieron ciertos paquetes que no se obtuvieron respuestas, como la prueba de "Visualizar el Hardware y Software de una computadora" y "Enviar a cerrar la sesión de un usuario". Asimismo, se presentan pruebas de funcionalidades básicas de administración de un ordenador, como lo son el enviar a apagar y reiniciar una computadora con un tiempo de ejecución registrado de 2.49 y 2.81 segundos respectivamente.

La prueba de comunicación que conecta la aplicación cliente desarrollada con "Akka", "Scala" y "Java", y el administrador que usa "Node.js" y "React.js", con las tecnologías de la plataforma Fiware *(Draco y Orion)*, han desarrollado una distribución de flujo de paquetes en capa cruzada, e.g., en la ruta de comunicación *(Administrador - Cliente)* los paquetes viajan usando el protocolo HTTP y luego los datos se transportan usando el protocolo UDP. Sin embargo, algo diferente ocurre en la ruta de comunicación *(Cliente - Administrador)*, porque el cliente primero debe notificar a Fiware-Orion a través de HTTP para interactuar con el administrador. Los resultados muestran que en el escenario general hubo un retraso promedio en la comunicación de (1.3 ± 0.2)s, esto se debe al modelo general del escenario, donde se utiliza una Red Privada Virtual para conectarse al campus universitario (ESPOL) y trabajar con los servidores dispuestos para el sistema en la FIEC. Lo contrario ocurre cuando se usa una WLAN para todas las computadoras, donde el paquete de datos promedio se recibe entre $(20 - 30)ms$. Además, para analizar varios canales de comunicación se realizó un entorno simulado alterno con 30 clientes *(IP: "192.168.100.10" hasta "192.168.100.40")* enviando un flujo constante de datos a Fiware-Orion (atributos: estado y tiempo), y consiguiente hacia Fiware-Draco por las supscripciones de los cambios en los atributos. Con esta simulación se pudo comprobar que los paquetes en todas las vías (Cliente -

Fiware-Orion, Fiware-Orion - Fiware-Draco, Administrador - Fiware-Draco) fueron exitosos, y sin pérdida de paquetes

Finalmente, si comparamos los tiempos actuales del sistema SMALC con respecto a la prueba de concepto, se presenta una mejora considerable en los tiempos de respuesta. En base a esto, se puede concluir que el modelo tiene grandes ventajas en una implementación a gran escala, debido a que las tecnologías incorporadas permiten mantener el control y la administración de un sistema de préstamo de computadoras de manera centralizada (Centro de datos de ESPOL) y distribuida (Ubicaciones/Facultades), y con procesamiento de datos en tiempo real para el reconocimiento de eventos y sucesos inesperados.

AGRADECIMIENTO

Los autores desean agradecer a la Escuela Superior Politécnica del Litoral (ESPOL) por financiar este proyecto de investigación con código FIEC-50-2020.

BIBLIOGRAFÍA

[1] J. Stewart, "Cafematics: The cybercafe and the community," in *Community informatics: Enabling communities with information and communications technologies*. IGI Global, 2000, pp. 320–338.
[2] B. Jaafar, "Cyber cafe management system," 2010.
[3] F. R. Aladeniyi and J. K. Fasae, "Use of cybercafé for internet access by the students of rufus giwa polytechnic, owo, nigeria," *Program*, 2013.
[4] I. Cheng, S. Cruz, and L. Unda, "Sistema multiplataforma para la administración de laboratorios de computación (smalc)," 02 2009.
[5] K. Wilson, "About windows," in *Everyday Computing with Windows 8.1*. Springer, 2015, pp. 1–4.
[6] I. Park, "Core system event analysis on windows vista," in *Int. CMG Conference*, 2006, pp. 919–932.
[7] S. Krčo, B. Pokrić, and F. Carrez, "Designing iot architecture(s): A european perspective," in *2014 IEEE World Forum on Internet of Things (WF-IoT)*, 2014, pp. 79–84.
[8] E. M. Lanchimba Lanchimba, "Proyecto de inversión para la creación de un cibercafé en el sector de carcelén de la ciudad de quito." B.S. thesis, Quito: UCE., 2016.
[9] R. G. Camana, "Situación actual de los cybercafés en el ecuador," *Revista Tecnológica-ESPOL*, vol. 29, no. 1, 2016.
[10] A. Noor and M. S. Hossen, "A java based university library management system," *International Journal of Computer Applications*, vol. 975, p. 8887, 2018.
[11] S. Narayan and Y. Shi, "Tcp/udp network performance analysis of windows operating systems with ipv4 and ipv6," in *2010 2nd International Conference on Signal Processing Systems*, vol. 2. IEEE, 2010, pp. V2–219.
[12] J. TANG and T. TEAM, "Exploring control flow guard in windows 10," vol. 10, 2015.
[13] D. Hintea, R. Bird, and M. Green, "An investigation into the forensic implications of the windows 10 operating system: recoverable artefacts and significant changes from windows 8.1." *IJESDF*, vol. 9, no. 4, pp. 326–345, 2017.
[14] P. A. Japon Albuja, "Propuesta de un sistema de control interno informático para los laboratorios de computación de la uaic de la utmach." 2019.
[15] M. Šimunek, P. Bisták, and M. Huba, "Virtual laboratory for control of real systems," in *Conference Procedings ICETA*, 2005.
[16] Z. Li, Z. Yin, and F. Zhang, "Research on university open laboratory management system based on" internet+," 2019.
[17] D. Adrianto, M. Martani, D. Indriani, and R. Susanti, "Development of online learning system for software laboratory center in bina nusantara university," *ComTech: Computer, Mathematics and Engineering Applications*, vol. 8, no. 2, pp. 83–94, 2017.
[18] C. D. BAQUE PINCAY, "Diseño de un sistema de control de acceso biométrico con lector rfid para la sala de cómputo# 14 de la carrera ingeniería en computación y redes." B.S. thesis, JIPIJAPA-UNESUM, 2018.
[19] E. D. Tipantuña Lizano, "Diseño e implementación de un sistema de control de acceso de laboratorios, mediante una aplicación móvil." B.S. thesis, Quito, 2017.
[20] Y. M. Vasquez Chamay and D. M. Primo Paico, "Diseño de un sistema inmótico utilizando la plataforma arduino para el laboratorio de computación e informática de la facultad de ciencias físicas y matemáticas de la unprg, lambayeque-2017." 2019.
[21] J. D. ALVARADO SOLEDISPA, "Prototipo de sistema de comunicación conacceso a internet parael laboratorio de telecomunicaciones de la carrera ingeniería en computación y redesa través de un dron." B.S. thesis, JIPIJAPA-UNESUM, 2019.
[22] P. VELÁSQUEZ and M. ANGÉLICA, "Implementación de un sistema inalámbrico para la comunicación de datos en el laboratorio de hardware en la carrera de ingeniería en computación y redes," B.S. thesis, JIPIJAPA-UNESUM, 2019.
[23] O. Gheorghioiu, "Statistically determining memory consumption of real-time java threads," Ph.D. dissertation, Massachusetts Institute of Technology, 2002.
[24] J. Hunt, "Further akka actors," in *A beginner's guide to scala, object orientation and functional programming*. Springer, 2018, pp. 345–360.
[25] W. Velasquez Vargas, A. Muñoz Arcentales, and J. Salvachúa, "A distributed system model for managing data ingestion in a wireless sensor network," in *Computing and Communication Workshop and Conference (CCWC), 2017 IEEE 7th Annual*. IEEE, 2017, pp. 1–5.
[26] J. Hunt, "Play framework," in *A Beginner's Guide to Scala, Object Orientation and Functional Programming*. Springer, 2018, pp. 431–446.
[27] I. A. T. Hashem, V. Chang, N. B. Anuar, K. Adewole, I. Yaqoob, A. Gani, E. Ahmed, and H. Chiroma, "The role of big data in smart city," *International Journal of Information Management*, vol. 36, no. 5, pp. 748–758, 2016.
[28] W. Velásquez, A. Munoz-Arcentales, and J. Salvachúa, "Fast-data architecture proposal to alert people in emergency," in *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE, 2018, pp. 165–168.
[29] A. Munoz-Arcentales, W. Velásquez, and J. Salvachüa, "Practical approach of fast-data architecture applied to alert generation in emergency evacuation systems," in *2018 International Symposium on Networks, Computers and Communications (ISNCC)*. IEEE, 2018, pp. 1–6.

Washington Velasquez (Guayaquil - Ecuador, 1988) He's got a Ph.D. in Telematics System Engineering in July 2019, and MS.C in Telematic Services and Network Engineering in July 2014 both by Universidad Politécnica de Madrid (UPM, Spain). He was graduated as Telematics Engineer in 2013 (ESPOL, Ecuador). He has worked as Professor of Telematic's Department since 2019 at Escuela Superior Politécnica del Litoral. He has been researching in fields like Telemetry and Remote Control, Smart Cities and Big data with a major research interest in Telemedicine, E-health service in an emergency, and Embedded Systems.

Margarita Filian-Gomez (Guayaquil - Ecuador, 1984) She's got a master's degree in Computer Security in December 2015. She was graduated as Computer Engineer in 2009 (ESPOL, Ecuador). She works at ESPOL's Faculty of Electrical and Computer Engineering (FIEC) like Chief Information Technology and she is responsible for the oversight and management of all technological aspects of that Faculty. She likes to learn about security issues and new technologies to improve what already exists as a service.