SAVIA: Smart City Citizen Security Application Based on Fog Computing Architecture

M. Castillo-Cara, G. Mondragón-Ruíz, E. Huaranga-Junco, E. Arias, and L. Orozco-Barbosa

***Abstract*—To demonstrate and promote the use of fog computing and complex event processing relative to the smart city context, this paper proposes a suite of applications developed to address gender-based violence, which is a significant problem in many jurisdictions. To better protect victims, we propose the open-source Surveillance and Alarm for gender VIolence Application (SAVIA) system, whose primary goal is to ensure the safety of women under the protection of restraining orders. In the proposed SAVIA system, surveillance is carried out by mobile/server applications that continuously evaluate the location of both victims and aggressors, and, if a restraining order is violated, the system generates multiple alerts that are sent to both the victim and police. The proposed SAVIA system effectively represents a citizen-centric software and hardware implementation developed using fog computing and complex event processing paradigms and the benefits of computational resources.**

***Index Terms*—Smart City, Distributed systems, Fog Computing, Complex Event Processing, Mobile applications, Server application, Green energy, Energy consumption.**

I. Introducción

En la actualidad, una de cada tres mujeres se someten a la violencia de género, lo que indica que la violencia contra las mujeres puede ser considerada una pandemia. La Organización Mundial de la Salud (OMS) define una pandemia como "la propagación mundial de una nueva enfermedad" y afirma que "la mayoría de las personas no tienen inmunidad". Obviamente, la violencia contra las mujeres no es un fenómeno nuevo; sin embargo, esta forma de abuso desafortunadamente se está extendiendo por todo el mundo.

Este documento, para demostrar la prevalencia de la violencia de género, se centra en España y Perú. En España, en [1], tuvieron lugar 60 feminicidios en 2015 y 44 en 2016. Para mayo de 2017, el número de feminicidios era 27. Otra estadística importante es el número de llamadas a 016, el número de teléfono para ayudar a las mujeres. El número de feminicidios en Perú fue de 124 en 2016. El número de víctimas de violencia de género que recibieron asistencia del Centro de Emergencia para Mujeres fue de 49,138 en 2014, 50,485 en 2015 y 70,510 en 2016 [2], que indica que las campañas de sensibilización en Perú también son efectivas.

En este contexto, el siguiente trabajo propone la Aplicación de Vigilancia y Alerta para la Violencia de Género (Surveillance and Alert for gender VIolence Application, SAVIA). SAVIA comprende un ecosistema de aplicaciones basadas en una arquitectura fog computing [3, 4] que integra información de víctimas, agresores y policías para detectar situaciones potencialmente peligrosas. El ecosistema SAVIA comprende aplicaciones móviles para víctimas (SAVIApp) y agentes de policía (SAVIAPol), un grillete electrónico para agresores (SAVIAtt) y una aplicación de servidor (SAVIA) administrada por una institución policial. Los componentes se desarrollaron teniendo en cuenta la eficiencia, la eficacia, la confiabilidad, el costo y la facilidad de uso. En otras palabras, al desarrollar SAVIA, consideramos el uso apropiado de los recursos (principalmente el consumo de energía o la duración de la batería), la funcionalidad óptima, la flexibilidad en relación con diferentes situaciones, los beneficios para una amplia gama de usuarios y la facilidad de uso.

La Figura 1 muestra el esquema general de la aplicación. Hay que tener en cuenta que empleamos el procesamiento de eventos complejos (CEP) [5, 6] en el core level de una arquitectura fog computing para optimizar dos funciones críticas en los sistemas de seguridad ciudadana, es decir, restringir órdenes y alertas.

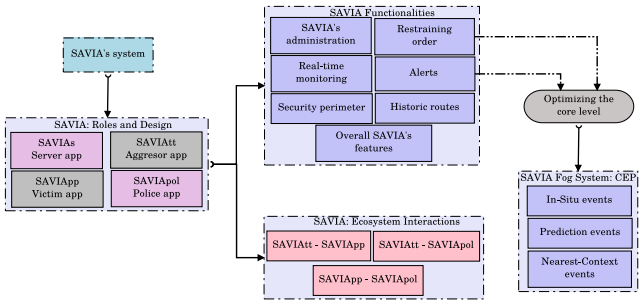

Fig. 1. Esquema propuesto de SAVIA.

El artículo está organizado de la siguiente manera. La sección I pone en contexto el problema a tratar con la solución tecnológica propuesta en este trabajo, SAVIA. El trabajo relacionado se discute en la Sección II, exponiendo las contribuciones principales. El diseño, roles e interacciones en el ecosistema SAVIA se describen en la Sección III. La Sección IV describe las funcionalidades del sistema SAVIA y la interacción de estas con los cuatro componentes desarrollados. Además, la optimización del core level utilizando CEP es explicada en la Sección V, demostrando un menor consumo computacional de una arquitectura fog computing respecto a cloud computing. Finalmente, las conclusiones y trabajos futuros son expuestas en la Sección VI.

Manuel Castillo-Cara, Giovanny Mondragón-Ruíz and Edgar Huaranga-Junco, Center of Information and Communciation Technologies, Unviersidad Nacional de Ingeniería, Lima, Peru.

Enrique Arias Antúnez and Luis Orozco-Barbosa are with Computing Systems Department, Universidad de Castilla-La Mancha, Albacete, Spain.

II. Trabajos Relacionados

Actualmente, las tecnologías de Smart Cities representan un desarrollo importante, particularmente para aplicaciones centradas en el ciudadano. En este estudio, desarrollamos aplicaciones móviles/servidor para minimizar la violencia de género; sin embargo, el sistema SAVIA propuesto podría extenderse a otras áreas relacionadas con la seguridad ciudadana. El sistema SAVIA ha sido desarrollado bajo una arquitectura fog computing, y empleamos CEP para optimizar algunos procesos críticos en el core level de esta arquitectura, así como el consumo de energía. Por tanto, en esta sección, se discute los sistemas de violencia de género y las arquitecturas cloud/fog para, posteriormente, mostrar las principales contribuciones.

A. Sistemas Contra la Violencia de Género

Los desarrolladores han tenido el desafío de crear aplicaciones basadas en smartphones relacionadas con la violencia de género, y se han desarrollado diversas herramientas de software/hardware para ayudar a las víctimas. La mayoría de las herramientas brindan información pero no incluyen funciones de monitoreo, por ejemplo, Aurora, que es una aplicación de violencia doméstica y familiar [7].

En este sentido, el gobierno español está emprendiendo varias iniciativas importantes para prevenir y, con suerte, erradicar la violencia de género. Por ejemplo, el gobierno ha implementado la línea de ayuda 016. Además, la aplicación LIBREs [8] está disponible que tiene una funcionalidad similar a SAVIA. Sin embargo, SAVIA tiene características adicionales, como es una solución de código abierto y basada en fog computing, utilizando CEP para analizar las diferentes reglas que se implementan para las alertas y, los smartphones, puede ser utilizados para el posicionamiento inteligente de las fuerzas policiales.

Además, SAVIA es un proyecto open software en dos sentidos. En primer lugar, todo el software utilizado en su implementación es libre. En segundo lugar, el código fuente de todas las aplicaciones SAVIA está disponible de forma gratuita, ya que es un software de código abierto. Por lo tanto, el sistema puede mejorarse y adaptarse libremente en relación con diferentes regulaciones nacionales y aplicaciones relacionadas para facilitar la interoperabilidad entre diferentes sistemas de violencia de género. Creemos que una solución de mejor esfuerzo puede realizarse si diferentes organizaciones, instituciones e investigadores usan las mismas tecnologías base.

B. Arquitecturas Fog/Cloud

Fog computing es una tecnología emergente dentro de Internet of Things (IoT) [9] donde cada plataforma toma decisiones independientes del cloud. En este contexto, es importante desarrollar e integrar, varios dispositivos, protocolos, software y hardware para mejorar el rendimiento y reducir el consumo de energía [10]. Un estudio relacionado [11], analizó el rendimiento y el consumo de energía de diferentes técnicas de encriptación en el nivel central. Por ejemplo, fog computing intenta optimizar el rendimiento, los recursos y la seguridad tanto en el core level como en edge level [12, 13]. En este sentido, los dispositivos móviles desempeñan un papel importante en la computación perimetral porque existen múltiples conexiones entre el edge y core level [4].

Se han desarrollado aplicaciones IoT relacionadas como [14], la cual consideró diferentes escenarios en los que la arquitectura de fog computing podría usarse para mejorar los diversos servicios proporcionados por una Smart City, por ejemplo, transporte, gestión de desechos y agua, y medio ambiente. Otro estudio [15] presentó la plataforma SmartCityWare, que emplea diferentes aplicaciones y compara su trabajo para arquitecturas fog/cloud en un contexto Smart City. Hay que tener en cuenta que diferentes estudios han investigado los diferentes servicios; sin embargo, no se ha considerado la optimización de diferentes aplicaciones en los niveles core/edge, así como el consumo computacional que lleva consigo.

Fog computing también se aplica al análisis de datos con un enfoque en la toma de decisiones y predicciones en tiempo real. Por ejemplo, un estudio anterior [16] investigó el uso de técnicas de big data para tomar decisiones en tiempo real en diferentes aplicaciones para localización en interiores con Smartphones. Además, CEP se ha empleado para analizar eventos generados tanto en edge como core level [6] para facilitar la toma de decisiones antes de almacenar datos en una base de datos, lo que elimina reiteración de las consultas y servicios web. En relación con las tecnologías de predicción en el contexto de las Smart Cities, muchos estudios han propuesto machine learning o las técnicas de posprocesamiento de big data en el core level [17].

En comparación con los sistemas anteriormente descritos, SAVIA representa una solución Open Source de bajo costo, dinámica, fácil de usar, eficiente, robusta y confiable que podría ayudar a prevenir la violencia de género. Además, consideramos optimizar el core level utilizando CEP para minimizar los problemas de concurrencia de datos y reducir la dependencia de los servicios web para proporcionar diferentes tipos alertas a través de estos eventos generados. Por tanto, demostramos que, con estas tecnologías características de una arquitectura fog computing, se obtiene un menor consumo de energía que en cloud computing [11, 18].

C. Contribuciones

Teniendo en cuenta las investigaciones anteriores, las principales contribuciones de este artículo se pueden resumir en:

- El uso de una arquitectura fog computing, que ninguna otra solución tiene implementada, permitiendo un importante ahorro de energía.
- La basculación entre GPS y triangulación en función de la distancia que separan a los diferentes agentes que participan en SAVIA permitiendo, finalmente, un segundo ahorro de energía.
- El uso de CEP, y principalmente la velocidad de "filtrado", ha permitido poder considerar varios tipos de contextos de orden de alejamiento generando diferentes eventos complejos. Además, esta herramienta permite hasta un millón de eventos complejos por segundo.

- El software utilizado es *Open Source*, lo que permite una implementación más económica y, además, una mejora continua que puede ser llevada a cabo por la comunidad.
- Esta solución, debido a la información sensible que recoge, cumple con los requisitos de la ISO/IEC 27018:2014 relativa a la protección de la privacidad en este tipo de entornos [19]. Esto supone tener un cloud privado de la propia policía o de la entidad judicial al cargo de las órdenes de alejamiento; y son estas autoridades las que velan porque no se produzcan filtraciones de los datos (ver artículo 13.8 de Ley de Protección de datos personales, n. 27933 de la República del Perú). Así mismo, la comunicación entre dispositivos y cloud se realiza de manera encriptada.

III. Ecosistema SAVIA: Roles, Diseño e Interacciones

El sistema SAVIA comprende un conjunto de aplicaciones que abordan el problema de la seguridad ciudadana en relación con la violencia de género. En este sentido, SAVIA consta de dos aplicaciones para smartphones (una para víctimas y otra para la policía, SAVIApp y SAVIApol, respectivamente) y una aplicación de grillete electrónico para agresores (SAVIAtt); las cuales envían la información de posición a la aplicación del servidor (SAVIAs).

A. Roles

El diseño de los componentes del sistema SAVIA, desde una perspectiva de desarrollo de software, se muestra en la Figura 2, los cuales puede verse los cuatro roles fundamentales de SAVIA.

*1) **SAVIAs**:* SAVIAs fue desarrollado utilizando lenguajes de programación y tecnologías que soportan muchas tareas simultáneamente para aprovechar la concurrencia entre diferentes aplicaciones de dispositivos móviles. Por lo tanto, SAVIAs utiliza MongoDB, un sistema de base de datos concurrente, para las consultas con gran cantidad de recursos que se ejecutan constantemente [20]. Además, SAVIAs utiliza la base de datos relacional MySQL para establecer una arquitectura de datos jerárquica entre los roles (es decir, víctima, agresor y policía) y sus datos, que generalmente son datos no críticos. Los componentes de SAVIAs se muestran en la Figura 2 (ver componente "Servidor web: SAVIAs").

Relativo a la arquitectura fog computing, el componente principal en el sistema SAVIAs es el "análisis de eventos a través de CEP". Cuando se infringe una orden de restricción, CEP encuentra al SAVIApol más cercano a SAVIApp y envía alertas a ambos a través de Message Queue Telemetry Transport (MQTT), este proceso es explicado en detalle en la Sección V.

*2) **SAVIAtt**:* El rol SAVIAtt representa al agresor definido en una orden de restricción. La Figura 2 (ver componente "Aplicación: SAVIAtt") muestra los componentes de este rol. Para rastrear a un agresor, debe usar un grillete electrónico u otro dispositivo electrónico con un sensor GPS y GSM (Global System for Mobile communications). Tenga en cuenta que SAVIAtt no recibe ninguna información relevante de SAVIAs.

*3) **SAVIApp**:* En nuestro contexto, el rol de SAVIApp representa a la víctima. Los componentes de desarrollo se muestran en la Figura 2 (ver componente "Aplicación de Android: SAVIApp"). El smartphone de la víctima almacena la posición y la información del usuario en una base de datos interna. SAVIApp está asociado con uno o más agresores, es decir, SAVIAtt, y verifica internamente su propia ubicación y la del agresor. El sistema enviará una alerta a ambos dispositivos si se ha infringido el perímetro de seguridad definido por una orden de restricción o pulsa el botón de pánico.

*4) **SAVIApol**:* SAVIApol es responsable de garantizar la seguridad de la víctima y puede ver las ubicaciones de la víctima y el agresor en tiempo real cuando SAVIApp envía una alerta a SAVIAs. Con esta alerta, SAVIApol puede verificar la ubicación de SAVIAtt y SAVIApp y asistir a la alerta, ver Figura 2 (ver componente "Android Application: SAVIApol").

La única diferencia entre SAVIApol, SAVIApp y SAVIAtt es que SAVIApol administra las alertas y, una vez que recibe una alerta de SAVIAs, puede monitorear las ubicaciones de SAVIApp y SAVIAtt en tiempo real y en caso de emergencia.

B. Diseño

La característica principal de SAVIA es el monitoreo en tiempo real de la información de posición de la víctima y el agresor para mantener las órdenes de restricción e identificar situaciones comprometidas. Se han implementado dos enfoques para calcular las posiciones de víctimas y agresores, es decir, información de ubicación GPS y triangulación de las señales de la antena Wi-Fi. Hay que tener en cuenta que el enfoque de triangulación es menos preciso que el de GPS; sin embargo, consume menos energía [21]. Por lo tanto, el seguimiento de ubicación basado en GPS no está habilitado cuando la distancia entre una víctima y un agresor es significativa. Sin embargo, si la distancia entre una víctima y un agresor es la definida por una orden de restricción, el sistema cambia automáticamente al modo de triangulación por GPS y SAVIAs envían un mensaje automático a los dispositivos de la víctima y del agresor.

SAVIAs almacena, en un servidor perteneciente a la autoridad competente, la posición de cada víctima y agresor porque esta información se puede usar en un juicio o para estudiar el comportamiento y la ubicación del usuario. Además, los datos históricos de posición pueden usarse para determinar si un agresor se mantiene relativamente cerca de una víctima pero no viola la orden de restricción. Tales perfiles de comportamiento pueden usarse para justificar acciones preventivas adicionales.

SAVIAs envía mensajes a SAVIApol y SAVIApp cuando detecta situaciones anómalas. Además, SAVIApp tiene un botón de pánico para emergencias. El formato del mensaje es un objeto JSON (JavaScript Object Notation) para habilitar la comunicación entre las aplicaciones SAVIApp, SAVIAtt y SAVIApol con SAVIAs. Este objeto JSON se envía al servidor mediante HTTPS (Hypertext Transfer Protocol Secure) para garantizar la seguridad y privacidad de los mensaje. A esto, se le suma la seguridad implementada en el cloud privado perteneciente a la autoridad competente que sigue los requisitos del estándar ISO/IEC 27018:2014 [19].

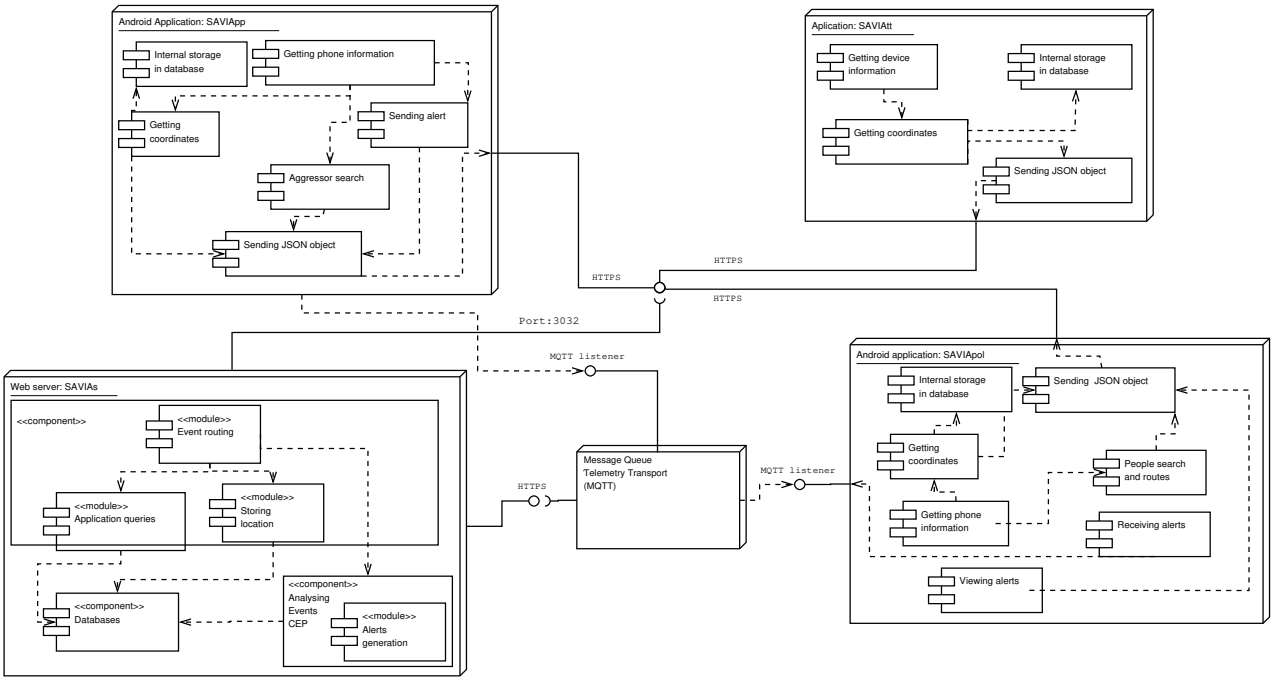

Fig. 2. Diseño del sistema SAVIA.

C. Interacción entre las Aplicaciones

En el sistema SAVIA, todas las aplicaciones móviles, es decir, SAVIApp, SAVIAtt y SAVIApol, se comunican constantemente con SAVIAs. Hay que tener en cuenta que las aplicaciones/dispositivos solo procesan su propia información, es decir, no se comunican directamente entre sí. Dado que SAVIAs intercambia toda la información regularizada entre las aplicaciones bajo algunas reglas específicas, existe un flujo de información indirecto entre estas tres aplicaciones, como lo representan las líneas de puntos en la Figura 3.

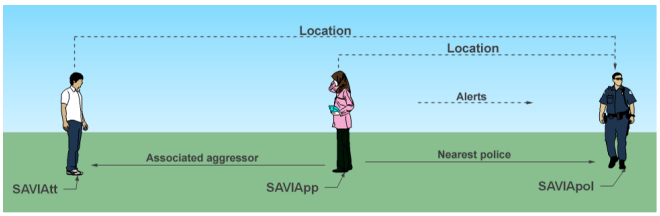

Fig. 3. Proceso del flujo de información entre las aplicaciones.

1) ***SAVIAtt – SAVIApp****:* Para obtener el control total de SAVIAtt, que tiene reglas dictadas por un tribunal, debe tener un enlace indirecto con SAVIApp, ver Figura 3. Esta asociación, que no debe implicar el intercambio directo de información bajo ninguna circunstancia, se crea a través de SAVIAs, es decir, los dos roles están definidos por SAVIAs. Luego, los usuarios se vinculan a través de los identificadores de las aplicaciones, los dispositivos y la información de estos. Por lo tanto, una vez que ambos roles están asociados, SAVIAs pueden determinar si se está violando una orden de restricción.

2) ***SAVIAtt – SAVIApol****:* SAVIApol obtiene las coordenadas de SAVIAtt (ver Figura 3) en caso de que se active una alerta de emergencia según los siguientes casos: (i) Cuando SAVIAtt viola una orden de restricción; o (ii) SAVIAtt manipula el grillete electrónico y/o no hay comunicación con SAVIAs. En estos casos, SAVIApol puede ver a SAVIAtt en un mapa del smartphone porque la información de ubicación de SAVIAtt es enviada por SAVIAs. Hay que tener en cuenta que SAVIApol no puede mostrar la ubicación de SAVIAtt si no se producen los eventos anteriores.

3) ***SAVIApp – SAVIApol****:* Las relaciones de estos dos roles son las más críticas. SAVIApp puede enviar una alerta manual presionando el botón de pánico o una alerta automática cuando SAVIAtt viola una orden de restricción. En ambos casos, SAVIApp envía su ubicación a SAVIAs, que busca el SAVIApol más cercano. El envío y la recepción de la información de ubicación a SAVIAs se realiza constantemente hasta que un administrador de SAVIAs cierra la alerta generada, ver Figura 3.

IV. Funcionalidades de SAVIA

SAVIA tiene diferentes funcionalidades para mejorar las relaciones entre SAVIAtt, SAVIApp y SAVIApol. Estas funcionalidades son reglas básicas desarrolladas principalmente por SAVIAs, aunque las funciones tienen características importantes para verificar el funcionamiento correcto de todo el sistema. La Tabla I muestra todas las características del proyecto y sus roles correspondientes. De estas características a continuación se detallarán las más relevantes.

A. Orden de Alejamiento

Es la primera regla a verificar constantemente es la distancia entre una víctima y un agresor(es). Normalmente, una orden de restricción se aplica por orden judicial y establece la distancia mínima entre estas partes. SAVIA distingue dos tipos de órdenes de restricción:

TABLA I
CARACTERÍSTICAS DE SAVIA Y ROLES CORRESPONDIENTES

CARACTERÍSTICA	SAVIAtt	SAVIApp	SAVIAPol	SAVIAs
Registrar agresor	X	X	X	✓
Registrar policía	X	X	X	✓
Registrar víctima	X	X	X	✓
Visualización de alertas	X	X	✓	✓
Visualización de alertas históricas	X	X	X	✓
Gestión de alertas	X	X	X	✓
Enviar posición	✓	✓	✓	X
Enviar alerta	X	✓	X	X
Seguimiento Víctima/Agresor	X	✓	✓	✓
Encontrar policía	X	✓	X	X
Rutas históricas del agresor	X	✓	✓	✓
Rutas históricas de la víctima	X	✓	✓	✓
Rutas históricas del policía	X	X	✓	✓
Visualización usuario-lugar	X	✓	✓	✓

- Orden de restricción estática: este tipo de orden se otorga cuando el agresor tiene acceso limitado a diferentes áreas normalmente visitadas por la víctima.
- Orden de restricción dinámica: en este caso, SAVIAs monitorea la ubicación de estos roles en tiempo real manteniendo una distancia entre ellos. Cuando se viola la orden judicial, se envía una alerta a SAVIApp a través de SAVIAs.

Hay que tener en cuenta que SAVIA analiza las órdenes de restricción utilizando una metodología CEP (Sección V).

B. Alertas

SAVIA emite dos tipos de alertas:

- Alertas automáticas: se generan cuando (i) Un agresor viola una orden de restricción; (ii) El agresor manipula el grillete electrónico; o (iii) SAVIAtt no envían la ubicación a SAVIAs durante un período determinado.
- Alertas manuales: la víctima genera alertas manuales cuando se presiona un botón de pánico en el smartphone.

SAVIApp y SAVIAtt envían información de ubicación a SAVIAs, el servidor almacena la información y, en paralelo, CEP determina si se ha violado la orden de restricción. En este caso, SAVIAs envía la alerta a través de MQTT (con las últimas posiciones conocidas de SAVIAtt y SAVIApp) al SAVIApol más cercano a la víctima.

C. Perímetro de Seguridad

Una de las tareas más importantes de SAVIA surge cuando SAVIAtt viola una orden de restricción dinámica. En relación con la optimización del consumo de energía, esta precisión da lugar a dos conceptos para cada orden de restricción, es decir, el perímetro de seguridad y el perímetro de prevención, que se ilustran en la Figura 4.

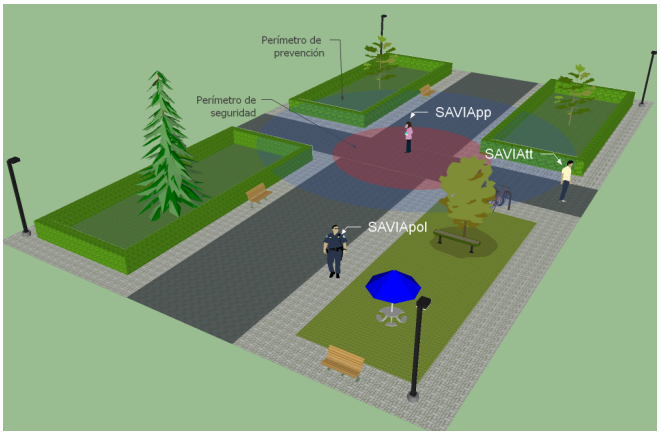

Fig. 4. Perímetro de prevención y seguridad.

Para explicar este proceso, asumimos que hay una víctima (SAVIApp), un agresor (SAVIAtt) y un oficial de policía (SAVIApol) registrado en el servidor (SAVIAs). Por ejemplo, SAVIAtt tiene una orden de restricción con un perímetro de seguridad de 1km en relación con SAVIApp, y ambos envían constantemente su información de ubicación a SAVIA.

En el paso inicial, se establece automáticamente un perímetro de prevención de 2km alrededor de SAVIApp. El sistema recibe la información de ubicación de SAVIApp y busca su SAVIAtt asociado, cuya ubicación también se envía al sistema. Simultáneamente, la posición de SAVIApp también se envía al sistema. Una vez que se ha encontrado el SAVIAtt, la distancia a SAVIAtt se calcula utilizando el módulo "Geolib". Si esta distancia es mayor que el perímetro de prevención, SAVIAs envía una respuesta que indica que la operación se realizó con éxito usando una señal de "OK", y también indica que el modo de triangulación, que usa los sensores de Wi-Fi en dispositivos móviles, debe ser utilizado como el servicio de localización. Su precisión es 30 metros menos que la de un sensor GPS, y su consumo de batería es un 40% menos [21].

Cuando la distancia es inferior a 2km pero superior a 1km, el smartphone cambia el servicio de ubicación al modo GPS, que tiene una mejor precisión de ubicación, y las coordenadas se envían a SAVIAs con mayor frecuencia que en el modo de triangulación. Finalmente, en este caso, cuando SAVIAtt rompe el perímetro de seguridad, SAVIAs envía la información de ubicación de SAVIAtt y SAVIApp al SAVIApol más cercano a SAVIApp. Esta información de ubicación se muestra en Google Maps en el dispositivo SAVIApol.

V. OPTIMIZACIÓN DEL CORE LEVEL

Un aspecto crítico en el sistema SAVIA es la capacidad de controlar la concurrencia generada por múltiples solicitudes en la base de datos generada para los diferentes roles, es decir, SAVIAtt, SAVIApp y SAVIApol. Además, como hemos discutido anteriormente, la orden de restricción y las alertas automáticas deben redefinirse debido a que muchas consultas se envían a la base de datos, donde el sistema verifica la distancia correcta entre estos dos roles. En este contexto, el primer objetivo es desarrollar un marco para analizar los datos

almacenados en paralelo, eliminando así la concurrencia y las consultas iterativas.

En el sistema SAVIA, esta tarea se refiere al procesamiento de la fusión de los datos recopilados por las aplicaciones. El principal resultado de esta tarea es notificar a las partes afectadas las alertas derivadas de eventos de nivel inferior [6, 9]. Para ello, se ha utilizado Apache Flink como motor CEP para eventos generales en el core level. Finalmente, esta sección compara el uso de utilizar CEP respecto a una arquitectura típica POST para la generación de alertas, realizando un benchmark centrado en el consumo de energía.

A. Procesamiento de Evento Complejos

Para cada uno de los casos utilizados, Flink-CEP usa cada nuevo valor de entrada a través de un socket como un evento x en el tiempo T_x, minimizando así el conjunto de eventos. El proceso mediante el cual Flink-CEP detecta, analiza y genera alarmas es el siguiente: (i) Reconocimiento de patrones, el cual Flink-CEP considera el orden de los eventos y sus condiciones individuales para generar un patrón; (ii) Análisis de eventos donde Flink-CEP analiza las relaciones entre eventos evaluando el conjunto de eventos recibidos; y (iii) Acciones, el cual Flink-CEP toma una decisión de acuerdo con el análisis anterior.

*1) **Caso 1: Eventos In-situ**:* Esta situación ocurre cuando la posición de SAVIAtt viola una orden de restricción relativa a la posición de SAVIApp. La Figura 5 ilustra este caso, donde cada evento recibido, de cada rol, se evalúa de acuerdo con la orden de restricción en la línea de tiempo T_x. Por lo tanto, tenemos el siguiente proceso:

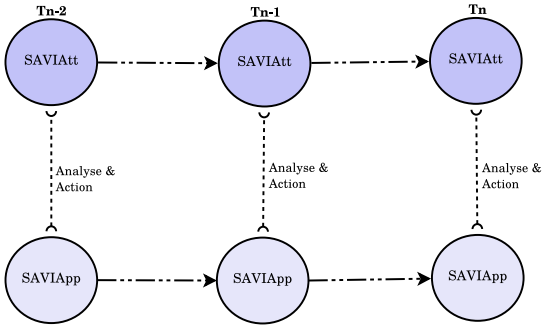

Fig. 5. Esquema general de evetos In-situ.

1) Análisis: Flink-CEP determina la distancia entre los últimos eventos enviados por SAVIAtt y SAVIApp. Si esta distancia es mayor que la establecida por la orden de restricción, se descartan los eventos; de lo contrario, pasa al proceso de "Acción". Además, si algunos de los roles no envían su información de ubicación durante esta línea de tiempo, pasa al proceso de "Acción".
2) Acción: Si "Acción" está activada, la posición más cercana del rol de SAVIApol, respecto a SAVIApp, se extrae a la base de datos y SAVIAs envía una alerta a SAVIApol y SAVIApp siguiendo el proceso descrito en la Sección III-C.

*2) **Case 2: Eventos predictivos**:* Este tipo de eventos se analizan para predecir las posiciones futuras de SAVIAtt y SAVIApp. Por lo tanto, este caso intenta verificar si un evento futuro a tiempo $T'n+1$ violará la orden de restricción, ver Figura 6, obteniendo el siguiente proceso:

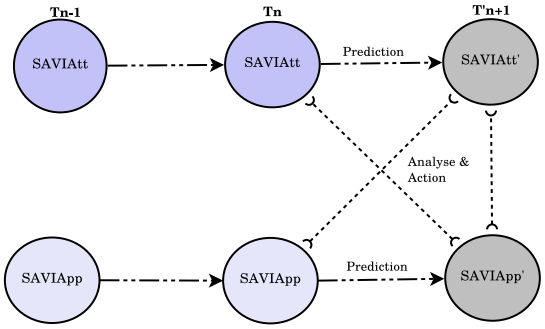

Fig. 6. Esquema general de eventos predictivos.

1) Predicción: Primero, las ubicaciones anteriores de los roles se evalúan para predecir la posición futura, es decir, $Pos(T'_n + 1)$. Esta predicción se establece para calcular $Pos(T'_n + 1)$ utilizando un vector de distancia, $\vec{V}_d$, que se calcula utilizando las dos últimas posiciones $Pos(T_n)$ y $Pos(T'_n + 1)$.

$$\vec{V}_d = Pos(T_n) - Pos(T_{n-1})$$
$$Pos(T'_n + 1) = \vec{V}_d + Pos(T_n)$$

2) Análisis: Una vez que las posiciones de ambos roles $SAVIAtt'$ y $SAVIApp'$ han sido predichas, la orden de restricción se evalúa en relación con los eventos obtenidos en los tiempos T_n y T'_{n+1}. Si la orden de restricción es violada o será violada, el siguiente paso es el proceso "Acción"; de lo contrario, los eventos se descartan.
3) Acción: Si "Acción" está activada, SAVIAs envían una alerta a SAVIApp y al SAVIApol más cercano.

*3) **Caso 3: Eventos de contexto más cercano**:* Este caso intenta evitar situaciones de proximidad entre SAVIAtt y SAVIApp al determinar si ambos roles han estado en una ubicación cercana en diferentes momentos, por ejemplo, si en el momento T_{n-2}, SAVIAtt ha violado una orden de restricción en T_n. Este caso, intenta determinar si ambos roles han estado cerca y/o si uno de ellos está siguiendo la ruta del otro, ver Figura 7. La secuencia para este caso es:

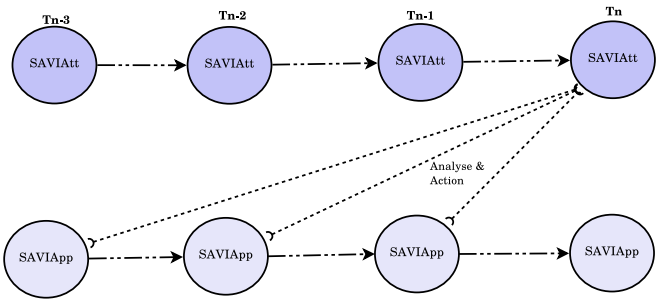

Fig. 7. Esquema general de eventos de contexto más cercano.

1) Análisis: Este caso analiza el último evento SAVIAtt en el tiempo T_n en comparación con los eventos SAVIApp anteriores para verificar si se ha infringido la orden de restricción. Aquí, pasa al proceso de "Acción"; de lo contrario, los eventos se descartan.
2) Acción: La acción es generar una alerta en SAVIAs. Aquí, el administrador de SAVIAs es responsable de supervisar ambas funciones y de determinar si se envía una alerta o si se descartan ambos eventos.

B. Evaluación de Arquitecturas

Como se ha podido observar, Flink-CEP proporciona una serie de opciones para poder crear diversos tipos de eventos de manera muy sencilla. Pero uno de los aspectos a evaluar es conocer si realmente esta diversidad de eventos tiene una mejora en las capacidades computacionales del servidor que las analiza.

En este sentido, para este experimento, se evalúa una arquitectura tradicional, POST, con la implementada para SAVIA, es decir, CEP. Con la finalidad de evaluar el rendimiento entre estas dos arquitecturas, se ha escogido como caso de estudio de CEP "Eventos In-situ". Por tanto, el objetivo para ambas arquitecturas, CEP y POST, es igual, es decir, evaluar la distancia en tiempo real de SAVIAtt con SAVIApp y, de romperse la distancia de alejamiento, enviar una alerta al SAVIApol más cercano. A continuación, se especifica la implementación de cada sistema:

- CEP: Los datos de los usuarios al llegar al core level se abre dos hilos de comunicación, uno para almacenar en la BBDD (Base de Datos) y el otro para analizarlos con CEP. Si el evento es generado, este se envía al Broker Global y, posteriormente, a los usuarios suscritos.
- POST: Los datos de los usuario al llegar al cloud son almacenados; posteriormente extraídos para analizarlos y, en caso de que la alerta sea generada, enviarle a los usuarios directamente.

Destacar que, para este experimento, solamente se ha cambiado el mecanismo de generación de alertas, toda la arquitectura y aplicaciones que subyacen del ecosistema SAVIA son exactamente iguales.

C. Benchmark: Ahorro de Energía

Con el fin de poder evaluar el rendimiento, se ha realizado una prueba de referencia con el paquete de software PERF en una computadora equipada con 8GB de RAM y un procesador Intel i7 3.60GHz x8. Específicamente, lo que se evalúa con estas pruebas es el consumo de energía empleado a la hora de la generación de las alertas.

En el trascurso de esta prueba, se ha llevado a cabo un test de estrés tomando como caso de estudio la generación de 30, 60, 90, 120 y 150 alertas durante un minuto de manera equiespaciada. Por ejemplo, primeramente se ha tomado el caso de 30 alertas, de manera que cada minuto se han establecido estas alertas durante un tiempo total de 15 minutos; obteniendo, finalmente, una generación 30 alertas repetidas 15 veces. Una vez finalizada la prueba con 30 alertas, se reinician todos los servicios y realiza el mismo procedimiento para los demás casos, aunque cambiando el número de alertas a 60, 90, 120 y 150, respectivamente. Esta generación se ha desarrollado de manera que, para cada uno de los casos se ha obtenido el consumo de energía y así evaluar qué sistema tiene un mayor consumo de recursos, ver Figura 8.

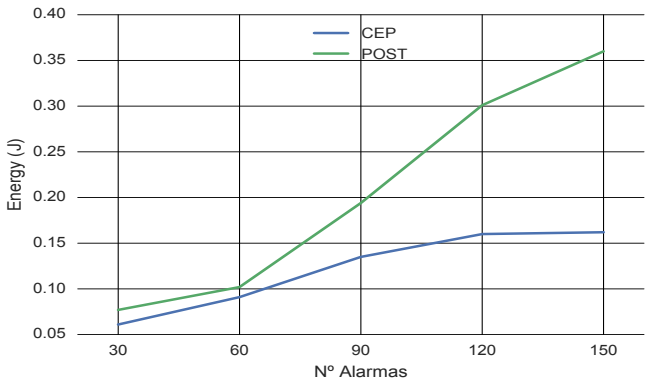

Fig. 8. Consumo de energía para los dos sistemas de generación de alertas para CEP (en azul) y POST (en rojo).

En la anterior Figura, se puede observar como el consumo de energía para el sistema tipo POST, es más elevado que CEP en dos sentidos principales:

1) En el menor número de alertas, 30 en este estudio, puede verse que el motor CEP tiene un menor consumo de energía que POST. Este patrón se observa para los otros casos de alertas.
2) El consumo de energía en CEP mantiene una dinámica de proporcionalidad respecto número de alertas, sin afectarlo apenas. Sin embargo, con POST podemos observar un consumo de energía exponencial, el cual, conforme aumentan el número de alertas aumenta notoriamente el consumo de energía.

VI. Conclusiones y Trabajo Futuro

En este trabajo se muestra el desarrollo de una familia de aplicaciones, denominada SAVIA, orientadas a tratar el problema de la violencia de género desde un punto de vista exclusivamente tecnológico. Las aplicaciones del sistema SAVIA (es decir, SAVIApp, SAVIAtt, SAVIApol y SAVIAs) representan un esfuerzo colectivo para abordar el problema de la violencia contra las mujeres. SAVIA demuestra las siguientes ventajas con respecto a otras soluciones:

- Eficiente en el uso de los recursos computacionales tanto por el uso combinado de fog computing y CEP, como el balanceo entre GPS y triangulación. Hasta donde los autores tienen conocimiento, no hay ninguna aplicación que incorpore estas funcionalidades.
- Confiable ya que se basa en protocolos seguros y siguiendo estándares internacionales.
- Usabilidad en su diseño e implementación, pensado para personas que no tienen grandes conocimientos sobre TICs.
- Bajo costo basándose en software libre y gratuito pero, además, potente. El sistema SAVIA al basarse en software

de código abierto, permite extender sus funcionalidades a través de la contribución de la comunidad para su aplicación en otras áreas relacionadas con la seguridad ciudadana, característica que claramente otras soluciones no ofrecen.

Sin embargo, SAVIA podría ser más inteligente, es decir, las reglas utilizadas para generar alarmas podrían mejorarse para crear un sistema más cognitivo. Un sistema más cognitivo podría aprender, crear o modificar reglas de acuerdo con los datos dinámicos obtenidos por los sistemas de monitoreo. Esta tarea se deja como trabajo futuro.

AGRADECIMIENTOS

This work has been funded by the "Programa Nacional de Innovación para la Competitividad y Productividad, Innóvate - Perú" of the Peruvian government, under grant number 363-PNICP-PIAP-2014.

REFERENCIAS

[1] Boletín Estadístico Mensual. Editado por Delegación del Gobierno. Recopilatorio de conocimiento sobre violencia de género. http://observatorioviolencia.org/estadisticas, April 2016. [Online; accedido 05-Nov-2017].

[2] Boletín Estadístico. Programa nacional contra la violencia familiar y sexual. http://www.mimp.gob.pe/files/programas_nacionales/pncvfs/estadistica/boletin_diciembre_2015/BV_Diciembre_2015.pdf, December 2015. [Online; accedido 05-Nov-2017].

[3] Geraldo Pereira Rocha Filho, Leandro Yukio Mano, Alan Demetrius Baria Valejo, Leandro Aparecido Villas, and Jo Ueyama. A low-cost smart home automation to enhance decision-making based on fog computing and computational intelligence. *IEEE Latin America Transactions*, 16(1):186–191, 2018.

[4] X. Masip-Bruin, E. Marín-Tordera, G. Tashakor, A. Jukan, and G. J. Ren. Foggy clouds and cloudy fogs: a real need for coordinated management of fog-to-cloud computing systems. *IEEE Wireless Communications*, 23(5):120–128, October 2016.

[5] M. B. A. P. Madumal, D. A. S. Atukorale, and T. M. H. A. Usoof. Adaptive event tree-based hybrid cep computational model for fog computing architecture. In *2016 Sixteenth International Conference on Advances in ICT for Emerging Regions (ICTer)*, pages 5–12, Sept 2016.

[6] Alfonso Garcia de Prado, Guadalupe Ortiz, and Juan Boubeta-Puig. Collect: {COLLaborativE} context-aware service oriented architecture for intelligent decision-making in the internet of things. *Expert Systems with Applications*, 85:231 – 248, 2017.

[7] Aurora: Domestic and family violence app. https://www.women.nsw.gov.au/violence_prevention/domestic_and_family_violence_app. [Online; accedido 05-Oct-2017].

[8] Libres app. http://www.violenciagenero.msssi.gob.es/informacionUtil/recursos/appLibres/home.htm. [Online; accedido 05-Nov-2017].

[9] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash. Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys Tutorials*, 17(4):2347–2376, Fourthquarter 2015.

[10] Manuel Castillo-Cara, Edgar Huaranga-Junco, Milner Quispe-Montesinos, Luis Orozco-Barbosa, and Enrique Arias Antúnez. Frog: A robust and green wireless sensor node for fog computing platforms. *Journal of Sensors*, 2018, 2018.

[11] Manuel Suárez-Albela, Tiago M. Fernández-Caramés, Paula Fraga-Lamas, and Luis Castedo. A practical evaluation of a high-security energy-efficient gateway for iot fog computing applications. *Sensors*, 17(9), 2017.

[12] Rodrigo Roman, Javier Lopez, and Masahiro Mambo. Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges. *Future Generation Computer Systems*, 78:680–698, 2018.

[13] Shanhe Yi, Cheng Li, and Qun Li. A survey of fog computing: concepts, applications and issues. In *Proceedings of the 2015 Workshop on Mobile Big Data*, pages 37–42. ACM, 2015.

[14] Charith Perera, Yongrui Qin, Julio C. Estrella, Stephan Reiff-Marganiec, and Athanasios V. Vasilakos. Fog computing for sustainable smart cities: A survey. *ACM Computing Surveys*, 50(3):32:1–32:43, June 2017.

[15] Nader Mohamed, Jameela Al-Jaroodi, Imad Jawhar, Sanja Lazarova-Molnar, and Sara Mahmoud. Smartcityware: A service-oriented middleware for cloud and fog enabled smart city services. *Ieee Access*, 5:17576–17588, 2017.

[16] Manuel Castillo-Cara, Jesús Lovón-Melgarejo, Gusseppe Bravo-Rocca, Luis Orozco-Barbosa, and Ismael García-Varea. An analysis of multiple criteria and setups for bluetooth smartphone-based indoor localization mechanism. *Journal of Sensors*, 2017, 2017.

[17] M. Mazhar Rathore, Awais Ahmad, Anand Paul, and Seungmin Rho. Urban planning and building smart cities based on the internet of things using big data analytics. *Computer Networks*, 101:63 – 80, 2016. Industrial Technologies and Applications for the Internet of Things.

[18] Fatemeh Jalali, Kerry Hinton, Robert Ayre, Tansu Alpcan, and Rodney S Tucker. Fog computing may help to save energy in cloud computing. *IEEE Journal on Selected Areas in Communications*, 34(5):1728–1739, 2016.

[19] International Organization for Standardization. Iso/iec 27018:2014 - information technology – security techniques – code of practice for protection of personally identifiable information (pii) in public clouds acting as pii processors. https://www.iso.org/standard/61498.html, January 2016. [Online; accedido 12-Jan-2019].

[20] B. Vanelli, M. Pereira da Silva, G. Manerichi, A. Sandro Roschildt Pinto, M. Antonio Ribeiro Dantas, M. Ferrandin, and A. Boava. Internet of things data storage infrastructure in the cloud using nosql databases. *IEEE Latin America Transactions*, 15(4):737–743, April 2017.

[21] Android Developers. Location and sensors apis. https:

//developer.android.com/guide/topics/sensors/index.html. [Online; accedido 10-Nov-2017].

Manuel Castillo-Cara received the PhD degree from the University of Castilla-La Mancha (UCLM) in July 2018. He has been working on university educational issues at the Computer Science as an Associate Professor and head of Intelligent Ubiquitous Technologies – Smart City (IUT-SCi) Lab at Universidad Nacional de Ingeniería. His current research is focused on Intelligent Ubiquitous Technologies, especially on in Wireless Sensor Networks, Distributed Computing, Pattern Recognition and Artificial Intelligence.

Giovanny Mondragón-Ruíz received the B.Sc. degree in Computer Science from the Universidad Nacional de Ingenieria (UNI) from Lima - Perú in 2018. He is currently a member of Intelligent Ubiquitous Technologies – Smart City (IUT-Sci) Lab at the Universidad Nacional de Ingeniería. He is researching Edge Level Architectures and Fog Computing optimizations. His research interests include Complex Event Processing, Mobile Computing and Internet of things.

Edgar Huaranga-Junco received de B.Sc. degree in Computer Science from the Universidad Nacional de Ingenieria (UNI), Lima, Perú, in 2018. He is a member of Intelligent Ubiquitous Technologies – Smart City (IUT-SCi) Lab at the Universidad Nacional de Ingenieria. His research interests includes Mobile Computing, Wireless Sensor Networks and Internet of Things technologies.

Enrique Arias Antúnez received the PhD Degree from the Technical University of Valencia, Spain, in 2003. He is Associate Professor at the School of Computer Science and Engineering (Department of Computing Systems) at the University of Castilla-La Mancha (UCLM) in Albacete since 1999. His topics of interest include Parallel Processing applied to different areas in science and engineering. In this field, he has published more tan 100 papers in relevant journals and conferences.

Luis Orozco-Barbosa received the Doctorat de l'Université from the Université e Pierre et Marie Curie, France, in 1987. From 1987 to 2001, he was a faculty member at the Electrical and Computer Engineering Department of the University of Ottawa, Canada. In 2002, he joined the Department of Computer Engineering at the Universidad de Castilla-La Mancha (Spain). His current research interests include Internet protocols, wireless sensor communications, and IoT technologies. He is a member of the IEEE.